INHALT

Dieses Handbuch gliedert sich in zwei Teile. Im ersten Teil erhalten Sie einen Überblick über das Erstellen von ArCon Makros. Die Zusammenhänge und Prinzipien werden erläutert. Dieser Teil beantwortet Fragen der Form “ wie kann ich ...” .

Der zweite Teil ist als Referenz zum Nachschlagen gestaltet. Hier finden Sie systematisch geordnet die Erläuterung aller Eigenschaften, Methoden und Ereignisse der ArCon Makro-Objekte. In diesem Teil werden Fragen der Form “ was ist ...” oder “ welche Parameter hat ...” beantworet.

EINFÜHRUNG IN DIE ARCON MAKRO PROGRAMMIERUNG

Ab der Version 3.0 ist ArCon programmierbar. Um Sie als Programmierer dabei nicht auf eine bestimmte Programmiersprache- und Entwicklungsumgebung festzulegen, benutzt ArCon eine OLE-Automatisierungsschnittstelle. Sie können diese Schnittstelle von allen modernen Softwareentwicklungsumgebungen aus bedienen (z.B.: Visual Basic, Visual C++, Delphi, Visual J++, PowerBuilder), wobei der Aufwand, abhängig vom Konzept der Entwicklungsumgebung bei der Einbindung von OLE-Automatisierungsobjekten, unterschiedlich ist. Sie finden im folgenden Kurzanleitungen zum Einsatz der ArCon Makro Schnittstelle bei der Programmierung mit Delphi 2. 0, Visual C++ 4.2 und 5.0 sowie Visual Basic. In der Regel sind Beispiele in dieser Hilfedatei in Visual Basic geschrieben. Es sollte Ihnen aber nicht schwer fallen, die Details auf andere Programmiersprachen zu übertragen. Sollten bei einem Thema grundlegende Unterschiede bestehen, werden diese besonders erläutert.

Bevor Sie mit der Erstellung eines einfachen Makros beginnen können, sollten Sie einen Überblick über die Zusammenhänge der beteiligten Komponenten haben:

Bei der ArCon Makro-Programmierung haben Sie es mit zwei getrennten Programmen (aus Betriebssystemsicht: vor einander geschützten Prozessen) zu tun. Auf der einen Seite (hier links dargestellt) ArCon als eigenständige Anwendung. Demgegenüber steht Ihre Applikation, hier mit einem eigenen Applikationsfenster dargestellt. Wenn Sie die Benutzeroberfläche Ihrer Applikation komplett in ArCon integrieren (was durchaus möglich ist), kommt Ihre Applikation ohne eigenes Fenster aus. Der Einfachheit halber kann stattdessen ein unsichtbares Applikationsfenster verwendet werden.

Damit Ihre Applikation ArCon steuern kann, muß eine Verbindung hergestellt werden. Das dafür nötige Bindeglied ist ein sogenanntes Active-X Control (ArCon.OCX), im Bild als rotes Quadrat mit einem X dargestellt. Dieses Active-X Control wird Bestandteil Ihrer Applikation und baut selbstständig eine Verbindung zu ArCon auf und übernimmt die komplette Kommunikation. Aus der Sicht Ihrer Anwendung verhält sich das Active-X Control wie ArCon selbst.

Es ist durchaus möglich, daß gleichzeitig mehrere Makros mit einem ArCon kommunizieren. Sie brauchen darauf bei der Makro-Programmierung in der Regel keine Rücksicht nehmen. In Ausnahmefällen (z.B. wenn ein AVA-System einen Schnappschuß des aktuellen Projektes erstellt um damit weitere Daten zu berechnen) kann Ihr Makro konkurrierende Zugriffe zeitweise verbieten.

Einbinden des ArCon-Active-X Controls

Je nach Entwicklungsumgebung sind verschiedene Schritte nötig, um einem Programm die Steuerung von ArCon zu ermöglichen. Im folgenden finden Sie detaillierte Anleitungen für häufig verwendete Entwicklungsumgebungen. Das beispielhaft erstellte Programm dient nur der Verdeutlichung der ersten Schritte und zur Demonstration der Unterschiede in der Integration der OLE-Automatisierung in verschiedenen Entwicklungsumgebungen. Das erstellte Programm erzeugt einen ungefähr 4 m x 4 m großen Raum mit dem Zentrum am Ursprung des Weltkoordinatensystems. ArCon setzt diesen Punkt beim Erzeugen eines neuen Projektes in die Mitte des Blattes. Wenn Sie den Ursprung des Koordinatensystems in die Mitte des Blattes setzen, nachdem Sie ein neues Projekt erstellt haben, stimmen die Weltkoordinaten mit den in der Statuszeile angezeigten Koordinaten überein.

Unabhängig von der Programmiersprache, in der es implementiert ist, geht das Beispielprogramm folgendermaßen vor:

1. Starten von ArCon bzw. Herstellen der Verbindung
1. Beim Aktivieren der entsprechenden Schaltfläche die folgenden Schritte ausführen:
1. Prüfen, ob ArCon ein Projekt geladen hat und gegebenenfalls ein neues Projekt erzeugen
1. Das aktuelle Stockwerk ermitteln
1. Vier mal eine neue Wand erzeugen und dem aktuellen Stockwerk befehlen, sie entsprechend zu plazieren.

Beim Schritt (1) ist auf Fehlerbehandlung zu achten - es könnte immerhin sein, daß das Beispielprogramm auf einem Computer gestartet wird, auf dem kein ArCon installiert ist. In diesem Fall zeigt das Beispielprogramm eine Meldung an und beendet sich umgehend. Beim Verbindungsaufbau bzw. ArCon-Start mit Hilfe der Methode StartMe benötigt ArCon drei Parameter, die nicht selbsterklärend sind: der erste ist eine Fensterkennzahl (technisch: ein Window Handle, Datentyp HWND), die das Hauptfenster Ihrer Applikation identifiziert. Falls es in Ihrer Entwicklungsumgebung schwierig oder unmöglich ist, eine solche Kennzahl zu ermitteln (z. B. in Java), können Sie hier eine 0 angeben. Der zweite Parameter ist ein Abbildungsmaßstab, der das interne Weltkoordinatensystem festlegt, der Wert " 1" definiert 1 m als Grundeinheit. Der dritte Parameter gibt den Namen einer Windows-Hilfe-Datei (*.HLP) an, in der Ihr Makro dokumentiert ist. Das Beispielprogramm ist nicht dokumentiert und benutzt deshalb einen leeren Namen.

Ein weiterer auffälliger Punkt ist der Parameter der in Schritt (3) verwendeten Methode CreateProject. Sie können hier einen Nullzeiger angeben (" NULL" in C/C++, " Nothing" in Visual Basic), wenn das zu erzeugende Projekt mit Standardwerten initialisiert werden soll. Delphi 2.0 meldet dabei allerdings einen Typfehler (" NIL" ist nicht kompatibel zu " VARIANT"), weshalb das Delphi-Beispiel ein Projekt mit selbstdefinierten Werten erzeugt.

Falls beim Starten des Beispielprogrammes kein ArCon auf Ihrem Computer läuft, versucht das Makro es zu starten. Damit dies funktionieren kann, muß entweder die Eigenschaften ExeFilePath den Dateinamen der Programmdatei ARCON.EXE als vollständigen Pfad enthalten, oder die Programmdatei muß über den Suchpfad gefunden werden können. Lief beim Start des Makros bereits ein ArCon, verbindet sich das Makro mit diesem. Sollten zu diesem Zeitpunkt bereits mehrere ArCon' s laufen, wird die Verbindung zu einem beliebigen aufgebaut, Sie haben keinen Einfluß auf die Auswahl. Es ist aber sichergestellt, daß ein ArCon, das von einem Makro gestartet wird, mit diesem Makro kommuniziert. Andersherum ist ebenfalls garantiert, daß ein von ArCon gestartetes Makro (aus dem Makro-Menü) die Verbindung zum startenden ArCon aufnehmen wird, selbst wenn weitere ArCon' s aktiv sind.

Visual Basic

Um das Beispielprojekt mit Visual Basic (4.0 oder 5.0) zu implementieren, gehen Sie folgendermaßen vor:

1. Erstellen Sie ein neues Projekt.

1. Wählen Sie Projekt/Komponenten und wählen Sie den Reiter “ Steuerelemente” aus, falls ein anderer Reiter aktiv ist. Kreuzen Sie den Eintrag “ ArCon+ Visuelle Architektur” an und schließen Sie den Dialog mit “ OK” . Visual Basic lädt nun das ArCon Active-X Control und zeigt dessen Symbol in der Werkzeugsammlung an.

1. Aktivieren Sie dieses Symbol und erzeugen damit im Form ein Steuerelement. Tragen Sie im Eigenschaftsfenster als Name dieses Elementes “ ArCon” ein.

1. Klicken Sie doppelt auf den Form-Hintergrund und tragen sie folgenden Code in der darufhin erstellten Methode Form_Load ein:

```
Private Sub Form_Load()
    If Not ArCon.StartMe(hWnd, 1, "") Then
   MsgBox "Kann ArCon nicht starten!"
   Unload Me
    End If
End Sub
```

Dadurch wird beim Öffnen des Forms eine Verbindung zu ArCon hergestellt und Ihr Programm kann mit der ArCon Steuerung beginnen.

Es spielt keine Rolle, wo Sie auf Ihrem Form das ArCon Steuerelement plazieren, zur Laufzeit ist es unsichtbar.

Lassen Sie Ihr Programm laufen. Wenn es die ArCon.EXE Datei im Suchpfad findet (oder Sie im Eigenschaftsfenster für das ArCon Steuerelement die Eigenschaft “ ExeFilePath” entsprechend gesetzt haben) wird Ihr Programm ArCon starten, falls noch kein ArCon läuft.

Darüberhinaus macht das Makro derzeit gar nichts, insbesondere erzeugt es noch keine Wände. Erweitern Sie es daher um eine Schaltfläche auf dem Form mit dem Namen “ Schachtel” und der Beschriftung ” Einfacher Grundriß” . Klicken Sie doppelt auf die Schaltfläche und tragen Sie den folgenden Code in die daraufhin automatisch erstellte Prozedur Schachtel_Click() ein.

```
Private Sub Schachtel_Click()
    Dim Stockwerk As ArCon.Story

    If ArCon.Mode = AC_NoMode Then
   ArCon.CreateProject Nothing
    End If

    Set Stockwerk = ArCon.CurrentStory
    Stockwerk.PlaceWall ArCon.NewWall(0), -2, -2, 2, -2
    Stockwerk.PlaceWall ArCon.NewWall(0), 2, -2, 2, 2
    Stockwerk.PlaceWall ArCon.NewWall(0), 2, 2, -2, 2
    Stockwerk.PlaceWall ArCon.NewWall(0), -2, 2, -2, -2
End Sub
```

Diese Routine prüft zunächst, ob ArCon bereits ein Projekt geladen hat (dies kann der Fall sein, wenn beim Starten des Makros schon ein ArCon läuft). Falls nicht, wird ein Standard-Projekt ohne besondere Einstellungen (Parameter “ Nothing” bei CreateProject) erzeugt. Um in diesem Projekt Wände erzeugen zu können, wird ein Stockwerk benötigt, das die Wände aufnimmt. Dazu dient die Variable Stockwerk, die als “ Story” definiert ist.

“ Story” ist dabei eine Klasse, die aus dem ArCon Active-X Control exportiert wird und von Visual Basic nach hinzufügen des Active-X Controls genau wie eine der vordefinierten Klassen behandelt wird. Beim Eingeben des Codes werden Sie bemerken, daß Visual Basic bereits die nötigen Parameter aller Funktionen kennt und entsprechende Tips anzeigt. Sie können auch Hilfe zu einzelnen Befehlen erhalten, indem Sie die Schreibmarke auf dem Befehl positionieren (z.B. PlaceWall) und F1 drücken. Visual Basic zeigt daraufhin den Eintrag zu Story.PlaceWall in der ArCon Makro-Hilfe an.

Im Beispiel wird der Variablen Stockwerk mit Hilfe der “ Set” Anweisung ein Stockwerk zugeordnet. Aus technischer Sicht handelt es sich hier nicht um eine einfache Zuweisung (wie etwa “ summe = 0”), sondern um eine Referenz mit Referenzierungszähler. Stockwerk referenziert nach dieser Anweisung ein Story-Objekt innerhalb des ArCon Active-X Controls und

hält dieses Objekt “ am Leben” . Erst wenn Stockwerk nicht mehr auf das Active-X Objekt zeigt, kann dieses aufhören zu existieren. Dies passiert automatisch, wenn die Variable Stockwerk aufgelöst wird (als lokale Variable am Ende der Prozedur), oder Sie die Zuordnung explizit mit der Anweisung “ Set Stockwerk = Nothing” aufgeben. Achten Sie darauf, daß Sie solche Zuordnungen in globalen Variablen aufgeben, bevor Sie das Form mit dem Active-X Control schließen - ansonsten kann es beim Beenden Ihres Programmes zu Problemen kommen (das Programm stürzt ab oder läßt sich nicht mehr beenden).

Wenn Ihr Programm kein eigenes Applikationsfenster anzeigen soll, stellen Sie die Eigenschaften “ Visible” und “ ShownInTaskBar” des Forms auf “ False” .

Delphi

Um das Beispielprojekt mit Delphi 2.0 zu implementieren, gehen Sie folgendermaßen vor:

1. Erstellen Sie ein neues Projekt mit dem Befehl Datei/Neue Anwendung.

1. Erzeugen Sie ein ArCon-Unit indem Sie den Befehl Komponente/Installieren wählen und im Dialog die Schaltfläche OCX anklicken. Wählen Sie aus der erscheinenden Liste " ArCon Visuelle Architektur" . Delphi importiert daraufhin das Active-X Control und fügt es der System-Bibliothek hinzu. Dieser Schritt ist nur bei der ersten Verwendung eines neuen Active-X Controls nötig.

1. Aktivieren Sie in der Palette Ihrer Form "OCX" . Ordnen Sie nun das ArCon-Steuerelement in Ihrer Form an und benennen Sie es im Eigenschaftsfenster " Arcon" benennen.

1. Fügen Sie eine Behandlungsroutine für das Form-Ereignis " OnCreate" ein und geben Sie dort folgenden Code ein:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  if (not ArCon.StartMe(Handle, 1.0, '')) then
  begin
    Application.MessageBox('Kann ArCon nicht starten!', 'Fataler Fehler', mb_OK)
;
    PostQuitMessage(1);
  end;
end;
```

Wie schon im Visual Basic Beispiel wird hier versucht, eine Verbindung zu ArCon aufzubauen und, falls dies nicht gelingt, das Programm beendet. Anders als Visual Basic " kennt" Delphi nicht automatisch alle Objekte des Active-X Controls, sondern importiert sie. Dazu erzeugt Delphi Pascal Quelltexte und fügt in Ihrem Projekt automatisch Verweise darauf ein. Am Anfang Ihres Quelltextes finden Sie nach dem Einsetzen des Active-X Controls in Ihr Form folgende Zeilen:

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, OleCtrls, ARCON;
```

Beim Unit " ARCON" handelt es sich um den Quelltext, der beim Import des Active-X Controls erzeugt wurde. Leider werden Konstante des Active-X Controls nicht importiert. Sie finden im ArCon-Entwicklerpaket eine Datei " acGlobal.pas" , die die fehlenden Konstanten enthält. Fügen Sie diese Datei Ihrem Projekt hinzu und ergänzen Sie die " uses" Anweisung entsprechend.

Fügen Sie in Ihr Form eine Schaltfläche mit der Beschriftung " Einfacher Grundriß" und dem Namen " Schachtel" ein. Erzeugen Sie eine Behandlungsprozedur für deren " OnClick" Ereignis und fügen Sie dort folgenden Code ein:

```
procedure TForm1.SchachtelClick(Sender: TObject);
var prj, story : VARIANT;
begin
  if (Arcon.Mode = AC_NoMode) then
  begin
    prj := Arcon.NewProject;
    prj.PaperSize := ACPF_A4quer;
    prj.Scale := 30;
    prj.Unit := ACME_Meter;
    Arcon.CreateProject(prj);
  end;
  story := Arcon.CurrentStory;
  story.PlaceWall(Arcon.NewWall(0), -2.1, -2.1, +2.1, -2.1);
  story.PlaceWall(Arcon.NewWall(0), +2.1, -2.1, +2.1, +2.1);
  story.PlaceWall(Arcon.NewWall(0), +2.1, +2.1, -2.1, +2.1);
  story.PlaceWall(Arcon.NewWall(0), -2.1, +2.1, -2.1, -2.1);
end;
```

Hinter Befehlen wie " story.PlaceWall" steckt eine Menge Magie - schließlich handelt es sich bei " story" um eine untypisierte Variable (deklariert als VARIANT), die abhängig vom OLE-Objekt, auf das sie zeigt, unterschiedliche Eigenschaften hat und andere Methoden kennt. Die Ähnlichkeit zu Visual Basic ist sicherlich gewollt. Leider verzichtet hier die ansonsten typsichere Sprache Pascal auf die Typstrenge, die in Visual Basic zumindest optional erreicht werden kann (z.B. durch die Deklaration eines Stockwerkes als " ArCon.Story" statt als " Object").

Visual C++

Um das Beispielprojekt mit Visual C++ Version 4.2 zu implementieren, gehen Sie folgendermaßen vor:

1. Erstellen Sie ein neues Projekt mit dem Application Wizard: File/New/Project Workspace/ MFC App Wizard (EXE)
1. Wählen Sie als Anwendungstyp " Dialog based" .
1. Im Schritt 2 des Wizards aktivieren Sie OLE controls und beenden Sie den Wizard
1. Wählen Sie im Project-View den Resource-Reiter an.
1. Öffnen Sie den Hauptdialog der neuen Anwendung und entfernen Sie die Schrift " TODO: place dialog controls here" .
1. Klicken Sie den Dialoghintergrund mit der rechten Maustaste an und wählen Sie " Insert OLE Control..." .
1. Wählen Sie aus der erscheinenden Liste " ArCon Script Control" aus.
1. Klicken Sie auf das erscheinende ArCon Symbol doppelt und geben Sie ihm die ID IDC_ARCON.
1. Aktivieren Sie nun den Class Wizard und wählen Sie den Reiter " Member Variables"
1. Wählen Sie in der Liste der verfügbaren ID' s IDC_ARCON aus und klicken Sie auf " Add Variable..." . Der Class Wizard bietet daraufhin an, die Klassen des Active-X Controls zu importieren. Bestätigen Sie dies mit " OK" , ebenso wie die folgende Liste der zu importierenden Klassen und ihrer Import-Namen.
1. Geben Sie der neuen Variablen für das ArCon Active-X Control den Namen " m_arcon" und beenden Sie den Wizard mit " OK" .
1. Klicken Sie im Project View auf den Klassen-Reiter und erschrecken Sie nicht: Ihr Projekt enthält jetzt mehr als 50 Klassen (und über 100 Dateien!). Wählen Sie Ihre Dialogklasse und dort die " OnInitDialog" Methode. Fügen Sie folgenden Code ein:

```
// TODO: Add extra initialization here
if (!m_arcon.StartMe((long)m_hWnd, 1.0f, AfxGetApp()->m_pszHelpFilePath)) {
  AfxMessageBox("Kann ArCon nicht starten!");
  PostQuitMessage(1);
}
```

Damit ist das Grundgerüst des Beispiels erstellt.

Plazieren Sie eine Schaltfläche mit der Beschriftung " Einfacher Grundriß" und der ID " IDC_SCHACHTEL" . Aktivieren Sie den Class Wizard und ordnen Sie IDC_SCHACHTEL im Reiter " Message Maps" eine neue Funktion zu. Geben Sie dort folgenden Code ein:

```
void CTestDlg::OnSchachtel()
{
  // TODO: Add your control notification handler code here
  if (m_arcon.GetMode() == AC_NoMode)
    m_arcon.CreateProject(NULL);

  CStory stockwerk = m_arcon.GetCurrentStory();
  stockwerk.PlaceWall(m_arcon.NewWall(0).m_lpDispatch, -2.0f, -2.0f, 2.0f, -2.
0f);
  stockwerk.PlaceWall(m_arcon.NewWall(0).m_lpDispatch, 2.0f, -2.0f, 2.0f, 2.0f);
  stockwerk.PlaceWall(m_arcon.NewWall(0).m_lpDispatch, 2.0f, 2.0f, -2.0f, 2.0f);
  stockwerk.PlaceWall(m_arcon.NewWall(0).m_lpDispatch, -2.0f, 2.0f, -2.0f, -2.
0f);
}
```

Genau wie Delphi importiert der Class Wizard keine Konstanten aus dem Active-X Control, weshalb Sie auf die mitgelieferte Datei " acGlobal.h" zurückgreifen müssen. Außerdem fügt Class Wizard (anders als Delphi) nicht automatisch #include Anweisungen in Ihrer Datei ein, wenn Sie importierte Typen benutzen. Ergänzen Sie daher am Anfang Ihrer Datei hinter den bereits vorhandenen #include Anweisungen folgende Zeilen:

```
#include "acGlobal.h"
#include "Story.h"
#include "Wall.h"
```

Dies setzt voraus, daß Sie die Datei “ acGlobal.h” in Ihr Projektverzeichnis kopiert haben. Andernfalls können Sie auch einen relativen Pfad zu dieser Datei angeben oder unter Build/ Settings entsprechende Include-Pfade konfigurieren.

Falls Sie Visual C++ 5.0 verwenden, können Sie die “ PlaceWall” Aufrufe vereinfachen:

```
stockwerk.PlaceWall(m_arcon.NewWall(0), -2.0f, -2.0f, 2.0f, -2.0f);
```

Der Zugriff auf das public Attribut “ m_lpDispatch” eines Automatisierungsobjektes ist in dieser Version nicht mehr nötig, da ein entsprechender cast-Operator vorhanden ist und vom Compiler hier automatisch eingesetzt wird.

Soll Ihre Applikation kein eigenes Fenster haben, genügt es nicht, das “ Visible” Flag in der Resource Ihres Dialoges zu entfernen. Modale Dialoge (dazu gehört der Hauptdialog einer “ Dialog based” MFC-Applikation) sind immer sichtbar. Ändern Sie die vom App Wizard generierte Anwendung, sodaß sie einen nicht-modalen Dialog benutzt.

GRUNDLEGENDE PROGRAMMIERHINWEISE

Dieses Kapitel gibt Ihnen einen Überblick über wichtige Philosophien und Hintergründe der ArCon Programmierschnittstelle. Die Kenntnis dieser Grundlagen erleichtert Ihnen den Umgang mit konkreten Befehlen und die Suche nach bestimmten Eigenschaften/Verfahren.

Erzeugen konstruktiver Elemente

Alle programmierbaren Elemente des ArCon Konstruktionsmoduses haben ein entsprechendes OLE-Objekt. Der Wand entspricht zum Beispiel das Wall Objekt, dem Raum das Room Objekt und so weiter. Einige Objekte werden von ArCon implizit erzeugt (zum Beispiel der Raum), andere auf expliziten Befehl Ihres Markos (zum Beispiel die Wand). Die explizit erzeugten Objekte werden zweistufig generiert: zunächst erstellen Sie ein OLE-Objekt

```
Dim wand As ArCon.Wall
Set wand = ArCon.NewWall(0)
```

Die Methode NewWall erzeugt ein neues OLE Objekt vom Typ Wall. Der Parameter 0 gibt den Typ der Wand an, hier ist es der erste Wandtyp (Außenwand, 36 cm dick). Dieses Wandobjekt ist zunächst noch ungebunden, es entspricht keiner Wand in der ArCon Planung. Sie können das Objekt nun wie gewünscht manipulieren und schließlich in der ArCon Planung plazieren (und es damit sichtbar machen):

```
Dim stockwerk As ArCon.Story
Set stockwerk = ArCon.CurrentStory
stockwerk.PlaceWall wand, 1, 1, 3.4, 3.2
```

Die Methode PlaceWall gehört zum Objekt Stockwerk vom Typ Story. Ohne das aufnehmende Stockwerk können Sie keine Wände verlegen, genauso wie Sie ohne aufnehmende Wände keine Fenster plazieren können. Nach der Plazierung geht das Wand-Objekt in den gebundenen Zustand über, es entspricht jetzt der von Ihnen erzeugten Wand in der ArCon Planung. Jede Änderung, die Sie jetzt noch am Wandobjekt durchführen, wirkt sich sofort auf die ArCon Planung aus.

Die Grundregeln für explizit zu konstruierende Objekte lauten zusammengefaßt:

Ein Objekt vom Typ XYZ wird (in der Regel) mit der Methode ArCon.NewXYZ erzeugt. Falls es unterschiedliche Typen von XYZ gibt, wird die Typkennung hierbei als Parameter angegeben.

Ein neu erzeugtes Objekt ist zunächst ungebunden, es tritt (noch) nicht in der ArCon Planung auf. Änderungen an einem ungebundenen Objekt haben keine Konsequenzen für die ArCon Planung, da das ungebundene Objekt dort nicht sichtbar ist.

Ein ungebundenes Objekt wird durch eine entsprechende Methode PlaceXYZ des zu XYZ passenden aufnehmenden Objektes plaziert. Anschließend ist das Objekt gebunden und tritt damit in der Planung auf.

Änderungen an einem gebundenen Objekt wirken sich sofort auf die ArCon Planung aus.

Im Gegensatz dazu stehen implizit erzeugte Objekte. Sie können mit Hilfe einer Find-Methode von ArCon ermittelt werden oder von Ihrem Makroprogramm über die Objekthierachie gefunden werden.

```
Dim raum As ArCon.Room
Set raum = stockwerk.FindRoom(2, 2)
```

Hier ermittelt das Stockwerk den Raum an den Koordinaten 2/2.

Implizit erzeugte Objekte sind immer gebunden.

Ereignisverarbeitung

OLE-Automatisierung funktioniert in zwei Richtungen: zum einen gibt Ihr Makro Befehle an ArCon, indem es Methoden aufruft. Hierbei erhalten Sie in der Regel eine Bestätigung über die erfolgreiche Ausführung bzw. das Fehlschlagen der Aktion, oder einen anderen Wert als Rückgabewert der Methode. Wesentliches Merkmal dabei ist der Zeitpunkt, der einzig von Ihrem Makro bestimmt wird.

Ereignisse funktionieren genau umgekehrt: ArCon bestimmt den Zeitpunkt und ruft Ihre Ereignismethode selbstständig auf. Es gibt keine Möglichkeit, für Ihr Makro die “ Annahme” eines Ereignisses zu verweigern. Allerdings findet die Synchronisation mit der Windows Message Queue automatisch statt, Ihr Makro muß nicht multi-threading fähig sein.

Das ArCon Programmiermodell versucht Ihrem Makro die Illusion zu vermitteln, es gäbe auf dem lokalen Computer nur ein ArCon und nur ein Makro. Bei Ereignissen funktioniert dies nicht immer. ArCon muß die meisten Ereignisse an alle gerade laufenden Makros versenden. Ausnahme sind z.B. Button-Events, die nur an das Makro gehen, das die entsprechende Schaltfläche definiert hat.

Da die ArCon Benutzeroberfläche stillgelegt ist, bis ein Ereignis an alle Empfänger verteilt wurde, ist es besonders wichtig, daß Ihr Makro innerhalb von Ereignisroutinen möglichst keine aufwendigen Aktionen durchführt. Aus technischen Gründen sind einige ArCon Befehle innerhalb von Ereignismethoden nicht verfügbar. Falls Sie, angestoßen durch ein Ereignis, einmal umfangreichere Aktionen durchführen möchten, setzen Sie innerhalb der Ereignisroutine einen Timer, z.B. auf 1 ms. Innerhalb der Timer-Behandlung können Sie beliebig aufwendige Aktionen durchführen.

Listen

Listen treten an vielen Stellen in der ArCon Programmierung auf. Für jede Listeneigenschaft gibt es einen entsprechenden Typ mit dem Namen ...Collection. Zum Beispiel hat ein Stockwerk (Story-Objekt) eine Eigenschaft Walls, das vom Typ WallCollection ist. Diese Liste enthält alle Wände des Stockwerks.

Einige Listentypen haben besondere Fähigkeiten, die Sie bitte der Dokumentation zum konkreten Listentyp entnehmen. Allen Listen gemeinsam sind die Eigenschaft “ Count” vom Typ “ long” und die Methode “ Item” , die ein Objekt der Liste liefert.

Je nach Programmierumgebung können Sie solche Listen unterschiedlich verwenden. In Visual Basic ist es zum Beispiel möglich, folgendes zu schreiben:

```
Dim laenge As Double, area As Double
For Each wand In stockwerk.Walls
    ' Position der Wand ermitteln
    wand.GetPos x1, y1, x2, y2
    ' Länge daraus ausrechnen
    laenge = Sqr((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1))
    ' Fläche berechnen
    area = laenge * wand.Thickness
    Debug.Print "Länge="; laenge; " Dicke="; wand.Thickness; " Fläche="; area
Next
```

Diese Schleife gibt die ungefähre Fläche aller Wände im Stockwerk aus. In anderen Programmiersprachen geht es nicht ganz so elegant, dort müssen Sie auf die Count Eigenschaft und die Item Methode der Liste zurückgreifen. Das gleiche Beispiel wie oben sieht in Visual C++ so aus:

```
  CWall wand;
  CStory stockwerk = m_arcon.GetCurrentStory();
  CWallCollection liste = stockwerk.GetWalls();
  int i, num = liste.GetCount();
  for (i = 1; i <= num; i++) {
    wand = liste.Item(i);
    double x1, x2, y1, y2;
    wand.GetPos(&x1, &y1, &x2, &y2);
    double laenge = sqrt((x2-x1)*(x2-x1) + (y2-y1)*(y2-y1));
    double area = laenge * wand.GetThickness();
    TRACE("Länge = %.3g, Fläche = %.3g\n", laenge, area);
  }
```

Beachten Sie, daß OLE Listen im Gegensatz zu C/C++ Arrays immer mit dem Index 1 beginnen!

Achtung: natürlich lassen sich Grundflächen von Wänden nicht so einfach berechnen, wie das Beispiel es hier demonstriert. Durch Wandverschneidungen sind Wände nicht rechteckig, daher ist “ Achsenlänge * Dicke” nur eine grobe Näherung für ihre Fläche. Ein genaueres Verfahren zur Berechnung dieser Flächen finden Sie im mitgelieferten Beispielprogramm “ Wandflächen” .

Magische Zahlen

Es gibt zwei Arten von magischen Zahlen in der ArCon Programmierung: (variable) Kennzahlen (ID’ s) für interne Objekte, die ArCon auf Befehl Ihres Programmes erzeugt und (konstante) Typkodierungen oder ID’ s.

Alle Typkodierungen beginnen mit dem Typ 0. Der maximale Wert kann entweder mit einer Funktion von ArCon erfragt werden (siehe zum Beispiel AvailableWindows), oder ist (für eine konkrete ArCon Version) konstant. Im letzteren Fall existiert eine symbolische Konstante, die die Anzahl der verfügbaren Typen angibt (und damit auch den Maximalwert).

Bei Kennzahlen ist der Wert 0 immer ein ungültiger Wert. Kennzahlen sind nicht bei jedem Lauf Ihres Programmes gleich - abhängig von internen Details, geladenem Projekt, sonstigen laufenden Makros und vielen anderen unabwägbaren Einflüssen werden diese Zahlen schwanken.

OBJEKTHIERACHIE

Ein ArCon Projekt besteht aus vielen, zusammenhängenden Teilen. Dieses Kapitel gibt Ihnen einen Überblick über die Verbindungen der einzelnen Bestandteile untereinander und zeigt Ihnen Wege zum Auffinden bestimmter Einzelteile. Eine typische Frage ist z.B.: welche Fenster gibt es im Erdgeschoß?

ArCon verknüpft Bestandteile der Planung durch direkte Referenzen (ein Raum kennt das Stockwerk in dem er liegt) und durch Listen (eine Wand enthält eine Liste aller Fenster in dieser Wand). Die Hierachie ist dabei nicht eindeutig - mehrere Wege führen zu den gleichen Objekten, manche sogar mehrfach.

Die Hierachie beginnt mit dem ArCon Objekt selbst.

ArCon enthält:

Eigenschaft	Anzahl	Objekt(e)
Buildings	n	alle Gebäude der gesamten Planung
CurrentBuilding	1	das aktuelle Gebäude
CurrentStory	1	das aktuelle Stockwerk
Dimensions	n	alle Vermaßungen der gesamten Planung
GlobalTerrain	1	das Umgebungsgelände, das alle Grundstücke und Gebäude umschließt
Guides	n	alle Hilfslinien der gesamten Planung
Labelings	n	alle Beschriftungen der gesamten Planung
Terrains	n	alle Gelände der gesamten Planung

Building enthält:

Eigenschaft	Anzahl	Objekt(e)
Stories	n	alle Stockwerke des Gebäudes

Story enthält:

Eigenschaft	Anzahl	Objekt(e)
Building	1	das Gebäude, in dem das Stockwerk liegt
CeilingsOpenings	n	alle Deckenlöcher des Stockwerks
Ceilings	n	alle Deckenplatten des Stockwerks
Chimneys	n	alle Schornsteine des Stockwerks
Conturs	n	alle Konturen des Stockwerks
Roofs	n	alle Dächer des Stockwerks
Rooms	n	alle Räume des Stockwerks
Stairs	n	alle Treppenhäuser des Stockwerks
Supports	n	alle Stützen des Stockwerks
Walls	n	alle Wände des Stockwerks

Dimension enthält:

Eigenschaft	Anzahl	Objekt(e)
Story	1	das Stockwerk, in dem sich die Vermaßung befindet

Terrain enthält:

Eigenschaft	Anzahl	Objekt(e)
Buildings	n	alle Gebäude auf diesem Gelände
Hedge	1	das Umrandungsobjekt des Geländes
Owner	1	das Gelände, dessen Bestandteil dieses Gelände ist (falls es nicht das globale Gelände ist)

Terrains	n	alle Untergelände dieses Geländes
Guide enthält:		
Eigenschaft	Anzahl	Objekt(e)
Story	1	das Stockwerk, in dem die Hilfslinie gezeichnet wird
Labeling enthält:		
Eigenschaft	Anzahl	Objekt(e)
Story	1	das Stockwerk, in dem sich die Beschriftung befindet
Chimney enthält:		
Eigenschaft	Anzahl	Objekt(e)
Story	1	das Stockwerk, in dem sich der Schornstein befindet
Contur enthält:		
Eigenschaft	Anzahl	Objekt(e)
Room	1	der Raum, dessen Umrandung diese Kontur beschreibt
WallSegments	n	alle Wandsegmente, die Bestandteil der Kontur sind
Roof enthält:		
Eigenschaft	Anzahl	Objekt(e)
Story	1	das Stockwerk, in dem sich das Dach befindet
Windows	n	alle Dachfenster in diesem Dach
Room enthält:		
Eigenschaft	Anzahl	Objekt(e)
Chimneys	n	alle Schornsteine in diesem Raum
Conturs	2	innere und äußere Kontur, die diesen Raum umschließen
Story	1	Stockwerk, in dem der Raum liegt
Supports	n	alle Stützen in diesem Raum
Staircase enthält:		
Eigenschaft	Anzahl	Objekt(e)
Story	1	das Stockwerk, in dem sich das Treppenhaus befindet
Support enthält:		
Eigenschaft	Anzahl	Objekt(e)
Story	1	das Stockwerk, in dem sich die Stütze befindet
Wall enthält:		
Eigenschaft	Anzahl	Objekt(e)
Doors	n	alle Türen in dieser Wand
Holes	n	alle Wandlöcher in dieser Wand
Story	1	das Stockwerk, in dem sich die Wand befindet
WallSegments	n	alle Wandsegmente der Wand
Windows	n	alle Fenster in dieser Wand
WallSegment enthält:		
Eigenschaft	Anzahl	Objekt(e)
Contur	1	die Kontur, zu der die Wandseite gehört
Doors	n	Türen in dieser Wandseite
Holes	n	Löcher in dieser Wandseite
LambdaHs	n	Höhenpunkte der Wandseite (z.B. Schrägen unter einem Dach)
Room	1	der an diese Wandseite grenzende Raum

Wall	1	die Wand, zu der die Wandseite gehört
Windows	n	alle Fenster in dieser Wandseite
Window enthält:		
Eigenschaft	Anzahl	Objekt(e)
LeftSegment	1	Innenwandseite, in der sich das Fenster befindet
RightSegment	1	Außenwandseite, in der sich das Fenster befindet
Wall	1	Wand, in der das Fenster sitzt
RoofWindow enthält:		
Eigenschaft	Anzahl	Objekt(e)
Roof	1	Dach, in dem sich das Fenster befindet
Door enthält:		
Eigenschaft	Anzahl	Objekt(e)
LeftSegment	1	linke Wandseite, in der sich die Tür befindet
RightSegment	1	rechte Wandseite, in der sich die Tür befindet
Wall	1	Wand, in der sich die Tür befindet

ELEMENTE DER ARCON BENUTZEROBERFLÄCHE

Neben eigenständigen Applikationen, die ArCon+ nur zur Datenerfassung oder Visualisierung nutzen, können Makros neue Funktionen für ArCon+ bereitstellen, oder bestehende Funktionalitäten erweitern. In diesen Fällen ist es sinnvoll (anders als bei eigenständigen Applikation mit eigenem Applikationsfenster), die ArCon+ Benutzeroberfläche für das Makro mitzunutzen. Dazu gibt es umfangreiche Möglichkeiten für verschiedene Elemente der ArCon+ Benutzeroberfläche. Die folgenden Abschnitte zeigen die Möglichkeiten und Verfahren anhand kleiner Beispiele.

Vorhandene Dialoge erweitern

Eigene Dialoge

Eigene Menüeinträge

Eigene Schaltflächen

Automatische Schaltflächen

Vorhandene Dialoge erweitern

Wenn Sie einen ArCon+ Dialog erweitern wollen, ermitteln Sie zunächst seine Kennzahl und die Kennungen aller Datenfelder im Dialog, von/zu denen Sie Daten er-/übermitteln wollen. Hierzu können Sie den Dialog-Explorer einsetzen.

Angenommen, Ihr Makro integriert ein automatisches Bestellsystem für Fenster in ArCon. Sie haben eine Datenbank, in der Sie Details wie Bestellnummern, Preise und Varianten für Fenster nachschlagen können. Sie möchten nun diese Daten auch im Eigenschaftsdialog für Fenster anzeigen. Dazu erstellen Sie einen eigenständigen Dialog, der diese Daten anzeigt.

Verändern Sie folgende Eigenschaften des Forms: BorderStyle = 0 - Kein, ControlBox = False, ShownInTaskBar = False. Damit ist der Dialog vorbereitet, um in den ArCon+ Dialog mit aufgenommen zu werden.

Sobald der Benutzer einen Dialog öffnet, löst ArCon das Ereignis " LoadDialog" aus. In der Ereignisprozedur entscheidet Ihr Makro zunächst, ob der Dialog erweitert werden soll. Das ist im Beispiel sehr einfach, da dieses genau einen Dialog erweitert. Der Code sieht folgendermaßen aus:

```
Private Sub ArCon_LoadDialog(ByVal dialogID As Long, ByVal ArConDlgToken As Long,
ByVal dialogObject As Object)
    If dialogID = ACDLGID_TuerEinstellungen Then
   Load Katalog
   ArCon.AttachDialog ArConDlgToken, Katalog.hWnd, Katalog.Icon, 0, False, False
    End If
End Sub
```

Sie erhalte von ArCon+ die Kennzahl des Dialogs, die hier mit der einzig bekannten Kennzahl ACDLGID_TuerEinstellungen verglichen wird. Weiterhin erhalten Sie einen magischen " Token" , das Sie benutzen können, um an diesen Dialog Erweiterungen anzuhängen. Dazu aktivieren Sie das Formular " Katalog" und übergeben dessen Fensterzugriffszahl (das " Window-Handle" vom Typ " HWND"), sowie das Token an die AttachDialog Funktion. ArCon+ kennzeichnet alle externen Erweiterungen, um dem Benutzer einen Hinweis zu geben, in welchem Handbuch er nachschauen muß, wenn er Fragen zu einem Dialog oder Menüeintrag hat. Dazu verwendet das Beispiel das Icon des Katalog-Forms. Die beiden " False" Parameter geben an, daß die Erweiterung des Dialogs keine besondere Priorität haben soll und daß weitere Erweiterungen (z. B. durch andere Makros) zulässig sind.

ArCon+ fügt dem Standard-Dialog einen Reiter für Ihre Erweiterung hinzu. Sobald der Benutzer diesen Reiter aktiviert, löst ArCon+ das Ereignis DialogActivation aus. In diesem Moment kann Ihr Makro die aktuellen Werte aller Steuerelemente abfragen (wenn z.B. der Inhalt des hinzugefügten Dialogs von diesen Werten abhängig ist). Das gleiche Ereignis wird ausgelöst, wenn der zugefügte Dialog wieder deaktiviert wird, sodaß Ihr Makro in diesem Fall Steuerelementen aus dem Standard-Dialog neue Werte zuweisen kann. Im Beispiel sieht der Code folgendermaßen aus (die Datenbankabfrage wurde zur Vereinfachung ausgelassen):

```
Private Sub ArCon_DialogActivation(ByVal dlgId As Long, ByVal ArConDlgToken As
Long, ByVal subObjectCount As Long, ByVal activated As Boolean)
    If dlgId = ACDLGID_TuerEinstellungen And activated Then
   Dim typName As String
   Dim breite As Single, hoehe As Single
   typName = ArCon.GetDialogData(ArConDlgToken, 0, ACDLGCTL_GenericTyp)
   breite = ArCon.GetDialogData(ArConDlgToken, 0, ACDLGCTL_GenericBreite)
   hoehe = ArCon.GetDialogData(ArConDlgToken, 0, ACDLGCTL_PM_GenericHoehe)
   Katalog.Typ.Caption = typName
   ' Nachschlagen von "typName" in der Katalog-Datenbank
   Katalog.BestellNr.Text = "0815/47-11"
   Katalog.Preis.Text = "12,50 DM"
    End If
End Sub
```

Die Kennzahlen für Steuerelemente wie z.B. ACDLGCTL_GenericTyp ermitteln Sie wiederum mit dem DialogExplorer.

Schließt der Benutzer den erweiterten Dialog, löst ArCon+ das Ereignis EndDialog aus. Auch hier prüft das Makro zunächst, ob es sich um einen bekannten Dialog handelt und entsorgt dann das Form " Katalog" .

```
Private Sub ArCon_EndDialog(ByVal dialogID As Long, ByVal ok As Boolean)
    If dialogID = ACDLGID_TuerEinstellungen Then
```

```
   Unload Katalog
    End If
End Sub
```

Der Parameter “ ok” gibt an, ob der Benutzer den Dialog abgebrochen (ok = False) oder bestätigt hat (ok = True). Wenn die “ Als Standard” Schaltfläche betätigt wird, löst ArCon das Ereignis SaveDialogDefaults aus.

Viele ArCon+ Objekte kennen die Methode “ Edit” und öffnen damit ihren eigenen Eigenschaftsdialog. Damit kann Ihr Makro z.B. eine Tür in eine Planung einfügen und dann dem Benutzer über diesen Dialog die Möglichkeit zur Bearbeitung der neu erzeugten Tür geben. Dabei würden natürlich auch alle Erweiterungen dieses Eigenschaftsdialoges aktiviert.

Eigene Dialoge

Makros können neben der Erweiterung bereits vorhandener ArCon+ Dialoge auch eigene Dialoge anzeigen. Diese Dialoge verhalten sich zumindest ähnlich wie gebundene (modale) Dialoge in ArCon. Aus technischen Gründen gibt es Grenzen in der Gleichbehandlung (Dialog und Elternfenster gehören nicht zur gleichen Applikation) und leider auch geringfügige Unterschiede zwischen Windows 95 und Windows NT.

Da der Dialog nicht von ArCon+ initiert wird, schaltet Ihr Makro ArCon+ nachträglich in den modalen Zustand, sobald der Dialog geöffnet wurde. Das kann z.B. in der Form_Load Prozedur geschehen:

```
Private Sub Form_Load()
    Main.ArCon.StartModalDialog hWnd
End Sub
```

Beim Schließen des Dialoges muß dieser sich wieder abmelden und ArCon+ aus dem modalen Zustand befreien:

```
Private Sub Form_Unload(Cancel As Integer)
    Main.ArCon.EndModalDialog
End Sub
```

Eigene Menüeinträge

ArCon Makros können beliebig komplexe eigene Menüs in die ArCon Menüs integrieren. ArCon verwendet, abhängig vom Anzeigemodus (kein Projekt, 2D Konstruktionsmodus oder 3D Designmodus) eines von drei Menüs. Von Makros erzeugte Menüeinträge erhalten eine Bitmaske aus den Konstanten AC_MenuNoMode, AC_MenuModeConstruct und AC_MenuModeDesign, die die Gültigkeit des Menüeintrages im jeweiligen Modus angibt. Beim Erzeugen eines Menüeintrages teilt ArCon diesem eine Kennzahl (ID) zu, die Ihr Makro speichern muß. Bei späteren Ereignissen (z.B. der Anwahl des Menüpunktes durch den Benutzer) wird der Menüeintrag anhand dieser Kennzahl identifiziert.

Auch alle Standard-Menüeinträge haben (allerdings feste) Kennzahlen. Es sind entsprechende Konstanten vordefiniert, deren Namen selbsterklärend sind. So steht zB. AC_MenuNoMode_Optionen für das Optionsmenü im Modus ohne geladenes Projekt. Alle in verschiedenen Modi auftretenden Menüs und Menüeinträge haben in allen Modi den gleichen Wert, dennoch ist für alle relevanten Modi eine eigene symbolische Konstante definiert.

Außer Menüeinträgen können Sie auch ganze eigene Menüs erzeugen, Trennstriche (Separatoren) in Menüs zur optischen Gruppierung der Befehle einfügen und Untermenüs erzeugen. Das folgende Beispiel zeigt alle diese Funktionen. Es handelt sich um eine Fortsetzung des bereits vorgestellten Türenkataloges. Diesmal erzeugt das Makro ein eigenes “ Katalog” Menü, in dem die Menüpunkte “ Datenbank öffnen” , “ Eintrag suchen” , sowie “ Datenbank schließen” zu finden sind. Unter “ Eintrag suchen” wird ein Untermenü erzeugt, in dem die Einträge “ Nach Namen” und “ Nach Bestellnummer” angeboten werden. Der Eintrag “ Datenbank schließen” ist durch einen Trennstrich abgeteilt.

Alle Menüeinträge erhalten Kennzahlen, daher beginnt das Makro mit der Deklaration von Variablen, um diese Kennzehalen aufzunehmen. Zur Vereinfachung wird außerdem eine Konstante deklariert, die die Gültigkeit der Menüeinträge in allen ArCon Betriebsarten regelt.

```
' Unsere Menüeinträge sind in allen Modi gültig
Private Const allModes As Long = AC_MenuNoMode _
    + AC_MenuModeConstruct + AC_MenuModeDesign
' Variablen für die Menüpunkte
Private itemOpen As Long, searchByName As Long, searchById As Long, itemClose As
Long
```

Die Ids des gesamten Menüs, sowie des “ Eintrag suchen” -Untermenüs werden nur für die Konstruktion des Menüs benötigt. Da Sie später nicht vom Benutzer angewählt werden können (sie enthalten (Unter-)menüs), könnten Sie auch als lokale Variablen in der Prozedur, die das Menü erzeugt, deklariert werden. Diese Prozedur könnte folgendermaßen aussehen:

```
' Variablen für Menüs und Untermenüs
Dim menu As Long, itemSearch As Long


    ' Erzeuge ein eigenes Menü
    menu = ArCon.CreateMenuItem(0, AC_MenuNoMode_Optionen, _
"&Katalog", "", allModes, 0, 0)
```

Hier wird der Befehl CreateMenuItem benutzt, um ein Hauptmenü zu erzeugen. Dazu wird der “ parent” Parameter, der das Menü angibt, in das der neue Eintrag kommen soll, auf 0 gesetzt. Die Positionsangabe erfolgt hier durch die vordefinierte Konstante AC_MenuNoMode_Optionen, sodaß das neue Menü vor das Optionsmenü kommt. Der Name des Menüs (“ Katalog”) enthält das Markierungszeichen “ &” vor dem ersten Buchstaben, sodaß dieser Buchstabe im Menü unterstrichen dargestellt wird und das Menü mit Alt-K aktiviert werden kann. Es folgt der Code, um die Einträge im Katalog-Menü vorzunehmen:

```
    ' Füge darin Einträge ein
    itemOpen = ArCon.CreateMenuItem(menu, -1, _
"&Datenbank öffnen...", _
"Öffnet eine Datenbank", _
allModes, 0, 0)
    itemSearch = ArCon.CreateMenuItem(menu, -1, _
"Eintrag &suchen", _
"Sucht einen Eintrag in der Datenbank", _
allModes, 0, 0)
    ArCon.CreateMenuSeparator menu, -1, allModes
    itemClose = ArCon.CreateMenuItem(menu, -1, _
"Datenbank schließen", _
```

```
"Schließt die aktuelle Datenbank", _
allModes, 0, 0)
```

Mit der gleichen Methode, mit der auch das Menü selbst erzeugt wurde, werden nun einzelne Einträge vorgenommen. Diesmal wird allerdings als Elternmenü die Kennzahl des Katalog-Menüs angegeben (sie könnten hier auch eine der vordefinierten Konstanten für die Standard-ArCon Menüs verwenden, wenn Ihre Einträge in einem Standardmenü erscheinen sollen). Weiterhin wird als Positionsangabe " -1" benutzt, wodurch die Einträge jeweils am Ende des Menüs angehängt werden, also in der Reihenfolge ihrer Erzeugung im Menü stehen.

```
    ' Und ein Untermenü unter "Eintrag suchen"
    searchByName = ArCon.CreateMenuItem(itemSearch, -1, _
"Nach Name..." + Chr$(9) + "F5", _
"Sucht einen Datenbankeintrag mit Hilfe des Namens", _
allModes, 0, 0)
    searchById = ArCon.CreateMenuItem(itemSearch, -1, _
"Nach Bestellnummer..." + Chr$(9) + "F6", _
"Sucht einen Datenbankeintrag mit Hilfe der Bestellnummer", _
allModes, 0, 0)
```

Schließlich wird noch einer der Einträge in ein Untermenü verwandelt, indem seine Kennzahl als Elternmenüangabe in weiteren CreateMenuItem Aufrufen verwendet wird. Die hier verwendeten Namen enthalten, durch Tabulatoren getrennt (Chr$(9)), Tastaturbelegungen für die entsprechenden Menüeinträge. Diese werden im Menü rechtsbündig neben den Text gestellt. Das gesamte Menü sieht nun so aus:

Sobald der Benutzer einen dieser Menüpunkte auswählt, löst ArCon folgendes Ereignis aus:

```
Private Sub ArCon_ExecuteMenuItem(ByVal menuId As Long)
```

Als " menuId" erhält Ihr Makro einen der Werte, die von CreateMenuItem zurückgeliefert wurden und die Ihr Makro in einer der oben definierten globalen Variablen gespeichert hat. Durch den Vergleich mit diesen Werten, erfahren Sie, um welchen Menüpunkt es sich handelt.

Sie können Menüeinträge deaktivieren, sodaß sie nicht anwählbar sind (die Schrift wird grau), oder sie komplett entfernen. Alle Operationen auf Menüeinträgen können Sie nur für die von Ihnen erzeugten Einträge durchführen. Einzige Ausnahme: wenn Ihr Makro durch einen Eintrag im " Makro" Menü gestartet wurde, können Sie diesen Eintrag ebenfalls manipulieren, z.B. deaktivieren, damit Ihr Makro nicht noch einmal gestartet wird:

```
' Falls wir aus dem ArCon "Makro" Menü gestartet wurden, diesen Menüeintrag
' sperren
If ArCon.StartupMenuID <> 0 Then
    ArCon.EnableMenuItem ArCon.StartupMenuID, False
End If
```

Oder Sie entfernen Ihn komplett, wenn Ihr Makro eine imanente Erweiterung von ArCon+ darstellt und sich selbst nicht mehr beendet, bevor ArCon+ beendet wird:

```
If ArCon.StartupMenuID <> 0 Then
    ArCon.RemoveMenuItem ArCon.StartupMenuID
End If
```

Dieser Trick stellt allerdings nicht sicher, daß Ihr Makro nicht mehrmals gleichzeitig aufgerufen wird, da ein versierter Benutzer es natürlich auch über den Explorer oder die Kommandozeile, also unabhängig vom ArCon Menü, starten kann. Wenn Ihr Makro auf keinen Fall gleichzeitig mehrmals laufen darf, müssen Sie dies durch eine Schutzdatei oder andere geeignete Mittel (z.B. Win32-Semaphore) selbst sicherstellen.

Sie können auch dem Benutzer verbieten, Ihr Makro manuell zu starten, indem Sie die Eigenschaft StartupMenuID abfragen und eine entsprechende Meldung ausgeben, wenn sie 0 ist:

```
Private Sub Form_Load()
    If ArCon.StartupMenuID = 0 Then
   MsgBox "Bitte starten Sie dieses Makro nur über das Makro-Menü in ArCon!", _
  vbOKOnly + vbCritical, _
  "FATAL: ArCon Makro wurde nicht aus ArCon gestartet"
   Unload Me
   Exit Sub
    End If

    If Not ArCon.StartMe(hWnd, 1, "Macros.hlp") Then
   MsgBox "ArCon läuft nicht"
```

```
Unload Me
Exit Sub
 End If

 ArCon.RemoveMenuItem ArCon.StartupMenuID
```

StartupMenuID gehört zu den wenigen Eigenschaften, die bereits vor dem Verbindungsaufbau zu ArCon mit dem Befehl StartMe verfügbar sind.

Eigene Schaltflächen

ArCon kennt verschiedene Arten von Schaltflächen:

- **Taster** bleiben nur solange im gedrückten Zustand, wie Sie den Mausknopf festhalten. Sobald Sie loslassen, springt die Schaltfläche wieder hervor.
- **Schalter** wechseln beim Anklicken ihre Stellung und verbleiben dann entweder im gedrückten oder ungedrückten Zustand.
- **Variantenschalter** verhalten sich wie Schalter, können aber zusätzlich zum gedrückt/ ungedrückt noch zwischen verschiedenen Varianten auswählen.
- **Automatische Variantenschalter** treten nur in der " Wie" -Leiste auf. Sie sind gekoppelt an den gedrückten Zustand eines jeweils zuständigen Variantenschalters in der " Was" -Leiste.

Während Taster, Schalter und Variantenschalter weitgehend einheitlich erstellt werden und auch durch gemeinsame Ereignisse mit Ihrem Makro kommunizieren, ist die Programmierung von automatischen Variantenschaltern wesentlich aufwendiger, weshalb dieses Thema im nachfolgenden Kapitel einzeln behandelt wird.

0 Jeder Knopf in ArCon hat mindestens ein Bild. Daher stellt sich zunächst die zentrale Frage: woher kommt das Bild für Ihre Schaltfläche. Jede Funktion zur Knopferstellung gibt es in zwei Varianten, zum Beispiel DefButton und DefButtonRes. Die Namen unterscheiden sich jeweils durch den Postfix " Res" für " Resource" . In der nicht-Res Version erwartet ArCon von Ihrem Makro ein Bitmap-Handle (HBITMAP), wie es in Visual Basic zum Beispiel die Funktion " LoadPicture" liefert. Diese Version bietet sich an, wenn Sie entweder die Bitmap bereits zu anderen Zwecken geladen haben (zum Beispiel in einem " PictureBox" Steuerelement, dessen " Picture" Eigenschaft das gewünscht Bitmap-Handle liefert), oder wenn Sie das Aussehen des Knopfes variabel halten und die zu ladende Datei erst zur Laufzeit ermitteln (zum Beispiel bei Variantenschaltern, deren Varianten von optional installierten Komponenten stammen, wie etwa geladene Türen und Fenster in ArCon).

1 Alternativ können Sie die " Res" Version benutzen und die Bitmap aus fest in Ihr Programm oder eine seiner DLL' s eingebundenen Resourcen laden. Sie identifzieren die Bitmap durch seine Kennzahl und das Instance-Handle (HINSTANCE) der DLL oder EXE-Datei, aus der die Resourcen stammen. Falls Sie keine Resourcen in DLL' s oder externenen EXE-Dateien ausgelagert haben, können Sie als Instance-Handle auch " 0" übergeben, in diesem Fall lädt ArCon die Bitmap aus den Resourcen Ihrer EXE-Datei. Achtung: in Visual Basic funktioniert dies nur, wenn Sie Ihr Programm auch aus einer EXE-Datei heraus starten. Falls Sie es in der Visual-Basic Entwicklungsumgebung direkt starten, sind seine Resourcen für ArCon nicht verfügbar und die Knöpfe erhalten falsche Bitmaps.

2 Neben dem Aussehen des Knopfes müssen Sie noch weitere Eigenschaften angeben: wo soll der Knopf erscheinen, was für ein Knopf ist es und in welchen ArCon-Modi ist er sichtbar. Diese Daten werden als Sammlung von einzelnen Flags in einem 32-bit Wert (long) kodiert, der Button-Info. Achten Sie darauf, mindestens eines der Bits für die Art des Buttons (ACBI_CheckButton oder ACBI_PushButton) zu definieren - ansonsten bleibt Ihr Knopf unsichtbar. Das folgende Beispiel erzeugt einen Schalter in der oberen Knopfleiste in allen Darstellungsmodi:

```
    Const Info As Long = ACBI_UpperPannel + ACBI_CheckButton + ACBI_AllModes
0      Const hInst As Long = 0
1      Dim btn As Long
2      btn = ArCon.DefButtonRes(101, hInst, Info, _
3    "Dieser Knopf bedeutet seinem Autor sehr viel - " & _
4    "auch wenn er keine wirkliche Bedeutung hat.|" & _
5    "Nur eine kurze Bublehelp", 4733)
```

Die Bitmap für den Knopf stammt aus den Resourcen der EXE-Datei (hInst = 0) und hat die Kennzahl 101. Als weitere Parameter folgt nach dem bereits erläuterten Info-Parameter eine Erklärung des Knopfes, die in der Statuszeile angezeigt wird. Falls diese Erklärung lang ist, kann durch einen " |" getrennt eine Kurzfassung für die Schnellhilfe angegeben werden - ansonsten wird die gleiche Erklärung wie in der Statuszeile angezeigt. Schließlich folgt noch ein Hilfekontext, der auf die Erklärung dieses Knopfes in der Online-Hilfe Ihres Makros verweist.

0 Als Rückgabewert der Funktion erhalten Sie eine eindeutige Kennung für den neuen Knopf, über die ArCon ihn in allen Knöpfe-betreffenden Ereignissen identifiziert.

1 Um einen Variantenschalter zu definieren, erstellen Sie zunächst den eigentlichen Knopf:

```
    btn = ArCon.DefMultiButton(Info)
```

Außer der Button-Info sind hier noch keine weiteren Daten nötig, da diese variantenspezifisch sind. Fügen Sie nun nacheinander alle gewünschten Varianten hinzu:

```
    ArCon.DefMultiButtonVariantRes btn, 101, hInst, _
0     "Dieser Knopf bedeutet seinem Autor sehr viel - " & _
1     "auch wenn er keine wirkliche Bedeutung hat.|" & _
2     "Nur eine kurze Bubblehelp", 4733
3       ArCon.DefMultiButtonVariantRes btn, 102, hInst, _
4     "Die zweite Bubble-Help", 4734
5       ArCon.DefMultiButtonVariantRes btn, 103, hInst, _
6     "Die dritte Bubble-Help", 4735
```

Um dem Knopf mitzuteilen, daß nun alle Varianten verfügbar sind, müssen Sie ihn einmal explizit auf “ sichtbar” schalten (dies ist beim einzelnen Schalter nicht nötig, da diese automatisch sichtbar erzeugt werden). Dazu dient die Funktion SetButtonState. Erst durch diesen Aufruf wird der Knopf von ArCon angezeigt. Nachträglich können Sie keine weiteren Varianten hinzufügen.

```
    ArCon.SetButtonState btn, 2, True, True, True
```

Parameter sind hier der zu bearbeitende Knopf, die auszuwählende Variante (“ 2” , also die dritte, da “ 0” die erste ist) sowie die Eigenschaften “ pressed” , “ visible” und “ enabled” .

0 Alle Aktivitäten des Benutzers bezüglich eines Knopfes werden zentral durch das ButtonChange Ereignis gemeldet. Eine Ereignisprozedur könnte so aussehen:

```
Private Sub ArCon_ButtonChange(ByVal btnId As Long, ByVal evnt As Long, _
0     ByVal selectedIndex As Long, ByVal pressed As Boolean, _
1     ByVal visible As Boolean, ByVal enabled As Boolean)
2      If btnId = btn Then
3     Select Case evnt
4    Case ACBE_SelChanged
5   Debug.Print "Auswahl wurde geändert: "; selectedIndex
6    Case ACBE_LeftClicked
7   If pressed Then
8  Debug.Print "Sie haben Funktion "; selectedIndex; _
9    " eingeschaltet"
10  Else
11 Debug.Print "Sie haben Funktion "; selectedIndex; _
12   " ausgeschaltet"
13  End If
14   Case ACBE_RightClicked
15  PropertyDlg.Show 1
16    End Select
17     End If
18 End Sub
```

Hier wird zunächst die Kennzahl des Schalters überprüft und anschließend nach Art des aufgetretenen Ereignisses unterschieden und die entsprechende Debug-Ausgabe erzeugt. Die möglichen Ereignisse sind:

ACBE_SelChanged	Die ausgewählte Variante des Knopfes hat sich geändert (der Knopf zeigt jetzt eine andere Bitmap an)
ACBE_LeftClicked	Der Knopf wurde mit der linken Maustate angeklickt
ACBE_RightClicked	Der Knopf wurde mit der rechten Maustaste angeklickt

Über die Kennzahl und das Ereignis hinaus wird die komplette Knopfstatusinformation mitgeliefert, auch wenn dies bei einigen Knopfarten keinen Sinn macht. So ist der Parameter “ selectedIndex” nur für Variantenschalter interessant und der Parameter “ pressed” für Taster sinnlos.

Automatische Schaltflächen

Dieses Kapitel behandelt ausschließlich die sehr speziellen automatischen Variantenschalter in ArCon. Um sich über weitere Schaltflächenarten zu informieren, schlagen Sie im vorhergehenden Kapitel nach.

0 Die Programmierung automatischer Schaltflächen ist aufwendig, da es sich um komplexe Objekte der Benutzeroberfläche handelt, die verschiedene Zustände (unterschiedlich je nach Schaltfläche) durchlaufen. ArCon bemüht sich, Ihrem Makro bei der Verwaltung dieser Zustände zu helfen. Um einen Überblick über die Mächtigkeit automatischer Schaltflächen zu bekommen, betrachten Sie einige der fest in ArCon integrierten:

- Um Wände zu verlegen, wählen Sie aus einem Variantenschalter in der " Was" -Leiste eine Wandart aus. Unabhängig von der gewählten Variante werden Ihnen in der " Wie" -Leiste verschiedene Hilfsmittel zum Verlegen der Wände angeboten, z.B. " Mehrere Wände einziehen" , " Einzelne Wand einziehen" und " Wand senkrecht zu einer Referenzlinie einziehen" . Jeder Schalter in der " Wie" -Leiste ist in diesem Fall eine automatische Schaltfläche (genaugenommen benutzt ArCon in der " Wie" -Leiste ausschließlich automatische Variantenschalter, lediglich Makros können dort auch andere Schaltflächen unterbringen).
- Um Treppen einzufügen, wählen Sie die gewünschte Treppenart im Variantenschalter der " Was" -Leiste. In der " Wie" -Leiste erscheint daraufhin ein einziger automatischer Variantenschalter. Wenn Sie diesen aktivieren, können Sie nacheinander drei Punkte in der Planung anklicken, wobei zwischenzeitlich bereits eine Skizze der Treppe erscheint, soweit sie bisher definiert ist.
- Um Gelände zu bearbeiten, gibt es vier Varianten in der " Was" -Leiste. Je nach hier ausgewählter Variante werden unterschiedliche automatische Variantenschalter in der " Wie" -Leiste angeboten.

Alle diese Dinge kann Ihr Makro ebenfalls. Zunächst benötigen Sie einen Schalter in der " Was" -Leiste, der den oder die automatischen Variantenschalter aktiviert.

```
    Const Info As Long = ACBI_AllModes + ACBI_LeftPannel
0       Const hInst As Long = 0
1       Dim was As Long

2       ' Erstmal den Knopf anlegen
3       Button = ArCon.DefMultiButton(Info)
```

Für diesen Variantenschalter definieren Sie nun Varianten:

```
    ' Erzeuge die einzelnen WAS Varianten
0       was = ArCon.DefMultiButtonVariantRes(Button, bm1, hInst, _
1   "Weg im Gelände erzeugen", 5032)
```

Dieser Knopf ist noch nicht sichtbar, denn es handelt sich um einen Variantenschalter, zu dem Sie jetzt noch weitere Varianten hinzufügen könnten. Für jede der Varianten (hier: nur eine) können Sie nun automatische Variantenschalter für die " Wie" -Leiste erzeugen. Sobald der Knopf einmal sichtbar gesetzt wurde (mit SetButtonState), können Sie weder weitere Varianten, noch weitere automatische Schaltflächen hinzufügen. Die Erzeugung der automatischen Varianten kann so aussehen:

```
    ' Jetzt noch ein paar automatische Variantenschalter in der Wie-Leiste
0       How1 = ArCon.DefHowButtonRes(was, bm1, hInst, Info, "Spline", 5035)
1       How2 = ArCon.DefHowButtonRes(was, bm1, hInst, Info, "Bezier", 5035)
2       How3 = ArCon.DefHowButtonRes(was, bm1, hInst, Info, "Exakt", 5035)
```

Dieser Code ist der Erzeugung von Varianten für Variantenschalter sehr ähnlich, nur daß hier als " Eltern-Knopf" nicht ein " MultiButton" dient, sondern eine Variante eines " MultiButton" . Nach diesen Vorbereitungen kann der gesamte Knopf nun sichtbar gesetzt werden:

```
    ' Fertig, jetzt das Ganze aktivieren und sichtbar machen
0       ArCon.SetButtonState Button, 0, True, True, True
```

ArCon kennt nun Optik und Aktivierungsabhängigkeiten Ihres automatischen Variantenschalters, es weiß aber noch nichts über das Laufzeitverhalten. Andere Knopfarten schicken einzelne Ereignisse an Ihr Programm, nicht so die automatischen Knöpfe. Hinter jeder Variante steckt ein komplexer Bedienungsablauf in mehreren Schritten.

0 Sobald ein automatischer Knopf aktiv ist, löst ArCon HowMove Ereignisse aus, sobald der Benutzer die Maus bewegt. Ihr Programm kann nun Vorschau-Bilder erzeugen, indem es 2D-Grafikprimitiven einsetzt. Diese werden nicht in die globale Grafikliste Graphics2D gestellt, sondern in die spezielle Liste RunningTool. Die Elemente dieser Liste werden von ArCon automatisch so verwaltet, daß ein " Gummiband-Effekt" entsteht, ohne daß Ihr Programm sich um das Neumalen und wieder löschen kümmern muß. In dieser Liste können Sie z.B. die Skizze für eine zu verlegende Wand oder zu plazierende Treppe unterbringen, oder einen Gauben-Cursor zeichnen. Da ArCon das Zeichnen für Sie übernimmt, ist das einzige, was Sie in der Ereignisprozedur für das HowMove Ereignis erledigen müssen, die eventuelle Korrektur der Koordinaten der Grafikelemente, sofern diese von der aktuellen Mausposition abhängen.

```
Private Sub ArCon_HowMove(ByVal btnId As Long, ByVal x As Double, _
0   ByVal y As Double, ByVal viewHandle As Long)
1      If ArCon.RunningTool.Points.Count <> 0 Then
2     ' nach dem ersten Punkt eine Gummibandlinie zeigen
3     Dim i As Integer, l As Object
4     i = ArCon.RunningTool.Graphics2D.Count
5     Set l = ArCon.RunningTool.Graphics2D.Item(i)
6     l.X2 = x
7     l.Y2 = y
8      End If
9  End Sub
```

Diese Prozedur geht davon aus, daß das letzte Element der Grafikliste eine Linie vom letzten gesetzten Punkt zur aktuellen Mausposition ist, deren Endpunkt korrigiert werden muß, wenn die Maus bewegt wurde. Falls Ihr Programm mehrere automatische Variantenschalter implementiert, muß es natürlich hier zunächst die " btnId" testen.

0 Wenn der Benutzer mit der Maus einen Punkt anklickt oder die Eingabetaste betätigt, wird von ArCon ein HowInput erzeugt. In der Regel wird Ihr Programm in diesem Moment drei Dinge tun:

- die aktuellen Koordinaten (der Maus) speichern
- (neue) Grafikelemente in die temporäre Liste einfügen und andere eventuell anpassen.

Zur Speicherung der Koordinaten bietet ArCon das RunningTool.Points Objekt an. Damit kann die Ereignis-Prozedur so aussehen:

```
Private Sub ArCon_HowInput(ByVal tool As Long, _
   ByVal x As Double, ByVal y As Double, _
   ByVal dirX as Double, ByVal dirY as Double, _
   ByVal state As Long)


    Dim t As Object
    Set t = ArCon.RunningTool
    t.Points.AddPoint x, y


    If t.Points.Count >= 3 Then
   t.Finish
    Else
   Dim l As Object
   Set l = ArCon.NewLine(AC_LayerLast)
   Dim i As Integer
   i = t.Graphics2D.Count
   If i >= 1 Then
  l.X1 = t.Graphics2D.Item(i).X2
  l.Y1 = t.Graphics2D.Item(i).Y2
   Else
  l.X1 = x
  l.Y1 = y
   End If
   l.X2 = x
   l.Y2 = y
   t.Graphics2D.Add l
   t.Continue
    End If
End Sub
```

Hier wird der neu eingegebene Punkt in die Punktliste aufgenommen. Dann wird geprüft, wieviele Punkte bereits gespeichert sind. Bei drei oder mehr Punkten wird der Eingabevorgang beendet

(vergleiche z.B. die Treppenpositionierung in ArCon), indem die Finish Methode des RunningTool Objektes aufgerufen wird. Andernfalls wird die temporäre (Cursor-) Grafik erzeugt bzw. aktualisiert und der Eingabevorgang durch Aufruf der Continue Methode fortgesetzt.

In realen Programmen würde beim Aufruf der Finish Methode noch die eigentliche Arbeit ausgeführt werden, z.B. ein Objekt an der ermittelten Position eingefügt.

SPEICHERN UND LADEN

Ihre Makros können Zusatzinformationen in ArCon-Projektdateien speichern und sie daraus wieder lesen. Falls Ihr Makro dateibasiert arbeitet, kann es seine gesamten Information im ArCon-Projekt ablegen und damit dem Benutzer “ alles in einer Datei” liefern. Sollte Ihr Makro Daten in einer Datenbank speichern, können Sie zum Beispiel eindeutige Schlüssel zur Identifikation des ArCon-Projektes in diesem abspeichern.

ArCon verwendet ein hierachisches Speicherverfahren, in dem einzelne Elemente durch Kennzahlen, sogenannte “ Chunk-ID’ s” identifiziert werden. ArCon ignoriert alle unbekannten Chunk-ID’ s, sodaß Sie in solchen Elementen beliebige Informationen in einem Dateiformat Ihrer Wahl speichern können.

Da Sie als Entwickler keine Möglichkeit haben, in eine ArCon Projektdatei zu schauen, empfiehlt es sich, Ihr Makro zunächst Dateien schreiben zu lassen. Nachdem Sie diese Version getestet haben, können Sie die Dateien ArCon übergeben, das sie daraufhin in seine Projektdatei kopiert. Genauso funktioniert das Laden von Daten: ArCon kopiert einen Chunk aus seiner Projektdatei in eine Datei Ihrer Wahl, woraufhin Ihr Makro diese Datei liest.

ArCon speichert und liest Projekte nach Gebäuden geordnet. Zusätzlich werden globale Daten gelesen und gespeichert. Der Benutzer kann auch einzelne Gebäude in ein anderes Projekt importieren, wobei dann keine globalen Daten gelesen werden sollten - da ArCon Ihre Daten aber nicht interpretieren kann, kann es auch nicht verhindern, daß Sie sich nicht an diese Regelung halten.

Da Gebäude an sich keine feste Reihenfolge haben, wird ihnen beim Speichern eine willkürliche Ordnungsnummer zugeordnet, die ArCon Ihrem Makro vor dem Speichern des jeweiligen Gebäudes mitteilt. Beim Laden werden die Gebäude dann in der Reihenfolge dieser Nummern gelesen und Ihr Makro erhält wiederum die Ordnungszahl, unter der dieses Gebäude gespeichert wurde. Sie können anhand dieser Identifikation Ihre gespeicherten Daten beim Laden wieder eindeutig identifizieren, sie können sich aber nicht darauf verlassen, daß ein Gebäude beim nächsten Speichern wieder die gleiche Ordnungszahl zugeordnet bekommt.

Die Speicherplätze innerhalb eines ArCon Projektes (“ Chunks”) haben projektglobale Namen (“ Chunk-ID’ s”), Ihr Makro kann daher nicht die Daten für das erste und das zweite Gebäude jeweils unter der gleichen Chunk-ID speichern. Sie werden daher in der Regel einen ganzen Bereich von Chunk-ID’ s verwenden - oder die Verschachtelung bereits intern im eigenen Dateiformat regeln und nur das Gesamtergebnis in einen einzelnen ArCon-Chunk schreiben. Die Chunk-ID’ s für Ihre Daten müssen aus dem im Kapitel Benutzerdefinierte Chunk-IDs festgelegten Bereich stammen. Falls Ihr Makro vermarktet werden soll, lassen Sie sich bitte einen eigenen Chunk-ID-Bereich von mb-Software reservieren.

Das folgende Beispiel speichert im globalen Bereich des Projektes (das übergebene zu speichernde/ladende Gebäude ist Nothing) eine Kennung, die den Schlüssel des entsprechenden Projektes in eine Datenbank enthält. Sie finden das vollständige Beispiel im Verzeichnis “ Load-n-Save” der Makro-Beispielprogramme.

Zunächst benötigen Sie eine Chunk-ID (da nur globale Daten gespeichert werden, genügt eine einzige lD).

```
' Um unsere Daten innerhalb eines ArCon Projektes zu identifizieren, benötigt
' ArCon eine ID.
Private Const ChunkID As Long = AC_CHUNKID_MIN + 4711
```

Unter dieser Chunk-ID sollen nun Daten gespeichert werden. ArCon löst beim Speichern mehrere Ereignisse aus. Zunächst wird Ihr Makro durch ein “ SaveStart” Ereignis informiert, daß eine Speicherung beginnt. Ihr Programm muß daraufhin eventuell zu speichernde Chunks anmelden, indem es den “ numChunksToSave” Parameter entsprechend erhöht. ArCon wird später nur soviele Chunks speichern, wie hier insgesamt angegeben wurden (bedenken Sie, daß Ereignisse von allen gerade laufenden Makros empfangen werden). Ihr Code zur Behandlung dieses Ereignisses könnte z.B. so aussehen:

```
Private Sub ArCon_SaveStart(ByVal fileName As String, ByVal NumBuildings As Long,
ByVal isAutoSave As Boolean, numChunksToSave As Long)
    Debug.Print "ArCon speichert das aktuelle Projekt als "; fileName
    If isAutoSave Then
   Debug.Print "Es handelt sich um eine automatische Sicherung"
    End If
    Debug.Print "Gespeichert werden "; NumBuildings; " Gebäude"
    ' Wir werden einen zusätzlichen Chunk speichern
```

```
    numChunksToSave = numChunksToSave + 1
End Sub
```

Anschließend wird je ein “ SaveBuilding” Ereignis für die globalen Daten (Gebäude = Nothing) und für jedes zu speichernde Gebäude ausgelöst. Das Beispielprogramm speichert nur globale Daten:

```
Private Sub ArCon_SaveBuilding(ByVal BuildingNo As Long, ByVal B As Building)
    If B Is Nothing Then
   ' Globale Daten speichern
   Debug.Print "Speichere Chunk"

   Dim fileName As String, line As String

   ' Erzeuge einen temporären Dateinamen
   fileName = Environ("TEMP") + "\tmp4711.$$$"
   ' Schreibe die Daten in eine temporäre Datei
   Open fileName For Output As #1
   Print #1, "AVA-ID=X345-222-982091"
   Close #1
   ' Kopiere die Datei ins ArCon Projekt
   ArCon.WriteChunk ChunkID, fileName
   ' Lösche die temporäre Datei
   Kill fileName
    End If
End Sub
```

Wird ein Projekt geladen, löst ArCon zunächst ein “ LoadStart” Ereignis aus, das Ihrem Programm die Möglichkeit gibt, Datenstrukturen vorzubereiten. Das Beispielprogramm ignoriert dieses Ereignis. Anschließend wird für die globalen Daten, sowie für jedes geladene Gebäude ein “ LoadBuilding” Ereignis ausgelöst. Das Beispielprogramm lädt die globalen Daten, die es zuvor gespeichert hat:

```
Private Sub ArCon_LoadBuilding(ByVal B As Building, ByVal SaveNo As Long)
    If B Is Nothing Then
   ' Globale Projektdaten laden
   Dim fileName As String, line As String

   ' Erzeuge einen temporären Dateinamen
   fileName = Environ("TEMP") + "\tmp4711.$$$"

   ' Kopiere die Daten aus dem ArCon Projekt in die temporäre Datei
   If ArCon.ReadChunk(ChunkID, fileName) Then
  ' Lese die Datei
  Open fileName For Input As #1
  Line Input #1, line' Im Test nur eine einzige Zeile...
  Close #1
  ' Lösche die Datei
  Kill fileName

  ' Fertig, die Daten sind geladen
  Debug.Print "Globale Projektdaten aus dem ArCon Projekt:"
  Debug.Print line
   End If
    End If
End Sub
```

Abschließend wird ein “ LoadComplete” Ereignis ausgelöst, das Ihrem Programm mitteilt, ob der Ladevorgang erfolgreich war oder nicht. Ihr Programm kann in diesem Ereignis Zuordnungen der Gebäude zu Speichernnummern abgleichen und/oder Datenstrukturen wieder freigeben. Das Beispielprogramm ignoriert dieses Ereignis.

EINDEUTIGE KENNUNGEN

Um Zusatzdaten über die von ArCon zu konstruktiven Elementen erfassten Daten hinaus zu verwalten, ist es notwendig, diese Elemente eindeutig zu identifizieren. ArCon bietet dazu eine einfache Möglichkeit. Innerhalb eines Projektes erhalten alle Objekte eindeutige Nummern, die Sie mit der Eigenschaft “ ID” abfragen können. Diese Kennzahlen sind persistent, das heißt, beim nächsten Laden eines Projektes werden die Objekte wieder die gleiche Kennzahl benutzen. Die Kennzahl ist aber nicht über Projektgrenzen hinaus eindeutig.

Konflikte treten nur dann auf, wenn aus einem anderen Projekt einzelne Gebäude geladen werden. ArCon löst dabei die entsprechenden LoadStart, LoadBuilding und LoadComplete Ereignisse aus, sodaß Ihr Makro darauf reagieren kann. Während des Ladens werden für die importierten Objekte neue Kennzahlen vergeben. Wenn Sie extern gespeicherte Daten zu diesen importierten Objekten übernehmen möchten, müssen Sie eine Zuordnung der alten Kennungen im Ursprungsobjekt zur beim Import neu vergebenen Kennung herstellen.

Angenommen, Sie speichern Metainformationen zu Wänden in einer Datenbank. Ihre Datenbankschlüssel bestehen dabei aus einer Projekt-ID (die Sie vom Datenbanksystem generieren lassen) und den ArCon Ids. Die Projekt-ID in der Datenbank speichern Sie als benutzerdefinierten Chunk im ArCon Projekt, wie im vorigen Kapitel Speichern und Laden beschrieben. Werden einzelne Gebäude geladen, versuchen Sie diesen Chunk auch im Importprojekt zu lesen. Falls dies funktioniert gibt es auch zu diesem Projekt bereits Metainformationen. Sie bewahren daher die Projekt-ID auf und warten auf das LoadComplete Ereignis. In dessen Ereignisprozedur müssen Sie nun für alle neu importierten Objekte die Metadaten kopieren und mit der neuen ID dieser Objekte verknüpfen. Dazu iterieren Sie über alle Objekte und ermitteln von jedem die “ History” Eigenschaft. Diese liefert entweder “ Nothing” (falls das Objekt nicht nachgeladen wurde) oder ein IDHistory Objekt. Dieses History-Objekt liefert Ihnen den Namen des Projektes, aus dem importiert wurde (dem also Ihre zu kopierenden Metadaten bisher zugeordnet sind) ID im alten Projekt und neu vergebene ID im aktuellen Projekt. Damit können Sie die Metadaten kopieren und neu zuordnen.

History-Objekte sollten nicht aufbewahrt werden. Sie verlieren beim nächsten Ladevorgang Ihre Gültigkeit.

REFERENZ DER MAKROBEFEHLE

Das folgende Kapitel beschreibt alle Objekttypen.

ArCon
BackgroundSettings
Building
BuildingCollection
Ceiling
CeilingCollection
CeilingOpening
CeilingOpeningsCollection
Chimney
ChimneyCollection
Contur
ConturCollection
Cut
CutCollection
CutView
Dimension
DimensionCollection
Door
DoorCollection
Gaube
GaubenCollection
Graphics2DCollection
Graphics2DObject
Guide
GuideCollection
Hedge
Hole
HoleCollection
HolePolygon
IDHistory
Image
Label
Labeling
LabelingCollection
LambdaH
LambdaHCollection
Line
Material
Object3D
Object3DCollection

ObjectConstructor
ObjectConstructorCollection
ObjectTransformer
ObjectTransformerCollection
Point2D
Point2DCollection
Polygon2D
PolygonWendelConstruction
PrintSettings
Project
ProjectPreview
Roof
RoofArea
RoofAreaCollection
RoofCollection
RoofConstruction
RoofWindow
RoofWood
RoofWoodCollection
Room
RoomCollection
SavePictureSettings
Shape
SnapObject
StairCase
StairCaseCollection
Story
StoryCollection
Support
SupportCollection
Terrain
TerrainCollection
Texture
TextureCollection
TextureName
ToolData
UnterUeberzug
UnterUeberzugCollection
View
ViewCollection
Viewing
VirtualWall
VirtualWallCollection
WalkSettings

Wall
WallCollection
WallSegment
WallSegmentCollection
Window
WindowCollection
ZoomSettings

ArCon

Das ArCon-Objekt stellt die Verbindung zu ArCon dar, es steuert globale Eigenschaften von ArCon, des aktuellen ArCon-Projektes und verwaltet die Benutzeroberfläche.

In der Regel wird jedes ArCon Makro genau ein Element dieses Typs benutzen und dieses in einer globalen Variablen allen Programmteilen zur Verfügung stellen (oder z.B. in Visual Basic durch qualifizierte Namen wie “ Main.ArCon” aus anderen Forms heraus ansprechen).

Mehrere ArCon Objekte in einem Makro machen nur Sinn, wenn Ihr Makro mit mehreren ArCon. EXE kommunizieren muß. Sie benötigen dann für jede ArCon.EXE ein ArCon Objekt.

