2D

COLLABORATORS			
	TITLE : 2D		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		January 23, 2025	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

1 2D **1**

1.1 2D Drawing library . . . 1
1.2 backcolour . . . 2
1.3 boxfill . . . 2
1.4 circle . . . 2
1.5 cls . . . 2
1.6 copybitmap . . . 2
1.7 cursx . . . 3
1.8 cursy . . . 3
1.9 drawingfont . . . 3
1.10 drawingmode . . . 3
1.11 drawingoutput . . . 4
1.12 drawingrastport . . . 4
1.13 ellipse . . . 4
1.14 frontcolour . . . 5
1.15 line . . . 5
1.16 locate . . . 5
1.17 obtainbestpen . . . 5
1.18 plot . . . 6
1.19 point . . . 6
1.20 printtext . . . 6
1.21 releasepen . . . 6
1.22 textlength . . . 6
1.23 textstyle . . . 7

Chapter 1

2D

1.1 2D Drawing library

```
Pure Basic - 2D Drawing library V1.00

  Das 2D Zeichnen beinhaltet alle 2D Zeichenfunktionen, die Sie
  in einem sichtbaren Bereich ausführen können. Das Zeichnen einer
  Linie, einer Box, eines Kreies oder einfach eines Textes gehört
  zu den 2D Zeichenoperationen. Sie müssen einfach das Ausgabeziel
  (Bildschirm, Fenster, Bitmap..) angeben und können anfangen zu
  zeichnen...

Befehlübersicht:

   BackColour
   BoxFill
   Circle
   Cls
   CopyBitmap
   CursX
   CursY
   DrawingFont
   DrawingMode
   DrawingOutput
   DrawingRastPort
   Ellipse
   FrontColour
   Line
   Locate
   ObtainBestPen
   Plot
   Point
   PrintText
   ReleasePen
   TextLength
   TextStyle

Beispiel

   2D Drawing demo
```

1.2 backcolour

```
  SYNTAX
BackColour(Colour)

  STATEMENT
Definiert die standardmäßige Hintergrundfarbe für alle Grafikfunktionen
und die Textanzeige.
```

1.3 boxfill

```
  SYNTAX
BoxFill(x1, y1, x2, y2)

  STATEMENT
Zeichnet einen ausgefüllten