ArCon Übersicht
Eigenschaften

AllFloorsVisible	Anzeige aller Geschosse/nur des aktuellen
ArConDirectory	Das Progammverzeichnis der ArCon Installation
ArConWindowHandle	Kennzahl des ArCon Hauptfensters
AutoLoaded	Wahr, wenn das Makro durch das Laden eines Projektes mit makrospezifischen Daten automatisch geladen wurde.
AvailableChimneys	Anzahl der verfügbaren Schornsteintypen
AvailableDoors	Anzahl der verfügbaren Türtypen
AvailableGauben	Anzahl der verfügbaren Gaubentypen
AvailableRoofWindows	Anzahl der verfügbaren Dachfenstertypen
AvailableStairCases	Anzahl der verfügbaren Treppentypen
AvailableSupports	Anzahl der verfügbaren Stützentypen
AvailableWalls	Anzahl der verfügbaren Wandtypen
AvailableWindows	Anzahl der verfügbaren Fenstertypen
Buildings	Liste aller Gebäude
ConstructedRoofWindows	Anzahl der konstruierten Dachfenstertypen
ConstructedWindows	Anzahl der konstruierten Fenstertypen
CurrentBuilding	Liefert das aktuelle Gebäude
CurrentProject	Liefert das aktuelle Projekt
CurrentStory	Liefert das aktuelle Stockwerk
Cuts	Liste aller Schnitte
DesignObjectConstructors	Die Klassen aller geladenen Einrichtungsgegenstände
DesignObjects	Alle geladenen Einrichtungsgegenstände
Dimensions	Liste aller Bemaßungen
GlobalTerrain	Umgebungsgelände
Graphics2D	Die aktiven 2D Graphikobjekte
Guides	Liste aller Hilfslinien
InternalMode	Interner Modus (undokumentiert)
Labelings	Liste aller Beschriftungen
Mode	Aktueller Bearbeitungsmodus
MultiUserMode	Flags zur Steuerung der Multi-User Eigenschaften von ArCon
ProgramName	Name und Version der ArCon EXE Datei
ProgramProperties	Eigenschaften der EXE Version, z.B. ARCONPROP_3DS
ProgramTypeID	Art der EXE Version, z.B. ARCONVERSION_ArCon
ProgramVersion	Versionsnummer (Hiword.Loword) der EXE Datei
Running	Das Makro ist mit ArCon verbunden
RunningTool	Liefert bei entsprechendem Event ausgewähltes Tool
StartupMenuID	Menü-ID des Eintrages, über den dieses Makro gestartet wurde
State	Welche Hilfsmittel sind aktiviert?
Terrains	Liste aller Grundstücke

Methoden

ActiveExternalViews	Bitmaske der gerade sichtbare Zusatzansichten (z.B. Explorer)
ActiveView	Die momentan aktive Ansicht
AddPredefinedViewing3D	Erzeugt einen neuen Betractherstandpunkt, liefert dessen Index (-1 bei Fehlern). Alle anderen Indizes werden ungültig.
AttachDialog	Dialog an ArCon-Dialog anhängen
AvailableExternalViews	Bitmaske der verfügbaren Zusatzansichten (wie z.B. der Explorer)
AvailablePfostenObjects	Liefert die Anzahl der Treppenpfosten-Objekttypen
ChangePredefinedViewing3D	Ändert einen vordefinierten Betrachterstandpunkt
ChangeTypeNotifyMask	Ändert die typabhängig globale Eventmaske
Clear3DObjectSelection	Hebt die Selektion aller 3D Objekte auf
CloseDialog	Schließt einen mit AttachDialog exklusiv übernommenen Dialog
CloseProject	Projekt schließen
CreateBuilding	Entspricht Gebäude/Neues Gebäude unter/oberhalb
CreateMenuItem	Menü-Eintrag erzeugen
CreateMenuSeparator	Separator in Menü eintragen
CreateProject	Neues Projekt erzeugen
CreateUserPanel	Erzeugt eine benutzerdefinierte Knopfleiste
DBIDToString	Erzeugt aus den Einzelteilen eines Datenbankschlüssels einen (Pseudo-) Dateinamen
DefButton	Schaltfläche definieren
DefButtonRes	Schaltfläche definieren, Bitmap aus Resource laden
DefHowButton	Automatische Wie-Schaltfläche erzeugen
DefHowButtonRes	Automatische Wie-Schaltfläche erzeugen, Bitmap aus Resource laden
DefMultiButton	Multifunktionsschaltfläche definieren
DefMultiButtonVariant	Multifunktionsschaltfläche definieren, Bitmap aus Resource laden
DefMultiButtonVariantRes	Variante für Multifunktionsschaltfläche definieren
DeleteButton	Schaltfläche löschen
DeletePredefinedViewing3D	Löscht einen vordefinierten Betrachterstandpunkt
DestroyUserPanel	Löscht eine benutzerdefinierte Knopfleiste und alle Knöpfe darin
DisableKeyboard	Verhindert Viewing-Veränderungen per Keyboard (aView = Nothing: für alle Ansichten)
DisableKeyboardShortcuts	Schaltet Acceleratoren aus
DoorName	Liefert den Namen des Tür-Objektes für den angegebenen Türtyp
DragEnd	Beendet einen Drag&Drop Vorgang
DragMove	Übergibt neue Mauskoordinaten während eines Drag&Drop Vorganges
DragRButtonPressed	Übergibt einen Mausklick während eines Drag&Drop Vorganges

DragTextureStart	Beginnt einen Textur-Drag&Drop Vorgang
DragTextureStart2	Beginnt einen Textur-Drag&Drop Vorgang
EnableAllButtonsInPanel	Aktiviert/Deaktiviert alle Knöpfe in einem Panel. IDs entnehmen Sie bitte der Dokumentation.
EnableButtonByID	Aktiviert/Deaktiviert einen Knopf. IDs und Sub-IDs entnehmen Sie bitte der Dokumentation.
EnableMenuItem	Menü-Eintrag ein/ausschalten
EndArCon	ArCon beenden
EndMe	Verbindung von ArCon trennen
EndModalDialog	ArCon bei Ende von nichteigenen, modalen Dialogboxen benachrichtigen
GetBackgroundSettings	Liefert die Standardeinstellungen für Hintergründe
GetButtonState	Status von Schaltfläche lesen
GetChimneyName	Liefert den Namen des n-ten Schornsteintyps
GetCompas	Fragt die Daten der Nordrichtungsanzeige ab
GetConstructionModeSnapSettings	Holt die Fangeinstellungen für den Konstruktionsmodus
GetDBInfo	Liefert Informationen für eine angemeldete Datenbank
GetDBInfoByID	Liefert Informationen über eine konkrete Datenbank
GetDatabaseConfiguration	Liefert interne Daten zur Datenbankkonfiguration, Ergebnis ist Falsch wenn es sich nicht um eine Datenbankversion handelt
GetDefaultCeiling	Liefert die Standardeinstellungen für Deckenplatten
GetDefaultChimney	Liefert die Standardeinstellungen für den i-ten Schornsteintyp
GetDefaultDimension	Liefert die Standardeinstellungen für Vermaßungen
GetDefaultDoor	Liefert die Standardeinstellungen für den i-ten Türtyp
GetDefaultGuide	Liefert die Standardeinstellungen für Hilfslinien
GetDefaultHedge	Liefert die Standardeinstellungen für den i-ten Heckentyp
GetDefaultHole	Liefert die Standardeinstellungen für Wandlöcher
GetDefaultLabeling	Liefert die Standardeinstellungen für Beschriftungen
GetDefaultSupport	Liefert die Standardeinstellungen für den i-ten Stützentyp
GetDefaultTerrain	Liefert die Standardeinstellungen für den i-ten Terraintyp
GetDefaultUnterUeberzug	Liefert das Standard-Objekt für Unter- oder Überzüge
GetDefaultWall	Liefert die Standardeinstellungen für den i-ten Wandtyp
GetDesignModeSnapSettings	Holt die Fangeinstellungen für den Einrichtungsmodus
GetDialogData	Wert aus ArCon-Dialogelement lesen
GetDoorName	Liefert den Namen des n-ten Türtyps
GetExternalViewDockMode	Liefert den Dock-Modus einer externen Ansicht
GetFileNameOfLoadingProject	Nur während ein Projekt/Gebäude geladen wird: liefert den Dateinamen des Projektes, aus dem geladen wird.
GetGaubenConstructionRange	Liefert Eckdaten für eine Gaubenkonstruktion
GetGaubenName	Liefert den Namen des n-ten Gaubenyps
GetOneClick	Von ArCon einen Klick Anfordern (liefert Event)
GetOnlySomeBuildingsLoading	Nur während ein Projekt/Gebäude geladen wird: werden einzelne Gebäude geladen (True) oder ein komplettes Projekt (False).
GetPath	Liefert einen Dateipfad (z.B. mit AC_GETPATH_TEXTURES

	den Texturpfad)
GetPfostenBeschreibung	Liefert die Beschreibung eines Treppenpfostens
GetPfostenObject	Liefert den (internen) Objektnamen eines Treppenpfostens
GetPredefinedViewing3D	Liefert Name und Daten eines vordefinierten Betractherstandpunkte
GetRoofWindowName	Liefert den Namen des n-ten Dachfenstertyps
GetStairCaseName	Liefert den Namen des n-ten Treppentyps
GetSupportName	Liefert den Namen des n-ten Stützentyps
GetTerrainHeight	Geländehöhe an Position lesen
GetTypeIDFromDB	Liefert den Typcode eine Tür, eines Fensters oder Dachfensters
GetWindowName	Liefert den Namen des n-ten Fenstertyps
GroupDesignObjects	Faßt mehrere 3D-Objekte zu einer Gruppe zusammen
ImportFolie	Liest eine DXF oder HPGL Folie ein
InvalidateLightSettings	Setzt die DLL-globalen Beleuchtungs/ Hintergrundeinstellungen neu (nur für interne Zwecke)
LoadObjectConstructor	Lädt eine Objektklasse aus einer Datei
LoadObjectDialog	Öffnet den Objekt-laden Dialog
LoadProject	Projekt laden
LoadProjectDialog	Öffnet den Projekt-laden Dialog
LoadTextureDialog	Öffnet den Textur-laden Dialog
NewCeiling	Deckenplattenobjekt erzeugen
NewCeilingOpening	Deckenaussparungsobjekt erzeugen
NewChimney	Schornsteinobjekt erzeugen
NewDimension	Vermaßungsobjekt erzeugen
NewDoor	Türobjekt erzeugen
NewGuide	Hilfslinienobjekt erzeugen
NewHedge	Objekt für Geländeumrandung erzeugen
NewHole	Erzeugt ein neues Wandloch
NewHolePolygon	Erzeugt ein neues Loch-Polygon für Wände u.ä.
NewImage	Bitmapobjekt als 2D-Element erzeugen
NewLabel	Textobjekt als 2D-Element erzeugen
NewLabeling	Beschriftungsobjekt erzeugen
NewLine	Linienobjekt als 2D-Element erzeugen
NewMaterial	Erzeugt ein neues Material
NewObjectConstructor	Erzeugt einen Objekt-Konstruierer
NewObjectTransformerCollection	Erzeugt eine neue Animationsliste
NewPoint2DCollection	Erzeugt eine neue Punktliste
NewPolygon2D	2D-Polygon-Objekt für Dach/Aussparung etc. erzeugen
NewPolygonWendelConstruction	Erzeugt eine neue (leere) Konstruktion für eine Polygonale Wendeltreppe.
NewPrintSettings	Liefert Druckereinstellung (die mit den aktuellen globalen Druckeinstellungen initialisiert werden) für einen späteren Druckvorgang.
NewProject	Objekt für die Definition eines neuen Projektes erzeugen
NewProjectPreview	Liefert Daten für einen Projekt-Preview

NewRoof	Dachobjekt erzeugen
NewRoofConstruction	Liefert eine neue (leere) Dachkonstruktion um ein neues Dach zu erstellen
NewRoofWindow	Dachfenster erzeugen
NewSavePictureSettings	Liefert neue Bildspeichereinstellungen (die mit den aktuellen globalen Einstellungen initialisiert werden).
NewShape	Objekt für Rechteck/Kreis.. als 2D-Element erzeugen
NewSnapObject	Erzeugt ein neues Beschreibungsobjekt für 3D Fanginformation
NewStairCase	Treppenobjekt erzeugen
NewSupport	Stützenobjekt erzeugen
NewTerrain	Geländeobjekt erzeugen
NewTexture	Texturobjekt erzeugen
NewTextureCollection	Erstellt eine neue Texturnamensliste für 3D-Objekte
NewUnterUeberzug	Erzeugt einen neuen Unter- oder Überzug
NewVirtualWall	Erzeugt eine neue virtuelle Wand
NewWall	Wandobjekt erzeugen
NewWindow	Fensterobjekt erzeugen
NotifyOnChange	Meldet eine Überwachung des Objektes auf Änderungen an
ParseDBIDString	Übersetzt einen Datenbank-Dateinamen in seine einzelnen Bestandteile
PlaceTerrain	Gelände plazieren
PredefinedViewing3DCount	Liefert die Anzahl der vordefinierten Betractherstandpunkte
ReadChunk	Daten aus ArCon-Projektdatei lesen
Redraw3DViews	Zeichnet alle 3D Ansichten neu
RemoveAnyMenuItem	Nicht benutzen!
RemoveMenuItem	Menü-Eintrag löschen
RenamePredefinedViewing3D	Benennt einen vordefinierten Betrachterstandpunkt um
RoofWindowName	Liefert den Namen des Dachfenster-Objektes für den angegebenen Dachfenstertyp
RunInProcMacro	Startet nachträglich ein In-Proccess Makro
SaveProject	Projekt speichern
SaveProjectAs	Projekt speichern als
SelectAll3DObjects	Selektiert alle 3D Objekte in der ArCon-Welt
Selected3DObjects	Liefert ein Array mit allen 3D Objekten, die gerade selektiert sind
SetButtonPosition	Setzt die absolute Position eines Buttons (nur während der Konstruktion des Buttons, vor dem ersten SetButtonState!)
SetButtonState	Status von Schaltfläche setzen
SetCompas	Setzt die Daten der Nordrichtungsanzeige
SetConstructionModeSnapSettings	Setzt die Fangeinstellungen für den Konstruktionsmodus
SetDesignModeSnapSettings	Setzt die Fangeinstellungen für den Einrichtungsmodus
SetDialogData	Wert aus ArCon-Dialogelement schreiben
SetExternalViewDockMode	Ändert Dock-Modus und Position einer externen Ansicht
SetExternalViews	Macht Zusatzansichten (z.B. Explorer) sichtbar/unsichtbar
SetInputMode	Aktiviert eine Standard-Funktion in der Was-Leiste

SetObject3DEventMask	Setzt die Ereignismaske für speziellen 3D Objekt Ereignisse, z.B. ACO3D_EVENT_DBLCLK um WorldObject3DDoubleClicked auszulösen.
SetParentWindow	Macht ArCon zum Child-Window eines anderen Fensters oder entfernt es aus einem ehemaligen Parent-Window (parent = 0). Funktioniert nur in speziellen ArCon Versionen.
SetProgressbarSubTitle	Text rechts neben Fortschrittsbalken setzen
SetProgressbarValue	Fortschrittsbalken auf Wert setzen
SetStatusText	Text in ArCon-Statuszeile ausgeben
SetTerrainHeight	Geländehöhe an Position setzen
SetUserPanelState	Ändert den Status eine benutzerdefinierten Knopfleiste
ShowAll	Entspricht Schaltfläche 'alles zeigen'
ShowAllButtonsInPanel	Zeigt oder verbirgt alle Knöpfe in einem Panel. IDs entnehmen Sie bitte der Dokumentation.
ShowButtonByID	Zeigt oder verbirgt einen Knopf. IDs und Sub-IDs entnehmen Sie bitte der Dokumentation.
ShowMenu	Zeigt oder verbirgt ein komplettes Menü
ShowPanel	Zeigt oder verbirgt ein komplettes Panel.
ShowWaitCursor	Zeigt eine Sanduhr als Mauscursor an
SpaceMouseAvailable	Prüft, ob eine 3D Eingabe vorhanden ist
StartDragFromDB	Beginnt einen Drag & Drop-Vorgang aus der Datenbank
StartMe	Verbindung zu ArCon aufbauen
StartMe2	Verbindung zu ArCon aufbauen, aber ArCon optional vorerst verbergen.
StartModalDialog	ArCon für Ausgabe von modalen, nicheigenen Dialogboxen vorbereiten
StartProgressbar	ArCon-Fortschrittsbalken setzen
StopProgressbar	ArCon-Fortschrittsbalken ausschalten
TextureToPicture	Wandelt eine Textur in ein Bild um (auch für Datenbank-Texturnamen)
ThePrintSettings	Liefert die globalen Druckereinstellungen
TheSavePictureSettings	Liefert die globalen Bildspeichereinstellungen
TheWalkSettings	Liefert die globalen Einstellungen zum Durchwandern
TheZoomSettings	Liefert die globalen Zoomeinstellungen
UpdateWindowPos	Informiert ArCon von einer Positionsänderung
VRDeviceAvailable	Prüft, ob eine für die VR-Funktionalität geeignete Grafikkarte installiert ist
Views	Liefert eine Liste aller derzeit offenen Ansichten
WallName	Name des Wandtyps
WindowName	Liefert den Namen des Fenster-Objektes für den angegebenen Fenstertyp
WriteChunk	Daten in ArCon-Projektdatei schreiben

Ereignisse

ButtonChange	Multifunktionsknopf ändert Variante
ChangeNotify	Ein Objekt hat sich geändert
DialogActivation	ArCon Dialog-Erweiterung kommt in den Vordergrund

EndDialog	Wird aufgerufen, wenn ArCon irgend einen Dialog schließt
ExecuteMenuItem	Menübefehl wurde gestartet
ExternalViewsVisibilityChanged	Die Sichtbarkeite eines exteren Views (z.B. des Explorers) hat sich geändert
GotOneClick	Im Arbeitsbereich wurde geklickt
Graphics2DDeleted	Ein 2D Graphikelement wurde gelöscht
Graphics2DDoubleClick	Ein 2D Graphikelement wurde doppelt angeklickt
Graphics2DEndMoving	Ein 2D Graphikelement beendet den Verschiebe/ Skalierungsvorgang
Graphics2DMove	Ein 2D Graphikelement wird skaliert oder verschoben und meldet die aktuelle Mausposition
Graphics2DObjectTransform	Das übergebene Grafikobjekt muß mit der übergebenen 3x3 Matrix transformiert werden, falls es seine relative Stockwerk/Gebäudeposition behalten soll.
Graphics2DSelectionChanged	Die Selektion eines 2D Grafikelementes hat sich geändert
Graphics2DStartMoving	Ein 2D Graphikelement beginnt einen Skalierungs- oder Verschiebevorgang
Graphics2DStoryHeightsChanged	Das übergebene Grafikobjekt ist von Änderungen in Stockwerkhöhen betroffen
HowInput	Bei automatischen Wie-Tools: Maus wurde geklickt oder Eingabe betätigt
HowMove	Bei automatischen Wie-Tools: Maus wurde bewegt
HowSnap	Ein automatischer Variantenschalter fängt Mauskoordinaten
HowStatusLineRedraw	Ein automatischer Variantenschalter zeigt Koordinaten an
InputModeChanged	Ein neues Eingabwerkzeug der Was-Leiste wurde aktiviert
LoadBuilding	ArCon lädt einzelnes Gebäude
LoadComplete	Projekt laden wurde beedet
LoadDialog	Wird aufgerufen, wenn ArCon irgend einen Dialog öffnet
LoadStart	Projekt laden beginnt
NewCurrentProject	Ein neues Projekt wurde erzeugt oder ein anderes geladen, CurrentProject hat einen neuen Wert
Object3DDoubleClicked	Ein 3D Objekt mit 2D Ersatzdarstellung wurde angeklickt
Object3DInserted	Ein 3D Objekt wurde in die ArCon-Welt eingefügt
ProgramExit	ArCon wurde beendet
ProjectChange	Einstellungen des aktuellen Projektes, z.B. der Maßstab, wurden geändert
ProjectClosed	Das aktuelle Projekt wird geschlossen
RoofDoubleClicked	Ein selbst konstruiertes Dach wurde doppelt angeclickt
SaveBuilding	Ein Gebäude wird gespeichert
SaveDialogDefaults	'Als Standard'-Schaltfläche wurde in ArCon-Dialogbox betätigt
SaveStart	Projekt speichern beginnt
TextureDropped	Eine Textur wird vom Benutzer geändert
UserPanelAdjustSize	Eine selbstdefinierte Knopfleiste verändert die Größe
UserPanelPosChanged	Eine selbstdefinierte Knopfleiste wird positioniert (Position in Client-Koordinaten)
UserPanelRightClicked	Auf einer selbstdefinierten Knopfleiste wurde rechts geklickt
WorldObject3DDoubleClicked	Ein 3D Objekt in der ArCon Welt wurde doppelt geklickt

WorldObject3DMaterialDropped	Ein 3D Objekt in der ArCon Welt bekommt per Drag & Drop ein neues Material
WorldObject3DMoved	Ein 3D Objekt in der ArCon Welt wurde verschoben
WorldObject3DTextureDropped	Ein 3D Objekt in der ArCon Welt bekommt per Drag & Drop eine neue Textur

ArCon.AllFloorsVisible

Deklaration:

Boolean

HRESULT get_AllFloorsVisible(VARIANT_BOOL* retVal);
HRESULT put_AllFloorsVisible(VARIANT_BOOL newVal);

Siehe auch: ArCon Übersicht

Durch Zuweisung des Wertes “ Wahr” an dieses Attribut schalten Sie den Menüpunkt Geschoß/ Anzeige auf die Option “ Anzeige aller Geschosse” . Mit dem Wert “ Falsch” stellen Sie die Option “ Anzeige des aktuellen Geschosses” ein.

ArCon.ArConDirectory

Deklaration:

String (nur lesbar)

HRESULT get_ArConDirectory(BSTR* retVal);

Siehe auch: ArCon Übersicht

Gibt das Verzeichnis an, indem das ArCon Programm installiert ist, mit dem das Active-X Control derzeit kommuniziert. In diesem Verzeichnis befindet sich die ArCon.INI Datei, der Ihr Programm alle übrigen relevanten Pfadangaben entnehmen kann.

ArCon.ArConWindowHandle

Deklaration:

long (nur lesbar)

HRESULT get_ArConWindowHandle(long* retVal);

Siehe auch: ArCon Übersicht

ArCon.AutoLoaded

Deklaration:

Boolean (nur lesbar)

HRESULT get_AutoLoaded(VARIANT_BOOL* retVal);

Siehe auch: ArCon Übersicht

Dieses Attribut ist “ Wahr” , wenn Ihr Makro automatisch beim Start von ArCon durch einen entsprechenden Eintrag in der Makros.INI Datei gestartet wurde.

ArCon.AvailableChimneys

Deklaration:

long (nur lesbar)

HRESULT get_AvailableChimneys(long* retVal);

Siehe auch: ArCon Übersicht

ArCon.AvailableDoors

Deklaration:

long (nur lesbar)

HRESULT get_AvailableDoors(long* retVal);

Siehe auch: ArCon Übersicht

Gibt die Anzahl der verfügbaren Türtypen an. Dieser Wert hängt von der ArCon Installation ab und kann durch Installation zusätzlicher Türen auch nachträglich geändert werden. Typcodes beginnen mit dem Wert 0, daher ist der maximal zulässige Wert AvailableDoors-1.

Den Namen eines Türtyps erhalten Sie mit der Funktion GetDoorName.

ArCon.AvailableGauben

Deklaration:

long (nur lesbar)

HRESULT get_AvailableGauben(long* retVal);

Siehe auch: ArCon Übersicht

ArCon.AvailableRoofWindows

Deklaration:

long (nur lesbar)

HRESULT get_AvailableRoofWindows(long* retVal);

Siehe auch: ArCon Übersicht

ArCon.AvailableStairCases

Deklaration:

long (nur lesbar)

HRESULT get_AvailableStairCases(long* retVal);

Siehe auch: ArCon Übersicht

Gibt die Anzahl der verfügbaren Treppentypen an. Dieser Wert kann erst zur Laufzeit bestimmt werden, da er von der Installation abhängt. Gültige Typcodes sind 0 bis AvailableStaircases-1.

Um den Namen eines Treppentyps zu bestimmen, benutzen Sie die Funktion GetStairCaseName.

ArCon.AvailableSupports

Deklaration:

long (nur lesbar)

HRESULT get_AvailableSupports(long* retVal);

Siehe auch: ArCon Übersicht

ArCon.AvailableWalls

Deklaration:

long (nur lesbar)

HRESULT get_AvailableWalls(long* retVal);

Siehe auch: ArCon Übersicht

ArCon.AvailableWindows

Deklaration:

long (nur lesbar)

HRESULT get_AvailableWindows(long* retVal);

Siehe auch: ArCon Übersicht

Gibt die Anzahl der verfügbaren Fenstertypen an. Typcodes zwischen 0 und AvailableWindows-1 sind gültig.

Um den Namen eines Fenstertyps zu ermitteln, benutzen Sie die Funktion GetWindowName.

ArCon.Buildings

Deklaration:

BuildingCollection (nur lesbar)

HRESULT get_Buildings(IBuildingCollection** retVal);

Siehe auch: ArCon Übersicht, BuildingCollection

Eine Liste aller Gebäude im Projekt.

ArCon.ConstructedRoofWindows

Deklaration:

long (nur lesbar)

HRESULT get_ConstructedRoofWindows(long* retVal);

Siehe auch: ArCon Übersicht

ArCon.ConstructedWindows

Deklaration:

long (nur lesbar)

HRESULT get_ConstructedWindows(long* retVal);

Siehe auch: ArCon Übersicht

ArCon unterscheidet zwischen konstruierten und geladenen Fenstertypen. Die Typcodes 0 bis ConstructedWindows-1 geben konstruierbare Fenster an. Alle übrigen (gültigen) Typcodes sind geladene Fenster. Die insgesamt verfügbare Anzahl von Fenstertypen erhalten Sie mit der Eigenschaft AvailableWindows. Um einen Nbamen für einen Fenstertyp zu ermitteln bentuzen Sie die Funktion GetWindowName.

ArCon.CurrentBuilding

Deklaration:

Building (nur lesbar)

HRESULT get_CurrentBuilding(IBuilding** retVal);

Siehe auch: ArCon Übersicht, Building

Ein Verweis auf das aktuelle Gebäude.

ArCon Projekte besitzen immer ein aktuelles Gebäude, auch wenn das Projekt noch ganz leer ist. Sollte allerdings gerade kein Projekt geladen sein (ArCon.Mode = AC_NoMode), existiert kein CurrentBuilding - weshalb der Wert auf Nothing (bzw. Nil oder NULL) gesetzt wird.

ArCon.CurrentProject

Deklaration:

Project (nur lesbar)

HRESULT get_CurrentProject(IProject** retVal);

Siehe auch: ArCon Übersicht, Project

Eine Beschreibung der aktuellen Projekteinstellungen.

ArCon.CurrentStory

Deklaration:

Story

HRESULT get_CurrentStory(IStory** retVal);
HRESULT put_CurrentStory(IStory* newVal);

Siehe auch: ArCon Übersicht, Story

Ein Verweis auf das aktuelle Stockwerk.

Durch Zuweisen dieser Eigenschaft wechseln Sie das aktuelle Stockwerk in ArCon.

Sofern ein Projekt geladen ist (ArCon.Mode <> AC_NoMode) gibt es immer ein aktuelles Stockwerk. Andernfalls ist diese Eigenschaft Nothing (Nil, NULL).

ArCon.Cuts

Deklaration:

CutCollection (nur lesbar)

HRESULT get_Cuts(ICutCollection** retVal);

Siehe auch: ArCon Übersicht, CutCollection

ArCon.DesignObjectConstructors

Deklaration:

ObjectConstructorCollection (nur lesbar)

HRESULT get_DesignObjectConstructors
(IObjectConstructorCollection** retVal);

Siehe auch: ArCon Übersicht, ObjectConstructorCollection

Die Liste aller Klassen von 3D-Objekten im gesamten Projekt.

ArCon.DesignObjects

Deklaration:

Object3DCollection (nur lesbar)

HRESULT get_DesignObjects(IObject3DCollection** retVal);

Siehe auch: ArCon Übersicht, Object3DCollection

Die Liste aller Instanzen von 3D-Objekten im gesamten Projekt.

ArCon.Dimensions

Deklaration:

DimensionCollection (nur lesbar)

HRESULT get_Dimensions(IDimensionCollection** retVal);

Siehe auch: ArCon Übersicht, DimensionCollection

Eine Liste aller Vermaßungen in der gesamten Planung.

ArCon.GlobalTerrain

Deklaration:

Terrain (nur lesbar)

HRESULT get_GlobalTerrain(ITerrain** retVal);

Siehe auch: ArCon Übersicht, Terrain

Ein Verweis auf das globale Umgebungsgelände., die Wurzel des Gelände-Baumes.

ArCon.Graphics2D

Deklaration:

Graphics2DCollection (nur lesbar)

HRESULT get_Graphics2D(IGraphics2DCollection** retVal);

Siehe auch: ArCon Übersicht, Graphics2DCollection

Eine Liste mit 2D Grafikelementen. Alle in dieser Liste eingetragenen Elemente werden ständig dargestellt, im Gegensatz zu den temporären Grafikobjekten in der RunningTool.Graphics2D Liste.

Wenn die Sichtbarkeit eines 2D Grafikelementes von der Sichtbarkeit eines Stockwerkes abhängen soll, oder das Element nur in einem Schnitt gezeigt werden soll, verwenden Sie statt dieser Liste die Listen Story.Graphics2D beziehungsweise Cut.Graphics2D.

Siehe auch: Automatische Variantenschalte

ArCon.Guides

Deklaration:

GuideCollection (nur lesbar)

HRESULT get_Guides(IGuideCollection** retVal);

Siehe auch: ArCon Übersicht, GuideCollection

Eine Liste mit allen Hilfslinien der Planung.

ArCon.InternalMode

Deklaration:

long (nur lesbar)

HRESULT get_InternalMode(long* retVal);

Siehe auch: ArCon Übersicht

Einige Befehle in ArCon dürfen nur von priveligierten Makros ausgeführt werden. Diese Eigenschaft ist “ Wahr” , wenn Ihr Makro ein solches Makro ist. Zum Beispiel sind ArCon startenden Makros einer ArCon-OEM Version priveligiert.

Sie können durch Abfragen dieses Wertes Fehler vermeiden und so Ihr Makro sowohl in der “ normalen” Version als auch in der Spezialversion lauffähig machen.

ArCon.Labelings

Deklaration:

LabelingCollection (nur lesbar)

HRESULT get_Labelings(ILabelingCollection** retVal);

Siehe auch: ArCon Übersicht, LabelingCollection

Eine Liste aller Beschriftungen der Planung.

ArCon.Mode

Deklaration:

long

HRESULT get_Mode(long* retVal);
HRESULT put_Mode(long newVal);

Siehe auch: ArCon Übersicht

Gibt den aktuellen Darstellungsmodus an und kann zwischen 2D und 3D Ansicht wechseln. Mögliche Werte finden Sie im Kapitel Display Modes.

ArCon.MultiUserMode

Deklaration:

long

HRESULT get_MultiUserMode(long* retVal);
HRESULT put_MultiUserMode(long newVal);

Siehe auch: ArCon Übersicht

Verschiedene Bits, die das Multi-User beziehungsweise Multi-Makro Verhalten, die Interaktion mit dem Benutzer und die Bildschirmaktualisierung von ArCon steuern. Die möglichen Werte und Ihre Bedeutung können Sie dem Kapitel Multi User Verhalten entnehmen.

Der Name dieser Eigenschaft ist historisch bedingt und trift nicht mehr vollständig zu, da diese Eigenschaft weit mehr als das Multi-User Verhalten steuert.

ArCon.ProgramName

Deklaration:

String (nur lesbar)

HRESULT get_ProgramName(BSTR* retVal);

Siehe auch: ArCon Übersicht

ArCon.ProgramProperties

Deklaration:

long (nur lesbar)

HRESULT get_ProgramProperties(long* retVal);

Siehe auch: ArCon Übersicht

ArCon.ProgramTypeID

Deklaration:

long (nur lesbar)

HRESULT get_ProgramTypeID(long* retVal);

Siehe auch: ArCon Übersicht

ArCon.ProgramVersion

Deklaration:

long (nur lesbar)

HRESULT get_ProgramVersion(long* retVal);

Siehe auch: ArCon Übersicht

ArCon.Running

Deklaration:

Boolean (nur lesbar)

HRESULT get_Running(VARIANT_BOOL* retVal);

Siehe auch: ArCon Übersicht

“ Wahr” wenn Ihr Makro läuft und eine Verbindung zu ArCon besteht. “ Falsch” vor dem Aufruf von StartMe oder nach dem Verbindungsabbau, zum Beispiel wenn ArCon.EXE beendet wurde.

Wenn diese Eigenschaft den Wert “ Falsch” liefert, sollten Sie keine Befehle mehr an ArCon geben. Eine Auswertung dieses Wertes hilft in seltenen Situation im Verlauf des gleichzeitigen Beendes von ArCon und eines Makros. Nur wenn Sie in solchen Situation auf Probleme stoßen, sollten Sie dieses Attribut verweden.

ArCon.RunningTool

Deklaration:

ToolData (nur lesbar)

HRESULT get_RunningTool(IToolData** retVal);

Siehe auch: ArCon Übersicht, ToolData

Liefert das ToolData Objekt, falls gerade ein von Ihnen definiertes Werkzeug aktiv ist.

ArCon.StartupMenuID

Deklaration:

long (nur lesbar)

HRESULT get_StartupMenuID(long* retVal);

Siehe auch: ArCon Übersicht

Falls Ihr Makro über einen Eintrag im Menü “ Makros” gestartet wurde, liefert diese Eigenschaft die Kennzahl des Menüpunktes, der für Ihr Makro erzeugt wurde. Andernfalls liefert diese Eigenschaft den Wert 0.

Sie können mit diesem Wert z.B. den zugehörigen Menüpunkt im “ Makro” Menü sperren, um ein mehrmaliges Starten des Makros zu verhindern. Beachten Sie aber, daß dies nicht sicher ausreicht, da der (bösartige) Benutzer das Makro dennoch durch Doppelklick auf die EXE Datei im Makro-Verzeichnis nochmals starten kann.

ArCon.State

Deklaration:

long

HRESULT get_State(long* retVal);
HRESULT put_State(long newVal);

Siehe auch: ArCon Übersicht

Eine Bitmaske zur Steuerung der anzuzeigenden Details und Hilfsmittel. Die möglichen Bitwerte finden Sie im Kaptiel State Bits.

ArCon.Terrains

Deklaration:

TerrainCollection (nur lesbar)

HRESULT get_Terrains(ITerrainCollection** retVal);

Siehe auch: ArCon Übersicht, TerrainCollection

Eine Liste aller Gelände in der Planung.

ArCon.ActiveExternalViews

Deklaration:

```
Function ActiveExternalViews() as long
```

```
HRESULT ActiveExternalViews(long* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.ActiveView

Deklaration:

```
Function ActiveView() as View
```

```
HRESULT ActiveView(IView** retVal);
```

Siehe auch: ArCon Übersicht, View

ArCon.AddPredefinedViewing3D

Deklaration:

```
Function AddPredefinedViewing3D(ByVal Description as String, ByVal VRPx as Single, ByVal VRPy as Single, ByVal VRPz as Single, ByVal EyeX as Single, ByVal EyeY as Single, ByVal EyeZ as Single, ByVal tanViewAngleHalf as Single) as long
```

```
HRESULT AddPredefinedViewing3D(BSTR Description, float VRPx, float VRPy, float VRPz, float EyeX, float EyeY, float EyeZ, float tanViewAngleHalf, long* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.AttachDialog

Deklaration:

```
Sub AttachDialog(ByVal ArConDlg as long, ByVal hWnd as
long, ByVal icon as long, ByVal helpContext as long, ByVal
isActive as Boolean, ByVal exclusive as Boolean)
```

```
HRESULT AttachDialog(long ArConDlg, long hWnd, long
icon, long helpContext, VARIANT_BOOL isActive,
VARIANT_BOOL exclusive);
```

Siehe auch: ArCon Übersicht

Mit dieser Funktion können Sie eigene Dialoge zu bereits existierenden Dialogen in ArCon hinzufügen. Ein Beispiel für die Anwendung dieser Funktion finden Sie im Kapitel Vorhandene Dialoge erweitern.

Die Parameter sind:

ArConDlg	ein Dialog-“ Token” , den Sie vom Ereignis OpenDialog erhalten.
hWnd	Die Fensternummer Ihres Dialoges. ArCon übernimmt dieses Fenster in seinen Dialog.
Icon	Dialogerweiterungen werden durch ein Icon identifziert. Falls Ihr Programm nicht das Applikationsicon benutzen möchte, geben Sie hier ein Icon-Handle eines anderen Icons an.
helpContext	Nummer des Hilfekontextes in Ihrer Hilfedatei, der bei einem Druck auf den “ Hilfe” Knopf des Dialoges angezeigt werden soll, falls gerade Ihr Dialog aktiv ist.
IsActive	Wenn Sie diesen Parameter auf “ Wahr” setzen, wird Ihr Dialog sofort aktiviert.
Exclusive	Durch den Wert “ Wahr” werden alle übrigen Reiter im Dialog gesperrt, der Benutzer kann nur mit Ihrer Erweiterung des Dialoges arbeiten. Sie können dennoch Daten in die gesperrten Dialogbereiche schreiben.

ArCon.AvailableExternalViews

Deklaration:

```
Function AvailableExternalViews() as long
```

```
HRESULT AvailableExternalViews(long* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.AvailablePfostenObjects

Deklaration:

```
Function AvailablePfostenObjects() as long
```

```
HRESULT AvailablePfostenObjects(long* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.ChangePredefinedViewing3D

Deklaration:

```
Function ChangePredefinedViewing3D(ByVal index
as long, ByVal VRPx as Single, ByVal VRPy as
Single, ByVal VRPz as Single, ByVal EyeX as Single,
ByVal EyeY as Single, ByVal EyeZ as Single, ByVal
tanViewAngleHalf as Single) as Boolean
```

```
HRESULT ChangePredefinedViewing3D(long index,
float VRPx, float VRPy, float VRPz, float EyeX, float
EyeY, float EyeZ, float tanViewAngleHalf,
VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.ChangeTypeNotifyMask

Deklaration:

```
Function ChangeTypeNotifyMask(ByVal typeID as
AC_ArCon_Objekt_Typen, ByVal add as long, ByVal
remove as long) as long
```

```
HRESULT ChangeTypeNotifyMask
(AC_ArCon_Objekt_Typen typeID, long add, long
remove, long *oldMask, long* retVal);
```

Siehe auch: ArCon Übersicht

Unabhängig von konkreten Instanzen eines Typs können Sie sich über allgemeine Ereignisse, z. B. die Erzeugung eines neuen Objektes dieses Typs, informieren lassen. In der Regel erhalten Sie die Nachrichten über das Ereignis ChangeNotify, einzige Ausnahme ist der Typ Object3D, dessen AC_CHANGE_Inserted Event Sie über das Ereignis Object3Dinserted erhalten.

Bitmasken für die möglichen Ereignisse finden Sie unter 3D Objekt-Ereignisse. Die Konstanten für die einzelnen Objekttypen finden Sie unter Typkonstanten.

Parameter sind:

typeID	Kodierung des Typs, von dem Sie Ereignisse erhalten möchten
add	Bits für zusätzlich zu aktivierende Ereignisse
remove	Bits für nicht mehr gewünschte Ereignisse
oldMask	liefert den alten Wert der Ereignismaske zurück

Die Funktion liefert als Rückgabewert die neue Ereignismaske.

ArCon.Clear3DObjectSelection

Deklaration:

```
Function Clear3DObjectSelection() as Boolean
```

```
HRESULT Clear3DObjectSelection(VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.CloseDialog

Deklaration:

```
Function CloseDialog(ByVal ArConDlgToken as long, ByVal withOK
as Boolean) as Boolean
```

```
HRESULT CloseDialog(long ArConDlgToken, VARIANT_BOOL
withOK, VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.CloseProject

Deklaration:

```
Function CloseProject() as Boolean
```

```
HRESULT CloseProject(VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

Schließt das aktuelle Projekt. Ob der Benutzer zum Speichern aufgefordert wird, hängt von der Einstellung der ArCon-Installation ab.

ArCon.CreateBuilding

Deklaration:

```
Function CreateBuilding(ByVal initialFloorType as short) as Building
```

```
HRESULT CreateBuilding(short initialFloorType, IBuilding** retVal)
;
```

Siehe auch: ArCon Übersicht, Building

Erzeugt ein neues Gebäude mit einem Stockwerk vom angegebenen Typ. Symbolische Namen für die Typkodierung finden Sie im Kapitel Stockwerk Typen.

ArCon.CreateMenuItem

Deklaration:

```
Function CreateMenuItem(ByVal parentMenu as long, ByVal
beforeId as long, ByVal name as String, ByVal description as
String, ByVal modes as long, ByVal picture as long, ByVal helpId as
long) as long
```

```
HRESULT CreateMenuItem(long parentMenu, long beforeId, BSTR
name, BSTR description, long modes, long picture, long helpId,
long* retVal);
```

Siehe auch: ArCon Übersicht

Erzeugt einen neuen Menüeintrag.

Beispiele für die Programmierung mit Menüs finden Sie im Kapitel Eigene Menüeinträge.

Die Parameter sind:

parentMenu	Die Kennzahl des Menüs, in das Sie einen Eintrag einfügen möchten oder 0, falls es sich um ein eigenes Menü handelt.
BeforeID	Die Kennzahl des Menüpunktes, vor den Ihr Eintrag eingefügt werden soll. Geben Sie -1 an, wenn der neue Menüpunkt an das Ende des Menüs angefügt werden soll. Sie können auch eine absolute Position im Menü angeben.
Name	Der Name, der im Menü erscheint. Ein “ &” Zeichen vor einem Buchstaben macht diesen zum “ Hotkey” und unterstreicht ihn. Ein Tabulator-Zeichen (Chr$ (9)) trennt Text und Accelerator-Taste.
Description	Die Beschreibung in der Statuszeile
modes	Eine Bitmenge von ArCon Modi, in denen dieser Menüeintrag existieren soll
picture	Ein Icon-Handle oder 0, falls das Applikationsicon Ihres Makros verwendet werden soll, um diesen Menüeintrag zu kennzeichnen.
HelpID	Die Themen-Kennzahl in Ihrer Hilfedatei, unter der dieser Menüpunkt beschrieben ist.

ArCon.CreateMenuSeparator

Deklaration:

```
Function CreateMenuSeparator(ByVal menu as long,
ByVal before as long, ByVal modes as long) as long
```

```
HRESULT CreateMenuSeparator(long menu, long
before, long modes, long* retVal);
```

Siehe auch: ArCon Übersicht

Erzeugt einen Trennstrich im Menü. Vergleiche auch CreateMenuItem.

Beispiele zur Programmierung mit Menüs finden Sie im Kapitel Eigene Menüeinträge.

Der Trennstrich erhält genau wie ein Menüpunkt eine Kennung, sodaß Sie Ihn später wieder löschen können.

Die Parameter sind:

menu	Die Kennung des Menüs, in das der Trennstrich eingefügt werden soll.
Before	Kennung des Menüpunktes, vor dem der Strich erzeugt wird
modes	Eine Bitmaske mit ArCon Modi, in denen der Trennstrich vorhanden sein soll.

ArCon.CreateProject

Deklaration:

```
Sub CreateProject(ByVal aProject as Project)
```

```
HRESULT CreateProject(IProject* aProject);
```

Siehe auch: ArCon Übersicht, Project

Erzeugt ein neues Projekt gemäß Ihren Vorgaben.

Parameter:

Project	Beschreibt das neue Projekt. Um ein Standard-Projekt zu erzeugen, übergeben Sie hier das unveränderte Projekt-Objekt, das Sie von einem Aufruf der Funktion NewProject erhalten haben..

ArCon.CreateUserPanel

Deklaration:

```
Function CreateUserPanel(ByVal position
as long, ByVal helpId as long, ByVal visible
as Boolean, ByVal enabled as Boolean) as
long
```

```
HRESULT CreateUserPanel(long position,
long helpId, VARIANT_BOOL visible,
VARIANT_BOOL enabled, long* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.DBIDToString

Deklaration:

```
Function DBIDToString(ByVal dbID as
long, ByVal objId as long, ByVal guid as
String, ByVal objType as
AC_ArCon_DB_Types) as String
```

```
HRESULT DBIDToString(long dbID, long
objId, BSTR guid, AC_ArCon_DB_Types
objType, BSTR* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.DefButton

Deklaration:

```
Function DefButton(ByVal pic as VARIANT,
ByVal Info as long, ByVal str as String,
ByVal helpId as long) as long
```

```
HRESULT DefButton(VARIANT pic, long
Info, BSTR str, long helpId, long* retVal);
```

Siehe auch: ArCon Übersicht

Definiert eine Schaltfläche. Beispiele für die Programmierung solcher Knöpfe finden Sie im Kapitel ” Eigene Schaltflächen” .

Als Rückgabewert erhalten Sie eine eindeutige ID für Ihren neu definierten Knopf oder 0, wenn ein Fehler in den Parametern vorlag.

Die Parameter sind:

pic	die Bitmap, die der Knopf darstellt. ArCon verwendet in der Regel 20x20 Pixel große Symbole. Um eine Bitmap in ein VARIANT Objekt zu verpacken, benutzen Sie die Hilfsfunktion ArConPictureFromBitmap aus der Bibliothek MakroUtil.DLL.
Info	ein Button-Info, Konstanten dafür finden Sie im Kapitel “ Button Info” .
Str	die Beschreibung des Knopfes für die Statuszeile. Optional kann eine Kurzbeschreibung für die Soforthilfe durch “ |” getrennt folgen.
HelpID	die Kennzahl der Beschreibung des Knopfes in Ihrer Hilfedatei

ArCon.DefButtonRes

Deklaration:

```
Function DefButtonRes(ByVal id as long, ByVal
moduleFileName as String, ByVal Info as long, ByVal str
as String, ByVal helpId as long) as long
```

```
HRESULT DefButtonRes(long id, BSTR moduleFileName,
long Info, BSTR str, long helpId, long* retVal);
```

Siehe auch: ArCon Übersicht

Ähnlich wie DefButton, nur wird hier die Bitmap des Knopfes aus Resourcen geladen.

Die Parameter sind:

id	Kennzahl der Bitmap-Resource.
moduleFileName	Modul-Name Ihres Makros oder der DLL, aus der die Resource für das Bitmap entnommen werden soll.
Info	ein Button-Info, Konstanten dafür finden Sie im Kapitel “ Button Info” .
Str	die Beschreibung des Knopfes für die Statuszeile. Optional kann eine Kurzbeschreibung für die Soforthilfe durch “ |” getrennt folgen.
HelpID	die Kennzahl der Beschreibung des Knopfes in Ihrer Hilfedatei

Achtung: ein Fehler in Windows 95 verhindet unter bestimmten Umständen, daß ArCon Ihre Bitmap-Resourcen laden kann. Ersatzweise erscheint dann ein Fragezeichensymbol. Microsoft hat diesen Fehler bestätigt, konnte aber keinen Workaround angeben. Wir empfehlen daher, die . ..Res Varianten der Button-Funktionen nicht zu benutzen, wenn Ihr Makro auch unter Windows 95 lauffähig sein soll.

ArCon.DefHowButton

Deklaration:

```
Function DefHowButton(ByVal
MultiButtonVariantID as long, ByVal pic as
VARIANT, ByVal Info as long, ByVal str as String,
ByVal helpId as long) as long
```

```
HRESULT DefHowButton(long
MultiButtonVariantID, VARIANT pic, long Info,
BSTR str, long helpId, long* retVal);
```

Siehe auch: ArCon Übersicht

Definiert einen automatischen Variantenschalter. Beispiele für die Programmierung solcher Knöpfe finden Sie im Kapitel " Automatische Variantenschalter" .

Als Rückgabewert erhalten Sie eine eindeutige ID für Ihren neu definierten Button - oder 0 im Fehlerfall.

Die Parameter sind:

MultiButton-VariantID	Kennzahl der Variante des steuernden Schalters in der " Was" -Leiste, dem dieser Knopf zugeordnet ist
pic	die Bitmap, die der Knopf darstellt. ArCon verwendet in der Regel 20x20 Pixel große Symbole. Um eine Bitmap in ein VARIANT Objekt zu verpacken, benutzen Sie die Hilfsfunktion ArConPictureFromBitmap aus der Bibliothek MakroUtil.DLL.
Info	ein Button-Info, Konstanten dafür finden Sie im Kapitel " Button Info" CONST_ButtonInfo.
Str	die Beschreibung des Knopfes für die Statuszeile. Optional kann eine Kurzbeschreibung für die Soforthilfe durch " |" getrennt folgen.
HelpID	die Kennzahl der Beschreibung des Knopfes in Ihrer Hilfedatei

ArCon.DefHowButtonRes

Deklaration:

```
Function DefHowButtonRes(ByVal
MultiButtonVariantID as long, ByVal id as long,
ByVal moduleFileName as String, ByVal Info
as long, ByVal str as String, ByVal helpId as
long) as long
```

```
HRESULT DefHowButtonRes(long
MultiButtonVariantID, long id, BSTR
moduleFileName, long Info, BSTR str, long
helpId, long* retVal);
```

Siehe auch: ArCon Übersicht

Ähnlich wie DefHowButton, nur wird hier die Bitmap des Knopfes aus Resourcen geladen.

Als Rückgabewert erhalten Sie eine eindeutige ID Ihres neu definierten Knopfes - oder im Fehlerfall 0.

Die Parameter sind:

MultiButton-VariantID	Kennzahl der Variante des steuernden Schalters in der “ Was” -Leiste, dem dieser Knopf zugeordnet ist
id	Kennzahl der Bitmap-Resource.
moduleFileName	Modulname Ihres Makros oder der DLL, aus der die Resourcen entnommen werden sollen.
Info	ein Button-Info, Konstanten dafür finden Sie im Kapitel “ Button Info” CONST_ButtonInfo.
Str	die Beschreibung des Knopfes für die Statuszeile. Optional kann eine Kurzbeschreibung für die Soforthilfe durch “ |” getrennt folgen.
HelpID	die Kennzahl der Beschreibung des Knopfes in Ihrer Hilfedatei

Achtung: ein Fehler in Windows 95 verhindet unter bestimmten Umständen, daß ArCon Ihre Bitmap-Resourcen laden kann. Ersatzweise erscheint dann ein Fragezeichensymbol. Microsoft hat diesen Fehler bestätigt, konnte aber keinen Workaround angeben. Wir empfehlen daher, die . ..Res Varianten der Button-Funktionen nicht zu benutzen, wenn Ihr Makro auch unter Windows 95 lauffähig sein soll.

ArCon.DefMultiButton

Deklaration:

```
Function DefMultiButton(ByVal Info as long) as long
```

```
HRESULT DefMultiButton(long Info, long* retVal);
```

Siehe auch: ArCon Übersicht

Definiert einen Variantenschalter. Beispiele für die Programmierung solcher Knöpfe finden Sie im Kapitel ” Eigene Schaltflächen” .

Ein so erzeugter Knopf ist noch nicht sichtbar. Zunächst müssen Sie Varianten hinzufügen und abschließend mit der Methode SetButtonState den Knopf sichtbar schalten.

Als Rückgabewert erhalten Sie eine eindeutige Kennung für den Variantenschalter, die Sie zum hinzufügen von Varianten zu diesem Schalter benutzen Können.

Die Parameter sind:

Info	ein Button-Info, Konstanten dafür finden Sie im Kapitel “ Button Info” .

ArCon.DefMultiButtonVariant

Deklaration:

```
Function DefMultiButtonVariant(ByVal MultiButtonID as
long, ByVal pic as VARIANT, ByVal str as String, ByVal
helpId as long) as long
```

```
HRESULT DefMultiButtonVariant(long MultiButtonID,
VARIANT pic, BSTR str, long helpId, long* retVal);
```

Siehe auch: ArCon Übersicht

Definiert eine Variante für einen Variantenschalter. Beispiele für die Programmierung solcher Knöpfe finden Sie im Kapitel ” Eigene Schaltflächen” .

Als Rückgabewert erhalten Sie eine eindeutige Kennung des neuen Knopfes - oder 0 im Fehlerfall.

Die Parameter sind:

Multi-ButtonID	Die Kennzahl des Variantenschalters, die von der Methode DefMultiButton zurückgeliefert wurde.
pic	Die Bitmap, die der Knopf darstellt. ArCon verwendet in der Regel 20x20 Pixel große Symbole. Um eine Bitmap in ein VARIANT Objekt zu verpacken, benutzen Sie die Hilfsfunktion ArConPictureFromBitmap aus der Bibliothek MakroUtil.DLL.
Str	die Beschreibung des Knopfes für die Statuszeile. Optional kann eine Kurzbeschreibung für die Soforthilfe durch “ |” getrennt folgen.
HelpID	die Kennzahl der Beschreibung des Knopfes in Ihrer Hilfedatei

ArCon.DefMultiButtonVariantRes

Deklaration:

```
Function DefMultiButtonVariantRes(ByVal
MultiButtonID as long, ByVal id as long, ByVal
moduleFileName as String, ByVal str as String,
ByVal helpId as long) as long
```

```
HRESULT DefMultiButtonVariantRes(long
MultiButtonID, long id, BSTR moduleFileName,
BSTR str, long helpId, long* retVal);
```

Siehe auch: ArCon Übersicht

Definiert eine Variante für einen Variantenschalter. Beispiele für die Programmierung solcher Knöpfe finden Sie im Kapitel " Eigene Schaltflächen" .

Als Rückgabewert erhalten Sie eine eindeutige ID für Ihren neu definierten Knopf oder 0, falls die Parameter ungültig waren.

Die Parameter sind:

Multi-ButtonID	Die Kennzahl des Variantenschalters, die von der Methode DefMultiButton zurückgeliefert wurde.
id	Kennzahl der Bitmap-Resource.
hInst	Instance-Handle der DLL, die die Resourcen enthält, oder Ihres Makros. Wenn Sie kein Instance-Handle kennen, übergeben Sie hier eine 0, es werden dann die Resourcen Ihres Makro-Hauptmoduls benutzt.
Str	die Beschreibung des Knopfes für die Statuszeile. Optional kann eine Kurzbeschreibung für die Soforthilfe durch " |" getrennt folgen.
HelpID	die Kennzahl der Beschreibung des Knopfes in Ihrer Hilfedatei

Achtung: ein Fehler in Windows 95 verhindet unter bestimmten Umständen, daß ArCon Ihre Bitmap-Resourcen laden kann. Ersatzweise erscheint dann ein Fragezeichensymbol. Microsoft hat diesen Fehler bestätigt, konnte aber keinen Workaround angeben. Wir empfehlen daher, die . ..Res Varianten der Button-Funktionen nicht zu benutzen, wenn Ihr Makro auch unter Windows 95 lauffähig sein soll.

ArCon.DeleteButton

Deklaration:

```
Function DeleteButton(ByVal btnId as long) as Boolean
```

```
HRESULT DeleteButton(long btnId, VARIANT_BOOL*
retVal);
```

Siehe auch: ArCon Übersicht

Löscht eine selbstdefinierte Schaltfläche.

Sie können nur “ ganze Knöpfe” löschen, nicht aber einzelne Varianten eines Variantenschalters - diese werden automatisch gelöscht, wenn der Variantenschalter gelöscht wird.

Der Rückgabewert gibt an, ob der Button erfolgreich entfernt werden konnte.

Wenn Ihr Makro beendet wird, bevor ArCon beendet wird, werden auch alle von diesem Makro erzeugten Schaltflächen gelöscht - da anschließend niemand mehr auf die Benutzung dieser Buttons reagieren könnte.

Die Parameter sind:

btnId	Die Kennung der Schaltfläche

ArCon.DeletePredefinedViewing3D

Deklaration:

```
Function DeletePredefinedViewing3D(ByVal
index as long) as Boolean
```

```
HRESULT DeletePredefinedViewing3D(long
index, VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.DestroyUserPanel

Deklaration:

```
Function DestroyUserPanel(ByVal buttonInfo
as long) as Boolean
```

```
HRESULT DestroyUserPanel(long
buttonInfo, VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.DisableKeyboard

Deklaration:

```
Function DisableKeyboard(ByVal aView as
View, ByVal disabled as Boolean) as
Boolean
```

```
HRESULT DisableKeyboard(IView * aView,
VARIANT_BOOL disabled,
VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht, View

ArCon.DisableKeyboardShortcuts

Deklaration:

```
Sub DisableKeyboardShortcuts(ByVal
disabled as Boolean)
```

```
HRESULT DisableKeyboardShortcuts
(VARIANT_BOOL disabled);
```

Siehe auch: ArCon Übersicht

ArCon.DoorName

Deklaration:

```
Function DoorName(ByVal DoorTypeIndex
as long) as String
```

```
HRESULT DoorName(long DoorTypeIndex,
BSTR* retVal);
```

Siehe auch: ArCon Übersicht

Diese Funktion ist obsolet und wird nur noch aus Kompatibilitätsgründen unterstützt - bitte benutzen Sie stattdessen GetDoorName.

ArCon.DragEnd

Deklaration:

```
Function DragEnd(ByVal x as long, ByVal y
as long, ByVal canceld as Boolean) as
Boolean
```

```
HRESULT DragEnd(long x, long y,
VARIANT_BOOL canceld, VARIANT_BOOL*
retVal);
```

Siehe auch: ArCon Übersicht

ArCon.DragMove

Deklaration:

```
Sub DragMove(ByVal x as long, ByVal y as
long)
```

```
HRESULT DragMove(long x, long y);
```

Siehe auch: ArCon Übersicht

ArCon.DragRButtonPressed

Deklaration:

```
Sub DragRButtonPressed(ByVal x as long,
ByVal y as long, ByVal down as Boolean)
```

```
HRESULT DragRButtonPressed(long x, long
y, VARIANT_BOOL down);
```

Siehe auch: ArCon Übersicht

ArCon.DragTextureStart

Deklaration:

```
Function DragTextureStart(ByVal fileName
as String) as Boolean
```

```
HRESULT DragTextureStart(BSTR fileName,
VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.DragTextureStart2

Deklaration:

```
Function DragTextureStart2(ByVal fileName
as String, ByVal DragFlags as
AC_ArCon_Drag_Modi, ByVal DefaultZ as
Single) as Boolean
```

```
HRESULT DragTextureStart2(BSTR
fileName, AC_ArCon_Drag_Modi DragFlags,
float DefaultZ, VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.EnableAllButtonsInPanel

Deklaration:

```
Function EnableAllButtonsInPanel(ByVal
panel as long, ByVal enabled as Boolean) as
Boolean
```

```
HRESULT EnableAllButtonsInPanel(long
panel, VARIANT_BOOL enabled,
VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.EnableButtonByID

Deklaration:

```
Function EnableButtonByID(ByVal panel as
long, ByVal ID as long, ByVal SubID as long,
ByVal enabled as Boolean) as Boolean
```

```
HRESULT EnableButtonByID(long panel,
long ID, long SubID, VARIANT_BOOL
enabled, VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.EnableMenuItem

Deklaration:

```
Function EnableMenuItem(ByVal ID as long,
ByVal doEnable as Boolean) as Boolean
```

```
HRESULT EnableMenuItem(long ID,
VARIANT_BOOL doEnable,
VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

Aktiviert oder deaktiviert einen Menüeintrag. Sie können nur die von Ihrem Makro erzeugten Menüeinträge sowie den automatisch generierten Eintrag im Makro-Menü (den Sie mit der Eigenschaft StartupMenuID ermitteln können) manipulieren.

Die Parameter sind:

ID	Die Kennzahl des Menüeintrages
enable	“ Wahr” , wenn Sie den Menüeintrag aktivieren möchten, “ Falsch” wenn Sie ihn daktivieren möchten.

ArCon.EndArCon

Deklaration:

```
Function EndArCon() as Boolean
```

```
HRESULT EndArCon(VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

Beendet ArCon. Dadurch wird das aktuelle Projekt geschlossen und die Verbindung zu Ihrem Makro beendet. Ihr Makro kann natürlich dennoch weiterarbeiten. Falls Sie dies nicht wünschen, sollten Sie das Ereignis ProgramExit abfangen.

ArCon.EndMe

Deklaration:

```
Sub EndMe()
```

```
HRESULT EndMe();
```

Siehe auch: ArCon Übersicht

Beendet die Verbindung zu ArCon - das Gegenstück zu StartMe.

Wenn Sie diese Funktion nicht aufrufen, bevor Ihr Programm beendet wird, übernimmt das Active-X Control dies für Sie.

ArCon.EndModalDialog

Deklaration:

```
Function EndModalDialog() as Boolean
```

```
HRESULT EndModalDialog(VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

Beendet einen pseudo-modalen Dialog. Beispiele zur Programmierung solcher Dialog finden Sie im Kapitel Eigene Dialoge

ArCon.GetBackgroundSettings

Deklaration:

```
Function GetBackgroundSettings(ByVal day as Boolean,
ByVal current as Boolean) as BackgroundSettings
```

```
HRESULT GetBackgroundSettings(VARIANT_BOOL day,
VARIANT_BOOL current, IBackgroundSettings** retVal);
```

Siehe auch: ArCon Übersicht, BackgroundSettings

ArCon.GetButtonState

Deklaration:

```
Function GetButtonState(ByVal btnId as long, ByRef
selected as long, ByRef pressed as Boolean, ByRef visible
as Boolean, ByRef enabled as Boolean) as Boolean
```

```
HRESULT GetButtonState(long btnId, long* selected,
VARIANT_BOOL* pressed, VARIANT_BOOL* visible,
VARIANT_BOOL* enabled, VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

Erfragt den aktuellen Zustand einer Schaltfläche.

Die Parameter sind:

btnId	Die Kennung des Knopfes, dessen Zustand ermittelt werden soll
selected	wird auf die Nummer der aktiven Variante gesetzt
pressed	wird “ Wahr” wenn der Knopf gedrückt ist
visible	wird “ Wahr” , wenn der Knopf sichtbar ist
enabled	wird “ Falsch” , wenn der Knopf gesperrt (grau) ist

ArCon.GetChimneyName

Deklaration:

```
Function GetChimneyName(ByVal typeIndex as long) as
String
```

```
HRESULT GetChimneyName(long typeIndex, BSTR*
retVal);
```

Siehe auch: ArCon Übersicht

ArCon.GetCompas

Deklaration:

```
Function GetCompas(ByVal isDefault as Boolean) as
Boolean
```

```
HRESULT GetCompas(VARIANT_BOOL isDefault, float
*phi, float *x, float *y, float *radius, VARIANT_BOOL*
retVal);
```

Siehe auch: ArCon Übersicht

ArCon.GetConstructionModeSnapSettings

Deklaration:

```
Function GetConstructionModeSnapSettings() as
Boolean
```

```
HRESULT GetConstructionModeSnapSettings
(VARIANT_BOOL *doSnap, long *snapPixelRange, long
*flags, VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.GetDBInfo

Deklaration:

```
Function GetDBInfo(ByVal index as long, ByRef Active
as Boolean, ByRef dbID as long, ByRef PhysicalName
as String, ByRef LogicalName as String) as Boolean
```

```
HRESULT GetDBInfo(long index, VARIANT_BOOL *
Active, long * dbID, BSTR * PhysicalName, BSTR *
LogicalName, VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.GetDBInfoByID

Deklaration:

```
Function GetDBInfoByID(ByVal dbID as long, ByRef
Active as Boolean, ByRef PhysicalName as String,
ByRef LogicalName as String) as Boolean
```

```
HRESULT GetDBInfoByID(long dbID, VARIANT_BOOL
* Active, BSTR * PhysicalName, BSTR * LogicalName,
VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.GetDatabaseConfiguration

Deklaration:

```
Function GetDatabaseConfiguration(ByRef registryKey
as String, ByRef dbDllName as String) as Boolean
```

```
HRESULT GetDatabaseConfiguration(BSTR *
registryKey, BSTR * dbDllName, VARIANT_BOOL*
retVal);
```

Siehe auch: ArCon Übersicht

ArCon.GetDefaultCeiling

Deklaration:

```
Function GetDefaultCeiling() as Ceiling
```

```
HRESULT GetDefaultCeiling(ICeiling** retVal);
```

Siehe auch: ArCon Übersicht, Ceiling

Liefert ein Deckenplattenobjekt mit den aktuellen Standardeinstellungen für Deckenplatten.

ArCon.GetDefaultChimney

Deklaration:

```
Function GetDefaultChimney(ByVal type as long) as
Chimney
```

```
HRESULT GetDefaultChimney(long type, IChimney**
retVal);
```

Siehe auch: ArCon Übersicht, Chimney

Liefert ein Schornstein-Objekt mit den aktuellen Standardeinstellungen für Schornsteine.

ArCon.GetDefaultDimension

Deklaration:

```
Function GetDefaultDimension() as Dimension
```

```
HRESULT GetDefaultDimension(IDimension** retVal);
```

Siehe auch: ArCon Übersicht, Dimension

Liefert ein Vermaßungsobjekt mit den aktuellen Standardeinstellungen für Vermaßungen.

ArCon.GetDefaultDoor

Deklaration:

```
Function GetDefaultDoor(ByVal type as long) as Door
```

```
HRESULT GetDefaultDoor(long type, IDoor** retVal);
```

Siehe auch: ArCon Übersicht, Door

Liefert ein Tür-Objekt mit den aktuellen Standardeinstellungen.

ArCon.GetDefaultGuide

Deklaration:

```
Function GetDefaultGuide() as Guide
```

```
HRESULT GetDefaultGuide(IGuide** retVal);
```

Siehe auch: ArCon Übersicht, Guide

Liefert ein Hilfslinienobjekt mit den Standardeinstellungen für Hilfslinien.

ArCon.GetDefaultHedge

Deklaration:

```
Function GetDefaultHedge(ByVal type as long) as
Hedge
```

```
HRESULT GetDefaultHedge(long type, IHedge** retVal)
;
```

Siehe auch: ArCon Übersicht, Hedge

Liefert ein Heckenobjekt mit den aktuellen Standardeinstellungen für Hecken.

ArCon.GetDefaultHole

Deklaration:

```
Function GetDefaultHole() as Hole
```

```
HRESULT GetDefaultHole(IHole** retVal);
```

Siehe auch: ArCon Übersicht, Hole

Liefert ein Wandloch mit den aktuellen Standardeinstellungen für Wandlöcher.

ArCon.GetDefaultLabeling

Deklaration:

```
Function GetDefaultLabeling() as Labeling
```

```
HRESULT GetDefaultLabeling(ILabeling** retVal);
```

Siehe auch: ArCon Übersicht, Labeling

Liefert ein Beschriftungsobjekt mit den aktuellen Standardeinstellungen für Beschriftungen.

ArCon.GetDefaultSupport

Deklaration:

```
Function GetDefaultSupport(ByVal type as long) as
Support
```

```
HRESULT GetDefaultSupport(long type, ISupport**
retVal);
```

Siehe auch: ArCon Übersicht, Support

Liefert ein Stützenobjekt mit den aktuellen Standardeinstellungen für Stützen.

ArCon.GetDefaultTerrain

Deklaration:

```
Function GetDefaultTerrain(ByVal type as long) as
Terrain
```

```
HRESULT GetDefaultTerrain(long type, ITerrain**
retVal);
```

Siehe auch: ArCon Übersicht, Terrain

Liefert ein Geländeobjekt mit den aktuellen Standardeinstellungen für Gelände.

ArCon.GetDefaultUnterUeberzug

Deklaration:

```
Function GetDefaultUnterUeberzug(ByVal ueberzug as
Boolean) as UnterUeberzug
```

```
HRESULT GetDefaultUnterUeberzug(VARIANT_BOOL
ueberzug, IUnterUeberzug** retVal);
```

Siehe auch: ArCon Übersicht, UnterUeberzug

Liefert einen Unter- oder Überzug, der die Standardeinstellungen für Unter- bzw. Überzüge darstellt.

Parameter:

uerberzug	Ist dieser Parameter Wahr, liefert die Methode den Standard-Überzug, ansonsten den Standard-Unterzug.

ArCon.GetDefaultWall

Deklaration:

```
Function GetDefaultWall(ByVal type as long) as
Wall
```

```
HRESULT GetDefaultWall(long type, IWall** retVal)
;
```

Siehe auch: ArCon Übersicht, Wall

Liefert ein Wandobjekt mit den aktuellen Standardeinstellungen.

ArCon.GetDesignModeSnapSettings

Deklaration:

```
Function GetDesignModeSnapSettings() as
Boolean
```

```
HRESULT GetDesignModeSnapSettings
(VARIANT_BOOL *doSnap, long *snapDistance,
float *SnapMaxAutoRotAngle, long *flags,
VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.GetDialogData

Deklaration:

```
Function GetDialogData(ByVal token as long,
ByVal subObjectNo as long, ByVal dataId as long)
as VARIANT
```

```
HRESULT GetDialogData(long token, long
subObjectNo, long dataId, VARIANT* retVal);
```

Siehe auch: ArCon Übersicht

Ermittelt Daten aus ArCon Dialogboxen. Beispiele für die Anwendung dieser Funktion finden Sie im Kapitel Vorhandene Dialoge erweitern.

Die Parameter sind:

token	Ein ArCon Dialog-Token, den Sie z.B. in einem LoadDialog Ereignis oder in einem DialogActivation Ereignis bekommen.
SubObjectNo	Einige ArCon Dialoge bearbeiten mehrere Subobjekte, z.B. die Einstellungen für Tag- und Nachtsicht oder der Dachdialog, der einzelne Dachseiten als Subobjekte bearbeitet. SubObjectNo gibt den Index des Subobjektes an, dessen Daten Sie ermitteln möchten.
DataID	Die Kennzahl des Steuerelementes, dessen Wert Sie suchen

ArCon.GetDoorName

Deklaration:

Function GetDoorName(ByVal typeIndex as long) as String

HRESULT GetDoorName(long typeIndex, BSTR* retVal);

Siehe auch: ArCon Übersicht

Liefert den Namen eines Türtyps. Türen werden von ArCon dynamisch geladen, daher ist der maximal gültige Typcode abhängig von der Installation und kann erst zur Laufzeit ermittelt werde. Zuständig dafür ist die Funktion AvailableDoors.

Das folgende Beispielprogramm füllt eine ComboBox mit den Namen aller Türtypen und setzt den jeweiligen ItemData Wert auf die Typkodierung:

```
    Dim i As Integer, last As Integer
    last = ArCon.AvailableDoors - 1
    For i = 0 To last
   Combo.AddItem ArCon.GetDoorName(i)
   Combo.ItemData(Combo.NewIndex) = i
    Next
```

ArCon.GetExternalViewDockMode

Deklaration:

```
Function GetExternalViewDockMode(ByVal viewFlag
as long) as long
```

```
HRESULT GetExternalViewDockMode(long viewFlag,
long* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.GetFileNameOfLoadingProject

Deklaration:

```
Function GetFileNameOfLoadingProject() as String
```

```
HRESULT GetFileNameOfLoadingProject(BSTR*
retVal);
```

Siehe auch: ArCon Übersicht

ArCon.GetGaubenConstructionRange

Deklaration:

```
Sub GetGaubenConstructionRange(ByVal typeIndex
as long, ByRef minConturPoints as long, ByRef
maxConturPoints as long)
```

```
HRESULT GetGaubenConstructionRange(long
typeIndex, long * minConturPoints, long *
maxConturPoints);
```

Siehe auch: ArCon Übersicht

ArCon.GetGaubenName

Deklaration:

```
Function GetGaubenName(ByVal typeIndex as long)
as String
```

```
HRESULT GetGaubenName(long typeIndex, BSTR*
retVal);
```

Siehe auch: ArCon Übersicht

ArCon.GetOneClick

Deklaration:

```
Sub GetOneClick(ByVal message as String, ByVal
picture as VARIANT, ByVal id as long)
```

```
HRESULT GetOneClick(BSTR message, VARIANT
picture, long id);
```

Siehe auch: ArCon Übersicht

Mit dieser Methode fordern Sie den Benutzer auf, einen Punkt anzuklicken. ArCon liefert Ihnen dann die Koordinaten dieses Punktes (siehe GotOneClick).

Parameter:

message	Die Anforderung an den Benutzer. Diese Nachricht wird in der Statuszeile angezeigt. Beispiel: “ Bitte klicken Sie auf die gewünschte Wand” .
Picture	Ein Bitmap für den Knopf in der “ Wie” -Leiste, der während der Bearbeitung dieses Befehls aktiviert wird. Sie können hier eine leere Variante angeben, um den Standard (Selektions)-Knopf zu verwendend. Um eine Bitmap in ein VARIANT Objekt zu verpacken, benutzen Sie die Hilfsfunktion ArConPictureFromBitmap aus der Bibliothek MakroUtil.DLL.
id	Ein beliebiger Wert, den Sie sich ausdenken. ArCon liefert diesen Wert bei der Rückmeldung des Mausklicks mit, sodaß Sie in der Ereignisbehandlungsroutine unterscheiden können, um welchen Klick es sich handelt.

ArCon.GetOnlySomeBuildingsLoading

Deklaration:

```
Function GetOnlySomeBuildingsLoading() as
Boolean
```

```
HRESULT GetOnlySomeBuildingsLoading
(VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.GetPath

Deklaration:

```
Function GetPath(ByVal index as
AC_GETPATH_INDEX) as String
```

```
HRESULT GetPath(AC_GETPATH_INDEX index,
BSTR* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.GetPfostenBeschreibung

Deklaration:

```
Function GetPfostenBeschreibung(ByVal Index as
long) as String
```

```
HRESULT GetPfostenBeschreibung(long Index,
BSTR* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.GetPfostenObject

Deklaration:

```
Function GetPfostenObject(ByVal Index as long)
as String
```

```
HRESULT GetPfostenObject(long Index, BSTR*
retVal);
```

Siehe auch: ArCon Übersicht

ArCon.GetPredefinedViewing3D

Deklaration:

```
Function GetPredefinedViewing3D(ByVal index
as long, ByRef Description as String) as Boolean
```

```
HRESULT GetPredefinedViewing3D(long index,
BSTR * Description, float *VRPx, float *VRPy,
float *VRPz, float *EyeX, float *EyeY, float *EyeZ,
float *tanViewAngleHalf, VARIANT_BOOL*
retVal);
```

Siehe auch: ArCon Übersicht

ArCon.GetRoofWindowName

Deklaration:

```
Function GetRoofWindowName(ByVal typeIndex
as long) as String
```

```
HRESULT GetRoofWindowName(long typeIndex,
BSTR* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.GetStairCaseName

Deklaration:

```
Function GetStairCaseName(ByVal typeIndex as
long) as String
```

```
HRESULT GetStairCaseName(long typeIndex,
BSTR* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.GetSupportName

Deklaration:

```
Function GetSupportName(ByVal typeIndex as
long) as String
```

```
HRESULT GetSupportName(long typeIndex,
BSTR* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.GetTerrainHeight

Deklaration:

```
Function GetTerrainHeight(ByVal x as Single,
ByVal y as Single) as Single
```

```
HRESULT GetTerrainHeight(float x, float y, float*
retVal);
```

Siehe auch: ArCon Übersicht

Interpoliert die Geländehöhe an der angegebenen Position.

ArCon.GetTypeIDFromDB

Deklaration:

```
Function GetTypeIDFromDB(ByVal dbID as long,
ByVal itemID as long, ByVal objType as
AC_ArCon_Objekt_Typen, ByRef typeCode as
long) as Boolean
```

```
HRESULT GetTypeIDFromDB(long dbID, long
itemID, AC_ArCon_Objekt_Typen objType, long *
typeCode, VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.GetWindowName

Deklaration:

```
Function GetWindowName(ByVal typeIndex as
long) as String
```

```
HRESULT GetWindowName(long typeIndex,
BSTR* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.GroupDesignObjects

Deklaration:

```
Function GroupDesignObjects(ByVal Name as
String, ByVal NumOfObjects as long, ByVal
ObjectArray as VARIANT) as Object3D
```

```
HRESULT GroupDesignObjects(BSTR Name,
long NumOfObjects, VARIANT ObjectArray,
IObject3D** retVal);
```

Siehe auch: ArCon Übersicht, Object3D

Faßt eine Menge von 3D Objekten zu einer Gruppe zusammen. Die relative Position der Objekte innerhalb der Gruppe ergibt sich aus der aktuellen absoluten Position der Einzelobjekte.

Parameter:

Name	Der logische Name des resultierenden Gruppenobjektes
NumObjects	Anzahl der Objekte im Array
ObjectArray	ein Variant, der ein eindimensionales Array von 3D-Objekten enthält.

ArCon.ImportFolie

Deklaration:

```
Function ImportFolie(ByVal layer as long, ByRef
fileName as String, ByRef minX as Single, ByRef
minY as Single, ByRef maxX as Single, ByRef maxY
as Single) as Graphics2DCollection
```

```
HRESULT ImportFolie(long layer, BSTR* fileName,
float* minX, float* minY, float* maxX, float* maxY,
IGraphics2DCollection** retVal);
```

Siehe auch: ArCon Übersicht, Graphics2DCollection

Importiert eine Folie (DXF oder HPGL Format) und liefert die Elemente dieser Folie als Graphics2DCollection. Sie können diese Elemente weiterbearbeiten (z.B. skalieren oder positionieren) und anzeigen, in dem Sie sie entweder in die globale Grafikliste ArCon.Graphics2D oder eine Stockwerspezifische Liste Story.Graphics2D kopieren.

Parameter sind:

layer	Zeichnungsebene
fileName	Der Name der importierten Datei. Durch Vorbelegung dieses Parameters können Sie das Startverzeichnis des Import-Dialoges vorgeben.
minX, maxX, minY, maxY	Größe der importierten Folie (Achtung: hier wird die tatsächlich von den gelieferten Grafikelementen belegte Fläche geliefert, nicht die eventuell in der importierten Datei vorhande Größenangabe)

ArCon.InvalidateLightSettings

Deklaration:

```
Sub InvalidateLightSettings()
```

```
HRESULT InvalidateLightSettings();
```

Siehe auch: ArCon Übersicht

ArCon.LoadObjectConstructor

Deklaration:

```
Function LoadObjectConstructor(ByVal fileName as
String, ByVal duration as long) as
ObjectConstructor
```

```
HRESULT LoadObjectConstructor(BSTR fileName,
long duration, IObjectConstructor** retVal);
```

Siehe auch: ArCon Übersicht, ObjectConstructor

Lädt eine 3D-Objektklasse aus einer Datei. Je nach verwendeter ArCon Version sind Dateien in den Formaten “ .ACO” , “ .LMO” und “ .3DS” ladbar.

Parameter:

fileName	Der Name der ACO Datei
duration	Flags zur Cache-Verwaltung des Objektes. Sollte derzeit immer den Wert ACO_DURATION_CACHEABLE (= 1) haben.

ArCon.LoadObjectDialog

Deklaration:

```
Function LoadObjectDialog(ByVal ParentWindow
as long, ByVal Caption as String, ByVal objects as
Boolean, ByVal groups as Boolean, ByRef
FileName as String, ByRef Directory as String) as
Boolean
```

```
HRESULT LoadObjectDialog(long ParentWindow,
BSTR Caption, VARIANT_BOOL objects,
VARIANT_BOOL groups, BSTR* FileName,
BSTR* Directory, VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.LoadProject

Deklaration:

```
Function LoadProject(ByVal fileName as String)
as Boolean
```

```
HRESULT LoadProject(BSTR fileName,
VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

Lädt ein anderes Projekt.

Die Parameter sind:

fileName	der Dateiname inclusive Endung (.acp bzw. .lmp)

ArCon.LoadProjectDialog

Deklaration:

```
Function LoadProjectDialog(ByVal ParentWindow as
long, ByVal Caption as String, ByRef FileName as
String, ByRef Directory as String) as Boolean
```

```
HRESULT LoadProjectDialog(long ParentWindow,
BSTR Caption, BSTR* FileName, BSTR* Directory,
VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.LoadTextureDialog

Deklaration:

```
Function LoadTextureDialog(ByVal ParentWindow as
long, ByVal Caption as String, ByRef FileName as
String, ByRef Directory as String) as Boolean
```

```
HRESULT LoadTextureDialog(long ParentWindow,
BSTR Caption, BSTR* FileName, BSTR* Directory,
VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.NewCeiling

Deklaration:

```
Function NewCeiling() as Ceiling
```

```
HRESULT NewCeiling(ICeiling** retVal);
```

Siehe auch: ArCon Übersicht, Ceiling

Generiert ein Deckenplattenobjekt und initialisiert es mit den aktuellen Standardwerten.

ArCon.NewCeilingOpening

Deklaration:

```
Function NewCeilingOpening() as CeilingOpening
```

```
HRESULT NewCeilingOpening(ICeilingOpening**
retVal);
```

Siehe auch: ArCon Übersicht, CeilingOpening

Generiert ein Deckenöffnungsobjekt und initialisiert es mit den aktuellen Standardwerten.

ArCon.NewChimney

Deklaration:

```
Function NewChimney(ByVal type as long) as
Chimney
```

```
HRESULT NewChimney(long type, IChimney** retVal)
;
```

Siehe auch: ArCon Übersicht, Chimney

Generiert ein Schornsteinobjekt und initialisiert es mit den Standardwerten für den angegebenen Typ. Schornsteintypen sind Zahlen zwischen 0 und AvailableChimneys - 1.

Parameter sind:

type	Typcode der gewünschten Schornsteinvariante.

ArCon.NewDimension

Deklaration:

```
Function NewDimension() as Dimension
```

```
HRESULT NewDimension(IDimension** retVal);
```

Siehe auch: ArCon Übersicht, Dimension

Erzeugt ein neues Vermaßungsobjekt und initialisiert es mit den aktuellen Standardeinstellungen des angegebenen Typs. Gültige Typcodes finden Sie unter Bemaßungs-Typen.

Parameter sind:

Type	Art der Darstellung der Vermaßung.

ArCon.NewDoor

Deklaration:

```
Function NewDoor(ByVal type as long) as Door
```

```
HRESULT NewDoor(long type, IDoor** retVal);
```

Siehe auch: ArCon Übersicht, Door

Erzeugt ein neues Türobjekt und initialisiert es mit den Standardeinstellungen des angegebenen Türtyps.

Parameter sind:

type	Die gewünschte Türvariante, eine Zahl zwischen 0 und AvailableDoors - 1.

ArCon.NewGuide

Deklaration:

```
Function NewGuide() as Guide
```

```
HRESULT NewGuide(IGuide** retVal);
```

Siehe auch: ArCon Übersicht, Guide

Erzeugt ein neues Stützenobjekt und initialisiert es mit den aktuellen Standardeinstellungen.

ArCon.NewHedge

Deklaration:

```
Function NewHedge(ByVal type as long) as Hedge
```

```
HRESULT NewHedge(long type, IHedge** retVal);
```

Siehe auch: ArCon Übersicht, Hedge

Erzeugt eine neues Heckenobjekt und initialisiert es mit den Standardeinstellungen für den angegebenen Heckentyp. Heckentypen sind Zahlen zwischen 0 und AC_MaxHedgeType.

Parameter sind:

type	Der gewünschte Heckentyp.

ArCon.NewHole

Deklaration:

```
Function NewHole() as Hole
```

```
HRESULT NewHole(IHole** retVal);
```

Siehe auch: ArCon Übersicht, Hole

Erzeugt ein neues Wandlochobjekt und initialisiert es mit den aktuellen Standardeinstellungen.

ArCon.NewHolePolygon

Deklaration:

```
Function NewHolePolygon(ByVal points as
Point2DCollection) as HolePolygon
```

```
HRESULT NewHolePolygon(IPoint2DCollection* points,
IHolePolygon** retVal);
```

Siehe auch: ArCon Übersicht, HolePolygon, Point2DCollection

ArCon.NewImage

Deklaration:

```
Function NewImage(ByVal Layer as long) as Image
```

```
HRESULT NewImage(long Layer, IImage** retVal);
```

Siehe auch: ArCon Übersicht, Image

Diese Methode liefert ein neues Image Objekt.

Parameter:

Layer Gibt an, in welcher Zeichnungsebene das Objekt erscheinen soll.

ArCon.NewLabel

Deklaration:

```
Function NewLabel(ByVal Layer as long) as Label
```

```
HRESULT NewLabel(long Layer, ILabel** retVal);
```

Siehe auch: ArCon Übersicht, Label

Diese Methode liefert ein neues Label Objekt.

Parameter:

Layer Gibt an, in welcher Zeichnungsebene das Objekt erscheinen soll.

ArCon.NewLabeling

Deklaration:

```
Function NewLabeling() as Labeling
```

```
HRESULT NewLabeling(ILabeling** retVal);
```

Siehe auch: ArCon Übersicht, Labeling

Erzeugt ein neues Beschriftungsobjekt und initialisert es mit den aktuellen Standardeinstellungen.

ArCon.NewLine

Deklaration:

```
Function NewLine(ByVal Layer as long) as Line
```

```
HRESULT NewLine(long Layer, ILine** retVal);
```

Siehe auch: ArCon Übersicht, Line

Diese Methode liefert ein neues Line Objekt.

Parameter:

Layer Gibt an, in welcher Zeichnungsebene das Objekt erscheinen soll.

ArCon.NewMaterial

Deklaration:

```
Function NewMaterial() as Material
```

```
HRESULT NewMaterial(IMaterial** retVal);
```

Siehe auch: ArCon Übersicht, Material

Erzeugt ein neues Material für die Konstruktion von 3D Objekten (siehe ObjectConstructor).

ArCon.NewObjectConstructor

Deklaration:

```
Function NewObjectConstructor(ByVal epsilon as Single,
ByVal sharpAngle as Single) as ObjectConstructor
```

```
HRESULT NewObjectConstructor(float epsilon, float
sharpAngle, IObjectConstructor** retVal);
```

Siehe auch: ArCon Übersicht, ObjectConstructor

Erzeugt ein Fabrik-Objekt, mit dessen Hilfe Sie 3D Objekte definieren können.

Parameter sind:

epsilon	Abstand zwischen zwei Punkten, bis zu dem diese Punkte als identisch betrachtet werden. Wenn Sie exakte Punktkoordinaten angeben, können Sie diesen Parameter auf 0 setzen. Falls bei der Konstruktion des Objektes (bedingt durch Rechenungenauigkeiten) gleiche Koordinaten nicht mit gleichen Werten entstehen, geben Sie hier die Rechenungenauigkeit an.
SharpAngle	Ein Winkel in Bogenmaß. Die Kante zwischen zwei benachbarten Flächen Ihres Objektes wird scharf dargestellt, wenn der Winkel, mit dem die Flächen aufeinanderstoßen, größer als der hier angegebene Winkel ist, oder Sie explizit bei der Konstruktion den entsprechenden “ sharpEdges” Parameter gesetzt haben (siehe AddTriangle). Falls Sie z.B. einen Würfel konstruieren, setzen Sie diesen Parameter auf 0, wodurch alle Kanten scharf dargestellt werden. Wenn Ihr Objekt keinerlei scharfe Kanten hat (z.B. ein Ball), benutzen Sie z.B Pi.

ArCon.NewObjectTransformerCollection

Deklaration:

```
Function NewObjectTransformerCollection() as
ObjectTransformerCollection
```

```
HRESULT NewObjectTransformerCollection
(IObjectTransformerCollection** retVal);
```

Siehe auch: ArCon Übersicht, ObjectTransformerCollection

Erzeugt eine neue Liste von Animations-Objekten. Mit Hilfe einer solchen Liste können mehrere Objekte gleichzeitig animiert werden.

Diese “ Animation” geschieht über eine externe Transformation des gesamten Objektes und hat nichts gemeinsam mit den internen Eigenbewegungen von Objekten, wie Sie z.B. durch klicken mit der rechten Maustaste auf ein Objekt aktiviert werden können.

ArCon.NewPoint2DCollection

Deklaration:

```
Function NewPoint2DCollection() as Point2DCollection
```

```
HRESULT NewPoint2DCollection(IPoint2DCollection**
retVal);
```

Siehe auch: ArCon Übersicht, Point2DCollection

ArCon.NewPolygon2D

Deklaration:

```
Function NewPolygon2D(ByVal Layer as long) as
Polygon2D
```

```
HRESULT NewPolygon2D(long Layer, IPolygon2D**
retVal);
```

Siehe auch: ArCon Übersicht, Polygon2D

Diese Methode liefert ein neues Polygon2D Objekt.

Parameter:

Layer Gibt an, in welcher Zeichnungsebene das Objekt erscheinen soll.

ArCon.NewPolygonWendelConstruction

Deklaration:

```
Function NewPolygonWendelConstruction() as
PolygonWendelConstruction
```

```
HRESULT NewPolygonWendelConstruction
(IPolygonWendelConstruction** retVal);
```

Siehe auch: ArCon Übersicht, PolygonWendelConstruction

ArCon.NewPrintSettings

Deklaration:

```
Function NewPrintSettings() as PrintSettings
```

```
HRESULT NewPrintSettings(IPrintSettings** retVal);
```

Siehe auch: ArCon Übersicht, PrintSettings

ArCon.NewProject

Deklaration:

```
Function NewProject() as Project
```

```
HRESULT NewProject(IProject** retVal);
```

Siehe auch: ArCon Übersicht, Project

Erzeugt eine neue Projektbeschreibung und initialisiert sie mit den Standardeinstellungen für neue Projekte. Anschließend können Sie diese Projektbeschreibung an die Funktion CreateProject übergeben.

ArCon.NewProjectPreview

Deklaration:

```
Function NewProjectPreview(ByVal FileName as String)
as ProjectPreview
```

```
HRESULT NewProjectPreview(BSTR FileName,
IProjectPreview** retVal);
```

Siehe auch: ArCon Übersicht, ProjectPreview

ArCon.NewRoof

Deklaration:

```
Function NewRoof() as Roof
```

```
HRESULT NewRoof(IRoof** retVal);
```

Siehe auch: ArCon Übersicht, Roof

Erzeugt ein neues Dachobjekt und initialisiert es mit den aktuellen Standardeinstellungen.

ArCon.NewRoofConstruction

Deklaration:

```
Function NewRoofConstruction() as RoofConstruction
```

```
HRESULT NewRoofConstruction(IRoofConstruction**
retVal);
```

Siehe auch: ArCon Übersicht, RoofConstruction

ArCon.NewRoofWindow

Deklaration:

```
Function NewRoofWindow(ByVal type as long) as
RoofWindow
```

```
HRESULT NewRoofWindow(long type, IRoofWindow**
retVal);
```

Siehe auch: ArCon Übersicht, RoofWindow

ArCon.NewSavePictureSettings

Deklaration:

```
Function NewSavePictureSettings() as
SavePictureSettings
```

```
HRESULT NewSavePictureSettings
(ISavePictureSettings** retVal);
```

Siehe auch: ArCon Übersicht, SavePictureSettings

ArCon.NewShape

Deklaration:

```
Function NewShape(ByVal Layer as long) as Shape
```

```
HRESULT NewShape(long Layer, IShape** retVal);
```

Siehe auch: ArCon Übersicht, Shape

Diese Methode liefert ein neues Shape Objekt.

Parameter:

Layer Gibt an, in welcher Zeichnungsebene das Objekt erscheinen soll.

ArCon.NewSnapObject

Deklaration:

```
Function NewSnapObject() as SnapObject
```

```
HRESULT NewSnapObject(ISnapObject** retVal);
```

Siehe auch: ArCon Übersicht, SnapObject

ArCon.NewStairCase

Deklaration:

```
Function NewStairCase() as StairCase
```

```
HRESULT NewStairCase(IStairCase** retVal);
```

Siehe auch: ArCon Übersicht, StairCase

Erzeugt ein neues Treppenobjekt und initialisiert es mit den aktuellen Standardwerten.

ArCon.NewSupport

Deklaration:

```
Function NewSupport(ByVal type as long) as Support
```

```
HRESULT NewSupport(long type, ISupport** retVal);
```

Siehe auch: ArCon Übersicht, Support

Erzeugt ein neues Stützenobjekt vom angegebenen Typ und initialisiert es mit den aktuellen Standardwerten. Stützentypen sind Zahlen zwischen 0 und AvailableSupports - 1.

Parameter sind:

type	Der gewünschte Stützentyp

ArCon.NewTerrain

Deklaration:

```
Function NewTerrain(ByVal type as long) as Terrain
```

```
HRESULT NewTerrain(long type, ITerrain** retVal);
```

Siehe auch: ArCon Übersicht, Terrain

Erzeugt ein neues Geländeobjekt und initialisiert es mit den aktuellen Standardeinstellungen dieses Geländetyps. Geländetypen sind Zahlen zwischen 0 und AC_MaxTerrainType.

Parameter sind:

type	Der gewünschte Geländetyp

ArCon.NewTexture

Deklaration:

```
Function NewTexture() as Texture
```

```
HRESULT NewTexture(ITexture** retVal);
```

Siehe auch: ArCon Übersicht, Texture

Erzeugt ein neues Texturobjekt und initialisiert es als leere Textur.

ArCon.NewTextureCollection

Deklaration:

```
Function NewTextureCollection(ByVal numElements as
long) as TextureCollection
```

```
HRESULT NewTextureCollection(long numElements,
ITextureCollection** retVal);
```

Siehe auch: ArCon Übersicht, TextureCollection

Erzeugt eine neue Texturliste. Damit kann z.B. eine neu erzeugte 3D-Objektinstanz mit vom Standard abweichenden Texturen erzeugt werden.

ArCon.NewUnterUeberzug

Deklaration:

```
Function NewUnterUeberzug(ByVal ueberzug as
Boolean) as UnterUeberzug
```

```
HRESULT NewUnterUeberzug(VARIANT_BOOL
ueberzug, IUnterUeberzug** retVal);
```

Siehe auch: ArCon Übersicht, UnterUeberzug

Erzeugt einen neuen Unter- oder Überzug.

Parameter:

ueberzug	Wahr, wenn ein Überzug erzeugt werden soll, andernfalls wird ein Unterzug erstellt.

ArCon.NewVirtualWall

Deklaration:

```
Function NewVirtualWall() as VirtualWall
```

```
HRESULT NewVirtualWall(IVirtualWall** retVal);
```

Siehe auch: ArCon Übersicht, VirtualWall

ArCon.NewWall

Deklaration:

```
Function NewWall(ByVal type as long) as Wall
```

```
HRESULT NewWall(long type, IWall** retVal);
```

Siehe auch: ArCon Übersicht, Wall

Erzeugt ein neues Wandobjekt und initialisiert es mit den aktuellen Standardeinstellungen für diesen Wandtyp. Gültige Wandtypen liegen zwischen 0 AvailableWalls - 1.

Parameter sind:

type	Der gewünschte Wandtyp.

ArCon.NewWindow

Deklaration:

```
Function NewWindow(ByVal type as long) as Window
```

```
HRESULT NewWindow(long type, IWindow** retVal);
```

Siehe auch: ArCon Übersicht, Window

Erzeugt ein neues Fensterobjekt und initialisiert es mit den aktuellen Standardeinstellungen. Gültige Fenstertypen sind 0 bis AvailableWindows - 1, wobei die erste 0 .. ConstructedWindows - 1 zur Laufzeit konstruierte Fenster sind, während die übrigen geladene Objekte mit statischen Eigenschaften sind.

Die Parameter sind:

type	Typ des zu erzeugeneden Fensters.

ArCon.NotifyOnChange

Deklaration:

```
Function NotifyOnChange(ByVal anyObject as Object,
ByVal events as long) as Boolean
```

```
HRESULT NotifyOnChange(IDispatch* anyObject, long
events, VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

Einzelne Objekte können Ihrem Makro mitteilen, wenn sie geändert werden. Da solche Änderungen häufig und die Zahl der Objekte oft groß ist, werden diese Ereignisse nur geliefert, wenn Sie sich für das konkrete Objekt und die konkrete Ereignismenge angemeldet haben.

Unabhängig von konkreten Instanzen eines Typs können Sie sich über allgemeine Ereignisse, z. B. die Erzeugung eines neuen Objektes dieses Typs, informieren lassen, indem Sie mit der Funktion ArCon.ChangeTypeNotifyMask ein Typ-Ereignismaske setzen.

Parameter sind:

anyObject	Das Objekt, von dem Sie Änderungen mitgeteilt bekommen möchten
events	Die Bitmaske der Ereignisse, an denen Sie interessiert sind, siehe Ereigniskonstanten

ArCon.ParseDBIDString

Deklaration:

```
Function ParseDBIDString(ByVal dbIdStr as String, ByRef dbID as
long, ByRef objID as long, ByRef objType as
AC_ArCon_DB_Types, ByRef guidStr as String) as Boolean
```

```
HRESULT ParseDBIDString(BSTR dbIdStr, long * dbID, long *
objID, AC_ArCon_DB_Types * objType, BSTR * guidStr,
VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.PlaceTerrain

Deklaration:

```
Function PlaceTerrain(ByVal aTerrain as Terrain, ByVal aPolygon
as Polygon2D) as Boolean
```

```
HRESULT PlaceTerrain(ITerrain* aTerrain, IPolygon2D* aPolygon,
VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht, Terrain, Polygon2D

Plaziert ein vorbereitetes Geländeobjekt innerhalb des angegebenen Polygons.

Rückgabewert ist “ Wahr” , wenn das Gelände plaziert werden konnte, ansonsten “ Falsch” .

Parameter sind:

aTerrain	Ein Terrain-Objekt mit den Einstellung für das zu plazierende Terrain
aPolygon2D	Die Umrandung des neu plazierten Terrains. Vom Polygon2D Objekt werden nur die Koordinaten (Eigenschaft Points) ausgewertet.

ArCon.PredefinedViewing3DCount

Deklaration:

```
Function PredefinedViewing3DCount() as long
```

```
HRESULT PredefinedViewing3DCount(long* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.ReadChunk

Deklaration:

```
Function ReadChunk(ByVal ChunkID as long, ByVal
FileName as String) as Boolean
```

```
HRESULT ReadChunk(long ChunkID, BSTR FileName,
VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

Kopiert benutzerdefinierte Daten aus einem ArCon-Projekt in eine getrennte Datei. Beispiele zur Anwendung der Lade- und Speicherfunktionen finden Sie unter Speichern und Laden .

Parameter sind:

ChunkID	Eine Kennung für die benutzerdefinierten Daten
FileName	Der Name der Datei, in die diese Daten kopiert werden sollen.

ArCon.Redraw3DViews

Deklaration:

```
Sub Redraw3DViews()
```

```
HRESULT Redraw3DViews();
```

Siehe auch: ArCon Übersicht

Aktualisiert die Darstellung aller 3D Ansichten.

In der Regel ist dies nicht nötig, da ArCon bei allen relevanten Operationen diese Ansichten automatisch aktualisiert. Um umfangreichere Aktionen zu beschleunigen können Sie aber durch setzen beziehungsweise löschen des Bits ACMU_REDRAWENABLED die automatische Aktualisierung steuern. Wenn diese Bit nicht gesetzt ist, können Sie durch Aufruf dieser Funktion dennoch Bildschirmaktualisierungen erzwingen.

ArCon.RemoveAnyMenuItem

Deklaration:

```
Function RemoveAnyMenuItem(ByVal ID as long) as
Boolean
```

```
HRESULT RemoveAnyMenuItem(long ID,
VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

Falls die Eigenschaft InternalMode “ Wahr” ist, können Sie mit Hilfe dieser Funktion beliebige ArCon Menü-Einträge löschen (mit der Funktion RemoveMenuItem können Sie nur von Ihrem Makro erzeugte Menüeinträge entfernen).

Der Rückgabewert ist “ Wahr” wenn der angegegebene Menüeintrag entfernt wurde, “ Falsch” wenn dieser Eintrag nicht gefunden wurde.

Die Parameter sind:

ID	Kennzahl des zu löschenden Menüeintrages.

ArCon.RemoveMenuItem

Deklaration:

```
Function RemoveMenuItem(ByVal ID as long) as
Boolean
```

```
HRESULT RemoveMenuItem(long ID,
VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

Entfernt einen benutzerdefinierten Menüeintrag. Beispiele für die Menüprogrammierung finden Sie unter Eigene Menüeinträge.

Parameter sind:

ID	Kennzahl des zu entfernenden Menüeintrags. Sie können nur Menüeinträge entfernen, die Sie selbst erzeugt haben sowie den von der Funktion StartupMenuID gelieferten.

ArCon.RenamePredefinedViewing3D

Deklaration:

```
Function RenamePredefinedViewing3D(ByVal index as
long, ByVal newDescription as String) as Boolean
```

```
HRESULT RenamePredefinedViewing3D(long index,
BSTR newDescription, VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.RoofWindowName

Deklaration:

```
Function RoofWindowName(ByVal WindowTypeIndex as
long) as String
```

```
HRESULT RoofWindowName(long WindowTypeIndex,
BSTR* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.RunInProcMacro

Deklaration:

```
Function RunInProcMacro(ByVal fileName as String) as
Boolean
```

```
HRESULT RunInProcMacro(BSTR fileName,
VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.SaveProject

Deklaration:

```
Function SaveProject(ByVal mayAskForName as
Boolean) as Boolean
```

```
HRESULT SaveProject(VARIANT_BOOL
mayAskForName, VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

Speichert das aktuelle Projekt. Falls noch kein Name angegeben wurde, wird der “ Speichern unter” Dialog geöffnet.

Parameter sind:

mayAskForName	“ Wahr” , wenn der Benutzer nach einem Namen gefragt werden darf, “ Falsch” , wenn keine Benutzerinteraktion erlaubt ist. In diesem Fall schlägt der Befehl fehl, falls in dem Projekt Änderungen vorgenommen wurden, die noch nicht gespeichert wurden.

ArCon.SaveProjectAs

Deklaration:

```
Function SaveProjectAs(ByVal fileName as
String) as Boolean
```

```
HRESULT SaveProjectAs(BSTR fileName,
VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

Speichert das aktuelle Projekt unter einem neuen Namen.

Parameter sind:

fileName	Der Dateiname, unter dem das Projekt gespeichert wird.

ArCon.SelectAll3DObjects

Deklaration:

```
Function SelectAll3DObjects() as Boolean
```

```
HRESULT SelectAll3DObjects(VARIANT_BOOL*
retVal);
```

Siehe auch: ArCon Übersicht

ArCon.Selected3DObjects

Deklaration:

```
Function Selected3DObjects() as VARIANT
```

```
HRESULT Selected3DObjects(VARIANT* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.SetButtonPosition

Deklaration:

```
Function SetButtonPosition(ByVal button as long,
ByVal PosOrId as long, ByVal SubId as long, ByVal
byPosition as Boolean) as Boolean
```

```
HRESULT SetButtonPosition(long button, long
PosOrId, long SubId, VARIANT_BOOL byPosition,
VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.SetButtonState

Deklaration:

```
Function SetButtonState(ByVal btnId as long, ByVal
selectedIndex as long, ByVal pressed as Boolean,
ByVal visible as Boolean, ByVal enabled as Boolean)
as Boolean
```

```
HRESULT SetButtonState(long btnId, long
selectedIndex, VARIANT_BOOL pressed,
VARIANT_BOOL visible, VARIANT_BOOL enabled,
VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

Ändert den Zustand einer Schaltfläche. Beispiele für die Programmierung von Schaltflächen finden Sie unter Eigene Schaltflächen.

Parameter sind:

btnId	Die Kennzahl der Schaltfläche
selectedIndex	Bei Variantenschaltern die Nummer der ausgewählten Variante
pressed	“ Wahr” , wenn der Knopf gedrückt sein soll
visible	“ Wahr” , wenn der Knopf sichtbar sein soll
enabled	“ Wahr” , wenn der Knopf aktiviert sein soll

ArCon.SetCompas

Deklaration:

```
Function SetCompas(ByVal isDefault as Boolean,
ByVal phi as Single, ByVal x as Single, ByVal y as
Single, ByVal radius as Single) as Boolean
```

```
HRESULT SetCompas(VARIANT_BOOL isDefault,
float phi, float x, float y, float radius,
VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.SetConstructionModeSnapSettings

Deklaration:

```
Function SetConstructionModeSnapSettings(ByVal
doSnap as Boolean, ByVal snapPixelRange as long,
ByVal flags as long) as Boolean
```

```
HRESULT SetConstructionModeSnapSettings
(VARIANT_BOOL doSnap, long snapPixelRange,
long flags, VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.SetDesignModeSnapSettings

Deklaration:

```
Function SetDesignModeSnapSettings(ByVal
doSnap as Boolean, ByVal snapDistance as long,
ByVal SnapMaxAutoRotAngle as Single, ByVal flags
as long) as Boolean
```

```
HRESULT SetDesignModeSnapSettings
(VARIANT_BOOL doSnap, long snapDistance, float
SnapMaxAutoRotAngle, long flags,
VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.SetDialogData

Deklaration:

```
Function SetDialogData(ByVal token as long, ByVal
subObjectNo as long, ByVal dataId as long, ByVal
value as VARIANT) as Boolean
```

```
HRESULT SetDialogData(long token, long
subObjectNo, long dataId, VARIANT value,
VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

Verändert Daten in einer ArCon Dialogbox. Beispiele zur Programmierung von Dialogboxen finden Sie im Kapitel Vorhandene Dialoge erweitern.

Parameter sind:

token	Ein ArCon Dialog-Token, den Sie z.B. im Ereignis LoadDialog oder DialogActivation erhalten.
subObjectNo	Bei Dialogen mit mehreren Objekten, z.B. dem Dacheditor mit mehreren Dachseiten, die Nummer des bearbeiteten Subobjektes
dataID	Die Kennzahl des zu verändernden Eingabelementes
value	Der neue Wert für dieses Eingabefeld

ArCon.SetExternalViewDockMode

Deklaration:

```
Function SetExternalViewDockMode(ByVal viewFlag
as long, ByVal DockMode as long, ByVal screenLeft
as long, ByVal screenTop as long, ByVal
screenBottom as long, ByVal screeRight as long) as
Boolean
```

```
HRESULT SetExternalViewDockMode(long viewFlag,
long DockMode, long screenLeft, long screenTop, long
screenBottom, long screeRight, VARIANT_BOOL*
retVal);
```

Siehe auch: ArCon Übersicht

ArCon.SetExternalViews

Deklaration:

```
Sub SetExternalViews(ByVal newVal as long)
```

```
HRESULT SetExternalViews(long newVal);
```

Siehe auch: ArCon Übersicht

ArCon.SetInputMode

Deklaration:

```
Sub SetInputMode(ByVal newMode as long)
```

```
HRESULT SetInputMode(long newMode);
```

Siehe auch: ArCon Übersicht

Schaltet das aktive Eingabewerkzeug um (in der Was-Leiste) um. Die Werkzeuge sind von 0 an durchnummeriert, 0 ist Selektion, 1 ist Wand, ...

Der Parameter -1 schaltet auf das vorher aktive Werkzeug zurück.

Parameter ist:

newMode	Nummer des neuen Eingabemodus oder -1 für den zuletzt aktiven

ArCon.SetObject3DEventMask

Deklaration:

```
Function SetObject3DEventMask(ByVal obj as
Object3D, ByVal eventMask as long) as long
```

```
HRESULT SetObject3DEventMask(IObject3D *
obj, long eventMask, long* retVal);
```

Siehe auch: ArCon Übersicht, Object3D

Sie könne sich von konkreten 3D-Objekt Instanzen über verschiedene Ereignisse informieren lassen. Da der Aufruf Ihrer Ereignisroutine nicht unerheblich Zeit benötigt, sind alle diese Ereignisse standardmäßig ausgeschaltet.

Bitmasken für die möglichen Ereignisse finden Sie unter 3D Objekt-Ereignisse.

Parameter sind:

obj	das Objekt, von dem Sie die Ereignisse empfangen möchten
eventMask	Bitmenge der zu liefernden Ereignisse

ArCon.SetParentWindow

Deklaration:

```
Sub SetParentWindow(ByVal ParentWindow as long)
```

```
HRESULT SetParentWindow(long ParentWindow);
```

Siehe auch: ArCon Übersicht

ArCon.SetProgressbarSubTitle

Deklaration:

```
Sub SetProgressbarSubTitle(ByVal subTitle as String)
```

```
HRESULT SetProgressbarSubTitle(BSTR subTitle);
```

Siehe auch: ArCon Übersicht

Zeigt einen Untertitel rechts neben dem Fortschrittsbalken an. ArCon benutzt diese Anzeige z.B. beim Laden eines Projektes um die aktuelle Ladephase anzuzeigen: “ Texturen” , “ Geschosse” , “ Fenster und Türen” , “ Einrichtungsgegenstände” u.s.w.

Dies setzt natürlich voraus, daß ein Fortschrittsbalken angezeigt wird, also vorher die Funktion StartProgressbar aufgerufen wurde.

Parameter sind:

subTitle	Der rechts neben dem Forstchrittsbalken anzuzeigende Text.

ArCon.SetProgressbarValue

Deklaration:

```
Sub SetProgressbarValue(ByVal percent as long)
```

```
HRESULT SetProgressbarValue(long percent);
```

Siehe auch: ArCon Übersicht

Setzt den aktuellen Prozentwert des Fortschrittsbalkens auf den angegebenen Wert.

Dazu muß vorher mit der Funktion StartProgressbar eine Fortschrittsanzeige aktiviert worden sein.

Parameter sind:

percent	der neue Prozentwert

ArCon.SetStatusText

Deklaration:

```
Sub SetStatusText(ByVal msg as String)
```

```
HRESULT SetStatusText(BSTR msg);
```

Siehe auch: ArCon Übersicht

Zeigt eine Nachricht in der Statuszeile an.

Parameter sind:

msg	Die anzuzeigende Nachricht

ArCon.SetTerrainHeight

Deklaration:

```
Function SetTerrainHeight(ByVal x as Single, ByVal y as
Single, ByVal height as Single) as Boolean
```

```
HRESULT SetTerrainHeight(float x, float y, float height,
VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

Erzeugt oder ändert einen Höhenpunkt im Gelände.

Parameter sind:

x, y	Die Koordinaten des Punktes
height	Die neue Höhe des Punktes

ArCon.SetUserPanelState

Deklaration:

```
Function SetUserPanelState(ByVal buttonInfo as long,
ByVal visible as Boolean, ByVal enabled as Boolean) as
Boolean
```

```
HRESULT SetUserPanelState(long buttonInfo,
VARIANT_BOOL visible, VARIANT_BOOL enabled,
VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.ShowAll

Deklaration:

```
Function ShowAll() as Boolean
```

```
HRESULT ShowAll(VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

Vergrößert oder verkleinert die aktuelle Ansicht so, daß das gesamte Projekt zu sehen ist.

ArCon.ShowAllButtonsInPanel

Deklaration:

```
Function ShowAllButtonsInPanel(ByVal panel as long,
ByVal visible as Boolean) as Boolean
```

```
HRESULT ShowAllButtonsInPanel(long panel,
VARIANT_BOOL visible, VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.ShowButtonByID

Deklaration:

```
Function ShowButtonByID(ByVal panel as long, ByVal ID
as long, ByVal SubID as long, ByVal visible as Boolean)
as Boolean
```

```
HRESULT ShowButtonByID(long panel, long ID, long
SubID, VARIANT_BOOL visible, VARIANT_BOOL*
retVal);
```

Siehe auch: ArCon Übersicht

ArCon.ShowMenu

Deklaration:

```
Function ShowMenu(ByVal mode as long, ByVal showIt
as Boolean) as Boolean
```

```
HRESULT ShowMenu(long mode, VARIANT_BOOL
showIt, VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.ShowPanel

Deklaration:

```
Function ShowPanel(ByVal panel as long, ByVal showIt
as Boolean) as Boolean
```

```
HRESULT ShowPanel(long panel, VARIANT_BOOL
showIt, VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.ShowWaitCursor

Deklaration:

```
Sub ShowWaitCursor(ByVal Hourglass as Boolean)
```

```
HRESULT ShowWaitCursor(VARIANT_BOOL Hourglass)
;
```

Siehe auch: ArCon Übersicht

ArCon.SpaceMouseAvailable

Deklaration:

```
Function SpaceMouseAvailable() as Boolean
```

```
HRESULT SpaceMouseAvailable(VARIANT_BOOL*
retVal);
```

Siehe auch: ArCon Übersicht

ArCon.StartDragFromDB

Deklaration:

```
Function StartDragFromDB(ByVal dbID as long, ByVal
itemID as long, ByVal dbType as AC_ArCon_DB_Types,
ByVal DragFlags as AC_ArCon_Drag_Modi, ByVal
DefaultZ as Single) as Boolean
```

```
HRESULT StartDragFromDB(long dbID, long itemID,
AC_ArCon_DB_Types dbType, AC_ArCon_Drag_Modi
DragFlags, float DefaultZ, VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.StartMe

Deklaration:

```
Function StartMe(ByVal hWnd as long, ByVal
helpFileName as String) as Boolean
```

```
HRESULT StartMe(long hWnd, BSTR helpFileName,
VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

Aktiviert ArCon, bzw. versucht eine Verbindung zu einem bereits laufenden ArCon herzustellen.

Parameter:

hWnd	Die Fensternummer des Hauptfensters Ihrer Applikation. Dieses Fenster muß während der gesamten Laufzeit des Makros existieren, kann aber unsichtbar sein.
HelpFile	Alle Benutzeroberflächenobjekte in ArCon haben kontextsensitive Hilfe. Falls Ihr Makro solche Elemente erzeugt, müssen Sie ArCon mitteilen, aus welcher Windows-Hilfedatei die Hilfe dafür stammt. Falls dies nicht der Fall ist, oder Sie (noch) keine Hilfedatei verfügbar haben, geben Sie hier einen leeren String an.

Falls ArCon vom Makro neu gestartet wurde, wird es bei diesem Aufruf sichtbar.

Ereignisse von ArCon werden erst nach dem Aufruf dieser Methode an das Makro versendet.

Makros, die eine Autostart-Funktionalität nutzen können (also von ArCon automatisch geladen werden, wenn ein Projekt mit Makro-spezifischen Daten gefunden wird), müssen sicherstellen, daß Sie nach ihrem Start innerhalb von ca. 45 Sekunden diese Methode aufrufen. Ansonsten bricht ArCon den Autostartversuch mit einer Fehlermeldung ab. Falls solche Makros längere Initialisierungsphasen benötigen, können diese nach dem Aufruf von “ StartMe” durchgeführt werden.

ArCon.StartMe2

Deklaration:

```
Function StartMe2(ByVal hWnd as long, ByVal
helpFileName as String, ByVal hidden as
Boolean, ByVal ParentWindow as long) as
Boolean
```

```
HRESULT StartMe2(long hWnd, BSTR
helpFileName, VARIANT_BOOL hidden, long
ParentWindow, VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

Aktiviert ArCon, bzw. versucht eine Verbindung zu einem bereits laufenden ArCon herzustellen.

Parameter:

hWnd	Die Fensternummer des Hauptfensters Ihrer Applikation. Dieses Fenster muß während der gesamten Laufzeit des Makros existieren, kann aber unsichtbar sein.
HelpFile	Alle Benutzeroberflächenobjekte in ArCon haben kontextsensitive Hilfe. Falls Ihr Makro solche Elemente erzeugt, müssen Sie ArCon mitteilen, aus welcher Windows-Hilfedatei die Hilfe dafür stammt. Falls dies nicht der Fall ist, oder Sie (noch) keine Hilfedatei verfügbar haben, geben Sie hier einen leeren String an.
Hidden	" Wahr" wenn ArCon das eigenene Hauptfenster noch nicht sichtbar machen soll.
ParentWindow	Fensternummer des Fensters, als dessenChild-Window ArCon sich installlieren soll (falls es jetzt schon sichtbar werden soll).

Im Gegensatz zur StartMe Methode bleibt ArCon bei dieser Aktivierung unsichtbar, falls es vom Makro neu gestartet wurde. Das Makro kann daraufhin weitere Initialisierungen durchführen und anschließend das ArCon Hauptfenster sichtbar machen, indem es die Eigenschaft " MultiUserMode" auf den Wert " ACMU_DEFAULT" setzt. Alternativ kann Ihr Makro später die Funktion SetParentWindow aufrufen.

ArCon.StartModalDialog

Deklaration:

```
Function StartModalDialog(ByVal hWnd as long)
as Boolean
```

```
HRESULT StartModalDialog(long hWnd,
VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

Zeigt einen Ihrer eigenen Dialoge als pseudo-modalen Dialog in ArCon an. Beispiele zur Verwendung finden Sie im Kapitel Eigene Dialoge

Durch den Aufruf dieser Funktion wird Ihr Dialog vor das ArCon Hauptfenster positioniert und das ArCon Hauptfenster gesperrt. Sie müssen diesen Zustand unbedingt beim Schließen Ihres Dialoges wieder aufheben, indem Sie die Funktion EndModalDialog aufrufen.

Parameter sind:

hWnd	Die Fensternummer Ihres Dialoges

ArCon.StartProgressbar

Deklaration:

```
Sub StartProgressbar(ByVal title as String, ByVal
percent as long)
```

```
HRESULT StartProgressbar(BSTR title, long percent);
```

Siehe auch: ArCon Übersicht

Zeigt einen Fortschrittsbalken in der Statuszeile an. Nach dem Abschluß Ihrer Aktion sollten Sie diesen durch einen Aufruf von StopProgressbar wieder entfernen.

Parameter sind:

title	der links neben der Fortschrittsanzeige eingeblendete Titel. Er sollte Ihre gerade ablaufende Aktion beschreiben (z.B. “ Projekt wird geladen”)
percent	initialer Prozentwert

ArCon.StopProgressbar

Deklaration:

```
Sub StopProgressbar()
```

```
HRESULT StopProgressbar();
```

Siehe auch: ArCon Übersicht

Beendet die Anzeige eines Fortschrittsbalkens, der mit der Funktion StartProgressbar erzeugt wurde.

ArCon.TextureToPicture

Deklaration:

```
Function TextureToPicture(ByVal texName as String,
ByRef texPicture as VARIANT) as Boolean
```

```
HRESULT TextureToPicture(BSTR texName,
VARIANT_BOOL *isBitmap, long *rgbColor, VARIANT *
texPicture, float *hSizeInMeter, float *vSizeInMeter,
VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.ThePrintSettings

Deklaration:

```
Function ThePrintSettings() as PrintSettings
```

```
HRESULT ThePrintSettings(IPrintSettings** retVal);
```

Siehe auch: ArCon Übersicht, PrintSettings

ArCon.TheSavePictureSettings

Deklaration:

```
Function TheSavePictureSettings() as
SavePictureSettings
```

```
HRESULT TheSavePictureSettings
(ISavePictureSettings** retVal);
```

Siehe auch: ArCon Übersicht, SavePictureSettings

ArCon.TheWalkSettings

Deklaration:

```
Function TheWalkSettings() as WalkSettings
```

```
HRESULT TheWalkSettings(IWalkSettings** retVal);
```

Siehe auch: ArCon Übersicht, WalkSettings

ArCon.TheZoomSettings

Deklaration:

```
Function TheZoomSettings() as ZoomSettings
```

```
HRESULT TheZoomSettings(IZoomSettings** retVal);
```

Siehe auch: ArCon Übersicht, ZoomSettings

ArCon.UpdateWindowPos

Deklaration:

```
Sub UpdateWindowPos()
```

```
HRESULT UpdateWindowPos();
```

Siehe auch: ArCon Übersicht

ArCon.VRDeviceAvailable

Deklaration:

```
Function VRDeviceAvailable() as Boolean
```

```
HRESULT VRDeviceAvailable(VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.Views

Deklaration:

```
Function Views() as ViewCollection
```

```
HRESULT Views(IViewCollection** retVal);
```

Siehe auch: ArCon Übersicht, ViewCollection

ArCon.WallName

Deklaration:

```
Function WallName(ByVal WallType as long) as String
```

```
HRESULT WallName(long WallType, BSTR* retVal);
```

Siehe auch: ArCon Übersicht

ArCon.WindowName

Deklaration:

```
Function WindowName(ByVal WindowTypeIndex as long)
as String
```

```
HRESULT WindowName(long WindowTypeIndex, BSTR*
retVal);
```

Siehe auch: ArCon Übersicht

Liefert den Namen eines geladenen Fenstertyps. Frei konstruierbare Fenster haben keine Namen. ArCon unterteilt Fenstertypen in konstruierte Fenster (Typ 0 bis ConstructedWindows - 1) und geladenene Fenster (Typ ConstructedWindows bis AvailableWindows - 1).

Parameter sind:

WindowTypeIndex	Der gewünschte Fenstertyp

Das folgende Beispiel füllt eine ComboBox mit den Namen aller geladenen Fenster und gleichzeitig das ItemData Array mit den zugehörigen Type-Codes:

```
Dim i As Long, max As Long
max = ArCon.AvailableWindows - 1
Combo.Clear
For i = ArCon.ConstructedWindows To max
   Combo.AddItem ArCon.WindowName(i)
   Combo.ItemData(Combo.NewIndex) = i
Next
```

ArCon.WriteChunk

Deklaration:

```
Function WriteChunk(ByVal ChunkID as
long, ByVal FileName as String) as Boolean
```

```
HRESULT WriteChunk(long ChunkID, BSTR
FileName, VARIANT_BOOL* retVal);
```

Siehe auch: ArCon Übersicht

Kopiert eine Datei in ein ArCon Projekt. Beispiele zum Laden und Speichern finden Sie im Kapitel Speichern und Laden .

Parameter sind:

ChunkID	Eine Kennzahl, mit der diese Daten beim Laden wieder identifiziert werden können. Entspricht einem “ Dateinamen” . Sie können einen beliebige Zahl aus dem Bereich AC_MinChunkID und AC_MaxChunkID verwenden.
FileName	Der Name der zu kopierenden Datei.

ButtonChange (Ereignis von ArCon)

Deklaration:

```
Sub ButtonChange(ByVal btnId as long, ByVal evnt as
long, ByVal selectedIndex as long, ByVal pressed as
Boolean, ByVal visible as Boolean, ByVal enabled as
Boolean)
```

```
void ButtonChange(long btnId, long evnt, long
selectedIndex, VARIANT_BOOL pressed,
VARIANT_BOOL visible, VARIANT_BOOL enabled);
```

Siehe auch: ArCon Übersicht

Dieses Ereignis tritt ein, wenn eine von Ihrem Makro erzeugte Schaltfläche benutzt wird. Mögliche Werte für den Parameter “ evnt” finden Sie im Kapitel Button Events.

Beispiele zur Programmierung mit Schaltflächen finden Sie unter Eigene Schaltflächen.

Die Parameter sind:

btnId	Die Kennzahl Ihres Buttons, der das Ereignis auslöst.
evnt	Eine Bitmenge, die angibt, um welche Ereignisse es sich handelt
selectedIndex	Falls es sich um einen Variantenschater (Multifunktionsknopf) handelt, gibt dieser Wert an, welche Variante gerade aktiv ist.
pressed	“ Wahr” wenn der Knopf momentan gedrückt ist
visible	“ Wahr” wenn der Knopf derzeit angezeigt werden darf
enabled	“ Wahr” wenn der Knopf drückbar ist

ChangeNotify (Ereignis von ArCon)

Deklaration:

```
Sub ChangeNotify(ByVal obj as Object, ByVal events
as long)
```

```
void ChangeNotify(IDispatch* obj, long events);
```

Siehe auch: ArCon Übersicht

Dieses Ereignis wird nur für Objekte ausgelöst, für die Ihr Makro sich mit Hilfe der Funktion NotifyOnChange angemeldet hat. Mit Hilfe der Ereigniskonstanten können Sie einzelne Bits der übergebenen Ereignisbitmaske prüfen.

Parameter sind:

obj	Das betroffene Objekt (für dieses Objekt haben Sie NotifyOnChange aufgerufen
events	Bitmaske aller aktuellen Events

DialogActivation (Ereignis von ArCon)

Deklaration:

```
Sub DialogActivation(ByVal dlgId as long, ByVal ArConDlgToken as
long, ByVal subObjectCount as long, ByVal activated as Boolean)
```

```
void DialogActivation(long dlgId, long ArConDlgToken, long
subObjectCount, VARIANT_BOOL activated);
```

Siehe auch: ArCon Übersicht

Dieses Ereignis wird ausgelöst, wenn Sie in einen existierenden ArCon Dialog eigene Dialogseiten integriert haben. Beispiele dazu finden Sie im Kapitel Vorhandene Dialoge erweitern.

ArCon hat entweder eine Ihrer Seiten aktiviert oder ist dabei, sie zu deaktivieren. Sie können in diesem Moment alle Abhängigkeiten zwischen Ihren Dialogseiten und Standardseiten abgleichen, indem Sie z.B. Einstellungen auf Ihrer Seite auswerten und entsprechende Veränderungen an Standardseiten vornehmen.

Parameter sind:

dlgId	Die Kennzahl des Dialoges
ArConDlgToken	Ein Token zur Weitergabe an GetDialogData oder SetDialogData.
SubObjectCount	Bei Dialogen mit mehreren Unterobjekten (z.B. der Dacheditor mit beliebig vielen Dachseiten) die Nummer des aktuellen Unterobjektes
activated	“ Wahr” , wenn Ihre Dialogseite aktiviert wird, “ Falsch” wenn sie deaktiviert wird

EndDialog (Ereignis von ArCon)

Deklaration:

```
Sub EndDialog(ByVal dialogID as long, ByVal
ok as Boolean)
```

```
void EndDialog(long dialogID, VARIANT_BOOL
ok);
```

Siehe auch: ArCon Übersicht

Ein ArCon Dialog wird geschlossen. Dies ist die letzte Gelegeneheit für Ihr Makro, Daten aus einer in diesen Dialog eingehängten Dialogseite zu speichern und Daten von anderen Dialogseiten abzufragen.

Dieses Ereignis wird von ArCon erst ausgelöst, nachdem Ihr Dialog schon unsichtbar und nicht mehr Bestandteil des ArCon Dialoges ist. Sie haben daher die volle Windows-Kontrolle über Ihren Dialog zurück und können Ihn z.B. in Ihrer Ereignisprozedur löschen.

Parameter sind:

dialogID	gibt an, um welchen Dialog es sich handelt
ok	" Wahr" wenn OK gedrückt wurde, " Falsch" bei Abbruch

Wichtig: Dieses Ereignis wird synchron abgearbeitet, ArCon und alle übrigen Makros warten, bis Ihre Ereignisprozedur beendet wurde. Sie dürfen daher in dieser Prozedur keine Message-Boxen öffnen oder ähnliche interaktive Vorgänge abwarten.

ExecuteMenuItem (Ereignis von ArCon)

Deklaration:

```
Sub ExecuteMenuItem(ByVal menuId as long)
```

```
void ExecuteMenuItem(long menuId);
```

Siehe auch: ArCon Übersicht

Der Benutzer hat einen von Ihnen erzeugten Menüpunkt aktiviert.

Parameter sind:

menuId	Der angewählte Menüpunkt

ExternalViewsVisibilityChanged (Ereignis von ArCon)

Deklaration:

```
Sub ExternalViewsVisibilityChanged(ByVal newVisibility
as long)
```

```
void ExternalViewsVisibilityChanged(long newVisibility);
```

Siehe auch: ArCon Übersicht

Die Sichtbarkeit der externen Hilfsmittel in ArCon (Objek/Datenbank-Explorer, Storyboard) hat sich geändert. Sie können diese Sichtbarkeit mit Hilfe der Funktion ArCon.SetExternalViews steuern.

Die Bitmasken für die einzelnen Ansichten finden Sie unter Externe Ansichten.

Parameter sind:

newVisibility	Neue Maske aller sichtbaren externen Ansichten.

GotOneClick (Ereignis von ArCon)

Deklaration:

```
Sub GotOneClick(ByVal valid as Boolean, ByVal x as Single, ByVal
y as Single, ByVal id as long)
```

```
void GotOneClick(VARIANT_BOOL valid, float x, float y, long id);
```

Siehe auch: ArCon Übersicht

Dieses Ereignis meldet einen mit der Methode GetOneClick angeforderten Mausklick zurück.

Parameter:

valid	Gibt an, ob die Eingabe gültig ist (TRUE), oder der Benutzer den Vorgang abgebrochen hat (FALSE)
x	Die x-Koordinate des angeklickten Punktes, falls die Eingabe gültig ist.
y	Die y-Koordinate.
id	Die beim Aufruf von GetOneClick angegebene Zahl.

Graphics2DDeleted (Ereignis von ArCon)

Deklaration:

```
Sub Graphics2DDeleted(ByVal obj as Object, ByRef
allowDelete as Boolean)
```

```
void Graphics2DDeleted(IDispatch* obj, VARIANT_BOOL *
allowDelete);
```

Siehe auch: ArCon Übersicht

Ein 2D Element wird vom Benutzer gelöscht.

Parameter:

obj	Das gelöschte Objekt
allowDelete	Setzen Sie diesen Wert auf Falsch, um das Löschen zu verbieten

Graphics2DDoubleClick (Ereignis von ArCon)

Deklaration:

```
Sub Graphics2DDoubleClick(ByVal obj as
Object)
```

```
void Graphics2DDoubleClick(IDispatch* obj);
```

Siehe auch: ArCon Übersicht

Ein 2D Graphikelement wurde doppelt angeklickt.

Parameter:

obj	Das angeklickte Graphikelement

Graphics2DEndMoving (Ereignis von ArCon)

Deklaration:

```
Sub Graphics2DEndMoving(ByVal obj as Object,
ByVal aborted as Boolean, ByVal extended as
Boolean, ByVal copied as Boolean)
```

```
void Graphics2DEndMoving(IDispatch* obj,
VARIANT_BOOL aborted, VARIANT_BOOL
extended, VARIANT_BOOL copied);
```

Siehe auch: ArCon Übersicht

Ein 2D Element beendet einen Verschiebe- oder Skalierungsvorgang

Parameter sind:

obj	Das bewegte Objekt
aborted	Wahr, wenn der Vorgang mit ESC abgebrochen wurde (das Objekt hat wieder seine ursprüngliche Position eingenommen)
extended	Wahr, wenn die Erweiterungs (Umschalttaste) gedrückt war
copied	Wahr, wenn die Kopiertaste (Strg) gedrückt war

Graphics2DMove (Ereignis von ArCon)

Deklaration:

```
Sub Graphics2DMove(ByVal obj as Object,
ByVal x as Single, ByVal y as Single, ByVal
viewHandle as long)
```

```
void Graphics2DMove(IDispatch* obj, float x,
float y, long viewHandle);
```

Siehe auch: ArCon Übersicht

Ein gerade verschobenens/skaliertes 2D Element liefert neue Koordinaten.

Achtung: bei gedrehten Objekten werden die Koordinaten relativ zum ungedrehten Zustand geliefert!

Parameter sind:

obj	Das verschobene Objekt
x, y	Die aktuelle Position
viewHandle	Kennung des Fensters, in dem die Verschiebung durchgeführt wird

Graphics2DObjectTransform (Ereignis von ArCon)

Deklaration:

```
Sub Graphics2DObjectTransform(ByVal obj2D
as Object, ByRef matrix as VARIANT)
```

```
void Graphics2DObjectTransform(IDispatch*
obj2D, VARIANT* matrix);
```

Siehe auch: ArCon Übersicht

Ein Gebäude wird gedreht oder gespiegelt. Da eventuell einige der von Ihrem Makro erzeugten 2D Objekte dabei ebenfalls verändert werden müssen (falls Sie zur Hauskonstruktion gehören), teilt ArCon in diesem Ereignis die nötigen Informationen mit.

Die Sichtbarkeit der externen Hilfsmittel in ArCon (Objek/Datenbank-Explorer, Storyboard) hat sich geändert. Sie können diese Sichtbarkeit mit Hilfe der Funktion ArCon.SetExternalViews steuern.

Die Bitmasken für die einzelnen Ansichten finden Sie unter Externe Ansichten.

Parameter sind:

obj2D	das zu betrachtende 2D Graphikobjekt
matrix	die 2D Transformationsmatrix (eine homogene 3x3 Matrix, vergleiche auch die Informationen zu 4x4 Matrix)

Graphics2DSelectionChanged (Ereignis von ArCon)

Deklaration:

```
Sub Graphics2DSelectionChanged(ByVal obj as Object, ByVal x as
Single, ByVal y as Single, ByVal selected as Boolean, ByVal
extended as Boolean, ByVal hasAlreadyBeenSelected as Boolean,
ByVal pixelScale as Single, ByVal snapRadius as long)
```

```
void Graphics2DSelectionChanged(IDispatch* obj, float x, float y,
VARIANT_BOOL selected, VARIANT_BOOL extended,
VARIANT_BOOL hasAlreadyBeenSelected, float pixelScale, long
snapRadius);
```

Siehe auch: ArCon Übersicht

Dieses Ereignis wird ausgelöst, sobald ein 2D Graphikelement seinen Selektionszustand wechselt. Wenn die Selektion von einem Element zu einem anderen wechselt, löst ArCon zunächst das Selektionsereignis für das neu selektierte Objekt aus, anschließend das Deselektionsereignis für das jetzt nicht mehr selektierte Objekt.

Mit Hilfe der übergebenen Parameter ist es möglich, den Selektionspunkt innerhalb eines komplexeren Objektes genauer zu bestimmen, falls dies erforderlich ist (z.B. die Selektionsgriffe zum Verschieben oder Skalieren von Objekten in einem 2D Editor)

Parameter:

obj	Das 2D Graphikelement, das gerade selektiert oder deselektiert wird
x, y	Koordinaten des Mausklicks, der zu der Selektionsänderung führte
selected	Wahr, wenn dieses Objekt jetzt selektiert ist, Falsch wenn es nicht mehr selektiert ist
extended	Wahr, wenn eine Selektion erweitert wird (Mehrfachselektion mit gedrückter Umschalttaste)
hasAlreadyBeenSelected	Der Selektionszustand des Objektes vor diesem Ereignis. ArCon setzt die Eigenschaft “ selected” des Grafikelementes vor der Auslösung dieses Ereignises auf den Wert des “ selected” Parameters. Dieser Parameter liefert den Wert vor dieser Änderung.
pixelScale	Gibt die Größe eines Bildschirmpunktes in Weltkoordinaten an
snapRadius	die eingestellte Fang-Entfernung in Bildschirmpunkten

Graphics2DStartMoving (Ereignis von ArCon)

Deklaration:

```
Sub Graphics2DStartMoving(ByVal obj as Object,
ByVal x as Single, ByVal y as Single, ByVal
pixelFactor as Single, ByVal extended as
Boolean, ByVal markerIndex as long, ByRef
allowMove as Boolean)
```

```
void Graphics2DStartMoving(IDispatch* obj, float
x, float y, float pixelFactor, VARIANT_BOOL
extended, long markerIndex, VARIANT_BOOL *
allowMove);
```

Siehe auch: ArCon Übersicht

Ein 2D Element beginnt einen Verschiebe/Skalierungsvorgang

Parameter sind:

obj	Das verschobene Objekt
x, y	Die Startposition, an der das Objekt “ angefaßt” wurde
pixelFaktor	Die Größe eines Bildschirmpunktes in Weltkoordinaten (diesen Wert benötigen Sie eventuell, um x/y zu interpretieren)
extended	Wahr, wenn die Erweiterungstaste (Umschalten) beim Start der Bewegung gedrückt war
markerIndex	Nummer des Griffes, an dem das Objekt angefaßt wurde
allowMove	Setzen Sie diesen Wert auf Falsch, um die Verschiebung/ Skalierung zu verbieten.

Der Wert von “ markerIndex” ist nur gültig, wenn das Objekt Markierungskästchen anzeigt (siehe SetMarks). Bei Polygon-Markierungen oder Objekten mit benutzerdefinierten Markierungspunkten gilt:

0	Das ganze Objekt wird verschoben
i	Der i’ te Punkt wurde angefaßt

Bei Linien gilt:

0	Die gesamte Linie wird verschoben
1	Der Punkt X1/Y1 wird verschoben
2	Der Punkt X2/Y2 wird verschoben

Bei Objekten mit linear skalierten Umhüllungspunkten oder Standard-Umhüllungspunkten gilt:

0	Das ganze Objekt wird verschoben,
1	der Punkt oben links
2	die linke Seite
3	der Punkt unten links
4	die untere Seite
5	der Punkt rechts unten
6	die rechte Seite

7	der Punkt oben rechts
8	die obere Seite

wird verschoben. Bei linearer Skalierung können die Werte 2, 4, 6 und 8 nicht auftreten.

Graphics2DStoryHeightsChanged (Ereignis von ArCon)

Deklaration:

```
Sub Graphics2DStoryHeightsChanged(ByVal
obj2D as Object)
```

```
void Graphics2DStoryHeightsChanged
(IDispatch* obj2D);
```

Siehe auch: ArCon Übersicht

Die Höhe des Stockwerkes oder der darunterliegenden Stockwerke wurde geändert. Eventuell muß Ihr 2D Graphikobjekt entsprechend Verschoben werden, z.B. wenn es einen Teil der Gebäudegeometrie darstellen soll.

Parameter sind:

obj2D	das zu betrachtende 2D Graphikobjekt

HowInput (Ereignis von ArCon)

Deklaration:

```
Sub HowInput(ByVal btnId as long, ByVal x as Single, ByVal y as
Single, ByVal dirX as Single, ByVal dirY as Single, ByVal state as
long, ByVal CutView as Object)
```

```
void HowInput(long btnId, float x, float y, float dirX, float dirY, long
state, IDispatch* CutView);
```

Siehe auch: ArCon Übersicht

Eine automatische Schaltfläche in der " Wie" -Leiste ist aktiv (siehe auch ArCon.RunningTool) und der Benutzer hat einen Punkt eingegeben oder die Eingabe abgebrochen. Details können Sie dem " state" Parameter entnehmen, mögliche Werte dafür finden Sie im Kapitel How-Button Input Events.

Beispiele für die Programmierung finden Sie unter Automatische Schaltflächen

Parameter sind:

btnId	Die Kennzahl der Schaltfläche des aktiven Wie-Werkzeuges
x, y	aktuelle Mausposition in Weltkoordinaten
dirX, dirY	normierter Tangentenvektor eines Fangpunktes, falls das Fangobjekt eine Richtung definiert
state	Kennzahl für den Grund des Ereignisses
CutView	Falls Ihr Tool in einem Schnitt ablief, ein Cut, im Normalfall (ausserhalb eines Schnittes) ist dieser Parameter " Nothing" (Nil, NULL).

HowMove (Ereignis von ArCon)

Deklaration:

```
Sub HowMove(ByVal btnId as long, ByVal x as
Single, ByVal y as Single, ByVal viewHandle as
long)
```

```
void HowMove(long btnId, float x, float y, long
viewHandle);
```

Siehe auch: ArCon Übersicht

Eine automatische Schaltfläche in der “ Wie” -Leiste ist aktiv (siehe auch ArCon.RunningTool) und der Benutzer hat die Maus bewegt.

Beispiele für die Programmierung finden Sie unter Automatische Schaltflächen.

Parameter sind:

btnId	Die Kennzahl der aktiven automatischen Schaltfläche
x, y	aktuelle Mauskoordinaten
viewHandle	reserviert

HowSnap (Ereignis von ArCon)

Deklaration:

```
Sub HowSnap(ByVal btnId as long, ByVal x as Single,
ByVal y as Single, ByVal viewHandle as long, ByVal
snapDistance as Single)
```

```
void HowSnap(long btnId, float x, float y, long
viewHandle, float snapDistance);
```

Siehe auch: ArCon Übersicht

Ein automatischer Variantenschalter in der How-Leiste ist aktiv und der Benutzer bewegt die Maus. Ihr Makro kann innerhalb der Event-Prozedur durch Aufruf der Methode ArCon. RunningTool.AddSnapPoint eine eigene Fang-Funktion implementieren und so auf von Ihnen selbst verwaltete Eigenschaften oder Objekte fangen, die ArCon (noch) nicht kennt.

Dieses Ereignis wird nur ausgelöst, wenn Sie im ButtonInfo Parameter der DefButton Methode das Flag AC_HowCustomSnap gesetzt haben.

Parameter sind:

btnId	der gerade aktive automatische Variantenschalter
x, y	die Mauskoordinaten
viewHandle	Eine Identifikation der aktuellen Ansicht
snapDistance	die aktuell eingestellte Fangentfernung

HowStatusLineRedraw (Ereignis von ArCon)

Deklaration:

```
Sub HowStatusLineRedraw(ByVal btnId as long,
ByVal x as Single, ByVal y as Single, ByVal
viewHandle as long)
```

```
void HowStatusLineRedraw(long btnId, float x, float
y, long viewHandle);
```

Siehe auch: ArCon Übersicht

Ein automatischer Variantenschalter in der Wie-Leiste ist aktiv, bei dessen Definition sie das Flag AC_HowCustomStatusInfo im Parameter ButtonInfo gesetzt haben. Ihre Ereignisprozedur sollte einen Hinweistext zur aktuellen Mausposition in der Statuszeile ausgeben.

Parameter sind:

btnId	Der aktive Variantenschalter
x, y	Die aktuelle Mausposition
viewHandle	Eine Identifikation der aktuellen Ansicht

InputModeChanged (Ereignis von ArCon)

Deklaration:

```
Sub InputModeChanged(ByVal newMode as long)
```

```
void InputModeChanged(long newMode);
```

Siehe auch: ArCon Übersicht

Das aktuelle Werkzeug wechselt (s. SetInputMode).

LoadBuilding (Ereignis von ArCon)

Deklaration:

```
Sub LoadBuilding(ByVal aBuilding as Object,
ByVal SaveNo as long)
```

```
void LoadBuilding(IDispatch* aBuilding, long
SaveNo);
```

Siehe auch: ArCon Übersicht

ArCon lädt gerade ein Projekt und bietet Ihnen nun die Möglichkeit, benutzerdefinierte Daten aus diesem Projekt zu übernehmen.

Siehe auch: Beispielprogramm.

Die Parameter sind:

aBuilding	das gerade geladene Gebäude oder “ Nothing” , wenn gerade globale Daten geladen werden.
SaveNo	die Speicherpositionsnummer dieses Gebäudes im SaveBuilding Ereignis

LoadComplete (Ereignis von ArCon)

Deklaration:

```
Sub LoadComplete(ByVal successfull as Boolean)
```

```
void LoadComplete(VARIANT_BOOL successfull);
```

Siehe auch: ArCon Übersicht

ArCon hat einen Ladevorgang abgeschlossen.

Siehe auch: Beispielprogramm.

Parameter sind:

successfull	“ Wahr” wenn alle Bestandteile des Projektes erfolgreich geladen wurden, “ Falsch” bei Abbruch durch den Benutzer

LoadDialog (Ereignis von ArCon)

Deklaration:

```
Sub LoadDialog(ByVal dialogID as long, ByVal
ArConDlgToken as long, ByVal dialogObject as Object)
```

```
void LoadDialog(long dialogID, long ArConDlgToken,
IDispatch* dialogObject);
```

Siehe auch: ArCon Übersicht

Dieses Ereignis wird ausgelöst, wenn ArCon+ einen Dialog öffnet. Ihr Makro kann daraufhin diesen Dialog um eigene Dialogseiten erweitern. Ein Beispiel dazu finden Sie im Kapitel Vorhandene Dialoge erweitern.

Parameter sind:

dialogId	Die Kennzahl des sich gerade öffnenden Dialogs
ArConDlgToken	Ein Token zur Weitergabe an die GetDialogData oder SetDialogData Funktionen.
DialogObject	das vom Dialog bearbeitete Objekt, z.B. ein Roof Objekt, wenn es sich um den Dach-Editor handelt. Dieser Parameter ist “ Nothing” wenn die Standardeinstellungen des jeweiligen Objektes bearbeitet werden.

LoadStart (Ereignis von ArCon)

Deklaration:

```
Sub LoadStart()
```

```
void LoadStart();
```

Siehe auch: ArCon Übersicht

ArCon beginnt mit dem Laden eines Projektes und gibt Ihnen hiermit die Möglichkeit, Datenstrukturen vorzubereiten.

Siehe auch: Beispielprogramm.

NewCurrentProject (Ereignis von ArCon)

Deklaration:

```
Sub NewCurrentProject()
```

```
void NewCurrentProject();
```

Siehe auch: ArCon Übersicht

Das aktuelle Projekt wechsel (ArCon.CurrentProjekt hat einen andere Wert). Entweder hat der Benutzer ein neues Projekt erstellt oder ein altes geladen.

Object3DDoubleClicked (Ereignis von ArCon)

Deklaration:

```
Sub Object3DDoubleClicked(ByVal obj3D as
Object, ByVal obj2D as Object, ByRef
redrawObject as Boolean, ByRef redrawAll as
Boolean)
```

```
void Object3DDoubleClicked(IDispatch* obj3D,
IDispatch* obj2D, VARIANT_BOOL *
redrawObject, VARIANT_BOOL * redrawAll);
```

Siehe auch: ArCon Übersicht

Ein 3D-Objekt, das an ein 2D-Objekt gebunden ist (siehe Object3D.SetOutline2D) wurde doppelt geklickt.

Parameter sind:

obj3D	das 3D Objekt (Object3D)
obj2D	das zugehörige 2D Objekt (z.B. Polygon)
redrawObject	setzen Sie diesen Parameter auf “ Wahr” , um das doppeltgeklickte Objekt neu zeichnen zu lassen
redrawAll	setzen Sie diesen Parameter auf “ Wahr” , um das gesamte Bild neu rendern zu lassen.

Object3DInserted (Ereignis von ArCon)

Deklaration:

```
Sub Object3DInserted(ByVal Obj as Object3D, ByVal SnapWallSeg
as WallSegment, ByVal SnapObj as Object3D)
```

```
void Object3DInserted(IObject3D * Obj, IWallSegment *
SnapWallSeg, IObject3D * SnapObj, VARIANT *Position,
VARIANT_BOOL *PositionChanged);
```

Siehe auch: ArCon Übersicht, Object3D, WallSegment, Object3D

Ein neues 3D-Objekt wurde vom Benutzer in die ArCon-Welt eingefügt und ist gegebenenfalls (nach unten oder nach oben) gefallen. Seine endgültige Position steht fest und kann nun von Ihnen noch beeinflußt werden.

Sie erhalten dieses Ereignis nur, wenn Sie die mit Hilfe von ArCon.ChangeTypeNotifyMask den AC_CHANGE_Inserted Event für den Typ Object3D aktiviert haben.

Parameter sind:

Obj	das neu eingfügte 3D Objekt
snapWallSeg	falls beim Einfügen/Fallen das Objekt von einer Wand gefangen wurde, das entsprechende Wandsegment.
snapObj	falls beim Einfügen/Fallen das Objekt von einem anderen 3D Objekt gefangen wurde, dieses Objekt
Position	Eine Referenz auf eine 4x4 Matrix zur Bestimmung der Einfügeposition. Wenn Sie diese Position beeinflussen möchten, ändern Sie die entsprechenden Einträge der Matrix
PositionChanged	setzen Sie diesen Parameter auf “ Wahr” , wenn die von Ihnen geänderte Matrix zur Bestimmung der tatsächlichen Position des Objektes übernommen werden soll.

ProgramExit (Ereignis von ArCon)

Deklaration:

```
Sub ProgramExit()
```

```
void ProgramExit();
```

Siehe auch: ArCon Übersicht

Dieses Ereignis wird ausgelöst, wenn ArCon beendet wurde. Ihre Anwendung wird sich daraufhin in der Regel auch beenden, eventuell nachdem noch nicht gespeicherte Daten gesichert wurden.

Sie können keine ArCon Funktionalitäten mehr verwenden.

ProjectChange (Ereignis von ArCon)

Deklaration:

```
Sub ProjectChange()
```

```
void ProjectChange();
```

Siehe auch: ArCon Übersicht

Projektspezifische Daten haben sich geändert (siehe Project). Dieses Ereignis tritt z.B. beim Verändern des Ursprungs oder des Maßstabes auf.

ProjectClosed (Ereignis von ArCon)

Deklaration:

```
Sub ProjectClosed()
```

```
void ProjectClosed();
```

Siehe auch: ArCon Übersicht

Dieses Ereignis wird von ArCon ausgelöst, sobald das aktuelle Projekt geschlossen wird.

RoofDoubleClicked (Ereignis von ArCon)

Deklaration:

Sub RoofDoubleClicked()

void RoofDoubleClicked(VARIANT_BOOL *showDialog);

Siehe auch: ArCon Übersicht

Ein von Ihrem Makro erzeugtes Dach wurde doppeltgeklickt.

Parameter sind:

showDialog	setzen Sie diesen Parameter auf “ Wahr” , wenn der Standard-Dacheditor angezeigt werden soll.

SaveBuilding (Ereignis von ArCon)

Deklaration:

```
Sub SaveBuilding(ByVal BuildingNo as long, ByVal aBuilding as
Object)
```

```
void SaveBuilding(long BuildingNo, IDispatch* aBuilding);
```

Siehe auch: ArCon Übersicht

ArCon speichert gerade ein Projekt und gibt Ihnen die Möglichkeit, benutzerdefinierte Daten für dieses Gebäude zu sichern. Da die Reihenfolge von Gebäuden willkürlich ist, erhält jedes Gebäude beim Speichern eine Nummer zugeteilt, die beim Laden wieder mit übergeben wird.

Siehe auch: Beispielprogramm.

Parameter sind:

BuildingNo	Eine Speicherplatznummer, anhand derer das Gebäude beim nächsten Laden wieder identifiziert werden kann. Achtung: diese Nummern sind nur bis zum nächsten Speichern gültig!
aBuilding	Das gespeicherte Gebäude oder “ Nothing” , wenn globale Daten gespeichert werden.

SaveDialogDefaults (Ereignis von ArCon)

Deklaration:

```
Sub SaveDialogDefaults(ByVal dlgId as long, ByVal token
as long)
```

```
void SaveDialogDefaults(long dlgId, long token);
```

Siehe auch: ArCon Übersicht

In einem ArCon Dialog, den Sie um eigene Dialogseiten erweitert haben, wurde der “ Als Standard” Knopf benutzt, um die aktuellen Einstellungen als Standardwerte zu speichern. ArCon kann dies nur für die Standarddialogdaten erledigen und löst daher dieses Ereignis aus.

Die Parameter sind:

dlgId	Die Kennzahl des Dialoges
token	Ein ArCon Dialog-Token zur Weitergabe an GetDialogData.

SaveStart (Ereignis von ArCon)

Deklaration:

```
Sub SaveStart(ByVal FileName as String, ByVal
NumBuildings as long, ByVal isAutoSave as Boolean,
ByRef numChunksToSave as long)
```

```
void SaveStart(BSTR FileName, long NumBuildings,
VARIANT_BOOL isAutoSave, long* numChunksToSave);
```

Siehe auch: ArCon Übersicht

ArCon beginnt mit dem Speichern eines Projektes. Falls Sie benutzerdefinierte Daten speichern möchten, müssen Sie den Referenzparameter “ numChunksToSave” entsprechend erhöhen, damit ArCon Ihnen Speicherplatz für Ihre Daten reserviert.

Siehe auch: Beispielprogramm.

Die Parameter sind:

fileName	Der Dateiname, in den gespeichert wird
NumBuildings	Wieviele Gebäude werden gespeichert
isAutoSave	“ Wahr” , wenn es sich um eine automatisch erstellte Sicherungskopie handelt (keine Benutzerinteraktion in diesem Fall!)
numChunksToSave	In diesem Referenzparameter wird gesammelt, wieviele benutzerdefinerte Chunks dem Projekt hinzugefügt werden. Setzen Sie diesen Wert niemals auf absolute Zahlen, inkrementieren Sie ihn entsprechend!

TextureDropped (Ereignis von ArCon)

Deklaration:

```
Sub TextureDropped(ByVal obj2D as Object,
ByVal obj3D as Object, ByVal matIndex as
long, ByVal objTexture as Object, ByVal
droppedTextureName as String, ByVal x as
Single, ByVal y as Single, ByVal z as Single,
ByRef redraw as Boolean, ByRef
redrawWholeScene as Boolean, ByRef
autoExchangeTexture as Boolean)
```

```
void TextureDropped(IDispatch* obj2D,
IDispatch* obj3D, long matIndex, IDispatch*
objTexture, BSTR droppedTextureName, float
x, float y, float z, VARIANT_BOOL * redraw,
VARIANT_BOOL * redrawWholeScene,
VARIANT_BOOL * autoExchangeTexture);
```

Siehe auch: ArCon Übersicht

Ein an ein 2D Objekt gebundenes 3D Objekt (siehe Object3D.SetOutline2D) bekommt per Drag & Drop eine neue Textur.

Parameter sind:

obj2D	das 2D Graphikobjekt (z.B. Polygon)
obj3D	das 3D Objekt (Object3D)
matIndex	der Materialindex der ausgetauschten Textur (je nach Art der Konstruktion des 3D Objektes können Sie daraus erkennen, welcher Teil des Objektes betroffen ist)
objTexture	ein Verweis auf die Texturliste des Objektes (TextureName), mit dem Sie eine andere Textur als die gedroppte setzen können. Außerdem liefert dieses Objekt den alten Texturnamen (falls verfügbar)
droppedTextureName	Name der neuen Textur
x, y, z	Position des Drop-Ereignisses
redraw	setzen Sie diesen Parameter auf “ Wahr” , um das Objekt neu zu zeichnen
redrawWholeScene	setzen Sie diesen Parameter auf “ Wahr” , um die gesamte Anzeige neu zu zeichnen.
autoExchangeTexture	setzen Sie diesen Parameter auf “ Wahr” , um den Texturaustausch automatisch von ArCon erledigen zu lassen.

UserPanelAdjustSize (Ereignis von ArCon)

Deklaration:

```
Sub UserPanelAdjustSize(ByVal ButtonInfo as long)
```

```
void UserPanelAdjustSize(long ButtonInfo, long *width, long *
height, VARIANT_BOOL *ok);
```

Siehe auch: ArCon Übersicht

Ein von Ihnen definiertes Panel verändert seine Größe (z.B. weil das Elternfenster verkleinert wurde). Sie können hier die Größe des Panels beeinflußen (z.B. um Minimalwerte nicht zu unterschreiten).

Parameter sind:

ButtonInfo	die ID des Panels
widht, height	setzen Sie diese Parameter auf die gewünschte Größe – oder lassen Sie sie unverändert
ok	setzen Sie diesen Paramneter auf “ Wahr” wenn die neue Größe des Panels akzeptabel ist.

UserPanelPosChanged (Ereignis von ArCon)

Deklaration:

```
Sub UserPanelPosChanged(ByVal ButtonInfo as long, ByVal left as
long, ByVal top as long, ByVal right as long, ByVal bottom as long)
```

```
void UserPanelPosChanged(long ButtonInfo, long left, long top,
long right, long bottom);
```

Siehe auch: ArCon Übersicht

Ein von Ihnen definiertes Panel wird verschoben.

Parameter sind:

ButtonInfo	die ID des Panels
left, top	obere linke Ecke des Fensters
bottom, right	untere und rechte Kante des Fensters

UserPanelRightClicked (Ereignis von ArCon)

Deklaration:

```
Sub UserPanelRightClicked(ByVal ButtonInfo as long)
```

```
void UserPanelRightClicked(long ButtonInfo);
```

Siehe auch: ArCon Übersicht

Ein von Ihnen definiertes Panel wurde rechtsgeklickt.

Parameter sind:

ButtonInfo	die ID des Panels

WorldObject3DDoubleClicked (Ereignis von ArCon)

Deklaration:

```
Sub WorldObject3DDoubleClicked(ByVal selObj as Object3D,
ByVal clickedObj as Object3D, ByVal objectPartID as long)
```

```
void WorldObject3DDoubleClicked(IObject3D * selObj, IObject3D *
clickedObj, long objectPartID, VARIANT_BOOL *Modified);
```

Siehe auch: ArCon Übersicht, Object3D, Object3D

Ein 3D Objekt in der ArCon Welt wurde doppeltgeklickt.

Um dieses Ereignis zu erhalten, müssen Sie für das entsprechende 3D Objekt die Funktion ArCon.SetObject3DEventMask benutzen um den Event ACO3D_EVENT_DBLCLK (siehe 3D Objekt-Ereignisse) anzufordern.

Parameter sind:

selObj	das derzeit selektierte Objekt. Diesem Objekt gilt der Doppelklick. Es kann sich dabei um eine Gruppe handeln.
clickedObj	das eigentliche angeklickte Objekt. Falls es sich bei selObj nicht um eine Gruppe handelt, ist clickedObj und selObj identisch. Andernfalls gibt clickedObj das Teilobjekt der Grupp an, das tatsächlich vom Benutzer angeklickt wurde.
objectPartID	Teil des Objektes, der angeklcikt wurde. Wenn Sie z.B. ein Objekt mit Hilfe eines ObjectConstructors per Programm erzeugen, können Sie neuen Flächen einen Kontext geben, den Sie hier übergeben bekommen. Siehe auch ObjectConstructor.SetContext.
Modified	Setzen Sie diesen Parameter auf “ Wahr” , wenn die Gruppe oder das Objekt sich in Ihrer Ereignisverarbeitung verändert hat und die Scene neu gezeichnet werden muß (z.B. wenn Sie einen Teil der Gruppe ausgetauscht haben).

WorldObject3DMaterialDropped (Ereignis von ArCon)

Deklaration:

```
Sub WorldObject3DMaterialDropped(ByVal obj as Object3D, ByVal
event as long, ByVal hitX as Single, ByVal hitY as Single, ByVal
hitZ as Single, ByVal pickedMat as Material, ByVal objectPartID as
long, ByVal oldMat as Material, ByVal newMat as Material)
```

```
void WorldObject3DMaterialDropped(IObject3D * obj, long event,
float hitX, float hitY, float hitZ, IMaterial * pickedMat, long
objectPartID, IMaterial * oldMat, IMaterial * newMat,
VARIANT_BOOL *mayDrop);
```

Siehe auch: ArCon Übersicht, Object3D, Material, Material, Material

Ein 3D Objekt in der ArCon Welt erhält ein neues Material.

Um dieses Ereignis zu erhalten, müssen Sie für das entsprechende 3D Objekt die Funktion ArCon.SetObject3DeventMask benutzen um den Event ACO3D_EVENT_MATERIAL_DROPPED (siehe 3D Objekt-Ereignisse) anzufordern.

Parameter sind:

Obj	das betroffene Objekt
event	Art des Ereignisses, das zum Texturaustausch führte. Mögliche Werte sind AC_DRAG_N_DROP_DRAG (die Maus wird über das Objekt gezogen, „ mayDrop“ schaltet zwischen Verbots- und Erlaubniscursor um), AC_DRAG_N_DROP_DROP (die Textur wurde fallengelassen, „ mayDrop“ bestimmt, ob der Austausch vorgenommen wird), AC_DRAG_N_DROP_UNDO sowie AC_DRAG_N_DROP_REDO (Undo bzw. Redo – „ mayDrop“ wird nicht ausgewertet)
hitX, hitY, hitZ	die Weltkoordinaten der Mausposition
pickedMat	das (derzeitige) Material, auf das die Maus zeigt
objectPartID	Objektspezifischen Kontext (Teil des Objektes, auf den die Maus zeigt). Siehe auch ObjectConstructor.SetContext
oldMat	zu ersetzendes Material
newMat	das neue Material
mayDrop	setzen Sie diesen Parameter auf “ Wahr” , wenn der Materialaustausch durchgeführt werden darf.

WorldObject3DMoved (Ereignis von ArCon)

Deklaration:

```
Sub WorldObject3DMoved(ByVal obj as Object3D, ByVal
snappingWallSeg as WallSegment, ByVal snappingObject as
Object3D, ByVal MC_WC as VARIANT)
```

```
void WorldObject3DMoved(IObject3D * obj, IWallSegment *
snappingWallSeg, IObject3D * snappingObject, VARIANT MC_WC)
;
```

Siehe auch: ArCon Übersicht, Object3D, WallSegment, Object3D

Ein 3D Objekt in der ArCon Welt wurde verschoben.

Um dieses Ereignis zu erhalten, müssen Sie für das entsprechende 3D Objekt die Funktion ArCon.SetObject3DeventMask benutzen um den Event ACO3D_EVENT_MOVED (siehe 3D Objekt-Ereignisse) anzufordern.

Parameter sind:

obj	das betroffene Objekt
snappingWallSeg	falls das Objekt von einer Wand gefangen wurde, das entsprechende Wandsegment
snappingObject	falls das Objekt von einem anderen Objekt gefangen wurde, dieses andere Objekt
MC_WC	Transformationsmatrix von Modell- in Weltkoordinaten. Hierbei handelt es sich um eine 4x4 Matrix.

WorldObject3DTextureDropped (Ereignis von ArCon)

Deklaration:

```
Sub WorldObject3DTextureDropped(ByVal obj as Object3D, ByVal
event as long, ByVal hitX as Single, ByVal hitY as Single, ByVal
hitZ as Single, ByVal pickedMat as Material, ByVal objectPartID as
long, ByVal oldTexName as String, ByVal newTexName as String)
```

```
void WorldObject3DTextureDropped(IObject3D * obj, long event,
float hitX, float hitY, float hitZ, IMaterial * pickedMat, long
objectPartID, BSTR oldTexName, BSTR newTexName,
VARIANT_BOOL *mayDrop);
```

Siehe auch: ArCon Übersicht, Object3D, Material

Ein 3D Objekt in der ArCon Welt erhält eine neue Textur.

Um dieses Ereignis zu erhalten, müssen Sie für das entsprechende 3D Objekt die Funktion ArCon.SetObject3DeventMask benutzen um den Event ACO3D_EVENT_MATERIAL_DROPPED (siehe 3D Objekt-Ereignisse) anzufordern.

Parameter sind:

Obj	das betroffene Objekt
event	Art des Ereignisses, das zum Texturaustausch führte. Mögliche Werte sind AC_DRAG_N_DROP_DRAG (die Maus wird über das Objekt gezogen, „ mayDrop“ schaltet zwischen Verbots- und Erlaubniscursor um), AC_DRAG_N_DROP_DROP (die Textur wurde fallengelassen, „ mayDrop“ bestimmt, ob der Austausch vorgenommen wird), AC_DRAG_N_DROP_UNDO sowie AC_DRAG_N_DROP_REDO (Undo bzw. Redo – „ mayDrop“ wird nicht ausgewertet)
hitX, hitY, hitZ	die Weltkoordinaten der Mausposition
pickedMat	das (derzeitige) Material, auf das die Maus zeigt
objectPartID	Objektspezifischen Kontext (Teil des Objektes, auf den die Maus zeigt). Siehe auch ObjectConstructor.SetContext
oldTexNamet	der Name der alten Textur
newTexName	der Name der neuen Textur
mayDrop	setzen Sie diesen Parameter auf “ Wahr” , wenn der Materialaustausch durchgeführt werden darf.

Room

Ein Raum in der ArCon Planung. Anders als die konstruktiven Elemente können Sie einen Raum nicht direkt erzeugen. Er entsteht automatisch durch die Plazierung von Wänden. Sie können Räume durch entsprechende Listen des Stockwerks oder die Story.FindRoom Methode erhalten.

Room Übersicht

Eigenschaften

Angle	Drehwinkel der Raumbeschriftung
Bodenflaeche	Bodenfläche
BodenflaechenFormel	Formel für Bodenfläche
CeilingTexture	Textur der Decke
Chimneys	Liste der Kamine des Raumes
Conturs	Liste der Konturen, die den Raum umschließen
Deckenflaeche	Deckenfläche
DeckenflaechenFormel	Formel für Deckenfläche
Flags	Selektiert etc.
FloorTexture	Textur des Bodens
Font	Schriftart der Raumbeschriftung
History	Daten zur Identifikation im ehemaligen Projekt für einzeln geladene Gebäude
ID	Eine eindeutige Kennung
Name	Raumbezeichnung
NettoDeckenflaeche	Netto-Deckenfläche
NettoDeckenflaechenFormel	Formel für Netto-Deckenfläche
NettoGrundflaeche	Netto-Grundfläche
NettoGrundflaechenFormel	Formel für Netto-Grundfläche
Nutzflaechenart	Nutzflächenart nach DIN277
Ordnungszahl	Ordnungszahl nach DIN277
PlasterName	Putzbezeichnung
PlasterThickness	Putzstärke
Remark	Raum-Bemerkung
RoofAreas	Liste der Deckenflächen
Story	Geschoß, in dem sich der Raum befindet
Supports	Liste der Stützen des Raumes
Umschliessungsart	Umschließungsart nach DIN277
Volumen	Rauminhalt
VolumenFormel	Formel für Rauminhalt

Methoden

AddHolePolygon	Schneidet ein Loch in die Decke oder den Boden
GetHatchStyle	Fragt die Schraffur der 2D Darstellung der Wand ab
Objects	Liefert ein Array mit allen 3D Objekten in diesem Raum
RemoveHolePolygon	Entfernt ein Loch aus Decke oder Boden
SetHatchStyle	Ändert die Schraffur der 2D Darstellung der Wand

Room.Angle

Deklaration:

long

HRESULT get_Angle(long* retVal);
HRESULT put_Angle(long newVal);

Siehe auch: Room Übersicht

Der Winkel der Raumbeschriftung gegenüber der X-Koordinatenachse in Bogenmaß.

Room.Bodenflaeche

Deklaration:

Single (nur lesbar)

HRESULT get_Bodenflaeche(float* retVal);

Siehe auch: Room Übersicht

Die Gesamtbodenfläche des Raumes in Quadratmeter.

Room.BodenflaechenFormel

Deklaration:

String (nur lesbar)

HRESULT get_BodenflaechenFormel(BSTR* retVal);

Siehe auch: Room Übersicht

Eine symbolische Formel zur Berechnung der Bodenfläche.

Room.CeilingTexture

Deklaration:

Texture (nur lesbar)

HRESULT get_CeilingTexture(ITexture** retVal);

Siehe auch: Room Übersicht, Texture

Room.Chimneys

Deklaration:

ChimneyCollection (nur lesbar)

HRESULT get_Chimneys(IChimneyCollection** retVal);

Siehe auch: Room Übersicht, ChimneyCollection

Liefert eine Liste aller Schornsteine in diesem Raum.

Room.Conturs

Deklaration:

ConturCollection (nur lesbar)

HRESULT get_Conturs(IConturCollection** retVal);

Siehe auch: Room Übersicht, ConturCollection

Liefert die beiden Konturen des Raumes: die innere begrenzt den Raum, die äußere besteht aus den jeweils einem Wandsegment der inneren Kontur gegennüberliegenden Wandsegmenten der den Raum begrenzenden Wände. Geometrisch befindet sich zwischen der äußeren und der inneren Kontur “ Wand” .

Room.Deckenflaeche

Deklaration:

Single (nur lesbar)

HRESULT get_Deckenflaeche(float* retVal);

Siehe auch: Room Übersicht

Die Deckenfläche des Raumes in Quadratmetern.

Room.DeckenflaechenFormel

Deklaration:

String (nur lesbar)

HRESULT get_DeckenflaechenFormel(BSTR* retVal);

Siehe auch: Room Übersicht

Eine symbolische Formel zur Berechnung der Deckenfläche.

Room.Flags

Deklaration:

long

HRESULT get_Flags(long* retVal);
HRESULT put_Flags(long newVal);

Siehe auch: Room Übersicht

Einzelne Bits, die die Sichtbarkeit der Raumbeschriftung sowie zusätzlicher Elemente steuern. Symbolische Konstanten für die einzelnen Bits finden Sie im Kapitel Flags für Räume.

Room.FloorTexture

Deklaration:

Texture (nur lesbar)

HRESULT get_FloorTexture(ITexture** retVal);

Siehe auch: Room Übersicht, Texture

Room.Font

Deklaration:

VARIANT

HRESULT get_Font(VARIANT* retVal);
HRESULT put_Font(VARIANT newVal);

Siehe auch: Room Übersicht

Die Schriftart der Raumbeschriftung. Das hier übergebene “ VARIANT” -Objekt muß einen OLE-Font (Interface IFont oder IFontDisp, bzw. in Basic StdFont) enthalten.

Room.History

Deklaration:

IDHistory (nur lesbar)

HRESULT get_History(IIDHistory** retVal);

Siehe auch: Room Übersicht, IDHistory

Falls dieses Room beim Ursprünglichen Laden (oder Erstellen) des aktuellen Projektes nicht in diesem Projekt war, sonder aus einem anderen Projekt nachgeladen wurde, liefert diese Eigenschaft das History-Objekt, das die Umsetzung der IDs beschreibt. Andernfalls ist diese Eigenschaft “ Nothing” .

Ein Anwendungsbeispiel für History-Objekte finden Sie im Kapitel Eindeutige Kennungen.

Room.ID

Deklaration:

long (nur lesbar)

HRESULT get_ID(long* retVal);

Siehe auch: Room Übersicht

Gibt eine eindeutige Kennung dieses Room an. Diese Kennung ist persistent, das heißt, sie bleibt innerhalb des aktuellen Projektes erhalten. Sie können diese Kennung benutzen, um Rooms miteinander zu vergleichen oder zusätzliche Daten zum Room in einer Datenbank abzulegen.

Die ID ist aber nur innerhalb des aktuellen Projektes eindeutig, wenn Sie (durch Nachladen einzelner Gebäude) das Objekt in ein anderes Projekt einfügen, erhält es eine neue ID. In diesem Fall müssen Sie sofort bei Beendigung des Ladevorganges mit Hilfe der Eigenschaft History objPropertyDetailsRoomHistory Ihre zusätzlichen Daten an die neuen Werte anpassen.

Room.Name

Deklaration:

String

HRESULT get_Name(BSTR* retVal);
HRESULT put_Name(BSTR newVal);

Siehe auch: Room Übersicht

Der Name des Raumes.

Room.NettoDeckenflaeche

Deklaration:

Single (nur lesbar)

HRESULT get_NettoDeckenflaeche(float* retVal);

Siehe auch: Room Übersicht

Die nutzbare Deckenfläche in Quadratmeter.

Room.NettoDeckenflaechenFormel

Deklaration:

String (nur lesbar)

HRESULT get_NettoDeckenflaechenFormel(BSTR* retVal);

Siehe auch: Room Übersicht

Eine symbolische Formel zur Berechnung der nutzbaren Deckenfläche.

Room.NettoGrundflaeche

Deklaration:

Single (nur lesbar)

HRESULT get_NettoGrundflaeche(float* retVal);

Siehe auch: Room Übersicht

Die nutzbare Grundfläche in Quadratmetern.

Room.NettoGrundflaechenFormel

Deklaration:

String (nur lesbar)

HRESULT get_NettoGrundflaechenFormel(BSTR* retVal);

Siehe auch: Room Übersicht

Eine symbolische Formel zur Berechnung der Netto-Grundfläche.

Room.Nutzflaechenart

Deklaration:

String

HRESULT get_Nutzflaechenart(BSTR* retVal);
HRESULT put_Nutzflaechenart(BSTR newVal);

Siehe auch: Room Übersicht

Nutzflächenart des Raumes nach DIN 277.

Room.Ordnungszahl

Deklaration:

String

HRESULT get_Ordnungszahl(BSTR* retVal);
HRESULT put_Ordnungszahl(BSTR newVal);

Siehe auch: Room Übersicht

Die Ordnungszahl des Raumes im Raumbuch.

Room.PlasterName

Deklaration:

String

HRESULT get_PlasterName(BSTR* retVal);
HRESULT put_PlasterName(BSTR newVal);

Siehe auch: Room Übersicht

Name der verwendeten Putzart.

Room.PlasterThickness

Deklaration:

Single

HRESULT get_PlasterThickness(float* retVal);
HRESULT put_PlasterThickness(float newVal);

Siehe auch: Room Übersicht

Die Dicke des Innenputzes.

Room.Remark

Deklaration:

String

HRESULT get_Remark(BSTR* retVal);
HRESULT put_Remark(BSTR newVal);

Siehe auch: Room Übersicht

Eine beliebige Bemerkung zum Raum.

Room.RoofAreas

Deklaration:

RoofAreaCollection (nur lesbar)

HRESULT get_RoofAreas(IRoofAreaCollection** retVal);

Siehe auch: Room Übersicht, RoofAreaCollection

Room.Story

Deklaration:

Story (nur lesbar)

```
HRESULT get_Story(IStory** retVal);
```

Siehe auch: Room Übersicht, Story

Das Stockwerk, in dem sich der Raum befindet.

Room.Supports

Deklaration:

SupportCollection (nur lesbar)

HRESULT get_Supports(ISupportCollection** retVal);

Siehe auch: Room Übersicht, SupportCollection

Eine Liste aller Stützen in diesem Raum.

Room.Umschliessungsart

Deklaration:

short

HRESULT get_Umschliessungsart(short* retVal);
HRESULT put_Umschliessungsart(short newVal);

Siehe auch: Room Übersicht

Die Umschließungsart des Raumes nach DIN 277.

Room.Volumen

Deklaration:

Single (nur lesbar)

HRESULT get_Volumen(float* retVal);

Siehe auch: Room Übersicht

Das Volumen des Raumes in Kubikmetern.

Room.VolumenFormel

Deklaration:

String (nur lesbar)

HRESULT get_VolumenFormel(BSTR* retVal);

Siehe auch: Room Übersicht

Eine symbolische Formel zur Berechnung des Raumvolumens.

Room.AddHolePolygon

Deklaration:

```
Function AddHolePolygon(ByVal thePolygon as
HolePolygon, ByVal holeInCeiling as Boolean) as Boolean
```

```
HRESULT AddHolePolygon(IHolePolygon* thePolygon,
VARIANT_BOOL holeInCeiling, VARIANT_BOOL* retVal);
```

Siehe auch: Room Übersicht, HolePolygon

Room.GetHatchStyle

Deklaration:

Sub GetHatchStyle()

HRESULT GetHatchStyle(AC_Hatch_Style *HatchStyle, long *HatchColor);

Siehe auch: Room Übersicht

Room.Objects

Deklaration:

```
Function Objects() as VARIANT
```

```
HRESULT Objects(VARIANT* retVal);
```

Siehe auch: Room Übersicht

Room.RemoveHolePolygon

Deklaration:

```
Function RemoveHolePolygon(ByVal thePolygon as
HolePolygon, ByVal holeInCeiling as Boolean) as Boolean
```

```
HRESULT RemoveHolePolygon(IHolePolygon* thePolygon,
VARIANT_BOOL holeInCeiling, VARIANT_BOOL* retVal);
```

Siehe auch: Room Übersicht, HolePolygon

Room.SetHatchStyle

Deklaration:

Sub SetHatchStyle(ByVal HatchStyle as AC_Hatch_Style, ByVal HatchColor as long)

HRESULT SetHatchStyle(AC_Hatch_Style HatchStyle, long HatchColor);

Siehe auch: Room Übersicht

Dimension

Eine Vermaßung.

Dimension Übersicht

Eigenschaften

Architektengerecht	Architektengerechte Darstellung
Decimals	Anzahl Nachkommastellen bei Ausgabe
Distance	Abstand der Referenzpunkte von der Maßlinie
History	Daten zur Identifikation im ehemaligen Projekt für nachgeladene Gebäude
ID	Liefert eine eindeutige Kennung
Story	Das Stockwerk, in dem die Vermaßung angezeigt wird
Type	Art der Beschriftung

Methoden

Delete	Beschriftung löschen
GetPos	Liefert die Position
SetPos	Setzt die Position

Dimension.Architektengerecht

Deklaration:

Boolean

HRESULT get_Architektengerecht(VARIANT_BOOL* retVal)
;
HRESULT put_Architektengerecht(VARIANT_BOOL newVal)
;

Siehe auch: Dimension Übersicht

Wenn dieses Attribut auf “ Wahr” gesetzt wird, benutzt ArCon die normierte, architektengerechte Darstellung für die Vemaßung.

Dimension.Decimals

Deklaration:

short

HRESULT get_Decimals(short* retVal);
HRESULT put_Decimals(short newVal);

Siehe auch: Dimension Übersicht

Anzahl der angezeigten Nachkommastellen.

Dimension.Distance

Deklaration:

Single

HRESULT get_Distance(float* retVal);
HRESULT put_Distance(float newVal);

Siehe auch: Dimension Übersicht

Abstand der beiden Referenzpunkte zu der angezeigten Maßlinie.

Dimension.History

Deklaration:

IDHistory (nur lesbar)

HRESULT get_History(IIDHistory** retVal);

Siehe auch: Dimension Übersicht, IDHistory

Falls dieses Dimension beim Ursprünglichen Laden (oder Erstellen) des aktuellen Projektes nicht in diesem Projekt war, sonder aus einem anderen Projekt nachgeladen wurde, liefert diese Eigenschaft das History-Objekt, das die Umsetzung der IDs beschreibt. Andernfalls ist diese Eigenschaft “ Nothing” .

Ein Anwendungsbeispiel für History-Objekte finden Sie im Kapitel Eindeutige Kennungen.

Dimension.ID

Deklaration:

long (nur lesbar)

HRESULT get_ID(long* retVal);

Siehe auch: Dimension Übersicht

Gibt eine eindeutige Kennung dieses Dimension an. Diese Kennung ist persistent, das heißt, sie bleibt innerhalb des aktuellen Projektes erhalten. Sie können diese Kennung benutzen, um Dimensions miteinander zu vergleichen oder zusätzliche Daten zum Dimension in einer Datenbank abzulegen.

Die ID ist aber nur innerhalb des aktuellen Projektes eindeutig, wenn Sie (durch Nachladen einzelner Gebäude) das Objekt in ein anderes Projekt einfügen, erhält es eine neue ID. In diesem Fall müssen Sie sofort bei Beendigung des Ladevorganges mit Hilfe der Eigenschaft History objPropertyDetailsDimensionHistory Ihre zusätzlichen Daten an die neuen Werte anpassen.

Dimension.Story

Deklaration:

Story (nur lesbar)

HRESULT get_Story(IStory** retVal);

Siehe auch: Dimension Übersicht, Story

Ein Verweis auf das Stockwerk, zu dem diese Vermaßung gehört.

Dimension.Type

Deklaration:

short

HRESULT get_Type(short* retVal);
HRESULT put_Type(short newVal);

Siehe auch: Dimension Übersicht

Variante der Vermaßung. Typcodes beginnen mit 0.

Dimension.Delete

Deklaration:

Sub Delete()

HRESULT Delete();

Siehe auch: Dimension Übersicht

Löscht die Vermaßung.

Dimension.GetPos

Deklaration:

```
Function GetPos(ByRef X1 as Single, ByRef Y1 as Single,
ByRef X2 as Single, ByRef Y2 as Single) as Boolean
```

```
HRESULT GetPos(float* X1, float* Y1, float* X2, float* Y2,
VARIANT_BOOL* retVal);
```

Siehe auch: Dimension Übersicht

Liefert die Position der Vermaßung.

Rückgabewert ist “ Wahr” , wenn die Position ermittelt werden konnte, “ Falsch” wenn nicht (z. B. weil die Vermaßung noch nicht plaziert wurde).

Die Parameter sind:

X1, Y1	Koordinaten des Startpunktes
X2, Y2	Koordinaten des Endpunktes

Dimension.SetPos

Deklaration:

```
Function SetPos(ByVal X1 as Single, ByVal Y1 as Single,
ByVal X2 as Single, ByVal Y2 as Single) as Boolean
```

```
HRESULT SetPos(float X1, float Y1, float X2, float Y2,
VARIANT_BOOL* retVal);
```

Siehe auch: Dimension Übersicht

Setze die Position der Vermaßung.

Rückgabewert ist “ Wahr” wenn die Vermaßung verschoben werden konnte.

Die Parameter sind:

X1, Y1	Koordinaten des Startpunktes
X2, Y2	Koordinaten des Endpunktes

Guide

Eine Hilfslinie im Konstruktionsmodus.

Guide Übersicht

Eigenschaften

Color	Farbe der Hilfslinien
DrawStyle	Linienart
Endless	Hilfsgerade oder -Strecke?
History	Daten zur Identifikation im ehemaligen Projekt für nachgeladene Gebäude
ID	Liefert eine eindeutige Kennung
Story	Stockwerk, in dem sich die Hilfslinie befindet

Methoden

Delete	Hilfslinien löschen
GetPos	Position der Hilfslinie lesen
SetPos	Hilfslinie an Position setzen

Guide.Color

Deklaration:

long

HRESULT get_Color(long* retVal);
HRESULT put_Color(long newVal);

Siehe auch: Guide Übersicht

Farbe der Hilfslinie (ein RGB-Wert).

Guide.DrawStyle

Deklaration:

short

HRESULT get_DrawStyle(short* retVal);
HRESULT put_DrawStyle(short newVal);

Siehe auch: Guide Übersicht

Die Art der Liniendarstellung. Es handelt sich dabei um die bekannten Pen-Style Konstanten, also PS_SOLID, PS_DASH, u.s.w., oder in Visual Basic: VtPenStyleSolid, VtPenStyleDash, ...

Guide.Endless

Deklaration:

Boolean

HRESULT get_Endless(VARIANT_BOOL* retVal);
HRESULT put_Endless(VARIANT_BOOL newVal);

Siehe auch: Guide Übersicht

“ Wahr” , wenn es sich um eine Hilfsgerade handelt, “ Falsch” wenn es eine Hilfsstrecke ist.

Guide.History

Deklaration:

IDHistory (nur lesbar)

HRESULT get_History(IIDHistory** retVal);

Siehe auch: Guide Übersicht, IDHistory

Falls dieses Guide beim Ursprünglichen Laden (oder Erstellen) des aktuellen Projektes nicht in diesem Projekt war, sonder aus einem anderen Projekt nachgeladen wurde, liefert diese Eigenschaft das History-Objekt, das die Umsetzung der IDs beschreibt. Andernfalls ist diese Eigenschaft “ Nothing” .

Ein Anwendungsbeispiel für History-Objekte finden Sie im Kapitel Eindeutige Kennungen.

Guide.ID

Deklaration:

long (nur lesbar)

HRESULT get_ID(long* retVal);

Siehe auch: Guide Übersicht

Gibt eine eindeutige Kennung dieses Guide an. Diese Kennung ist persistent, das heißt, sie bleibt innerhalb des aktuellen Projektes erhalten. Sie können diese Kennung benutzen, um Guides miteinander zu vergleichen oder zusätzliche Daten zum Guide in einer Datenbank abzulegen.

Die ID ist aber nur innerhalb des aktuellen Projektes eindeutig, wenn Sie (durch Nachladen einzelner Gebäude) das Objekt in ein anderes Projekt einfügen, erhält es eine neue ID. In diesem Fall müssen Sie sofort bei Beendigung des Ladevorganges mit Hilfe der Eigenschaft History objPropertyDetailsGuideHistory Ihre zusätzlichen Daten an die neuen Werte anpassen.

Guide.Story

Deklaration:

Story (nur lesbar)

HRESULT get_Story(IStory** retVal);

Siehe auch: Guide Übersicht, Story

Verweis auf das Stockwerk, in dem die Hilfslinie angeordnet ist.

Guide.Delete

Deklaration:

```
Sub Delete()
```

```
HRESULT Delete();
```

Siehe auch: Guide Übersicht

Löscht die Hilfslinie.

Guide.GetPos

Deklaration:

```
Function GetPos(ByRef X1 as Single, ByRef Y1 as Single,
ByRef X2 as Single, ByRef Y2 as Single) as Boolean
```

```
HRESULT GetPos(float* X1, float* Y1, float* X2, float* Y2,
VARIANT_BOOL* retVal);
```

Siehe auch: Guide Übersicht

Ermittelt die Position der Hilfslinie.

Rückgabewert ist “ Wahr” wenn die Position ermittelt werden konnte, “ Falsch” andernfalls (z.B. weil die Hilfslinie noch nicht plaziert wurde).

Die Parameter sind:

X1, Y1	Koordinaten des ersten Punktes
X2, Y2	Koordinaten des zweiten Punktes

Guide.SetPos

Deklaration:

```
Function SetPos(ByVal X1 as Single, ByVal Y1 as
Single, ByVal X2 as Single, ByVal Y2 as Single) as
Boolean
```

```
HRESULT SetPos(float X1, float Y1, float X2, float
Y2, VARIANT_BOOL* retVal);
```

Siehe auch: Guide Übersicht

Positioniert die Hilfslinie neu.

Rückgabewert ist “ Wahr” wenn die Hilfslinie verschoben werden konnte.

Die Parameter sind:

X1, Y1	Neue Koordinaten des ersten Punktes
X2, Y2	Neue Koordinaten des zweiten Punktes

Graphics2DObject

Die Klasse Graphics2DObject ist eine abstrakte Oberklasse aller 2D Grafikelemente. Sie können kein Objekt dieser Klasse erzeugen, sie dient lediglich zur Beschreibung der gemeinsamen Methoden und Eingenschaften.

Instanzen erzeugen können Sie von den Unterklassen: Line, Label, Shape, Image und Polygon2D.

Graphics2DObject Übersicht

Eigenschaften

CustomMarks	Enthält benutzerdefinierte Griffe für Editieroperationen
Cut	Der (optionale) Schnitt der 2D-Grafik
Deletable	Gibt an, ob das 2D-Grafikobjekt löschbar ist
Moveable	Gibt an, ob das 2D-Grafikobjekt verschiebbar oder editierbar ist
Selectable	Ist das 2D-Grafikobjekt selektierbar?
Selected	Ist das 2D-Grafikobjekt selektiert?
Snapable	Gibt an, ob auf das 2D-Grafikobjekt gefangen wird
Story	Das (optionale) Stockwerk des 2D-Grafikobjektes
UserData	Benutzerdefinierte Daten
Visible	Ist das 2D-Grafikobjekt sichtbar?

Methoden

EnableUpdate	Schaltet die Aktualisierung der Grafik ein oder aus
GetLayer	Liefert die aktuelle Zeichnungsebene
Refresh	Zeichne das 2D-Grafikobjekt neu
SetLayer	Setzt die Zeichnungsebene neu
SetMarks	Steuert die Darstellung von Griffen zum Verschieben oder Skalieren
ZOrder	Positioniere das 2D-Grafikobjektes neu in Z-Richtung

Graphics2DObject.CustomMarks

Deklaration:

Point2DCollection (nur lesbar)

HRESULT get_CustomMarks(IPoint2DCollection** retVal);

Siehe auch: Graphics2DObject Übersicht, Point2DCollection

Eine Liste von Punkten für benutzerdefinierte Selektionsmarkierungen.

Graphics2DObject.Cut

Deklaration:

Cut (nur lesbar)

HRESULT get_Cut(ICut** retVal);

Siehe auch: Graphics2DObject Übersicht, Cut

Graphics2DObject.Deletable

Deklaration:

Boolean

HRESULT get_Deletable(VARIANT_BOOL* retVal);
HRESULT put_Deletable(VARIANT_BOOL newVal);

Siehe auch: Graphics2DObject Übersicht

Gibt an, ob das Objekt vom Benutzer gelöscht werden kann. Diese Eigenschaft hat keinen Einfluß auf die Löschung des Elementes durch Ihr Programm!

Graphics2DObject.Moveable

Deklaration:

Boolean

HRESULT get_Moveable(VARIANT_BOOL* retVal);
HRESULT put_Moveable(VARIANT_BOOL newVal);

Siehe auch: Graphics2DObject Übersicht

Gibt an, ob das Objekt vom Benutzer verschoben werden kann. Diese Eigenschaft hat keinen Einfluß auf Positionsänderungen durch Ihr Programm.

Graphics2DObject.Selectable

Deklaration:

Boolean

HRESULT get_Selectable(VARIANT_BOOL* retVal);
HRESULT put_Selectable(VARIANT_BOOL newVal);

Siehe auch: Graphics2DObject Übersicht

Gibt an, ob das Objekt vom Benutzer durch Anklicken selektiert werden kann. Siehe auch Selected.

Graphics2DObject.Selected

Deklaration:

Boolean

HRESULT get_Selected(VARIANT_BOOL* retVal);
HRESULT put_Selected(VARIANT_BOOL newVal);

Siehe auch: Graphics2DObject Übersicht

Gibt an, ob das Objekt momentan selektiert ist. Dies ist nur möglich, wenn Selectable wahr ist.

Graphics2DObject.Snapable

Deklaration:

Boolean

HRESULT get_Snapable(VARIANT_BOOL* retVal);
HRESULT put_Snapable(VARIANT_BOOL newVal);

Siehe auch: Graphics2DObject Übersicht

Gibt an, ob ArCon auf den Rand dieses Objektes fängt.

Graphics2DObject.Story

Deklaration:

Story (nur lesbar)

```
HRESULT get_Story(IStory** retVal);
```

Siehe auch: Graphics2DObject Übersicht, Story

Graphics2DObject.UserData

Deklaration:

long

HRESULT get_UserData(long* retVal);
HRESULT put_UserData(long newVal);

Siehe auch: Graphics2DObject Übersicht

Ein beliebiger Wert, der von ArCon nicht weiter beachtet wird. Ihr Programm kann diesen Wert z. B. nutzen, um einen Tabellenindex oder einen Zeiger auf diesem Objekt zugeordnete (interne) Daten Ihres Programmes zu speichern. Auf diese Weise kann ein Programm das regelmäßige Suchen des Objektes in den internen Datenstrukturen vermeiden.

Graphics2DObject.Visible

Deklaration:

Boolean

HRESULT get_Visible(VARIANT_BOOL* retVal);
HRESULT put_Visible(VARIANT_BOOL newVal);

Siehe auch: Graphics2DObject Übersicht

Gibt an, ob das Objekt sichtbar ist. Ein unsichtbares Objekt wird von ArCon komplett ignoriert, kann aber sehr schnell wieder sichtbar gemacht werden.

Graphics2DObject.EnableUpdate

Deklaration:

```
Sub EnableUpdate(ByVal enabled as Boolean)
```

```
HRESULT EnableUpdate(VARIANT_BOOL enabled);
```

Siehe auch: Graphics2DObject Übersicht

Schaltet die Graphikaktualisierung an oder ab.

Parameter:

enabled	Wenn dieser Wert Falsch ist, werden keine Graphikaktualisierungen mehr durchgeführt. Wenn der Wert Wahr ist, wird die Graphik wieder aktualisiert. Beim Einschalten der Aktualisierung wird das Objekt auf jeden Fall einmal neu gezeichnet, so als ob die Refresh Methode aufgerufen worden wäre.

Graphics2DObject.GetLayer

Deklaration:

```
Function GetLayer() as long
```

```
HRESULT GetLayer(long* retVal);
```

Siehe auch: Graphics2DObject Übersicht

Liefert die Zeichnungsebene des Objektes.

Graphics2DObject.Refresh

Deklaration:

```
Sub Refresh()
```

```
HRESULT Refresh();
```

Siehe auch: Graphics2DObject Übersicht

Stellt die Grafik neu dar.

In der Regel ist es nicht nötig, diese Methode aufzurufen, da ArCon sich automatisch um die korrekte Darstellung des Bildes kümmert.

Graphics2DObject.SetLayer

Deklaration:

```
Function SetLayer(ByVal newLayer as long) as
Boolean
```

```
HRESULT SetLayer(long newLayer,
VARIANT_BOOL* retVal);
```

Siehe auch: Graphics2DObject Übersicht

Ändert die Zeichnungsebene des Objektes.

Parameter ist:

newLayer	die neue Zeichnungsebene

Graphics2DObject.SetMarks

Deklaration:

```
Sub SetMarks(ByVal markerType as short,
ByVal withLines as Boolean)
```

```
HRESULT SetMarks(short markerType,
VARIANT_BOOL withLines);
```

Siehe auch: Graphics2DObject Übersicht

Aktiviert oder Deaktiviert Markierungskästchen des Objektes.

Parameter sind:

markerType	Kennzahl für die Art der Markierung (s.u.)
withLines	Wahr, wenn die Markierungspunkte mit Linien verbunden werden sollen.

Mögliche Werte für den Parameter markerType sind:

ACG2D_NO_MARKS	Keine Markierungen anzeigen
ACG2D_POLYGON_MARKS	Jeder Punkt eines Polygons erhält einer Markierung
ACG2D_BBOX_MARKS	Jeder Eckpunkt und der Mittelpunkt jeder Kante des umschließenden Rechtecks erhält eine Markierung
ACG2D_BBOX_LINEAR	Wie ACG2D_BBOX_MARKS, aber die Kantenmarkierungen entfallen und das Objekt kann nur unter Beibehaltung des x/y Verhältnisses skaliert werden

Graphics2DObject.ZOrder

Deklaration:

```
Sub ZOrder(ByVal pos as short)
```

```
HRESULT ZOrder(short pos);
```

Siehe auch: Graphics2DObject Übersicht

Positioniert das Objekt neu in der Z-Richtung.

Parameter:

pos	gibt die neue Z-Position an. 0 bringt das Bild in den Vordergrund, während große Werte es weiter nach hinten positionieren.

ProjectPreview

ProjectPreview Übersicht

Eigenschaften

ConstructionPreview	Vorschau auf den Konstruktionsmodus
Description	Beschreibung des Projektes
DesignPreview	Vorschau auf den Designmodus

ProjectPreview.ConstructionPreview

Deklaration:

VARIANT (nur lesbar)

HRESULT get_ConstructionPreview(VARIANT* retVal);

Siehe auch: ProjectPreview Übersicht

ProjectPreview.Description

Deklaration:

String (nur lesbar)

HRESULT get_Description(BSTR* retVal);

Siehe auch: ProjectPreview Übersicht

ProjectPreview.DesignPreview

Deklaration:

VARIANT (nur lesbar)

HRESULT get_DesignPreview(VARIANT* retVal);

Siehe auch: ProjectPreview Übersicht

Polygon2D

Ein Polygon2D-Objekt gehört zu den 2D-Primitiven, durch die Sie eigene grafische Elemente in ArCon’ s 2D Darstellung einbringen können. Andere Grafikelemente sind: Label, Shape, Line und Image.

Polygon2D stellt einen geschlossenen Linienzug (definiert durch mehrere Punkte) dar. Jeder Punkt wird mit dem nächsten verbunden, der letzte mit dem ersten. Dabei dürfen sich Linien nicht überschneiden.

Polygon2D Übersicht

Eigenschaften

BackColor	Hintergrundfarbe
BackStyle	Hintergund verdeckt oder durchsichtig
BorderColor	Farbe der Linie
BorderStyle	Darstellungsart der Linie
BorderWidth	Die Strichstärke des Polygons
Closed	Der Polygonzug ist geschlossen (der erste Punkt ist mit dem letzten Verbunden)
DrawMode	Zeichenmodus
FillColor	Füllfarbe
FillStyle	Füllmuster
Points	Liste mit Punkten

Methoden

SetPoints	Setzt alle Punkte aus einem Array

Polygon2D.BackColor

Deklaration:

long

HRESULT get_BackColor(long* retVal);
HRESULT put_BackColor(long newVal);

Siehe auch: Polygon2D Übersicht

RGB Wert der Hintergrundfarbe. Diese Farbe wird nur sichtbar, wenn der entsprechende Hintergrundstil gewählt wird.

Polygon2D.BackStyle

Deklaration:

short

HRESULT get_BackStyle(short* retVal);
HRESULT put_BackStyle(short newVal);

Siehe auch: Polygon2D Übersicht

Gibt an, ob der Hintergrund durchsichtig ist, oder mit der Hintergrundfarbe gefüllt wird.

Mögliche Werte sind:

0	transparent, durchsichtig
1	opaque, in der Hintergrundfarbe gefüllt

Polygon2D.BorderColor

Deklaration:

long

HRESULT get_BorderColor(long* retVal);

HRESULT put_BorderColor(long newVal);

Siehe auch: Polygon2D Übersicht

RGB Wert der Farbe, mit der der Rand der Figur gezeichnet wird. Siehe auch BorderStyle.

Polygon2D.BorderStyle

Deklaration:

short

HRESULT get_BorderStyle(short* retVal);

HRESULT put_BorderStyle(short newVal);

Siehe auch: Polygon2D Übersicht

Gibt die Art des Randes der Figur an:

0	transparent, unsichtbar
1	solid, durchgezogen
2	dash, gestrichelt
3	dot, gepunktet
4	dash dot, Strich-Punkt
5	dash dot dot, Strich-Punkt-Punkt

Polygon2D.BorderWidth

Deklaration:

Single

HRESULT get_BorderWidth(float* retVal);

HRESULT put_BorderWidth(float newVal);

Siehe auch: Polygon2D Übersicht

Breite der Linie in Millimetern auf dem Papier. Gestrichelte (u.ä.) Linien (siehe BorderStyle) funktionieren nur mit der Breite 0 (Haarlinie).

Falls Ihr Polygon sich nicht auf die dargestellte Welt sondern den Bildschirm bezieht (z.B. Markierungen oder “ Griffe”), können Sie mit Hilfe einer negativen Breite eine konstante Pixel-Breite erreichen: bei negativen Werten gibt der Absolutwert die Breite in Pixel an

Polygon2D.Closed

Deklaration:

Boolean

```
HRESULT get_Closed
(VARIANT_BOOL* retVal);
HRESULT put_Closed
(VARIANT_BOOL newVal);
```

Siehe auch: Polygon2D Übersicht

Wahr, wenn das Polygon geschlossen gezeichnet wird (der letzte Punkt wird mit dem ersten verbunden), falsch wenn nicht.

Beispiel: ein offenes Polygon

Die gleichen Punkte, aber als geschlossenes Polygon:

Polygon2D.DrawMode

Deklaration:

short

HRESULT get_DrawMode(short* retVal);

HRESULT put_DrawMode(short newVal);

Siehe auch: Polygon2D Übersicht

Gibt an, wie die Zeichenfarbe ermittelt wird. Meist wird der Wert “ copy pen” (13) benutzt, gelegentlich auch “ XOR Pen” (7).

Einstellung	Beschreibung
1	Blackness (Schwarz).
2	Not Merge Pen (Nicht-Mischen-Stift): Inverse Darstellung der Einstellung 15 (Merge Pen).
3	Mask Not Pen (Maskieren-Nicht-Stift): Kombination der Farben, die der Hintergrund mit der invertierten Stiftfarbe gemeinsam hat.
4	Not Copy Pen (Nicht-Kopieren-Stift): Inverse Darstellung der Einstellung 13 (Copy Pen).
5	Mask Pen Not (Maskieren-Stift-Nicht): Kombination der Farben, die der Stift mit der invertierten Anzeigefarbe gemeinsam hat.
6	Invert (Invers): Inverse Darstellung der Anzeigefarbe.
7	Xor Pen (Xor-Stift): Kombination der Farben, die im Stift und in der Anzeigefarbe, aber nicht in beiden vorhanden sind.
8	Not Mask Pen (Nicht-Maskieren-Stift): Inverse Darstellung der Einstellung 9 (Mask Pen).
9	Mask Pen (Maskieren-Stift): Kombination der Farben, die der Stift mit der Anzeige gemeinsam hat.
10	Not Xor Pen (Nicht-Xor-Stift): Inverse Darstellung der Einstellung 7 (Xor Pen).
11	Nop: Keine OperationAusgabe bleibt unverändert, d.h. diese Einstellung schaltet die Ausgabe aus.
12	Merge Not Pen (Mischen-Nicht-Stift): Kombination der Anzeigefarbe und der invertierten Stiftfarbe.
13	Copy Pen (Kopieren-Stift): (Voreinstellung) Farbe, die durch die BorderColor-Eigenschaft angegeben ist.
14	Merge Pen Not (Mischen-Stift-Nicht): Kombination der Stiftfarbe und der invertierten Anzeigefarbe.

15	Merge Pen (Mischen-Stift): Kombination der Stiftfarbe und der Anzeigefarbe.
16	Whiteness (Weiß).

Polygon2D.FillColor

Deklaration:

long

HRESULT get_FillColor(long* retVal);
HRESULT put_FillColor(long newVal);

Siehe auch: Polygon2D Übersicht

RGB-Wert der Farbe, mit der das Innere der Figur gezeichnet wird, falls der Füllstil entsprechend gesetzt ist.

Polygon2D.FillStyle

Deklaration:

short

HRESULT get_FillStyle(short* retVal);
HRESULT put_FillStyle(short newVal);

Siehe auch: Polygon2D Übersicht

Gibt an, wie das Innere der Figur gezeichnet wird. Die zugehörige Farbe wird durch FillColor bestimmt.

Mögliche Werte sind:

0	solid, vollständig gefüllt
1	transparent, nicht gefüllt
2	Horizontal Line (horizontale Linien)
3	Vertical Line (vertikale Linien)
4	Upward Diagonal (Aufwärtsdiagonal)
5	Downward Diagonal (Abwärtsdiagonal)
6	Cross (senkrecht gekreuzt)
7	Diagonal Cross (diagonal gekreuzt)

Polygon2D.Points

Deklaration:

Point2DCollection (nur lesbar)

```
HRESULT get_Points
(IPoint2DCollection** retVal);
```

Siehe auch: Polygon2D Übersicht, Point2DCollection

Liefert die Liste aller Punkte des Polygons.

Polygon2D.SetPoints

Deklaration:

```
Sub SetPoints(ByVal numPoints as
long, ByVal Points as VARIANT)
```

```
HRESULT SetPoints(long numPoints,
VARIANT Points);
```

Siehe auch: Polygon2D Übersicht

Setzt alle Punkte des Polygons aus einem Array. Dabei muß es sich um ein eindimensionales Array aus einfach genauen Fließkommazahlen handeln (Basic: Single, C/C++: float). Die Koordinaten liegen abwechselnd als x und y Wert im Array, bei Index 0 befindet sich der erste x Wert, bei 1 der erste y Wert u.s.w.

Parameter:

numPoints	Anzahl der Punkte im Parameter " Points" (nicht die Anzahl der Einträge in diesem Array – die ist 2*numPoints)
Points	Ein OLE SafeArray, eindimensional, aus einfach genauen Fließkommazahlen

In Basic können Sie ein Array für den " Points" Parameter folgendermaßen deklarieren:

```
Dim polyPoints(1000) As Single
```

In diesem Array können Sie nun Koordinaten für 500 Punkte unterbringen.

In C++ ist der Zugriff etwas umständlicher.

Zunächst deklarieren Sie einen Hilfstyp zur Aufnahme der Punktkoordinaten:

```
struct Point {
  float x, y;
};
```

Dann schreiben Sie eine kleine Hilfsfunktion, um eine Kontur in ein SafeArray zu verpacken:

```
static void PointsToVariant(long num, const Point *points, VARIANT *v)
{
  // Das C-Array in ein OLE SafeArray konvertieren
  SAFEARRAY * array;
  SAFEARRAYBOUND bounds;
  bounds.lLbound = 0;
  bounds.cElements = num*2;
  array = SafeArrayCreate(VT_R4, 1, &bounds);
  void * mem = NULL;
  SafeArrayAccessData(array, &mem);
  memcpy(mem, points, sizeof(Point)*num);
  SafeArrayUnaccessData(array);

  VariantInit(v);
  v->vt = VT_ARRAY|VT_R4;
  v->parray = array;
}
```

Nachdem Sie eine Kontur mit Daten gefüllt haben, rufen Sie diese Methode folgendermaßen auf:

```
  VARIANT v;
  Point contur[4];
  // ...
  PointsToVariant(4, contur, &v);
  poly->SetPoints(4, v);
```

```
VariantClear(&v);
```

Window

Ein Fenster. ArCon unterscheidet zwischen frei konstruierbaren und geladenen Fenstern. Die Typkodierungen zwischen 0 und ArCon.ConstructedWindows -1 sind frei konstruierbar, die übrigen Typen sind geladen.

Window Übersicht

Eigenschaften

Area	Fensterfläche
AreaFormula	Formel für Fensterfläche
Height	Fensterhöhe
History	Daten zur Identifikation im ehemaligen Projekt für nachgeladene Gebäude
ID	Eine eindeutige Kennung
LeftHung	Links angeschlagen?
LeftSegment	Innenwandseite, in der das Fenster 'sitzt'
OpensInwards	Zur gegenüberliegenden Seite öffnen?
ParapetHeight	Brüstungshöhe
Remark	Bemerkung
RightSegment	Außenwandseite, in der das Fenster 'sitzt'
TextureCount	Anzahl der max. verfügbaren Texturen
Type	Fensterart (Variante des Multifunktionsschalters)
Wall	Wandobjekt, in dem das Fenster 'sitzt'
Width	Fensterbreite

Methoden

Delete	Fenster löschen
GetPolygons	Liefert die Wandlöcher zu beiden Seiten des Fensters, die durch das Fenster entstehen sowie deren Tiefe
GetPos	Fensterposition lesen
GetTexture	Liefert eine Textur
SetPos	Fenster auf Position setzen
SetTexture	Tauscht eine Textur aus

Window.Area

Deklaration:

Single (nur lesbar)

HRESULT get_Area(float* retVal);

Siehe auch: Window Übersicht

Die Fläche des Fensters in Qudratmetern.

Window.AreaFormula

Deklaration:

String (nur lesbar)

HRESULT get_AreaFormula(BSTR* retVal);

Siehe auch: Window Übersicht

Eine symbolische Formel für die Fensterfläche.

Window.Height

Deklaration:

Single

HRESULT get_Height(float* retVal);
HRESULT put_Height(float newVal);

Siehe auch: Window Übersicht

Die Höhe des Fensters.

Window.History

Deklaration:

IDHistory (nur lesbar)

HRESULT get_History(IIDHistory** retVal);

Siehe auch: Window Übersicht, IDHistory

Falls dieses Window beim Ursprünglichen Laden (oder Erstellen) des aktuellen Projektes nicht in diesem Projekt war, sonder aus einem anderen Projekt nachgeladen wurde, liefert diese Eigenschaft das History-Objekt, das die Umsetzung der IDs beschreibt. Andernfalls ist diese Eigenschaft “ Nothing” .

Ein Anwendungsbeispiel für History-Objekte finden Sie im Kapitel Eindeutige Kennungen.

Window.ID

Deklaration:

long (nur lesbar)

HRESULT get_ID(long* retVal);

Siehe auch: Window Übersicht

Gibt eine eindeutige Kennung dieses Window an. Diese Kennung ist persistent, das heißt, sie bleibt innerhalb des aktuellen Projektes erhalten. Sie können diese Kennung benutzen, um Windows miteinander zu vergleichen oder zusätzliche Daten zum Window in einer Datenbank abzulegen.

Die ID ist aber nur innerhalb des aktuellen Projektes eindeutig, wenn Sie (durch Nachladen einzelner Gebäude) das Objekt in ein anderes Projekt einfügen, erhält es eine neue ID. In diesem Fall müssen Sie sofort bei Beendigung des Ladevorganges mit Hilfe der Eigenschaft History objPropertyDetailsWindowHistory Ihre zusätzlichen Daten an die neuen Werte anpassen.

Window.LeftHung

Deklaration:

Boolean

HRESULT get_LeftHung(VARIANT_BOOL* retVal);
HRESULT put_LeftHung(VARIANT_BOOL newVal);

Siehe auch: Window Übersicht

“ Wahr” , wenn das Fenster links angeschlagen ist.

Window.LeftSegment

Deklaration:

WallSegment (nur lesbar)

HRESULT get_LeftSegment(IWallSegment** retVal);

Siehe auch: Window Übersicht, WallSegment

Das Wandsegment an der linken Fensterseite.

Window.OpensInwards

Deklaration:

Boolean

HRESULT get_OpensInwards(VARIANT_BOOL* retVal);
HRESULT put_OpensInwards(VARIANT_BOOL newVal);

Siehe auch: Window Übersicht

“ Wahr” , wenn das Fenster nach innen öffnet.

Window.ParapetHeight

Deklaration:

Single

HRESULT get_ParapetHeight(float* retVal);
HRESULT put_ParapetHeight(float newVal);

Siehe auch: Window Übersicht

Die Brüstungshöhe.

Window.Remark

Deklaration:

String

HRESULT get_Remark(BSTR* retVal);
HRESULT put_Remark(BSTR newVal);

Siehe auch: Window Übersicht

Eine beliebige Bemerkung zum Fenster.

Window.RightSegment

Deklaration:

WallSegment (nur lesbar)

HRESULT get_RightSegment(IWallSegment** retVal);

Siehe auch: Window Übersicht, WallSegment

Das Wandsegment an der rechten Seite des Fensters.

Window.TextureCount

Deklaration:

long (nur lesbar)

HRESULT get_TextureCount(long* retVal);

Siehe auch: Window Übersicht

Window.Type

Deklaration:

long

HRESULT get_Type(long* retVal);
HRESULT put_Type(long newVal);

Siehe auch: Window Übersicht

Der Türtyp. Es sind Werte von 0 bis AvailableWindows - 1 erlaubt.

Window.Wall

Deklaration:

Wall (nur lesbar)

HRESULT get_Wall(IWall** retVal);

Siehe auch: Window Übersicht, Wall

Die Wand, in der das Fenster sitzt.

Window.Width

Deklaration:

Single

HRESULT get_Width(float* retVal);
HRESULT put_Width(float newVal);

Siehe auch: Window Übersicht

Die Breite des Fensters.

Window.Delete

Deklaration:

Sub Delete()

HRESULT Delete();

Siehe auch: Window Übersicht

Löscht das Fenster.

Window.GetPolygons

Deklaration:

```
Function GetPolygons(ByRef leftPolygon as
Point2DCollection, ByRef rightPolygon as Point2DCollection,
ByRef leftDepth as Single, ByRef rightDepth as Single) as
Boolean
```

```
HRESULT GetPolygons(IPoint2DCollection** leftPolygon,
IPoint2DCollection** rightPolygon, float* leftDepth, float*
rightDepth, VARIANT_BOOL* retVal);
```

Siehe auch: Window Übersicht, Point2DCollection, Point2DCollection

Window.GetPos

Deklaration:

```
Function GetPos(ByRef X as Single, ByRef Y as Single) as
Boolean
```

```
HRESULT GetPos(float* X, float* Y, VARIANT_BOOL*
retVal);
```

Siehe auch: Window Übersicht

Liefert die Position des Fenstermittelpunktes.

Parameter sind:

x, y	werden auf die Koordinaten des Fenstermittelpunktes gesetzt.

Window.GetTexture

Deklaration:

```
Function GetTexture(ByVal index as long) as String
```

```
HRESULT GetTexture(long index, BSTR* retVal);
```

Siehe auch: Window Übersicht

Window.SetPos

Deklaration:

```
Function SetPos(ByVal X as Single, ByVal Y as Single) as
Boolean
```

```
HRESULT SetPos(float X, float Y, VARIANT_BOOL*
retVal);
```

Siehe auch: Window Übersicht

Setzt die Position neu.

Parameter sind:

x, y	Koordinaten des Fenstermittelpunktes.

Window.SetTexture

Deklaration:

```
Function SetTexture(ByVal index as long, ByVal
TextureName as String) as Boolean
```

```
HRESULT SetTexture(long index, BSTR TextureName,
VARIANT_BOOL* retVal);
```

Siehe auch: Window Übersicht

Project

Ein Objekt vom Typ Project beschreibt das aktuell geladene Projekt (siehe ArCon.CurrentProject) oder bereitet Einstellungen für ein noch zu erzeugendes Projekt vor (siehe ArCon.CreateProject und ArCon.NewProject).

Konstanten für die diversen Flags finden Sie im Kapitel Projekteinstellungen.

Project Übersicht
Eigenschaften

BackColorDay	Hintergrundfarbe bei Tag/zeitabhängiger Sicht
BackColorNight	Hintergrundfarbe bei Nachtsicht
Name	Name des Projektes inkl. Pfad
OriginX	X-Koordinate des Ursprungs
OriginY	Y-Koordinate des Ursprungs
OriginZ	Z-Kooridnate des Ursprungs
PaperHeight	Papierhöhe
PaperSize	Ausgewählte Papiergröße
PaperSizeName	Name des Papierformats
PaperSizePrinter	Welche Druckerpapiergröße, wenn ausgewählte Papiergröße 'vom Drucker vorgegeben'
PaperWidth	Papierbreite
Scale	Maßstab
SheetCenterX	Mittelpunkt des Blattes beim Drucken in Weltkoordinaten (X Wert)
SheetCenterY	Mittelpunkt des Blattes beim Drucken in Weltkoordinaten (Y Wert)
ShortName	Dateiname ohne Endung und Pfad bzw 'Neues Projekt'
Unit	Maßeinheit (m, cm, yard...)
WorldRotation	Rotation des Weltkoordinatensystems gegenüber dem Blatt

Project.BackColorDay

Deklaration:

long

HRESULT get_BackColorDay(long* retVal);
HRESULT put_BackColorDay(long newVal);

Siehe auch: Project Übersicht

Gibt die Hintergrundfarbe bei Tagessicht an. Bei Zeitabhängiger Sicht gilt diese Einstellung ebenfalls zwischen Sonnenauf- und untergang.

Project.BackColorNight

Deklaration:

long

HRESULT get_BackColorNight(long* retVal);
HRESULT put_BackColorNight(long newVal);

Siehe auch: Project Übersicht

Die Hintergrundfarbe bei Nachtsicht oder in zeitabhängiger Sicht wenn die Sonne bereits untergegangen ist.

Project.Name

Deklaration:

String

HRESULT get_Name(BSTR* retVal);
HRESULT put_Name(BSTR newVal);

Siehe auch: Project Übersicht

Der Dateiname des Projektes, inklusive Endung und kompletter Pfadangabe.

Project.OriginX

Deklaration:

Single

HRESULT get_OriginX(float* retVal);
HRESULT put_OriginX(float newVal);

Siehe auch: Project Übersicht

Die Weltkoordinaten des Ursprungs des Benutzer-Koordinatensystems. Siehe auch OriginY, OriginZ, und insbesondere WorldRotation, dort finden Sie Beispielcode zur Umrechnung von Welt- in Benutzerkoordinaten.

Project.OriginY

Deklaration:

Single

HRESULT get_OriginY(float* retVal);
HRESULT put_OriginY(float newVal);

Siehe auch: Project Übersicht

Die Weltkoordinaten des Ursprungs des Benutzer-Koordinatensystems. Siehe auch OriginX, OriginZ, und insbesondere WorldRotation, dort finden Sie Beispielcode zur Umrechnung von Welt- in Benutzerkoordinaten.

Project.OriginZ

Deklaration:

Single

HRESULT get_OriginZ(float* retVal);
HRESULT put_OriginZ(float newVal);

Siehe auch: Project Übersicht

Die Weltkoordinaten des Ursprungs des Benutzer-Koordinatensystems. Siehe auch OriginX, OriginY und insbesondere WorldRotation, dort finden Sie Beispielcode zur Umrechnung von Welt- in Benutzerkoordinaten.

Project.PaperHeight

Deklaration:

Single

HRESULT get_PaperHeight(float* retVal);
HRESULT put_PaperHeight(float newVal);

Siehe auch: Project Übersicht

Die Papierhöhe des aktuellen Papierformates. Wie alle Maße ist auch dieser Wert in Metern(!).

Project.PaperSize

Deklaration:

short

HRESULT get_PaperSize(short* retVal);
HRESULT put_PaperSize(short newVal);

Siehe auch: Project Übersicht

Ein Code für das gewählte Papierformat. Entsprechende symbolische Konstanten finden Sie im Kapitel Projekteinstellungen.

Project.PaperSizeName

Deklaration:

String

HRESULT get_PaperSizeName(BSTR* retVal);
HRESULT put_PaperSizeName(BSTR newVal);

Siehe auch: Project Übersicht

Der Name des aktuellen Papierformates.

Project.PaperSizePrinter

Deklaration:

short

HRESULT get_PaperSizePrinter(short* retVal);
HRESULT put_PaperSizePrinter(short newVal);

Siehe auch: Project Übersicht

Falls PaperSize auf ACPF_Drucker steht, gibt dieses Attribut an, welche Größe der Drucker vorgibt. Entsprechende symbolische Konstanten finden Sie im Kapitel Projekteinstellungen.

Project.PaperWidth

Deklaration:

Single

HRESULT get_PaperWidth(float* retVal);
HRESULT put_PaperWidth(float newVal);

Siehe auch: Project Übersicht

Die Papierbreite (in Meter!).

Project.Scale

Deklaration:

short

HRESULT get_Scale(short* retVal);
HRESULT put_Scale(short newVal);

Siehe auch: Project Übersicht

Der gewählte Maßstab. Bei der Einstellung 1:50 hätte dieses Attribut den Wert 50.

Project.SheetCenterX

Deklaration:

Single

```
HRESULT get_SheetCenterX(float* retVal);
HRESULT put_SheetCenterX(float newVal);
```

Siehe auch: Project Übersicht

Project.SheetCenterY

Deklaration:

Single

HRESULT get_SheetCenterY(float* retVal);
HRESULT put_SheetCenterY(float newVal);

Siehe auch: Project Übersicht

Project.ShortName

Deklaration:

String (nur lesbar)

HRESULT get_ShortName(BSTR* retVal);

Siehe auch: Project Übersicht

Der Hauptteil des Projektnamens - ohne Endung und Pfadangabe.

Project.Unit

Deklaration:

short

HRESULT get_Unit(short* retVal);
HRESULT put_Unit(short newVal);

Siehe auch: Project Übersicht

Die gewählte Maßeinheit. Entsprechende symbolische Konstanten finden Sie unter Projekteinstellungen.

Achtung: diese Maßeinheit ist die für die ArCon Planung verwendete Maßeinheit - sie hat nichts mit Längen- oder Koordinatenangaben Ihres Makros zu tun! Makros kommunizieren mit ArCon immer in der Einheit Meter.

Project.WorldRotation

Deklaration:

Single

HRESULT get_WorldRotation(float* retVal);
HRESULT put_WorldRotation(float newVal);

Siehe auch: Project Übersicht

Gibt den Winkel zwischen Benutzerkoordinatensystem und Weltkoordinatensystem an. Siehe auch OriginX, OriginY, OriginZ.

Um einen Punkt in Weltkoordinaten in Benutzerkoordinaten umzurechnen, können Sie z.B. in Visual Basic folgende Routinen verwenden:

```
Public Sub DreheUm(ByVal zx As Single, ByVal zy As Single, _
    ByVal w As Single, _
    ByRef x As Single, ByRef y As Single)
    Dim ox As Single, oy As Single, s As Single, c As Single
    s = Sin(w)
    c = Cos(w)
    ox = x - zx
    oy = y - zy
    x = ox * c - oy * s + zx
    y = ox * s + oy * c + zy
End Sub

Public Sub World2User(ByRef x As Single, ByRef y As Single)
    DreheUm ArCon.CurrentProject.OriginX, _
  ArCon.CurrentProject.OriginY, _
  -ArCon.CurrentProject.WorldRotation, x, y
    x = x - ArCon.CurrentProject.OriginX
    y = y - ArCon.CurrentProject.OriginY
End Sub
```

Gaube

Gaube Übersicht

Eigenschaften

FrontWall	Vorderwand der Gaube
History	Daten zur Identifikation im ehemaligen Projekt für nachgeladene Gebäude
ID	Eine eindeutige Kennung
LeftWall	Linke Wand der Gaube
RightWall	Rechte Wand der Gaube
Roof	Dach, zu dem die Gaube gehört
Story	Stockwerk, zu dem die Gaube gehört
TextureCount	Anzahl der max. verfügbaren Texturen
Type	Art der Gaube (Auswahl des Variantenschalters
Vertices	Eckpunkte der Gaubenkonstruktion (Eingabepunkte)

Methoden

GetAufgesetzt	Liefert den Schalter aufgesetzt 0/1
GetAufgesetzteFluegel	Liefert den Schalter aufgesetzte Fluegel 0/1
GetBreite	Liefert die Gaubenbreite
GetBreiteOben	Liefert die Breite der Gaube am Dachbruch (nur Trapez- und Fledermausgaube)
GetCursorAbstand	Liefert den Abstand der Gaube zum Cursor
GetDetailsUebernehmen	Liefert den Schalter Details uebernehmen
GetFluegelBreite	Liefert die Fluegelbreite (nur Trapez- und Fledermausgauben)
GetMitBlende	Liefert den Schalter mit Blende (nur Rundgaube)
GetNeigungLinks	Liefert die Dachneigung links
GetNeigungRechts	Liefert die Dachneigung rechts
GetOeffnungBisDecke	Liefert den Schalter Oeffnung bis zur Decke 0/1
GetSchnittHoehe	Liefert die Gaubenhöhe im Schnitt
GetSchnittNeigung	Liefert die Dachneigung im Schnitt (Schlepp-, Trapetz- , Walm- und Fledermausgaube)
GetSchnittTraufHoehe	Liefert die Traufhöhe im Schnitt (nur Walmdachgaube)
GetSchnittUeberstand	Liefert den Dachüberstand im Schnitt
GetStich	Liefert den Stich (nur Rundgauben)
GetTexture	Liefert eine Textur
GetTraufHoehe	Liefert die Traufhöhe links/rechts (Stehende- ,Walmdach- und Rundgaube)
GetUeberstandLinks	Liefert den Dachüberstand links
GetUeberstandRechts	Liefert den Dachüberstand rechts
GetWandDicke	Liefert die Gaubenwanddicke der Frontwand
SetAufgesetzt	Ändert für aufgesetzte Gauben
SetAufgesetzteFluegel	Ändert den Schalter für aufgesetzte Fluegel
SetBreite	Ändert die Gaubenbreite

SetBreiteOben	Ändert die Breite der Gaube am Dachbruch (nur Trapez- und Fledermausgaube)
SetCursorAbstand	Ändert den Abstand zum Cursor
SetDetailsUebernehmen	Ändert den Schalter Details uebernehmen
SetFluegelBreite	Ändert die Fluegelbreite (nur Trapez- und Fledermausgauben)
SetMitBlende	Ändert den Schalter mit Blende (nur Rundgaube)
SetNeigungLinks	Ändert die Dachneigung links
SetNeigungRechts	Ändert die Dachneigung rechts
SetOeffnungBisDecke	Ändert den Schalter Oeffnung bis zur Decke
SetSchnittHoehe	Ändert die Gaubenhöhe im Schnitt
SetSchnittNeigung	Ändert die Dachneigung im Schnitt
SetSchnittTraufHoehe	Ändert die Traufhöhe im Schnitt (nur Walmdachgaube)
SetSchnittUeberstand	Ändert den Dachüberstand im Schnitt
SetStich	Ändert den Stich (nur Rundgauben)
SetTexture	Tauscht eine Textur aus
SetTraufHoehe	Ändert die Traufhöhe links/rechts (Stehende- ,Walmdach- und Rundgaube)
SetUeberstandLinks	Ändert den Dachüberstand links
SetUeberstandRechts	Ändert den Dachüberstand rechts
SetWandDicke	Ändert die Gaubenwanddicke der Frontwand

Gaube.FrontWall

Deklaration:

Wall (nur lesbar)

HRESULT get_FrontWall(IWall** retVal);

Siehe auch: Gaube Übersicht, Wall

Gaube.History

Deklaration:

IDHistory (nur lesbar)

HRESULT get_History(IIDHistory** retVal);

Siehe auch: Gaube Übersicht, IDHistory

Gaube.ID

Deklaration:

long (nur lesbar)

HRESULT get_ID(long* retVal);

Siehe auch: Gaube Übersicht

Gaube.LeftWall

Deklaration:

Wall (nur lesbar)

HRESULT get_LeftWall(IWall** retVal);

Siehe auch: Gaube Übersicht, Wall

Gaube.RightWall

Deklaration:

Wall (nur lesbar)

HRESULT get_RightWall(IWall** retVal);

Siehe auch: Gaube Übersicht, Wall

Gaube.Roof

Deklaration:

Roof (nur lesbar)

HRESULT get_Roof(IRoof** retVal);

Siehe auch: Gaube Übersicht, Roof

Gaube.Story

Deklaration:

Story (nur lesbar)

HRESULT get_Story(IStory** retVal);

Siehe auch: Gaube Übersicht, Story

Gaube.TextureCount

Deklaration:

long (nur lesbar)

HRESULT get_TextureCount(long* retVal);

Siehe auch: Gaube Übersicht

Gaube.Type

Deklaration:

long

HRESULT get_Type(long* retVal);
HRESULT put_Type(long newVal);

Siehe auch: Gaube Übersicht

Gaube.Vertices

Deklaration:

Point2DCollection (nur lesbar)

HRESULT get_Vertices(IPoint2DCollection** retVal);

Siehe auch: Gaube Übersicht, Point2DCollection

Gaube.GetAufgesetzt

Deklaration:

```
Function GetAufgesetzt() as Boolean
```

```
HRESULT GetAufgesetzt(VARIANT_BOOL* retVal);
```

Siehe auch: Gaube Übersicht

Gaube.GetAufgesetzteFluegel

Deklaration:

```
Function GetAufgesetzteFluegel() as Boolean
```

```
HRESULT GetAufgesetzteFluegel(VARIANT_BOOL* retVal)
;
```

Siehe auch: Gaube Übersicht

Gaube.GetBreite

Deklaration:

```
Function GetBreite() as double
```

```
HRESULT GetBreite(double* retVal);
```

Siehe auch: Gaube Übersicht

Gaube.GetBreiteOben

Deklaration:

```
Function GetBreiteOben() as double
```

```
HRESULT GetBreiteOben(double* retVal);
```

Siehe auch: Gaube Übersicht

Gaube.GetCursorAbstand

Deklaration:

```
Function GetCursorAbstand() as double
```

```
HRESULT GetCursorAbstand(double* retVal);
```

Siehe auch: Gaube Übersicht

Gaube.GetDetailsUebernehmen

Deklaration:

```
Function GetDetailsUebernehmen() as Boolean
```

```
HRESULT GetDetailsUebernehmen(VARIANT_BOOL*
retVal);
```

Siehe auch: Gaube Übersicht

Gaube.GetFluegelBreite

Deklaration:

```
Function GetFluegelBreite() as double
```

```
HRESULT GetFluegelBreite(double* retVal);
```

Siehe auch: Gaube Übersicht

Gaube.GetMitBlende

Deklaration:

```
Function GetMitBlende() as Boolean
```

```
HRESULT GetMitBlende(VARIANT_BOOL* retVal);
```

Siehe auch: Gaube Übersicht

Gaube.GetNeigungLinks

Deklaration:

```
Function GetNeigungLinks() as double
```

```
HRESULT GetNeigungLinks(double* retVal);
```

Siehe auch: Gaube Übersicht

Gaube.GetNeigungRechts

Deklaration:

```
Function GetNeigungRechts() as double
```

```
HRESULT GetNeigungRechts(double* retVal);
```

Siehe auch: Gaube Übersicht

Gaube.GetOeffnungBisDecke

Deklaration:

```
Function GetOeffnungBisDecke() as Boolean
```

```
HRESULT GetOeffnungBisDecke(VARIANT_BOOL* retVal);
```

Siehe auch: Gaube Übersicht

Gaube.GetSchnittHoehe

Deklaration:

```
Function GetSchnittHoehe() as double
```

```
HRESULT GetSchnittHoehe(double* retVal);
```

Siehe auch: Gaube Übersicht

Gaube.GetSchnittNeigung

Deklaration:

```
Function GetSchnittNeigung() as double
```

```
HRESULT GetSchnittNeigung(double* retVal);
```

Siehe auch: Gaube Übersicht

Gaube.GetSchnittTraufHoehe

Deklaration:

```
Function GetSchnittTraufHoehe() as double
```

```
HRESULT GetSchnittTraufHoehe(double* retVal);
```

Siehe auch: Gaube Übersicht

Gaube.GetSchnittUeberstand

Deklaration:

```
Function GetSchnittUeberstand() as double
```

```
HRESULT GetSchnittUeberstand(double* retVal);
```

Siehe auch: Gaube Übersicht

Gaube.GetStich

Deklaration:

Function GetStich() as double

HRESULT GetStich(double* retVal);

Siehe auch: Gaube Übersicht

Gaube.GetTexture

Deklaration:

```
Function GetTexture(ByVal index as long) as String
```

```
HRESULT GetTexture(long index, BSTR* retVal);
```

Siehe auch: Gaube Übersicht

Gaube.GetTraufHoehe

Deklaration:

```
Function GetTraufHoehe() as double
```

```
HRESULT GetTraufHoehe(double* retVal);
```

Siehe auch: Gaube Übersicht

Gaube.GetUeberstandLinks

Deklaration:

```
Function GetUeberstandLinks() as double
```

```
HRESULT GetUeberstandLinks(double* retVal);
```

Siehe auch: Gaube Übersicht

Gaube.GetUeberstandRechts

Deklaration:

```
Function GetUeberstandRechts() as double
```

```
HRESULT GetUeberstandRechts(double* retVal);
```

Siehe auch: Gaube Übersicht

Gaube.GetWandDicke

Deklaration:

```
Function GetWandDicke() as double
```

```
HRESULT GetWandDicke(double* retVal);
```

Siehe auch: Gaube Übersicht

Gaube.SetAufgesetzt

Deklaration:

```
Sub SetAufgesetzt(ByVal nVal as Boolean, ByVal update as
Boolean)
```

```
HRESULT SetAufgesetzt(VARIANT_BOOL nVal,
VARIANT_BOOL update);
```

Siehe auch: Gaube Übersicht

Gaube.SetAufgesetzteFluegel

Deklaration:

```
Sub SetAufgesetzteFluegel(ByVal nVal as Boolean, ByVal
update as Boolean)
```

```
HRESULT SetAufgesetzteFluegel(VARIANT_BOOL nVal,
VARIANT_BOOL update);
```

Siehe auch: Gaube Übersicht

Gaube.SetBreite

Deklaration:

```
Sub SetBreite(ByVal nVal as double, ByVal update as
Boolean)
```

```
HRESULT SetBreite(double nVal, VARIANT_BOOL update)
;
```

Siehe auch: Gaube Übersicht

Gaube.SetBreiteOben

Deklaration:

```
Sub SetBreiteOben(ByVal nVal as double, ByVal update as
Boolean)
```

```
HRESULT SetBreiteOben(double nVal, VARIANT_BOOL
update);
```

Siehe auch: Gaube Übersicht

Gaube.SetCursorAbstand

Deklaration:

```
Sub SetCursorAbstand(ByVal nVal as double, ByVal update
as Boolean)
```

```
HRESULT SetCursorAbstand(double nVal, VARIANT_BOOL
update);
```

Siehe auch: Gaube Übersicht

Gaube.SetDetailsUebernehmen

Deklaration:

```
Sub SetDetailsUebernehmen(ByVal nVal as Boolean, ByVal
update as Boolean)
```

```
HRESULT SetDetailsUebernehmen(VARIANT_BOOL nVal,
VARIANT_BOOL update);
```

Siehe auch: Gaube Übersicht

Gaube.SetFluegelBreite

Deklaration:

```
Sub SetFluegelBreite(ByVal nVal as double, ByVal update
as Boolean)
```

```
HRESULT SetFluegelBreite(double nVal, VARIANT_BOOL
update);
```

Siehe auch: Gaube Übersicht

Gaube.SetMitBlende

Deklaration:

```
Sub SetMitBlende(ByVal nVal as Boolean, ByVal update as
Boolean)
```

```
HRESULT SetMitBlende(VARIANT_BOOL nVal,
VARIANT_BOOL update);
```

Siehe auch: Gaube Übersicht

Gaube.SetNeigungLinks

Deklaration:

```
Sub SetNeigungLinks(ByVal nVal as double, ByVal update
as Boolean)
```

```
HRESULT SetNeigungLinks(double nVal, VARIANT_BOOL
update);
```

Siehe auch: Gaube Übersicht

Gaube.SetNeigungRechts

Deklaration:

```
Sub SetNeigungRechts(ByVal nVal as double, ByVal update
as Boolean)
```

```
HRESULT SetNeigungRechts(double nVal, VARIANT_BOOL
update);
```

Siehe auch: Gaube Übersicht

Gaube.SetOeffnungBisDecke

Deklaration:

```
Sub SetOeffnungBisDecke(ByVal nVal as Boolean, ByVal
update as Boolean)
```

```
HRESULT SetOeffnungBisDecke(VARIANT_BOOL nVal,
VARIANT_BOOL update);
```

Siehe auch: Gaube Übersicht

Gaube.SetSchnittHoehe

Deklaration:

```
Sub SetSchnittHoehe(ByVal nVal as double, ByVal update
as Boolean)
```

```
HRESULT SetSchnittHoehe(double nVal, VARIANT_BOOL
update);
```

Siehe auch: Gaube Übersicht

Gaube.SetSchnittNeigung

Deklaration:

```
Sub SetSchnittNeigung(ByVal nVal as double, ByVal update
as Boolean)
```

```
HRESULT SetSchnittNeigung(double nVal, VARIANT_BOOL
update);
```

Siehe auch: Gaube Übersicht

Gaube.SetSchnittTraufHoehe

Deklaration:

```
Sub SetSchnittTraufHoehe(ByVal nVal as double, ByVal
update as Boolean)
```

```
HRESULT SetSchnittTraufHoehe(double nVal,
VARIANT_BOOL update);
```

Siehe auch: Gaube Übersicht

Gaube.SetSchnittUeberstand

Deklaration:

```
Sub SetSchnittUeberstand(ByVal nVal as double, ByVal
update as Boolean)
```

```
HRESULT SetSchnittUeberstand(double nVal,
VARIANT_BOOL update);
```

Siehe auch: Gaube Übersicht

Gaube.SetStich

Deklaration:

```
Sub SetStich(ByVal nVal as double, ByVal update as
Boolean)
```

```
HRESULT SetStich(double nVal, VARIANT_BOOL update);
```

Siehe auch: Gaube Übersicht

Gaube.SetTexture

Deklaration:

```
Function SetTexture(ByVal index as long, ByVal
TextureName as String) as Boolean
```

```
HRESULT SetTexture(long index, BSTR TextureName,
VARIANT_BOOL* retVal);
```

Siehe auch: Gaube Übersicht

Gaube.SetTraufHoehe

Deklaration:

```
Sub SetTraufHoehe(ByVal nVal as double, ByVal update as
Boolean)
```

```
HRESULT SetTraufHoehe(double nVal, VARIANT_BOOL
update);
```

Siehe auch: Gaube Übersicht

Gaube.SetUeberstandLinks

Deklaration:

```
Sub SetUeberstandLinks(ByVal nVal as double, ByVal
update as Boolean)
```

```
HRESULT SetUeberstandLinks(double nVal,
VARIANT_BOOL update);
```

Siehe auch: Gaube Übersicht

Gaube.SetUeberstandRechts

Deklaration:

```
Sub SetUeberstandRechts(ByVal nVal as double, ByVal
update as Boolean)
```

```
HRESULT SetUeberstandRechts(double nVal,
VARIANT_BOOL update);
```

Siehe auch: Gaube Übersicht

Gaube.SetWandDicke

Deklaration:

```
Sub SetWandDicke(ByVal nVal as double, ByVal update as
Boolean)
```

```
HRESULT SetWandDicke(double nVal, VARIANT_BOOL
update);
```

Siehe auch: Gaube Übersicht

Texture

Eine Texturbeschreibung. Texturen werden von vielen ArCon Objekten verwendet, um eine realistische Visualisierung zu erreichen. Wesentliche Eigenschaft ist der Name, da er die verwendete Bitmap bestimmt.

Texture Übersicht
Eigenschaften

Angle	Drehwinkel für Textur
Color	Konstante Farbe
Height	Höhe, wenn nicht die der Bitmap-Datei
MixColorAndTexture	Konstante Farbe und Texturfarbe mischen?
Name	Dateiname der Textur (inkl. Pfad, ggfs. mit '>')
UseOriginalSize	In Bitmap-Datei gespeicherte Größe verwenden?
UseTexture	Soll die Textur verwendet werden?
Width	Breite, wenn nicht die der Bitmap-Datei
xRaport	Größe der Textur in x-Richtung
yRaport	Größe der Textur in y-Richtung

Texture.Angle

Deklaration:

Single

HRESULT get_Angle(float* retVal);
HRESULT put_Angle(float newVal);

Siehe auch: Texture Übersicht

Der Drehwinkel der Textur gegenüber dem Objekt-Koordinatensystem in Bogenmaß. Ist dieser Wert 0, wird die Textur nicht gedreht, ist er Pi/2 steht die Textur quer.

Texture.Color

Deklaration:

long

HRESULT get_Color(long* retVal);
HRESULT put_Color(long newVal);

Siehe auch: Texture Übersicht

Eine Hintergrundfarbe (RGB), die optional mit der Textur gemischt werden kann.

Texture.Height

Deklaration:

Single

HRESULT get_Height(float* retVal);
HRESULT put_Height(float newVal);

Siehe auch: Texture Übersicht

Die Höhe der Textur, falls nicht die Originalgröße aus der Bitmapdatei verwendet wird.

Texture.MixColorAndTexture

Deklaration:

Boolean

HRESULT get_MixColorAndTexture(VARIANT_BOOL* retVal);

HRESULT put_MixColorAndTexture(VARIANT_BOOL newVal);

Siehe auch: Texture Übersicht

Wenn dieser Wert “ Wahr” ist, wird die Bitmap mit der eingestellten Farbe “ Color” gemischt.

Texture.Name

Deklaration:

String

HRESULT get_Name(BSTR* retVal);
HRESULT put_Name(BSTR newVal);

Siehe auch: Texture Übersicht

Gibt den Dateinamen der Textur Bitmap an. Dabei kann der ArCon Standard-Pfad für Texturen durch ein “ >” angegeben werden, also z.B. “ >Texturen\Boden\Fliessen\Bfliess05.bmp” .

Texture.UseOriginalSize

Deklaration:

Boolean

HRESULT get_UseOriginalSize(VARIANT_BOOL* retVal);
HRESULT put_UseOriginalSize(VARIANT_BOOL newVal);

Siehe auch: Texture Übersicht

Wenn dieser Wert “ Wahr” ist, wird die Größe der Textur aus dem Bitmapheader ermittelt (biXPelsPerMeter und biYPelsPerMeter geben die horizontale und vertikale Auflösung der Bitmap an, sodaß die Größe eines Pixels bekannt ist). Leider speichern viele Programme keinen korrekten Bitmapheader, sodaß diese Größenangabe nicht immer sinnvolle Werte liefert. In solchen Fällen kann diese Eigenschaft auf “ Falsch” gestellt werden und mit den Attributen Height und Width die korrekten Ausmaße der Bitmap angegeben werden.

Texture.UseTexture

Deklaration:

Boolean

HRESULT get_UseTexture(VARIANT_BOOL* retVal);
HRESULT put_UseTexture(VARIANT_BOOL newVal);

Siehe auch: Texture Übersicht

Nur wenn diese Eigenschaft “ Wahr” ist, wird die Bitmap verwendet. Ansonsten wird eine einfache Schattierung mit der durch Color definierten Farbe durchgeführt.

Texture.Width

Deklaration:

Single

HRESULT get_Width(float* retVal);
HRESULT put_Width(float newVal);

Siehe auch: Texture Übersicht

Wenn die Größe der Textur nicht aus der Bitmapdatei übernommen wird, bestimmt diese Eigenschaft die Breite.

Texture.xRaport

Deklaration:

Single

HRESULT get_xRaport(float* retVal);
HRESULT put_xRaport(float newVal);

Siehe auch: Texture Übersicht

Originalgröße der Textur in x Richtung

Texture.yRaport

Deklaration:

Single

HRESULT get_yRaport(float* retVal);
HRESULT put_yRaport(float newVal);

Siehe auch: Texture Übersicht

Originalgröße der Textur in y Richtung

Chimney

Ein Schornstein.

Chimney Übersicht

Eigenschaften

Angle	Drehwinkel
Height	Höhe, wenn nicht bis zur Decke
History	Daten zur Identifikation im ehemaligen Projekt für nachgeladene Gebäude
ID	Eine eindeutige Kennung
Remark	Bemerkung
Story	Geschoß, in dem der Schornstein liegt
Texture	Verwendete Textur
Thickness	Tiefe
ToCeiling	Schornstein bis zur Decke?
Type	Art des Schornsteins (Variante des Multifunktionsschalters)
Width	Breite

Methoden

Delete	Schornstein löschen
GetPos	Position des Schornsteins lesen
SetPos	Position des Schornsteins auf x/y im aktuellen Geschoß setzen

Chimney.Angle

Deklaration:

Single

HRESULT get_Angle(float* retVal);
HRESULT put_Angle(float newVal);

Siehe auch: Chimney Übersicht

Der Winkel, um den der Schornstein gegenüber seiner Normallage gedreht ist. In Bogenmaß.

Chimney.Height

Deklaration:

Single

HRESULT get_Height(float* retVal);
HRESULT put_Height(float newVal);

Siehe auch: Chimney Übersicht

Die Höhe des Schornsteins, falls er nicht bis zur Decke reicht. Wenn ToCeiling auf “ Wahr” gesetzt ist, hat dieses Attribut keine Bedeutung.

Chimney.History

Deklaration:

IDHistory (nur lesbar)

HRESULT get_History(IIDHistory** retVal);

Siehe auch: Chimney Übersicht, IDHistory

Falls dieses Chimney beim Ursprünglichen Laden (oder Erstellen) des aktuellen Projektes nicht in diesem Projekt war, sonder aus einem anderen Projekt nachgeladen wurde, liefert diese Eigenschaft das History-Objekt, das die Umsetzung der IDs beschreibt. Andernfalls ist diese Eigenschaft “ Nothing” .

Ein Anwendungsbeispiel für History-Objekte finden Sie im Kapitel Eindeutige Kennungen.

Chimney.ID

Deklaration:

long (nur lesbar)

HRESULT get_ID(long* retVal);

Siehe auch: Chimney Übersicht

Gibt eine eindeutige Kennung dieses Chimney an. Diese Kennung ist persistent, das heißt, sie bleibt innerhalb des aktuellen Projektes erhalten. Sie können diese Kennung benutzen, um Chimneys miteinander zu vergleichen oder zusätzliche Daten zum Chimney in einer Datenbank abzulegen.

Die ID ist aber nur innerhalb des aktuellen Projektes eindeutig, wenn Sie (durch Nachladen einzelner Gebäude) das Objekt in ein anderes Projekt einfügen, erhält es eine neue ID. In diesem Fall müssen Sie sofort bei Beendigung des Ladevorganges mit Hilfe der Eigenschaft History objPropertyDetailsChimneyHistory Ihre zusätzlichen Daten an die neuen Werte anpassen.

Chimney.Remark

Deklaration:

String

HRESULT get_Remark(BSTR* retVal);
HRESULT put_Remark(BSTR newVal);

Siehe auch: Chimney Übersicht

Eine Bemerkung zum Schornstein.

Chimney.Story

Deklaration:

Story (nur lesbar)

HRESULT get_Story(IStory** retVal);

Siehe auch: Chimney Übersicht, Story

Ein Verweis auf das Stockwerk, in dem dieser Schornstein steht.

Chimney.Texture

Deklaration:

Texture (nur lesbar)

HRESULT get_Texture(ITexture** retVal);

Siehe auch: Chimney Übersicht, Texture

Die Textur, mit der der Schornstein tapeziert ist.

Chimney.Thickness

Deklaration:

Single

HRESULT get_Thickness(float* retVal);
HRESULT put_Thickness(float newVal);

Siehe auch: Chimney Übersicht

Die Tiefe des Schornsteins.

Chimney.ToCeiling

Deklaration:

Boolean

HRESULT get_ToCeiling(VARIANT_BOOL* retVal);
HRESULT put_ToCeiling(VARIANT_BOOL newVal);

Siehe auch: Chimney Übersicht

Wenn dieses Attribut auf “ Wahr” gesetzt ist, wird die Eigenschaft Height ignoriert und der Schornstein bis zur Decke des Raumes dargestellt.

Chimney.Type

Deklaration:

long

HRESULT get_Type(long* retVal);
HRESULT put_Type(long newVal);

Siehe auch: Chimney Übersicht

Die Variante des Schornsteins. Typnummern beginnen bei Null und gehen bis AvailableChimneys - 1.

Chimney.Width

Deklaration:

Single

HRESULT get_Width(float* retVal);
HRESULT put_Width(float newVal);

Siehe auch: Chimney Übersicht

Die Breite des Schornsteins.

Chimney.Delete

Deklaration:

Sub Delete()

HRESULT Delete();

Siehe auch: Chimney Übersicht

Löscht den Schornstein.

Chimney.GetPos

Deklaration:

Function GetPos(ByRef X as Single, ByRef Y as Single) as Boolean

HRESULT GetPos(float* X, float* Y, VARIANT_BOOL* retVal);

Siehe auch: Chimney Übersicht

Liefert die Position des Schornsteins.

Der Rückgabewert ist “ Wahr” , wenn die Position des Schornsteins ermittelt werden konnte, “ Falsch” wenn ein Fehler auftrat (z.B. der Schornstein zwar erzeugt aber noch nicht plaziert war) .

Die Parameter sind:

x	x-Koordinate des Schornsteinzentrums
y	y-Koordinate

Chimney.SetPos

Deklaration:

```
Function SetPos(ByVal X as Single, ByVal Y as Single) as
Boolean
```

```
HRESULT SetPos(float X, float Y, VARIANT_BOOL* retVal);
```

Siehe auch: Chimney Übersicht

Setzt die Position des Schornsteins.

Rückgabewert ist “ Wahr” wenn der Schornstein erfolgreich verschoben werden konnte, “ Falsch” andernfalls.

Die Parameter sind:

x	neue x-Koordinate des Schornsteinmittelpunktes
y	neue y-Koordinate

Support

Eine ArCon Stütze.

Support Übersicht

Eigenschaften

Angle	Drehwinkel
Height	Höhe, wenn nicht bis zur Decke
History	Daten zur Identifikation im ehemaligen Projekt für nachgeladene Gebäude
ID	Eine eindeutige Kennung
Remark	Bemerkung
Story	Geschoß, in dem die Stütze liegt
Texture	Verwendete Textur
Thickness	Tiefe
ToCeiling	Stütze bis zur Decke?
Type	Art der Stütze (Variante des Multifunktionsschalters)
Width	Breite

Methoden

Delete	Stütze löschen
GetPos	Position der Stütze lesen
SetPos	Stütze an Position x/y im aktuellen Geschoß setzen

Support.Angle

Deklaration:

Single

HRESULT get_Angle(float* retVal);
HRESULT put_Angle(float newVal);

Siehe auch: Support Übersicht

Drehwinkel der Stütze gegenüber ihrer Standardausrichtung in Bogenmaß.

Support.Height

Deklaration:

Single

HRESULT get_Height(float* retVal);
HRESULT put_Height(float newVal);

Siehe auch: Support Übersicht

Höhe der Stütze, falls Sie nicht bis zur Decke reicht, also ToCeiling “ Wahr” ist.

Support.History

Deklaration:

IDHistory (nur lesbar)

HRESULT get_History(IIDHistory** retVal);

Siehe auch: Support Übersicht, IDHistory

Falls dieses Support beim Ursprünglichen Laden (oder Erstellen) des aktuellen Projektes nicht in diesem Projekt war, sonder aus einem anderen Projekt nachgeladen wurde, liefert diese Eigenschaft das History-Objekt, das die Umsetzung der IDs beschreibt. Andernfalls ist diese Eigenschaft “ Nothing” .

Ein Anwendungsbeispiel für History-Objekte finden Sie im Kapitel Eindeutige Kennungen.

Support.ID

Deklaration:

long (nur lesbar)

HRESULT get_ID(long* retVal);

Siehe auch: Support Übersicht

Gibt eine eindeutige Kennung dieses Support an. Diese Kennung ist persistent, das heißt, sie bleibt innerhalb des aktuellen Projektes erhalten. Sie können diese Kennung benutzen, um Supports miteinander zu vergleichen oder zusätzliche Daten zum Support in einer Datenbank abzulegen.

Die ID ist aber nur innerhalb des aktuellen Projektes eindeutig, wenn Sie (durch Nachladen einzelner Gebäude) das Objekt in ein anderes Projekt einfügen, erhält es eine neue ID. In diesem Fall müssen Sie sofort bei Beendigung des Ladevorganges mit Hilfe der Eigenschaft History objPropertyDetailsSupportHistory Ihre zusätzlichen Daten an die neuen Werte anpassen.

Support.Remark

Deklaration:

String

HRESULT get_Remark(BSTR* retVal);
HRESULT put_Remark(BSTR newVal);

Siehe auch: Support Übersicht
Beliebige Bemerkung zur Stütze.

Support.Story

Deklaration:

Story (nur lesbar)

HRESULT get_Story(IStory** retVal);

Siehe auch: Support Übersicht, Story

Verweis auf das Stockwerk, in dem sich die Stütze befindet.

Support.Texture

Deklaration:

Texture (nur lesbar)

HRESULT get_Texture(ITexture** retVal);

Siehe auch: Support Übersicht, Texture

Die Textur der Stütze.

Diese Eigenschaft ist zwar nur lesbar, aber das so erhaltene Textur-Objekt können Sie manipulieren, um die Textur der Stütze zu verändern!

Support.Thickness

Deklaration:

Single

HRESULT get_Thickness(float* retVal);
HRESULT put_Thickness(float newVal);

Siehe auch: Support Übersicht

Die Tiefe der Stütze in Meter.

Support.ToCeiling

Deklaration:

Boolean

HRESULT get_ToCeiling(VARIANT_BOOL* retVal);
HRESULT put_ToCeiling(VARIANT_BOOL newVal);

Siehe auch: Support Übersicht

Gibt an, ob die Stütze bis zur Decke reicht, oder ob die Height Eigenschaft gültig ist.

Support.Type

Deklaration:

long

HRESULT get_Type(long* retVal);
HRESULT put_Type(long newVal);

Siehe auch: Support Übersicht

Der Typ der Stütze. Gültige Werte liegen zwischen 0 und AC_MaxSupportType.

Support.Width

Deklaration:

Single

HRESULT get_Width(float* retVal);
HRESULT put_Width(float newVal);

Siehe auch: Support Übersicht

Die Breite der Stütze in Meter.

Support.Delete

Deklaration:

Sub Delete()

HRESULT Delete();

Siehe auch: Support Übersicht

Löscht die Stütze.

Support.GetPos

Deklaration:

```
Function GetPos(ByRef X as Single, ByRef Y as Single) as
Boolean
```

```
HRESULT GetPos(float* X, float* Y, VARIANT_BOOL*
retVal);
```

Siehe auch: Support Übersicht

Ermittelt die Position des Stützenmittelpunktes.

Parameter sind:

x, y	Referenzen, die auf die x und y Koordinaten des Mittelpunktes der Stütze gesetzt werden.

Support.SetPos

Deklaration:

```
Function SetPos(ByVal X as Single, ByVal Y as Single) as
Boolean
```

```
HRESULT SetPos(float X, float Y, VARIANT_BOOL*
retVal);
```

Siehe auch: Support Übersicht

Setzt die Position der Stütze.

Parameter sind:

x, y	Koordinaten des Mittelpunktes der Stütze.

Door

Eine ArCon Tür. ArCon unterscheidet zwischen geladenen und konstruierten Türtypen. Nicht alle Eigenschaften dieses Objektes haben Einfluß auf alle vorhandenen Varianten von Türen.

Door Übersicht

Eigenschaften

Area	Türfläche
AreaFormula	Formel für Türfläche
FrameTexture	Textur der Zarge
Height	Höhe der Tür
History	Daten zur Identifikation im ehemaligen Projekt für nachgeladene Gebäude
HungLeft	Links angeschlagen?
ID	Eine eindeutige Kennung
LeftSegment	Erstes Wandseitenobjekt, in dem die Tür 'sitzt'
LeftWingAngle	Öffnungswinkel des linken Flügels
LeftWingTexture	Textur des linken /einzigen Türflügels
LeftWingType	Art des linken Flügels
OpensInwards	Zur gegenüberliegenden Seite öffnend?
Remark	Bemerkung
RightSegment	Zweites Wandseitenobjekt, in dem die Tür 'sitzt'
RightWingAngle	Öffnungswinkel des rechten Flügels
RightWingTexture	Textur des rechten Türflügels bei zweiflügeligen Türen
RightWingType	Art des rechten Flügels
Type	Türtyp (Variante des Multifunktionsschalters)
Wall	Wandobjekt, in dem die Tür 'sitzt'
Width	Breite der Tür

Methoden

Delete	Tür löschen
GetPolygons	Liefert die Wandlöcher zu beiden Seiten der Tür, die durch die Tür entstehen sowie deren Tiefe
GetPos	Position der Tür lesen
SetPos	Position der Tür setzen

Door.Area

Deklaration:

Single (nur lesbar)

HRESULT get_Area(float* retVal);

Siehe auch: Door Übersicht

Die Fläche der Tür in Quadratmetern.

Door.AreaFormula

Deklaration:

String (nur lesbar)

HRESULT get_AreaFormula(BSTR* retVal);

Siehe auch: Door Übersicht

Eine symbolische Formel zur Berechnung der Türfläche.

Door.FrameTexture

Deklaration:

Texture

HRESULT get_FrameTexture(ITexture** retVal);
HRESULT put_FrameTexture(ITexture* newVal);

Siehe auch: Door Übersicht, Texture

Die Textur der Zarge.

Door.Height

Deklaration:

Single

HRESULT get_Height(float* retVal);
HRESULT put_Height(float newVal);

Siehe auch: Door Übersicht

Die Höhe der Tür.

Door.History

Deklaration:

IDHistory (nur lesbar)

HRESULT get_History(IIDHistory** retVal);

Siehe auch: Door Übersicht, IDHistory

Falls dieses Door beim Ursprünglichen Laden (oder Erstellen) des aktuellen Projektes nicht in diesem Projekt war, sonder aus einem anderen Projekt nachgeladen wurde, liefert diese Eigenschaft das History-Objekt, das die Umsetzung der IDs beschreibt. Andernfalls ist diese Eigenschaft “ Nothing” .

Ein Anwendungsbeispiel für History-Objekte finden Sie im Kapitel Eindeutige Kennungen.

Door.HungLeft

Deklaration:

Boolean

HRESULT get_HungLeft(VARIANT_BOOL* retVal);
HRESULT put_HungLeft(VARIANT_BOOL newVal);

Siehe auch: Door Übersicht

Befindet sich das Scharnier auf der linken oder der rechten Seite der Tür?

Door.ID

Deklaration:

long (nur lesbar)

HRESULT get_ID(long* retVal);

Siehe auch: Door Übersicht

Gibt eine eindeutige Kennung dieses Door an. Diese Kennung ist persistent, das heißt, sie bleibt innerhalb des aktuellen Projektes erhalten. Sie können diese Kennung benutzen, um Doors miteinander zu vergleichen oder zusätzliche Daten zum Door in einer Datenbank abzulegen.

Die ID ist aber nur innerhalb des aktuellen Projektes eindeutig, wenn Sie (durch Nachladen einzelner Gebäude) das Objekt in ein anderes Projekt einfügen, erhält es eine neue ID. In diesem Fall müssen Sie sofort bei Beendigung des Ladevorganges mit Hilfe der Eigenschaft History objPropertyDetailsDoorHistory Ihre zusätzlichen Daten an die neuen Werte anpassen.

Door.LeftSegment

Deklaration:

WallSegment (nur lesbar)

HRESULT get_LeftSegment(IWallSegment** retVal);

Siehe auch: Door Übersicht, WallSegment

Eines der Wandsegmente, in dem die Tür sitzt. Siehe auch RightSegment.

Door.LeftWingAngle

Deklaration:

Single

HRESULT get_LeftWingAngle(float* retVal);
HRESULT put_LeftWingAngle(float newVal);

Siehe auch: Door Übersicht

Öffnungswinkel des linken Flügels der Tür. In Bogenmaß.

Door.LeftWingTexture

Deklaration:

Texture

HRESULT get_LeftWingTexture(ITexture** retVal);
HRESULT put_LeftWingTexture(ITexture* newVal);

Siehe auch: Door Übersicht, Texture

Die Textur des linken Türflügels.

Door.LeftWingType

Deklaration:

long

HRESULT get_LeftWingType(long* retVal);
HRESULT put_LeftWingType(long newVal);

Siehe auch: Door Übersicht

Art des linken Türflügels.

Door.OpensInwards

Deklaration:

Boolean

HRESULT get_OpensInwards(VARIANT_BOOL* retVal);
HRESULT put_OpensInwards(VARIANT_BOOL newVal);

Siehe auch: Door Übersicht

Gibt an, ob die Tür sich nach innen oder außen öffnet.

Door.Remark

Deklaration:

String

HRESULT get_Remark(BSTR* retVal);
HRESULT put_Remark(BSTR newVal);

Siehe auch: Door Übersicht
Eine Bemerkung zur Tür.

Door.RightSegment

Deklaration:

WallSegment (nur lesbar)

HRESULT get_RightSegment(IWallSegment** retVal);

Siehe auch: Door Übersicht, WallSegment

Die andere Wandseite, in der die Tür sitzt. Siehe auch LeftSegment.

Door.RightWingAngle

Deklaration:

Single

HRESULT get_RightWingAngle(float* retVal);
HRESULT put_RightWingAngle(float newVal);

Siehe auch: Door Übersicht

Der Öffnungswinkel des rechten Türflügels. In Bogenmaß.

Door.RightWingTexture

Deklaration:

Texture

HRESULT get_RightWingTexture(ITexture** retVal);
HRESULT put_RightWingTexture(ITexture* newVal);

Siehe auch: Door Übersicht, Texture

Die Textur des rechten Türflügels.

Door.RightWingType

Deklaration:

long

HRESULT get_RightWingType(long* retVal);
HRESULT put_RightWingType(long newVal);

Siehe auch: Door Übersicht

Art des rechten Türflügels.

Door.Type

Deklaration:

long

HRESULT get_Type(long* retVal);
HRESULT put_Type(long newVal);

Siehe auch: Door Übersicht

Variante der Tür. Türen werden dynamisch geladen, der minimale Typcode ist immer 0, der maximale hängt von der jeweiligen Installation ab und kann mit Hilfe der Funktion AvailableDoors ermittelt werden: der maximal gültige Typcode ist AvailableDoors - 1.

Door.Wall

Deklaration:

Wall (nur lesbar)

HRESULT get_Wall(IWall** retVal);

Siehe auch: Door Übersicht, Wall

Verweis auf die Wand, in der die Tür sitzt.

Door.Width

Deklaration:

Single

HRESULT get_Width(float* retVal);
HRESULT put_Width(float newVal);

Siehe auch: Door Übersicht

Breite der Tür.

Door.Delete

Deklaration:

Sub Delete()

HRESULT Delete();

Siehe auch: Door Übersicht

Löscht die Tür.

Door.GetPolygons

Deklaration:

```
Function GetPolygons(ByRef leftPolygon as
Point2DCollection, ByRef rightPolygon as Point2DCollection,
ByRef leftDepth as Single, ByRef rightDepth as Single) as
Boolean
```

```
HRESULT GetPolygons(IPoint2DCollection** leftPolygon,
IPoint2DCollection** rightPolygon, float* leftDepth, float*
rightDepth, VARIANT_BOOL* retVal);
```

Siehe auch: Door Übersicht, Point2DCollection, Point2DCollection

Door.GetPos

Deklaration:

```
Function GetPos(ByRef X as Single, ByRef Y as Single) as
Boolean
```

```
HRESULT GetPos(float* X, float* Y, VARIANT_BOOL*
retVal);
```

Siehe auch: Door Übersicht

Liefert die Position der Tür.

Rückgabewert ist “ Wahr” wenn die Position ermittelt werden konnte, “ Falsch” wenn nicht - z. B. weil die Tür noch gar nicht plaziert wurde.

Die Parameter sind:

X	x-Koordinate des Türmittelpunktes
Y	y-Koordinate

Door.SetPos

Deklaration:

```
Function SetPos(ByVal X as Single, ByVal Y as Single)
as Boolean
```

```
HRESULT SetPos(float X, float Y, VARIANT_BOOL*
retVal);
```

Siehe auch: Door Übersicht

Ceiling

Eine Decken- oder Bodenplatte. Damit kann ein Raum nach oben oder unten abgeschlossen werden, der weder durch ein Dach (siehe Roof), noch durch andere Räume abgeschlossen ist.

Ceiling Übersicht

Eigenschaften

EdgeTexture	Textur der Seitenflächen
History	Daten zur Identifikation im ehemaligen Projekt für nachgeladene Gebäude
ID	Liefert eine eindeutige Kennung
LowerSurfaceTexture	Textur der Unterseite
Openings	Liste der Öffnungen
Polygon	Umrandung der Deckenplatte
Story	Stockwerk, zu dem diese Decke gehört
UpperSurfaceTexture	Textur der Oberseite

Methoden

Delete	Deckenplatte löschen

Ceiling.EdgeTexture

Deklaration:

Texture

HRESULT get_EdgeTexture(ITexture** retVal);
HRESULT put_EdgeTexture(ITexture* newVal);

Siehe auch: Ceiling Übersicht, Texture

Die Kantentextur. Kanten von Deckenplatten sind z.B. an Öffnungen für Treppen sichtbar.

Ceiling.History

Deklaration:

IDHistory (nur lesbar)

HRESULT get_History(IIDHistory** retVal);

Siehe auch: Ceiling Übersicht, IDHistory

Falls dieses Ceiling beim Ursprünglichen Laden (oder Erstellen) des aktuellen Projektes nicht in diesem Projekt war, sonder aus einem anderen Projekt nachgeladen wurde, liefert diese Eigenschaft das History-Objekt, das die Umsetzung der IDs beschreibt. Andernfalls ist diese Eigenschaft “ Nothing” .

Ein Anwendungsbeispiel für History-Objekte finden Sie im Kapitel Eindeutige Kennungen.

Ceiling.ID

Deklaration:

long (nur lesbar)

HRESULT get_ID(long* retVal);

Siehe auch: Ceiling Übersicht

Gibt eine eindeutige Kennung dieses Ceiling an. Diese Kennung ist persistent, das heißt, sie bleibt innerhalb des aktuellen Projektes erhalten. Sie können diese Kennung benutzen, um Ceilings miteinander zu vergleichen oder zusätzliche Daten zum Ceiling in einer Datenbank abzulegen.

Die ID ist aber nur innerhalb des aktuellen Projektes eindeutig, wenn Sie (durch Nachladen einzelner Gebäude) das Objekt in ein anderes Projekt einfügen, erhält es eine neue ID. In diesem Fall müssen Sie sofort bei Beendigung des Ladevorganges mit Hilfe der Eigenschaft History objPropertyDetailsCeilingHistory Ihre zusätzlichen Daten an die neuen Werte anpassen.

Ceiling.LowerSurfaceTexture

Deklaration:

Texture

HRESULT get_LowerSurfaceTexture(ITexture** retVal);
HRESULT put_LowerSurfaceTexture(ITexture* newVal);

Siehe auch: Ceiling Übersicht, Texture

Die Textur der Unterseite der Deckenplatte.

Ceiling.Openings

Deklaration:

CeilingOpeningsCollection (nur lesbar)

```
HRESULT get_Openings(ICeilingOpeningsCollection**
retVal);
```

Siehe auch: Ceiling Übersicht, CeilingOpeningsCollection

Ceiling.Polygon

Deklaration:

Polygon2D

HRESULT get_Polygon(IPolygon2D** retVal);
HRESULT put_Polygon(IPolygon2D* newVal);

Siehe auch: Ceiling Übersicht, Polygon2D

Das Polygon, das die Deckenplatte begrenzt.

Ceiling.Story

Deklaration:

Story (nur lesbar)

HRESULT get_Story(IStory** retVal);

Siehe auch: Ceiling Übersicht, Story

Ceiling.UpperSurfaceTexture

Deklaration:

Texture

HRESULT get_UpperSurfaceTexture(ITexture** retVal);
HRESULT put_UpperSurfaceTexture(ITexture* newVal);

Siehe auch: Ceiling Übersicht, Texture

Die Textur der oberen Seite der Deckenplatte.

Ceiling.Delete

Deklaration:

```
Sub Delete()
```

```
HRESULT Delete();
```

Siehe auch: Ceiling Übersicht

Löscht die Deckenplatte.

Terrain

Das Objekt Terrain beschreibt die Umgebung außerhalb der Gebäude in ArCon. Bestandteile des Terrains sind z.B. Wege und Rasenflächen, nicht aber Außenobjekte wie z.B. Bäume, Autos und Gartenmöbel.

ArCon betrachtet ein Gelände als eine Fläche (angegeben durch ein Polygon) und einzelne Höhenpunkte innerhalb dieser Fläche. Die tatsächlichen 3D Daten entstehen aus einer Zerlegung der Fläche an den Höhenpunkten in einzelne Facetten. Je mehr Höhenpunkte vorhanden sind und je kleiner die Facettengröße gewählt wird, desto genauer wird die Darstellung. Da Gelände um Gebäude üblicherweise flach ist, gelingt bereits mit recht grober Facettierung eine realistische Darstellung.

Neben den 3D Daten wird für die Darstellung auch noch ein Material definiert.

Terrain Übersicht

Eigenschaften

AmbientCoefficient	Faktor für die Reflektion ambienten Lichtes
Area	Größe des Geländes
AreaFormula	Formel für Geländegröße
Buildings	Liste der Gebäude auf dem Gelände
DiffuseCoefficient	Faktor für die diffuse Reflektion
DiffuseColor	Farbe für diffus reflektierendes Licht
FacetteLength	Länge einer Facette
Flags	Verschiedene Flags, z.B. doppelseitig (ACMATFL_TWOSIDED)
Hedge	Ein (optionales) Heckenobjekt zur Umrandung des Terrains
Height	Default-Höhe beim Erzeugen
HighlightExponent	Phong'scher Exponent
History	Daten zur Identifikation im ehemaligen Projekt für nachgeladene Gebäude
ID	Eine eindeutige Kennung
Name	Name
Owner	übergeordnetes Gelände
Remark	Bemerkung
SpecularCoefficient	Faktor für gerichtete Reflektion
SpecularColor	Farbe für diffus reflektierendes Licht
Terrains	Liste der Untergelände
Texture	Verwendete Textur
Transparency	Faktor für die Lichtdurchlässigkeit
Transparent	Ist das Material des Terrains durchsichtig?
Type	Geländetyp (Grundstück, Geländebereich)
Umfang	Umfang des Geländes
UmfangsFormel	Formel für Umfang

Methoden

Delete	Gelände löschen
Edit	Dialogbox zum Bearbeiten des Geländes öffnen
GetBorderStyle	Liefert Stiftart- und Farbe der Darstellung der Terraingrenze
GetPolygon	Liefert die Koordinaten der Umrandung dieses Terrains
GetTransformation	Transformation für Texturkoordinaten lesen
SetBorderStyle	Ändert die Darstellung der Terraingrenze
SetPolygon	Ändert die Form und Position dieses Terrains
SetTransformation	Transformation für Texturkoordinaten setzen

Terrain.AmbientCoefficient

Deklaration:

Single

HRESULT get_AmbientCoefficient(float* retVal);
HRESULT put_AmbientCoefficient(float newVal);

Siehe auch: Terrain Übersicht

Dieser Wert gibt an, wieviel Prozent des einfallenden (diffusen) Umgebungslichtes vom Material wieder abgestrahlt wird. Im Gegensatz zu DiffuseCoefficient und SpecularCoefficient gibt AmbientCoefficient die Grundhelligkeit des Materials an.

Die Zahl muß im Bereich zwischen 0% (dunkel) und 100% (hell) liegen.

Terrain.Area

Deklaration:

Single (nur lesbar)

HRESULT get_Area(float* retVal);

Siehe auch: Terrain Übersicht

Die Fläche des Geländes in Quadratmetern.

Terrain.AreaFormula

Deklaration:

String (nur lesbar)

HRESULT get_AreaFormula(BSTR* retVal);

Siehe auch: Terrain Übersicht

Eine symbolische Formel zur Berechnung der Geländefläche.

Terrain.Buildings

Deklaration:

BuildingCollection (nur lesbar)

HRESULT get_Buildings(IBuildingCollection** retVal);

Siehe auch: Terrain Übersicht, BuildingCollection

Eine Liste mit Gebäuden auf diesem Gelände.

Terrain.DiffuseCoefficient

Deklaration:

Single

HRESULT get_DiffuseCoefficient(float* retVal);
HRESULT put_DiffuseCoefficient(float newVal);

Siehe auch: Terrain Übersicht

Dieser Wert gibt an, wieviel Prozent des einfallenden gerichteten Lichtes vom Material diffus (in verschiedene Richtungen) gespiegelt wird. Im Gegensatz zu AmbientCoefficient und SpecularCoefficient gibt AmbientCoefficient die Helligkeit des Materials an.

Die Zahl muß im Bereich zwischen 0% (dunkel) und 100% (hell) liegen.

Siehe auch: DiffuseColor

Terrain.DiffuseColor

Deklaration:

long

HRESULT get_DiffuseColor(long* retVal);
HRESULT put_DiffuseColor(long newVal);

Siehe auch: Terrain Übersicht

Der RGB-Wert der Farbe, mit der das Material Licht diffus reflektiert. Die Helligkeit dieser Reflektion wird mit DiffuseCoefficient angegeben.

Terrain.FacetteLength

Deklaration:

Single

HRESULT get_FacetteLength(float* retVal);
HRESULT put_FacetteLength(float newVal);

Siehe auch: Terrain Übersicht

Die maximale Länge einer Facette in Metern.

Terrain.Flags

Deklaration:

long

HRESULT get_Flags(long* retVal);
HRESULT put_Flags(long newVal);

Siehe auch: Terrain Übersicht

Die einzelnen Bits dieses Wertes geben diverse Materialeigenschaften des Terrains an. Der Gesamtwert errechnet sich durch Addition der einzelnen Werte. Die möglichen Konstanten finden Sie im Kapitel Material Flags.

Terrain.Hedge

Deklaration:

Hedge (nur lesbar)

HRESULT get_Hedge(IHedge** retVal);

Siehe auch: Terrain Übersicht, Hedge

Falls das Terrain durch eine Hecke begrenzt wird, enthält diese Eigenschaft das Heckenobjekt.

Terrain.Height

Deklaration:

Single

HRESULT get_Height(float* retVal);
HRESULT put_Height(float newVal);

Siehe auch: Terrain Übersicht

Die absolute Höhe der Terrain-Unterseite im Weltkoordinatensystem. Alle Höhenangaben der Höhenpunkte sind relativ zu diesem Niveau.

Terrain.HighlightExponent

Deklaration:

Single

HRESULT get_HighlightExponent(float* retVal);
HRESULT put_HighlightExponent(float newVal);

Siehe auch: Terrain Übersicht

Der “ Phong’ sche Exponent” des Materials. Er gibt (in etwa) an, wie die Helligkeit des Materials mit zunehmender Entfernung von der Lichtquelle abnimmt.

Terrain.History

Deklaration:

IDHistory (nur lesbar)

HRESULT get_History(IIDHistory** retVal);

Siehe auch: Terrain Übersicht, IDHistory

Falls dieses Terrain beim Ursprünglichen Laden (oder Erstellen) des aktuellen Projektes nicht in diesem Projekt war, sonder aus einem anderen Projekt nachgeladen wurde, liefert diese Eigenschaft das History-Objekt, das die Umsetzung der IDs beschreibt. Andernfalls ist diese Eigenschaft “ Nothing” .

Ein Anwendungsbeispiel für History-Objekte finden Sie im Kapitel Eindeutige Kennungen.

Terrain.ID

Deklaration:

long (nur lesbar)

HRESULT get_ID(long* retVal);

Siehe auch: Terrain Übersicht

Gibt eine eindeutige Kennung dieses Terrain an. Diese Kennung ist persistent, das heißt, sie bleibt innerhalb des aktuellen Projektes erhalten. Sie können diese Kennung benutzen, um Terrains miteinander zu vergleichen oder zusätzliche Daten zum Terrain in einer Datenbank abzulegen.

Die ID ist aber nur innerhalb des aktuellen Projektes eindeutig, wenn Sie (durch Nachladen einzelner Gebäude) das Objekt in ein anderes Projekt einfügen, erhält es eine neue ID. In diesem Fall müssen Sie sofort bei Beendigung des Ladevorganges mit Hilfe der Eigenschaft History objPropertyDetailsTerrainHistory Ihre zusätzlichen Daten an die neuen Werte anpassen.

Terrain.Name

Deklaration:

String

HRESULT get_Name(BSTR* retVal);
HRESULT put_Name(BSTR newVal);

Siehe auch: Terrain Übersicht

Der Name des Terrains.

Terrain.Owner

Deklaration:

Object (nur lesbar)

HRESULT get_Owner(IDispatch** retVal);

Siehe auch: Terrain Übersicht

Bei untergeordneten Geländen ein Verweis auf das nächste übergeordnete Gelände. Beim globalen Gelände “ Nothing” .

Terrain.Remark

Deklaration:

String

HRESULT get_Remark(BSTR* retVal);
HRESULT put_Remark(BSTR newVal);

Siehe auch: Terrain Übersicht

Eine beliebige Bemerkung zum Gelände.

Terrain.SpecularCoefficient

Deklaration:

Single

HRESULT get_SpecularCoefficient(float* retVal);
HRESULT put_SpecularCoefficient(float newVal);

Siehe auch: Terrain Übersicht

Dieser Wert gibt an, wieviel Prozent des einfallenden gerichteten Lichtes vom Material gespiegelt wird. Im Gegensatz zu DiffuseCoefficient und AmbientCoefficient gibt SpecularCoefficient den Glanz des Materials an.

Die Zahl muß im Bereich zwischen 0% (dunkel) und 100% (hell) liegen.

Terrain.SpecularColor

Deklaration:

long

HRESULT get_SpecularColor(long* retVal);
HRESULT put_SpecularColor(long newVal);

Siehe auch: Terrain Übersicht

Die Farbe, mit der gerichtetes Licht reflektiert wird. Die Stärke der Reflektion wird durch SpecularCoefficient bestimmt, eine eventuelle Mischung mit der Texturfarbe durch Flags.

Terrain.Terrains

Deklaration:

TerrainCollection (nur lesbar)

HRESULT get_Terrains(ITerrainCollection** retVal);

Siehe auch: Terrain Übersicht, TerrainCollection

Eine Liste von untergeordneten Geländen.

Terrain.Texture

Deklaration:

Texture (nur lesbar)

HRESULT get_Texture(ITexture** retVal);

Siehe auch: Terrain Übersicht, Texture

Falls das Terrain-Material nicht nur eine Farbe, sondern auch eine Oberflächenstruktur hat, enthält diese Eigenschaft das Textur-Objekt.

Terrain.Transparency

Deklaration:

Single

HRESULT get_Transparency(float* retVal);
HRESULT put_Transparency(float newVal);

Siehe auch: Terrain Übersicht

Falls das Material durchsichtig ist (durch Transparent bestimmt), gibt dieser Wert zwischen 0 und 100% an, wieviel Licht durch das Material hindurchscheint.

Terrain.Transparent

Deklaration:

Boolean

HRESULT get_Transparent(VARIANT_BOOL* retVal);
HRESULT put_Transparent(VARIANT_BOOL newVal);

Siehe auch: Terrain Übersicht

Gibt an, ob das Material des Terrains durchsichtig ist. Falls ja, wird mit Transparency das Ausmaß der Durchsichtigkeit bestimmt.

Terrain.Type

Deklaration:

long

HRESULT get_Type(long* retVal);
HRESULT put_Type(long newVal);

Siehe auch: Terrain Übersicht

Die Art des Geländes. Mögliche Werte finden Sie im Kapitel Terrain Types.

Terrain.Umfang

Deklaration:

Single (nur lesbar)

HRESULT get_Umfang(float* retVal);

Siehe auch: Terrain Übersicht

Der Umfang des Geländes.

Terrain.UmfangsFormel

Deklaration:

String (nur lesbar)

HRESULT get_UmfangsFormel(BSTR* retVal);

Siehe auch: Terrain Übersicht

Eine symbolische Formel zur Berechnung des Umfangs.

Terrain.Delete

Deklaration:

Sub Delete()

HRESULT Delete();

Siehe auch: Terrain Übersicht

Mit dieser Methode wird das Terrain im ArCon Projekt gelöscht.

Dies ist etwas anderes, als das Terrain-Objekt zu löschen. In Visual Basic können Sie z.B. schreiben:

```
Dim terra As Object
.....
Set terra = Nothing
```

Durch die letzte Zeile verlieren Sie den Zugriff auf das Terrain-Objekt und können im folgenden keine Eigenschaften dieses Geländes mehr verändern. Dabei wird aber im aktuellen ArCon Objekt keine Änderung vorgenommen. Hingegen würde die Zeile

```
terra.Delete
```

dazu führen, daß ArCon das Terrain entfernt, während Sie weiterhin Zugriff auf das Terrain-Objekt haben.

Terrain.Edit

Deklaration:

Function Edit() as Boolean

HRESULT Edit(VARIANT_BOOL* retVal);

Siehe auch: Terrain Übersicht

Ruft die Dialogbox zur Terrain-Eingabe auf.

Terrain.GetBorderStyle

Deklaration:

Sub GetBorderStyle(ByRef penColor as long)

HRESULT GetBorderStyle(long *penStyle, long * penColor);

Siehe auch: Terrain Übersicht

Terrain.GetPolygon

Deklaration:

```
Function GetPolygon() as Point2DCollection
```

```
HRESULT GetPolygon(IPoint2DCollection** retVal);
```

Siehe auch: Terrain Übersicht, Point2DCollection

Terrain.GetTransformation

Deklaration:

```
Sub GetTransformation(ByRef x1 as Single, ByRef y1 as
Single, ByRef z1 as Single, ByRef x2 as Single, ByRef y2 as
Single, ByRef z2 as Single, ByRef x3 as Single, ByRef y3 as
Single, ByRef z3 as Single)
```

```
HRESULT GetTransformation(float* x1, float* y1, float* z1,
float* x2, float* y2, float* z2, float* x3, float* y3, float* z3);
```

Siehe auch: Terrain Übersicht

Liefert die Transformationsmatrix. Wenn Sie sich mit Transformationsmatritzen auskennen, können Sie durch diese Matrix das Terrain drehen, verschieben und vergrößern oder verkleinern.

Zum Ändern der Matrix rufen Sie SetTransformation auf.

Terrain.SetBorderStyle

Deklaration:

```
Sub SetBorderStyle(ByVal penStyle as long, ByVal penColor
as long)
```

```
HRESULT SetBorderStyle(long penStyle, long penColor);
```

Siehe auch: Terrain Übersicht

Terrain.SetPolygon

Deklaration:

```
Function SetPolygon() as Boolean
```

```
HRESULT SetPolygon(IPoint2DCollection *newPoly,
VARIANT_BOOL* retVal);
```

Siehe auch: Terrain Übersicht

Terrain.SetTransformation

Deklaration:

```
Sub SetTransformation(ByVal x1 as Single, ByVal y1 as
Single, ByVal z1 as Single, ByVal x2 as Single, ByVal y2 as
Single, ByVal z2 as Single, ByVal x3 as Single, ByVal y3 as
Single, ByVal z3 as Single)
```

```
HRESULT SetTransformation(float x1, float y1, float z1, float
x2, float y2, float z2, float x3, float y3, float z3);
```

Siehe auch: Terrain Übersicht

Ändert die Transformationsmatrix des Geländes. Die aktuelle Matrix können Sie mit GetTransformation ermitteln.

Hedge

Ein Umrandungsobjekt, das ein Gelände abgrenzen kann und z.B. für die Darstellung von Hecken benutzt wird.

Hedge Übersicht
Eigenschaften

Continuous	Umrandungsobjekte aneinandergereiht?
Distance	Abstand zwischen zwei Umrandungsobjekten
FileName	Dateiname der Umrandungsobjekte
ScaleToFit	Größe so anpassen, daß Länge mit Grundstückskante übereinstimmt?

Hedge.Continuous

Deklaration:

Boolean

HRESULT get_Continuous(VARIANT_BOOL* retVal);
HRESULT put_Continuous(VARIANT_BOOL newVal);

Siehe auch: Hedge Übersicht

Wenn dieses Attribut “ Wahr” ist, wird das Objekt am Rand des zugehörigen Geländes wiederholt.

Hedge.Distance

Deklaration:

Single

HRESULT get_Distance(float* retVal);
HRESULT put_Distance(float newVal);

Siehe auch: Hedge Übersicht

Wenn das Objekt entlang des Randes wiederholt wird, gibt dieser Wert den Abstand zwischen zwei Objekten an.

Hedge.FileName

Deklaration:

String

HRESULT get_FileName(BSTR* retVal);
HRESULT put_FileName(BSTR newVal);

Siehe auch: Hedge Übersicht

Dateiname des Objektes.

Hedge.ScaleToFit

Deklaration:

Boolean

HRESULT get_ScaleToFit(VARIANT_BOOL* retVal);
HRESULT put_ScaleToFit(VARIANT_BOOL newVal);

Siehe auch: Hedge Übersicht

Wenn dieses Attribut “ Wahr” ist, wird das Objekt skaliert, sodaß es genau die passende Größe für die jeweilige Seitenlänge des Geländes hat.

Building

Dieses Objekt repräsentiert ein ArCon Gebäude. Es enthält im wesentlichen eine Liste aller Stockwerke dieses Gebäudes sowie einige statistische Daten. Ein Gebäude kann neue Stockwerke anlegen.

Building Übersicht

Eigenschaften

Area	Flächeninhalt
History	Daten zur Identifikation im ehemaligen Projekt für nachgeladene Gebäude
ID	Liefert eine eindeutige Kennung
Name	Name
Remark	Bemerkung
Rotation	Rotation des Gebäudes
Stories	Liste der Geschosse des Gebäudes
Volume	Brutto-Rauminhalt

Methoden

CreateStory	Geschoß ober/unterhalb erzeugen
Delete	Gebäude löschen
GetHeight	Liefert die Gebäudehöhe an einem Testpunkt
Mirror	Spiegelt das Gebäude um eine in Punkt/Richtungsform angegebene Achse
Move	Verschiebt das Gebäude
Objects	Liefert ein Array mit allen 3D Objekten in diesem Gebäude
Rotate	Rotiert das Gebäude um den angegebenen Drehpunkt (Winkel in Bogenmaß)

Building.Area

Deklaration:

Single (nur lesbar)

HRESULT get_Area(float* retVal);

Siehe auch: Building Übersicht

Die gesamte Nutzfläche des Gebäudes in Quadratmetern.

Building.History

Deklaration:

IDHistory (nur lesbar)

HRESULT get_History(IIDHistory** retVal);

Siehe auch: Building Übersicht, IDHistory

Falls dieses Building beim Ursprünglichen Laden (oder Erstellen) des aktuellen Projektes nicht in diesem Projekt war, sonder aus einem anderen Projekt nachgeladen wurde, liefert diese Eigenschaft das History-Objekt, das die Umsetzung der IDs beschreibt. Andernfalls ist diese Eigenschaft “ Nothing” .

Ein Anwendungsbeispiel für History-Objekte finden Sie im Kapitel Eindeutige Kennungen.

Building.ID

Deklaration:

long (nur lesbar)

HRESULT get_ID(long* retVal);

Siehe auch: Building Übersicht

Gibt eine eindeutige Kennung dieses Building an. Diese Kennung ist persistent, das heißt, sie bleibt innerhalb des aktuellen Projektes erhalten. Sie können diese Kennung benutzen, um Buildings miteinander zu vergleichen oder zusätzliche Daten zum Building in einer Datenbank abzulegen.

Die ID ist aber nur innerhalb des aktuellen Projektes eindeutig, wenn Sie (durch Nachladen einzelner Gebäude) das Objekt in ein anderes Projekt einfügen, erhält es eine neue ID. In diesem Fall müssen Sie sofort bei Beendigung des Ladevorganges mit Hilfe der Eigenschaft History objPropertyDetailsBuildingHistory Ihre zusätzlichen Daten an die neuen Werte anpassen.

Building.Name

Deklaration:

String

HRESULT get_Name(BSTR* retVal);
HRESULT put_Name(BSTR newVal);

Siehe auch: Building Übersicht

Die Bezeichnung des Gebäudes.

Building.Remark

Deklaration:

String

HRESULT get_Remark(BSTR* retVal);
HRESULT put_Remark(BSTR newVal);

Siehe auch: Building Übersicht

Eine beliebige Anmerkung zum Gebäude.

Building.Rotation

Deklaration:

Single

HRESULT get_Rotation(float* retVal);
HRESULT put_Rotation(float newVal);

Siehe auch: Building Übersicht

Die Drehung des Gebäudes gegenüber dem Grundstück in Bogenmaß. In der Regel 0.

Building.Stories

Deklaration:

StoryCollection (nur lesbar)

HRESULT get_Stories(IStoryCollection** retVal);

Siehe auch: Building Übersicht, StoryCollection

Die Liste mit allen Stockwerken des Gebäudes.

Building.Volume

Deklaration:

Single (nur lesbar)

HRESULT get_Volume(float* retVal);

Siehe auch: Building Übersicht

Das Gesamtvolumen des Gebäudes in Kubikmetern.

Building.CreateStory

Deklaration:

```
Function CreateStory(ByVal aboveCurrent as Boolean,
ByVal floorType as short) as Story
```

```
HRESULT CreateStory(VARIANT_BOOL aboveCurrent,
short floorType, IStory** retVal);
```

Siehe auch: Building Übersicht, Story

Erzeugt ein neues Stockwerk oberhalb aller bisherigen oder unterhalb aller bereits vorhandenen Stockwerke. Sie können nicht nachträglich ein Stockwerk zwischen anderen einfügen.

Mögliche Werte für den Parameter “ Typ” finden Sie im Kapitel Story Types. Dieser Parameter legt nur die Bezeichnung des Stockwerkes fest, technisch ist es möglich, ein Kellergeschoß oberhalb eines Spitzbodens anzulegen - aber natürlich nicht sinnvoll.

Das neu angelegte Geschoß wird automatisch zum aktuellen Stockwerk.

Die Parameter sind:

aboveCurrent	“ Wahr” wenn ein Stockwerk oberhalb angelegt werden soll, “ Falsch” wenn es unterhalb des jetzigen Gebäudes entsteht.
floorType	Typ des Stockwerkes.

Building.Delete

Deklaration:

Sub Delete()

HRESULT Delete();

Siehe auch: Building Übersicht

Löscht das gesamte Gebäude.

Building.GetHeight

Deklaration:

```
Function GetHeight(ByVal addEindeckung as Boolean,
ByVal x as Single, ByVal y as Single, ByRef foundRoof as
Roof, ByRef foundZ as Single) as Boolean
```

```
HRESULT GetHeight(VARIANT_BOOL addEindeckung,
float x, float y, IRoof** foundRoof, float * foundZ, float *
dirX_x, float *dirX_y, float *dirX_z, float *dirY_x, float *
dirY_y, float *dirY_z, VARIANT_BOOL* retVal);
```

Siehe auch: Building Übersicht, Roof

Building.Mirror

Deklaration:

```
Function Mirror(ByVal x0 as Single, ByVal y0 as Single,
ByVal dx as Single, ByVal dy as Single, ByVal canUndo as
Boolean) as Boolean
```

```
HRESULT Mirror(float x0, float y0, float dx, float dy,
VARIANT_BOOL canUndo, VARIANT_BOOL* retVal);
```

Siehe auch: Building Übersicht

Building.Move

Deklaration:

```
Function Move(ByVal dx as Single, ByVal dy as Single,
ByVal canUndo as Boolean) as Boolean
```

```
HRESULT Move(float dx, float dy, VARIANT_BOOL
canUndo, VARIANT_BOOL* retVal);
```

Siehe auch: Building Übersicht

Building.Objects

Deklaration:

```
Function Objects() as VARIANT
```

```
HRESULT Objects(VARIANT* retVal);
```

Siehe auch: Building Übersicht

Building.Rotate

Deklaration:

```
Function Rotate(ByVal PivotX as Single, ByVal PivotY as
Single, ByVal angle as Single, ByVal canUndo as Boolean)
as Boolean
```

```
HRESULT Rotate(float PivotX, float PivotY, float angle,
VARIANT_BOOL canUndo, VARIANT_BOOL* retVal);
```

Siehe auch: Building Übersicht

Story

Ein Stockwerk.

Dies ist eines der zentralen Objekte der Objekthierachie in ArCon. Wenn Ihr Programm eine Planung untersucht, finden Sie hier Listen mit eindeutigen Zuordnungen für fast alle konstruktiven Elemente. Darüberhinaus ist das Stockwerk zuständig für die Plazierung der meisten konstruktiven Elemente.

Stockwerke in ArCon beinhalten immer die sie abdeckende Decke und deren Fußboden – ein Stockwerk geht also von der Oberkannte des Fertigfußbodens bis zur Oberkannte des Fertigfußbodens im darüberliegenden Stockwerk.

Die folgende Skizze zeigt die unterschiedlichen Höhenangaben:

Dabei bedeuten:

Kürzel	Bedeutung	Berechnung
Hg	Geschoßhöhe	LichteHoehe + Unterbau + Rohdecke + Fussboden
Hw	Wandhöhe	LichteHoehe + Unterbau + DistanceStoryBelow
Hr	Raumhöhe	LichteHoehe + Unterbau
Hl	lichte Höhe	LichteHoehe
OKFF	Oberkannte Fertigfußboden	BaseHeight (in der Skizze die untere OKFF)
OKRD	Oberkannte Rohdecke	BaseHeight + LichteHoehe + Unterbau + Rohdecke + Fussboden (in der Skizze die obere OKRD)
Abh	Abhängung/Unterbau	Unterbau
RD	Rohdecke	Rohdecke
FB	Fußboden	Fussboden
FU	Fußboden des Stockwerkes unterhalb	DistanceStoryBelow

Das ArCon Stockwerk geht vom unteren OKFF zum oberen OKFF

Story Übersicht
Eigenschaften

BaseHeight	Basishöhe
BruttoGeschossflaeche	Brutto-Geschoßfläche
BruttoGeschossflaechenFormel	Formel für Brutto-Geschoßfläche
BruttoRauminhalt	Brutto-Rauminhalt
BruttoRauminhaltsFormel	Formel für Brutto-Rauminhalt
Building	Gebäude, zu dem das Geschoß gehört
CeilingOpenings	Liste der Deckenaussparungen des Geschosses
Ceilings	Liste der Deckenplatten des Geschosses
Chimneys	Liste der Schornsteine des Geschosses
Conturs	Liste der Wandseiten des Geschosses
DistanceStoryBelow	Dicke des Fußbodens
Fussboden	Dicke des Fußbodens des nächsten Geschosses
Gauben	Liste der Gauben des Stockwerks
Graphics2D	Liste der 2D Graphikelemente in diesem Stockwerk
History	Daten zur Identifikation im ehemaligen Projekt für nachgeladene Gebäude
ID	Eine eindeutige Kennung
LichteHoehe	Lichte Höhe
Name	Name
PlasterName	Außenputzbezeichnung
PlasterThickness	Außenputzstärke
Remark	Bemerkung
Rohdecke	Dicke der Rohdecke
Roofs	Liste der Dächer des Geschosses
Rooms	Liste der Räume des Geschosses
Stairs	Liste der Treppen des Geschosses
Supports	Liste der Stützen des Geschosses
UnterUeberzuege	Liste der Unter- und Überzüge des Geschosses
Unterbau	Dicke des Deckenunterbaus
VirtualWalls	Liste der virtuellen Wände des Stockwerks
Walls	Liste der Wände des Geschosses

Methoden

ChangeType	Ändert die Art des Stockwerks
Delete	Löscht das Stockwerk. Wenn das letzte Stockwerk eines Gebäudes gelöscht wird, wird damit auch das Gebäude gelöscht.
FindChimney	Liefert einen Schornstein an der angegebenen Position
FindRoom	Liefert Raum an Position x/y
FindSupport	Liefert eine Stütze an der angegebenen Position
FindUnterUeberzug	Liefert einen Unter- oder Überzug an der angegebenen

	Position
FindWall	Liefert Wand an Position x/y
GetType	Ermittelt die Art des Stockwerks
Label	Beschriftung einfügen
Objects	Liefert ein Array mit allen 3D Objekten in diesem Stockwerk
PlaceCeiling	Deckenplatte einfügen
PlaceCeilingOpening	Deckenaussparung einfügen
PlaceChimney	Schornstein einfügen
PlaceDimension	Vermaßung einfügen
PlaceGuide	Hilfslinie einfügen
PlaceRoof	Dach aufsetzen
PlaceRoofAutomatic	Dach mit automatischer Konturfindung aufsetzen
PlaceStairCase	Treppe einfügen
PlaceSupport	Stütze einfügen
PlaceUnterUeberzug	Unter- oder Überzug einfügen
PlaceVirtualWall	Virtuelle Wand generieren
PlaceWall	Wand generieren

Story.BaseHeight

Deklaration:

Single

HRESULT get_BaseHeight(float* retVal);
HRESULT put_BaseHeight(float newVal);

Siehe auch: Story Übersicht

Die Basishöhe des Geschosses (der Abstand zwischen der Unterseite des Geschoßes zum Null-Niveau des Gebäudes).

Im allgemeinen ergibt sich dieser Wert automatisch aus der Summe der Geschoßhöhen der unter diesem Stockwerk liegenden Geschoße. Eine Zuweisung an diese Eigenschaft wird dann von ArCon ignoriert.

Nur im untersten Stockwerk können Sie durch eine Zuweisung an diese Eigenschaft die Oberkannte des Fertigfußbodens auf ein beliebiges Niveau verschieben.

Eine Skizze der Höhenangaben zu Stockwerken finden Sie unter Stockwerk.

Story.BruttoGeschossflaeche

Deklaration:

Single (nur lesbar)

HRESULT get_BruttoGeschossflaeche(float* retVal);

Siehe auch: Story Übersicht

Die Brutto-Geschoßfläche des Stockwerks in Quadratmetern.

Story.BruttoGeschossflaechenFormel

Deklaration:

String (nur lesbar)

HRESULT get_BruttoGeschossflaechenFormel(BSTR* retVal);

Siehe auch: Story Übersicht

Eine symbolische Formel für die Brutto-Geschoßfläche.

Story.BruttoRauminhalt

Deklaration:

Single (nur lesbar)

HRESULT get_BruttoRauminhalt(float* retVal);

Siehe auch: Story Übersicht

Der Brutto-Rauminhalt des Stockwerks in Kubikmetern.

Story.BruttoRauminhaltsFormel

Deklaration:

String (nur lesbar)

HRESULT get_BruttoRauminhaltsFormel(BSTR* retVal);

Siehe auch: Story Übersicht

Eine symbolische Formel für den Brutto-Rauminhalt.

Story.Building

Deklaration:

Building (nur lesbar)

HRESULT get_Building(IBuilding** retVal);

Siehe auch: Story Übersicht, Building

Das Gebäude, zu dem das Stockwerk gehört.

Story.CeilingOpenings

Deklaration:

CeilingOpeningsCollection (nur lesbar)

```
HRESULT get_CeilingOpenings
(ICeilingOpeningsCollection** retVal);
```

Siehe auch: Story Übersicht, CeilingOpeningsCollection

Eine Liste aller Deckenaussparungen in diesem Stockwerk.

Story.Ceilings

Deklaration:

CeilingCollection (nur lesbar)

HRESULT get_Ceilings(ICeilingCollection** retVal);

Siehe auch: Story Übersicht, CeilingCollection

Eine Liste aller Deckenplatten des Stockwerks.

Story.Chimneys

Deklaration:

ChimneyCollection (nur lesbar)

HRESULT get_Chimneys(IChimneyCollection** retVal);

Siehe auch: Story Übersicht, ChimneyCollection

Eine Liste aller Schornsteine im Stockwerk.

Story.Conturs

Deklaration:

ConturCollection (nur lesbar)

HRESULT get_Conturs(IConturCollection** retVal);

Siehe auch: Story Übersicht, ConturCollection

Alle Konturen (Listen von zusammenhängenden Wandseiten) des Stockwerks.

Story.DistanceStoryBelow

Deklaration:

Single

HRESULT get_DistanceStoryBelow(float* retVal);
HRESULT put_DistanceStoryBelow(float newVal);

Siehe auch: Story Übersicht

Der Name dieses Attributes ist irreführend (kann aber zur Erhaltung der Binärkompatibilität mit existierenden Makros derzeit nicht geändert werden).

Dieser Wert beschreibt den Abstand zwischen diesem Stockwerk und der Oberkannte der Rohdecke des darunterliegenden Stockwerkes – oder in anderen Worten: die Dicke des Fußbodens unterhalb dieses Geschoßes (beachten Sie: in ArCon gehört die Decke eines Raumes und der zugehörige Fußboden zum Stockwerk!).

Wenn dieses Stockwerk das unterste ist, ist dies die einzige Möglichkeit, die Dicke des Fußbodens zu ermitteln bzw. zu ändern. Andernfalls ist der Wert identisch mit dem Attribut Fussboden des darunterliegenden Stockwerkes.

Eine Skizze der Höhenangaben zu Stockwerken finden Sie unter Stockwerk.

Story.Fussboden

Deklaration:

Single

HRESULT get_Fussboden(float* retVal);
HRESULT put_Fussboden(float newVal);

Siehe auch: Story Übersicht

Die Dicke des Fußbodens auf der oberhalb des Stockwerks liegenden Decke.

Eine Skizze der Höhenangaben zu Stockwerken finden Sie unter Stockwerk.

Story.Gauben

Deklaration:

GaubenCollection (nur lesbar)

HRESULT get_Gauben(IGaubenCollection** retVal);

Siehe auch: Story Übersicht, GaubenCollection

Story.Graphics2D

Deklaration:

Graphics2DCollection (nur lesbar)

HRESULT get_Graphics2D(IGraphics2DCollection** retVal);

Siehe auch: Story Übersicht, Graphics2DCollection

Die Liste von Grafikelemente, die spezifisch für dieses Stockwerk angezeigt werden sollen. Im Gegensatz zu den globalen Grafikelementen in ArCon.Graphics2D hängt die Sichtbarkeit der Elemente in dieser Liste von der Sichtbarkeit des zugehörigen Stockwerkes ab.

Story.History

Deklaration:

IDHistory (nur lesbar)

HRESULT get_History(IIDHistory** retVal);

Siehe auch: Story Übersicht, IDHistory

Falls dieses Story beim Ursprünglichen Laden (oder Erstellen) des aktuellen Projektes nicht in diesem Projekt war, sonder aus einem anderen Projekt nachgeladen wurde, liefert diese Eigenschaft das History-Objekt, das die Umsetzung der IDs beschreibt. Andernfalls ist diese Eigenschaft “ Nothing” .

Ein Anwendungsbeispiel für History-Objekte finden Sie im Kapitel Eindeutige Kennungen.

Story.ID

Deklaration:

long (nur lesbar)

HRESULT get_ID(long* retVal);

Siehe auch: Story Übersicht

Gibt eine eindeutige Kennung dieses Story an. Diese Kennung ist persistent, das heißt, sie bleibt innerhalb des aktuellen Projektes erhalten. Sie können diese Kennung benutzen, um Storys miteinander zu vergleichen oder zusätzliche Daten zum Story in einer Datenbank abzulegen.

Die ID ist aber nur innerhalb des aktuellen Projektes eindeutig, wenn Sie (durch Nachladen einzelner Gebäude) das Objekt in ein anderes Projekt einfügen, erhält es eine neue ID. In diesem Fall müssen Sie sofort bei Beendigung des Ladevorganges mit Hilfe der Eigenschaft History objPropertyDetailsStoryHistory Ihre zusätzlichen Daten an die neuen Werte anpassen.

Story.LichteHoehe

Deklaration:

Single

HRESULT get_LichteHoehe(float* retVal);
HRESULT put_LichteHoehe(float newVal);

Siehe auch: Story Übersicht

Die lichte Höhe des Geschosses.

Eine Skizze der Höhenangaben zu Stockwerken finden Sie unter Stockwerk.

Story.Name

Deklaration:

String

HRESULT get_Name(BSTR* retVal);
HRESULT put_Name(BSTR newVal);

Siehe auch: Story Übersicht

Der Name des Stockwerks.

Story.PlasterName

Deklaration:

String

HRESULT get_PlasterName(BSTR* retVal);
HRESULT put_PlasterName(BSTR newVal);

Siehe auch: Story Übersicht

Der Name des verwendeten Innen-Putzes.

Story.PlasterThickness

Deklaration:

Single

HRESULT get_PlasterThickness(float* retVal);
HRESULT put_PlasterThickness(float newVal);

Siehe auch: Story Übersicht

Die Dicke des Putzes.

Story.Remark

Deklaration:

String

HRESULT get_Remark(BSTR* retVal);
HRESULT put_Remark(BSTR newVal);

Siehe auch: Story Übersicht

Eine beliebige Bemerkung zum Stockwerk.

Story.Rohdecke

Deklaration:

Single

HRESULT get_Rohdecke(float* retVal);
HRESULT put_Rohdecke(float newVal);

Siehe auch: Story Übersicht

Die Dicke der Rohdecke.

Eine Skizze der Höhenangaben zu Stockwerken finden Sie unter Stockwerk.

Story.Roofs

Deklaration:

RoofCollection (nur lesbar)

HRESULT get_Roofs(IRoofCollection** retVal);

Siehe auch: Story Übersicht, RoofCollection

Eine Liste aller Dächer dieses Stockwerks.

Story.Rooms

Deklaration:

RoomCollection (nur lesbar)

HRESULT get_Rooms(IRoomCollection** retVal);

Siehe auch: Story Übersicht, RoomCollection

Eine Liste aller Räume des Stockwerks.

Story.Stairs

Deklaration:

StairCaseCollection (nur lesbar)

HRESULT get_Stairs(IStairCaseCollection** retVal);

Siehe auch: Story Übersicht, StairCaseCollection

Eine Liste aller Treppen im Stockwerk.

Story.Supports

Deklaration:

SupportCollection (nur lesbar)

HRESULT get_Supports(ISupportCollection** retVal);

Siehe auch: Story Übersicht, SupportCollection

Eine Liste aller Stützen des Stockwerks.

Story.UnterUeberzuege

Deklaration:

UnterUeberzugCollection (nur lesbar)

```
HRESULT get_UnterUeberzuege
(IUnterUeberzugCollection** retVal);
```

Siehe auch: Story Übersicht, UnterUeberzugCollection

Eine Liste aller Unter- und Überzüge in diesem Stockwerk.

Story.Unterbau

Deklaration:

Single

HRESULT get_Unterbau(float* retVal);
HRESULT put_Unterbau(float newVal);

Siehe auch: Story Übersicht

Die Dicke der Deckenabhängung.

Story.VirtualWalls

Deklaration:

VirtualWallCollection (nur lesbar)

HRESULT get_VirtualWalls(IVirtualWallCollection** retVal);

Siehe auch: Story Übersicht, VirtualWallCollection

Story.Walls

Deklaration:

WallCollection (nur lesbar)

HRESULT get_Walls(IWallCollection** retVal);

Siehe auch: Story Übersicht, WallCollection

Eine Liste aller Wände im Stockwerk.

Story.ChangeType

Deklaration:

```
Sub ChangeType(ByVal newStoryType as AC_Story_Types)
```

```
HRESULT ChangeType(AC_Story_Types newStoryType);
```

Siehe auch: Story Übersicht

Story.Delete

Deklaration:

```
Function Delete() as Boolean
```

```
HRESULT Delete(VARIANT_BOOL* retVal);
```

Siehe auch: Story Übersicht

Story.FindChimney

Deklaration:

```
Function FindChimney(ByVal x as Single, ByVal y as Single)
as Chimney
```

```
HRESULT FindChimney(float x, float y, IChimney** retVal);
```

Siehe auch: Story Übersicht, Chimney

Prüft, ob sich an der angegebenen Position ein Schornstein befindet und liefert diesen zurück. Im Fehlerfall wird Nothing (der Null-Zeiger) geliefert.

Parameter:

x, y	zu prüfende Position

Story.FindRoom

Deklaration:

```
Function FindRoom(ByVal x as Single, ByVal y
as Single) as Room
```

```
HRESULT FindRoom(float x, float y, IRoom**
retVal);
```

Siehe auch: Story Übersicht, Room

Liefert den Raum an der angegebenen Position, falls es dort einen Raum gibt. Andernfalls wird “ Nothing” zurückgeliefert.

Parameter:

x, y	zu prüfende Position

Story.FindSupport

Deklaration:

```
Function FindSupport(ByVal x as Single, ByVal
y as Single) as Support
```

```
HRESULT FindSupport(float x, float y, ISupport*
* retVal);
```

Siehe auch: Story Übersicht, Support

Prüft, ob sich an der angegebenen Position eine Stütze befindet und liefert diese zurück. Im Fehlerfall wird Nothing (der Null-Zeiger) geliefert.

Parameter sind

x, y	zu prüfende Position

Story.FindUnterUeberzug

Deklaration:

```
Function FindUnterUeberzug(ByVal x as Single,
ByVal y as Single) as UnterUeberzug
```

```
HRESULT FindUnterUeberzug(float x, float y,
IUnterUeberzug** retVal);
```

Siehe auch: Story Übersicht, UnterUeberzug

Prüft, ob sich an der angegebenen Position ein Unter-/Überzug befindet und liefert diesen zurück. Im Fehlerfall wird Nothing (der Null-Zeiger) geliefert.

Parameter sind

x, y	zu prüfende Position

Story.FindWall

Deklaration:

```
Function FindWall(ByVal x as Single, ByVal y as
Single) as Wall
```

```
HRESULT FindWall(float x, float y, IWall**
retVal);
```

Siehe auch: Story Übersicht, Wall

Findet eine Wand, falls die angegebenen Koordinaten in einer Wand liegen. Ansonsten wird “ Nothing” geliefert.

Parameter:

x, y	zu prüfende Position

Story.GetType

Deklaration:

```
Function GetType() as AC_Story_Types
```

```
HRESULT GetType(AC_Story_Types* retVal);
```

Siehe auch: Story Übersicht

Story.Label

Deklaration:

```
Function Label(ByVal aLabeling as Labeling,
ByVal x as Single, ByVal y as Single) as
Boolean
```

```
HRESULT Label(ILabeling* aLabeling, float x,
float y, VARIANT_BOOL* retVal);
```

Siehe auch: Story Übersicht, Labeling

Plaziert eine Beschriftung an der angegebenen Position. Die Beschriftung selbst wird mit Hilfe der Methode ArCon.NewLabeling erzeugt, kann dann parametrisiert werden und abschließend durch Plazierung in der Planung sichtbar gemacht werden.

Parameter:

aLabeling	das einzufügende Beschriftungsobjekt
x, y	die Position der Beschriftung

Story.Objects

Deklaration:

```
Function Objects() as VARIANT
```

```
HRESULT Objects(VARIANT* retVal);
```

Siehe auch: Story Übersicht

Story.PlaceCeiling

Deklaration:

```
Function PlaceCeiling(ByVal aCeiling as Ceiling,
ByVal aPolygon as Polygon2D) as Boolean
```

```
HRESULT PlaceCeiling(ICeiling* aCeiling,
IPolygon2D* aPolygon, VARIANT_BOOL*
retVal);
```

Siehe auch: Story Übersicht, Ceiling, Polygon2D

Plaziert eine Deckenplatte. Da eine Deckenplatte beliebige Formen haben kann, genügt es nicht, zur Plazierung Koordinaten anzugeben. Es wird daher das Polygon der Umrandung der Deckenplatte übergeben, sowie eine entsprechend parametrisierte Deckenplatte, die zuvor durch einen Aufruf von ArCon.NewCeiling erzeugt wurde.

Parameter:

aCeiling	das einzufügende Deckenplatten-Objekt
aPolygon	die Umrandung der Deckenplatte

Story.PlaceCeilingOpening

Deklaration:

```
Function PlaceCeilingOpening(ByVal opening
as CeilingOpening, ByVal aPolygon as
Polygon2D) as Boolean
```

```
HRESULT PlaceCeilingOpening
(ICeilingOpening* opening, IPolygon2D*
aPolygon, VARIANT_BOOL* retVal);
```

Siehe auch: Story Übersicht, CeilingOpening, Polygon2D

Erzeugt eine Deckenaussparung am angegebenen Polygon. Zuvor muß eine Deckenaussparung mit der Methode ArCon.NewCeilingOpening generiert und entsprechend parametrisiert werden.

Parameter:

opening	das einzufügende Deckenöffnungs-Objekt
aPolygon	die Umrandung des Deckenlochs

Story.PlaceChimney

Deklaration:

```
Function PlaceChimney(ByVal aChimney as
Chimney, ByVal x as Single, ByVal y as Single)
as Boolean
```

```
HRESULT PlaceChimney(IChimney* aChimney,
float x, float y, VARIANT_BOOL* retVal);
```

Siehe auch: Story Übersicht, Chimney

Plaziert einen Schornstein an der angegebenen Position. Der Schornstein wird zuvor durch den Aufruf von ArCon.NewChimney erzeugt.

Parameter:

aChimney	das einzufügende Schornstein-Objekt
x, y	die Position des Schornsteins

Story.PlaceDimension

Deklaration:

```
Function PlaceDimension(ByVal aDimension as
Dimension, ByVal x1 as Single, ByVal y1 as
Single, ByVal x2 as Single, ByVal y2 as Single)
as Boolean
```

```
HRESULT PlaceDimension(IDimension*
aDimension, float x1, float y1, float x2, float y2,
VARIANT_BOOL* retVal);
```

Siehe auch: Story Übersicht, Dimension

Plaziert eine Vermaßung in der Planung. Vermessen wird der Abstand zwischen den beiden angegebenen Punkten. Die Vermaßung selbst wird zuvor mit der Methode ArCon.NewDimension erzeugt und entsprechend parametrisiert.

Parameter:

aDimension	das einzufügende Bemaßungs-Objekt
x1, y1	Startpunkt der gemessenen Strecke
x2, y2	Endpunkt der gemessenen Strecke

Story.PlaceGuide

Deklaration:

```
Function PlaceGuide(ByVal aGuide as Guide,
ByVal x1 as Single, ByVal y1 as Single, ByVal
x2 as Single, ByVal y2 as Single) as Boolean
```

```
HRESULT PlaceGuide(IGuide* aGuide, float x1,
float y1, float x2, float y2, VARIANT_BOOL*
retVal);
```

Siehe auch: Story Übersicht, Guide

Plaziert eine Hilfslinie oder -Strecke. Die Hilfslinie selbst wird zuvor durch einen Aufruf von ArCon. NewGuide erzeugt.

Parameter:

aGuide	das einzufügende Hilfslinien-Objekt
x1, y1	Startpunkt der Hilfsstrecke bzw. ein Punkt auf der Hilfslinie
x2, y2	Endpunkt der Hilfsstrecke bzw. ein zweiter Punkt auf der Hilfslinie

Story.PlaceRoof

Deklaration:

```
Function PlaceRoof(ByVal aRoof as Roof, ByVal
aPolygon as Polygon2D, ByVal withDialog as
Boolean) as Boolean
```

```
HRESULT PlaceRoof(IRoof* aRoof,
IPolygon2D* aPolygon, VARIANT_BOOL
withDialog, VARIANT_BOOL* retVal);
```

Siehe auch: Story Übersicht, Roof, Polygon2D

Plaziert ein Dach unter Angabe des umrandenden Polygons. Falls der Parameter “ withDialog” “ Wahr” ist, wird der Dacheditor geöffnet. Bei “ Falsch” wird dieser Schritt unterdrückt.

Das Dach selbst muß zuvor mit der Methode ArCon.NewRoof erzeugt werden.

Parameter sind:

aRoof	Das zu plazierende Dach
aPolygon	Die Kontur des Daches
withDialog	Wahr, falls der Dacheditor erscheinen soll, um Eigenschaften des Daches näher zu bestimmen.

Story.PlaceRoofAutomatic

Deklaration:

```
Function PlaceRoofAutomatic(ByVal aRoof as
Roof, ByVal x as Single, ByVal y as Single,
ByVal withDialog as Boolean) as Boolean
```

```
HRESULT PlaceRoofAutomatic(IRoof* aRoof,
float x, float y, VARIANT_BOOL withDialog,
VARIANT_BOOL* retVal);
```

Siehe auch: Story Übersicht, Roof

Plaziert ein Dach mit automatischer Ermittlung der umgebenden Kontur. Vergleiche auch PlaceRoof.

Parameter sind:

aRoof	Das zu plazierende Dach
x, y	Koordinaten eines Punktes innerhalb der automatisch zu ermittelnden Kontur
withDialog	Wahr, wenn der Dacheditor erscheinen soll, um Details des Daches einzustellen.

Story.PlaceStairCase

Deklaration:

```
Function PlaceStairCase(ByVal stair as
StairCase, ByVal withDialog as Boolean) as
Boolean
```

```
HRESULT PlaceStairCase(IStairCase* stair,
VARIANT_BOOL withDialog, VARIANT_BOOL*
retVal);
```

Siehe auch: Story Übersicht, StairCase

Plaziert eine Treppe. Die Koordinaten werden dabei bereits vorher im Treppen-Objekt untergebracht, das mit der Methode ArCon.NewStaircase erzeugt wurde.

Parameter:

stair	das einzufügende Treppen-Objekt
withDialog	Wahr, wenn der Treppen-Dialog erscheinen soll, Falsch wenn die Treppe mit Standardeinstellungen erzeugt wird

Story.PlaceSupport

Deklaration:

```
Function PlaceSupport(ByVal aSupport as
Support, ByVal x as Single, ByVal y as Single)
as Boolean
```

```
HRESULT PlaceSupport(ISupport* aSupport,
float x, float y, VARIANT_BOOL* retVal);
```

Siehe auch: Story Übersicht, Support

Plaziert eine Stütze an den angegebenen Koordinaten. Die Stütze selbst wird durch einen Aufruf von ArCon.NewSupport erzeugt.

Parameter:

aSupport	das einzufügende Stützenobjekt
x, y	die Position der Stütze

Story.PlaceUnterUeberzug

Deklaration:

```
Function PlaceUnterUeberzug(ByVal
UnterUeberzug as UnterUeberzug, ByVal x1 as
Single, ByVal y1 as Single, ByVal x2 as Single,
ByVal y2 as Single, ByVal x3 as Single, ByVal
y3 as Single, ByVal x4 as Single, ByVal y4 as
Single) as Boolean
```

```
HRESULT PlaceUnterUeberzug
(IUnterUeberzug* UnterUeberzug, float x1, float
y1, float x2, float y2, float x3, float y3, float x4,
float y4, VARIANT_BOOL* retVal);
```

Siehe auch: Story Übersicht, UnterUeberzug

Plaziert einen Unter- oder Überzug in einem Stockwerk.

Parameter:

UnterUeberzug der zu plazierende Unter- oder Überzug.

x1, y1 - x4, y4 Koordinaten der vier Eckpunkte des Unter- oder Überzuges.

Story.PlaceVirtualWall

Deklaration:

```
Function PlaceVirtualWall(ByVal aWall as
VirtualWall, ByVal x1 as Single, ByVal y1 as
Single, ByVal x2 as Single, ByVal y2 as Single)
as Boolean
```

```
HRESULT PlaceVirtualWall(IVirtualWall* aWall,
float x1, float y1, float x2, float y2,
VARIANT_BOOL* retVal);
```

Siehe auch: Story Übersicht, VirtualWall

Story.PlaceWall

Deklaration:

```
Function PlaceWall(ByVal aWall as Wall, ByVal
x1 as Single, ByVal y1 as Single, ByVal x2 as
Single, ByVal y2 as Single) as Boolean
```

```
HRESULT PlaceWall(IWall* aWall, float x1, float
y1, float x2, float y2, VARIANT_BOOL* retVal);
```

Siehe auch: Story Übersicht, Wall

Plaziert eine Wand. Das Wandobjekt selbst muß zuvor mit einem Aufruf von ArCon.NewWall erzeugt worden sein. Die beiden angegebenen Koordinatenpaare beschreiben die Endpunkte der Wandachsen. Durch anschließende Wandverschneidungen ändern sich die Form der Wand und auch die Koordinaten der Achsenendpunkte in der Regel noch geringfügig.

Parameter:

aWall	das einzufügende Wand-Objekt
x1, y1	Startpunkt der Wandachse
x2, y2	Endpunkt der Wandachse

Wall

Eine Wand.

Wände werden von ArCon miteinander verschnitten, um “ ordentliche” Übergänge an den Berührungsstellen zu erhalten. Dadurch wird eine Wand in verschiedene Segmente unterteilt, die Sie mit der Liste WallSegments ermitteln können. Weiterhin kann eine Wand durch andere Elemente in ihrer Höhe verändert werden, was durch eine Liste von Höhenpunkten, die LambdaHs, ausgedrückt wird.

Wall Übersicht

Eigenschaften

AverageArea	Die mittlere Wandfläche
AverageAreaFormula	Formel für die mittlere Wandfläche
AverageLength	Die mittlere Wandlänge
Doors	Liste der Türen
History	Daten zur Identifikation im ehemaligen Projekt für nachgeladene Gebäude
Holes	Liste der Aussparungen
ID	Eine eindeutige Kennung
Story	Geschoß, in dem die Wand liegt
Texture	Ursprüngliche Textur der Wand
Thickness	Wanddicke
Type	Wandtyp (Variante des Multifunktionsschalters)
VirtualWall	Virtuelle Wand, die durch diese Wand implementiert wird
WallSegments	Liste der Wandseiten
Windows	Liste der Fenster

Methoden

Delete	Wand löschen
GetHatchStyle	Erfragt die Schraffur der 2D Darstellung
GetLineColor	Liefert die Linienfarbe für neue Wandsegmente in der 2D Darstellung
GetPos	Position der Wand lesen
PlaceDoor	Tür in Wand einsetzen
PlaceHole	Aussparung in Wand einsetzen
PlaceWindow	Fenster in Wand einsetzen
PlaceWindow2	Fenster mit oder ohne Gehrung in Wand einsetzen
SetHatchStyle	Ändert die Schraffur der 2D Darstellung
SetLineColor	Ändert die Linienfarbe für neu entstehende Wandsegmente in der 2D Darstellung
SetPos	Position der Wand ändern

Wall.AverageArea

Deklaration:

Single (nur lesbar)

HRESULT get_AverageArea(float* retVal);

Siehe auch: Wall Übersicht

Die gemittelte Fläche der Wand auf der Wandachse. Multiplizieren Sie diesen Wert mit der Dicke der Wand, so erhalten sie einen Näherungswert für die Wandmasse.

Wall.AverageAreaFormula

Deklaration:

String (nur lesbar)

HRESULT get_AverageAreaFormula(BSTR* retVal);

Siehe auch: Wall Übersicht

Eine Formel zur Berechnung der gemittelten Wandfläche.

Wall.AverageLength

Deklaration:

Single (nur lesbar)

HRESULT get_AverageLength(float* retVal);

Siehe auch: Wall Übersicht

Die Länge der Wandachse.

Wall.Doors

Deklaration:

DoorCollection (nur lesbar)

HRESULT get_Doors(IDoorCollection** retVal);

Siehe auch: Wall Übersicht, DoorCollection

Eine Liste der Türen in dieser Wand.

Wall.History

Deklaration:

IDHistory (nur lesbar)

HRESULT get_History(IIDHistory** retVal);

Siehe auch: Wall Übersicht, IDHistory

Falls dieses Wall beim Ursprünglichen Laden (oder Erstellen) des aktuellen Projektes nicht in diesem Projekt war, sonder aus einem anderen Projekt nachgeladen wurde, liefert diese Eigenschaft das History-Objekt, das die Umsetzung der IDs beschreibt. Andernfalls ist diese Eigenschaft “ Nothing” .

Ein Anwendungsbeispiel für History-Objekte finden Sie im Kapitel Eindeutige Kennungen.

Wall.Holes

Deklaration:

HoleCollection (nur lesbar)

HRESULT get_Holes(IHoleCollection** retVal);

Siehe auch: Wall Übersicht, HoleCollection

Eine Liste von Wandaussparungen in dieser Wand.

Wall.ID

Deklaration:

long (nur lesbar)

HRESULT get_ID(long* retVal);

Siehe auch: Wall Übersicht

Gibt eine eindeutige Kennung dieses Wall an. Diese Kennung ist persistent, das heißt, sie bleibt innerhalb des aktuellen Projektes erhalten. Sie können diese Kennung benutzen, um Walls miteinander zu vergleichen oder zusätzliche Daten zum Wall in einer Datenbank abzulegen.

Die ID ist aber nur innerhalb des aktuellen Projektes eindeutig, wenn Sie (durch Nachladen einzelner Gebäude) das Objekt in ein anderes Projekt einfügen, erhält es eine neue ID. In diesem Fall müssen Sie sofort bei Beendigung des Ladevorganges mit Hilfe der Eigenschaft History objPropertyDetailsWallHistory Ihre zusätzlichen Daten an die neuen Werte anpassen.

Wall.Story

Deklaration:

Story (nur lesbar)

HRESULT get_Story(IStory** retVal);

Siehe auch: Wall Übersicht, Story

Ein Verweis auf das Stockwerk, in dem sich die Wand befindet.

Wall.Texture

Deklaration:

Texture (nur lesbar)

HRESULT get_Texture(ITexture** retVal);

Siehe auch: Wall Übersicht, Texture

Eine Beschreibung der Wandtextur.

Wall.Thickness

Deklaration:

Single

HRESULT get_Thickness(float* retVal);
HRESULT put_Thickness(float newVal);

Siehe auch: Wall Übersicht

Die Dicke der Wand.

Wall.Type

Deklaration:

long

HRESULT get_Type(long* retVal);
HRESULT put_Type(long newVal);

Siehe auch: Wall Übersicht

Der Wandtyp. Gültige Typecodes liegen zwischen 0 und AC_MaxWallType.

Wall.VirtualWall

Deklaration:

VirtualWall (nur lesbar)

HRESULT get_VirtualWall(IVirtualWall** retVal);

Siehe auch: Wall Übersicht, VirtualWall

Wall.WallSegments

Deklaration:

WallSegmentCollection (nur lesbar)

```
HRESULT get_WallSegments(IWallSegmentCollection**
retVal);
```

Siehe auch: Wall Übersicht, WallSegmentCollection

Eine Liste mit kontinuierlichen Wandstücken (Wandseiten) dieser Wand.

Wall.Windows

Deklaration:

WindowCollection (nur lesbar)

HRESULT get_Windows(IWindowCollection** retVal);

Siehe auch: Wall Übersicht, WindowCollection

Eine Liste mit Fenstern in dieser Wand.

Wall.Delete

Deklaration:

Sub Delete()

HRESULT Delete();

Siehe auch: Wall Übersicht

Löscht die Wand.

Wall.GetHatchStyle

Deklaration:

Sub GetHatchStyle()

HRESULT GetHatchStyle(AC_Hatch_Style *HatchStyle, long *HatchColor);

Siehe auch: Wall Übersicht

Wall.GetLineColor

Deklaration:

```
Function GetLineColor() as long
```

```
HRESULT GetLineColor(long* retVal);
```

Siehe auch: Wall Übersicht

Wall.GetPos

Deklaration:

```
Function GetPos(ByRef X1 as Single, ByRef Y1 as Single,
ByRef X2 as Single, ByRef Y2 as Single) as Boolean
```

```
HRESULT GetPos(float* X1, float* Y1, float* X2, float* Y2,
VARIANT_BOOL* retVal);
```

Siehe auch: Wall Übersicht

Ermittelt die Koordinaten der Wandachsenendpunkte. Diese können sich durch weitere Wandverschneidungen später ändern!

Parameter sind:

x1, y1	Nehmen die Koordinaten des Anfangspunktes auf
x2, y2	Nehmen die Koordinaten des Endpunktes auf

Wall.PlaceDoor

Deklaration:

```
Function PlaceDoor(ByVal aDoor as Door, ByVal x as
Single, ByVal y as Single) as Boolean
```

```
HRESULT PlaceDoor(IDoor* aDoor, float x, float y,
VARIANT_BOOL* retVal);
```

Siehe auch: Wall Übersicht, Door

Plaziert eine Tür in der Wand.

Parameter sind:

aDoor	Ein Türobjekt
x, y	Koordinaten des Türmittelpunktes. Dieser muß in der Wand liegen, ansonsten schlägt der Aufruf fehl.

Wall.PlaceHole

Deklaration:

```
Function PlaceHole(ByVal aHole as Hole, ByVal aPolygon
as Polygon2D, ByVal x as Single, ByVal y as Single) as
Boolean
```

```
HRESULT PlaceHole(IHole* aHole, IPolygon2D*
aPolygon, float x, float y, VARIANT_BOOL* retVal);
```

Siehe auch: Wall Übersicht, Hole, Polygon2D

Plaziert eine Wandaussparung.

Parameter sind:

aHole	Ein Wandlochobjekt
aPolygon	Die Umrandung der Aussparung. Anders als in der Benutzeroberfläche kann ArCon intern beliebig geformte Aussparungen verwalten. Dieses Polygon gibt die Form an (senkrecht gekippt).
x, y	Der Mittelpunkt der Aussparung. Dieser Punkt muß in der Wand liegen.

Wall.PlaceWindow

Deklaration:

```
Function PlaceWindow(ByVal aWindow as Window,
ByVal x as Single, ByVal y as Single) as Boolean
```

```
HRESULT PlaceWindow(IWindow* aWindow, float x,
float y, VARIANT_BOOL* retVal);
```

Siehe auch: Wall Übersicht, Window

Plaziert ein Fenster in der Wand.

Parameter sind:

aWindow	ein Fensterobjekt
x, y	Koordinaten des Fenstermittelpunktes

Wall.PlaceWindow2

Deklaration:

```
Function PlaceWindow2(ByVal aWindow as Window,
ByVal x as Single, ByVal y as Single, ByVal
MitGehrungLinks as Boolean, ByVal MitGehrungRechts
as Boolean) as Boolean
```

```
HRESULT PlaceWindow2(IWindow* aWindow, float x,
float y, VARIANT_BOOL MitGehrungLinks,
VARIANT_BOOL MitGehrungRechts, VARIANT_BOOL*
retVal);
```

Siehe auch: Wall Übersicht, Window

Wall.SetHatchStyle

Deklaration:

```
Sub SetHatchStyle(ByVal HatchStyle as AC_Hatch_Style,
ByVal HatchColor as long)
```

```
HRESULT SetHatchStyle(AC_Hatch_Style HatchStyle,
long HatchColor);
```

Siehe auch: Wall Übersicht

Wall.SetLineColor

Deklaration:

```
Sub SetLineColor(ByVal col as long)
```

```
HRESULT SetLineColor(long col);
```

Siehe auch: Wall Übersicht

Wall.SetPos

Deklaration:

```
Function SetPos(ByVal X1 as Single, ByVal Y1 as Single,
ByVal X2 as Single, ByVal Y2 as Single) as Boolean
```

```
HRESULT SetPos(float X1, float Y1, float X2, float Y2,
VARIANT_BOOL* retVal);
```

Siehe auch: Wall Übersicht

VirtualWall

VirtualWall Übersicht

Eigenschaften

Color	Farbe der Linien in der 2D Darstellung
History	Daten zur Identifikation im ehemaligen Projekt für nachgeladene Gebäude
ID	Eine eindeutige Kennung
Story	Stockwerk, zu dem die virtuelle Wand gehört
Style	Linienstil in der 2D Darstellung
Thickness	Tiefe der Löcher (falls diese erlaubt sind)
Wall	Wand-Objekt der virtuellen Wand
WithHoles	Sind Löcher in der virtuellen Wand erlaubt?

Methoden

Delete	Löscht die virtuelle Wand
GetPos	Liefert die aktuelle Position
SetPos	Setzt die aktuelle Position

VirtualWall.Color

Deklaration:

long

HRESULT get_Color(long* retVal);
HRESULT put_Color(long newVal);

Siehe auch: VirtualWall Übersicht

VirtualWall.History

Deklaration:

IDHistory (nur lesbar)

HRESULT get_History(IIDHistory** retVal);

Siehe auch: VirtualWall Übersicht, IDHistory

VirtualWall.ID

Deklaration:

long (nur lesbar)

HRESULT get_ID(long* retVal);

Siehe auch: VirtualWall Übersicht

VirtualWall.Story

Deklaration:

Story (nur lesbar)

HRESULT get_Story(IStory** retVal);

Siehe auch: VirtualWall Übersicht, Story

VirtualWall.Style

Deklaration:

long

HRESULT get_Style(long* retVal);
HRESULT put_Style(long newVal);

Siehe auch: VirtualWall Übersicht

VirtualWall.Thickness

Deklaration:

Single

HRESULT get_Thickness(float* retVal);
HRESULT put_Thickness(float newVal);

Siehe auch: VirtualWall Übersicht

VirtualWall.Wall

Deklaration:

Wall (nur lesbar)

HRESULT get_Wall(IWall** retVal);

Siehe auch: VirtualWall Übersicht, Wall

VirtualWall.WithHoles

Deklaration:

Boolean

HRESULT get_WithHoles(VARIANT_BOOL* retVal);
HRESULT put_WithHoles(VARIANT_BOOL newVal);

Siehe auch: VirtualWall Übersicht

VirtualWall.Delete

Deklaration:

```
Function Delete() as Boolean
```

```
HRESULT Delete(VARIANT_BOOL* retVal);
```

Siehe auch: VirtualWall Übersicht

VirtualWall.GetPos

Deklaration:

```
Function GetPos(ByRef X1 as Single, ByRef Y1 as Single,
ByRef X2 as Single, ByRef Y2 as Single) as Boolean
```

```
HRESULT GetPos(float* X1, float* Y1, float* X2, float* Y2,
VARIANT_BOOL* retVal);
```

Siehe auch: VirtualWall Übersicht

VirtualWall.SetPos

Deklaration:

```
Function SetPos(ByVal X1 as Single, ByVal Y1 as Single,
ByVal X2 as Single, ByVal Y2 as Single) as Boolean
```

```
HRESULT SetPos(float X1, float Y1, float X2, float Y2,
VARIANT_BOOL* retVal);
```

Siehe auch: VirtualWall Übersicht

Labeling

Stellt eine freie Beschriftung im ArCon Konstruktionsmodus da. Nicht zu Verwechseln mit Label, einem 2D Grafikelement.

Labeling Übersicht

Eigenschaften

Angle	Drehwinkel
Color	Textfarbe
Font	Schriftart der Beschriftung
History	Daten zur Identifikation im ehemaligen Projekt für einzeln geladene Gebäude
ID	Liefert eine eindeutige Kennung
Story	Stockwerk, in dem die Beschriftung angezeigt wird
Text	Beschriftungstext

Methoden

Delete	Beschriftung löschen
GetPos	Position ermitteln
SetPos	Position setzen

Labeling.Angle

Deklaration:

Single

HRESULT get_Angle(float* retVal);
HRESULT put_Angle(float newVal);

Siehe auch: Labeling Übersicht

Der Winkel der Schrift zur Horizontalen. Er wird im mathematischen Drehsinn (positiv = gegen den Uhrzeigersinn) und in Bogenmaß angegeben.

Labeling.Color

Deklaration:

long

HRESULT get_Color(long* retVal);
HRESULT put_Color(long newVal);

Siehe auch: Labeling Übersicht

Die Textfarbe als RGB-Wert.

Labeling.Font

Deklaration:

VARIANT

HRESULT get_Font(VARIANT* retVal);
HRESULT put_Font(VARIANT newVal);

Siehe auch: Labeling Übersicht

Die verwendete Schriftart.

Das hier übergebene “ VARIANT” -Objekt muß einen OLE-Font (Interface IFont oder IFontDisp, bzw. in Basic StdFont) enthalten.

Labeling.History

Deklaration:

IDHistory (nur lesbar)

HRESULT get_History(IIDHistory** retVal);

Siehe auch: Labeling Übersicht, IDHistory

Falls dieses Labeling beim Ursprünglichen Laden (oder Erstellen) des aktuellen Projektes nicht in diesem Projekt war, sonder aus einem anderen Projekt nachgeladen wurde, liefert diese Eigenschaft das History-Objekt, das die Umsetzung der IDs beschreibt. Andernfalls ist diese Eigenschaft “ Nothing” .

Ein Anwendungsbeispiel für History-Objekte finden Sie im Kapitel Eindeutige Kennungen.

Labeling.ID

Deklaration:

long (nur lesbar)

HRESULT get_ID(long* retVal);

Siehe auch: Labeling Übersicht

Gibt eine eindeutige Kennung dieses Labeling an. Diese Kennung ist persistent, das heißt, sie bleibt innerhalb des aktuellen Projektes erhalten. Sie können diese Kennung benutzen, um Labelings miteinander zu vergleichen oder zusätzliche Daten zum Labeling in einer Datenbank abzulegen.

Die ID ist aber nur innerhalb des aktuellen Projektes eindeutig, wenn Sie (durch Nachladen einzelner Gebäude) das Objekt in ein anderes Projekt einfügen, erhält es eine neue ID. In diesem Fall müssen Sie sofort bei Beendigung des Ladevorganges mit Hilfe der Eigenschaft History objPropertyDetailsLabelingHistory Ihre zusätzlichen Daten an die neuen Werte anpassen.

Labeling.Story

Deklaration:

Story (nur lesbar)

HRESULT get_Story(IStory** retVal);

Siehe auch: Labeling Übersicht, Story

Das Geschoß, in dem die Beschriftung angeordnet ist.

Labeling.Text

Deklaration:

String

HRESULT get_Text(BSTR* retVal);
HRESULT put_Text(BSTR newVal);

Siehe auch: Labeling Übersicht

Der eigentliche Text.

Labeling.Delete

Deklaration:

Sub Delete()

HRESULT Delete();

Siehe auch: Labeling Übersicht

Löscht die Beschriftung.

Labeling.GetPos

Deklaration:

```
Function GetPos(ByRef X1 as Single, ByRef Y1 as Single)
as Boolean
```

```
HRESULT GetPos(float* X1, float* Y1, VARIANT_BOOL*
retVal);
```

Siehe auch: Labeling Übersicht

Liefert die Position der Beschriftung.

Labeling.SetPos

Deklaration:

```
Function SetPos(ByVal X1 as Single, ByVal Y1 as Single) as
Boolean
```

```
HRESULT SetPos(float X1, float Y1, VARIANT_BOOL*
retVal);
```

Siehe auch: Labeling Übersicht

Setzt die Position der Beschriftung.

CeilingOpening

Eine Deckenaussparung (ein “ Loch” im Boden oder der Decke).

CeilingOpening Übersicht

Eigenschaften

History	Daten zur Identifikation im ehemaligen Projekt für nachgeladene Gebäude
ID	Liefert eine eindeutige Kennung
Polygon	Kontur der Deckenaussparung

Methoden

Delete	Deckenaussparung löschen

CeilingOpening.History

Deklaration:

IDHistory (nur lesbar)

HRESULT get_History(IIDHistory** retVal);

Siehe auch: CeilingOpening Übersicht, IDHistory

Falls dieses CeilingOpening beim Ursprünglichen Laden (oder Erstellen) des aktuellen Projektes nicht in diesem Projekt war, sonder aus einem anderen Projekt nachgeladen wurde, liefert diese Eigenschaft das History-Objekt, das die Umsetzung der IDs beschreibt. Andernfalls ist diese Eigenschaft “ Nothing” .

Ein Anwendungsbeispiel für History-Objekte finden Sie im Kapitel Eindeutige Kennungen.

CeilingOpening.ID

Deklaration:

long (nur lesbar)

HRESULT get_ID(long* retVal);

Siehe auch: CeilingOpening Übersicht

Gibt eine eindeutige Kennung dieses CeilingOpening an. Diese Kennung ist persistent, das heißt, sie bleibt innerhalb des aktuellen Projektes erhalten. Sie können diese Kennung benutzen, um CeilingOpenings miteinander zu vergleichen oder zusätzliche Daten zum CeilingOpening in einer Datenbank abzulegen.

Die ID ist aber nur innerhalb des aktuellen Projektes eindeutig, wenn Sie (durch Nachladen einzelner Gebäude) das Objekt in ein anderes Projekt einfügen, erhält es eine neue ID. In diesem Fall müssen Sie sofort bei Beendigung des Ladevorganges mit Hilfe der Eigenschaft History objPropertyDetailsCeilingOpeningHistory Ihre zusätzlichen Daten an die neuen Werte anpassen.

CeilingOpening.Polygon

Deklaration:

Polygon2D

HRESULT get_Polygon(IPolygon2D** retVal);
HRESULT put_Polygon(IPolygon2D* newVal);

Siehe auch: CeilingOpening Übersicht, Polygon2D

Die Umrandung der Deckenaussparung.

CeilingOpening.Delete

Deklaration:

Sub Delete()

HRESULT Delete();

Siehe auch: CeilingOpening Übersicht

Löscht die Deckenaussparung.

StairCase

Eine ArCon Treppe.

StairCase Übersicht

Eigenschaften

AuftrittsBreite	Auftrittsbreite
GelaenderHoeheLinks	Bruestungshöhe senkrecht gemessen
GelaenderHoeheRechts	Bruestungshöhe senkrecht gemessen
GelaenderLaengeLinks	Laenge des Gelaenders im Grundriss
GelaenderLaengeRechts	Laenge des Gelaenders im Grundriss
History	Daten zur Identifikation im ehemaligen Projekt für nachgeladene Gebäude
Hoehe	Höhe der Treppe
ID	Eine eindeutige Kennung
LaufBreiteAussen	Äussere Laufbreite der Treppe
LaufLaenge	Lauflänge der Treppe
LaufPlattenDicke	Laufplattendicke der Treppe
Steigung	Steigung
Story	Geschoß, in dem die Treppe definiert ist
TextureCount	Anzahl der max. verfügbaren Texturen
Type	Typ der Treppe (Variante des Multifunktionsschalters)
X1	Für Dreipunkteingabe: x-Koordinate Punkt 1
X2	Für Dreipunkteingabe: x-Koordinate Punkt 2
X3	Für Dreipunkteingabe: x-Koordinate Punkt 3
Y1	Für Dreipunkteingabe: y-Koordinate Punkt 1
Y2	Für Dreipunkteingabe: y-Koordinate Punkt 2
Y3	Für Dreipunkteingabe: y-Koordinate Punkt 3

Methoden

Delete	Treppe löschen
Edit	Dialogbox für Treppe öffnen
GetPolygonWendelConstruction	Liefert ein Detailobjekt, wenn es sich um eine polygonale Wendeltreppe handelt.
GetTexture	Liefert eine Textur
SetTexture	Tauscht eine Textur aus

StairCase.AuftrittsBreite

Deklaration:

Single (nur lesbar)

HRESULT get_AuftrittsBreite(float* retVal);

Siehe auch: StairCase Übersicht

StairCase.GelaenderHoeheLinks

Deklaration:

Single (nur lesbar)

HRESULT get_GelaenderHoeheLinks(float* retVal);

Siehe auch: StairCase Übersicht

StairCase.GelaenderHoeheRechts

Deklaration:

Single (nur lesbar)

HRESULT get_GelaenderHoeheRechts(float* retVal);

Siehe auch: StairCase Übersicht

StairCase.GelaenderLaengeLinks

Deklaration:

Single (nur lesbar)

HRESULT get_GelaenderLaengeLinks(float* retVal);

Siehe auch: StairCase Übersicht

StairCase.GelaenderLaengeRechts

Deklaration:

Single (nur lesbar)

HRESULT get_GelaenderLaengeRechts(float* retVal);

Siehe auch: StairCase Übersicht

StairCase.History

Deklaration:

IDHistory (nur lesbar)

HRESULT get_History(IIDHistory** retVal);

Siehe auch: StairCase Übersicht, IDHistory

Falls dieses StairCase beim Ursprünglichen Laden (oder Erstellen) des aktuellen Projektes nicht in diesem Projekt war, sonder aus einem anderen Projekt nachgeladen wurde, liefert diese Eigenschaft das History-Objekt, das die Umsetzung der IDs beschreibt. Andernfalls ist diese Eigenschaft “ Nothing” .

Ein Anwendungsbeispiel für History-Objekte finden Sie im Kapitel Eindeutige Kennungen.

StairCase.Hoehe

Deklaration:

Single (nur lesbar)

HRESULT get_Hoehe(float* retVal);

Siehe auch: StairCase Übersicht

StairCase.ID

Deklaration:

long (nur lesbar)

HRESULT get_ID(long* retVal);

Siehe auch: StairCase Übersicht

Gibt eine eindeutige Kennung dieses StairCase an. Diese Kennung ist persistent, das heißt, sie bleibt innerhalb des aktuellen Projektes erhalten. Sie können diese Kennung benutzen, um StairCases miteinander zu vergleichen oder zusätzliche Daten zum StairCase in einer Datenbank abzulegen.

Die ID ist aber nur innerhalb des aktuellen Projektes eindeutig, wenn Sie (durch Nachladen einzelner Gebäude) das Objekt in ein anderes Projekt einfügen, erhält es eine neue ID. In diesem Fall müssen Sie sofort bei Beendigung des Ladevorganges mit Hilfe der Eigenschaft History objPropertyDetailsStairCaseHistory Ihre zusätzlichen Daten an die neuen Werte anpassen.

StairCase.LaufBreiteAussen

Deklaration:

Single (nur lesbar)

HRESULT get_LaufBreiteAussen(float* retVal);

Siehe auch: StairCase Übersicht

StairCase.LaufLaenge

Deklaration:

Single (nur lesbar)

HRESULT get_LaufLaenge(float* retVal);

Siehe auch: StairCase Übersicht

StairCase.LaufPlattenDicke

Deklaration:

Single (nur lesbar)

HRESULT get_LaufPlattenDicke(float* retVal);

Siehe auch: StairCase Übersicht

StairCase.Steigung

Deklaration:

Single (nur lesbar)

HRESULT get_Steigung(float* retVal);

Siehe auch: StairCase Übersicht

StairCase.Story

Deklaration:

Story (nur lesbar)

HRESULT get_Story(IStory** retVal);

Siehe auch: StairCase Übersicht, Story

Das Stockwerk, in dem sich die Treppe befindet.

StairCase.TextureCount

Deklaration:

long (nur lesbar)

HRESULT get_TextureCount(long* retVal);

Siehe auch: StairCase Übersicht

StairCase.Type

Deklaration:

long

HRESULT get_Type(long* retVal);
HRESULT put_Type(long newVal);

Siehe auch: StairCase Übersicht

Die Treppenvariante. Treppentypen werden dynamisch geladen, sodaß der Bereich der gültigen Typcodes von der ArCon Installation abhängig ist. Sie können die Anzahl der verfügbaren Typen mit der Funktion ArCon.AvailableStairCases erfragen. Typkodierungen beginnen in ArCon immer mit dem Typ 0.

StairCase.X1

Deklaration:

Single

HRESULT get_X1(float* retVal);
HRESULT put_X1(float newVal);

Siehe auch: StairCase Übersicht

Die X Koordinate des ersten Referenzpunktes aus der Dreipunkteingabe (siehe auch Y1, X2, Y2, X3, Y3)

Diese Koordinaten entsprechen den drei Punkten, die der Benutzer beim Erstellen der Treppe angeklickt hat.

StairCase.X2

Deklaration:

Single

HRESULT get_X2(float* retVal);
HRESULT put_X2(float newVal);

Siehe auch: StairCase Übersicht

Die X-Koordinate des zweiten Punktes aus der Dreipunkteingabe (siehe auch X1, Y1, Y2, X3, Y3)

Diese Koordinaten entsprechen den drei Punkten, die der Benutzer beim Erstellen der Treppe angeklickt hat.

StairCase.X3

Deklaration:

Single

HRESULT get_X3(float* retVal);
HRESULT put_X3(float newVal);

Siehe auch: StairCase Übersicht

Die X-Koordinate des dritten Punktes aus der Dreipunkteingabe (siehe auch X1, Y1, X2, Y2, Y3).

Diese Koordinaten entsprechen den drei Punkten, die der Benutzer beim Erstellen der Treppe angeklickt hat.

StairCase.Y1

Deklaration:

Single

HRESULT get_Y1(float* retVal);
HRESULT put_Y1(float newVal);

Siehe auch: StairCase Übersicht

Die Y-Koordinate des ersten Punktes aus der Dreipunkteingabe (siehe auch X1, X2, Y2, X3, Y3)

Diese Koordinaten entsprechen den drei Punkten, die der Benutzer beim Erstellen der Treppe angeklickt hat.

StairCase.Y2

Deklaration:

Single

HRESULT get_Y2(float* retVal);
HRESULT put_Y2(float newVal);

Siehe auch: StairCase Übersicht

Die Y-Koordinate des zweiten Punktes aus der Dreipunkteingabe (siehe auch X1, Y1, X2, X3, Y3)

Diese Koordinaten entsprechen den drei Punkten, die der Benutzer beim Erstellen der Treppe angeklickt hat.

StairCase.Y3

Deklaration:

Single

HRESULT get_Y3(float* retVal);
HRESULT put_Y3(float newVal);

Siehe auch: StairCase Übersicht

Die Y-Koordinate des dritten Punktes aus der Dreipunkteingabe (siehe auch X1, Y1, X2, Y2, X3)

Diese Koordinaten entsprechen den drei Punkten, die der Benutzer beim Erstellen der Treppe angeklickt hat.

StairCase.Delete

Deklaration:

Sub Delete()

HRESULT Delete();

Siehe auch: StairCase Übersicht

Löscht die gesamte Treppe.

StairCase.Edit

Deklaration:

```
Function Edit() as Boolean
```

```
HRESULT Edit(VARIANT_BOOL* retVal);
```

Siehe auch: StairCase Übersicht

Aktiviert den Treppeneditor, so als ob der Benutzer die Treppe selektiert und dann doppelt angeklickt hätte.

StairCase.GetPolygonWendelConstruction

Deklaration:

```
Function GetPolygonWendelConstruction() as
PolygonWendelConstruction
```

```
HRESULT GetPolygonWendelConstruction
(IPolygonWendelConstruction** retVal);
```

Siehe auch: StairCase Übersicht, PolygonWendelConstruction

StairCase.GetTexture

Deklaration:

```
Function GetTexture(ByVal index as long) as String
```

```
HRESULT GetTexture(long index, BSTR* retVal);
```

Siehe auch: StairCase Übersicht

StairCase.SetTexture

Deklaration:

```
Function SetTexture(ByVal index as long, ByVal
TextureName as String) as Boolean
```

```
HRESULT SetTexture(long index, BSTR TextureName,
VARIANT_BOOL* retVal);
```

Siehe auch: StairCase Übersicht

PolygonWendelConstruction

PolygonWendelConstruction Übersicht

Eigenschaften

Achslage	Lage des Eingabepolygons relativ zur Treppe (ACPWA_Links, ACPWA_Mitte oder ACPWA_Rechts)
AnzahlSteigungen	Anzahl der Stufen
AuftrittsBreite	Breite der Stufen
Bauart	Bauart der Treppe (ACPWS_Holztreppe oder ACPWS_Massivtreppe
FusslaufBreite	Bei Holzwangentreppen die Wangenbreite
FusslaufDicke	Bei Holzwangentreppen die Wangendicke
GelaenderBauart	Art des Geländers (ACPW_Gelaender_Standard, ...)
HandlaufBreite	Breite des Handlaufs
HandlaufDicke	Dicke des Handlaufs
HandlaufDurchmesser	Durchmesser des Handlaufs
HandlaufHoehe	Höhe des Handlaufs
HandlaufMitKnicken	Hat der Handlauf Knicke?
HandlaufQuerschnitt	Querschnitt des Handlaufs (ACPWQ_Rechteck oder ACPWQ_Kreis)
Hoehe	Höhe der Treppe
LaufBreiteAussen	Breite des Außenlaufs
LaufLaenge	Länge des Geländers
MaximaleAuslenkungLinks	Maximale Abweichung von der Ideallinie links
MaximaleAuslenkungRechts	Maximale Abweichung von der Ideallinie rechts
MitGelaenderLinks	Hat die Treppe links ein Geländer?
MitGelaenderRechts	Hat die Treppe rechts ein Geländer?
MitSetzStufen	Hat die Treppe Setzstufen?
MittlererFusslaufAbstand	Mittlerer Abstand des Fußlaufs von der Wange
NiveauAmAntritt	Höhe der untersten Stufe
PfostenBreite	Breite der Pfosten
PfostenDicke	Dicke der Pfosten
PfostenDurchmesser	Durchmesser der Pfosten
PfostenLaenge	Länge der Pfosten
PfostenObjektName	Name des Pfostenobjektes, fals der Querschnitt ACPWQ_3DObjekt ist. Maximal 12 Zeichen lang, ohne . ACO Endung.
PfostenQuerschnitt	Querschnitt der Pfosten (ACPWQ_Rechteck, ACPWQ_Kreis oder ACPWQ_3DObjekt)
SchrittMassOptimieren	Soll die Schrittweite der Treppe optimiert werden?
StabBreite	Breite der Stäbe
StabDicke	Dicke der Stäbe
StabDurchmesser	Durchmesser der Stäbe
StabMaximalerAbstand	Maximaler Abstand der Stäbe
StabQuerschnitt	Querschnitt der Stäbe (ACPWQ_Rechteck oder ACPWQ_Kreis)
WinkelAmAntritt	Winkel der untersten Stufe

WinkelAmAustritt	Winkel der letzten Stufe

Methoden

CreateStairCase	Erzeugt eine Treppe mit diesen Einstellungen
GetPolygon	Liefert das Eingabepolygon der Wendeltreppe
SetPolygon	Definiert ein neues Polygon für die Wendeltreppe

PolygonWendelConstruction.Achslage

Deklaration:

long

HRESULT get_Achslage(long* retVal);
HRESULT put_Achslage(long newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.AnzahlSteigungen

Deklaration:

long

HRESULT get_AnzahlSteigungen(long* retVal);
HRESULT put_AnzahlSteigungen(long newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.AuftrittsBreite

Deklaration:

Single (nur lesbar)

HRESULT get_AuftrittsBreite(float* retVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.Bauart

Deklaration:

long

HRESULT get_Bauart(long* retVal);
HRESULT put_Bauart(long newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.FusslaufBreite

Deklaration:

Single

HRESULT get_FusslaufBreite(float* retVal);
HRESULT put_FusslaufBreite(float newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.FusslaufDicke

Deklaration:

Single

HRESULT get_FusslaufDicke(float* retVal);
HRESULT put_FusslaufDicke(float newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.GelaenderBauart

Deklaration:

long

HRESULT get_GelaenderBauart(long* retVal);
HRESULT put_GelaenderBauart(long newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.HandlaufBreite

Deklaration:

Single

HRESULT get_HandlaufBreite(float* retVal);
HRESULT put_HandlaufBreite(float newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.HandlaufDicke

Deklaration:

Single

HRESULT get_HandlaufDicke(float* retVal);
HRESULT put_HandlaufDicke(float newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.HandlaufDurchmesser

Deklaration:

Single

HRESULT get_HandlaufDurchmesser(float* retVal);
HRESULT put_HandlaufDurchmesser(float newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.HandlaufHoehe

Deklaration:

Single

HRESULT get_HandlaufHoehe(float* retVal);
HRESULT put_HandlaufHoehe(float newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.HandlaufMitKnicken

Deklaration:

Boolean

HRESULT get_HandlaufMitKnicken(VARIANT_BOOL* retVal);

HRESULT put_HandlaufMitKnicken(VARIANT_BOOL newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.HandlaufQuerschnitt

Deklaration:

long

HRESULT get_HandlaufQuerschnitt(long* retVal);
HRESULT put_HandlaufQuerschnitt(long newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.Hoehe

Deklaration:

Single

HRESULT get_Hoehe(float* retVal);
HRESULT put_Hoehe(float newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.LaufBreiteAussen

Deklaration:

Single

HRESULT get_LaufBreiteAussen(float* retVal);
HRESULT put_LaufBreiteAussen(float newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.LaufLaenge

Deklaration:

Single (nur lesbar)

HRESULT get_LaufLaenge(float* retVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.MaximaleAuslenkungLinks

Deklaration:

Single

HRESULT get_MaximaleAuslenkungLinks(float* retVal);
HRESULT put_MaximaleAuslenkungLinks(float newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.MaximaleAuslenkungRechts

Deklaration:

Single

HRESULT get_MaximaleAuslenkungRechts(float* retVal);
HRESULT put_MaximaleAuslenkungRechts(float newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.MitGelaenderLinks

Deklaration:

Boolean

```
HRESULT get_MitGelaenderLinks(VARIANT_BOOL* retVal)
;
HRESULT put_MitGelaenderLinks(VARIANT_BOOL newVal)
;
```

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.MitGelaenderRechts

Deklaration:

Boolean

HRESULT get_MitGelaenderRechts(VARIANT_BOOL* retVal);

HRESULT put_MitGelaenderRechts(VARIANT_BOOL newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.MitSetzStufen

Deklaration:

Boolean

HRESULT get_MitSetzStufen(VARIANT_BOOL* retVal);
HRESULT put_MitSetzStufen(VARIANT_BOOL newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.MittlererFusslaufAbstand

Deklaration:

Single

HRESULT get_MittlererFusslaufAbstand(float* retVal);
HRESULT put_MittlererFusslaufAbstand(float newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.NiveauAmAntritt

Deklaration:

Single

HRESULT get_NiveauAmAntritt(float* retVal);
HRESULT put_NiveauAmAntritt(float newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.PfostenBreite

Deklaration:

Single

HRESULT get_PfostenBreite(float* retVal);
HRESULT put_PfostenBreite(float newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.PfostenDicke

Deklaration:

Single

HRESULT get_PfostenDicke(float* retVal);
HRESULT put_PfostenDicke(float newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.PfostenDurchmesser

Deklaration:

Single

HRESULT get_PfostenDurchmesser(float* retVal);
HRESULT put_PfostenDurchmesser(float newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.PfostenLaenge

Deklaration:

Single

HRESULT get_PfostenLaenge(float* retVal);
HRESULT put_PfostenLaenge(float newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.PfostenObjektName

Deklaration:

String

HRESULT get_PfostenObjektName(BSTR* retVal);
HRESULT put_PfostenObjektName(BSTR newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.PfostenQuerschnitt

Deklaration:

long

HRESULT get_PfostenQuerschnitt(long* retVal);
HRESULT put_PfostenQuerschnitt(long newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.SchrittMassOptimieren

Deklaration:

Boolean

HRESULT get_SchrittMassOptimieren(VARIANT_BOOL* retVal);

HRESULT put_SchrittMassOptimieren(VARIANT_BOOL newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.StabBreite

Deklaration:

Single

HRESULT get_StabBreite(float* retVal);
HRESULT put_StabBreite(float newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.StabDicke

Deklaration:

Single

HRESULT get_StabDicke(float* retVal);
HRESULT put_StabDicke(float newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.StabDurchmesser

Deklaration:

Single

HRESULT get_StabDurchmesser(float* retVal);
HRESULT put_StabDurchmesser(float newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.StabMaximalerAbstand

Deklaration:

Single

HRESULT get_StabMaximalerAbstand(float* retVal);
HRESULT put_StabMaximalerAbstand(float newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.StabQuerschnitt

Deklaration:

long

HRESULT get_StabQuerschnitt(long* retVal);
HRESULT put_StabQuerschnitt(long newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.WinkelAmAntritt

Deklaration:

Single

HRESULT get_WinkelAmAntritt(float* retVal);
HRESULT put_WinkelAmAntritt(float newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.WinkelAmAustritt

Deklaration:

Single

HRESULT get_WinkelAmAustritt(float* retVal);
HRESULT put_WinkelAmAustritt(float newVal);

Siehe auch: PolygonWendelConstruction Übersicht

PolygonWendelConstruction.CreateStairCase

Deklaration:

```
Function CreateStairCase(ByVal TreppenTyp as long) as
StairCase
```

```
HRESULT CreateStairCase(long TreppenTyp, IStairCase**
retVal);
```

Siehe auch: PolygonWendelConstruction Übersicht, StairCase

PolygonWendelConstruction.GetPolygon

Deklaration:

```
Function GetPolygon() as Point2DCollection
```

```
HRESULT GetPolygon(IPoint2DCollection** retVal);
```

Siehe auch: PolygonWendelConstruction Übersicht, Point2DCollection

PolygonWendelConstruction.SetPolygon

Deklaration:

```
Sub SetPolygon(ByVal nVal as Point2DCollection)
```

```
HRESULT SetPolygon(IPoint2DCollection* nVal);
```

Siehe auch: PolygonWendelConstruction Übersicht, Point2DCollection

Roof

Ein Dach mitsamt seiner Holzkonstruktion.

Roof Übersicht

Eigenschaften

Construction	Konstruktive Details des Daches
DachSummeFirstlaenge	Länge aller Firste (ohne Gauben)
DachSummeFlaeche	Fläche des Daches (ohne Gauben)
DachSummeGratlaenge	Länge aller Grate (ohne Gauben)
DachSummeKehlenlaenge	Länge aller Kehlen (ohne Gauben)
DachSummeOrtganglaenge	Länge aller Ortgänge (ohne Gauben)
DachSummePultdachFirstlaenge	Länge aller Pultdachfirste (ohne Gauben)
DachSummeTrauflaenge	Länge aller Traufgänge (ohne Gauben)
FlaecheDerEindeckungUebermessen	Gesamtfläche der Dachkonstruktion in Quadratmetern
Gauben	Liste der Gauben
GaubenSummeFirstlaenge	Länge aller Firste (nur Gauben)
GaubenSummeFlaeche	Fläche des Daches (nur Gauben)
GaubenSummeGratlaenge	Länge aller Grate (nur Gauben)
GaubenSummeKehlenlaenge	Länge aller Kehlen (nur Gauben)
GaubenSummeOrtganglaenge	Länge aller Ortgänge (nur Gauben)
GaubenSummePultdachFirstlaenge	Länge aller Pultdachfirste (nur Gauben)
GaubenSummeTrauflaenge	Länge aller Traufgänge (nur Gauben)
History	Daten zur Identifikation im ehemaligen Projekt für nachgeladene Gebäude
ID	Liefert eine eindeutige Kennung
Polygon	Form des Daches
RoofAreas	Liste der Deckenflächen
Story	Stockwerk, in dem das Dach liegt
TextureCount	Anzahl der max. verfügbaren Texturen
TotalSummeFirstlaenge	Länge aller Firste
TotalSummeFlaeche	Fläche des Daches
TotalSummeGratlaenge	Länge aller Grate
TotalSummeKehlenlaenge	Länge aller Kehlen
TotalSummeOrtganglaenge	Länge aller Ortgänge
TotalSummePultdachFirstlaenge	Länge aller Pultdachfirste
TotalSummeTrauflaenge	Länge aller Traufgänge
Windows	Liste der Dachfenster

Methoden

CreateGaube	Erzeugt eine Gaube
Delete	Dach löschen
Edit	Dialogbox für Dach öffnen
GetTexture	Liefert eine Textur
GetWoodConstruction	Liefert eine Liste der Holzbalken. Optional werden Balken, deren Maße sich um weniger als epsilon unterscheiden, zusammengefaßt.

PlaceWindow	Plaziert ein Dachfenster
SetTexture	Tauscht eine Textur aus

Roof.Construction

Deklaration:

RoofConstruction (nur lesbar)

HRESULT get_Construction(IRoofConstruction** retVal);

Siehe auch: Roof Übersicht, RoofConstruction

Roof.DachSummeFirstlaenge

Deklaration:

double (nur lesbar)

HRESULT get_DachSummeFirstlaenge(double* retVal);

Siehe auch: Roof Übersicht

Gibt die Summe der Längen aller Firste des Daches - mit Ausnahme der Gaubenfirste – an (in Meter).

Roof.DachSummeFlaeche

Deklaration:

double (nur lesbar)

HRESULT get_DachSummeFlaeche(double* retVal);

Siehe auch: Roof Übersicht

Gibt die Gesamtfläche (in m²) des Daches - ohne die Gaubenflächen - an.

Roof.DachSummeGratlaenge

Deklaration:

double (nur lesbar)

HRESULT get_DachSummeGratlaenge(double* retVal);

Siehe auch: Roof Übersicht

Gibt die Summe der Längen (in m) aller Grate - mit Ausnahme der Gaubengrate - an.

Roof.DachSummeKehlenlaenge

Deklaration:

double (nur lesbar)

HRESULT get_DachSummeKehlenlaenge(double* retVal);

Siehe auch: Roof Übersicht

Gibt die Summe aller Kehlenlängen (in m) - mit Ausnahme der Gaubenkehlen - an.

Roof.DachSummeOrtganglaenge

Deklaration:

double (nur lesbar)

HRESULT get_DachSummeOrtganglaenge(double* retVal);

Siehe auch: Roof Übersicht

Gibt die Gesamtlänge aller Ortgänge (in m) - mit Ausnahem der Gaubenortgänge - an.

Roof.DachSummePultdachFirstlaenge

Deklaration:

double (nur lesbar)

HRESULT get_DachSummePultdachFirstlaenge(double* retVal);

Siehe auch: Roof Übersicht

Gibt die Gesamtlänge aller Pultdach-Firste - mit Ausnahme der Gauben-Pultdachfirste – an (in Meter).

Roof.DachSummeTrauflaenge

Deklaration:

double (nur lesbar)

HRESULT get_DachSummeTrauflaenge(double* retVal);

Siehe auch: Roof Übersicht

Gibt die Gesamtlänge (in m) aller Traufen - mit Ausnahme der Gaubentraufen - an.

Roof.FlaecheDerEindeckungUebermessen

Deklaration:

double (nur lesbar)

HRESULT get_FlaecheDerEindeckungUebermessen
(double* retVal);

Siehe auch: Roof Übersicht

Nettofläche des gesamten Daches (in m²).

Roof.Gauben

Deklaration:

GaubenCollection (nur lesbar)

HRESULT get_Gauben(IGaubenCollection** retVal);

Siehe auch: Roof Übersicht, GaubenCollection

Roof.GaubenSummeFirstlaenge

Deklaration:

double (nur lesbar)

HRESULT get_GaubenSummeFirstlaenge(double* retVal);

Siehe auch: Roof Übersicht

Gibt die Summe der Längen (in m) aller Gaubenfirste des Daches an.

Roof.GaubenSummeFlaeche

Deklaration:

double (nur lesbar)

HRESULT get_GaubenSummeFlaeche(double* retVal);

Siehe auch: Roof Übersicht

Gibt die Gesamtfläche (in m²) der Gauben des Daches an.

Roof.GaubenSummeGratlaenge

Deklaration:

double (nur lesbar)

HRESULT get_GaubenSummeGratlaenge(double* retVal);

Siehe auch: Roof Übersicht

Gibt die Summe der Längen aller Grate der Gauben an (in Meter).

Roof.GaubenSummeKehlenlaenge

Deklaration:

double (nur lesbar)

HRESULT get_GaubenSummeKehlenlaenge(double* retVal)
;

Siehe auch: Roof Übersicht

Gibt die Summe aller Kehlenlängen (in m) der Gauben an.

Roof.GaubenSummeOrtganglaenge

Deklaration:

double (nur lesbar)

HRESULT get_GaubenSummeOrtganglaenge(double* retVal);

Siehe auch: Roof Übersicht

Gibt die Gesamtlänge aller Ortgänge der Gauben an (in Meter).

Roof.GaubenSummePultdachFirstlaenge

Deklaration:

double (nur lesbar)

HRESULT get_GaubenSummePultdachFirstlaenge(double* retVal);

Siehe auch: Roof Übersicht

Gibt die Gesamtlänge aller Pultdach-Firste (in m) der Gauben an.

Roof.GaubenSummeTrauflaenge

Deklaration:

double (nur lesbar)

HRESULT get_GaubenSummeTrauflaenge(double* retVal);

Siehe auch: Roof Übersicht

Gibt die Gesamtlänge (in m) aller Traufen der Gauben an.

Roof.History

Deklaration:

IDHistory (nur lesbar)

HRESULT get_History(IIDHistory** retVal);

Siehe auch: Roof Übersicht, IDHistory

Falls dieses Roof beim Ursprünglichen Laden (oder Erstellen) des aktuellen Projektes nicht in diesem Projekt war, sonder aus einem anderen Projekt nachgeladen wurde, liefert diese Eigenschaft das History-Objekt, das die Umsetzung der IDs beschreibt. Andernfalls ist diese Eigenschaft “ Nothing” .

Ein Anwendungsbeispiel für History-Objekte finden Sie im Kapitel Eindeutige Kennungen.

Roof.ID

Deklaration:

long (nur lesbar)

HRESULT get_ID(long* retVal);

Siehe auch: Roof Übersicht

Gibt eine eindeutige Kennung dieses Roof an. Diese Kennung ist persistent, das heißt, sie bleibt innerhalb des aktuellen Projektes erhalten. Sie können diese Kennung benutzen, um Roofs miteinander zu vergleichen oder zusätzliche Daten zum Roof in einer Datenbank abzulegen.

Die ID ist aber nur innerhalb des aktuellen Projektes eindeutig, wenn Sie (durch Nachladen einzelner Gebäude) das Objekt in ein anderes Projekt einfügen, erhält es eine neue ID. In diesem Fall müssen Sie sofort bei Beendigung des Ladevorganges mit Hilfe der Eigenschaft History objPropertyDetailsRoofHistory Ihre zusätzlichen Daten an die neuen Werte anpassen.

Roof.Polygon

Deklaration:

Polygon2D

HRESULT get_Polygon(IPolygon2D** retVal);
HRESULT put_Polygon(IPolygon2D* newVal);

Siehe auch: Roof Übersicht, Polygon2D

Liefert das zweidimensionale Polygon der äußeren Dachumrandung.

Roof.RoofAreas

Deklaration:

RoofAreaCollection (nur lesbar)

HRESULT get_RoofAreas(IRoofAreaCollection** retVal);

Siehe auch: Roof Übersicht, RoofAreaCollection

Roof.Story

Deklaration:

Story (nur lesbar)

HRESULT get_Story(IStory** retVal);

Siehe auch: Roof Übersicht, Story

Liefert das Stockwerk, zu dem das Dach gehört.

Roof.TextureCount

Deklaration:

long (nur lesbar)

HRESULT get_TextureCount(long* retVal);

Siehe auch: Roof Übersicht

Roof.TotalSummeFirstlaenge

Deklaration:

double (nur lesbar)

HRESULT get_TotalSummeFirstlaenge(double* retVal);

Siehe auch: Roof Übersicht

Gibt die Summe der Längen aller Firste (in m) des Daches, inklusive Gaubenfirste, an.

Roof.TotalSummeFlaeche

Deklaration:

double (nur lesbar)

HRESULT get_TotalSummeFlaeche(double* retVal);

Siehe auch: Roof Übersicht

Gibt die Gesamtfläche des Daches (in m²), inklusive Gaubenflächen, an.

Roof.TotalSummeGratlaenge

Deklaration:

double (nur lesbar)

HRESULT get_TotalSummeGratlaenge(double* retVal);

Siehe auch: Roof Übersicht

Gibt die Summe der Längen aller Grate - inklusive Gaubengrate – an (in Meter).

Roof.TotalSummeKehlenlaenge

Deklaration:

double (nur lesbar)

HRESULT get_TotalSummeKehlenlaenge(double* retVal);

Siehe auch: Roof Übersicht

Gibt die Summe allen Kehlenlängen (in m), inklusive Gaubenkehlen, an.

Roof.TotalSummeOrtganglaenge

Deklaration:

double (nur lesbar)

HRESULT get_TotalSummeOrtganglaenge(double* retVal);

Siehe auch: Roof Übersicht

Gibt die Gesamtlänge (in m) aller Ortgänge, inklusive Gaubenortgänge, an.

Roof.TotalSummePultdachFirstlaenge

Deklaration:

double (nur lesbar)

HRESULT get_TotalSummePultdachFirstlaenge(double* retVal);

Siehe auch: Roof Übersicht

Gibt die Gesamtlänge aller Pultdach-Firste - inklusive der Gauben-Pultdachfirste – an (in Meter).

Roof.TotalSummeTrauflaenge

Deklaration:

double (nur lesbar)

HRESULT get_TotalSummeTrauflaenge(double* retVal);

Siehe auch: Roof Übersicht

Gibt die Gesamtlänge (in m) aller Traufen, inklusive den Gaubentraufen, an.

Roof.Windows

Deklaration:

WindowCollection (nur lesbar)

HRESULT get_Windows(IWindowCollection** retVal);

Siehe auch: Roof Übersicht, WindowCollection

Liefert eine Liste von Dachfenstern, die in diesem Dach liegen. Die Einträge der Liste sind vom Typ RoofWindow.

Roof.CreateGaube

Deklaration:

```
Function CreateGaube(ByVal type as long, ByVal contur as
Point2DCollection) as Gaube
```

```
HRESULT CreateGaube(long type, IPoint2DCollection *
contur, IGaube** retVal);
```

Siehe auch: Roof Übersicht, Gaube, Point2DCollection

Roof.Delete

Deklaration:

Sub Delete()

HRESULT Delete();

Siehe auch: Roof Übersicht

Löscht das gesamte Dach.

Roof.Edit

Deklaration:

```
Function Edit() as Boolean
```

```
HRESULT Edit(VARIANT_BOOL* retVal);
```

Siehe auch: Roof Übersicht

Aktiviert den Dacheditor, so als ob der Benutzer das Dach selektiert und dann doppelt angeklickt hätte.

Roof.GetTexture

Deklaration:

```
Function GetTexture(ByVal index as long) as String
```

```
HRESULT GetTexture(long index, BSTR* retVal);
```

Siehe auch: Roof Übersicht

Roof.GetWoodConstruction

Deklaration:

```
Function GetWoodConstruction(ByVal collapse as Boolean,
ByVal epsilon as Single) as RoofWoodCollection
```

```
HRESULT GetWoodConstruction(VARIANT_BOOL collapse,
float epsilon, IRoofWoodCollection** retVal);
```

Siehe auch: Roof Übersicht, RoofWoodCollection

Roof.PlaceWindow

Deklaration:

```
Function PlaceWindow(ByVal aRoofWindow as
RoofWindow, ByVal x as Single, ByVal y as Single) as
Boolean
```

```
HRESULT PlaceWindow(IRoofWindow* aRoofWindow, float
x, float y, VARIANT_BOOL* retVal);
```

Siehe auch: Roof Übersicht, RoofWindow

Roof.SetTexture

Deklaration:

```
Function SetTexture(ByVal index as long, ByVal
TextureName as String) as Boolean
```

```
HRESULT SetTexture(long index, BSTR TextureName,
VARIANT_BOOL* retVal);
```

Siehe auch: Roof Übersicht

Point2D

Ein 2D Punkt. Mit der Hilfe von Punktobjekten können Sie z.B. Polygone konstruieren. Point-Objekte werden auch beim Iterieren durch eine Punktliste geliefert.

Point2D Übersicht

Eigenschaften

x	X-Koordinate
y	Y-Koordinate

Point2D.x

Deklaration:

Single

HRESULT get_x(float* retVal);
HRESULT put_x(float newVal);

Siehe auch: Point2D Übersicht

Die X-Koordinate des Punktes.

Point2D.y

Deklaration:

Single

HRESULT get_y(float* retVal);
HRESULT put_y(float newVal);

Siehe auch: Point2D Übersicht

Die Y-Koordinate des Punktes.

Point2DCollection

Eine Liste von 2D-Punkten. Sie wird verschiedentlich benutzt, um Polygone anzugeben.

Point2DCollection Übersicht

Eigenschaften

Count	Anzahl der Punkte

Methoden

Add	Fügt ein Punkt-Objekt am Ende der Liste an.
AddPoint	Fügt einen Punkt am Ende der Liste an.
Item	Der i-te Punkt

Point2DCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: Point2DCollection Übersicht, Listen

Anzahl der Punkte in der Liste. Eine Zuweisung an diese Eigenschaft ist nicht möglich.

Point2DCollection.Add

Deklaration:

```
Sub Add(ByVal aPoint as Point2D)
```

```
HRESULT Add(IPoint2D* aPoint);
```

Siehe auch: Point2DCollection Übersicht, Point2D, Listen

Fügt einen Punkt in die Liste ein.

Parameter:

aPoint	Der einzufügende Punkt

Point2DCollection.AddPoint

Deklaration:

```
Sub AddPoint(ByVal x as Single, ByVal
y as Single)
```

```
HRESULT AddPoint(float x, float y);
```

Siehe auch: Point2DCollection Übersicht, Listen

Fügt einen Punkt zur Liste hinzu.

Parameter sind:

x, y	Koordinaten des neuen Punktes.

Point2DCollection.Item

Deklaration:

```
Function Item(ByVal Index as long) as Point2D
```

```
HRESULT Item(long Index, IPoint2D** retVal);
```

Siehe auch: Point2DCollection Übersicht, Point2D, Listen

Liefert einen Punkt aus der Liste als Point2D Objekt.

Parameter:

index	Index des gewünschten Punktes. Indizes beginnen bei 1, der maximale Index ist Count

Image

Ein Image-Objekt gehört zu den 2D-Primitiven, durch die Sie eigene grafische Elemente in die 2D Darstellung einbringen können. Andere Grafikelemente sind: Label, Shape, Line und Polygon2D.

Image ist in der Lage, ein Bild (eine Bitmap) im ArCon Konstruktionsmodus darzustellen. Es ähnelt dem Visual Basic “ Image Control” .

Image Übersicht
Eigenschaften

AutoSize	Steuert die automatische Größenanpassung
BorderStyle	Gibt die Art des Rahmens an
FileName	Ein mit dieser Grafik assoziierter Dateiname (keine Funktion in ArCon)
Height	Höhe des Bildes
Left	Die linke Seite des Bildes
Picture	Das dargestellte Bild
Top	Die obere Seite des Bildes
Width	Die Breite des Bildes

Image.AutoSize

Deklaration:

short

HRESULT get_AutoSize(short* retVal);
HRESULT put_AutoSize(short newVal);

Siehe auch: Image Übersicht

Gibt an, wie die Größe des dargestellten Bildes ermittelt bzw. angepaßt wird. Folgende Werte sind möglich:

0	clip, das Bild wird am Rande der vorgebenen Objektgröße abgeschnitten.
1	scale, das Bild wird an die vorgegebene Größe angepaßt (skaliert).
2	original, das Bild wird in Originalgröße angezeigt, das Image-Objekt paßt seine Größe an die der Bitmap an.

Image.BorderStyle

Deklaration:

short

HRESULT get_BorderStyle(short* retVal);
HRESULT put_BorderStyle(short newVal);

Siehe auch: Image Übersicht

Gibt an, ob und welchen Rand das Bild hat. Mögliche Werte sind:

0	none, kein Rand
1	fixed single, ein schmaler Rand

Image.FileName

Deklaration:

String

HRESULT get_FileName(BSTR* retVal);
HRESULT put_FileName(BSTR newVal);

Siehe auch: Image Übersicht

Name der Datei, aus der dieses Bild geladen wurde. Diese Eigenschaft dient nur zur Information bzw. weiteren Verwendung in Ihrem Makro-Programm, sie wird von ArCon nicht verwendet.

Image.Height

Deklaration:

Single

HRESULT get_Height(float* retVal);
HRESULT put_Height(float newVal);

Siehe auch: Image Übersicht

Gibt die Höhe des Objektes in Metern an. Eventuell wird diese Höhe durch den Inhalt des Bildes überschrieben, siehe dazu AutoSize.

Image.Left

Deklaration:

Single

HRESULT get_Left(float* retVal);
HRESULT put_Left(float newVal);

Siehe auch: Image Übersicht

Gibt die Koordinate des linken Randes des Bildes an.

Image.Picture

Deklaration:

VARIANT

HRESULT put_Picture(VARIANT newVal);

Siehe auch: Image Übersicht

Die darzustellende Bitmap des Image-Objektes.

Der Wert dieser Eigenschaft ist ein Objekt vom OLE-Typ “ VARIANT” . Sie erzeugen ein entsprechendes Objekt (bzw. initialisieren es) mit Hilfe der Funktion ArConPictureFromBitmap aus der MakroUtil.DLL.

Image.Top

Deklaration:

Single

HRESULT get_Top(float* retVal);
HRESULT put_Top(float newVal);

Siehe auch: Image Übersicht

Gibt die Koordinate der oberen Ecke des Bildes an.

Image.Width

Deklaration:

Single

HRESULT get_Width(float* retVal);
HRESULT put_Width(float newVal);

Siehe auch: Image Übersicht

Die Breite des Image-Objektes. Falls AutoSize entsprechend gewählt ist, kann dieser Wert durch die tatsächliche Größe der Bitmap überschrieben werden.

Line

Ein Line-Objekt gehört zu den 2D-Primitiven, durch die Sie eigene grafische Elemente in ArCon’ s 2D Darstellung einbringen können. Andere Grafikelemente sind: Label, Shape, Image und Polygon2D.

Line ist in der Lage, eine Linie im ArCon Konstruktionsmodus darzustellen. Es ähnelt dem Visual Basic “ Line Control” .

Line Übersicht
Eigenschaften

BorderColor	Farbe der Linie
BorderStyle	Art der Liniendarstellung
BorderWidth	Dicke der Linie
DrawMode	Zeichenmodus
X1	X-Koordinate des Startpunktes
X2	X-Koordinate des Endpunktes
Y1	Y-Koordinate des Startpunktes
Y2	Y-Koordinate des Endpunktes

Line.BorderColor

Deklaration:

long

HRESULT get_BorderColor(long* retVal);
HRESULT put_BorderColor(long newVal);

Siehe auch: Line Übersicht

RGB Wert der Farbe der Linie.

Line.BorderStyle

Deklaration:

short

HRESULT get_BorderStyle(short* retVal);
HRESULT put_BorderStyle(short newVal);

Siehe auch: Line Übersicht

Gibt die Art der Linienzeichnung an:

0	transparent, unsichtbar
1	solid, durchgezogen
2	dash, gestrichelt
3	dot, gepunktet
4	dash dot, Strich-Punkt
5	dash dot dot, Strich-Punkt-Punkt

Line.BorderWidth

Deklaration:

Single

HRESULT get_BorderWidth(float* retVal);

HRESULT put_BorderWidth(float newVal);

Siehe auch: Line Übersicht

Breit der Linie in Millimetern auf dem Papier. Gestrichelte (u.ä.) Linien (siehe BorderStyle) funktionieren nur mit der Breite 0 (Haarlinie).

Falls Ihre Linien nicht Objekte in der ArCon-Welt beschreiben, sondern sich auf die Bildschirmdarstellung beziehen (z.B. Markierungen für gerade bearbeitete Objekte), benutzen Sie eine negative Breite: der Absolutwert gibt dann die Breite in Bildschirmpunkten (Pixel) an.

Line.DrawMode

Deklaration:

short

HRESULT get_DrawMode(short* retVal);

HRESULT put_DrawMode(short newVal);

Siehe auch: Line Übersicht

Gibt an, wie die Zeichenfarbe ermittelt wird. Meist wird der Wert “ copy pen” (13) benutzt, gelegentlich auch “ XOR Pen” (7).

Einstellung	Beschreibung
1	Blackness (Schwarz).
2	Not Merge Pen (Nicht-Mischen-Stift): Inverse Darstellung der Einstellung 15 (Merge Pen).
3	Mask Not Pen (Maskieren-Nicht-Stift): Kombination der Farben, die der Hintergrund mit der invertierten Stiftfarbe gemeinsam hat.
4	Not Copy Pen (Nicht-Kopieren-Stift): Inverse Darstellung der Einstellung 13 (Copy Pen).
5	Mask Pen Not (Maskieren-Stift-Nicht): Kombination der Farben, die der Stift mit der invertierten Anzeigefarbe gemeinsam hat.
6	Invert (Invers): Inverse Darstellung der Anzeigefarbe.
7	Xor Pen (Xor-Stift): Kombination der Farben, die im Stift und in der Anzeigefarbe, aber nicht in beiden vorhanden sind.
8	Not Mask Pen (Nicht-Maskieren-Stift): Inverse Darstellung der Einstellung 9 (Mask Pen).
9	Mask Pen (Maskieren-Stift): Kombination der Farben, die der Stift mit der Anzeige gemeinsam hat.
10	Not Xor Pen (Nicht-Xor-Stift): Inverse Darstellung der Einstellung 7 (Xor Pen).
11	Nop: Keine Operation, Ausgabe bleibt unverändert, d.h. diese Einstellung schaltet die Ausgabe aus.
12	Merge Not Pen (Mischen-Nicht-Stift): Kombination der Anzeigefarbe und der invertierten Stiftfarbe.
13	Copy Pen (Kopieren-Stift): (Voreinstellung) Farbe, die durch die BorderColor-Eigenschaft angegeben ist.
14	Merge Pen Not (Mischen-Stift-Nicht): Kombination

der Stiftfarbe und der invertierten Anzeigefarbe.

15 Merge Pen (Mischen-Stift): Kombination der Stiftfarbe und der Anzeigefarbe.

16 Whiteness (Weiß).

Line.X1

Deklaration:

Single

HRESULT get_X1(float* retVal);
HRESULT put_X1(float newVal);

Siehe auch: Line Übersicht

X-Koordinate das Startpunktes der Linie. Durch Zuweisen an diese Eigenschaft verschieben Sie die Linie.

Line.X2

Deklaration:

Single

HRESULT get_X2(float* retVal);
HRESULT put_X2(float newVal);

Siehe auch: Line Übersicht

X-Koordinate des Endpunktes der Linie.

Line.Y1

Deklaration:

Single

HRESULT get_Y1(float* retVal);
HRESULT put_Y1(float newVal);

Siehe auch: Line Übersicht

Y-Koordinate des Startpunktes der Linie.

Line.Y2

Deklaration:

Single

HRESULT get_Y2(float* retVal);
HRESULT put_Y2(float newVal);

Siehe auch: Line Übersicht

Y-Koordinate des Linienendpunktes.

Shape

Ein Shape-Objekt gehört zu den 2D-Primitiven, durch die Sie eigene grafische Elemente in die 2D Darstellung einbringen können. Andere Grafikelemente sind: Label, Line, Image und Polygon2D.

Shape ist in der Lage, verschiedene geometrische Grundformen im ArCon Konstruktionsmodus darzustellen. Es ähnelt dem Visual Basic “ Shape Control” .

Shape Übersicht
Eigenschaften

Angle	Rotationswinkel der Figur in Bogenmaß
BackColor	Hintergrundfarbe der Figur
BackStyle	Sichtbarkeit des Hintergrundes
BorderColor	Farbe der Linie
BorderStyle	Art der Linienzeichnung
BorderWidth	Dicke der Linie
DrawMode	Zeichenmodus
FillColor	Farbe innerhalb der Figur
FillStyle	Art der inneren Fläche
Height	Höhe der Figur
Left	Die linke Seite der Figur
Shape	Welche Form hat die Figur?
Top	Obere Seite der Figur
Width	Breite der Figur

Shape.Angle

Deklaration:

Single

HRESULT get_Angle(float* retVal);
HRESULT put_Angle(float newVal);

Siehe auch: Shape Übersicht

Falls es sich bei dieser Figur um ein Rechteck oder ein Qudrat handelt, wird es um den angegebenen Winkel gedreht dargestellt. Der Winkel wird im mathematischen Drehsinn (positiv = gegen den Uhrzeigersinn) und in Bogenmaß angegeben.

Shape.BackColor

Deklaration:

long

HRESULT get_BackColor(long* retVal);
HRESULT put_BackColor(long newVal);

Siehe auch: Shape Übersicht

RGB Wert der Hintergrundfarbe. Diese Farbe wird nur sichtbar, wenn der entsprechende Hintergrunstil gewählt wird.

Shape.BackStyle

Deklaration:

short

HRESULT get_BackStyle(short* retVal);
HRESULT put_BackStyle(short newVal);

Siehe auch: Shape Übersicht

Gibt an, ob der Hintergrund durchsichtig ist oder mit der Hintergrundfarbe gefüllt wird.

Mögliche Werte sind:

0	transparent, durchsichtig
1	opaque, in der Hintergrundfarbe gefüllt

Shape.BorderColor

Deklaration:

long

HRESULT get_BorderColor(long* retVal);

HRESULT put_BorderColor(long newVal);

Siehe auch: Shape Übersicht

RGB Wert der Farbe, mit der der Rand der Figur gezeichnet wird. Siehe auch BorderStyle.

Shape.BorderStyle

Deklaration:

short

HRESULT get_BorderStyle(short* retVal);

HRESULT put_BorderStyle(short newVal);

Siehe auch: Shape Übersicht

Gibt die Art des Randes der Figur an:

0	transparent, unsichtbar
1	solid, durchgezogen
2	dash, gestrichelt
3	dot, gepunktet
4	dash dot, Strich-Punkt
5	dash dot dot, Strich-Punkt-Punkt

Shape.BorderWidth

Deklaration:

Single

HRESULT get_BorderWidth(float* retVal);

HRESULT put_BorderWidth(float newVal);

Siehe auch: Shape Übersicht

Breite der Linie in Millimetern auf dem Papier. Gestrichelte (u.ä.) Linien (siehe BorderStyle) funktionieren nur mit der Breite 0 (Haarlinie).

Falls Ihr Shape sich nicht auf Inhalte der ArCon-Welt, sondern auf den Bildschirm bezieht (z.B. als Markierung für ein gerade bearbeitetes Objekt) können Sie die Breite der Linie in Bildschirmpunkten (Pixel) angeben, indem Sie die gewünschte Pixelbreite als negativen Wert dieser Eigenschaft zuweisen.

Shape.DrawMode

Deklaration:

short

HRESULT get_DrawMode(short* retVal);

HRESULT put_DrawMode(short newVal);

Siehe auch: Shape Übersicht

Gibt an, wie die Zeichenfarbe ermittelt wird. Meist wird der Wert “ copy pen” (13) benutzt, gelegentlich auch “ XOR Pen” (7).

Wert	Beschreibung
1	Blackness (Schwarz).
2	Not Merge Pen (Nicht-Mischen-Stift): Inverse Darstellung der Einstellung 15 (Merge Pen).
3	Mask Not Pen (Maskieren-Nicht-Stift): Kombination der Farben, die der Hintergrund mit der invertierten Stiftfarbe gemeinsam hat.
4	Not Copy Pen (Nicht-Kopieren-Stift): Inverse Darstellung der Einstellung 13 (Copy Pen).
5	Mask Pen Not (Maskieren-Stift-Nicht): Kombination der Farben, die der Stift mit der invertierten Anzeigefarbe gemeinsam hat.
6	Invert (Invers): Inverse Darstellung der Anzeigefarbe.
7	Xor Pen (Xor-Stift): Kombination der Farben, die im Stift und in der Anzeigefarbe, aber nicht in beiden vorhanden sind.
8	Not Mask Pen (Nicht-Maskieren-Stift): Inverse Darstellung der Einstellung 9 (Mask Pen).
9	Mask Pen (Maskieren-Stift): Kombination der Farben, die der Stift mit der Anzeige gemeinsam hat.
10	Not Xor Pen (Nicht-Xor-Stift): Inverse Darstellung der Einstellung 7 (Xor Pen).
11	Nop: Keine Operation, Ausgabe bleibt unverändert, d.h. diese Einstellung schaltet die Ausgabe aus.
12	Merge Not Pen (Mischen-Nicht-Stift): Kombination der Anzeigefarbe und der invertierten Stiftfarbe.
13	Copy Pen (Kopieren-Stift): (Voreinstellung) Farbe, die durch die BorderColor-Eigenschaft angegeben ist.
14	Merge Pen Not (Mischen-Stift-Nicht): Kombination

der Stiftfarbe und der invertierten Anzeigefarbe.

15 Merge Pen (Mischen-Stift): Kombination der Stiftfarbe und der Anzeigefarbe.

16 Whiteness (Weiß).

Shape.FillColor

Deklaration:

long

HRESULT get_FillColor(long* retVal);
HRESULT put_FillColor(long newVal);

Siehe auch: Shape Übersicht

RGB-Wert der Farbe, mit der das Innere der Figur gezeichnet wird, falls der Füllstil entsprechend gesetzt ist.

Shape.FillStyle

Deklaration:

short

HRESULT get_FillStyle(short* retVal);
HRESULT put_FillStyle(short newVal);

Siehe auch: Shape Übersicht

Gibt an, wie das Innere der Figur gezeichnet wird. Die zugehörige Farbe wird durch FillColor bestimmt.

Mögliche Werte sind:

0	solid, vollständig gefüllt
1	transparent, nicht gefüllt
2	Horizontal Line (horizontale Linien)
3	Vertical Line (vertikale Linien)
4	Downward Diagonal (Abwärtsdiagonal)
5	Upward Diagonal (Aufwärtsdiagonal)
6	Cross (senkrecht gekreuzt)
7	Diagonal Cross (diagonal gekreuzt)

Shape.Height

Deklaration:

Single

HRESULT get_Height(float* retVal);
HRESULT put_Height(float newVal);

Siehe auch: Shape Übersicht

Die Höhe der Figur.

Shape.Left

Deklaration:

Single

HRESULT get_Left(float* retVal);
HRESULT put_Left(float newVal);

Siehe auch: Shape Übersicht

Die X-Koordinate der linken Seite der Figur.

Shape.Shape

Deklaration:

short

HRESULT get_Shape(short* retVal);
HRESULT put_Shape(short newVal);

Siehe auch: Shape Übersicht

Gibt die Art der Figur an:

0	rectangle, Rechteck
1	square, Quadrat
2	oval, Oval
3	circle, Kreis
4	Rounded Rectangle, Rechteck mit abgerundeten Ecken
5	Rounded Square, Quadrat mit abgerundeten Ecken

Shape.Top

Deklaration:

Single

HRESULT get_Top(float* retVal);
HRESULT put_Top(float newVal);

Siehe auch: Shape Übersicht

Die Y-Koordinate der oberen Ecke der Figur.

Shape.Width

Deklaration:

Single

HRESULT get_Width(float* retVal);
HRESULT put_Width(float newVal);

Siehe auch: Shape Übersicht

Breite der Figur.

Label

Ein Label-Objekt gehört zu den 2D-Primitiven, durch die Sie eigene grafische Elemente in die 2D Darstellung einbringen können. Andere Grafikelemente sind: Shape, Line, Image und Polygon2D.

Label ist in der Lage, einen Text, z.B. eine Beschriftung, darzustellen. Es ähnelt dem Visual Basic “ Label Control” .

Label Übersicht
Eigenschaften

Alignment	Ausrichtung des Textes
Angle	Rotationswinkel des Textes in Bogenmaß
AutoSize	Automatische Größenanpassung an den Text
BackColor	Hintergrundfarbe
BackStyle	Art der Hintergrundzeichnung
BorderStyle	Art des Randes
Caption	Textinhalt
Font	Schriftart
ForeColor	Schriftfarbe
Height	Höhe des Textes
Left	Die linke Seite des Textes
Top	Obere Seite des Textes
Width	Breite des Textes
WordWrap	Erlaubt den automatischen Zeilenumbruch

Label.Alignment

Deklaration:

short

HRESULT get_Alignment(short* retVal);
HRESULT put_Alignment(short newVal);

Siehe auch: Label Übersicht

Gibt die Ausrichtung des eigentlichen Textinhaltes innerhalb des Rechteckes ein, den das gesamte Objekt einnimmt.

Mögliche Werte sind:

0	left, der Text wird linksbündig dargestellt
1	right, der Text wird am rechten Rand ausgerichtet
2	center, der Text wird zentriert

Label.Angle

Deklaration:

Single

HRESULT get_Angle(float* retVal);
HRESULT put_Angle(float newVal);

Siehe auch: Label Übersicht

Gibt den Winkel an, um den der Text gedreht werden soll. Der Winkel ist im mathematiscen Drehsinn (positiv = gegen den Uhrzeigersinn) und in Bogenmaß.

Achtung: nur TrueType Schriften können gedreht werden!

Label.AutoSize

Deklaration:

Boolean

HRESULT get_AutoSize
(VARIANT_BOOL* retVal);

HRESULT put_AutoSize
(VARIANT_BOOL newVal);

Siehe auch: Label Übersicht

Wenn diese Eigenschaft WAHR ist, wird das Label-Objekt automatisch vergrößert, wenn der Text nicht mehr in den angegebenen Bereich paßt. Sie können die tatsächlich eingenommene Größe mit den Eigenschaften Width und Height erfragen.

Label.BackColor

Deklaration:

long

HRESULT get_BackColor(long* retVal);
HRESULT put_BackColor(long newVal);

Siehe auch: Label Übersicht

RGB-Wert der Hintergrundfarbe. Diese Farbe wird nur sichtbar, wenn BackStyle den Wert “ opaque” (1) hat.

Label.BackStyle

Deklaration:

short

HRESULT get_BackStyle(short* retVal);

HRESULT put_BackStyle(short newVal);

Siehe auch: Label Übersicht

Gibt an, ob der Texthintergrund sichtbar ist:

0	transparent, der Text erscheint direkt auf dem ArCon Hintergrund (oder unter dem Label-Objekt liegenden anderen Grafikobjekten)
1	opaque, der Texthintergrund wird gezeichnet.

Label.BorderStyle

Deklaration:

short

HRESULT get_BorderStyle(short* retVal);

HRESULT put_BorderStyle(short newVal);

Siehe auch: Label Übersicht

Gibt an, ob das Textfeld umrandet ist.

Mögliche Werte sind:

0	none, kein Rand
1	fixed single, einfacher Rand

Label.Caption

Deklaration:

String

HRESULT get_Caption(BSTR* retVal);
HRESULT put_Caption(BSTR newVal);

Siehe auch: Label Übersicht

Der vom Label-Objekt dargestellte Text.

Label.Font

Deklaration:

VARIANT

HRESULT get_Font(VARIANT* retVal);
HRESULT put_Font(VARIANT newVal);

Siehe auch: Label Übersicht

Schriftart, -größe und Farbe des dargestellten Textes.

Das hier übergebene “ VARIANT” Objekt muß einen OLE-Font (Interface IFont oder IFontDisp, bzw. in Basic StdFont) enthalten.

Label.ForeColor

Deklaration:

long

HRESULT get_ForeColor(long* retVal);
HRESULT put_ForeColor(long newVal);

Siehe auch: Label Übersicht

Die Schriftfarbe.

Label.Height

Deklaration:

Single

HRESULT get_Height(float* retVal);
HRESULT put_Height(float newVal);

Siehe auch: Label Übersicht

Höhe des Objektes. Falls AutoSize WAHR ist, wird eine Zuweisung an diese Eigenschaft ignoriert.

Label.Left

Deklaration:

Single

HRESULT get_Left(float* retVal);
HRESULT put_Left(float newVal);

Siehe auch: Label Übersicht

Die Koordinate der linken Kante des Objektes.

Label.Top

Deklaration:

Single

HRESULT get_Top(float* retVal);
HRESULT put_Top(float newVal);

Siehe auch: Label Übersicht

Die y-Koordinate der oberen Grenze des Textes.

Label.Width

Deklaration:

Single

HRESULT get_Width(float* retVal);
HRESULT put_Width(float newVal);

Siehe auch: Label Übersicht

Die Breite des Objektes. Falls AutoSize WAHR ist, hat eine Zuweisung an diese Eigenschaft keine Wirkung.

Label.WordWrap

Deklaration:

Boolean

```
HRESULT get_WordWrap
(VARIANT_BOOL* retVal);
HRESULT put_WordWrap(VARIANT_BOOL
newVal);
```

Siehe auch: Label Übersicht

Gibt an, ob der Text automatisch umgebrochen kann, sofern er nicht in eine Zeile paßt und genügend Platz für weitere Zeilen innerhalb der Objektmaße ist.

Graphics2DCollection

Eine Liste von Grafikelementen.
Näheres über die Programmierung mit Listen erfahren Sie im Kapitel Listen.

Graphics2DCollection Übersicht

Eigenschaften

Count	Anzahl der 2D Objekte

Methoden

Add	Fügt ein Objet hinzu
EnableUpdate	Schaltet die Aktualisierung aller Grafikelemente ein oder aus
Item	Das i-te Graphikelement
Remove	Entfernt ein Objekt
RemoveAll	Entfernt alle Elemente.
Sort	Sortiert die Elemente in Auf- oder Absteigender Z-Sortierung

Graphics2DCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: Graphics2DCollection Übersicht, Listen

Anzahl der Elemente in der Liste. Eine Zuweisung an diese Eigenschaft ist nicht möglich.

Graphics2DCollection.Add

Deklaration:

```
Function Add(ByVal gr2DObj as Object) as Boolean
```

```
HRESULT Add(IDispatch* gr2DObj, VARIANT_BOOL*
retVal);
```

Siehe auch: Graphics2DCollection Übersicht, Listen

Fügt ein Objekt der Liste hinzu. Das übergebene Objekt muß vom Typ Label, Shape, Line, Polygon2D oder Image sein.

Rückgabewert ist “ Wahr” , wenn das Objekt der Liste hinzugefügt werden konnte.

Die Parameter sind:

gr2Dobj	Das 2D Grafikelement

Graphics2DCollection.EnableUpdate

Deklaration:

```
Sub EnableUpdate(ByVal enabled as Boolean)
```

```
HRESULT EnableUpdate(VARIANT_BOOL
enabled);
```

Siehe auch: Graphics2DCollection Übersicht, Listen

Schaltet die Graphikaktualisierung für alle Elemente der Liste an oder ab

Diese globale An/Abschaltung hat Vorrang vor der einzelnen An-/Abschaltung des einzelnen Graphikelementes.

Parameter:

enabled	Wenn dieser Wert Falsch ist, werden keine Graphikaktualisierungen mehr durchgeführt. Wenn der Wert Wahr ist, wird die Graphik wieder aktualisiert. Beim Einschalten der Aktualisierung werden alle Objekte auf jeden Fall einmal neu gezeichnet, so als ob die Refresh Methode der einzelnen Elemente aufgerufen worden wäre.

Graphics2DCollection.Item

Deklaration:

```
Function Item(ByVal Index as long) as Object
```

```
HRESULT Item(long Index, IDispatch** retVal);
```

Siehe auch: Graphics2DCollection Übersicht, Listen

Liefert ein Grafikelement aus der Liste.

Parameter:

Index	Index des gewünschten Grafikelements. Indizes beginnen bei 1, der maximale Index ist Count

Graphics2DCollection.Remove

Deklaration:

```
Function Remove(ByVal gr2DObj as
Object) as Boolean
```

```
HRESULT Remove(IDispatch*
gr2DObj, VARIANT_BOOL* retVal);
```

Siehe auch: Graphics2DCollection Übersicht, Listen

Entfernt ein Objekt aus der Liste.

Graphics2DCollection.RemoveAll

Deklaration:

```
Sub RemoveAll()
```

```
HRESULT RemoveAll();
```

Siehe auch: Graphics2DCollection Übersicht, Listen

Graphics2DCollection.Sort

Deklaration:

```
Sub Sort(ByVal increasingZOrder as
Boolean)
```

```
HRESULT Sort(VARIANT_BOOL
increasingZOrder);
```

Siehe auch: Graphics2DCollection Übersicht, Listen

Sortiert die Grafikelemente in dieser Liste in aufsteigender oder absteigender Z-Reihenfolge.

Parameter ist:

increasingZOrder	Wahr, wenn die Liste von hinten nach vorne sortiert werden soll, Falsch, wenn von vorne nach hinten sortiert wird.

ToolData

Es gibt genau ein Objekt des Typs ToolData - das RunningTool des ArCon Objektes. Dieses Objekt ist nur gültig, während ein von Ihnen definierter automatischer Variantenschalter aktiv ist. Ein Beispiel zur Anwendung dieses Objektes finden Sie im Kapitel Automatische Schaltflächen.

ToolData Übersicht

Eigenschaften

Graphics2D	Liste der 2D-Elemente für Tool
Points	Liste der eingebenden Punkte des Tools

Methoden

Abort	Tooleingabe abbrechen
AddSnapPoint	Registriert einen Fangpunkt und Richtung
Continue	Fortfahren mit Tooleingabe
Finish	Tooleingabe beenden

ToolData.Graphics2D

Deklaration:

Graphics2DCollection (nur lesbar)

HRESULT get_Graphics2D(IGraphics2DCollection** retVal);

Siehe auch: ToolData Übersicht, Graphics2DCollection

Liefert ein Objekt vom Typ Graphics2DCollection, das die temporären (während das Tool läuft und noch kein permanentes Ergebnis produziert hat) Grafikobjekte enthält. Manipulieren Sie die Inhalte dieser Liste, um die Zeichnung Ihres Tools an Interaktionen des Benutzers anzupassen.

ToolData.Points

Deklaration:

Point2DCollection (nur lesbar)

HRESULT get_Points(IPoint2DCollection** retVal);

Siehe auch: ToolData Übersicht, Point2DCollection

Eine Punktliste vom Typ Point2DCollection, das Punkte während des Laufs eines Wie-Werkzeugs aufsammelt.

ToolData.Abort

Deklaration:

```
Sub Abort()
```

```
HRESULT Abort();
```

Siehe auch: ToolData Übersicht

Bricht das gerade laufende Werkzeug nach dem nächsten Schritt ab.

ToolData.AddSnapPoint

Deklaration:

```
Sub AddSnapPoint(ByVal x as Single, ByVal y as Single,
ByVal dirX as Single, ByVal dirY as Single, ByVal distance
as Single, ByVal isOnVertex as Boolean)
```

```
HRESULT AddSnapPoint(float x, float y, float dirX, float dirY,
float distance, VARIANT_BOOL isOnVertex);
```

Siehe auch: ToolData Übersicht

Fügt einen Fangpunkt in die Liste der aktuellen Fangpunkte ein. Nur möglich innerhalb eines HowSnap Ereignisses.

Parameter sind:

x, y	Die Koordinaten des Fangpunktes
dirX, dirY	normalisierter Richtungsvektor der Tangente, falls dieser Fangpunkt eine Richtung definiert. Ansonsten 0/0.
distance	Der Abstand des Fangpunktes von der Mausposition.
isOnVertex	Wahr, wenn es sich um einen Eckpunkt handelt

ArCon sortiert Fangpunkte nach zunehmender Entfernung und bevorzugt Eckpunkte.

ToolData.Continue

Deklaration:

```
Sub Continue()
```

```
HRESULT Continue();
```

Siehe auch: ToolData Übersicht

Führt das gerade laufende Werkzeug nach dem nächsten Schritt weiter aus.

Alternativen: Finish, Abort

ToolData.Finish

Deklaration:

```
Sub Finish()
```

```
HRESULT Finish();
```

Siehe auch: ToolData Übersicht

Beendet das gerade laufende Werkzeug nach dem nächsten Schritt.

Alternativen: Abort, Continue

BuildingCollection

Eine Liste von Gebäuden.
Näheres über die Programmierung mit Listen erfahren Sie im Kapitel Listen.

BuildingCollection Übersicht

Eigenschaften

Count	Anzahl der Gebäude

Methoden

Item	Das i-te Gebäude

BuildingCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: BuildingCollection Übersicht, Listen

Anzahl der Elemente in der Liste. Eine Zuweisung an diese Eigenschaft ist nicht möglich.

BuildingCollection.Item

Deklaration:

```
Function Item(ByVal Index as long) as Building
```

```
HRESULT Item(long Index, IBuilding** retVal);
```

Siehe auch: BuildingCollection Übersicht, Building, Listen

Liefert ein Gebäude aus der Liste.

Parameter:

Index	Index des gewünschten Gebäudes. Indizes beginnen bei 1, der maximale Index ist Count

StoryCollection

Eine Liste von Geschossen.
Näheres über die Programmierung mit Listen erfahren Sie im Kapitel Listen.

StoryCollection Übersicht

Eigenschaften

Count	Anzahl der Stockwerke

Methoden

Item	Das i-te Stockwerk

StoryCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: StoryCollection Übersicht, Listen

Anzahl der Elemente in der Liste. Eine Zuweisung an diese Eigenschaft ist nicht möglich.

StoryCollection.Item

Deklaration:

```
Function Item(ByVal Index as long) as Story
```

```
HRESULT Item(long Index, IStory** retVal);
```

Siehe auch: StoryCollection Übersicht, Story, Listen

Liefert ein Geschoß aus der Liste.

Parameter:

Index	Index des gewünschten Stockwerks. Indizes beginnen bei 1, der maximale Index ist Count

WallCollection

Eine Liste von Wänden.
Näheres über die Programmierung mit Listen erfahren Sie im Kapitel Listen.

WallCollection Übersicht

Eigenschaften

Count	Anzahl der Wände

Methoden

Item	Die i-te Wand

WallCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: WallCollection Übersicht, Listen

Anzahl der Elemente in der Liste. Eine Zuweisung an diese Eigenschaft ist nicht möglich.

WallCollection.Item

Deklaration:

```
Function Item(ByVal Index as long) as Wall
```

```
HRESULT Item(long Index, IWall** retVal);
```

Siehe auch: WallCollection Übersicht, Wall, Listen

Liefert eine Wand aus der Liste.

Parameter:

Index	Index der gewünschten Wand. Indizes beginnen bei 1, der maximale Index ist Count

VirtualWallCollection

VirtualWallCollection Übersicht

Eigenschaften

Count	Anzahl der virtuellen Wände

Methoden

Item	Die i-te virtuelle Wand

VirtualWallCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: VirtualWallCollection Übersicht, Listen

VirtualWallCollection.Item

Deklaration:

```
Function Item(ByVal Index as long) as VirtualWall
```

```
HRESULT Item(long Index, IVirtualWall** retVal);
```

Siehe auch: VirtualWallCollection Übersicht, VirtualWall, Listen

RoomCollection

Eine Liste von Räumen.
Näheres über die Programmierung mit Listen erfahren Sie im Kapitel Listen.

RoomCollection Übersicht

Eigenschaften

Count	Anzahl der Räume

Methoden

Item	Der i-te Raum

RoomCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: RoomCollection Übersicht, Listen

Anzahl der Elemente in der Liste. Eine Zuweisung an diese Eigenschaft ist nicht möglich.

RoomCollection.Item

Deklaration:

```
Function Item(ByVal Index as long) as Room
```

```
HRESULT Item(long Index, IRoom** retVal);
```

Siehe auch: RoomCollection Übersicht, Room, Listen

Liefert einen Raum aus der Liste.

Parameter:

Index	Index des gewünschten Raumes. Indizes beginnen bei 1, der maximale Index ist Count

ConturCollection

Eine Liste von Konturen.
Näheres über die Programmierung mit Listen erfahren Sie im Kapitel Listen.

ConturCollection Übersicht

Eigenschaften

Count	Anzahl der Konturen ('Raumwände')

Methoden

Item	Die i-te Kontur

ConturCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: ConturCollection Übersicht, Listen

Anzahl der Elemente in der Liste. Eine Zuweisung an diese Eigenschaft ist nicht möglich.

ConturCollection.Item

Deklaration:

```
Function Item(ByVal Index as long) as Contur
```

```
HRESULT Item(long Index, IContur** retVal);
```

Siehe auch: ConturCollection Übersicht, Contur, Listen

Liefert eine Kontur aus der Liste.

Parameter:

Index	Index der gewünschten Kontur. Indizes beginnen bei 1, der maximale Index ist Count

Contur

Eine Kontur faßt die inneren oder äußeren Wandseiten eines Raumes zu einem geschlossenen Streckenzug zusammen. Damit können Sie die entsprechend zusammengehörenden Wandseiten eines Raumes ermitteln.

Contur Übersicht
Eigenschaften

Inner	Ist dies die innere Kontur des Raumes oder die äußere?
Room	Der Raum, zu dem die Kontur gehört
WallSegments	Liste der Wandseiten dieser Kontur

Contur.Inner

Deklaration:

Boolean (nur lesbar)

HRESULT get_Inner(VARIANT_BOOL* retVal);

Siehe auch: Contur Übersicht

“ Wahr” , wenn es sich bei dieser Kontur um die innere Kontur eines Raumes handelt. “ Falsch” , wenn es die äußere Kontur ist.

Contur.Room

Deklaration:

Room (nur lesbar)

HRESULT get_Room(IRoom** retVal);

Siehe auch: Contur Übersicht, Room

Verweis auf den Raum dessen Umrandung von dieser Kontur beschrieben wird.

Contur.WallSegments

Deklaration:

WallSegmentCollection (nur lesbar)

HRESULT get_WallSegments(IWallSegmentCollection** retVal);

Siehe auch: Contur Übersicht, WallSegmentCollection

Eine Liste mit Wandsegmenten (Wandseiten), die diese Kontur bilden.

WallSegment

Ein Wandsegment ist Teil einer Wand. Durch Wandverschneidungen und natürliche Ecken wird die Oberfläche einer Wand unterteilt. Sie können niemals ein Wandsegment selbst erzeugen. Mit jeder neuen Wand oder Änderung an der Geometrie einer Wand werden alle vorher existierenden Wandsegmente vernichtet und eventuell neu berechnet. Sie sollten daher niemals Verweise auf Wandsegmente längerfristig speichern.

WallSegment Übersicht

Eigenschaften

Area	Fläche der Wandseite
AreaFormula	Formel für Fläche der Wandseite
Contur	Kontur, zu der diese Wandseite gehört
Doors	Türen, die in dieser Wandseite 'sitzen'
History	Daten zur Identifikation im ehemaligen Projekt für nachgeladene Gebäude
Holes	Aussparungen
ID	Eine eindeutige Kennung
LambdaHs	Höhenpunkte dieses Wandsegmentes
MaterialFromBelow	Für Außenwände: Material von darunterliegender Wandseite übernehmen?
Room	Raum, zu dem diese Wandseite gehört
Texture	Verwendete Textur
VisMode	Sichtbar, unsichtbar, da von Dach abgeschnitten...
Visible	In 3D-Darstellung sichtbar?
Wall	Wand, zu der diese Wandseite gehört
WallSide	Erste, zweite, dritte oder vierte Seite der Wand?
Windows	Fenster, die in dieser Wandseite 'sitzen'
zBottom	Höhe der Wandseitenunterkante
zTop	Höhe der Wandseitenoberkante

Methoden

AddHolePolygon	Schneidet ein Loch in das Wandsegment
GetLineColor	Erfragt die Linienfarbe der 2D Darstellung
GetPos	Position der Wandseite lesen
RemoveHolePolygon	Entfernt ein Loch aus dem Wandsegment
SetLineColor	Ändert die Linienfarbe der 2D Darstellung

WallSegment.Area

Deklaration:

Single (nur lesbar)

HRESULT get_Area(float* retVal);

Siehe auch: WallSegment Übersicht

Die Fläche des Wandsegmentes.

WallSegment.AreaFormula

Deklaration:

String (nur lesbar)

HRESULT get_AreaFormula(BSTR* retVal);

Siehe auch: WallSegment Übersicht

Eine symbolische Formel für die Segmentfläche.

WallSegment.Contur

Deklaration:

Contur (nur lesbar)

HRESULT get_Contur(IContur** retVal);

Siehe auch: WallSegment Übersicht, Contur

Referenz auf die Kontur, zu der dieses Segment gehört, oder “ Nothing” , falls es in keine Kontur integriert ist.

WallSegment.Doors

Deklaration:

DoorCollection (nur lesbar)

HRESULT get_Doors(IDoorCollection** retVal);

Siehe auch: WallSegment Übersicht, DoorCollection

Eine Liste aller Türen in diesem Segment.

WallSegment.History

Deklaration:

IDHistory (nur lesbar)

HRESULT get_History(IIDHistory** retVal);

Siehe auch: WallSegment Übersicht, IDHistory

Falls dieses WallSegment beim Ursprünglichen Laden (oder Erstellen) des aktuellen Projektes nicht in diesem Projekt war, sonder aus einem anderen Projekt nachgeladen wurde, liefert diese Eigenschaft das History-Objekt, das die Umsetzung der IDs beschreibt. Andernfalls ist diese Eigenschaft “ Nothing” .

Ein Anwendungsbeispiel für History-Objekte finden Sie im Kapitel Eindeutige Kennungen.

WallSegment.Holes

Deklaration:

HoleCollection (nur lesbar)

HRESULT get_Holes(IHoleCollection** retVal);

Siehe auch: WallSegment Übersicht, HoleCollection

Eine Liste aller Aussparungen in diesem Segment.

WallSegment.ID

Deklaration:

long (nur lesbar)

HRESULT get_ID(long* retVal);

Siehe auch: WallSegment Übersicht

Gibt eine eindeutige Kennung dieses WallSegment an. Diese Kennung ist persistent, das heißt, sie bleibt innerhalb des aktuellen Projektes erhalten. Sie können diese Kennung benutzen, um WallSegments miteinander zu vergleichen oder zusätzliche Daten zum WallSegment in einer Datenbank abzulegen.

Die ID ist aber nur innerhalb des aktuellen Projektes eindeutig, wenn Sie (durch Nachladen einzelner Gebäude) das Objekt in ein anderes Projekt einfügen, erhält es eine neue ID. In diesem Fall müssen Sie sofort bei Beendigung des Ladevorganges mit Hilfe der Eigenschaft History objPropertyDetailsWallSegmentHistory Ihre zusätzlichen Daten an die neuen Werte anpassen.

WallSegment.LambdaHs

Deklaration:

LambdaHCollection (nur lesbar)

HRESULT get_LambdaHs(ILambdaHCollection** retVal);

Siehe auch: WallSegment Übersicht, LambdaHCollection

Eine Liste von Höhenpunkten.

WallSegment.MaterialFromBelow

Deklaration:

Boolean

```
HRESULT get_MaterialFromBelow(VARIANT_BOOL* retVal)
;
HRESULT put_MaterialFromBelow(VARIANT_BOOL
newVal);
```

Siehe auch: WallSegment Übersicht

Wenn diese Eigenschaft “ Wahr” ist und es sich um eine Außenwand handelt, übernimmt sie das Material von der darunterliegenden Wand.

WallSegment.Room

Deklaration:

Room (nur lesbar)

HRESULT get_Room(IRoom** retVal);

Siehe auch: WallSegment Übersicht, Room

Ein Verweis auf den Raum, zu dem diese Wandseite zeigt. Bei Außenwänden “ Nothing” bei einem nach außen zeigenden Segment.

WallSegment.Texture

Deklaration:

Texture (nur lesbar)

HRESULT get_Texture(ITexture** retVal);

Siehe auch: WallSegment Übersicht, Texture

Die Textur dieses Wandsegmentes.

WallSegment.VisMode

Deklaration:

long (nur lesbar)

HRESULT get_VisMode(long* retVal);

Siehe auch: WallSegment Übersicht

Gibt Detailinformation über den Verschneidevorgang an, durch den dieses Segment entstanden ist.

WallSegment.Visible

Deklaration:

Boolean

HRESULT get_Visible(VARIANT_BOOL* retVal);
HRESULT put_Visible(VARIANT_BOOL newVal);

Siehe auch: WallSegment Übersicht

“ Wahr” , wenn dieses Segment sichtbar ist. Unsichtbare Segmente sind z.B. Wandenden, die in einer anderen Wand stecken.

WallSegment.Wall

Deklaration:

Wall (nur lesbar)

HRESULT get_Wall(IWall** retVal);

Siehe auch: WallSegment Übersicht, Wall

Die Wand, deren Bestandteil dieses Segment ist.

WallSegment.WallSide

Deklaration:

long (nur lesbar)

HRESULT get_WallSide(long* retVal);

Siehe auch: WallSegment Übersicht

Gibt die Nummer der Wandseite an, zu der dieses Segment gehört (mehrere Segmente können in einer Wandseite liegen).

WallSegment.Windows

Deklaration:

WindowCollection (nur lesbar)

HRESULT get_Windows(IWindowCollection** retVal);

Siehe auch: WallSegment Übersicht, WindowCollection

Eine Liste von Fenstern in diesem Segment.

WallSegment.zBottom

Deklaration:

Single (nur lesbar)

HRESULT get_zBottom(float* retVal);

Siehe auch: WallSegment Übersicht

Höhe der Wandseitenunterkante.

WallSegment.zTop

Deklaration:

Single (nur lesbar)

HRESULT get_zTop(float* retVal);

Siehe auch: WallSegment Übersicht

Höhe der Wandseitenoberkante.

WallSegment.AddHolePolygon

Deklaration:

```
Function AddHolePolygon(ByVal thePolygon as
HolePolygon) as Boolean
```

```
HRESULT AddHolePolygon(IHolePolygon* thePolygon,
VARIANT_BOOL* retVal);
```

Siehe auch: WallSegment Übersicht, HolePolygon

WallSegment.GetLineColor

Deklaration:

```
Function GetLineColor() as long
```

```
HRESULT GetLineColor(long* retVal);
```

Siehe auch: WallSegment Übersicht

WallSegment.GetPos

Deklaration:

```
Function GetPos(ByRef X1 as Single, ByRef Y1 as Single,
ByRef X2 as Single, ByRef Y2 as Single) as Boolean
```

```
HRESULT GetPos(float* X1, float* Y1, float* X2, float* Y2,
VARIANT_BOOL* retVal);
```

Siehe auch: WallSegment Übersicht

Liefert die Position der beiden (virtuellen) Endpunkte der (virtuellen) Achse dieses Segments.

Parameter sind:

x1, y1	Werden auf die Koordinaten des Startpunktes gesetzt
x2, y2	Werden auf die Koordinaten des Endpunktes gesetzt

WallSegment.RemoveHolePolygon

Deklaration:

```
Function RemoveHolePolygon(ByVal thePolygon as
HolePolygon) as Boolean
```

```
HRESULT RemoveHolePolygon(IHolePolygon*
thePolygon, VARIANT_BOOL* retVal);
```

Siehe auch: WallSegment Übersicht, HolePolygon

WallSegment.SetLineColor

Deklaration:

```
Sub SetLineColor(ByVal col as long)
```

```
HRESULT SetLineColor(long col);
```

Siehe auch: WallSegment Übersicht

WallSegmentCollection

Eine Liste von Wandsegmenten.
Näheres über die Programmierung mit Listen erfahren Sie im Kapitel Listen.

WallSegmentCollection Übersicht

Eigenschaften

Count	Anzahl Segmente der Wandseite

Methoden

Item	Das i-te Segment

WallSegmentCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: WallSegmentCollection Übersicht, Listen

Anzahl der Elemente in der Liste. Eine Zuweisung an diese Eigenschaft ist nicht möglich.

WallSegmentCollection.Item

Deklaration:

```
Function Item(ByVal Index as long) as WallSegment
```

```
HRESULT Item(long Index, IWallSegment** retVal);
```

Siehe auch: WallSegmentCollection Übersicht, WallSegment, Listen

Liefert ein Wandsegment aus der Liste.

Parameter:

Index	Index des gewünschten Wandsegmentes. Indizes beginnen bei 1, der maximale Index ist Count

WindowCollection

Eine Liste von Fenstern.
Näheres über die Programmierung mit Listen erfahren Sie im Kapitel Listen.

WindowCollection Übersicht

Eigenschaften

Count	Anzahl der Fenster

Methoden

Item	Das i-te Fenster (Window oder RoofWindow)

WindowCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: WindowCollection Übersicht, Listen

Anzahl der Elemente in der Liste. Eine Zuweisung an diese Eigenschaft ist nicht möglich.

WindowCollection.Item

Deklaration:

```
Function Item(ByVal Index as long) as Object
```

```
HRESULT Item(long Index, IDispatch** retVal);
```

Siehe auch: WindowCollection Übersicht, Listen

Liefert ein Fenster aus der Liste.

Parameter:

Index	Index des gewünschten Fensters. Indizes beginnen bei 1, der maximale Index ist Count

DoorCollection

Eine Liste von Türen.
Näheres über die Programmierung mit Listen erfahren Sie im Kapitel Listen.

DoorCollection Übersicht

Eigenschaften

Count	Anzahl der Türen

Methoden

Item	Die i-te Tür

DoorCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: DoorCollection Übersicht, Listen

Anzahl der Elemente in der Liste. Eine Zuweisung an diese Eigenschaft ist nicht möglich.

DoorCollection.Item

Deklaration:

```
Function Item(ByVal Index as long) as Door
```

```
HRESULT Item(long Index, IDoor** retVal);
```

Siehe auch: DoorCollection Übersicht, Door, Listen

Liefert eine Tür aus der Liste.

Parameter:

Index	Index der gewünschten Tür. Indizes beginnen bei 1, der maximale Index ist Count

RoofWindow

Ein Dachfenster.

RoofWindow Übersicht

Eigenschaften

Area	Fensterfläche
AreaFormula	Formel für Fensterfläche
Height	Höhe
History	Daten zur Identifikation im ehemaligen Projekt für nachgeladene Gebäude
ID	Eine eindeutige Kennung
OpensInwards	Nach innen öffnend?
Remark	Bemerkung
RightHung	Rechts angeschlagen?
Roof	Dach, in dem das Fenster 'sitzt'
TextureCount	Anzahl der max. verfügbaren Texturen
Type	Typ des Dachfensters (Variante des Multifunktionsschalters)
Width	Breite

Methoden

GetPos2D	Position lesen
GetPos3D	Position inkl. Höhe lesen
GetTexture	Liefert eine Textur
SetTexture	Tauscht eine Textur aus

RoofWindow.Area

Deklaration:

Single (nur lesbar)

HRESULT get_Area(float* retVal);

Siehe auch: RoofWindow Übersicht

Die Fläche des Dachfensters in Quadratmetern.

RoofWindow.AreaFormula

Deklaration:

String (nur lesbar)

HRESULT get_AreaFormula(BSTR* retVal);

Siehe auch: RoofWindow Übersicht

Eine symbolische Formel für die Fläche des Fensters.

RoofWindow.Height

Deklaration:

Single

HRESULT get_Height(float* retVal);
HRESULT put_Height(float newVal);

Siehe auch: RoofWindow Übersicht

Die Höhe (in m) des Dachfensters (nicht die Höhe im Geschoß).

RoofWindow.History

Deklaration:

IDHistory (nur lesbar)

HRESULT get_History(IIDHistory** retVal);

Siehe auch: RoofWindow Übersicht, IDHistory

Falls dieses RoofWindow beim Ursprünglichen Laden (oder Erstellen) des aktuellen Projektes nicht in diesem Projekt war, sonder aus einem anderen Projekt nachgeladen wurde, liefert diese Eigenschaft das History-Objekt, das die Umsetzung der IDs beschreibt. Andernfalls ist diese Eigenschaft “ Nothing” .

Ein Anwendungsbeispiel für History-Objekte finden Sie im Kapitel Eindeutige Kennungen.

RoofWindow.ID

Deklaration:

long (nur lesbar)

HRESULT get_ID(long* retVal);

Siehe auch: RoofWindow Übersicht

Gibt eine eindeutige Kennung dieses RoofWindow an. Diese Kennung ist persistent, das heißt, sie bleibt innerhalb des aktuellen Projektes erhalten. Sie können diese Kennung benutzen, um RoofWindows miteinander zu vergleichen oder zusätzliche Daten zum RoofWindow in einer Datenbank abzulegen.

Die ID ist aber nur innerhalb des aktuellen Projektes eindeutig, wenn Sie (durch Nachladen einzelner Gebäude) das Objekt in ein anderes Projekt einfügen, erhält es eine neue ID. In diesem Fall müssen Sie sofort bei Beendigung des Ladevorganges mit Hilfe der Eigenschaft History objPropertyDetailsRoofWindowHistory Ihre zusätzlichen Daten an die neuen Werte anpassen.

RoofWindow.OpensInwards

Deklaration:

Boolean

HRESULT get_OpensInwards(VARIANT_BOOL* retVal);
HRESULT put_OpensInwards(VARIANT_BOOL newVal);

Siehe auch: RoofWindow Übersicht

“ Wahr” , wenn das Fenster sich nach innen öffnet, “ Falsch” wenn es nach außen öffnet.

RoofWindow.Remark

Deklaration:

String

HRESULT get_Remark(BSTR* retVal);
HRESULT put_Remark(BSTR newVal);

Siehe auch: RoofWindow Übersicht

Eine beliebige Bemerkung zum Fenster.

RoofWindow.RightHung

Deklaration:

Boolean

HRESULT get_RightHung(VARIANT_BOOL* retVal);
HRESULT put_RightHung(VARIANT_BOOL newVal);

Siehe auch: RoofWindow Übersicht

“ Wahr” wenn das Fenster rechts angeschlagen ist, “ Falsch” wenn links.

RoofWindow.Roof

Deklaration:

Roof (nur lesbar)

HRESULT get_Roof(IRoof** retVal);

Siehe auch: RoofWindow Übersicht, Roof

Liefert das Dach, in dem das Fenster ist.

RoofWindow.TextureCount

Deklaration:

long (nur lesbar)

HRESULT get_TextureCount(long* retVal);

Siehe auch: RoofWindow Übersicht

RoofWindow.Type

Deklaration:

long

HRESULT get_Type(long* retVal);
HRESULT put_Type(long newVal);

Siehe auch: RoofWindow Übersicht

Der Typ des Fensters (entspricht der Auswahl des Variantenschalters beim Plazieren).

RoofWindow.Width

Deklaration:

Single

HRESULT get_Width(float* retVal);
HRESULT put_Width(float newVal);

Siehe auch: RoofWindow Übersicht

Die Breite des Fensters (in m).

RoofWindow.GetPos2D

Deklaration:

```
Function GetPos2D(ByRef X as Single, ByRef Y as Single)
as Boolean
```

```
HRESULT GetPos2D(float* X, float* Y, VARIANT_BOOL*
retVal);
```

Siehe auch: RoofWindow Übersicht

Ermittelt die Position in 2D Koordinaten. Die beiden Referenzparameter werden auf x und y Koordinaten gesetzt.

Parameter:

X	X-Koordinate des Fenstermittelpunktes
Y	Y-Koordinate des Fenstermittelpunktes

RoofWindow.GetPos3D

Deklaration:

```
Function GetPos3D(ByRef X as Single, ByRef Y as
Single, ByRef Z as Single) as Boolean
```

```
HRESULT GetPos3D(float* X, float* Y, float* Z,
VARIANT_BOOL* retVal);
```

Siehe auch: RoofWindow Übersicht

Liefert die vollständigen Koordinaten des Fensters. Die drei Referenzparameter werden entsprechend auf x, y, und z gesetzt.

Parameter:

X	X-Koordinate des Fenstermittelpunktes
Y	Y-Koordinate
Z	Z-Koordinate

Es ist nicht möglich, ein Dachfenster nachträglich zu verschieben. Löschen Sie es und fügen stattdessen ein neues ein (mit Roof.PlaceWindow).

RoofWindow.GetTexture

Deklaration:

```
Function GetTexture(ByVal index as long) as String
```

```
HRESULT GetTexture(long index, BSTR* retVal);
```

Siehe auch: RoofWindow Übersicht

RoofWindow.SetTexture

Deklaration:

```
Function SetTexture(ByVal index as long, ByVal TextureName
as String) as Boolean
```

```
HRESULT SetTexture(long index, BSTR TextureName,
VARIANT_BOOL* retVal);
```

Siehe auch: RoofWindow Übersicht

TerrainCollection

Eine Liste von Geländen.
Näheres über die Programmierung mit Listen erfahren Sie im Kapitel Listen.

TerrainCollection Übersicht

Eigenschaften

Count	Anzahl der Gelände

Methoden

Item	Das i-te Gelände

TerrainCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: TerrainCollection Übersicht, Listen

Anzahl der Elemente in der Liste. Eine Zuweisung an diese Eigenschaft ist nicht möglich.

TerrainCollection.Item

Deklaration:

```
Function Item(ByVal Index as long) as Terrain
```

```
HRESULT Item(long Index, ITerrain** retVal);
```

Siehe auch: TerrainCollection Übersicht, Terrain, Listen

Liefert ein Gelände aus der Liste.

Parameter:

Index	Index des gewünschten Geländes. Indizes beginnen bei 1, der maximale Index ist Count

GuideCollection

Eine Liste von Stützen.
Näheres über die Programmierung mit Listen erfahren Sie im Kapitel Listen.

GuideCollection Übersicht

Eigenschaften

Count	Anzahl der Hilfslinien

Methoden

Item	Die i-te Hilfslinie

GuideCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: GuideCollection Übersicht, Listen

Anzahl der Elemente in der Liste. Eine Zuweisung an diese Eigenschaft ist nicht möglich.

GuideCollection.Item

Deklaration:

```
Function Item(ByVal Index as long) as Guide
```

```
HRESULT Item(long Index, IGuide** retVal);
```

Siehe auch: GuideCollection Übersicht, Guide, Listen

Liefert eine Stütze aus der Liste.

Parameter:

Index	Index der gewünschten Stütze. Indizes beginnen bei 1, der maximale Index ist Count

LabelingCollection

Eine Liste von Beschriftungen.
Näheres über die Programmierung mit Listen erfahren Sie im Kapitel Listen.

LabelingCollection Übersicht

Eigenschaften

Count	Anzahl der Beschriftungen

Methoden

Item	Die i-te Beschriftung

LabelingCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: LabelingCollection Übersicht, Listen

Anzahl der Elemente in der Liste. Eine Zuweisung an diese Eigenschaft ist nicht möglich.

LabelingCollection.Item

Deklaration:

```
Function Item(ByVal Index as long) as Labeling
```

```
HRESULT Item(long Index, ILabeling** retVal);
```

Siehe auch: LabelingCollection Übersicht, Labeling, Listen

Liefert eine Beschriftung aus der Liste.

Parameter:

Index	Index der gewünschten Beschriftung. Indizes beginnen bei 1, der maximale Index ist Count

SupportCollection

Eine Liste von

Näheres über die Programmierung mit Listen erfahren Sie im Kapitel Listen.

SupportCollection Übersicht

Eigenschaften

Count	Anzahl der Stützen

Methoden

Item	Die i-te Stütze

SupportCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: SupportCollection Übersicht, Listen

Anzahl der Elemente in der Liste. Eine Zuweisung an diese Eigenschaft ist nicht möglich.

SupportCollection.Item

Deklaration:

```
Function Item(ByVal Index as long) as Support
```

```
HRESULT Item(long Index, ISupport** retVal);
```

Siehe auch: SupportCollection Übersicht, Support, Listen

Liefert eine Stütze aus der Liste.

Parameter:

Index	Index der gewünschten Stütze. Indizes beginnen bei 1, der maximale Index ist Count

ChimneyCollection

Eine Liste von Schornsteinen.
Näheres über die Programmierung mit Listen erfahren Sie im Kapitel Listen.

ChimneyCollection Übersicht

Eigenschaften

Count	Anzahl der Schornsteine

Methoden

Item	Der i-te Schornstein

ChimneyCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: ChimneyCollection Übersicht, Listen

Anzahl der Elemente in der Liste. Eine Zuweisung an diese Eigenschaft ist nicht möglich.

ChimneyCollection.Item

Deklaration:

```
Function Item(ByVal Index as long) as Chimney
```

```
HRESULT Item(long Index, IChimney** retVal);
```

Siehe auch: ChimneyCollection Übersicht, Chimney, Listen

Liefert einen Schornstein aus der Liste.

Parameter:

Index	Index des gewünschten Schornsteins. Indizes beginnen bei 1, der maximale Index ist Count

RoofCollection

Eine Liste von Dächern.
Näheres über die Programmierung mit Listen erfahren Sie im Kapitel Listen.

RoofCollection Übersicht

Eigenschaften

Count	Anzahl der Dächer

Methoden

Item	Das i-te Dach

RoofCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: RoofCollection Übersicht, Listen

Anzahl der Elemente in der Liste. Eine Zuweisung an diese Eigenschaft ist nicht möglich.

RoofCollection.Item

Deklaration:

```
Function Item(ByVal Index as long) as Roof
```

```
HRESULT Item(long Index, IRoof** retVal);
```

Siehe auch: RoofCollection Übersicht, Roof, Listen

Liefert ein Dach aus der Liste.

Parameter:

Index	Index des gewünschten Dachs. Indizes beginnen bei 1, der maximale Index ist Count

CeilingCollection

Eine Liste von Deckenplatten.
Näheres über die Programmierung mit Listen erfahren Sie im Kapitel Listen.

CeilingCollection Übersicht

Eigenschaften

Count	Anzahl der Deckenplatten

Methoden

Item	Die i-te Deckenplatte

CeilingCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: CeilingCollection Übersicht, Listen

Anzahl der Elemente in der Liste. Eine Zuweisung an diese Eigenschaft ist nicht möglich.

CeilingCollection.Item

Deklaration:

```
Function Item(ByVal Index as long) as Ceiling
```

```
HRESULT Item(long Index, ICeiling** retVal);
```

Siehe auch: CeilingCollection Übersicht, Ceiling, Listen

Liefert eine Decke aus der Liste.

Parameter:

Index	Index der gewünschten Decke. Indizes beginnen bei 1, der maximale Index ist Count

CeilingOpeningsCollection

Eine Liste von Dächern.
Näheres über die Programmierung mit Listen erfahren Sie im Kapitel Listen.

CeilingOpeningsCollection Übersicht

Eigenschaften

Count	Anzahl der Deckenaussparungen

Methoden

Item	Die i-te Deckenaussparung

CeilingOpeningsCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: CeilingOpeningsCollection Übersicht, Listen

Anzahl der Elemente in der Liste. Eine Zuweisung an diese Eigenschaft ist nicht möglich.

CeilingOpeningsCollection.Item

Deklaration:

```
Function Item(ByVal Index as long) as CeilingOpening
```

```
HRESULT Item(long Index, ICeilingOpening** retVal);
```

Siehe auch: CeilingOpeningsCollection Übersicht, CeilingOpening, Listen

Liefert eine Dachöffnung aus der Liste.

Parameter:

Index	Index der gewünschten Dachöffnung. Indizes beginnen bei 1, der maximale Index ist Count

StairCaseCollection

Eine Liste von Treppen.
Näheres über die Programmierung mit Listen erfahren Sie im Kapitel Listen.

StairCaseCollection Übersicht

Eigenschaften

Count	Anzahl der Treppen

Methoden

Item	Die i-te Treppe

StairCaseCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: StairCaseCollection Übersicht, Listen

Anzahl der Elemente in der Liste. Eine Zuweisung an diese Eigenschaft ist nicht möglich.

StairCaseCollection.Item

Deklaration:

```
Function Item(ByVal Index as long) as StairCase
```

```
HRESULT Item(long Index, IStairCase** retVal);
```

Siehe auch: StairCaseCollection Übersicht, StairCase, Listen

Liefert eine Treppe aus der Liste.

Parameter:

Index	Index der gewünschten Treppe. Indizes beginnen bei 1, der maximale Index ist Count

Hole

Ein Wandloch.

Hole Übersicht

Eigenschaften

History	Daten zur Identifikation im ehemaligen Projekt für nachgeladene Gebäude
ID	Eine eindeutige Kennung
LeftAngle	Gehrung der linken Seite in Radiant, Pi/2 = keine Gehrung
LeftBase	Der Wert 'lu' im Dialog
LeftHeight	Der Wert 'lh' im Dialog
LeftToTop	Linke Seite bis zur Decke
Polygon	Das Polygon des Wandloches (nicht für handverlegte Löcher!)
RightAngle	Gehrung der rechten Seite in Radiant, Pi/2 = keine Gehrung
RightBase	Der Wert 'ru' im Dialog
RightHeight	Der Wert 'rh' im Dialog
RightToTop	Rechte Seite bis zur Decke
Texture	Liefert die Textur des Wandloches
Width	Breite des Wandlochs

Methoden

Delete	Löscht das Wandloch
GetPolygons	Liefert die Löcher zu beiden Seiten des Wandlochs, die durch das Loch entstehen sowie deren Tiefe
GetPos	Liefert die Position des Wandlochs
LeftSegment	Liefert das Wandsegment an der linken Lochseite
RightSegment	Liefert das Wandsegment an der rechten Lochseite
SetPos	Positioniert das Wandloch neu

Hole.History

Deklaration:

IDHistory (nur lesbar)

HRESULT get_History(IIDHistory** retVal);

Siehe auch: Hole Übersicht, IDHistory

Falls dieses Hole beim Ursprünglichen Laden (oder Erstellen) des aktuellen Projektes nicht in diesem Projekt war, sonder aus einem anderen Projekt nachgeladen wurde, liefert diese Eigenschaft das History-Objekt, das die Umsetzung der IDs beschreibt. Andernfalls ist diese Eigenschaft “ Nothing” .

Ein Anwendungsbeispiel für History-Objekte finden Sie im Kapitel Eindeutige Kennungen.

Hole.ID

Deklaration:

long (nur lesbar)

HRESULT get_ID(long* retVal);

Siehe auch: Hole Übersicht

Gibt eine eindeutige Kennung dieses Hole an. Diese Kennung ist persistent, das heißt, sie bleibt innerhalb des aktuellen Projektes erhalten. Sie können diese Kennung benutzen, um Holes miteinander zu vergleichen oder zusätzliche Daten zum Hole in einer Datenbank abzulegen.

Die ID ist aber nur innerhalb des aktuellen Projektes eindeutig, wenn Sie (durch Nachladen einzelner Gebäude) das Objekt in ein anderes Projekt einfügen, erhält es eine neue ID. In diesem Fall müssen Sie sofort bei Beendigung des Ladevorganges mit Hilfe der Eigenschaft History objPropertyDetailsHoleHistory Ihre zusätzlichen Daten an die neuen Werte anpassen.

Hole.LeftAngle

Deklaration:

Single

HRESULT get_LeftAngle(float* retVal);
HRESULT put_LeftAngle(float newVal);

Siehe auch: Hole Übersicht

Die Gehrung der linken Seite des Lochs in Bogenmaß. Falls das Loch keine Gehrung hat, wird der Wert Pi/2 benutzt.

Hole.LeftBase

Deklaration:

Single

HRESULT get_LeftBase(float* retVal);
HRESULT put_LeftBase(float newVal);

Siehe auch: Hole Übersicht

Die linke untere Basishöhe des Lochs. Vergleiche auch die Skizze im Wandlochdialog, dort entspricht diese Eigenschaft dem Wert “ lu” .

Hole.LeftHeight

Deklaration:

Single

HRESULT get_LeftHeight(float* retVal);
HRESULT put_LeftHeight(float newVal);

Siehe auch: Hole Übersicht

Die Höhe der linken Seite. Siehe auch die Skizze im Wandlochdialog: dieser Wert entspricht “ lh” .

Hole.LeftToTop

Deklaration:

Boolean

HRESULT get_LeftToTop(VARIANT_BOOL* retVal);
HRESULT put_LeftToTop(VARIANT_BOOL newVal);

Siehe auch: Hole Übersicht

Falls dieses Attribut “ Wahr” ist, wird die Höhenangabe der linken Seite ignoriert und das Loch dort bis zur Decke dargestellt.

Hole.Polygon

Deklaration:

Polygon2D

HRESULT get_Polygon(IPolygon2D** retVal);
HRESULT put_Polygon(IPolygon2D* newVal);

Siehe auch: Hole Übersicht, Polygon2D

Das Polygon des Wandlochs, falls es sich um ein von einem anderen Makro erzeugtes Polygon handelt. Derzeit gibt es keine Möglichkeit für Benutzer, ein Lochpolygon einzugeben.

Hole.RightAngle

Deklaration:

Single

HRESULT get_RightAngle(float* retVal);
HRESULT put_RightAngle(float newVal);

Siehe auch: Hole Übersicht

Die Gehrung der rechten Lochseite. Der Wert Pi/2 steht für keine Gehrung.

Hole.RightBase

Deklaration:

Single

HRESULT get_RightBase(float* retVal);
HRESULT put_RightBase(float newVal);

Siehe auch: Hole Übersicht

Die rechte untere Basishöhe. Vergleiche auch die Skizze im Wandlochdialog; dort entspricht diese Eigenschaft dem Wert “ ru” .

Hole.RightHeight

Deklaration:

Single

HRESULT get_RightHeight(float* retVal);
HRESULT put_RightHeight(float newVal);

Siehe auch: Hole Übersicht

Die Höhe der rechten Seite des Lochs.

Hole.RightToTop

Deklaration:

Boolean

HRESULT get_RightToTop(VARIANT_BOOL* retVal);
HRESULT put_RightToTop(VARIANT_BOOL newVal);

Siehe auch: Hole Übersicht

Falls dieses Flag “ Wahr” ist, wird die Höhenangabe für die rechte Seite ignoriert und das Loch dort bis zur Decke dargestellt.

Hole.Texture

Deklaration:

Texture (nur lesbar)

HRESULT get_Texture(ITexture** retVal);

Siehe auch: Hole Übersicht, Texture

Die Textur des Wandlochs.

Hole.Width

Deklaration:

Single

HRESULT get_Width(float* retVal);
HRESULT put_Width(float newVal);

Siehe auch: Hole Übersicht

Die Breite des Lochs.

Hole.Delete

Deklaration:

```
Sub Delete()
```

```
HRESULT Delete();
```

Siehe auch: Hole Übersicht

Löscht das Wandloch.

Hole.GetPolygons

Deklaration:

```
Function GetPolygons(ByRef leftPolygon as
Point2DCollection, ByRef rightPolygon as Point2DCollection,
ByRef leftDepth as Single, ByRef rightDepth as Single) as
Boolean
```

```
HRESULT GetPolygons(IPoint2DCollection** leftPolygon,
IPoint2DCollection** rightPolygon, float* leftDepth, float*
rightDepth, VARIANT_BOOL* retVal);
```

Siehe auch: Hole Übersicht, Point2DCollection, Point2DCollection

Hole.GetPos

Deklaration:

```
Function GetPos(ByRef x as Single, ByRef y as Single) as
Boolean
```

```
HRESULT GetPos(float* x, float* y, VARIANT_BOOL*
retVal);
```

Siehe auch: Hole Übersicht

Hole.LeftSegment

Deklaration:

```
Function LeftSegment() as WallSegment
```

```
HRESULT LeftSegment(IWallSegment** retVal);
```

Siehe auch: Hole Übersicht, WallSegment

Hole.RightSegment

Deklaration:

```
Function RightSegment() as WallSegment
```

```
HRESULT RightSegment(IWallSegment** retVal);
```

Siehe auch: Hole Übersicht, WallSegment

Hole.SetPos

Deklaration:

```
Function SetPos(ByVal x as Single, ByVal y as Single) as
Boolean
```

```
HRESULT SetPos(float x, float y, VARIANT_BOOL* retVal);
```

Siehe auch: Hole Übersicht

Positioniert ein Wandloch.

Rückgabewert ist “ Wahr” , wenn die Positionsänderung erfolgreich war.

Die Parameter sind:

x	x-Koordinate des Loch-Mittelpunktes
y	y-Koordinate des Loch-Mittelpunktes

HoleCollection

Eine Liste von Wandlöchern.
Näheres über die Programmierung mit Listen erfahren Sie im Kapitel Listen.

HoleCollection Übersicht

Eigenschaften

Count	Anzahl der Aussparungen

Methoden

Item	Die i-te Aussparung

HoleCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: HoleCollection Übersicht, Listen

Anzahl der Elemente in der Liste. Eine Zuweisung an diese Eigenschaft ist nicht möglich.

HoleCollection.Item

Deklaration:

```
Function Item(ByVal Index as long) as Hole
```

```
HRESULT Item(long Index, IHole** retVal);
```

Siehe auch: HoleCollection Übersicht, Hole, Listen

Liefert ein Wandloch aus der Liste.

Parameter:

Index	Index des gewünschten Wandlochs. Indizes beginnen bei 1, der maximale Index ist Count

DimensionCollection

Eine Liste von Vermaßungen.
Näheres über die Programmierung mit Listen erfahren Sie im Kapitel Listen.

DimensionCollection Übersicht

Eigenschaften

Count	Anzahl der Vermaßungen

Methoden

Item	Die i-te Vermaßung

DimensionCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: DimensionCollection Übersicht, Listen

Anzahl der Elemente in der Liste. Eine Zuweisung an diese Eigenschaft ist nicht möglich.

DimensionCollection.Item

Deklaration:

```
Function Item(ByVal Index as long) as Dimension
```

```
HRESULT Item(long Index, IDimension** retVal);
```

Siehe auch: DimensionCollection Übersicht, Dimension, Listen

Liefert eine Vermaßung aus der Liste.

Parameter:

Index	Index der gewünschten Vermaßung. Indizes beginnen bei 1, der maximale Index ist Count

LambdaH

Beim Verschneiden von Wänden unter Dachschrägen und gleichzeitig mit anderen Wänden kommt es zu Wandsegmenten, die keine einheitliche Höhe haben. Um für diese Segmente den Höhenverlauf darzustellen, werden LambdaH Objekte benutzt. Sie geben für einen Höhenpunkt im Höhenverlauf jeweils die Höhe (h) und den Bruchteil der Länge des gesamten Wandsegments bis zu diesem Höhenpunkt (lambda) an.

LambdaH Übersicht

Eigenschaften

Height	Höhe der Wand an diesem Punkt
Lambda	Anteil der Wandlänge bis zu diesem Höhenpunkt (0 <= lambda <= 1)

LambdaH.Height

Deklaration:

Single

HRESULT get_Height(float* retVal);
HRESULT put_Height(float newVal);

Siehe auch: LambdaH Übersicht

Die Höhe des Wandsegmentes an diesem Höhenpunkt.

LambdaH.Lambda

Deklaration:

Single (nur lesbar)

HRESULT get_Lambda(float* retVal);

Siehe auch: LambdaH Übersicht

Ein Wert zwischen 0 und 1, wobei 0 dem Anfang des Wandsegmentes und 1 dessen Ende bezeichnet. Wenn Sie mit Hilfe der Methode GetPosdie Koordinaten p1 und p2 des Anfangs- bzw. Endpunktes des Segmentes ermitteln, können Sie die Koordinaten des Höhenpunktes ermitteln, indem Sie den Differenzvektor d von p1 nach p2 berechnen und erhalten dann die Koordinaten ph des Höhenpunktes als ph = p1 + d * lambda.

LambdaHCollection

Eine Liste von Höhenpunkten.
Näheres über die Programmierung mit Listen erfahren Sie im Kapitel Listen.

LambdaHCollection Übersicht

Eigenschaften

Count	Anzahl der Höhenpunkte

Methoden

Item	Der i-te Höhenpunkt des Wandsegmentes

LambdaHCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: LambdaHCollection Übersicht, Listen

Anzahl der Elemente in der Liste. Eine Zuweisung an diese Eigenschaft ist nicht möglich.

LambdaHCollection.Item

Deklaration:

```
Function Item(ByVal Index as long) as LambdaH
```

```
HRESULT Item(long Index, ILambdaH** retVal);
```

Siehe auch: LambdaHCollection Übersicht, LambdaH, Listen

Liefert einen Höhenpunkt aus der Liste.

Parameter:

Index	Index des gewünschten Höhenpunktes. Indizes beginnen bei 1, der maximale Index ist Count

Material

Ein Material beschreibt die Oberflächeneigenschaften von 3D Objekten, die für eine Darstellung in ArCon entscheidend sind. Bei der Erzeugung von Objekten legen Sie Standard-Materialien fest, bei der Erzeugung von Objektinstanzen können Sie diese durch das konkrete Material ersetzen. Sie können auch selbst definierte oder aus einer Datei geladene Materialien nachträglich auf Oberflächen von Objekten ziehen.

Material Übersicht

Eigenschaften

AmbientCoefficient	Faktor für die Reflektion ambienten Lichtes
DiffuseCoefficient	Faktor für die diffuse Reflektion
DiffuseColor	Farbe für diffus reflektierendes Licht
Flags	Verschiedene Flags, z.B. doppelseitig (ACMATFL_TWOSIDED)
HighlightExponent	Phong'scher Exponent
SpecularCoefficient	Faktor für gerichtete Reflektion
SpecularColor	Farbe für gerichtet reflektiertes Licht
Transparency	Faktor für die Lichtdurchlässigkeit
Transparent	Ist das Material durchsichtig?

Methoden

DragStart	Begint einen Material-Drag&Drop Vorgang

Material.AmbientCoefficient

Deklaration:

Single

HRESULT get_AmbientCoefficient(float* retVal);
HRESULT put_AmbientCoefficient(float newVal);

Siehe auch: Material Übersicht

Dieser Wert gibt an, wieviel Prozent des einfallenden (diffusen) Umgebungslichtes vom Material wieder abgestrahlt wird. Im Gegensatz zu DiffuseCoefficient und SpecularCoefficient gibt AmbientCoefficient die Grundhelligkeit des Materials an.

Die Zahl muß im Bereich zwischen 0% (dunkel) und 100% (hell) liegen.

Material.DiffuseCoefficient

Deklaration:

Single

HRESULT get_DiffuseCoefficient(float* retVal);
HRESULT put_DiffuseCoefficient(float newVal);

Siehe auch: Material Übersicht

Dieser Wert gibt an, wieviel Prozent des einfallenden gerichteten Lichtes vom Material diffus (in verschiedene Richtungen) gespiegelt wird. Im Gegensatz zu AmbientCoefficient und SpecularCoefficient gibt AmbientCoefficient die Helligkeit des Materials an.

Die Zahl muß im Bereich zwischen 0% (dunkel) und 100% (hell) liegen.

Siehe auch: DiffuseColor

Material.DiffuseColor

Deklaration:

long

HRESULT get_DiffuseColor(long* retVal);
HRESULT put_DiffuseColor(long newVal);

Siehe auch: Material Übersicht

Der RGB-Wert der Farbe, mit der das Material Licht diffus reflektiert. Die Helligkeit dieser Reflektion wird mit DiffuseCoefficient angegeben.

Material.Flags

Deklaration:

long

HRESULT get_Flags(long* retVal);
HRESULT put_Flags(long newVal);

Siehe auch: Material Übersicht

Die einzelnen Bits dieses Wertes geben diverse Materialeigenschaften an. Der Gesamtwert errechnet sich durch Addition der einzelnen Werte. Die möglichen Konstanten finden Sie im Kapitel Material Flags.

Material.HighlightExponent

Deklaration:

Single

HRESULT get_HighlightExponent(float* retVal);
HRESULT put_HighlightExponent(float newVal);

Siehe auch: Material Übersicht

Der “ Phong’ sche Exponent” des Materials. Er gibt (in etwa) an, wie die Helligkeit des Materials mit zunehmender Entfernung von der Lichtquelle abnimmt.

Material.SpecularCoefficient

Deklaration:

Single

HRESULT get_SpecularCoefficient(float* retVal);
HRESULT put_SpecularCoefficient(float newVal);

Siehe auch: Material Übersicht

Dieser Wert gibt an, wieviel Prozent des einfallenden gerichteten Lichtes vom Material gespiegelt wird. Im Gegensatz zu DiffuseCoefficient und AmbientCoefficient gibt SpecularCoefficient den Glanz des Materials an.

Die Zahl muß im Bereich zwischen 0% (dunkel) und 100% (hell) liegen.

Material.SpecularColor

Deklaration:

long

HRESULT get_SpecularColor(long* retVal);
HRESULT put_SpecularColor(long newVal);

Siehe auch: Material Übersicht

Die Farbe, mit der gerichtetes Licht reflektiert wird. Die Stärke der Reflektion wird durch SpecularCoefficient bestimmt, eine eventuelle Mischung mit der Texturfarbe durch Flags.

Material.Transparency

Deklaration:

Single

HRESULT get_Transparency(float* retVal);
HRESULT put_Transparency(float newVal);

Siehe auch: Material Übersicht

Falls das Material durchsichtig ist (durch Transparent bestimmt), gibt dieser Wert zwischen 0 und 100% an, wieviel Licht durch das Material hindurchscheint.

Material.Transparent

Deklaration:

Boolean

HRESULT get_Transparent(VARIANT_BOOL* retVal);
HRESULT put_Transparent(VARIANT_BOOL newVal);

Siehe auch: Material Übersicht

Gibt an, ob das Material durchsichtig ist. Falls ja, wird mit Transparency das Ausmaß der Durchsichtigkeit bestimmt.

Material.DragStart

Deklaration:

```
Function DragStart() as Boolean
```

```
HRESULT DragStart(VARIANT_BOOL* retVal);
```

Siehe auch: Material Übersicht

ObjectConstructor

Die “ Klasse” eines 3D-Objektes. Gleichartige 3D-Objekte sind in einer Klasse. Fast alle Daten, die zur Erzeugung eines 3D Objektes benötigt werden, sind klassenspezifisch. Sie müssen für alle Objekte in dieser Klasse nur einmal gepseichert (und erzeugt) werden. Beispielsweise können Sie vier gleiche Stühle in ein Projekt einfügen, ohne dazu vier mal die Geometrie dieses Stuhltyps definieren zu müssen, auch wenn sich die einzelnen Stühle in Details (wie etwa dem Bezugsstoff oder der Position in der Welt) unterscheiden.

Jedes 3D Objekt ist in einer Klasse, auch wenn es die einzige Instanz dieser Klasse ist. Wenn Sie ein Objekt erzeugen, erstllen Sie zunächst eine Objektklasse (den ObjectConstructor) und anschließend Instanzen dieser Klasse (vom Typ Object3D).

Bei der Generierung von selbstmodellierten 3D Objekten durch Ihr Programm dient der ObjectConstructor als Hilfsobjekt zur Sammlung von temporären Daten. Das Konstruktor-Objekt speichert unfertige Objekte und sammelt die bereits festgelegten Geometrie- und Materialeigenschaften. Aus diesen Daten wird dann das optimierte 3D Objekt erstellt, von dem Sie Instanzen bilden können.

Alternativ .können Sie einen ObjectConstructor aus einer vorgefertigten ACO-Datei laden. Eine ACO Datei erzeugen können Sie nicht aus einem ObjectConstructor, da dazu einige Informationen fehlen. Sie können allerdings eine Instanz (ein Object3D) erzeugen und diese als ACO Datei speichern.

Ein ObjectConstructor wird mit Hilfe der Methode ArCon.NewObjectConstructor erzeugt.

ObjectConstructor Übersicht

Eigenschaften

Duration	Lebensdauer der Objektklasse
FileName	Dateiname, falls das Objekt geladen wurde
History	Information über ehemalige ID beim Nachladen einzelner Gebäude
ID	Eindeutige Kennung der Objektklasse
Objects	Alle Instanzen dieser Klasse
RelativeFileName	relativer Dateiname (> Notation)

Methoden

AddPolygonWithHoles	Fügt ein Polygon mit Löchern dem Objekt hinzu
AddPolygonWithHoles2	Fügt ein Polygon mit Löchern dem Objekt hinzu, dies kann ein schnappendes Polygon sein
AddQuadriliteral	Fügt ein Viereck dem Objekt hinzu
AddTriangle	Fügt ein Dreieck dem Objekt hinzu
Create	Erzeugt eine neue Instanz der Klasse (isRoomObject = gehört zum Raum, z.B. ein Fensterrahmen, und wirft daher keine Schatten in der Tagesansicht)
Finish	Beendet die Konstruktion eines Objektes
GetTextureCollection	Liefert die Texturnamensliste der Klasse
GetTextureCount	Liefert die Anzahl der von diesem Objekt verwendeten Texturen.
GetTextureName	Liefert den (logischen) Namen der angegebenen Textur (1 ist die erste).
GetURL	Liefert die gewünschte URL eines Internetobjektes (oder Leerstring, falls diese nicht vorhanden ist).
InitSnapObject	Erzeugt Fanginformation zu diesem Objekt
Properties	Liefert die Bitmask der Objekteigenschaften, z.B. ACDOP_HAS_LAMPS.
ReCreateFromFile	Ersetzt das Objekt und alle seine Instanzen durch ein neu geladenes
SetContext	Setzt den Pick-, Drag- und Drop-Kontext für die neu hinzugefügten Flächen. Dieser Wert wird bei Textur- und Material-Drag&Drop Operationen zurückgeliefert, um den betroffenen Teil des Objektes zu identifizieren.
SetContur	Setzt die Koordinaten einer Kontur aus einem Array
SetHoleContur	Setzt eine gesamte Lochkontur aus einem Array
SetHolePoint	Setzt die Koordinaten eines Lochpunktes
SetPoint	Setzt einen Punkt inklusive Texturkoordinaten bei der Konstruktion eines neuen Objektes

ObjectConstructor.Duration

Deklaration:

long (nur lesbar)

HRESULT get_Duration(long* retVal);

Siehe auch: ObjectConstructor Übersicht

Flags für das Cache-Verhalten der Objektklasse. Derzeit immer 1 (ACO_DURATION_CACHEABLE).

Die Klassen von 3D Objekten bleiben auf jeden Fall solange im Speicher, wie Instanzen dieser Klasse existieren (da ArCon die Klassendaten benötigt, um die Objekte anzeigen zu können). Anschließend werden sie nicht mehr benötigt – es sei denn, Sie laden das gleichen Objekt noch einmal. Der Ladevorgang benötigt Zeit (wenn auch auf heutigen Computern nicht sehr viel). Diese Zeit kann eingespart werden, wenn die Klassendaten noch im Speicher vorhanden sind. Es gibt nun verschiedene Strategien, wielange ArCon eigentlich nicht mehr benötigte Klassendaten im Speicher hält:

Wert	Symbolischer Name	Beschreibung
0	ACO_DURATION_ONLYWITHINSTANCES	Die Klasse wird überhaupt nicht gecached – sobald die letzte Instanz gelöscht wird, wird auch die Klasse gelöscht. Dieser Wert ist immer dann sinnvoll, wenn Ihre Objektklassen temporär und einmalig sind (z. B. dynamisch generiert werden).
1	ACO_DURATION_CACHEABLE	Der Standardwert: ArCon behält die Klasse im Speicher, solange genügend Speicherplatz verfügbar ist. Wird der Speicher knapp, werden die am längsten nicht mehr benötigten Klassen gelöscht.
2	ACP_DURATION_PERMANENT	Die Objektklasse wird nie gelöscht – erst am Programmende (wenn der gesamte Speicher freigegeben wird), werden die Klassendaten gelöscht.

ObjectConstructor.FileName

Deklaration:

String

HRESULT get_FileName
(BSTR* retVal);

HRESULT put_FileName
(BSTR newVal);

Siehe auch: ObjectConstructor Übersicht

Der absolute Dateiname der Objektklasse (die ACO-Datei), falls dieser ObjectConstructor aus einer ACO-Datei geladen wurde. Bei selbstdefinierten Objekten ist der Dateiname immer leer.

Siehe auch RelativeFileName.

ObjectConstructor.History

Deklaration:

IDHistory (nur lesbar)

```
HRESULT get_History
(IIDHistory** retVal);
```

Siehe auch: ObjectConstructor Übersicht, IDHistory

Falls dieses ObjectConstructor beim Ursprünglichen Laden (oder Erstellen) des aktuellen Projektes nicht in diesem Projekt war, sonder aus einem anderen Projekt nachgeladen wurde, liefert diese Eigenschaft das History-Objekt, das die Umsetzung der IDs beschreibt. Andernfalls ist diese Eigenschaft “ Nothing” .

Ein Anwendungsbeispiel für History-Objekte finden Sie im Kapitel Eindeutige Kennungen.

ObjectConstructor.ID

Deklaration:

long (nur lesbar)

HRESULT get_ID(long*
retVal);

Siehe auch: ObjectConstructor Übersicht

Gibt eine eindeutige Kennung dieses ObjectConstructor an. Diese Kennung ist persistent, das heißt, sie bleibt innerhalb des aktuellen Projektes erhalten. Sie können diese Kennung benutzen, um ObjectConstructors miteinander zu vergleichen oder zusätzliche Daten zum ObjectConstructor in einer Datenbank abzulegen.

Die ID ist aber nur innerhalb des aktuellen Projektes eindeutig, wenn Sie (durch Nachladen einzelner Gebäude) das Objekt in ein anderes Projekt einfügen, erhält es eine neue ID. In diesem Fall müssen Sie sofort bei Beendigung des Ladevorganges mit Hilfe der Eigenschaft History objPropertyDetailsObjectConstructorHistory Ihre zusätzlichen Daten an die neuen Werte anpassen.

ObjectConstructor.Objects

Deklaration:

Object3DCollection (nur lesbar)

HRESULT get_Objects (IObject3DCollection** retVal);

Siehe auch: ObjectConstructor Übersicht, Object3DCollection

Eine Liste aller Instanzen dieser 3D-Objekt-Klasse.

ObjectConstructor.RelativeFileName

Deklaration:

String (nur lesbar)

```
HRESULT
get_RelativeFileName
(BSTR* retVal);
```

Siehe auch: ObjectConstructor Übersicht

Der relative Dateiname der ACO Datei, aus der dieser ObjectConstructor geladen wurde. Bei selbstdefinierten Objekten ist dieser Name leer.

Ein relativer Name kann von ArCon nur ermittelt werden, wenn das Objekt aus einem Unterverzeichnis des Objektpfades geladen wurde. Ansonsten ist dieser Wert identisch mit der Eigenschaft FileName.

ObjectConstructor.AddPolygonWithHoles

Deklaration:

```
Sub AddPolygonWithHoles
(ByVal SharpEdges as
Boolean, ByVal mat as
Material, ByVal
TextureName as String)
```

```
HRESULT
AddPolygonWithHoles
(VARIANT_BOOL
SharpEdges, IMaterial*
mat, BSTR TextureName);
```

Siehe auch: ObjectConstructor Übersicht, Material

Fügt der Geometrie des gerade konstruierten Objekts ein Polygon mit Löchern hinzu. Dazu müssen vorher die Koordinaten der Punkte der äußeren Kontur mit SetPoint und der Löcher mit SetHolePoint definiert worden sein.

Diese Methode löscht den Punktzähler und macht damit alle Koordinaten ungültig.

Parameter sind:

sharpEdges	Gibt an, ob alle Ecken des Polygons scharfkantig sind
mat	Das Material für diese Fläche
textureName	Die zu verwendende Textur oder ein Leerstring, wenn dieses Polygon nicht texturiert werden soll.

ObjectConstructor.AddPolygonWithHoles2

Deklaration:

```
Sub AddPolygonWithHoles2(ByVal SharpEdges
as Boolean, ByVal Snap as Boolean, ByVal mat
as Material, ByVal TextureName as String)
```

```
HRESULT AddPolygonWithHoles2
(VARIANT_BOOL SharpEdges,
VARIANT_BOOL Snap, IMaterial* mat, BSTR
TextureName);
```

Siehe auch: ObjectConstructor Übersicht, Material

Fügt der Geometrie des gerade konstruierten Objekts ein Polygon mit Löchern hinzu. Dazu müssen vorher die Koordinaten der Punkte der äußeren Kontur mit SetPoint und der Löcher mit SetHolePoint definiert worden sein.

Diese Methode löscht den Punktzähler und macht damit alle Koordinaten ungültig.

Parameter sind:

sharpEdges	Gibt an, ob alle Ecken des Polygons scharfkantig sind
Snap	Das Polygon wird zum Fangen benutzt
mat	Das Material für diese Fläche
textureName	Die zu verwendende Textur oder ein Leerstring, wenn dieses Polygon nicht texturiert werden soll.

ObjectConstructor.AddQuadriliteral

Deklaration:

```
Sub AddQuadriliteral(ByVal
SharpEdgesAndSnapFlag as long, ByVal mat as
Material, ByVal TextureName as String)
```

```
HRESULT AddQuadriliteral(long
SharpEdgesAndSnapFlag, IMaterial* mat, BSTR
TextureName);
```

Siehe auch: ObjectConstructor Übersicht, Material

Fügt der Geometrie des gerade konstruierten Objektes ein Viereck hinzu.

Die Koordinaten der vier Punkte müssen zuvor mit SetPoint definiert worden sein.

Diese Methode löscht den Punktezähler, sodaß anschließend keine Punkte mehr definiert sind.

Parameter sind:

SharpEdgesAnd SnapFlag	Eine Bitmaske, welche der Seiten des Viereckes scharfkantig dargestellt werden sollen. Die Seite zwischen Punkt 0 und Punkt 1 ist scharf, wenn das 1. Bit gesetzt ist (mit dem Wert 1). Der Wert 10 würde also die Seiten zwischen Punkt 1 und 2 sowie die Seite zwischen Punkt 3 und 0 scharfkantig erscheinen lassen. Ist Bit 16 gesetzt (0x10000) wird das Viereck zum Fangen benutzt.
Mat	Das Material der Oberfläche
textureName	Der Name der Standardtextur für diese Fläche oder ein Leerstring, falls dieses Viereck nicht texturiert werden soll.

ObjectConstructor.AddTriangle

Deklaration:

```
Sub AddTriangle(ByVal
SharpEdgesAndSnapFlag as long, ByVal mat as
Material, ByVal TextureName as String)
```

```
HRESULT AddTriangle(long
SharpEdgesAndSnapFlag, IMaterial* mat, BSTR
TextureName);
```

Siehe auch: ObjectConstructor Übersicht, Material

Fügt der Geometrie des gerade konstruierten Objektes ein Dreieck hinzu.

Die Koordinaten der drei Eckpunkte müssen zuvor mit SetPoint definiert worden sein.

Diese Methode löscht den Punktezähler, sodaß anschließend keine Punkte mehr definiert sind.

Parameter sind:

sharpEdgesAnd SnapFlag	Eine Bitmaske, welche der Seiten des Viereckes scharfkantig dargestellt werden sollen. Die Seite zwischen Punkt 0 und Punkt 1 ist scharf, wenn das 1. Bit gesetzt ist (mit dem Wert 1). Der Wert 6 würde also die Seiten zwischen Punkt 1 und 2 sowie die Seite zwischen Punkt 2 und 0 scharfkantig erscheinen lassen. Ist Bit 16 gesetzt (0x10000) wird das Dreieck zum Fangen benutzt.
Mat	Das Material der Oberfläche
textureName	Der Name der Standardtextur für diese Fläche oder ein Leerstring, falls dieses Dreieck nicht texturiert sein soll.

ObjectConstructor.Create

Deklaration:

```
Function Create(ByVal textureList as
TextureCollection, ByVal isRoomObject as
Boolean) as Object3D
```

```
HRESULT Create(ITextureCollection*
textureList, VARIANT_BOOL isRoomObject,
IObject3D** retVal);
```

Siehe auch: ObjectConstructor Übersicht, Object3D, TextureCollection

Erzeugt eine Instanz dieser Objektklasse. Dabei können eigene Texturen statt der Standard-Texturliste verwendet werden.

Parameter:

textureList	Eine Texturliste mit Ersatztexturen für diese Instanz. Diese Liste muß ausreichend Texturen für die Instanziierung des Objektes enthalten.
IsRoomObject	Wahr, wenn das Objekt einem Raum zugeordnet ist. Raumobjekte sind z.B. Türen und Fensterbänke

ObjectConstructor.Finish

Deklaration:

```
Function Finish(ByVal Name as String, ByVal
storeInProject as Boolean, ByVal duration as long)
as Boolean
```

```
HRESULT Finish(BSTR Name, VARIANT_BOOL
storeInProject, long duration, VARIANT_BOOL*
retVal);
```

Siehe auch: ObjectConstructor Übersicht

Beendet die Objektkonstruktion und erstellt die Klassendaten des 3D Objektes.

Dabei werden diverse Optimierungen durchgeführt, die später eine schnelle Darstellung von Objektinstanzen erlauben. Sie sollten zuvor bei der Definition des Objektes keinerlei manuelle Optimierungen durchführen, da sie sich nicht auf die Darstellungsqualität und -geschwindigkeit des entstehenden Objektes auswirken werden.

ObjectConstructor.GetTextureCollection

Deklaration:

```
Function GetTextureCollection() as
TextureCollection
```

```
HRESULT GetTextureCollection
(ITextureCollection** retVal);
```

Siehe auch: ObjectConstructor Übersicht, TextureCollection

Liefert eine Texturliste der Objektklasse. Diese Liste beschreibt die Standard-Texturen des Objektes.

Sie können die Eigenschaft Count dieser Liste benutzen, um sich mit Hilfe von NewTextureCollection eine passende Texturliste zur Erzeugung einer Instanz dieser Klasse zu generieren und diese mit den Standard-Texturnamen zu initialisieren. Falls Sie die Standard-Texturnamen nicht benötigen, ist es geschickter, die Methode GetTextureCount zur Ermittlung der Anzahl der Texturen zu verwenden.

Anders als die aktiven Texturlisten von Instanzen ist diese Klassen-Texturliste nur lesbar – ihren Elementen kann kein neuer Texturname zugewiesen werden.

ObjectConstructor.GetTextureCount

Deklaration:

```
Function GetTextureCount() as long
```

```
HRESULT GetTextureCount(long* retVal);
```

Siehe auch: ObjectConstructor Übersicht

ObjectConstructor.GetTextureName

Deklaration:

```
Function GetTextureName(ByVal index as long) as
String
```

```
HRESULT GetTextureName(long index, BSTR*
retVal);
```

Siehe auch: ObjectConstructor Übersicht

ObjectConstructor.GetURL

Deklaration:

```
Function GetURL(ByVal language as String, ByVal
urlID as AcObjUrlNos) as String
```

```
HRESULT GetURL(BSTR language, AcObjUrlNos
urlID, BSTR* retVal);
```

Siehe auch: ObjectConstructor Übersicht

ObjectConstructor.InitSnapObject

Deklaration:

```
Function InitSnapObject(ByVal storeInProject as
Boolean) as Boolean
```

```
HRESULT InitSnapObject(ISnapObject *snapObj,
VARIANT_BOOL storeInProject, VARIANT_BOOL*
retVal);
```

Siehe auch: ObjectConstructor Übersicht

ObjectConstructor.Properties

Deklaration:

```
Function Properties() as long
```

```
HRESULT Properties(long* retVal);
```

Siehe auch: ObjectConstructor Übersicht

ObjectConstructor.ReCreateFromFile

Deklaration:

```
Function ReCreateFromFile(ByVal FileName as
String, ByVal duration as long) as Boolean
```

```
HRESULT ReCreateFromFile(BSTR FileName,
long duration, VARIANT_BOOL* retVal);
```

Siehe auch: ObjectConstructor Übersicht

ObjectConstructor.SetContext

Deklaration:

```
Sub SetContext(ByVal contextID as long)
```

```
HRESULT SetContext(long contextID);
```

Siehe auch: ObjectConstructor Übersicht

ObjectConstructor.SetContur

Deklaration:

Sub SetContur(ByVal numPoints as short, ByVal points as VARIANT)

HRESULT SetContur(short numPoints, VARIANT points);

Siehe auch: ObjectConstructor Übersicht

Setzt die äußere Kontur eines Polygons. Intern ist diese Methode identisch mit dem Aufruf der Methode SetHoleContur mit einem Loch-Index von 0.

Parameter:

numPoints	Anzahl der Punkte, die im Parameter points tatsächlich belegt sind
points	Ein Array mit Punktkoordinaten (s.u.)

Der Parameter points ist ein OLE-SafeArray mit zwei Dimensionen, die erste Dimension enthält fünf Werte – die Koordinaten x, y, z sowie die Texturkoordinaten u und v (in dieser Reihenfolge), die zweite Dimension enthält soviele Werte, wie Sie zum übergeben der Punkte benötigen.

In Basic können Sie ein solches Array folgendermaßen deklarieren:

```
Dim contur(4, 3) As Single
```

Um die Z-Koordinate des ersten Punktes der Kontur auf 2,5 zu setzen, schreiben Sie dann:

```
contur(2, 0) = 2.5
```

Bei der Übergabe “ verpackt” Basic dieses Array automatisch als “ VARIANT” , Sie können also direkt

```
constr.SetContur 4, contur
```

aufrufen.

In C++ ist der Zugriff etwas umständlicher.

Zunächst deklarieren Sie einen Hilfstyp zur Aufnahme der Punktkoordinaten:

```
struct Point {
  float x, y, z, u, v;
};
```

Dann schreiben Sie eine kleine Hilfsfunktion, um eine Kontur in ein SafeArray zu verpacken:

```
typedef Point ConturType[4];
static void PointsToVariant(ConturType &contur, VARIANT &v)
{
  // Das C-Array in ein OLE SafeArray konvertieren
  SAFEARRAY * array;
  SAFEARRAYBOUND bounds[2];
  bounds[0].lLbound = 0;
  bounds[0].cElements = 5;
  bounds[1].lLbound = 0;
  bounds[1].cElements = 4;
  array = SafeArrayCreate(VT_R4, 2, bounds);
  void * mem = NULL;
  SafeArrayAccessData(array, &mem);
  memcpy(mem, contur, sizeof contur);
  SafeArrayUnaccessData(array);

  VariantInit(&v);
```

```
  v.vt = VT_ARRAY|VT_R4;
  v.parray = array;
}
```

Die genauen Parameter und der typedef für ConturType sind natürlich individuelle Stilfrage und vom Kontext abhängig. Nachdem Sie eine Kontur (vom oben definierten Typ ConturType) mit Daten gefüllt haben, rufen Sie diese Methode folgendermaßen auf:

```
  VARIANT v;
  PointsToVariant(contur, v);
  constr->SetContur(4, v);
  VariantClear(&v);
```

ObjectConstructor.SetHoleContur

Deklaration:

```
Sub SetHoleContur(ByVal holeIndex as short, ByVal
numPoints as short, ByVal points as VARIANT)
```

```
HRESULT SetHoleContur(short holeIndex, short numPoints,
VARIANT points);
```

Siehe auch: ObjectConstructor Übersicht

In einem zweidimensionalen Array werden x, y, z Koordinaten sowie die Texturkoordinaten u und v für alle Punkte eine Kontur übergeben. Das erste Loch im Polygon hat den Loch-Index 1, die Kontur 0 ist die äußere Kontur des Polygons.

Parameter:

holeIndex	Die Nummer des Lochs (holeIndex > 0) oder der Kontur (holeIndex = 0)
numPoints	Anzahl der Punkte, die im Parameter points tatsächlich belegt sind
points	Ein Array mit Punktkoordinaten. Details über diesen Parameter finden Sie in der Beschreibung der Methode SetContur.

ObjectConstructor.SetHolePoint

Deklaration:

```
Sub SetHolePoint(ByVal holeIndex as short, ByVal pointIndex
as short, ByVal x as Single, ByVal y as Single, ByVal z as
Single, ByVal u as Single, ByVal v as Single)
```

```
HRESULT SetHolePoint(short holeIndex, short pointIndex,
float x, float y, float z, float u, float v);
```

Siehe auch: ObjectConstructor Übersicht

Definiert eine Punkt für eine beliebige Kontur. Die äußere Kontur (deren Koordinaten in der Regel mit SetPoint definiert werden) hat den Index 0, die Löcher innerhalb dieser Kontur haben fortlaufende Indizes beginnend bei 1. Diese Methode kann Punkte für beliebige Konturen setzen, ein Aufruf von SetPoint ist in der Tat identisch mit einem Aufruf dieser Methode mit dem Loch-Index 0.

Parameter sind:

holeIndex	Die Nummer der Kontur, für die ein Punkt definiert wird (1 = erstes Loch)
pointIndex	Die laufende Nummer (beginnend bei 0) des Punktes in dieser Kontur
x, y, z	Die Koordinaten des Punktes im Objektkoordinatensystem
u, v	Die Texturkoordinaten des Punktes

ObjectConstructor.SetPoint

Deklaration:

```
Sub SetPoint(ByVal index as short, ByVal x as Single,
ByVal y as Single, ByVal z as Single, ByVal u as Single,
ByVal v as Single)
```

```
HRESULT SetPoint(short index, float x, float y, float z,
float u, float v);
```

Siehe auch: ObjectConstructor Übersicht

Definiert einen Punkt. Für ein Dreieck benötigen Sie drei Punkte mit den Indizes 0 bis 2, für ein Viereck entsprechend vier Punkte mit den Indizes 0 bis 3. Für ein Polygon können Sie beliebig viele Punkte definieren, die Indizes müssen aber mit 0 beginnen und fortlaufend numeriert sein.

Die Parameter sind:

index	Der Punktindex des definierten Punktes
x, y, z	Die Koordinaten des Punktes im Objektkoordinatensystem
u, v	Die Texturkoordinaten des Punktes

IDHistory

Dieses Objekt beschreibt die Änderung einer Objekt-ID (siehe “ Eindeutige Kennungen”). ArCon verwaltet Ojekt-Ids projektlokal eindeutig. Sobald Objekte aus verschiedenen Projekten in ein Projekt importiert werden (durch nachträgliches Laden einzelner Gebäude) erhalten die betroffenen Objekte eine neue ID, da die ID aus dem alten Projekt im neuen zu Kollisionen führen kann. Ein so “ umnumeriertes” Objekt liefert als Eigenschaft “ History” ein IDHistory-Objekt, dem Sie eine eindeutige Zuordnung der alten ID zur neuen entnehmen können.

IDHistory Übersicht

Eigenschaften

CurrentID	Aktuelle ID des Objektes
ParentObject	Das Objekt, dessen Geschichte durch dieses History-Objekt beschrieben wird
PreviousID	ID des Objektes im vorherigen Projekt
PreviousProjectName	Der Name des Projektes, aus dem dieses Objekt geladen wurde
RecentlyLoaded	Wahr, wenn das zugehörige Objekt im letzten Nachladevorgang geladen wurde

IDHistory.CurrentID

Deklaration:

long (nur lesbar)

HRESULT get_CurrentID(long* retVal);

Siehe auch: IDHistory Übersicht

Die aktuelle ID des Eltern-Objektes.

IDHistory.ParentObject

Deklaration:

Object (nur lesbar)

HRESULT get_ParentObject(IDispatch** retVal);

Siehe auch: IDHistory Übersicht

Liefert das Elternobjekt, dessen ID-Geschichte mit diesem History-Objekt beschrieben wird.

IDHistory.PreviousID

Deklaration:

long (nur lesbar)

HRESULT get_PreviousID(long* retVal);

Siehe auch: IDHistory Übersicht

Die ID des Elternobjektes im vorherigen Projekt.

IDHistory.PreviousProjectName

Deklaration:

String (nur lesbar)

HRESULT get_PreviousProjectName(BSTR* retVal);

Siehe auch: IDHistory Übersicht

Der Dateiname des Projektes, aus dem das Elternobjekt nachgeladen wurde. In diesem Projekt gilt die PreviousID.

IDHistory.RecentlyLoaded

Deklaration:

Boolean (nur lesbar)

HRESULT get_RecentlyLoaded(VARIANT_BOOL* retVal);

Siehe auch: IDHistory Übersicht

Object3D

Ein 3D-Objekt. Alle gleichartigen Objekte haben eine gemeinsame Klasse, ihren ObjectConstructor.

Object3D Übersicht

Eigenschaften

DarfVerzerren	Das Objekt darf verzerrt skaliert werden
Flags	Flags zur Steuerung der 3D/2D Ersatzdarstellung (z.B. AC_3DFL_CONSTMODECOLORED)
Group	Wahr, wenn das Objekt eine Gruppe ist
History	Informationen über frühere ID in anderen Projekten
ID	Eindeutige Kennung dieser Instanz
KeineSchatten	Das Objekt verursacht keine Schatten
LampenBeiNachtAn	In der Nachtansicht oder nachts in der zeitabhängigen Sicht gehen die Lampen an
LampenSindAn	Lampen (leuchtende Materialien im Objekt) sind an
LichtEinfluss	Reichweite des ausgestrahlten Lichtes
LichtFarbe	Farbe des Lichtes (bei selbstleuchtenden Objekten)
LichtIntensitaet	Helligkeit des ausgestrahlten Lichtes
LichtMachtSchatten	Das von diesem Objekt (Lampe) ausgestrahlte Licht verursacht Schatten
LokalAnTerrainAnpassen	Das Objekt paßt sich der Geländeneigung an
Name	Logischer Name der Instanz
ObjectConstructor	Objektklasse, aus der diese Instanz erzeugt wurde
ParentObject	Falls das Objekt gruppiert ist, die Gruppe
RoomObject	Das Objekt ist Bestandteil der Raumdarstellung und kein Einrichtungsgegenstand. Es wirft in der Tagesansicht keine Schatten.
SollFallen	Das Objekt fällt nach unten
SubObjects	Liste aller Subobjekte, falls dies eine Gruppe ist

Methoden

Building	Liefert das Gebäude, in dem das Objekt sich befindet.
DatabaseInfo	Liefert Informationen über die Datenbankidentität des Objektes, falls es aus einer Datenbank stammt.
Delete	Löscht die Instanz
Deselect	Hebt die Selektion des Objektes auf
GetBoundingBox	Liefert die Umhüllung des Objektes
GetExtend	Liefert die Größe des Objektes (Original/aktuelle Skalierung)
GetExtendedFlags	Liefert die erweiterten Geometrie-Flags (z.B. ACGPL_NichtLoeschbar)
GetGUID	Liefert den Global Unique IDentifier des Objektes, falls es aus einer Datenbank stammt.
GetModelToGroupTransformation	Liefert die Transformationsmatrix von Modellkoordinaten dieser Instanz zu Modellkoordinaten der Gruppe
GetModelToWorldTransformation	Liefert die Transformationsmatrix von Modell- in Weltkoordinaten
GetTextureCollection	Liefert die Liste der von dieser Instanz verwendeten Texturen

GroupAddObject	Fügt das übergebene Objekt einer Gruppe hinzu.
GroupRemoveObject	Entfernt das übergebene Objekt aus einer Gruppe
GroupReplaceObject	Ersetzt ein Objekt in einer Gruppe durch ein anderes
InitSnapObject	Erzeugt Fanginformation zu diesem Objekt
InsertIntoWorld	Fügt das Objekt in die Welt ein (macht es sichtbar)
IsInternalObject	Gibt an, ob das Objekt ein internes ArCon Objekt ist
IsO2CObject	Testet, ob das Objekt als O2C-Objekt geladen wurde
IsSelected	Testet, ob das Objekt derzeit selektiert ist
NotRepublishable	Testet, ob das Objekt als O2C-Objekt gespeichert werden kann
RemoveFromWorld	Entfernt das Objekt aus der ArCon-Welt
ReplaceConstructor	Ersetzt den bisheringen ObjectConstructor dieser Instanz durch einen neuen und ändert damit die Geometrie des Objektes.
Room	Liefert einen der Räume, in dem das Objekt sich befindet (Index kann 1 oder 2 sein)
Select	Selektiert das Objekt
SetExtendedFlags	Ändert die erweiterten Geometrie-Flags (z.B. ACGPL_NichtLoeschbar)
SetModelToWorldTransformation	Setzt die Transformationsmatrix von Modell- in Weltkoordinaten
SetOutline2D	Setzt eine 2D Ersatzdarstellung für das 3D Objekt
Story	Liefert das Stockwerk, in dem sich das Objekt befindet.
WriteToFile	Speichert die Instanz als *.ACO Datei ab

Object3D.DarfVerzerren

Deklaration:

Boolean

HRESULT get_DarfVerzerren(VARIANT_BOOL* retVal);
HRESULT put_DarfVerzerren(VARIANT_BOOL newVal);

Siehe auch: Object3D Übersicht

Wenn diese Eigenschaft Wahr ist, darf das Objekt auch nicht-linear skaliert werden. Ist der Wert Falsch, kann das Seitenverhältnis in x/y/z Richtung durch Skalierung nicht verändert werden.

Object3D.Flags

Deklaration:

long

HRESULT get_Flags(long* retVal);
HRESULT put_Flags(long newVal);

Siehe auch: Object3D Übersicht

Flags (Bits) zur Steuerung verschiedener Objekteigenschaften.

Ein gesetztes Bit bedeutet

Bit	Bedeutung
AC_3DFL_CONSTMODE	Das Objekt ist im Konstruktionsmodus sichtbar (als 2D Ersatzdarstellung)
AC_3DFL_CONSTMODEARCHITECT	Das Objekt ist auch bei Architektengerechter Darstellung im Konstruktionsmodus sichtbar
AC_3DFL_CONSTMODECOLORED	Die 2D Ersatzdarstellung im Konstruktionsmodus ist farbig
AC_3DFL_CONSTMODECUTS	Das Objekt ist in Schnittansichten sichtbar
AC_3DFL_DESIGNMODE	Das Objekt wird im Einrichtungsmodus dargestellt
AC_3DFL_DBLCLICK	Das Objekt ist im Einrichtungsmodus per doppelklick bearbeitbar
AC_3DFL_SHOWALL	Das Objekt wird bei der Berechnung der benötigten Ansichtsgröße beim Befehl “ Alles zeigen” berücksichtigt

Object3D.Group

Deklaration:

Boolean (nur lesbar)

HRESULT get_Group(VARIANT_BOOL* retVal);

Siehe auch: Object3D Übersicht

Wahr, falls dieses Objekt Bestandteil einer Gruppe ist. In diesem Fall können Sie das übergeordnete Objekt als ParentObject erfragen.

Object3D.History

Deklaration:

IDHistory (nur lesbar)

HRESULT get_History(IIDHistory** retVal);

Siehe auch: Object3D Übersicht, IDHistory

Falls dieses Object3D beim Ursprünglichen Laden (oder Erstellen) des aktuellen Projektes nicht in diesem Projekt war, sonder aus einem anderen Projekt nachgeladen wurde, liefert diese Eigenschaft das History-Objekt, das die Umsetzung der IDs beschreibt. Andernfalls ist diese Eigenschaft “ Nothing” .

Ein Anwendungsbeispiel für History-Objekte finden Sie im Kapitel Eindeutige Kennungen.

Object3D.ID

Deklaration:

long (nur lesbar)

HRESULT get_ID(long* retVal);

Siehe auch: Object3D Übersicht

Gibt eine eindeutige Kennung dieses Object3D an. Diese Kennung ist persistent, das heißt, sie bleibt innerhalb des aktuellen Projektes erhalten. Sie können diese Kennung benutzen, um Object3Ds miteinander zu vergleichen oder zusätzliche Daten zum Object3D in einer Datenbank abzulegen.

Die ID ist aber nur innerhalb des aktuellen Projektes eindeutig, wenn Sie (durch Nachladen einzelner Gebäude) das Objekt in ein anderes Projekt einfügen, erhält es eine neue ID. In diesem Fall müssen Sie sofort bei Beendigung des Ladevorganges mit Hilfe der Eigenschaft History objPropertyDetailsObject3DHistory Ihre zusätzlichen Daten an die neuen Werte anpassen.

Object3D.KeineSchatten

Deklaration:

Boolean

HRESULT get_KeineSchatten(VARIANT_BOOL* retVal);

HRESULT put_KeineSchatten(VARIANT_BOOL newVal);

Siehe auch: Object3D Übersicht

Wahr, wenn das Objekt keine Schatten wirft.

Object3D.LampenBeiNachtAn

Deklaration:

Boolean

HRESULT get_LampenBeiNachtAn
(VARIANT_BOOL* retVal);

HRESULT put_LampenBeiNachtAn
(VARIANT_BOOL newVal);

Siehe auch: Object3D Übersicht

Wahr, wenn selbstleuchtende Materialien des Objektes (die Lampen) bei Nachtsicht automatisch eingeschaltet werden bzw. bei zeitabhängiger Sicht entsprechend der Uhrzeit an- und ausgeschaltet werden.

Object3D.LampenSindAn

Deklaration:

Boolean

HRESULT get_LampenSindAn(VARIANT_BOOL* retVal);

HRESULT put_LampenSindAn(VARIANT_BOOL newVal);

Siehe auch: Object3D Übersicht

Wahr, wenn die selbstleuchtenden Materialien (die Lampen) des Objektes permanent angeschaltet sind.

Object3D.LichtEinfluss

Deklaration:

Single

HRESULT get_LichtEinfluss(float* retVal);
HRESULT put_LichtEinfluss(float newVal);

Siehe auch: Object3D Übersicht

Ein Faktor, der die Reichweite des von diesem Objekt ausgestrahlten Lichtes beschreibt.

Object3D.LichtFarbe

Deklaration:

long

HRESULT get_LichtFarbe(long* retVal);
HRESULT put_LichtFarbe(long newVal);

Siehe auch: Object3D Übersicht

Die Farbe des von diesem Objekt abgestrahlten Lichtes.

Object3D.LichtIntensitaet

Deklaration:

Single

HRESULT get_LichtIntensitaet(float* retVal);
HRESULT put_LichtIntensitaet(float newVal);

Siehe auch: Object3D Übersicht

Die Helligkeit des abgestrahlten Lichtes.

Object3D.LichtMachtSchatten

Deklaration:

Boolean

HRESULT get_LichtMachtSchatten
(VARIANT_BOOL* retVal);

HRESULT put_LichtMachtSchatten
(VARIANT_BOOL newVal);

Siehe auch: Object3D Übersicht

Wahr, wenn das von diesem Objekt ausgestrahlte Licht Schatten wirft.

Object3D.LokalAnTerrainAnpassen

Deklaration:

Boolean

HRESULT get_LokalAnTerrainAnpassen
(VARIANT_BOOL* retVal);

HRESULT put_LokalAnTerrainAnpassen
(VARIANT_BOOL newVal);

Siehe auch: Object3D Übersicht

Wahr, wenn das Objekt auf Terrainhöhe fällt, also “ auf dem Boden” steht.

Falsch, wenn es seine Höhe relativ zu einem Gebäude behält, also z.B. am Gebäude befestigt ist.

Object3D.Name

Deklaration:

String

HRESULT get_Name(BSTR* retVal);
HRESULT put_Name(BSTR newVal);

Siehe auch: Object3D Übersicht

Der logische Name des Objektes. Nicht zu verwechseln mit dem Dateinamen (ObjectConstructor.FileName) des zugehörigen ObjectConstructors (der ACO-Datei).

Object3D.ObjectConstructor

Deklaration:

ObjectConstructor (nur lesbar)

```
HRESULT get_ObjectConstructor
(IObjectConstructor** retVal);
```

Siehe auch: Object3D Übersicht, ObjectConstructor

Die Klasse des Objektes, der ObjectConstructor, durch den dieses Objekt instanziiert wurde.

Object3D.ParentObject

Deklaration:

Object (nur lesbar)

HRESULT get_ParentObject(IDispatch** retVal);

Siehe auch: Object3D Übersicht

Das Elternobjekt, wenn dieses Objekt ein Sub-Objekt (bzw. Teil einer Gruppe, siehe Group) ist.

Object3D.RoomObject

Deklaration:

Boolean (nur lesbar)

HRESULT get_RoomObject(VARIANT_BOOL* retVal);

Siehe auch: Object3D Übersicht

Wahr, wenn das Objekt zum Raum gehört, Falsch bei Einrichtungsgegenständen.

Raumobjekte sind zum Beispiel Türen, Fensterbänke und ähnliches.

Object3D.SollFallen

Deklaration:

Boolean

HRESULT get_SollFallen(VARIANT_BOOL* retVal);

HRESULT put_SollFallen(VARIANT_BOOL newVal);

Siehe auch: Object3D Übersicht

Wahr, wenn das Objekt nach unten (Stuhl) oder nach oben (Lampe) fällt, Falsch wenn es seine Positionierungshöhe behällt (Wandschrank).

Object3D.SubObjects

Deklaration:

	Object3DCollection (nur lesbar)
	HRESULT get_SubObjects(IObject3DCollection** retVal);

Siehe auch: Object3D Übersicht, Object3DCollection

Liste aller Sub-Objekte dieses Objektes (z.B. wenn es sich um eine Gruppe handelt, siehe Group)
.

Object3D.Building

Deklaration:

```
Function Building() as Building
```

```
HRESULT Building(IBuilding** retVal);
```

Siehe auch: Object3D Übersicht, Building

Object3D.DatabaseInfo

Deklaration:

```
Function DatabaseInfo(ByRef dbID as long, ByRef
objectID as long) as Boolean
```

```
HRESULT DatabaseInfo(long * dbID, long *
objectID, VARIANT_BOOL* retVal);
```

Siehe auch: Object3D Übersicht

Object3D.Delete

Deklaration:

```
Function Delete(ByVal withSubObjects as Boolean)
as Boolean
```

```
HRESULT Delete(VARIANT_BOOL
withSubObjects, VARIANT_BOOL* retVal);
```

Siehe auch: Object3D Übersicht

Löscht das 3D-Objekt.

Parameter:

withSubObjects	Wenn dieser Wert Wahr ist, werden auch die Sub-Objekte gelöscht

Object3D.Deselect

Deklaration:

```
Function Deselect() as Boolean
```

```
HRESULT Deselect(VARIANT_BOOL* retVal);
```

Siehe auch: Object3D Übersicht

Object3D.GetBoundingBox

Deklaration:

```
Function GetBoundingBox(ByRef minX as Single,
ByRef minY as Single, ByRef minZ as Single, ByRef
maxX as Single, ByRef maxY as Single, ByRef maxZ
as Single) as Boolean
```

```
HRESULT GetBoundingBox(float* minX, float* minY,
float* minZ, float* maxX, float* maxY, float* maxZ,
VARIANT_BOOL* retVal);
```

Siehe auch: Object3D Übersicht

Liefert die Bounding-Box des Objektes.

Parameter:

minX, minY, minZ	Ergebnisparameter: eine Ecke der Bounding-Box
maxX, maxY, maxZ	Ergebnisparameter: gegenüberliegende Seite der Bounding-Box

Object3D.GetExtend

Deklaration:

```
Sub GetExtend(ByVal scaled as Boolean,
ByRef dx as Single, ByRef dy as Single,
ByRef dz as Single)
```

```
HRESULT GetExtend(VARIANT_BOOL
scaled, float * dx, float * dy, float * dz);
```

Siehe auch: Object3D Übersicht

Object3D.GetExtendedFlags

Deklaration:

```
Function GetExtendedFlags() as long
```

```
HRESULT GetExtendedFlags(long* retVal);
```

Siehe auch: Object3D Übersicht

Liefert die erweiterten Eigenschafte des Objektes (siehe Erweiterte Designobjekteigenschaften).

Object3D.GetGUID

Deklaration:

```
Function GetGUID() as String
```

```
HRESULT GetGUID(BSTR* retVal);
```

Siehe auch: Object3D Übersicht

Object3D.GetModelToGroupTransformation

Deklaration:

```
Function GetModelToGroupTransformation
(ByRef matrix as VARIANT) as Boolean
```

```
HRESULT GetModelToGroupTransformation
(VARIANT* matrix, VARIANT_BOOL* retVal)
;
```

Siehe auch: Object3D Übersicht

Ermittelt die Transformationsmatrix, mit der von Modell- in Gruppenkoordinaten umgerechnet wird.

Parameter:

matrix	Eine 4x4 Matrix aus einfach genauen Fließkommazahlen.

Object3D.GetModelToWorldTransformation

Deklaration:

```
Function GetModelToWorldTransformation(ByRef
matrix as VARIANT) as Boolean
```

```
HRESULT GetModelToWorldTransformation
(VARIANT* matrix, VARIANT_BOOL* retVal);
```

Siehe auch: Object3D Übersicht

Liefert die Matrix, mit deren Hilfe von Modell- in Weltkoordinaten umgerechnet wird.

Parameter:

matrix	Eine 4x4 Matrix aus einfach genauen Fließkommazahlen.

Object3D.GetTextureCollection

Deklaration:

```
Function GetTextureCollection() as
TextureCollection
```

```
HRESULT GetTextureCollection
(ITextureCollection** retVal);
```

Siehe auch: Object3D Übersicht, TextureCollection

Liefert die Texturliste des Objektes.

Dabei handelt es sich um die aktive Liste der Objektinstanz – durch Zuweisung eines Texturnamens an ein Element dieser Liste tauschen Sie die entsprechende Textur des Objektes aus.

Unter welchem Index der Liste sich welche Textur verbirgt, ist von der Modellierung des 3D Objektes abhängig.

Object3D.GroupAddObject

Deklaration:

```
Function GroupAddObject(ByVal objectToAdd as
Object3D) as Boolean
```

```
HRESULT GroupAddObject(IObject3D *
objectToAdd, VARIANT_BOOL* retVal);
```

Siehe auch: Object3D Übersicht, Object3D

Fügt ein neues Objekt einer existierenden Gruppe hinzu.

Die Parameter sind:

objectToAdd	Das neu hinzuzufügende Objekt

Object3D.GroupRemoveObject

Deklaration:

```
Function GroupRemoveObject(ByVal
objectToRemove as Object3D, ByVal deleteOldObject
as Boolean) as Boolean
```

```
HRESULT GroupRemoveObject(IObject3D *
objectToRemove, VARIANT_BOOL deleteOldObject,
VARIANT_BOOL* retVal);
```

Siehe auch: Object3D Übersicht, Object3D

Entfernt ein Teilobjekt aus einer Gruppe.

Die Parameter sind:

objectToRemove	Das aus der Gruppe zu entfernende Objekt
deleteOldObject	“ Wahr” wenn das entfernte Objekt gelöscht werden soll, “ Falsch” wenn es noch erhalten bleiben soll. Im letzteren Fall kann es mit InsertIntoWorld eigenständig in die Welt eingefügt werden.

Object3D.GroupReplaceObject

Deklaration:

Function GroupReplaceObject(ByVal objectToRemove as Object3D, ByVal newObject as Object3D, ByVal deleteOldObject as Boolean) as Boolean

HRESULT GroupReplaceObject(IObject3D * objectToRemove, IObject3D * newObject, VARIANT_BOOL deleteOldObject, VARIANT_BOOL* retVal);

Siehe auch: Object3D Übersicht, Object3D, Object3D

Ersetzt ein Teilobjekt einer Gruppe durch ein anderes.

Die Parameter sind:

objectToRemove	Das aus der Gruppe zu entfernende Objekt
newObject	Das Ersatzobjekt
deleteOldObject	“ Wahr” wenn das entfernte Objekt gelöscht werden soll, “ Falsch” wenn es noch erhalten bleiben soll. Im letzteren Fall kann es mit InsertIntoWorld eigenständig in die Welt eingefügt werden.

Object3D.InitSnapObject

Deklaration:

```
Function InitSnapObject(ByVal storeInProject as
Boolean) as Boolean
```

```
HRESULT InitSnapObject(ISnapObject *snapObj,
VARIANT_BOOL storeInProject, VARIANT_BOOL*
retVal);
```

Siehe auch: Object3D Übersicht

Initialisiert die Fang-Daten dieses Objektes aus einer Fangobjekt-Beschreibung. Anschließend fängt das Objekt nicht mehr wie in seiner ACO-Beschreibung vorgegeben bzw. mit seiner Bounding-Box, sondern gemäß den Anweisungen aus dem SnapObject.

Die Parameter sind:

snapObj	Die Beschreibung der Fangeigenschaften
storeInProject	“ Wahr” wenn die Fangbeschreibung mit in der Projektdatei (ACP) gespeichert werden soll.

Object3D.InsertIntoWorld

Deklaration:

```
Function InsertIntoWorld(ByVal selected as Boolean)
as Boolean
```

```
HRESULT InsertIntoWorld(VARIANT_BOOL selected,
VARIANT_BOOL* retVal);
```

Siehe auch: Object3D Übersicht

Fügt das Objekt in die Welt ein. Vorher muß eine entsprechende Transformationsmatrix gesetzt worden sein!

Parameter:

selected	Wahr, wenn das Objekt beim Einfügen sofort selektiert werden soll.

Ein Beispiel zum positionsgenauen Einfügen von Einrichtungsgegenständen in ein Projekt finden Sie im Beispielprogramm Badewanne.

Statt ein Objekt in die ArCon-Welt einzufügen (und damit ArCon die Verwaltung des Objektes zu überlassen) können Sie ein Objekt auch anzeigen, indem Sie es an eine 2D-Ersatzdarstellung binden (siehe SetOutline2D).

Object3D.IsInternalObject

Deklaration:

```
Function IsInternalObject() as Boolean
```

```
HRESULT IsInternalObject(VARIANT_BOOL*
retVal);
```

Siehe auch: Object3D Übersicht

Object3D.IsO2CObject

Deklaration:

```
Function IsO2CObject() as Boolean
```

```
HRESULT IsO2CObject(VARIANT_BOOL* retVal);
```

Siehe auch: Object3D Übersicht

Object3D.IsSelected

Deklaration:

```
Function IsSelected() as Boolean
```

```
HRESULT IsSelected(VARIANT_BOOL* retVal);
```

Siehe auch: Object3D Übersicht

Object3D.NotRrepublishable

Deklaration:

```
Function NotRepublishable() as Boolean
```

```
HRESULT NotRepublishable(VARIANT_BOOL*
retVal);
```

Siehe auch: Object3D Übersicht

Object3D.RemoveFromWorld

Deklaration:

```
Function RemoveFromWorld() as Boolean
```

```
HRESULT RemoveFromWorld(VARIANT_BOOL*
retVal);
```

Siehe auch: Object3D Übersicht

Object3D.ReplaceConstructor

Deklaration:

```
Function ReplaceConstructor(ByVal newGeometry
as ObjectConstructor, ByVal textureList as
TextureCollection, ByVal isRoomObject as
Boolean) as Boolean
```

```
HRESULT ReplaceConstructor(IObjectConstructor
* newGeometry, ITextureCollection* textureList,
VARIANT_BOOL isRoomObject, VARIANT_BOOL*
retVal);
```

Siehe auch: Object3D Übersicht, ObjectConstructor, TextureCollection

Wechselt die Geometrie des Objektes aus, indem es einem anderen ObjectConstructor zugeordnet wird.

Die Parameter sind:

newGeometry	Der neue ObjectConstructor
textureList	Eine Texturlist (oder Nothing, wenn die Standard-Texturen verwendet werden sollen)
isRoomObject	“ Wahr” wenn das Objekt zum Raum gehört (als Bestandteil der Konstruktion, z.B. Fensterbänke).

Object3D.Room

Deklaration:

```
Function Room(ByVal index as long) as Room
```

```
HRESULT Room(long index, IRoom** retVal);
```

Siehe auch: Object3D Übersicht, Room

Object3D.Select

Deklaration:

```
Function Select(ByVal exclusive as Boolean) as
Boolean
```

```
HRESULT Select(VARIANT_BOOL exclusive,
VARIANT_BOOL* retVal);
```

Siehe auch: Object3D Übersicht

Object3D.SetExtendedFlags

Deklaration:

```
Sub SetExtendedFlags(ByVal flags as long)
```

```
HRESULT SetExtendedFlags(long flags);
```

Siehe auch: Object3D Übersicht

Ändert die erweiterten Eigenschafte des Objektes (siehe Erweiterte Designobjekteigenschaften).

Object3D.SetModelToWorldTransformation

Deklaration:

```
Function SetModelToWorldTransformation(ByVal
matrix as VARIANT) as Boolean
```

```
HRESULT SetModelToWorldTransformation
(VARIANT matrix, VARIANT_BOOL* retVal);
```

Siehe auch: Object3D Übersicht

Setzt die neue Transformationsmatrix des Objektes. Dieses ändert dadurch nicht sofort seine sichtbare Position, zunächst muß eine entsprechende Aktualisierung der 3D Ansichten ausgelöst werden, z.B. durch Aufruf von Redraw3DViews.

Parameter:

matrix	Eine 4x4 Matrix aus einfach genauen Fließkommazahlen.

Diese Methode wird auch benuzt, um Objekten vor dem Einfügen in die ArCon-Welt eine Position zuzuweisen, ein Beispiel hierzu finden Sie im Beispielprogramm Badewanne.

Object3D.SetOutline2D

Deklaration:

```
Sub SetOutline2D(ByVal obj2D as Object, ByVal
cursor as long, ByVal hint as String)
```

```
HRESULT SetOutline2D(IDispatch* obj2D, long
cursor, BSTR hint);
```

Siehe auch: Object3D Übersicht

Ordnet ein 2D Grafikelement (z.B. ein Polygon oder eine Beschriftung) dem 3D Element als Ersatzdarstellung zu. Das 2D Grafikelement muß vorher in eine der Grafiklisten eingetragen sein (um angezeigt zu werden). Es wird durch die Zuordnung nicht beeinflußt, verhält sich also weiterhin genau wie ein 2D Grafikelement.

Die Zuordnung eines 3D-Objektes zu einer 2D-Ersatzdarstellung und damit die manuelle Steuerung der Sichtbarkeit des 3D-Objektes über das zugeordnete Träger-2D-Objekt ist die Alternative zum Einfügen des Objektes in die Welt (siehe InsertIntoWorld).

Bei an 2D-Träger gebundenen 3D-Objekten hat Ihr Programm volle Kontrolle über alle Benutzeraktionen. Zum Beispiel wird beim Doppelklick auf ein solches Objekt nicht der Standard-Objektdialog geöffnet, sondern ein Ereignis (Object3DDoubleClicked) ausgelöst. Ihr Programm kann darauf mit einem eigenen Dialog reagieren – oder das Ereignis ignorieren.

Object3D.Story

Deklaration:

```
Function Story() as Story
```

```
HRESULT Story(IStory** retVal);
```

Siehe auch: Object3D Übersicht, Story

Object3D.WriteToFile

Deklaration:

```
Function WriteToFile(ByVal FileName as String,
ByVal centerX as Single, ByVal centerY as Single,
ByVal centerZ as Single) as Boolean
```

```
HRESULT WriteToFile(BSTR FileName, float
centerX, float centerY, float centerZ,
VARIANT_BOOL* retVal);
```

Siehe auch: Object3D Übersicht

Speichert die Instanz als ACO-Datei ab.

Parameter:

FileName	Der Dateiname der zu erzeugenden ACO Datei
centerX, centerY, centerZ	Verschiebungsvektor, um den beim Einfügen eines Objektes aus der neuen ACO Datei das Zentrum der Modelkoordinaten gegenüber den Weltkoordinaten verschoben wird.

ObjectTransformer

Bei der Animation von 3D Objekten in ArCon werden in jedem Animationsschritt allen betroffenen 3D-Objekten neue Abbildungsmatritzen von Modell- in Weltkoordinaten zugewiesen. Ein ObjectTransformer entspricht einem Animateur-Objekt und speichert genau einen solchen Animationsschritt für ein Objekt: er enthält einen Verweis auf das zu animierende 3D-Objekt sowie die Abbildungsmatrix, die dieses Objekt im nächsten Animationsschritt erhalten soll.

Sie können mit Hilfe einer Liste von ObjectTransformern (ObjectTransformerCollection) viele Objekte auf den nächsten Animationsschritt vorbereiten und diesem dann alle Objekte gemeinsam ausführen lassen. Dabei wird die gesamte 3D Ansicht nur einmal neu berechnet, sodaß dieses Verfahren deutlich schneller ist, als alle betroffenen Objekte einzeln (mit Hilfe von SetModelToWorldTransformation) an eine neuen Position zu bewegen.

Die mit Hilfe von ObjectTransformern erzeugten Animationen sind nicht zu verwechseln mit den Eigenbewegungen von Objekten – es handelt sich lediglich um eine dem Objekt von außen aufgezwungene Verschiebung, Drehung oder Vergrößerung, wie der Benutzer sie manuell auch hätte durchführen können.

ObjectTransformer Übersicht

Eigenschaften

Matrix	Neue Transformationsmatrix von Modell- in Weltkoordinaten
Object3D	Die animierte 3D-Objekt Instanz

ObjectTransformer.Matrix

Deklaration:

VARIANT

HRESULT get_Matrix(VARIANT* retVal);
HRESULT put_Matrix(VARIANT newVal);

Siehe auch: ObjectTransformer Übersicht

Eine 4x4 Matrix aus einfach genauen Fließkommazahlen, die in links-prefix-Multiplikation die Transformation des Objektes von seinen Modellkoordinaten in Weltkoordinaten definiert. Der Variant muß genau diesen Typ haben, ansonsten wird die Fehlermeldung Parameter ist keine 4x4 Matrix... ausgelöst. Unter dieser Fehlermeldung finden Sie auch weitere Hinweise zur Kodierung solcher Matritzen in verschiedenen Programmiersprachen.

ObjectTransformer.Object3D

Deklaration:

Object3D

HRESULT get_Object3D(IObject3D** retVal);
HRESULT put_Object3D(IObject3D* newVal);

Siehe auch: ObjectTransformer Übersicht, Object3D

Das Objekt, das von diesem ObjectTransformer animiert wird.

ObjectTransformerCollection

Eine Liste von Animateur-Objekten. In dieser Liste befinden sich eine Reihe von 3D-Objekten, die gleichzeitig animiert werden sollen und einer jeweils zugehörigen Matrix, die den nächsten Animationsschritt für das jeweilige Objekt beschreibt.

Durch die Zusammenfassung der einzelnen Animationsschritte zu einer Gesamtanimation in dieser Liste ist ArCon in der Lage, einige zeitaufwendige Vorgänge bei der Bildaktualisierung optimiert nur einmal für die gesamte Liste statt mehrfach für jedes Objekt durchzuführen.

ObjectTransformerCollection Übersicht

Eigenschaften

Count	Anzahl der animierten Objekte

Methoden

Add	Fügt eine Objektinstanz mit zugehöriger Transformation der Liste hinzu
AddTransformer	Fügt einen fertigen Animateur der Liste hinzu
Item	Liefert den i-ten Animateur
Update	Führt den Animationsschritt aus

ObjectTransformerCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: ObjectTransformerCollection Übersicht, Listen

Die Anzahl der ObjectTransformer in dieser Liste.

ObjectTransformerCollection.Add

Deklaration:

```
Function Add(ByVal anObject as Object3D, ByVal aMatrix as
VARIANT) as Boolean
```

```
HRESULT Add(IObject3D* anObject, VARIANT aMatrix,
VARIANT_BOOL* retVal);
```

Siehe auch: ObjectTransformerCollection Übersicht, Object3D, Listen

Fügt ein 3D-Objekt und die Transformationsmatrix für den nächsten Animationsschritt in diese Liste ein.

Parameter:

anObject	das zu animierende 3D Objekt
aMatrix	die Transformationsmatrix für dieses Objekt

ObjectTransformerCollection.AddTransformer

Deklaration:

```
Function AddTransformer(ByVal aTransformer as
ObjectTransformer) as Boolean
```

```
HRESULT AddTransformer(IObjectTransformer*
aTransformer, VARIANT_BOOL* retVal);
```

Siehe auch: ObjectTransformerCollection Übersicht, ObjectTransformer, Listen

Fügt einen fertigen Animateur in diese Liste ein.

Parameter:

aTransformer	der neue Animateur

ObjectTransformerCollection.Item

Deklaration:

```
Function Item(ByVal Index as long) as
ObjectTransformer
```

```
HRESULT Item(long Index, IObjectTransformer**
retVal);
```

Siehe auch: ObjectTransformerCollection Übersicht, ObjectTransformer, Listen

Liefert einen Animateur aus dieser Liste.

Parameter:

Index	Nummer des gewünschten Animateurs. Das erste Element der Liste hat die Numer 1

ObjectTransformerCollection.Update

Deklaration:

```
Function Update(ByVal dropAfterMove as Boolean) as
Boolean
```

```
HRESULT Update(VARIANT_BOOL dropAfterMove,
VARIANT_BOOL* retVal);
```

Siehe auch: ObjectTransformerCollection Übersicht, Listen

Führt den Animationsschritt für alle Animateure dieser Liste aus.

Parameter:

dropAfterMove	Wahr, wenn alle Objekte nach der Transformation fallen (falls für die betroffenen Objekte Schwerkraft eingestellt ist). Falsch, wenn alle Objekte genau an der Zielposition positioniert werden.

UnterUeberzug

Dieses Objekt stellt einen Unterzug oder einen Überzug da.

UnterUeberzug Übersicht

Eigenschaften

DeckelTextur	Textur des Deckels
Height	Höhe des Zuges
History	Geschichte des Unter/Überzuges beim Gebäudeimport
ID	Eindeutige Kennung dieses Unter/Überzuges
Length	Länge des Zuges
Schraffur	Art der Schraffur in der 2D Darstellung
SchraffurFarbe	Farbe der Schraffur in der 2D Darstellung
Seg1LinienFarbe	Linienfarbe des 1. Segmentes
Seg1LinienTyp	Linienstil des 1. Segmentes
Seg1Textur	Textur des 1. Segmentes
Seg2LinienFarbe	Linienfarbe des 2. Segmentes
Seg2LinienTyp	Linienstil des 2. Segmentes
Seg2Textur	Textur des 2. Segmentes
Seg3LinienFarbe	Linienfarbe des 3. Segmentes
Seg3LinienTyp	Linienstil des 3. Segmentes
Seg3Textur	Textur des 3. Segmentes
Seg4LinienFarbe	Linienfarbe des 4. Segmentes
Seg4LinienTyp	Linienstil des 4. Segmentes
Seg4Textur	Textur des 4. Segmentes
Story	Geschoß, zu dem der Zug gehört
Thickness	Dicke des Zuges
Ueberzug	Wahr, wenn es sich um einen Überzug handelt

Methoden

Delete	Löscht den Unter/Überzug
GetPos	Liefert die Position des Unter/Überzuges
SetPos	Verändert die Position des Unter/Überzuges

UnterUeberzug.DeckelTextur

Deklaration:

Texture

HRESULT get_DeckelTextur(ITexture** retVal);
HRESULT put_DeckelTextur(ITexture* newVal);

Siehe auch: UnterUeberzug Übersicht, Texture

Die Textureigenschaften der Abdeckung des Unterzuges bzw. des unteren Deckels des Überzuges.

UnterUeberzug.Height

Deklaration:

Single

HRESULT get_Height(float* retVal);
HRESULT put_Height(float newVal);

Siehe auch: UnterUeberzug Übersicht

Die Höhe des Unter- oder Überzuges.

UnterUeberzug.History

Deklaration:

IDHistory (nur lesbar)

HRESULT get_History(IIDHistory** retVal);

Siehe auch: UnterUeberzug Übersicht, IDHistory

UnterUeberzug.ID

Deklaration:

long (nur lesbar)

HRESULT get_ID(long* retVal);

Siehe auch: UnterUeberzug Übersicht

UnterUeberzug.Length

Deklaration:

Single

HRESULT get_Length(float* retVal);
HRESULT put_Length(float newVal);

Siehe auch: UnterUeberzug Übersicht

Die Länge des Unter- oder Überzuges.

UnterUeberzug.Schraffur

Deklaration:

long

HRESULT get_Schraffur(long* retVal);
HRESULT put_Schraffur(long newVal);

Siehe auch: UnterUeberzug Übersicht

Die Schraffierungsvariante der 2D Darstellung.

Mögliche Werte:

0	nicht schraffiert
1	einfach umrandet
2	senkrecht schraffiert
3	links oben nach rechts unten schraffiert
4	rechts oben nach links unten schraffiert
5	senkrecht gekreuzt schraffiert
6	diagonal gekreuzt schraffiert
7	vollständig gefüllt

UnterUeberzug.SchraffurFarbe

Deklaration:

long

HRESULT get_SchraffurFarbe(long* retVal);
HRESULT put_SchraffurFarbe(long newVal);

Siehe auch: UnterUeberzug Übersicht

Farbe der Schraffur als RGB-Wert.

UnterUeberzug.Seg1LinienFarbe

Deklaration:

long

HRESULT get_Seg1LinienFarbe(long* retVal);
HRESULT put_Seg1LinienFarbe(long newVal);

Siehe auch: UnterUeberzug Übersicht

Farbe der Linie der ersten Seite.

UnterUeberzug.Seg1LinienTyp

Deklaration:

long

HRESULT get_Seg1LinienTyp(long* retVal);
HRESULT put_Seg1LinienTyp(long newVal);

Siehe auch: UnterUeberzug Übersicht

Linien-Typ der ersten Seite.

UnterUeberzug.Seg1Textur

Deklaration:

Texture

HRESULT get_Seg1Textur(ITexture** retVal);
HRESULT put_Seg1Textur(ITexture* newVal);

Siehe auch: UnterUeberzug Übersicht, Texture
Textureigenschaften der ersten Seite.

UnterUeberzug.Seg2LinienFarbe

Deklaration:

long

HRESULT get_Seg2LinienFarbe(long* retVal);
HRESULT put_Seg2LinienFarbe(long newVal);

Siehe auch: UnterUeberzug Übersicht

Linien-Farbe der zweiten Seite.

UnterUeberzug.Seg2LinienTyp

Deklaration:

long

HRESULT get_Seg2LinienTyp(long* retVal);
HRESULT put_Seg2LinienTyp(long newVal);

Siehe auch: UnterUeberzug Übersicht

Linien-Typ der zweiten Seite.

UnterUeberzug.Seg2Textur

Deklaration:

Texture

HRESULT get_Seg2Textur(ITexture** retVal);
HRESULT put_Seg2Textur(ITexture* newVal);

Siehe auch: UnterUeberzug Übersicht, Texture
Textureigenschaften der zweiten Seite.

UnterUeberzug.Seg3LinienFarbe

Deklaration:

long

HRESULT get_Seg3LinienFarbe(long* retVal);
HRESULT put_Seg3LinienFarbe(long newVal);

Siehe auch: UnterUeberzug Übersicht

Linien-Farbe der dritten Seite.

UnterUeberzug.Seg3LinienTyp

Deklaration:

long

HRESULT get_Seg3LinienTyp(long* retVal);
HRESULT put_Seg3LinienTyp(long newVal);

Siehe auch: UnterUeberzug Übersicht

Linien-Typ der dritten Seite.

UnterUeberzug.Seg3Textur

Deklaration:

Texture

HRESULT get_Seg3Textur(ITexture** retVal);
HRESULT put_Seg3Textur(ITexture* newVal);

Siehe auch: UnterUeberzug Übersicht, Texture
Textureigenschaften der dritten Seite.

UnterUeberzug.Seg4LinienFarbe

Deklaration:

long

HRESULT get_Seg4LinienFarbe(long* retVal);
HRESULT put_Seg4LinienFarbe(long newVal);

Siehe auch: UnterUeberzug Übersicht

Linien-Farbe der vierten Seite.

UnterUeberzug.Seg4LinienTyp

Deklaration:

long

HRESULT get_Seg4LinienTyp(long* retVal);
HRESULT put_Seg4LinienTyp(long newVal);

Siehe auch: UnterUeberzug Übersicht

Linien-Typ der vierten Seite.

UnterUeberzug.Seg4Textur

Deklaration:

Texture

HRESULT get_Seg4Textur(ITexture** retVal);
HRESULT put_Seg4Textur(ITexture* newVal);

Siehe auch: UnterUeberzug Übersicht, Texture
Textureigenschaften der vierten Seite.

UnterUeberzug.Story

Deklaration:

Story (nur lesbar)

HRESULT get_Story(IStory** retVal);

Siehe auch: UnterUeberzug Übersicht, Story

Das Stockwerk, in dem sich dieser Unter- oder Überzug befindet.

UnterUeberzug.Thickness

Deklaration:

Single

HRESULT get_Thickness(float* retVal);
HRESULT put_Thickness(float newVal);

Siehe auch: UnterUeberzug Übersicht

Die Dicke des Unter-/Überzuges.

UnterUeberzug.Ueberzug

Deklaration:

Boolean

HRESULT get_Ueberzug(VARIANT_BOOL* retVal);
HRESULT put_Ueberzug(VARIANT_BOOL newVal);

Siehe auch: UnterUeberzug Übersicht

Wahr, wenn es sich um einen Überzug handelt. Falsch bei einem Unterzug.

UnterUeberzug.Delete

Deklaration:

```
Function Delete() as Boolean
```

```
HRESULT Delete(VARIANT_BOOL* retVal);
```

Siehe auch: UnterUeberzug Übersicht

Löscht den Unter- bzw. Überzug aus der ArCon Planung.

UnterUeberzug.GetPos

Deklaration:

```
Function GetPos(ByRef x1 as Single, ByRef y1 as
Single, ByRef x2 as Single, ByRef y2 as Single,
ByRef x3 as Single, ByRef y3 as Single, ByRef x4 as
Single, ByRef y4 as Single) as Boolean
```

```
HRESULT GetPos(float* x1, float* y1, float* x2, float*
y2, float* x3, float* y3, float* x4, float* y4,
VARIANT_BOOL* retVal);
```

Siehe auch: UnterUeberzug Übersicht

Ermittelt die Position des Unter- oder Überzuges

Parameter:

x1, y1 - x4, y4	Ergebnisparameter: die Koordinaten der vier Eckpunkte des Unter-/Überzuges

UnterUeberzug.SetPos

Deklaration:

```
Function SetPos(ByVal x1 as Single, ByVal y1 as
Single, ByVal x2 as Single, ByVal y2 as Single,
ByVal x3 as Single, ByVal y3 as Single, ByVal x4
as Single, ByVal y4 as Single) as Boolean
```

```
HRESULT SetPos(float x1, float y1, float x2, float
y2, float x3, float y3, float x4, float y4,
VARIANT_BOOL* retVal);
```

Siehe auch: UnterUeberzug Übersicht

Positioniert den Unter-/Überzug neu.

Parameter:

x1, y1 - x4, y4	Koordinaten der neuen Eckpunkte

UnterUeberzugCollection

Eine Liste von Unter- und Überzügen.

UnterUeberzugCollection Übersicht

Eigenschaften

Count	Anzahl der Unter-/Überzüge in dieser Liste

Methoden

Item	Liefert den i-ten Unter-/Überzug

UnterUeberzugCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: UnterUeberzugCollection Übersicht, Listen

Anzahl der Unter- und Überzüge in dieser Liste.

UnterUeberzugCollection.Item

Deklaration:

```
Function Item(ByVal Index as long) as UnterUeberzug
```

```
HRESULT Item(long Index, IUnterUeberzug** retVal);
```

Siehe auch: UnterUeberzugCollection Übersicht, UnterUeberzug, Listen

Liefert den i-ten Unter-/Überzug der Liste.

TextureCollection

3D Objekte verwenden häufig die gleiche Textur für viele Flächen. Um diese Texturen nicht mehrfach im Speicher halten zu müssen, werden Texturen durch einen Index in einer Texturliste dargestellt. Alle logisch zusammenhängenden Flächen mit der gleichen Textur verwenden dabei den gleichen Index, sodaß das Austauschen einer Textur in der Texturliste alle diese Flächen gleichzeitig ändert.

Welcher Texturindex dabei welcher logischen Bedeutung/Fläche entspricht, entscheidet der Ersteller des 3D Objektes.

TextureCollection Übersicht

Eigenschaften

Count	Anzahl der Texturnamen in dieser Liste

Methoden

Destroy	Löscht die Texturliste - darf nur für selbst erzeugte Listen (ArCon.NewTextureCollection) aufgerufen werden, nicht für Listen, die einem Objekt gehören!
Item	Liefert den i-ten Texturnamen

TextureCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: TextureCollection Übersicht, Listen

Die Anzahl der Texturen in dieser Liste.

TextureCollection.Destroy

Deklaration:

```
Sub Destroy()
```

```
HRESULT Destroy();
```

Siehe auch: TextureCollection Übersicht, Listen

TextureCollection.Item

Deklaration:

```
Function Item(ByVal Index as long) as TextureName
```

```
HRESULT Item(long Index, ITextureName** retVal);
```

Siehe auch: TextureCollection Übersicht, TextureName, Listen

Liefert die i-te Textur dieser Liste.

TextureName

Ein Texturnamensobjekt beschreibt eine Textur innerhalb einer Texturliste für 3D Objekte. Da Texturen häufig innerhalb eines Objektes für mehrere Flächen verwendet werden, wird bei der Objektkonstruktion ein Texturindex benutzt, der die Position der gewünschten Textur innerhalb der Texturliste angibt. Ändern Sie eine Textur innerhalb einer Texturliste, werden alle Flächen des Objektes, die diesen Texturindex benutzen, entsprechend geändert.

TextureName Übersicht

Eigenschaften

Name	Name dieser Textur

TextureName.Name

Deklaration:

String

HRESULT get_Name(BSTR* retVal);
HRESULT put_Name(BSTR newVal);

Siehe auch: TextureName Übersicht

Der Dateiname der Textur. Dieser Name kann relativ zu den ArCon Standardpfaden für Texturen sein, wenn er mit einem “ >” beginnt, z.B. “ >aussen\boden\boden01.bmp” .

Object3DCollection

Eine Liste von 3D-Objekten.

Object3DCollection Übersicht

Eigenschaften

Count	Anzahl der 3D-Objekte in dieser Liste

Methoden

Item	Liefert das i-te 3D Objekt

Object3DCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: Object3DCollection Übersicht, Listen

Die Anzahl der 3D-Objekte in dieser Liste.

Object3DCollection.Item

Deklaration:

```
Function Item(ByVal Index as long) as Object3D
```

```
HRESULT Item(long Index, IObject3D** retVal);
```

Siehe auch: Object3DCollection Übersicht, Object3D, Listen

Liefert das i-te 3D-Objekt der Liste.

ObjectConstructorCollection

Eine Liste von 3D-Objektklassen.

ObjectConstructorCollection Übersicht

Eigenschaften

Count	Anzahl der Objektklassen in dieser Liste

Methoden

Item	Liefert die i-te Objektklasse

ObjectConstructorCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: ObjectConstructorCollection Übersicht, Listen

Die Anzahl der 3D Objektklassen in dieser Liste.

ObjectConstructorCollection.Item

Deklaration:

```
Function Item(ByVal Index as long) as ObjectConstructor
```

```
HRESULT Item(long Index, IObjectConstructor** retVal);
```

Siehe auch: ObjectConstructorCollection Übersicht, ObjectConstructor, Listen

Liefert die i-te 3D-Objektklasse.

HolePolygon

HolePolygon Übersicht

Methoden

AddHole	Erzeugt im Loch ein Loch
RemoveHole	Löscht ein Loch aus dem (äußeren) Loch

HolePolygon.AddHole

Deklaration:

```
Function AddHole(ByVal newHole as HolePolygon) as
Boolean
```

```
HRESULT AddHole(IHolePolygon* newHole,
VARIANT_BOOL* retVal);
```

Siehe auch: HolePolygon Übersicht, HolePolygon

HolePolygon.RemoveHole

Deklaration:

```
Function RemoveHole(ByVal removedHole as HolePolygon)
as Boolean
```

```
HRESULT RemoveHole(IHolePolygon* removedHole,
VARIANT_BOOL* retVal);
```

Siehe auch: HolePolygon Übersicht, HolePolygon

Cut

Cut Übersicht

Eigenschaften

Graphics2D	Liste der Grafikelemente des Schnittes
History	Geschichte des Schnitts beim Gebäudeimport
ID	Eindeutige Kennung dieses Schnittes
Story	Stockwerk des Schnittes

Methoden

GetReferenceLine	Liefert Anfangs und Endpunkt der Schnitt-Referenzlinie

Cut.Graphics2D

Deklaration:

Graphics2DCollection (nur lesbar)

HRESULT get_Graphics2D(IGraphics2DCollection** retVal);

Siehe auch: Cut Übersicht, Graphics2DCollection

Cut.History

Deklaration:

IDHistory (nur lesbar)

HRESULT get_History(IIDHistory** retVal);

Siehe auch: Cut Übersicht, IDHistory

Cut.ID

Deklaration:

long (nur lesbar)

HRESULT get_ID(long* retVal);

Siehe auch: Cut Übersicht

Cut.Story

Deklaration:

Story (nur lesbar)

HRESULT get_Story(IStory** retVal);

Siehe auch: Cut Übersicht, Story

Cut.GetReferenceLine

Deklaration:

```
Function GetReferenceLine(ByRef x1 as Single, ByRef y1 as
Single, ByRef x2 as Single, ByRef y2 as Single) as Boolean
```

```
HRESULT GetReferenceLine(float* x1, float* y1, float* x2,
float* y2, VARIANT_BOOL* retVal);
```

Siehe auch: Cut Übersicht

CutCollection

CutCollection Übersicht

Eigenschaften

Count	Anzahl der Schnitte in der Liste

Methoden

Item	Liefert den n-ten Schnitt der Liste

CutCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: CutCollection Übersicht, Listen

CutCollection.Item

Deklaration:

```
Function Item(ByVal index as long) as Cut
```

```
HRESULT Item(long index, ICut** retVal);
```

Siehe auch: CutCollection Übersicht, Cut, Listen

CutView

CutView Übersicht

Eigenschaften

Cut	Der zu dieser Ansicht gehörende Schnitt

Methoden

Viewer	Viewer Handle der Ansicht (siehe auch HowMove Parameter viewHandle und View.Viewer)

CutView.Cut

Deklaration:

Cut (nur lesbar)

HRESULT get_Cut(ICut** retVal);

Siehe auch: CutView Übersicht, Cut

CutView.Viewer

Deklaration:

```
Function Viewer() as long
```

```
HRESULT Viewer(long* retVal);
```

Siehe auch: CutView Übersicht

RoofAreaCollection

RoofAreaCollection Übersicht

Eigenschaften

Count	Anzahl der Dachschrägen in der Liste

Methoden

Item	Liefert die n-te Dachfläche der Liste

RoofAreaCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: RoofAreaCollection Übersicht, Listen

RoofAreaCollection.Item

Deklaration:

```
Function Item(ByVal index as long) as RoofArea
```

```
HRESULT Item(long index, IRoofArea** retVal);
```

Siehe auch: RoofAreaCollection Übersicht, RoofArea, Listen

GaubenCollection

GaubenCollection Übersicht

Eigenschaften

Count	Anzahl der Gauben in der Liste

Methoden

Item	Liefert die n-te Gaube der Liste

GaubenCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: GaubenCollection Übersicht, Listen

GaubenCollection.Item

Deklaration:

```
Function Item(ByVal index as long) as Gaube
```

```
HRESULT Item(long index, IGaube** retVal);
```

Siehe auch: GaubenCollection Übersicht, Gaube, Listen

RoofArea

RoofArea Übersicht
Eigenschaften

Area	Fläche
AreaFormula	Formel für die Fläche
History	Daten zur Identifikation im ehemaligen Projekt für einzeln geladene Gebäude
ID	Eine eindeutige Kennung
Outside	Ist diese Dachfläche außen?
Roof	Das Dach, zu dem diese Fläche gehört
Room	Der Raum, zu dem diese Dachfläche gehört
Texture	Die Textur dieser Dachfläche
Visible	Ist diese Dachfläche sichtbar?
Windows	Liste der Fenster in dieser Dachfläche

RoofArea.Area

Deklaration:

Single (nur lesbar)

HRESULT get_Area(float* retVal);

Siehe auch: RoofArea Übersicht

RoofArea.AreaFormula

Deklaration:

String (nur lesbar)

HRESULT get_AreaFormula(BSTR* retVal);

Siehe auch: RoofArea Übersicht

RoofArea.History

Deklaration:

IDHistory (nur lesbar)

HRESULT get_History(IIDHistory** retVal);

Siehe auch: RoofArea Übersicht, IDHistory

RoofArea.ID

Deklaration:

long (nur lesbar)

HRESULT get_ID(long* retVal);

Siehe auch: RoofArea Übersicht

RoofArea.Outside

Deklaration:

Boolean

HRESULT get_Outside(VARIANT_BOOL* retVal);
HRESULT put_Outside(VARIANT_BOOL newVal);

Siehe auch: RoofArea Übersicht

RoofArea.Roof

Deklaration:

Roof (nur lesbar)

HRESULT get_Roof(IRoof** retVal);

Siehe auch: RoofArea Übersicht, Roof

RoofArea.Room

Deklaration:

Room (nur lesbar)

HRESULT get_Room(IRoom** retVal);

Siehe auch: RoofArea Übersicht, Room

RoofArea.Texture

Deklaration:

Texture (nur lesbar)

HRESULT get_Texture(ITexture** retVal);

Siehe auch: RoofArea Übersicht, Texture

RoofArea.Visible

Deklaration:

Boolean

HRESULT get_Visible(VARIANT_BOOL* retVal);
HRESULT put_Visible(VARIANT_BOOL newVal);

Siehe auch: RoofArea Übersicht

RoofArea.Windows

Deklaration:

WindowCollection (nur lesbar)

HRESULT get_Windows(IWindowCollection** retVal);

Siehe auch: RoofArea Übersicht, WindowCollection

RoofConstruction

RoofConstruction Übersicht
Eigenschaften

AbstandOrtgangSparren	Abstand der Ortgang-Sparren
DachHorizontalBegrenzen	Wird das Dach seitlich begrenzt?
DachrinnenDurchmesser	Durchmesser der Dachrinne
DickeEindeckung	Dicke der Eindeckung
DickeKonstruktion	Dicke der Konstruktion
EindeckungTransparent	Ist die Eindeckung durchsichtig?
Fehler	letzte Fehlermeldung
FirstpfettenBreite	Breite der Firstpfetten
FirstpfettenDicke	Dicke der Firstpfetten
FusspfettenBreite	Breite der Fußpfetten
FusspfettenDicke	Dicke der Fußpfetten
GesimsHoeheStirn	Gesimshöhe an der Stirn
GesimsNeigungStirn	Gesimsneigung an der Stirn
GratsparrenBreite	Breite der Gratsparren
GratsparrenDicke	Dicke der Gratsparren
HoeheUKKehlbalken	Höhe der Kehlbalken
HoeheUKMittelpfetten	Höhe der Mittelpfetten unterm Kehlgang
KehlbalkenBreite	Breite der Kehlbalken
KehlbalkenDicke	Dicke der Kehlbalken
KehlbohlenBreite	Breite der Kehlbohlen
KehlbohlenDicke	Dicke der Kehlbohlen
MaxSparrenAbstand	Maximaler Sparren-Abstand
MaxUeberstandEindeckung	Maximaler Überstand der Eindeckung
MitDachrinne	Hat das Dach eine Dachrinne?
MitFirstSteinen	Hat die Dachkonstruktion First-Steine?
MitFirstpfetten	Holzkonstruktion mit Firstpfetten
MitFusspfetten	Holzkonstruktion mit Fußpfetten
MitGratsparren	Holzkonstruktion mit Gratsparren
MitKehlbalken	Holzkonstruktion mit Kehlbalken
MitMittelpfetten	Holszkonstruktion mit Mittelpfetten
MitSparren	Holzkonstruktion mit Sparren
MitTraufDetails	Sind die Trauf-Detail gültig?
MittelpfettenBreite	Breite der Mittelpfetten
MittelpfettenDicke	Dicke der Mittelpfetten
PfettenKoepfeSichtbar	Sind die Pfettenköpfe sichtbar?
SchnittHoehe	Höhe des Schnittes
SparrenBreite	Breite der Sparren
SparrenDicke	Dicke der Sparren
SparrenKopfLaenge	Länge der Sparren-Köpfe
SparrenKopfTiefe	Tiefe der Sparrenköpfe

TraufDetailTyp	Art der Traufdetails

Methoden

AddPoint	Fügt einen Eckkpunkt zu einer Dachfläche hinzu (vorher NewArea aufrufen!). Die Koordinaten sind in Meter.
Analyze	Testet, ob die Konstrutkion durchführbar ist. Wenn autoClosure wahr ist, werden fehlende Unterbauflächen durch ein Näherungsverfahren automatisch ergänzt.
BeginNewArea	Beginnt mit der Konstruktion einer neue Fläche. Achtung: diese Funktion deaktiviert die automatische Dachflächenberechnung für dieses Dach!
CreateRoof	Erzeugt ein Dach mit diesen Einstellungen
GetAnzahlFlaechen	Liefert die Anzahl der Flächen (Flächen-Indizes: 1 .. Anzahl)
GetAnzahlKanten	Liefert die Anzahl der Kanten der Fläche
GetDachkanteAktiv	Ist dies bei doppelten (parallelen) Kanten die wirksame Kante?
GetDachkantenNachbarflaeche	Liefert den Flächenindex der Nachbarfläche
GetDachkantenPunktAnfang	Liefert den Startpunkt der Dachkante
GetDachkantenPunktEnde	Liefert den Endpunkt der Dachkante
GetDachkantenTyp	Liefert den Kantentyp (z.B. ACDACH_Traufe)
GetFlaechenNeigung	Liefert die Neigung der Fläche (in Grad!)
GetFlaechenRichtung	Liefert die Richtung der Fläche (in Grad!)
GetFlaechenTyp	Liefert die Art der Fläche (z.B. ACDACH_FlaecheEindeckung)
GetFlaechenVater	Liefert den Flächenindex der Elternfläche
GetPfettenKopfTyp	Liefert die Art der Pfettenköpfe
HoleAnzahlKnicke	Liefert die Anzahl der Knicke einer Dachseite
HoleAnzahlKonturPunkte	Liefert die Anzahl der Konturpunkte
HoleKnickHoehe	Liefert die Höhe eines Knickes
HoleKnickNeigung	Liefert die Neigung eines Knickes
HoleKonturPunkt	Liefert einen Konturpunkt
HoleUeberstand	Liefert den Überstand
NewRafter	Beginnt mit der Konstruktion eines neuen Dachbalkens
RafterSetGeo	Fügt in die aktuelle Dachbalkenkonstruktion Daten ein. Vorher NewRafter aufrufen!
SetPfettenKopfTyp	Setzt die Art der Pfettenköpfe (ACPKT_PfettenkopfStandard, ACPKT_PfettenkopfModern oder ACPKT_PfettenkopfKlassisch)
SetzeAnzahlKnicke	Setzt die Anzahl der Knicke einer Dachseite
SetzeAnzahlKonturPunkte	Setzt die Anzahl der Konturpunkte
SetzeKnickHoehe	Setzt die Höhe eines Knickes
SetzeKnickNeigung	Setzt die Neigung eines Knickes
SetzeKonturPunkt	Setzt einen Konturpunkt
SetzeUeberstand	Setzt den Überstand

RoofConstruction.AbstandOrtgangSparren

Deklaration:

double

HRESULT get_AbstandOrtgangSparren(double* retVal);
HRESULT put_AbstandOrtgangSparren(double newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.DachHorizontalBegrenzen

Deklaration:

Boolean

HRESULT get_DachHorizontalBegrenzen
(VARIANT_BOOL* retVal);

HRESULT put_DachHorizontalBegrenzen(VARIANT_BOOL
newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.DachrinnenDurchmesser

Deklaration:

double

HRESULT get_DachrinnenDurchmesser(double* retVal);
HRESULT put_DachrinnenDurchmesser(double newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.DickeEindeckung

Deklaration:

double

HRESULT get_DickeEindeckung(double* retVal);
HRESULT put_DickeEindeckung(double newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.DickeKonstruktion

Deklaration:

double

HRESULT get_DickeKonstruktion(double* retVal);
HRESULT put_DickeKonstruktion(double newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.EindeckungTransparent

Deklaration:

Boolean

HRESULT get_EindeckungTransparent(VARIANT_BOOL* retVal);

HRESULT put_EindeckungTransparent(VARIANT_BOOL newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.Fehler

Deklaration:

String (nur lesbar)

HRESULT get_Fehler(BSTR* retVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.FirstpfettenBreite

Deklaration:

double

HRESULT get_FirstpfettenBreite(double* retVal);
HRESULT put_FirstpfettenBreite(double newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.FirstpfettenDicke

Deklaration:

double

HRESULT get_FirstpfettenDicke(double* retVal);
HRESULT put_FirstpfettenDicke(double newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.FusspfettenBreite

Deklaration:

double

HRESULT get_FusspfettenBreite(double* retVal);
HRESULT put_FusspfettenBreite(double newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.FusspfettenDicke

Deklaration:

double

HRESULT get_FusspfettenDicke(double* retVal);
HRESULT put_FusspfettenDicke(double newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.GesimsHoeheStirn

Deklaration:

double

HRESULT get_GesimsHoeheStirn(double* retVal);
HRESULT put_GesimsHoeheStirn(double newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.GesimsNeigungStirn

Deklaration:

double

HRESULT get_GesimsNeigungStirn(double* retVal);
HRESULT put_GesimsNeigungStirn(double newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.GratsparrenBreite

Deklaration:

double

HRESULT get_GratsparrenBreite(double* retVal);
HRESULT put_GratsparrenBreite(double newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.GratsparrenDicke

Deklaration:

double

HRESULT get_GratsparrenDicke(double* retVal);
HRESULT put_GratsparrenDicke(double newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.HoeheUKKehlbalken

Deklaration:

double

HRESULT get_HoeheUKKehlbalken(double* retVal);
HRESULT put_HoeheUKKehlbalken(double newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.HoeheUKMittelpfetten

Deklaration:

double

HRESULT get_HoeheUKMittelpfetten(double* retVal);
HRESULT put_HoeheUKMittelpfetten(double newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.KehlbalkenBreite

Deklaration:

double

HRESULT get_KehlbalkenBreite(double* retVal);
HRESULT put_KehlbalkenBreite(double newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.KehlbalkenDicke

Deklaration:

double

HRESULT get_KehlbalkenDicke(double* retVal);
HRESULT put_KehlbalkenDicke(double newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.KehlbohlenBreite

Deklaration:

double

HRESULT get_KehlbohlenBreite(double* retVal);
HRESULT put_KehlbohlenBreite(double newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.KehlbohlenDicke

Deklaration:

double

HRESULT get_KehlbohlenDicke(double* retVal);
HRESULT put_KehlbohlenDicke(double newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.MaxSparrenAbstand

Deklaration:

double

HRESULT get_MaxSparrenAbstand(double* retVal);
HRESULT put_MaxSparrenAbstand(double newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.MaxUeberstandEindeckung

Deklaration:

double

HRESULT get_MaxUeberstandEindeckung(double* retVal);
HRESULT put_MaxUeberstandEindeckung(double newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.MitDachrinne

Deklaration:

Boolean

HRESULT get_MitDachrinne(VARIANT_BOOL* retVal);
HRESULT put_MitDachrinne(VARIANT_BOOL newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.MitFirstSteinen

Deklaration:

Boolean

HRESULT get_MitFirstSteinen(VARIANT_BOOL* retVal);
HRESULT put_MitFirstSteinen(VARIANT_BOOL newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.MitFirstpfetten

Deklaration:

Boolean

HRESULT get_MitFirstpfetten(VARIANT_BOOL* retVal);
HRESULT put_MitFirstpfetten(VARIANT_BOOL newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.MitFusspfetten

Deklaration:

Boolean

HRESULT get_MitFusspfetten(VARIANT_BOOL* retVal);
HRESULT put_MitFusspfetten(VARIANT_BOOL newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.MitGratsparren

Deklaration:

Boolean

HRESULT get_MitGratsparren(VARIANT_BOOL* retVal);
HRESULT put_MitGratsparren(VARIANT_BOOL newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.MitKehlbalken

Deklaration:

Boolean

HRESULT get_MitKehlbalken(VARIANT_BOOL* retVal);
HRESULT put_MitKehlbalken(VARIANT_BOOL newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.MitMittelpfetten

Deklaration:

Boolean

HRESULT get_MitMittelpfetten(VARIANT_BOOL* retVal);
HRESULT put_MitMittelpfetten(VARIANT_BOOL newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.MitSparren

Deklaration:

Boolean

```
HRESULT get_MitSparren(VARIANT_BOOL* retVal);
HRESULT put_MitSparren(VARIANT_BOOL newVal);
```

Siehe auch: RoofConstruction Übersicht

RoofConstruction.MitTraufDetails

Deklaration:

Boolean

HRESULT get_MitTraufDetails(VARIANT_BOOL* retVal);
HRESULT put_MitTraufDetails(VARIANT_BOOL newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.MittelpfettenBreite

Deklaration:

double

HRESULT get_MittelpfettenBreite(double* retVal);
HRESULT put_MittelpfettenBreite(double newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.MittelpfettenDicke

Deklaration:

double

HRESULT get_MittelpfettenDicke(double* retVal);
HRESULT put_MittelpfettenDicke(double newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.PfettenKoepfeSichtbar

Deklaration:

Boolean

HRESULT get_PfettenKoepfeSichtbar(VARIANT_BOOL* retVal);

HRESULT put_PfettenKoepfeSichtbar(VARIANT_BOOL newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.SchnittHoehe

Deklaration:

double

HRESULT get_SchnittHoehe(double* retVal);
HRESULT put_SchnittHoehe(double newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.SparrenBreite

Deklaration:

double

HRESULT get_SparrenBreite(double* retVal);
HRESULT put_SparrenBreite(double newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.SparrenDicke

Deklaration:

double

HRESULT get_SparrenDicke(double* retVal);
HRESULT put_SparrenDicke(double newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.SparrenKopfLaenge

Deklaration:

double

HRESULT get_SparrenKopfLaenge(double* retVal);
HRESULT put_SparrenKopfLaenge(double newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.SparrenKopfTiefe

Deklaration:

double

HRESULT get_SparrenKopfTiefe(double* retVal);
HRESULT put_SparrenKopfTiefe(double newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.TraufDetailTyp

Deklaration:

long

HRESULT get_TraufDetailTyp(long* retVal);
HRESULT put_TraufDetailTyp(long newVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.AddPoint

Deklaration:

```
Function AddPoint(ByVal x as double, ByVal y as double,
ByVal z as double) as Boolean
```

```
HRESULT AddPoint(double x, double y, double z,
VARIANT_BOOL* retVal);
```

Siehe auch: RoofConstruction Übersicht

RoofConstruction.Analyze

Deklaration:

```
Function Analyze(ByVal autoClosure as Boolean) as
Boolean
```

```
HRESULT Analyze(VARIANT_BOOL autoClosure,
VARIANT_BOOL* retVal);
```

Siehe auch: RoofConstruction Übersicht

RoofConstruction.BeginNewArea

Deklaration:

```
Function BeginNewArea(ByVal typeOfArea as
AC_Dachflaechentypen) as Boolean
```

```
HRESULT BeginNewArea(AC_Dachflaechentypen
typeOfArea, VARIANT_BOOL* retVal);
```

Siehe auch: RoofConstruction Übersicht

RoofConstruction.CreateRoof

Deklaration:

```
Function CreateRoof(ByVal aStory as Story) as Roof
```

```
HRESULT CreateRoof(IStory* aStory, IRoof** retVal);
```

Siehe auch: RoofConstruction Übersicht, Roof, Story

RoofConstruction.GetAnzahlFlaechen

Deklaration:

```
Function GetAnzahlFlaechen() as long
```

```
HRESULT GetAnzahlFlaechen(long* retVal);
```

Siehe auch: RoofConstruction Übersicht

RoofConstruction.GetAnzahlKanten

Deklaration:

```
Function GetAnzahlKanten(ByVal index as long) as long
```

```
HRESULT GetAnzahlKanten(long index, long* retVal);
```

Siehe auch: RoofConstruction Übersicht

RoofConstruction.GetDachkanteAktiv

Deklaration:

```
Function GetDachkanteAktiv(ByVal index as long, ByVal
kantenIndex as long) as Boolean
```

```
HRESULT GetDachkanteAktiv(long index, long kantenIndex,
VARIANT_BOOL* retVal);
```

Siehe auch: RoofConstruction Übersicht

RoofConstruction.GetDachkantenNachbarflaeche

Deklaration:

```
Function GetDachkantenNachbarflaeche(ByVal index as
long, ByVal kantenIndex as long) as long
```

```
HRESULT GetDachkantenNachbarflaeche(long index, long
kantenIndex, long* retVal);
```

Siehe auch: RoofConstruction Übersicht

RoofConstruction.GetDachkantenPunktAnfang

Deklaration:

```
Function GetDachkantenPunktAnfang(ByVal index as long,
ByVal kantenIndex as long) as Boolean
```

```
HRESULT GetDachkantenPunktAnfang(long index, long
kantenIndex, float *x, float *y, float *z, VARIANT_BOOL*
retVal);
```

Siehe auch: RoofConstruction Übersicht

RoofConstruction.GetDachkantenPunktEnde

Deklaration:

```
Function GetDachkantenPunktEnde(ByVal index as long,
ByVal kantenIndex as long) as Boolean
```

```
HRESULT GetDachkantenPunktEnde(long index, long
kantenIndex, float *x, float *y, float *z, VARIANT_BOOL*
retVal);
```

Siehe auch: RoofConstruction Übersicht

RoofConstruction.GetDachkantenTyp

Deklaration:

```
Function GetDachkantenTyp(ByVal index as long, ByVal
kantenIndex as long) as long
```

```
HRESULT GetDachkantenTyp(long index, long kantenIndex,
long* retVal);
```

Siehe auch: RoofConstruction Übersicht

RoofConstruction.GetFlaechenNeigung

Deklaration:

```
Function GetFlaechenNeigung(ByVal index as long) as
double
```

```
HRESULT GetFlaechenNeigung(long index, double* retVal);
```

Siehe auch: RoofConstruction Übersicht

RoofConstruction.GetFlaechenRichtung

Deklaration:

```
Function GetFlaechenRichtung(ByVal index as long) as
double
```

```
HRESULT GetFlaechenRichtung(long index, double* retVal)
;
```

Siehe auch: RoofConstruction Übersicht

RoofConstruction.GetFlaechenTyp

Deklaration:

```
Function GetFlaechenTyp(ByVal index as long) as long
```

```
HRESULT GetFlaechenTyp(long index, long* retVal);
```

Siehe auch: RoofConstruction Übersicht

RoofConstruction.GetFlaechenVater

Deklaration:

```
Function GetFlaechenVater(ByVal index as long) as long
```

```
HRESULT GetFlaechenVater(long index, long* retVal);
```

Siehe auch: RoofConstruction Übersicht

RoofConstruction.GetPfettenKopfTyp

Deklaration:

```
Function GetPfettenKopfTyp() as long
```

```
HRESULT GetPfettenKopfTyp(long* retVal);
```

Siehe auch: RoofConstruction Übersicht

RoofConstruction.HoleAnzahlKnicke

Deklaration:

```
Function HoleAnzahlKnicke(ByVal seite as long, ByRef
anzahlKnicke as long) as Boolean
```

```
HRESULT HoleAnzahlKnicke(long seite, long* anzahlKnicke,
VARIANT_BOOL* retVal);
```

Siehe auch: RoofConstruction Übersicht

RoofConstruction.HoleAnzahlKonturPunkte

Deklaration:

```
Function HoleAnzahlKonturPunkte(ByRef AnzahlSeiten as
long) as Boolean
```

```
HRESULT HoleAnzahlKonturPunkte(long* AnzahlSeiten,
VARIANT_BOOL* retVal);
```

Siehe auch: RoofConstruction Übersicht

RoofConstruction.HoleKnickHoehe

Deklaration:

```
Function HoleKnickHoehe(ByVal seite as long, ByVal knick
as long, ByRef hoehe as double) as Boolean
```

```
HRESULT HoleKnickHoehe(long seite, long knick, double*
hoehe, VARIANT_BOOL* retVal);
```

Siehe auch: RoofConstruction Übersicht

RoofConstruction.HoleKnickNeigung

Deklaration:

```
Function HoleKnickNeigung(ByVal seite as long, ByVal knick
as long, ByRef neigung as double) as Boolean
```

```
HRESULT HoleKnickNeigung(long seite, long knick, double*
neigung, VARIANT_BOOL* retVal);
```

Siehe auch: RoofConstruction Übersicht

RoofConstruction.HoleKonturPunkt

Deklaration:

```
Function HoleKonturPunkt(ByVal seite as long, ByRef x as
double, ByRef y as double) as Boolean
```

```
HRESULT HoleKonturPunkt(long seite, double* x, double* y,
VARIANT_BOOL* retVal);
```

Siehe auch: RoofConstruction Übersicht

RoofConstruction.HoleUeberstand

Deklaration:

```
Function HoleUeberstand(ByVal seite as long, ByRef
ueberstand as double) as Boolean
```

```
HRESULT HoleUeberstand(long seite, double* ueberstand,
VARIANT_BOOL* retVal);
```

Siehe auch: RoofConstruction Übersicht

RoofConstruction.NewRafter

Deklaration:

```
Function NewRafter(ByVal typeOfRafter as AC_Dachbalken,
ByVal Name as String, ByVal width as double, ByVal
thickness as double) as Boolean
```

```
HRESULT NewRafter(AC_Dachbalken typeOfRafter, BSTR
Name, double width, double thickness, VARIANT_BOOL*
retVal);
```

Siehe auch: RoofConstruction Übersicht

RoofConstruction.RafterSetGeo

Deklaration:

```
Function RafterSetGeo(ByVal flag as AC_RafterGeoType,
ByVal x as double, ByVal y as double, ByVal z as double) as
Boolean
```

```
HRESULT RafterSetGeo(AC_RafterGeoType flag, double x,
double y, double z, VARIANT_BOOL* retVal);
```

Siehe auch: RoofConstruction Übersicht

RoofConstruction.SetPfettenKopfTyp

Deklaration:

Sub SetPfettenKopfTyp(ByVal nVal as long)

HRESULT SetPfettenKopfTyp(long nVal);

Siehe auch: RoofConstruction Übersicht

RoofConstruction.SetzeAnzahlKnicke

Deklaration:

```
Function SetzeAnzahlKnicke(ByVal seite as long, ByVal
anzahlKnicke as long) as Boolean
```

```
HRESULT SetzeAnzahlKnicke(long seite, long anzahlKnicke,
VARIANT_BOOL* retVal);
```

Siehe auch: RoofConstruction Übersicht

RoofConstruction.SetzeAnzahlKonturPunkte

Deklaration:

```
Function SetzeAnzahlKonturPunkte(ByVal AnzahlSeiten as
long) as Boolean
```

```
HRESULT SetzeAnzahlKonturPunkte(long AnzahlSeiten,
VARIANT_BOOL* retVal);
```

Siehe auch: RoofConstruction Übersicht

RoofConstruction.SetzeKnickHoehe

Deklaration:

```
Function SetzeKnickHoehe(ByVal seite as long, ByVal knick
as long, ByVal hoehe as double) as Boolean
```

```
HRESULT SetzeKnickHoehe(long seite, long knick, double
hoehe, VARIANT_BOOL* retVal);
```

Siehe auch: RoofConstruction Übersicht

RoofConstruction.SetzeKnickNeigung

Deklaration:

```
Function SetzeKnickNeigung(ByVal seite as long, ByVal
knick as long, ByVal neigung as double) as Boolean
```

```
HRESULT SetzeKnickNeigung(long seite, long knick, double
neigung, VARIANT_BOOL* retVal);
```

Siehe auch: RoofConstruction Übersicht

RoofConstruction.SetzeKonturPunkt

Deklaration:

```
Function SetzeKonturPunkt(ByVal seite as long, ByVal x as
double, ByVal y as double) as Boolean
```

```
HRESULT SetzeKonturPunkt(long seite, double x, double y,
VARIANT_BOOL* retVal);
```

Siehe auch: RoofConstruction Übersicht

RoofConstruction.SetzeUeberstand

Deklaration:

```
Function SetzeUeberstand(ByVal seite as long, ByVal
ueberstand as double) as Boolean
```

```
HRESULT SetzeUeberstand(long seite, double ueberstand,
VARIANT_BOOL* retVal);
```

Siehe auch: RoofConstruction Übersicht

RoofWood

RoofWood Übersicht
Eigenschaften

Height	Höhe
Length	Länge
Number	Anzahl gleichartiger Balken
Type	Art des Balkens
TypeName	Name des Balkentyps (entspricht Type, aber als Text)
Width	Breite

RoofWood.Height

Deklaration:

Single

HRESULT get_Height(float* retVal);
HRESULT put_Height(float newVal);

Siehe auch: RoofWood Übersicht

RoofWood.Length

Deklaration:

Single

HRESULT get_Length(float* retVal);
HRESULT put_Length(float newVal);

Siehe auch: RoofWood Übersicht

RoofWood.Number

Deklaration:

long (nur lesbar)

HRESULT get_Number(long* retVal);

Siehe auch: RoofWood Übersicht

RoofWood.Type

Deklaration:

long

HRESULT get_Type(long* retVal);
HRESULT put_Type(long newVal);

Siehe auch: RoofWood Übersicht

RoofWood.TypeName

Deklaration:

String (nur lesbar)

HRESULT get_TypeName(BSTR* retVal);

Siehe auch: RoofWood Übersicht

RoofWood.Width

Deklaration:

Single

HRESULT get_Width(float* retVal);
HRESULT put_Width(float newVal);

Siehe auch: RoofWood Übersicht

RoofWoodCollection

RoofWoodCollection Übersicht

Eigenschaften

Count	Anzahl der Balken

Methoden

Item	Der i-te Balken
Sort	Sortiert die Liste

RoofWoodCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: RoofWoodCollection Übersicht, Listen

RoofWoodCollection.Item

Deklaration:

```
Function Item(ByVal Index as long) as RoofWood
```

```
HRESULT Item(long Index, IRoofWood** retVal);
```

Siehe auch: RoofWoodCollection Übersicht, RoofWood, Listen

RoofWoodCollection.Sort

Deklaration:

Sub Sort(ByVal criteria as long)

HRESULT Sort(long criteria);

Siehe auch: RoofWoodCollection Übersicht, Listen

PrintSettings

PrintSettings Übersicht

Eigenschaften

AlsGrundriss	Der Ausdruck erfolgt als Grundriss
BackgroundWhite	Gibt die Hintergrundfarbe des Ausdrucks an
GreyColor	Gibt an, wie die Farben bei der Grundrissausgabe gedruckt werden
PagesToScaleTo	Wenn der Ausdruck skaliert wird, gibt dieses Attribut die Anzahl der Seiten an, auf die skaliert wird
Resolution	Gibt die zu verwendende Druckauflösung an
Scale	Der Ausdruck wird skaliert

PrintSettings.AlsGrundriss

Deklaration:

Boolean

HRESULT get_AlsGrundriss(VARIANT_BOOL* retVal);
HRESULT put_AlsGrundriss(VARIANT_BOOL newVal);

Siehe auch: PrintSettings Übersicht

PrintSettings.BackgroundWhite

Deklaration:

Boolean

```
HRESULT get_BackgroundWhite(VARIANT_BOOL* retVal);
HRESULT put_BackgroundWhite(VARIANT_BOOL newVal)
;
```

Siehe auch: PrintSettings Übersicht

PrintSettings.GreyColor

Deklaration:

long

HRESULT get_GreyColor(long* retVal);
HRESULT put_GreyColor(long newVal);

Siehe auch: PrintSettings Übersicht

PrintSettings.PagesToScaleTo

Deklaration:

Single

HRESULT get_PagesToScaleTo(float* retVal);
HRESULT put_PagesToScaleTo(float newVal);

Siehe auch: PrintSettings Übersicht

PrintSettings.Resolution

Deklaration:

long

HRESULT get_Resolution(long* retVal);
HRESULT put_Resolution(long newVal);

Siehe auch: PrintSettings Übersicht

PrintSettings.Scale

Deklaration:

Boolean

HRESULT get_Scale(VARIANT_BOOL* retVal);
HRESULT put_Scale(VARIANT_BOOL newVal);

Siehe auch: PrintSettings Übersicht

SavePictureSettings

SavePictureSettings Übersicht
Eigenschaften

AntiAliasing	Kantenfilter aktivieren
Height	Bei beliebiger Bildgröße: die Höhe in Pixel
Oversampling	Oversampling zur Verbesserung der Bildqualität
Raytrace	Bild per Raytrace erzeugen
Size	Größe des erzeugten Bildes
TrueColor	Bild im True-Color Modus (16,7 Millionen Farben) erzeugen
WMF	Grundrissansicht als Windows Meta File (WMF) speichern
Width	Bei beliebiger Bildgröße: die Breite in Pixel

SavePictureSettings.AntiAliasing

Deklaration:

Boolean

HRESULT get_AntiAliasing(VARIANT_BOOL* retVal);
HRESULT put_AntiAliasing(VARIANT_BOOL newVal);

Siehe auch: SavePictureSettings Übersicht

SavePictureSettings.Height

Deklaration:

long

HRESULT get_Height(long* retVal);
HRESULT put_Height(long newVal);

Siehe auch: SavePictureSettings Übersicht

SavePictureSettings.Oversampling

Deklaration:

long

HRESULT get_Oversampling(long* retVal);
HRESULT put_Oversampling(long newVal);

Siehe auch: SavePictureSettings Übersicht

SavePictureSettings.Raytrace

Deklaration:

Boolean

HRESULT get_Raytrace(VARIANT_BOOL* retVal);
HRESULT put_Raytrace(VARIANT_BOOL newVal);

Siehe auch: SavePictureSettings Übersicht

SavePictureSettings.Size

Deklaration:

long

HRESULT get_Size(long* retVal);
HRESULT put_Size(long newVal);

Siehe auch: SavePictureSettings Übersicht

SavePictureSettings.TrueColor

Deklaration:

Boolean

HRESULT get_TrueColor(VARIANT_BOOL* retVal);
HRESULT put_TrueColor(VARIANT_BOOL newVal);

Siehe auch: SavePictureSettings Übersicht

SavePictureSettings.WMF

Deklaration:

Boolean

HRESULT get_WMF(VARIANT_BOOL* retVal);
HRESULT put_WMF(VARIANT_BOOL newVal);

Siehe auch: SavePictureSettings Übersicht

SavePictureSettings.Width

Deklaration:

long

HRESULT get_Width(long* retVal);
HRESULT put_Width(long newVal);

Siehe auch: SavePictureSettings Übersicht

ZoomSettings

ZoomSettings Übersicht

Eigenschaften

Factor	Bei beliebigem Zoomfaktor: der verwendete Faktor
Mode	Art der Zoomfunktion

ZoomSettings.Factor

Deklaration:

Single

HRESULT get_Factor(float* retVal);
HRESULT put_Factor(float newVal);

Siehe auch: ZoomSettings Übersicht

ZoomSettings.Mode

Deklaration:

long

HRESULT get_Mode(long* retVal);
HRESULT put_Mode(long newVal);

Siehe auch: ZoomSettings Übersicht

WalkSettings

WalkSettings Übersicht
Eigenschaften

CollisionControl	Ist die Kollisionskontrolle beim Durchwandern eingeschaltet?
Manipulator	Eingabegerät zur Steuerung des Durchwanderns
Parallel	Parallel zum Gelände wandern

WalkSettings.CollisionControl

Deklaration:

Boolean

HRESULT get_CollisionControl(VARIANT_BOOL* retVal);
HRESULT put_CollisionControl(VARIANT_BOOL newVal);

Siehe auch: WalkSettings Übersicht

WalkSettings.Manipulator

Deklaration:

long

HRESULT get_Manipulator(long* retVal);
HRESULT put_Manipulator(long newVal);

Siehe auch: WalkSettings Übersicht

WalkSettings.Parallel

Deklaration:

Boolean

HRESULT get_Parallel(VARIANT_BOOL* retVal);
HRESULT put_Parallel(VARIANT_BOOL newVal);

Siehe auch: WalkSettings Übersicht

ViewCollection

ViewCollection Übersicht

Eigenschaften

Count	Anzahl der vorhandenen Views

Methoden

Item	Liefert einen View aus der Liste

ViewCollection.Count

Deklaration:

long (nur lesbar)

HRESULT get_Count(long* retVal);

Siehe auch: ViewCollection Übersicht, Listen

ViewCollection.Item

Deklaration:

```
Function Item(ByVal index as long) as View
```

```
HRESULT Item(long index, IView** retVal);
```

Siehe auch: ViewCollection Übersicht, View, Listen

View

View Übersicht

Eigenschaften

Active	Die Ansicht ist die aktive Ansicht
Caption	Name der Ansicht
CurrentStory	Das aktuelle Stockwerk dieser Ansicht
Cut	Der zu dieser Ansicht gehörende Schnitt - falls es eine Schnittansicht ist
Height	Die Höhe der Ansicht in Bildschirm-Koordinaten
Left	Die linke Seite der Ansicht in Bildschirm-Koordinaten
TimeOfDay	Beleuchtungseinstellung
Top	Die obere Seite der Ansicht in Bildschirm-Koordinaten
Viewer	Viewer Handle (vergleiche auch CutView.Viewer und Parameter viewHandle im Ereignis HowMove)
Walking	Die Ansicht ist im Durchwandern-Modus
Width	Die Breite der Ansicht in Bildschirm-Koordinaten
Zooming	Die Ansicht wird gerade gezoomt

Methoden

Activate	Aktiviert die Ansicht
Elevator	Betätigt den Aufzug
GetCutViewing2D	Liefert einen Schnappschuß des aktuellen Viewings für den Ansichts-Typ 2D Schnittansicht
GetMode	Ermittelt den aktuellen Typ und Modus der Ansicht
GetViewing	Liefert einen Schnappschuß des aktuellen Viewing für einen der möglichen Ansichts-Typen
Home	Initialisiert die Ansicht auf Grundeinstellungen
Pan	Verschiebt die Ansicht
PlayWalkFile	Spielt eine aufgezeichnete Durchwanderung (*.WLK Datei)
Print	Druckt die Ansicht
Rotate	Rotiert die Ansicht
SavePicture	Speichert die Ansicht als Bitmap oder WMF
SelectPredefinedViewing3D	Wählt einen vordefinierten Betrachterstandpunkt aus (nur in Perspektivansicht!)
SetCutViewing2D	Setzt das Viewing für den Ansichts-Typ 2D Schnittansicht
SetFloorVisibility	Schaltet die Sichtbarkeit auf 'nur das aktuelle Stockwerk' bzw. auf 'alle'
SetMode	Ändert Typ und Modus der Ansicht
SetViewing2D	Verändert das Viewing für einen der 2D Ansichtstypen
SetViewing3D	Verändert das Viewing für die perspektivische Ansicht
ShowAll	Zeigt alles
StartMirrorBuilding	Beginnt einen interaktive Gebäude-Spiegeln Vorgang
StartRotateBuilding	Beginnt einen interaktiven Gebäude-Drehen Vorgang
StartWalking	Beginnt den interaktiven Durchwandermodus
StartZoom	Beginnt einen interaktiven Zoom-Vorgang

StopWalking	Beendet den interaktiven Durchwandermodus
StopZoom	Beendetet einen interaktiven Zoom-Vorgang
Zoom	Zoomt programmgesteuert

View.Active

Deklaration:

Boolean (nur lesbar)

HRESULT get_Active(VARIANT_BOOL* retVal);

Siehe auch: View Übersicht

View.Caption

Deklaration:

String

HRESULT get_Caption(BSTR* retVal);
HRESULT put_Caption(BSTR newVal);

Siehe auch: View Übersicht

View.CurrentStory

Deklaration:

Story

HRESULT get_CurrentStory(IStory** retVal);
HRESULT put_CurrentStory(IStory* newVal);

Siehe auch: View Übersicht, Story

View.Cut

Deklaration:

Cut

HRESULT get_Cut(ICut** retVal);
HRESULT put_Cut(ICut* newVal);

Siehe auch: View Übersicht, Cut

View.Height

Deklaration:

long

HRESULT get_Height(long* retVal);
HRESULT put_Height(long newVal);

Siehe auch: View Übersicht

View.Left

Deklaration:

long

HRESULT get_Left(long* retVal);
HRESULT put_Left(long newVal);

Siehe auch: View Übersicht

View.TimeOfDay

Deklaration:

long

HRESULT get_TimeOfDay(long* retVal);
HRESULT put_TimeOfDay(long newVal);

Siehe auch: View Übersicht

View.Top

Deklaration:

long

HRESULT get_Top(long* retVal);
HRESULT put_Top(long newVal);

Siehe auch: View Übersicht

View.Viewer

Deklaration:

long (nur lesbar)

HRESULT get_Viewer(long* retVal);

Siehe auch: View Übersicht

View.Walking

Deklaration:

Boolean (nur lesbar)

HRESULT get_Walking(VARIANT_BOOL* retVal);

Siehe auch: View Übersicht

View.Width

Deklaration:

long

HRESULT get_Width(long* retVal);
HRESULT put_Width(long newVal);

Siehe auch: View Übersicht

View.Zooming

Deklaration:

Boolean (nur lesbar)

HRESULT get_Zooming(VARIANT_BOOL* retVal);

Siehe auch: View Übersicht

View.Activate

Deklaration:

```
Function Activate() as Boolean
```

```
HRESULT Activate(VARIANT_BOOL* retVal);
```

Siehe auch: View Übersicht

View.Elevator

Deklaration:

```
Function Elevator(ByVal up as Boolean, ByVal animated as
Boolean) as Boolean
```

```
HRESULT Elevator(VARIANT_BOOL up, VARIANT_BOOL
animated, VARIANT_BOOL* retVal);
```

Siehe auch: View Übersicht

View.GetCutViewing2D

Deklaration:

```
Function GetCutViewing2D(ByVal CutViewMode as long) as
Boolean
```

```
HRESULT GetCutViewing2D(long CutViewMode, IViewing *
*pViewing, VARIANT_BOOL* retVal);
```

Siehe auch: View Übersicht

View.GetMode

Deklaration:

Sub GetMode()

HRESULT GetMode(long *ViewType, long *ViewSubType, long *ViewMode);

Siehe auch: View Übersicht

View.GetViewing

Deklaration:

```
Function GetViewing(ByVal typeOfView as long) as Viewing
```

```
HRESULT GetViewing(long typeOfView, IViewing** retVal);
```

Siehe auch: View Übersicht, Viewing

View.Home

Deklaration:

```
Function Home() as Boolean
```

```
HRESULT Home(VARIANT_BOOL* retVal);
```

Siehe auch: View Übersicht

View.Pan

Deklaration:

```
Function Pan(ByVal PanCode as long, ByVal fast as
Boolean) as Boolean
```

```
HRESULT Pan(long PanCode, VARIANT_BOOL fast,
VARIANT_BOOL* retVal);
```

Siehe auch: View Übersicht

View.PlayWalkFile

Deklaration:

```
Function PlayWalkFile(ByVal fileName as String, ByVal
UseVRDevice as Boolean) as Boolean
```

```
HRESULT PlayWalkFile(BSTR fileName, VARIANT_BOOL
UseVRDevice, VARIANT_BOOL* retVal);
```

Siehe auch: View Übersicht

View.Print

Deklaration:

```
Function Print(ByVal showDialog as Boolean, ByVal
askMultiplePages as Boolean) as Boolean
```

```
HRESULT Print(IPrintSettings *settings, VARIANT_BOOL
showDialog, VARIANT_BOOL askMultiplePages,
VARIANT_BOOL* retVal);
```

Siehe auch: View Übersicht

View.Rotate

Deklaration:

```
Function Rotate(ByVal RotCode as long, ByVal MoveEye as
Boolean, ByVal fast as Boolean) as Boolean
```

```
HRESULT Rotate(long RotCode, VARIANT_BOOL
MoveEye, VARIANT_BOOL fast, VARIANT_BOOL* retVal);
```

Siehe auch: View Übersicht

View.SavePicture

Deklaration:

```
Function SavePicture(ByVal fileName as String) as Boolean
```

```
HRESULT SavePicture(ISavePictureSettings *settings,
BSTR fileName, VARIANT_BOOL* retVal);
```

Siehe auch: View Übersicht

View.SelectPredefinedViewing3D

Deklaration:

```
Function SelectPredefinedViewing3D(ByVal index as long)
as Boolean
```

```
HRESULT SelectPredefinedViewing3D(long index,
VARIANT_BOOL* retVal);
```

Siehe auch: View Übersicht

View.SetCutViewing2D

Deklaration:

```
Function SetCutViewing2D(ByVal CutViewMode as long,
ByVal VRPx as Single, ByVal VRPy as Single, ByVal VRPz
as Single, ByVal VPDist as Single, ByVal WCWidth as
Single) as Boolean
```

```
HRESULT SetCutViewing2D(long CutViewMode, float VRPx,
float VRPy, float VRPz, float VPDist, float WCWidth,
VARIANT_BOOL* retVal);
```

Siehe auch: View Übersicht

View.SetFloorVisibility

Deklaration:

```
Function SetFloorVisibility(ByVal OnlyCurrent as Boolean) as
Boolean
```

```
HRESULT SetFloorVisibility(VARIANT_BOOL OnlyCurrent,
VARIANT_BOOL* retVal);
```

Siehe auch: View Übersicht

View.SetMode

Deklaration:

```
Sub SetMode(ByVal ViewType as long, ByVal ViewSubType
as long, ByVal ViewMode as long)
```

```
HRESULT SetMode(long ViewType, long ViewSubType,
long ViewMode);
```

Siehe auch: View Übersicht

View.SetViewing2D

Deklaration:

```
Function SetViewing2D(ByVal whichViewing as long, ByVal
VRPx as Single, ByVal VRPy as Single, ByVal VRPz as
Single, ByVal VPDist as Single, ByVal WCWidth as Single)
as Boolean
```

```
HRESULT SetViewing2D(long whichViewing, float VRPx,
float VRPy, float VRPz, float VPDist, float WCWidth,
VARIANT_BOOL* retVal);
```

Siehe auch: View Übersicht

View.SetViewing3D

Deklaration:

```
Function SetViewing3D(ByVal VRPx as Single, ByVal VRPy
as Single, ByVal VRPz as Single, ByVal EyeX as Single,
ByVal EyeY as Single, ByVal EyeZ as Single, ByVal
tanViewAngleHalf as Single) as Boolean
```

```
HRESULT SetViewing3D(float VRPx, float VRPy, float
VRPz, float EyeX, float EyeY, float EyeZ, float
tanViewAngleHalf, VARIANT_BOOL* retVal);
```

Siehe auch: View Übersicht

View.ShowAll

Deklaration:

```
Function ShowAll() as Boolean
```

```
HRESULT ShowAll(VARIANT_BOOL* retVal);
```

Siehe auch: View Übersicht

View.StartMirrorBuilding

Deklaration:

```
Function StartMirrorBuilding(ByVal canUndo as Boolean) as
Boolean
```

```
HRESULT StartMirrorBuilding(VARIANT_BOOL canUndo,
VARIANT_BOOL* retVal);
```

Siehe auch: View Übersicht

View.StartRotateBuilding

Deklaration:

```
Function StartRotateBuilding(ByVal canUndo as Boolean) as
Boolean
```

```
HRESULT StartRotateBuilding(VARIANT_BOOL canUndo,
VARIANT_BOOL* retVal);
```

Siehe auch: View Übersicht

View.StartWalking

Deklaration:

```
Function StartWalking(ByVal UseVRDevice as Boolean) as
Boolean
```

```
HRESULT StartWalking(VARIANT_BOOL UseVRDevice,
VARIANT_BOOL* retVal);
```

Siehe auch: View Übersicht

View.StartZoom

Deklaration:

```
Function StartZoom() as Boolean
```

```
HRESULT StartZoom(VARIANT_BOOL* retVal);
```

Siehe auch: View Übersicht

View.StopWalking

Deklaration:

```
Function StopWalking() as Boolean
```

```
HRESULT StopWalking(VARIANT_BOOL* retVal);
```

Siehe auch: View Übersicht

View.StopZoom

Deklaration:

Function StopZoom() as Boolean

HRESULT StopZoom(VARIANT_BOOL* retVal);

Siehe auch: View Übersicht

View.Zoom

Deklaration:

```
Function Zoom(ByVal x as long, ByVal y as long, ByVal
factor as Single) as Boolean
```

```
HRESULT Zoom(long x, long y, float factor,
VARIANT_BOOL* retVal);
```

Siehe auch: View Übersicht

Viewing

Viewing Übersicht
Eigenschaften

BCDist	Abstand der hinteren Clip-Ebene vom Betrachter
DC_WC	4x4 float Matrix zur Transformation von Bildschirmkoordinaten in Weltkoordinaten
EyeX	x Koordinate des Betrachters (Eye Point)
EyeY	y Koordinate des Betrachters (Eye Point)
EyeZ	z Koordinate des Betrachters (Eye Point)
FCDist	Abstand der vorderen Clip-Ebene vom Betrachter
NPC_DC	4x4 float Matrix zur Transformation von normalisierten View-Plane Koordinaten in Bildschirmkoordinaten
NPC_WC	4x4 float Matrix zur Transformation von normalisierten View-Plane Koordinaten in Weltkoordinaten
Parallel	Die Abbildung erfolgt in Parallel-Projektion
Reverse	Die Abbildung steht auf dem Kopf
VPDist	Abstand der View-Plane vom Betrachter (Abstand zwischen Eye und VRP)
VRPx	x Koordinate des Blickpunktes (View Reference Point)
VRPy	y Koordinate des Blickpunktes (View Reference Point)
VRPz	z Koordinate des Blickpunktes (View Reference Point)
VUPx	x Anteil des View Up Vectors
VUPy	y Anteil des View Up Vectors
VUPz	z Anteil des View Up Vectors
WC_DC	4x4 float Matrix zur Transformation von Weltkoordinaten in Bildschirmkoordinaten
WC_NPC	4x4 float Matrix zur Transformation von Weltkoordinaten in normalisierte Koordinaten auf der View-Plane
WC_NPCX	4x4 float Matrix zur Transformation von Weltkoordinaten Koordinaten auf der View-Plane
WCxmax	x Weltkoordinate der rechten Seite des sichtbaren View-Plane Rechtecks
WCxmin	x Weltkoordinate der linken Seite des sichtbaren View-Plane Rechtecks
WCymax	y Weltkoordinate der oberen Seite des sichtbaren View-Plane Rechtecks
WCymin	y Weltkoordinate der unteren Seite des sichtbaren View-Plane Rechtecks

Viewing.BCDist

Deklaration:

Single (nur lesbar)

HRESULT get_BCDist(float* retVal);

Siehe auch: Viewing Übersicht

Viewing.DC_WC

Deklaration:

VARIANT (nur lesbar)

HRESULT get_DC_WC(VARIANT* retVal);

Siehe auch: Viewing Übersicht

Viewing.EyeX

Deklaration:

Single (nur lesbar)

HRESULT get_EyeX(float* retVal);

Siehe auch: Viewing Übersicht

Viewing.EyeY

Deklaration:

Single (nur lesbar)

HRESULT get_EyeY(float* retVal);

Siehe auch: Viewing Übersicht

Viewing.EyeZ

Deklaration:

Single (nur lesbar)

HRESULT get_EyeZ(float* retVal);

Siehe auch: Viewing Übersicht

Viewing.FCDist

Deklaration:

Single (nur lesbar)

HRESULT get_FCDist(float* retVal);

Siehe auch: Viewing Übersicht

Viewing.NPC_DC

Deklaration:

VARIANT (nur lesbar)

HRESULT get_NPC_DC(VARIANT* retVal);

Siehe auch: Viewing Übersicht

Viewing.NPC_WC

Deklaration:

VARIANT (nur lesbar)

HRESULT get_NPC_WC(VARIANT* retVal);

Siehe auch: Viewing Übersicht

Viewing.Parallel

Deklaration:

Boolean

```
HRESULT get_Parallel(VARIANT_BOOL* retVal);
HRESULT put_Parallel(VARIANT_BOOL newVal);
```

Siehe auch: Viewing Übersicht

Viewing.Reverse

Deklaration:

Boolean (nur lesbar)

HRESULT get_Reverse(VARIANT_BOOL* retVal);

Siehe auch: Viewing Übersicht

Viewing.VPDist

Deklaration:

Single (nur lesbar)

HRESULT get_VPDist(float* retVal);

Siehe auch: Viewing Übersicht

Viewing.VRPx

Deklaration:

Single (nur lesbar)

HRESULT get_VRPx(float* retVal);

Siehe auch: Viewing Übersicht

Viewing.VRPy

Deklaration:

Single (nur lesbar)

HRESULT get_VRPy(float* retVal);

Siehe auch: Viewing Übersicht

Viewing.VRPz

Deklaration:

Single (nur lesbar)

HRESULT get_VRPz(float* retVal);

Siehe auch: Viewing Übersicht

Viewing.VUPx

Deklaration:

Single (nur lesbar)

HRESULT get_VUPx(float* retVal);

Siehe auch: Viewing Übersicht

Viewing.VUPy

Deklaration:

Single (nur lesbar)

HRESULT get_VUPy(float* retVal);

Siehe auch: Viewing Übersicht

Viewing.VUPz

Deklaration:

Single (nur lesbar)

HRESULT get_VUPz(float* retVal);

Siehe auch: Viewing Übersicht

Viewing.WC_DC

Deklaration:

VARIANT (nur lesbar)

HRESULT get_WC_DC(VARIANT* retVal);

Siehe auch: Viewing Übersicht

Viewing.WC_NPC

Deklaration:

VARIANT (nur lesbar)

HRESULT get_WC_NPC(VARIANT* retVal);

Siehe auch: Viewing Übersicht

Viewing.WC_NPCX

Deklaration:

VARIANT (nur lesbar)

HRESULT get_WC_NPCX(VARIANT* retVal);

Siehe auch: Viewing Übersicht

Viewing.WCxmax

Deklaration:

Single (nur lesbar)

HRESULT get_WCxmax(float* retVal);

Siehe auch: Viewing Übersicht

Viewing.WCxmin

Deklaration:

Single (nur lesbar)

HRESULT get_WCxmin(float* retVal);

Siehe auch: Viewing Übersicht

Viewing.WCymax

Deklaration:

Single (nur lesbar)

HRESULT get_WCymax(float* retVal);

Siehe auch: Viewing Übersicht

Viewing.WCymin

Deklaration:

Single (nur lesbar)

HRESULT get_WCymin(float* retVal);

Siehe auch: Viewing Übersicht

SnapObject

In ArCon können 3D Objekte (ähnlich wie viele Dinge im Konstruktionsmodus) “ fangen” bzw. aneinander Schnappen. Das hilft bei der Positionierung von Möbeln: z.B. schnappt der Schrank in die Raumecke und steht anschließend dort exakt passend. Oder das zweite Schrankelement schnappt an das erste und steht damit fugenfrei angepaßt.

Bei den meisten Objekten (die in der Regel ungefähr Quaderförmige Aussenkonturen haben) genügt es, an der Bounding-Box zu schnappen – das ist die Voreinstellung für alle Objekte ohne nähere Schnappinformation.

Bei einigen Objekten müssen die Schnappdetails aber genauer bestimmt werden, um eine sinnvolle Verwendung der Schnappfunktionalität zu ermöglichen.

Sollten diese Objekte per Programm erzeugt worden sein, bietet es sich an, auch die Schnappinformation per Programm zu erzeugen. Aber natürlich können auch zu aus Dateien geladenen Objekten nachträglich Fangdaten gesetzt werden.

Das SnapObject hat dabei die Aufgabe, die Fanginformation stückweise zu sammeln und anschließend in einem Stück dem zugehörigen 3D Objekt zu übergeben. Dazu wird es an eine der Methoden ObjectConstructor.InitSnapObject oder Object3D.InitSnapObject übergeben. Anschließend kann das SnapObject mit Hilfe der Methode Clear zurückgesetzt und für den Aufbau einer weiteren Fanginformation wiederverwendet werden.

SnapObject Übersicht

Methoden

AddArea	Fügt eine Fangfläche hinzu
AddEdge	Fügt eine Fangkante hinzu
AddPoint	Fügt einen Fangpunkt hinzu
Clear	Löscht alle Daten einer Fangbeschreibung
SetBoundingBox	Setzt die Bounding-Box des Fangobjektes. Nur auf Objekte, deren Bounding-Box nah genug ist wird gefangen
SetReferencePoint	Setzt den Referenzpunkt (in der Regel der Ursprung des Modellkoordinatensystems)

SnapObject.AddArea

Deklaration:

Sub AddArea(ByVal p1X as Single, ByVal p1y as Single, ByVal p1z as Single, ByVal p2X as Single, ByVal p2y as Single, ByVal p2z as Single, ByVal p3X as Single, ByVal p3y as Single, ByVal p3z as Single, ByVal p4X as Single, ByVal p4y as Single, ByVal p4z as Single, ByVal flags as long, ByVal SnapClassIsA as long, ByVal SnapClassSnapsOn as long, ByVal SnapClassPrefered as long)

HRESULT AddArea(float p1X, float p1y, float p1z, float p2X, float p2y, float p2z, float p3X, float p3y, float p3z, float p4X, float p4y, float p4z, long flags, long SnapClassIsA, long SnapClassSnapsOn, long SnapClassPrefered);

Siehe auch: SnapObject Übersicht

Fügt eine dreieckige oder viereckige Fläche der Fanginformation hinzu.

Als Flags können Kombination der folgenden Bitmasken übergeben werden:

AC_SNAP_A_ISQUAD	Die übergebene Fläche ist ein Viereck, der letzte Punkt wird ausgewertet. Wenn dieses Bit nicht gesetzt ist, handelt es sich um ein Dreieck und die Koordinaten des letzten Punktes werden ignoriert.
AC_SNAP_A_SNAP_P0	Der erste Punkt ist (zusätzlich zur Fläche) ein Fangpunkt
AC_SNAP_A_SNAP_P1	Der zweite Punkt ist (zusätzlich zur Fläche) ein Fangpunkt
AC_SNAP_A_SNAP_P2	Der dritte Punkt ist (zusätzlich zur Fläche) ein Fangpunkt
AC_SNAP_A_SNAP_P3	Der vierte ist (zusätzlich zur Fläche) ein Fangpunkt
AC_SNAP_A_SNAP_E0	Die erste Kante (p1-p2) ist zusätzlich eine Fangkante
AC_SNAP_A_SNAP_E1	Die zweite Kante (p2-p3) ist zusätzlich eine Fangkante
AC_SNAP_A_SNAP_E2	Die dritte Kante (p3-p4 oder p3-p1) ist zusätzlich eine Fangkante
AC_SNAP_A_SNAP_E3	Die vierte Kante (p4-p1) ist zusätzlich eine Fangkante
AC_SNAP_A_ISTWOSIDED	Die Fläche fängt beidseitig (nicht nur “ von links”)

Die Parameter sind:

p1x, p1y, p1z	Koordinaten des ersten Punktes
p2x, p2y, p2z	Der zweite Punkt
p3x, p3y, p3z	Der dritte Punkt
p4x, p4y, p4z	Der vierte Punkt (evtl. ignoriert)
flags	siehe oben
SnapClassIsA	Dieses Objekt gehört zu dieser/n Fangklasse(n)
SnapClassSnapsOn	Dieses Objekt fängt auf Mitglieder dieser Fangklasse (n)
SnapClassPrefered	Die Fangklasse, auf die das Objekt bevorzugt fängt

SnapObject.AddEdge

Deklaration:

Sub AddEdge(ByVal p1X as Single, ByVal p1y as Single, ByVal p1z as Single, ByVal p2x as Single, ByVal p2y as Single, ByVal p2z as Single, ByVal flags as long, ByVal dirX as Single, ByVal dirY as Single, ByVal dirZ as Single, ByVal SnapClassIsA as long, ByVal SnapClassSnapsOn as long, ByVal SnapClassPrefered as long)

HRESULT AddEdge(float p1X, float p1y, float p1z, float p2x, float p2y, float p2z, long flags, float dirX, float dirY, float dirZ, long SnapClassIsA, long SnapClassSnapsOn, long SnapClassPrefered);

Siehe auch: SnapObject Übersicht

Fügt eine Kante der Fanginformation hinzu.

Als Flags können Kombination der folgenden Bitmasken übergeben werden:

AC_SNAP_E_USE_DIR1	Der Richtungsvektor dirX/dirY/dirZ wird benutzt, um der Fanginformation eine Richtung zu geben (ohne dieses Bit wird dieser Vektor ignoriert)
AC_SNAP_E_SNAP_P0	Der erste Punkt ist zusätzlich ein Fangpunkt
AC_SNAP_E_SNAP_P1	Der zweite Punkt ist zusätzlich ein Fangpunkt
AC_SNAP_E_ISTWOSIDED	Die Kante fängt auch in der entgegengesetzten Richtung (-dirX/-dirY/-dirZ)

Die Parameter sind:

p1x, p1y, p1z	Koordinaten des ersten Punktes
p2x, p2y, p2z	Der zweite Punkt
flags	siehe oben
dirX, dirY, dirZ	Richtung, in der gefangen wird
SnapClassIsA	Dieses Objekt gehört zu dieser/n Fangklasse(n)
SnapClassSnapsOn	Dieses Objekt fängt auf Mitglieder dieser Fangklasse (n)
SnapClassPrefered	Die Fangklasse, auf die das Objekt bevorzugt fängt

SnapObject.AddPoint

Deklaration:

Sub AddPoint(ByVal pointX as Single, ByVal pointY as Single, ByVal pointZ as Single, ByVal flags as long, ByVal dir1x as Single, ByVal dir1y as Single, ByVal dir1z as Single, ByVal dir2x as Single, ByVal dir2y as Single, ByVal dir2z as Single, ByVal SnapClassIsA as long, ByVal SnapClassSnapsOn as long, ByVal SnapClassPrefered as long)

HRESULT AddPoint(float pointX, float pointY, float pointZ, long flags, float dir1x, float dir1y, float dir1z, float dir2x, float dir2y, float dir2z, long SnapClassIsA, long SnapClassSnapsOn, long SnapClassPrefered);

Siehe auch: SnapObject Übersicht

Fügt der Fanginformation einen Fangpunkt hinzu.

Als Flags können Kombination der folgenden Bitmasken übergeben werden:

AC_SNAP_P_USE_DIR1	Der Richtungsvektor dir1x/dir1y/dir1z wird benutzt, um der Fanginformation eine Richtung zu geben (ohne dieses Bit wird dieser Vektor ignoriert)
AC_SNAP_P_USE_DIR2	Der zweite Richtungsvektor gibt eine alternative Fangrichtung an.
AC_SNAP_P_ISTWOSIDED	Die Kante fängt auch in der entgegengesetzten Richtung (-dir1x/-dir1y/-dir1z)

Die Parameter sind:

p1x, p1y, p1z	Koordinaten des Punktes
flags	siehe oben
dir1x, dir1y, dir1z	1. Richtung, in der gefangen wird
dir2x, dir2y, dir2z	2. Richtung, in der gefangen wird
SnapClassIsA	Dieses Objekt gehört zu dieser/n Fangklasse(n)
SnapClassSnapsOn	Dieses Objekt fängt auf Mitglieder dieser Fangklasse (n)
SnapClassPrefered	Die Fangklasse, auf die das Objekt bevorzugt fängt

SnapObject.Clear

Deklaration:

```
Sub Clear()
```

```
HRESULT Clear();
```

Siehe auch: SnapObject Übersicht

Löscht alle Fanginformationen. Nachdem ein SnapObject definiert wurde und mit Object3D.InitSnapObject oder ObjectConstructor.InitSnapObject an das 3D Objekt übergeben wurde, kann das SnapObject mit dieser Methode wieder in den ursprünglichen, leeren Zustand versetzt werden und zur Aufbereitung weiterer Fanginformation wiederverwendet werden.

SnapObject.SetBoundingBox

Deklaration:

```
Sub SetBoundingBox(ByVal xMin as Single, ByVal
yMin as Single, ByVal zMin as Single, ByVal xMax as
Single, ByVal yMax as Single, ByVal zMax as Single)
```

```
HRESULT SetBoundingBox(float xMin, float yMin,
float zMin, float xMax, float yMax, float zMax);
```

Siehe auch: SnapObject Übersicht

Setzt die Bounding-Box der Fanginformation. Nur Fangobjekte, deren Bounding-Box nah genug an der aktuellen Mausposition ist, werden beim Fangen berücksichtigt.

Die Parameter sind:

xMin, yMin, zMin	Koordinaten der einen Ecke
xMax, yMax, zMax	Koordinaten der gegenüberliegenden Ecke

SnapObject.SetReferencePoint

Deklaration:

```
Sub SetReferencePoint(ByVal x as Single, ByVal y as
Single, ByVal z as Single)
```

```
HRESULT SetReferencePoint(float x, float y, float z);
```

Siehe auch: SnapObject Übersicht

Um zu entscheiden, ob ein Objekt zum Fangen in Frage kommt, benutzt ArCon einen (einstellbaren) Abstand in Bildschirmkoordinaten (Pixel). Ein Fangobjekt hat aber eine räumliche Ausdehnung, und die Berechnung des kürzesten Abstandes aller möglichen Fangpunkte wäre zu aufwendig. Daher wird zur Entscheidung nur der Referenzpunkt des Fangobjektes herangezogen.

Sollte der Schwerpunkt Ihres Fangobjektes nicht im Ursprung des Modellkoordinatensystems liegen, setzen Sie mit dieser Methode die Koordinaten des Schwerpunktes im Modellkoordinatensystem.

Die Parameter sind:

x, y, z	Die Koordinaten des Referenzpunktes

BackgroundSettings

BackgroundSettings Übersicht

Eigenschaften

BackPictureFile	Dateiname eines eventuellen Hintergrundbildes
Color	Hintergrundfarbe (im AC_CM_CONSTANT Modus)
ColorDown	Hintergrundfarbe für Blickrichtung nach unten (im AC_CM_TWORAMPS Modus)
ColorHorizonDown	Horizontfarbe für Erde
ColorHorizonUp	Horizontfarbe für Himmel
ColorUp	Hintergrundfarbe für Blickrichtung nach oben (im AC_CM_TWORAMPS Modus)
ForegroundHeight	Höhe des Vordergrundbildes
ForegroundOffsetX	Pixeloffset (X) des Vordergrundbildes
ForegroundOffsetY	Pixeloffset (Y) des Vordergrundbildes
ForegroundOrientationX	X Orientierung des Vordergrundbildes
ForegroundOrientationY	Y Orientierung des Vordergrundbildes
ForegroundPictureFile	Dateiname eines eventuellen Vordergundbildes
ForegroundWidth	Breite des Vordergrundbildes
HorizontalViewAngle	Bei Anzeige eines Hintergrundbildes der horizontale Aufnahmewinkel der (virtuellen) Kamera dieses Hintergrundbildes
Mode	Art der Hintergrunddarstellung
MovePicture	Bewegt das Hintergrundbild sich mit dem Gebäude oder ist es fix
PosOfHorizon	Höhe des Horizontes (0.0 = ganz unten, 1.0 = ganz oben)
UseForegroundPicture	Aktiviert die Einblendung des Vordergrundbildes
WallpaperX	Versatz in X Richtung (Pixel) des Hintergrundbildes im AC_CM_WALLPAPER Modus
WallpaperY	Versatz in Y Richtung (Pixel) des Hintergrundbildes im AC_CM_WALLPAPER Modus

Methoden

EnableUpdate	Änderungen werden gesammelt, bis hiermit die Transaktion freigegeben wird

BackgroundSettings.BackPictureFile

Deklaration:

String

HRESULT get_BackPictureFile(BSTR* retVal);
HRESULT put_BackPictureFile(BSTR newVal);

Siehe auch: BackgroundSettings Übersicht

BackgroundSettings.Color

Deklaration:

long

HRESULT get_Color(long* retVal);
HRESULT put_Color(long newVal);

Siehe auch: BackgroundSettings Übersicht

BackgroundSettings.ColorDown

Deklaration:

long

HRESULT get_ColorDown(long* retVal);
HRESULT put_ColorDown(long newVal);

Siehe auch: BackgroundSettings Übersicht

BackgroundSettings.ColorHorizonDown

Deklaration:

long

HRESULT get_ColorHorizonDown(long* retVal);
HRESULT put_ColorHorizonDown(long newVal);

Siehe auch: BackgroundSettings Übersicht

BackgroundSettings.ColorHorizonUp

Deklaration:

long

HRESULT get_ColorHorizonUp(long* retVal);
HRESULT put_ColorHorizonUp(long newVal);

Siehe auch: BackgroundSettings Übersicht

BackgroundSettings.ColorUp

Deklaration:

long

HRESULT get_ColorUp(long* retVal);
HRESULT put_ColorUp(long newVal);

Siehe auch: BackgroundSettings Übersicht

BackgroundSettings.ForegroundHeight

Deklaration:

long

HRESULT get_ForegroundHeight(long* retVal);
HRESULT put_ForegroundHeight(long newVal);

Siehe auch: BackgroundSettings Übersicht

BackgroundSettings.ForegroundOffsetX

Deklaration:

long

HRESULT get_ForegroundOffsetX(long* retVal);
HRESULT put_ForegroundOffsetX(long newVal);

Siehe auch: BackgroundSettings Übersicht

BackgroundSettings.ForegroundOffsetY

Deklaration:

long

HRESULT get_ForegroundOffsetY(long* retVal);
HRESULT put_ForegroundOffsetY(long newVal);

Siehe auch: BackgroundSettings Übersicht

BackgroundSettings.ForegroundOrientationX

Deklaration:

long

HRESULT get_ForegroundOrientationX(long* retVal);
HRESULT put_ForegroundOrientationX(long newVal);

Siehe auch: BackgroundSettings Übersicht

BackgroundSettings.ForegroundOrientationY

Deklaration:

long

HRESULT get_ForegroundOrientationY(long* retVal);
HRESULT put_ForegroundOrientationY(long newVal);

Siehe auch: BackgroundSettings Übersicht

BackgroundSettings.ForegroundPictureFile

Deklaration:

String

HRESULT get_ForegroundPictureFile(BSTR* retVal);
HRESULT put_ForegroundPictureFile(BSTR newVal);

Siehe auch: BackgroundSettings Übersicht

BackgroundSettings.ForegroundWidth

Deklaration:

long

HRESULT get_ForegroundWidth(long* retVal);
HRESULT put_ForegroundWidth(long newVal);

Siehe auch: BackgroundSettings Übersicht

BackgroundSettings.HorizontalViewAngle

Deklaration:

Single

HRESULT get_HorizontalViewAngle(float* retVal);
HRESULT put_HorizontalViewAngle(float newVal);

Siehe auch: BackgroundSettings Übersicht

BackgroundSettings.Mode

Deklaration:

AC_ArCon_Background_Modi

HRESULT get_Mode(AC_ArCon_Background_Modi* retVal)
;
HRESULT put_Mode(AC_ArCon_Background_Modi newVal)
;

Siehe auch: BackgroundSettings Übersicht

BackgroundSettings.MovePicture

Deklaration:

Boolean

```
HRESULT get_MovePicture(VARIANT_BOOL* retVal);
HRESULT put_MovePicture(VARIANT_BOOL newVal);
```

Siehe auch: BackgroundSettings Übersicht

BackgroundSettings.PosOfHorizon

Deklaration:

Single

HRESULT get_PosOfHorizon(float* retVal);
HRESULT put_PosOfHorizon(float newVal);

Siehe auch: BackgroundSettings Übersicht

BackgroundSettings.UseForegroundPicture

Deklaration:

Boolean

HRESULT get_UseForegroundPicture(VARIANT_BOOL* retVal);

HRESULT put_UseForegroundPicture(VARIANT_BOOL newVal);

Siehe auch: BackgroundSettings Übersicht

BackgroundSettings.WallpaperX

Deklaration:

Single

HRESULT get_WallpaperX(float* retVal);
HRESULT put_WallpaperX(float newVal);

Siehe auch: BackgroundSettings Übersicht

BackgroundSettings.WallpaperY

Deklaration:

Single

HRESULT get_WallpaperY(float* retVal);
HRESULT put_WallpaperY(float newVal);

Siehe auch: BackgroundSettings Übersicht

BackgroundSettings.EnableUpdate

Deklaration:

```
Sub EnableUpdate(ByVal nVal as Boolean)
```

```
HRESULT EnableUpdate(VARIANT_BOOL nVal);
```

Siehe auch: BackgroundSettings Übersicht

MAKROS WEITERGEBEN

Falls Sie Makros nicht nur für den eigenen Bedarf programmieren, sollten Sie folgend Hinweise bei der Zusammenstellung einer Distribution beachten. Wir gehen davon aus, daß Sie ein Installationsprogramm erstellen, das dem Anwender eine funktionsfähige ArCon-Installation - mit Ihren Markros integriert - hinterläßt.

Die Installation erfolgt in zwei Schritten:

Kopieren der Makro-Dateien auf den Installationscomputer

Eintragen der neuen Makros in der ArCon Konfigurationsdatei Makros.INI

Makro-Dateien

Installieren Sie die ausführbaren Dateien Ihres Makros in das “ Programm\Standard\Makros” Unterverzeichnis der ArCon Installation. Nur dort werden Sie automatisch in das Makro-Menü aufgenommen bzw. können per Makros.INI Eintrag automatisch oder bei Bedarf automatisch gestartet werden.

Falls Ihr Makro aus vielen Dateien besteht, installieren Sie nur die direkte ArCon Schnittstelle in diesem Verzeichnis und legen ein eigenes Installationsverzeichnis für die übrigen Dateien an.

Makros.INI

Ihre Makros sollten in der Makros.INI Datei eingetragen werden.

Makros werden von ArCon automatisch im Menü “ Makros” aufgeführt, wenn die EXE Datei des Makros in das Verzeichnis ...Programm\Standard\Makros kopiert wird. Zusätzliche Informationen können optional in der Markos.INI Datei - ebenfalls in diesem Verzeichnis - angegeben werden.

Diese Datei besteht aus einzelnen Abschnitten, die durch den Namen des Makros gekennzeichnet sind. Heißt z.B. Ihr Makro “ Katalog.EXE” , können Sie folgenden Abschnitt in der Makros.INI Datei eintragen:

```
; ------------------------------------------------
[Katalog.EXE]
Name = &Katalog-Beispiel
Beschreibung = Fertighauskatalog
; Für dieses Makro gibt es eine Hilfedatei. Der Kontext
; zum Aufruf des Makros hat die Nummer 4712
Hilfedatei = Katalog.hlp
HilfeNr    = 4712
```

Ein ; leitet einen Kommentar ein, der Rest der Zeile wird von ArCon ignoriert. Alle Zeilen haben die Form

Schlüsselwort = Wert

Alle Einträge sind optional.

ArCon kennt folgende Schlüsselworte:

Schlüsselwort	**Bedeutung des Wertes**
Name	Der Name des Menüeintrages. Ein “ &” -Zeichen erzeugt einen “ Hotkey” , der im Menü durch einen Unterstrich des folgenden Buchstabens dargestellt wird. Durch Drücken der entsprechenden Taste können Sie direkt zu diesem Menüpunkt springen und ihn aktivieren. Wenn dieser Eintrag fehlt, benutzt ArCon den Namen der EXE-Datei (ohne .EXE) als Name.
Beschreibung	Die Beschreibung des Makros. Dieser Text wird in der Statuszeile angezeigt, wenn die Menüauswahl auf diesen Menüpunkt zeigt.
Hilfedatei	Gibt den Dateinamen einer Windows-Hilfedatei an, in der Hilfe zu diesem Menüpunkt gefunden werden kann.
HilfeNr	Kontext (ID) des Hilfepunktes, der den Menüeintrag erläutert.
Modi	Eine Bitmaske, die angibt, in welchen ArCon Modi dieser Menüeintrag gültig ist. Addieren Sie dazu folgende Werte: 1 = kein Projekt geladen, 2 = Konstruktionsmodus, 4 = Designmodus. Wenn dieser Eintrag fehlt, nimmt ArCon 7 an (alle Modi)
Automatisch	Das Makro wird automatisch gestartet. ArCon erzeugt keinen Menüeintrag. In der Regel wird das Makro unsichtbar bleiben und sich in der ArCon Oberfläche installieren, sodaß es beim Öffnen bestimmter Dialoge oder Anwahl von Menüpunkten aktiviert wird.
Ignorieren	Dieses Makro wird ignoriert. ArCon erzeugt keinen Menüeintrag. Auf diese Weise können Sie EXE-Dateien, die aus anderen Gründen im Makros-Verzeichnis liegen aber keine ArCon-

Makros sind, vor ArCon verbergen.

fehlermeldungen

Das folgende Kapitel beschreibt alle Fehlermeldungen, die vom ArCon Active-X Control zur Laufzeit ausgegeben werden können.

Dieses Objekt kann nicht als 2D Grafikelement benutzt werden

Ungültiger Dateiname, die Datei kann nicht erstellt werden

Datei nicht gefunden

Parameter ist keine 4x4 Matrix...

Parameter ist kein Array aus Objekten

Parameter ist keine Punktliste!

Parameter ist kein Lochpolygon!

Der Parameter enthält kein gültiges ArCon-Bild

Dieses Objekt unterstützt keine Änderungsereignisse!

Eine polygonale Wendeltreppe kann nur aus einem Polygon mit maximal 30 Punkten erstellt werden!

Dieses Objekt kann nicht als 2D Grafikelement benutzt werden

Sie haben versucht, ein Objekt in eine der 2D-Grafik-Listen einzufügen, das nicht dazu geeignet ist.

Verwenden Sie ausschließlich 2D Grafikelemente für diese Listen. Dazu gehören: Image, Label, Shape, Line und Polygon2D.

Ungültiger Dateiname, die Datei kann nicht erstellt werden

Beim Aufruf der Methode ReadChunk haben Sie einen Dateinamen angegeben, in den die gelesenen Daten exportiert werden sollten. ArCon war nicht in der Lage, unter dem angegebenen Namen eine Datei zu erstellen. Prüfen Sie, ob das angegebene Verzeichnis existiert oder die Festplatte voll ist.

Datei nicht gefunden!

Beim Aufruf der Methode WriteChunk haben Sie einen Dateinamen angegeben, aus dem Daten in das ArCon Projekt importiert werden sollten. ArCon hat die angegebene Datei nicht gefunden oder konnte Sie aus anderen Gründen (z.B. fehlende Berechtigungen) nicht lesen.

Parameter ist keine 4x4 Matrix...

Bei der Angabe von Transformationsmatritzen müssen Sie 4x4 große Matritzen mit einfach genauen Fließkommazahlen als Parameter verwenden. In Basic deklarieren Sie solche Matritzen zum Beispiel als

```
Dim matrix(3,3) As Single
```

Beachten Sie, daß Basic-Arrays bei Index 0 beginnen, sofern Sie keine Option Base Anweisung verwendet haben, um die untere Grenze auf einen anderen Wert zu legen. In diesem Fall müssten Sie die Obergrenzen entsprechend erhöhen.

In C/C++ ist die Deklaration eines solchen Parameters etwas aufwendiger:

```
{
  typedef float matrix[4][4];
  matrix einheit =
  {
    { 1.0f, 0.0f, 0.0f, 0.0f },
    { 0.0f, 1.0f, 0.0f, 0.0f },
    { 0.0f, 0.0f, 1.0f, 0.0f },
    { 0.0f, 0.0f, 0.0f, 1.0f }
  }, *pMat;

  SAFEARRAYBOUND bounds[2];
   bounds[0].lLbound = 0;
   bounds[0].cElements = 4;
   bounds[1].lLbound = 0;
   bounds[1].cElements = 4;
   SAFEARRAY *pArr = SafeArrayCreate(VT_R4, 2, bounds);
  SafeArrayAccessData(pArr, (void**)&pMat);
  memcpy(pMat, &einheit, sizeof einheit);
  SafeArrayUnaccessData(pArr);

  VARIANT v;
  VariantInit(&v);
  v.vt = VT_ARRAY|VT_R4;
  v.parray = pArr;

  // CObject3D.SetModelToWorldTransformation(v);
  // CObject3D.InsertIntoWorld(FALSE);

  SafeArrayDestroy(pArr);
}
```

Parameter ist kein Array aus Objekten

Beim Gruppieren von 3D-Objekten müssen Sie ein Array aus Objekten übergeben, das in Visual Basic z.B. mit der folgenden Deklaration erzeugen können:

```
Dim objs(500) As Object
```

In C/C++ verwenden Sie ein VARIANT mit der Kennung VT_DISPATCH|VT_ARRAY, das Sie z. B. so anlegen können:

```
SAFEARRAYBOUND bounds[1];
    bounds[0].lLbound = 0;
    bounds[0].cElements = 500;
    SAFEARRAY *pArr = SafeArrayCreate(VT_DISPATCH, 1, bounds);
  SafeArrayAccessData(pArr, (void**)&ppDisp);
  memcpy(ppDisp, src, size);
  SafeArrayUnaccessData(pArr);

  VARIANT v;
  VariantInit(&v);
  v.vt = VT_ARRAY|VT_DISPATCH;
  v.parray = pArr;
  // ...
  SafeArrayDestroy(pArr);
```

Parameter ist keine Punktliste!

Sie versuchen eine Kontur zu definieren und haben als “ Points” Parameter kein Array aus einfach genauen Fließkommazahlen angegeben.

Die Methode SetPoints erwartet ein eindimensionales Array, indem abwechselnd x- und y-Koordinaten der Punkte untergebracht sind.

Parameter ist kein Lochpolygon!

Die Method SetHoleContur erwartet als Parameter ein zweidimensionales Array aus einfach genauen Fließkommazahlen, dessen erste Dimension genau 5 Werte aufnimmt (x, y und z Koordinaten sowie die Texturkoordinaten u und v) und dessen zweite Dimension Platz genug für die im Parameter “ numPoints” übergebene Anzahl von Punkten bietet.

Der Parameter enthält kein gültiges ArCon-Bild!

Bilder werden als Varianten übergeben, die Sie am besten mit Hilfe der Bibliothek MakroUtil.dll erstellen. Die Funktion ArConPictureFromBitmap konvertiert ein Bitmap-Handle (HBITMAP) in eine für ArCon verwertbare Variant-Variable.

Die hier übergebene Variante enthält keine für ArCon verständliche Bilddarstellung.

Dieses Objekt unterstützt keine Änderungsereignisse!

Sie können das hier übergebene Objekt nicht für ArCon-Änderungsereignisse anmelden.

Nicht alle Objekte unterstützen diesen Dienst, zum Beispiel wenn es für den Benutzer oder andere Makros keinerlei Möglichkeiten gibt, ein Objekt zu verändern (weil es inhärent Ihrem Makro gehört und nur von diesem verändert werden kann).

Prüfen Sie auch, ob das übergebene Objekt tatsächlich ein ArCon-Objekt ist!

Eine polygonale Wendeltreppe kann nur aus einem Polygon mit maximal 30 Punkten erstellt werden!

In der aktuellen Version unterstüzt ArCon polygonale Wendeltreppen nur mit einer maximalen Komplexität von 30 Polygonpunkten. Ihr Programm hat versucht, aus einem größeren Polygon eine solche Wendeltreppe zu erzeugen oder das Polygon einer vorhandenen Wendeltreppe durch ein komplexeres Polygon zu ersetzen. Dies ist nicht möglich. Benutzen Sie ein kleineres Polygon.

GLOBALE KONSTANTEN

Die Eigenschaften und Funktionen des ArCon Active-X Controls enthalten häufig Parameter, die lediglich als “ long” deklariert sind und denen Sie “ magische Zahlen” zuweisen müssen. Um diese Zahlen auch später noch verstehen zu können, sollten Sie symbolische Konstanten dafür verwenden.

Sie finden eine Definition dieser Konstanten in der mitgelieferten Datei “ acGlobal.bas” für Visual Basic, “ acGlobal.h” für C/C++ sowie “ acGlobal.pas” für Delphi. Weiterhin sind diese Konstantendefinitionen Bestandteil der Active-X Typinformation des ArCon Controls, und daher in Entwicklungsumgebungen wie Visual Basic bereits bekannt, sobald das Active-X Control dem Projekt hinzugefügt wurde. In anderen Umgebungen, z.B. Visual J++, werden die Konstanten beim Import der Typinformation des Active-X Controls mit importiert. In Visual C++ und Delphi geschieht dies leider nicht, daher müssen Sie dort die oben aufgeführten Dateien in Ihr Projekt einbinden. Siehe dazu auch die Startbeispiele für verschiedene Programmiersprachen.

Die direkte Verwendung der Active-X Typinformation hat einen großen Vorteil: da das Active-X Control erst zur Laufzeit Ihres Programmes geladen wird, kann Ihr Programm auch mit zukünftigen Versionen von ArCon und dem Active-X Control zusammenarbeiten und benutzt dann automatisch eventuell aktualisierte Werte der Konstanten (z.B. wenn neue Wandtypen hinzugekommen sind).

In anderen Programmierumgebungen (z.B. Visual C++) ist dies nicht so einfach möglich, weshalb Sie hier in der Regel auf die statische Datei zurückgreifen werden (und z.B. per #include <acGlobal.h> die Konstanten verfügbar machen).

Mit ein wenig Aufwand ist aber auch in C/C++ die dynamische Einbindung aus der Typinformation möglich.

Beispiel für dynamisch geladene Konstanten in Visual C++

Zeichnungsebenen

Für die verschiedenen 2D Grafikelemente sind in ArCon unterschiedliche Zeichnungsebenen definiert. Sie finden diese globalen Konstanten in der Datei “ acGlobal.bas” . Binden Sie dieses Modul in Ihr Projekt ein, um auf die vordefinierten Werte zugreifen zu können.

2D Grafiken werden in der folgenden Reihenfolge von hinten nach vorne gezeichnet. Alle später gezeichneten (weiter vorne liegenden) Objekte können Ihre Grafikelemente verdecken, währen Ihre Grafikelemente alle bereits gezeichneten (weiter hinten liegenden) Objekte übermalen.

Layer-Konstante	Bedeutung
AC_LayerGrid	Elemente in dieser Ebene werden noch vor den Rasterpunkten gezeichnet und daher von diesen auch überdeckt.
AC_LayerDesignobjects	Elemente werden vor den “ Skizzen” der 3D Objekte gezeichnet.
AC_LayerFilm	Elemente werden vor den Folien gezeichnet.
AC_LayerTerrain	Elemente werden vor dem Gelände gezeichnet
AC_LayerFirewalls	Elemente werden vor Brandquerschnitten gezeichnet. (Diese Funktionalität ist nur in ArCon Spezialversionen verfügbar.)
AC_LayerObjectcluster	Elemente werden vor Rauchgruppen gezeichnet. (Diese Funktionalität ist nur in ArCon Spezialversionen verfügbar.)
AC_LayerFloor	Elemente werden vor den konstruktiven Elementen aller sichtbaren Stockwerke (Wände, Türen, Fenster, Treppen, ...) gezeichnet.
AC_LayerGuide	Elemente werden vor Hilfslinien gezeichnet.
AC_LayerMeasurement	Elemente werden vor Vermaßungen gezeichnet.
AC_LayerLabeling	Elemente werden vor Beschriftungen gezeichnet.
AC_LayerRuler	Elemente werden vor dem Lineal gezeichnet.
AC_LayerOrigin	Elemente werden vor der Ursprungsmarkierung gezeichnet.
AC_LayerCompass	Elemente werden vor dem Nordpfeil gezeichnet.
AC_LayerCuts	Elemente werden vor der Darstellung von Schnitten gezeichnet.
AC_LayerSchriftfeld	Elemente werden vor dem Schriftfeld gezeichnet.
AC_LayerLast	Elemente werden als letzte Ebene gezeichnet und überdecken damit alle ArCon Zeichnungen.

State Bits

Diese Bit’ s beschreiben die sichtbaren Dinge im Konstruktionsmodus.

Sie können mit “ ode” ´ bzw. “ +” zusammengefaßt werden und ergeben so Werte für die Eigenschaft Arcon.State. Sie steuern bzw. informieren über die Anzeige von Hilfsmitteln und Details.

ACST_Label	Beschriftungen (nur im Konstruktionsmodus)
ACST_Grid	Das Raster
ACST_Cut	Schnitte
ACST_FilledWalls	Wandschraffuren
ACST_Dimensions	Vermaßungen
ACST_Labeling	Schriftfeld
ACST_FurnishingItems	Inventar/Objekte (nur im Konstruktionsmodus)
ACST_Guides	Hilfslinien (nur im Konstruktionsmodus)
ACST_Transparencies	Folien (nur im Konstruktionsmodus)
ACST_Origins	Ursprung (nur im Konstruktionsmodus)
ACST_Roof	Dach
ACST_RoofConstruction	Dachkonstruktion
ACST_Ruler	Lineal (nur im Konstruktionsmodus)
ACST_ArchitecturalSymbols	Architektengerechte Ersatzdarstellung (nur im Konstruktionsmodus)
ACST_Terrain	Gelände/Umgebung
ACST_North	Nordpfeil

Button Info

Diese Bit’ s werden bei der Definition von Schaltflächen verwendet.

In welcher Leiste soll die Schaltfläche erscheinen? Mindestens einer dieser Werte muß im Button-Info enthalten sein.

ACBI_UpperPannel	Die obere Knopfleiste (Design- und Konstruktionsmodus)
ACBI_LowerPannel	Die untere Knopfleiste (nur Designmodus)
ACBI_HowPannel	Die `Wie´-Leiste (nur Konstruktionsmodus)
ACBI_LeftPannel	Die linke Werkzeugleiste (Design- und Konstruktionsmodus)

Was für eine Art von Schaltfläche ist es?

ACBI_PushButton	Ein Taster (springt nach dem Drücken wieder heraus)
ACBI_CheckButton	Ein Schalter (verbleibt in der gedrückten oder ungedrückten Position)

In welchem Arcon-Modus ist der Knopf sichtbar? Mehrere Angaben sind möglich (Werte mit `oder´ bzw. `+´ verknüpfen)

ACBI_NoMode	Kein Projekt geladen
ACBI_ConstructionMode	Im Konstruktionsmodus
ACBI_DesignMode	Im Desingmodus
ACBI_AllModes	Immer

Soll vor oder hinter dem Knopf etwas Platz gelassen werden? Mehrere Angaben sind möglich.

ACBI_SeparatorBefore	Abstand vor dem Knopf
ACBI_SeparatorAfer	Abstand hinter dem Knopf

Wo soll der Knopf eingefügt werden?

ACBI_AtNext	Der Knopf wird an die nächste freie Position gesetzt
ACBI_AtStart	Der Knopf wird vor allen anderen eingefügt
ACBI_AtEnd	Der Knopf wird hinter allen anderen (evtl. rechtsbündig) angefügt

Bei How-Buttons: welche Ereignisse darf der Knopf auslösen?

AC_HowCustomStatusInfo	Die Statuszeile wird vom Client-Programm selbst gesteuert
AC_HowCustomSnap	Das Client-Programm steuert die Fang-Funktion selbst

Button Events

Mögliche Werte für den Parameter “ evnt” des ButtonChange-Ereignisses

ACBE_SelChanged	Bei Multi-Funktionsknöpfen: die ausgewählte Funktion wurde gewechselt
ACBE_LeftClicked	Der Knopf wurde mit der linken Maustaste betätigt
ACBE_RightClicked	Der Knopf wurde mit der rechten Maustaste angeklickt
ACBE_Deactivated	Das aktuelle Tool wurde deaktiviert (z.B. durch Wechsel auf ein anderes Eingabewerkzeug oder Umschalten in den Designmodus)

Display Modes

Mögliche Werte für ArCon.Mode.

AC_NoMode	Keine Darstellung, es ist kein Projekt geladen
AC_ModeConstruct	Das Projekt wird in 2D dargestellt
AC_ModeDesign	Das Projekt wird in 3D dargestellt

Story Types

Stockwerktypen für die Methode Building.CreateStory.

AC_StockwerkFundament	Das Fundament eines Gebäudes
AC_StockwerkGelaende	Das Gelände
AC_StockwerkKeller	Ein Kellergeschoß
AC_StockwerkErdgeschoss	Das Erdgeschoß
AC_StockwerkObergeschoss	Ein Obergeschoß
AC_StockwerkDachgeschoss	Ein Dachgeschoß
AC_StockwerkFreierTyp	Ein anderer Geschoßtyp mit benutzerdefiniertem Namen
AC_StockwerkBoden	Ein Dachboden
AC_StockwerkSpitzboden	Ein Spitzboden

Terrain Types

Mögliche Werte für die Eigenschaft Terrain.Type.

AC_TerrainGlobal	Die Welt außerhalb des Geländes. Dieses Terrain kann keine Hecke/ Umrandung haben
AC_TerrainEstate	Grundstück oder Teil eines Grundstückes, kann Gebäude enthalten
AC_TerrainRegion	Teil eines Geländes mit gleichen Materialeigenschaften

Typen von Vermaßungen

Die verschiedenen Varianten der Darstellung von Vermaßungen (siehe Dimension.Type).

ACDT_BemassungUnten	Zahl unterhalb der Maßstrecke
ACDT_BemassungOben	Zahl oberhalb der Maßstrecke
ACDT_BemassungMitte	Maßstrecke unterbrochen für Zahl
ACDT_BemassungKringelUnten	Maßstrecke mit runden Enden, Zahl unterhalb
ACDT_BemassungKringelOben	Maßstrecke mit runden Enden, Zahl oberhalb
ACDT_BemassungKringelMitte	Maßstrecke mit runden Enden, unterbrochen für die Zahl
ACDT_BemassungHoeheFertigkonstruktion	Höhe der Fertigkonstruktion vermaßen
ACDT_BemassungHoeheRohkonstrutkion	Höhe der Rohkonstruktion vermaßen

2D Drawing

Zeichnungsebenen, Grafikelemente werden unterhalb der benannten Objekte gezeichnet, also von diesen noch verdeckt. Alle Angaben beziehen sich lediglich auf den Konstruktionsmodus.

AC_LayerGrid	Unterhalb des Rasters
AC_LayerDesignobjects	Unterhalb der Einrichtungsgegenstände
AC_LayerFilm	Unterhalb der Folien
AC_LayerTerrain	Unterhalb des Geländes
AC_LayerFirewalls	Unterhalb von Rauchwänden (nur in Spezialversionen)
AC_LayerObjectcluster	Unterhalb von Cluster-Objekten (nur in Spezialversionen)
AC_LayerFloor	Unterhalb von (nicht-aktuellen) Stockwerken
AC_LayerGuide	Unterhalb von Hilfslinien
AC_LayerMeasurement	Unterhalb von Vermaßungen
AC_LayerLabeling	Unterhalb von Beschriftungen
AC_LayerRuler	Unterhalb des Lineals
AC_LayerOrigin	Unterhalb des Ursprungs
AC_LayerCompass	Unterhalb des Nordpfeiles
AC_LayerCuts	Unterhalb von Schnitten
AC_LayerSchriftfeld	Unterhalb des Schriftfeldes
AC_LayerLast	Über allem, unverdeckt

Stilangaben für den Rand eines 2D-Elementes

ACG2D_BORDERSTYLE_TRANSPARENT	Kein sichtbarer Rand
ACG2D_BORDERSTYLE_SOLID	Durchgezogener Rand
ACG2D_BORDERSTYLE_DASH	Gestrichelter Rand
ACG2D_BORDERSTYLE_DOT	Gepunkteter Rand
ACG2D_BORDERSTYLE_DASH_DOT	Rand aus abwechselnd Strich und Punkt
ACG2D_BORDERSTYLE_DASH_DOT_DOT	Rand aus jeweils einem Strich gefolgt von zwei Punkten
ACG2D_BORDERSTYLE_INSIDE_SOLID	Innen ausgefüllt

Stilangaben für das Füllen einer Figur

ACG2D_FILLSTYLE_SOLID	vollständig ausgefüllter Hintergrund
ACG2D_FILLSTYLE_TRANSPARENT	durchsichtiger Hintergrund
ACG2D_FILLSTYLE_HORIZONTAL_LINE	horizontal schraffierter Hintergrund
ACG2D_FILLSTYLE_VERTICAL_LINE	vertikal schraffiert Hintergrund
ACG2D_FILLSTYLE_UPWARD_DIAGONAL	links unten nach rechts oben schraffierter Hintergrund
ACG2D_FILLSTYLE_DOWNWARD_DIAGONAL	links oben nach rechts unten schraffierter Hintergrund
ACG2D_FILLSTYLE_CROSS	horizontal und vertikal schraffierter Hintergrund
ACG2D_FILLSTYLE_DIAGONAL_CROSS	in beide Richtungen diagonal schraffierter Hintergrund

Stilangaben für das Füllen der Fläche ausserhalb der Figur

ACG2D_BACKSTYLE_TRANSPARENT	Außerhalb der Figur ist der Hintergrund durchsichtig
ACG2D_BACKSTYLE_OPAQUE	Außerhalb der Figur wird der Hintergrund gefüllt

How-Button Input Events

Flags für den Parameter “ State” beim HowInput Ereignis. Es können Kombinationen aus diesen Bits auftreten.

ACHI_Extended	Der Benutzer hat erweiterte Einstellungen angefordert (Umschalten wurde gedrückt)
ACHI_NoDialog	Der Benutzer möchte keine weiteren Nachfragen (Strg wurde gedrückt)
ACHI_Mouse	Die Eingabe erfolgte mit der Maus
ACHI_Abort	Das aktuelle Tool wurde abgebrochen (per ESC-Taste.)

Dialog IDs

Jeder Dialog in ArCon hat eine eindeutige Kennzahl, die Dialog-ID. Die symbolischen Namen dieser ID’ s sind direkt aus den internen Quelltexten für ArCon erzeugt worden. Da Dialoge in ArCon von unterschiedlichen Programmierern erstellt wurden, ist die Namensgebung nicht bei allen Dialogen einheitlich.

Um den symbolischen Namen eines konkreten Dialoges zu ermitteln, benutzen Sie das mitgelieferte Hilfsprogramm Dialog-Explorer.

Dialog Inhalte

Innerhalb eines Dialoges haben alle Steuerelemente, die variable Daten aufnehmen können, eine eindeutige Kennzahl (ID), die auch einen symbolischen Namen hat. Diese ID und ihren symbolischen Namen können Sie am einfachsten mit dem mitgelieferten Hilfsprogramm Dialog-Explorer ermitteln. Sie können auch auf Standard-Entwicklungswerkzeuge wie z.B. Microsoft's "Spy++" zurückgreifen.

Menü IDs

ArCon+ benutzt für jeden Modus ein eigenes Menü. Die entsprechenden symbolischen Konstanten für jeden Menüeintrag setzen sich aus einem Präfix für den Modus sowie dem Menüpfad, getrennt durch “ _” - zusammen. Die Namen der Konstanten sind selbsterklärend.

Benutzerdefinierte Chunk-IDs

Kennzahlen für benutzerdefienierte Daten, die in ArCon-Projekten gespeichert werden.

Wenn Sie persistente Daten in ArCon Projekten speichern und Ihrer Applikation veröffentlicht werden soll, wenden Sie sich bitte an mb-Software, um eine eindeutige ID außerhalb dieses Bereiches zugewiesen zu bekommen.

Für lokale Erweiterungen und während des Teststadiums benutzen Sie bitte

eine ID aus disem Bereich.

AC_CHUNKID_MIN	minimale benutzerdefinierte Chunk-ID
AC_CHUNKID_MAX	maximale benutzerdefineirte Chunk-ID

Projekteinstellungen

Konstanten für Eigenschaften von Project.

Papierformate

ACPF_A4hoch	Papierformat A4 hochkant
ACPF_A4quer	Papierformat A4 quer
ACPF_A3hoch	Papierformat A3 hochkant
ACPF_A3quer	Papierformat A3 quer
ACPF_Drucker	Papierformat vom Drucker vorgegeben
ACPF_Frei	Papierformat manuell definiert

Einheiten

ACME_Millimeter	Einheit ist mm
ACME_Zentimeter	Einheit ist cm
ACME_Meter	Einheit ist m
ACME_Inch	Einheit ist Inch
ACME_Yard	Einheit ist Yard
ACME_Feet	Einheit ist Feet
ACME_Grad	Einheit ist Grad
ACME_Procent	Einheit ist Prozent

Flags für Räume

Bits für die Eigenschaft Room.Flags

ACRO_RAUMOPTS_AutoClosure	Der Raum wird automatisch geschlossen
ACRO_RAUMOPTS_NoCeiling	Der Dach wird nicht automatisch mit einer Decke versehen
ACRO_RAUMOPTS_NoFloor	Der Raum erhält nicht automatisch einen Boden
ACRO_RAUMTEXT_Bemerkung	Die Bemerkung zum Raum wird angezeigt
ACRO_RAUMTEXT_Flaeche	Die Fläche des Raumes wird angezeigt
ACRO_RAUMTEXT_FlaechenFormel	Die Formel für die Raumfläche wird angezeigt
ACRO_RAUMTEXT_Name	Der Name des Raumes wird angezeigt
ACRO_RAUMTEXT_Volumen	Das Raum-Volumen wird angezeigt
ACRO_RAUMTEXT_Wohnung	Der Name der Wohnung, zu der der Raum gehört wird angezeigt

Materialflags

Konstanten für Eigenschaften von Materialien (z.B. bei Terrain).

ACMATFL_ISTEXTURED	Textur wird benutzt
ACMATFL_ISTEXMASK	Textur ist eine Maskentextur
ACMATFL_ISLIGHT	Das Material ist eine Lichtquelle
ACMATFL_MIXTEXCOL	Textur und Farbe mischen
ACMATFL_TWOSIDED	Beide Seiten sind sichtbar
ACMATFL_ISTEXREFL	Textur ist eine Reflektionsmaske

Multi User Verhalten

Die folgenden Bits steuern das ArCon Multi-User Verhalten. Die Kombination (“ oder” bzw. “ +”) wird der Eigenschaft Arcon.MultiUserMode zugewiesen. Achtung: vorher den alten Wert merken und nachher unbedingt wieder herstellen!

ACMU_NICE	ArCon benutzt möglichst wenig CPU Zeit
ACMU_UIENABLED	Der Benutzer kann auch bei laufendem Makro mit ArCon arbeiten
ACMU_OTHERMACROS	Andere Makros können gleichzeitig ausgeführt werden
ACMP_DEFAULT	Standardeinstellung für den MultiUserMode

Im Normalfall sind alle diese Eigenschaften, die das Multi-User-Verhalten des Gesamtsystems und ArCon’ s verbessern, eingeschaltet. Sie können aus verschiedenen Gründen von Ihrem Makro temporär abgeschaltet werden. Im einzelnen erreichen Sie durch das Abschalten der Bits folgendes:

ACMU_NICE	Dieses Flag hat eine doppelte Funktion. Durch sein Ausschalten verwendet ArCon wesentlich mehr CPU-Zeit für die Kommunikation mit den gerade laufenden Makros. Die CPU-Auslastung eines Einprozessorsystems steigt hierbei sofort auf 100%. Zum anderen führt ArCon zeitaufwendige Arbeiten wie das Verschneiden von neu plazierten Wänden und die Bildschirmaktualisierung nicht mehr immer sofort durch, sondern erst, wenn es sich nicht mehr vermeiden läßt. Dies beschleunigt z.B. Importe aus anderen Dateiformaten oder komplizierte Konstruktionen - wie runde Räume - deutlich.
ACMU_UIENABLED	Das ArCon Hauptfenster wird gesperrt, es sind keine Benutzereingaben mehr möglich. ArCon führt (falls nicht durch andere Flags unterbunden) weiterhin alle nötigen Bildschirmaktualisierungen durch.
ACMU_OTHERMACROS	Andere Makros werden vorübergehend gestoppt und können nur in (technisch bedingten) Ausnahmefällen noch mit ArCon kommunizieren. Sie können keine Änderungen am ArCon Projekt vornehmen.

Wenn Ihr Makro sowohl ACMU_UIENABLED als auch ACMU_OTHERMACROS abschaltet, ist sichergestellt, daß sich das ArCon Projekt nur durch Aktionen Ihres Makros ändern kann. Dies ist zum einen sinnvoll, wenn Ihr Makro eine langwierige Konstruktion durchführt und dabei nicht durch andere Makros oder den Benutzer gestört werden soll, zum anderen z.B. beim Erstellen eines Schnappschusses des aktuellen Projektes durch ein AVA System, um die Konsistenz der ermittelten Daten sicherzustellen.

Max Type IDs

Maximale Typkennung für verschiedene ArCon Objekten.

AC_MaxWallType	Maximaler Wert für Wall.Type
AC_MaxChimneyType	Maximaler Wert für Chimney.Type
AC_MaxSupportType	Maximaler Wert für Support.Type
AC_MaxTerrainType	Maximaler Wert für Terrain.Type
AC_MaxHedgeType	Maximaler Wert für Hedge.Type

Menü Modi

Mögliche Flags für die Gültigkeit von Menüeinträgen, siehe auch das Beispiel zur Erstellung eigener Menüeinträg.

AC_MenuNoMode	Keine Darstellung, es ist kein Projekt geladen
AC_MenuModeConstruct	Das Projekt wird in 2D dargestellt
AC_MenuModeDesign	Das Projekt wird in 3D dargestellt

Ereignsikonstanten

Flags zum Anfordern von Ereignissen mit Hilfe der Funktionen NotifyOnChange bzw. zum Testen der aktuellen Ereignismaske im Ereignis ArCon.ChangeNotify. Mögliche Bitwerte für diese Masken sind:

AC_CHANGE_None	Kein Ereignis
AC_CHANGE_Unspecified	Nicht näher definiertes Ereignis
AC_CHANGE_Created	Ein Objekt wurde per UNDO neu erzeugt
AC_CHANGE_Destroyed	Ein Objekt wurde gelöscht
AC_CHANGE_Moved	Ein Objekt wurde verschoben
AC_CHANGE_ConturChanged	Eine Kontur hat sich verändert
AC_CHANGE_TextureChanged	Eine Textur wurde ausgetauscht
AC_CHANGE_UndoDestroy	Ein Objekt wurde vorläufig gelöscht (und ist per UNDO wieder herstellbar)
AC_CHANGE_MaterialChanged	Ein Material wurde verändert

Nenben diesen Instanzbezogenen Ereignissen (zum Anmelden benötigen Sie ein konkretes Objekt) gibt es typbezogene Ereignisse, die Sie mit Hilfe der Funktion ArCon. ChangeTypeNotifyMask aktivieren. Mögliche Bitwerte dafür sind:

AC_CHANGE_Inserted	Ein neues Objekt wurde erzeugt

Externe Ansichten

Bitwerte zur Kodierung der Menge exteren Ansichten, die gerade angezeigt werden. Siehe z.B. ArCon.SetExternalViews oder ExternalViewsVisibilityChanged.

Die Bitmasken sind:

EXT_VIEWS_EXPLORER	Der Datei- bzw. Datenbankexplorer wird angezeigt
EXT_VIEWS_STORYBOARD	Das Storyboard wird angezeigt

Erweiterte Designobjektflags

Flags zur Steuerung vonn 3D Objekteigenschaften in der ArCon Welt. Sie können mit Object3D.GetExtendedFlags ermittelt und mit Object3D.SetExtendedFlags verändert werden.

Die Bitmasken sind:

ACGPL_LampenBeiNachtAn	In der Nachtansicht (bzw. Nachts bei Zeitabhängiger Sicht) sind die Lampen dieses Objektes an.
ACGPL_LampenSindAn	Die Lampen dieses Objektes sind (unabhängig von der Zeit/Ansicht) an.
ACGPL_DarfVerzerren	Das Objekt kann nicht-linear skaliert werden
ACGPL_SollFallen	Das Objekt fällt beim Einfügen
ACGPL_LokalAnTerrainAnpassen	Das Objekt fällt auf Terrainhöhe (nicht auf Gebäudehöhe)
ACGPL_KeineSchatten	Das Objekt wirft keine Schatten
ACGPL_NichtMitGebaeudeBewegen	Das Objekt wird nicht mit dem Gebäude verschoben/ gedreht
ACGPL_NichtPassivSnappen	An dem Objekt kann nicht gefangen werden
ACGPL_NichtAktivSnappen	Das Objekt fängt nicht
ACGPL_NichtLoeschbar	Das Objekt ist nicht (für den Anwender) löschbar
ACGPL_GruppeNichtAufloesbar	Die Gruppierung kann nicht aufgehoben werden
ACGPL_BreiteNichtSkalierbar	Das Objekt ist in der Breite nicht skalierbar
ACGPL_TiefeNichtSkalierbar	Das Objekt ist in der Tiefe nicht skalierbar
ACGPL_HoeheNichtSkalierbar	Das Objekt ist in der Höhe nicht skalierbar
ACGPL_NichtSelektierbar	Das Objekt kann nicht ausgewählt (und damit doppeltgeklickt) werden

3D Objekt-Ereignisse

Bitwerte zur Kodierung von Ereignissen, die mit Hilfe der Funktion ArCon.SetObject3DeventMask angefordert werden können

Die Werte sind:

ACO3D_EVENT_DBLCLK	Das Objekt wird doppetgeklickt (WorldObject3DdoubleClicked)
ACO3D_EVENT_MOVED	Das Objekt wurde bewegt (WorldObject3Dmoved)
ACO3D_EVENT_TEXTURE_DROPPED	Das Objekt erhält eine neue Textur (WorldObject3DtextureDropped)
ACO3D_EVENT_MATERIAL_DROPPED	Das Objekt erhält ein neues Material (WorldObject3DmaterialDropped)

Typkodierungen

Werte zur Kodierung des Datentyps für Ereignissen, die mit Hilfe der Funktion ArCon. ChangeTypeNotifyMask angefordert werden können

Die Werte sind:

AC_OBJTYPE_INVALID	ungültiger Objekttyp/kein Objekt
AC_OBJTYPE_ConnHandle	generischer Objekttyp
AC_OBJTYPE_Building	Gebäude
AC_OBJTYPE_Story	Stockwerk
AC_OBJTYPE_Room	Raum
AC_OBJTYPE_Wall	Wand
AC_OBJTYPE_Roof	Dach
AC_OBJTYPE_Window	Fenster
AC_OBJTYPE_Door	Tür
AC_OBJTYPE_StairCase	Treppe
AC_OBJTYPE_Graphics2D	2D Graphikelement
AC_OBJTYPE_Contur	Raumkontur
AC_OBJTYPE_WallSegment	Wandsegment
AC_OBJTYPE_Support	Stütze
AC_OBJTYPE_Chimney	Schornstein
AC_OBJTYPE_Terrain	Gelände, Bereich oder Grundstück
AC_OBJTYPE_CeilingOpening	Deckenöffnung
AC_OBJTYPE_Ceiling	Deckenplatte
AC_OBJTYPE_RoofWindow	Dachfenster
AC_OBJTYPE_Guide	Hilfslinie
AC_OBJTYPE_Labeling	Beschriftung
AC_OBJTYPE_Dimension	Bemaßung
AC_OBJTYPE_Hole	Wandloch
AC_OBJTYPE_UnterUeberzug	Unter/Überzug
AC_OBJTYPE_ObjectConstructor	3D Objekt Klasse
AC_OBJTYPE_Object3D	3D Objekt Instanz
AC_OBJTYPE_Cut	Schnitt
AC_OBJTYPE_CutView	Schnittansicht
AC_OBJTYPE_VirtualWall	virtuelle Wand
AC_OBJTYPE_Gaube	Dachgaube
AC_OBJTYPE_RoofArea	Dachfläche
AC_OBJTYPE_Schriftfeld	
AC_OBJTYPE_Podest	
AC_OBJTYPE_Kamera	
AC_OBJTYPE_View	Ansicht
AC_OBJTYPE_SENTINEL	letzter gültiger Typcode

Dynamisch geladene Konstanten in C++

Das folgende Beispiel zeigt Ihnen, wie Sie in Visual C++ ArCon Konstanten dynamisch aus der Typbibliothek des Active-X Controls ermitteln können. Dieses Verfahren macht Ihre Anwendungen von Änderungen einiger Konstanten (z.B. die Anzahl der verfügbaren Wandtypen) in künftigen ArCon Versionen unabhängig.

Nehmen Sie an, Sie haben eine dialogbasierte Anwendung erstellt und im Hauptdialog ein ArCon Active-X Control eingesetzt, das an die C++ Member-Variable “ m_arcon” Ihrer Dialogklasse gebunden ist. Ferner wissen Sie, welche der Konstanten aus der ArCon Typinformation Sie benötigen. In der “ OnInitDialog” Methode aktivieren Sie die Verbindung zu ArCon und ermitteln anschließend die benötigten Konstanten, indem Sie z.B. folgenden Code verwenden:

```
// Aktiviere die Verbindung zu ArCon. Wir benutzen die Standard-
  // Skalierung (Meter) und den Standard-Namen für die Hilfedatei.
  // Die Verbindung muß nicht explizit abgebaut werden, da m_arcon
  // eine member variable der Dialogklasse ist und daher beim Zerstören
  // des Dialogs auch m_arcon zerstört wird. Dies impliziert einen
  // Verbindungsabbau.
  if (!m_arcon.StartMe((long)m_hWnd, 1.0f, AfxGetApp()->m_pszHelpFilePath)) {
    AfxMessageBox("Kann ArCon nicht starten");
    PostQuitMessage(1);
    return TRUE;
  }


// Ermittle die Laufzeitkonstanten. GetControlUnknown() liefert
  // einen IUnknown Zeiger, das ist OLE's Gegenstück zu void*
  // bzw. CObject*. Es könnte auch der IUnknown Zeiger oder ein
  // LPDISPATCH eines beliebigen anderen Objektes sein, daß von
  // ArCon.OCX erzeugt wurde.
  if (!GetArConConstants(m_arcon.GetControlUnknown())) {
    AfxMessageBox("Kann ArCon Konstanten nicht ermitteln");
    PostQuitMessage(1);
    return TRUE;
  }
```

Hier ist lediglich der Aufruf der Funktion “ GetArConConstants” hinzugekommen, den übrigen Code haben Sie sicher selbst schon vielfach so oder ähnlich verwendet.

Die Funktion “ GetArConConstants” sieht nun so aus:

```
// ---------------------------------------------------------------------
// Schlage den Wert einiger Konstanten in der Type-Library des ArCon.OCX
// Controls nach. Auf diese Weise gewonnene Konstanten machen die EXE-Datei
// unabhängig von der ArCon-Version. Allerdings ist dieser Weg wesentlich
// umständlicher als die Benutzung der "acGlobal.h" per #include.


// ---------------------------------------------------------------------
// Variable, um die Konstanten aufzunehmen
unsigned long constAC_MaxChimneyType, constAC_MaxSupportType,
constAC_MaxWallType;


// ---------------------------------------------------------------------
// Eine Liste der gesuchten Konstanten und der zugehörigen Variablen
static struct { unsigned long *var; OLECHAR *name; } constList[] =
{
  // Adresse der VariableUNICODE Name der Konstante
  { &constAC_MaxChimneyType,  L"AC_MaxChimneyType"  },
  { &constAC_MaxSupportType,  L"AC_MaxSupportType"  },
  { &constAC_MaxWallType,L"AC_MaxWallType"},
};


// ---------------------------------------------------------------------
// Hilfsfunktion, um alle gewünschten Konstanten aus der Type-Library zu
// extrahieren. Bekommt den IUnknown-Zeiger eines beliebigen Objektes aus
// dem ArCon.OCX Control übergeben.
static BOOL GetArConConstants(LPUNKNOWN iUnkwn)
{
```

```
  LPDISPATCH iDisp;
  ITypeInfo* tInfo;
  ITypeLib *tLib;
  ITypeInfo *resTypeInfo;
  MEMBERID mid;
  unsigned int index;
  int i,
  num = sizeof constList / sizeof constList[0];  // Anzahl der gesuchten
Konstanten


  // Wir benötigen ein Automatisierungsobjekt (IDispatch), frage das übergebene
  // IUnknown danach
  if (FAILED(iUnkwn->QueryInterface(IID_IDispatch, (void**)&iDisp))) return
FALSE;
  // Hole die Typinformation zu diesem Objekt
  if (FAILED(iDisp->GetTypeInfo(0, GetUserDefaultLCID(), &tInfo))) return FALSE;
  // Aber diese konkrete Information interessiert uns gar nicht, wir wollen
  // an Daten von anderen Typen aus der gleichen Typelibrary
  if (FAILED(tInfo->GetContainingTypeLib(&tLib, &index))) {
    iDisp->Release();
    return FALSE;
  }


  // tLib zeigt nun auf die ArCon Typelibrary. Für alle gesuchten
  // Konstanten wiederhole:
  for (i = 0; i < num; i++) {
    unsigned short count = 1;


   // Suche das erste Auftreten dieses Namens. ArCon verwendet nur global
    // eindeutige Konstanten, es kann also nicht mehrere Typen mit diesem
    // Namen geben.
    if (SUCCEEDED(tLib->FindName(constList[i].name, 0, &resTypeInfo, &mid,
&count))
   && count == 1) {


 TYPEATTR *attr;
 VARDESC *var;
 int j;
 BOOL found;


 // Der (jetzt gefundene) Name ist Teil eines Typs (ein "enum"). Wir
 // ermitteln, wieviele Konstanten in dieser Enum enthalten sind und
 // suchen dann alle nach der MEMBERID mid unseres gesuchten Namens
 // ab
 if (SUCCEEDED(resTypeInfo->GetTypeAttr(&attr))) {
   for (found = FALSE, j = 0; !found && j < attr->cVars; j++) {
if (SUCCEEDED(resTypeInfo->GetVarDesc(j, &var))) {
  if (var->memid == mid) {
    // Gefunden - dies ist die gesuchte Konstante. Wert kopieren...
    *(constList[i].var) = var->lpvarValue->lVal;
    // und Schleife abbrechen
    found = TRUE;
  }
  resTypeInfo->ReleaseVarDesc(var);
}
   }
   resTypeInfo->ReleaseTypeAttr(attr);
 }
 resTypeInfo->Release();
    }
  }

  tLib->Release();
  iDisp->Release();

  return TRUE;
```

```
}
// ------------------------------------------------------------------------
```

Wenn Sie bisher nicht mit OLE-Interfaces programmiert haben (bzw. dem Class Wizard diese Aufgabe überlassen haben), können Sie diesen Code als “ BlackBox” betrachten. Um weitere/ andere Konstanten zu ermitteln, müssen Sie lediglich die Liste mit Adressen von Variablen und Namen der Konstanten ändern/ergänzen. Die Variablennamen werden als UNICODE Zeichen (also pro Zeichen mit 16bit) abgelegt, daher die etwas ungewohnte Notation mit vorangestelltem “ L” .

DIALOG-EXPLORER

Wenn Ihr Makro die ArCon Benutzeroberfläche erweitert und dazu Daten aus ArCon Dialogen ermittelt oder verändert, benötigen Sie die Kennzahlen für den betroffenen Dialog und die Dialogelemente. Statt einer umfangreichen und schwer verständlichen statischen Dokumentation wurde ein interaktives Werkzeug erstellt, das Ihnen das Ermitteln dieser Daten erleichtert: der Dialog-Explorer.

Es handelt sich dabei um ein ArCon Makro, das bei jedem sich öffnenden ArCon Dialog eine detaillierte Auflistung der Daten dieses Dialoges anzeigt.

TRANSFORMATIONSMATRITZEN

In der Computergraphik werden häufig 3D Koordinaten mit Hilfe von Matritzenmultiplikation in andere Koordinaten überführt. Diese Operation verwenden Matritzen in homogenen Koordinaten. Für einen 3D Vektor benötigt man eine 4x4 Matrix, um alle Operationen in einer Matrix ausdrücken zu können.

Wählen Sie einen dieser Verweise, um nähere Details zu erfahren: Deklaration und Benutzung der Matritzen in C/C++ , Deklaration in Visual Basic, Grundlagen zu homogenen Koordinaten und 4x4 Matritzen

Deklaration und Benutzung von 4x4 Matritzen in C/C++

Deklaration und Benutzung von 4x4 Matritzen in Visual Basic

Grundlagen zu homogenen Koordinaten und 4x4 Matritzen

Das Standard-Werk zum Nachschlagen weiterer Details zu diesem Thema ist:

Grundlagen der Computergraphik
Einführung, Konzepte, Methoden
James D. Foley, Andries van Dam, Steven K. Feiner u. a.
1994. XXV, 628 S. m. zahlr. Abb., 44 farb. Abb.

Addison-Wesley, Bonn; ISBN: 3893196471

HILFSFUNKTIONEN IN MAKROUTIL.DLL

Bei der Benutzung der ArCon Programmierschnittstelle gibt es wiederkehrende Aufgaben, die nicht in der Schnittstelle selbst erledigt werden können (z.B. weil Sie im Prozeß Ihres Makros ablaufen müssen). Für solche Aufgaben stellen wir Hilfsfunktionen in der Programmbibliothek “ MakroUtil.DLL” zur Verfügung, die im Makro-Verzeichnis einer ArCon Installation zu finden ist. Dadurch können alle Makros ohne zusätzlichen Installationsaufwand auf diese Bibliothek zugreifen.

Sie finden in den mitgelieferten Beispielmakros Details über die Einbindung dieser Bibliothek in verschiedene Programmierumgebungen.

Funktion ArConPictureFromBitmap

Konvertiert eine Bitmap (übergeben als Bitmap-Handle HBITMAP) in ein VARIANT, daß über Prozeß- und Rechnergrenzen hinweg an ArCon weitergereicht werden kann.

Die Deklaration in C lautet:

```
BOOL WINAPI ArConPictureFromBitmap(
  HBITMAP hBmp,/* Handle des an ArCon zu übergebenden Bitmaps */
  VARIANT * var);   /* VARIANT, das das Bitmap in der ArCon internen
   Kodierung aufnimmt */
```

In Basic wird diese Funktion folgendermaßen deklariert:

```
Public Declare Function ArConPictureFromBitmap Lib "MakroUtil" _
    (ByVal hBitmap As Long, ByRef result As Variant) As Long
```

Der Aufrufer muß die übergebene VARIANT Struktur nach der Verwendung wieder löschen, falls dies in der jeweiligen Entwicklungsumgebung nötig ist (in C z.B. mit “ VariantClear”).

Funktion ArConUtilityVersion

Liefert die interne Version der Bibliothek “ MakroUtil.DLL” . Da Makros die in der jeweiligen ArCon Version vorgefundene Version dieser Bibliothek nutzen (und nicht eine eigene installieren dürfen) , muß eventuell bei zukünftigen Änderungen dieser Bibliothek versionsabhängiger Code in Makros für die Verwendung unterschiedlicher Bibliotheken geschrieben werden.

Die Deklaration in C lautet:

```
BOOL WINAPI ArConUtilityVersion(
  DWORD * versMajor,    /* Hauptversion der DLL, derzeit 0 */
  DWORD * versMinor);   /* Unterversion der DLL, derzeit 1 */
```

Die Basicdeklaration dieser Funktion lautet:

```
Public Declare Function ArConUtilityVersion Lib "MakroUtil" _
    (ByRef versMajor As Long, ByRef versMinor As Long) As Long
```

Diese Funktion erlaubt Ihnen eine Versionskontrolle, da nicht alle Funktionen der MakroUtil.DLL in allen installierten Versionen vorhanden sind. Wenn Sie Funktionen der DLL dynamisch anbinden (mit der Win32 API Funktion GetProcAddress) können Sie genauso die Existenz eines Einsprungspunktes als Versionierungsmerkmal benutzen.

Funktion BitmapFromArConPicture

Wenn ArCon Ihnen ein Bild liefert, geschieht dies in Form eines “ VARIANT” . Dieser enthält ein OLE SafeArray mit den Bildinformationen. Um dieses interne Format in ein Bitmap-Handle umzuwandeln, benutzen Sie diese Funktion. Sie sind für das Bitmap-Handle verantwortlich, müssen es also selbst löschen, wenn Sie es nicht mehr benötigen.

C Deklaration:

```
HBITMAP WINAPI BitmapFromArConPicture(
  const VARIANT * var);    /* VARIANT, das das Bitmap in der ArCon
internen Kodierung enthält */
```

Basic Deklaration:

```
Public Declare Function BitmapFromArConPicture Lib "MakroUtil" _
    (ByRef picture As Variant) As Long
```

Verfügbar ab Version 0.1

Funktion PictureFromArConPicture

Diese Funktion liefert die gleiche Funktionalität wie “ BitmapFromArConPicture” . In einigen Programmierumgebungen ist es aber einfacher, statt eines Bitmap-Handles ein OLE Picture Objekt zu bekommen. In Visual Basic können Sie den Rückgabewert dieser Funktion z.B. direkt der Picture-Eigenschaft einer PictureBox zuweisen.

C Deklaration:

```
IPicture * WINAPI PictureFromArConPicture(
  const VARIANT * var);    /* VARIANT, das das Bitmap in der ArCon
internen Kodierung enthält */
```

Basic Deklaration:

```
Private Declare Function PictureFromArConPicture Lib "MakroUtil"_
   (ByRef picture As Variant) As StdPicture
```

Verfügbar ab Version 0.1

DIE BEISPIELE

In der ArCon-SPU finden Sie verschiedene Beispielprogramme, die Ihnen die ArCon Makro-Programmierung demonstrieren sollen.

Badewanne

Das Badewannen-Beispiel zeigt die punktgenaue Positionierung eines geladenen 3D-Objektes.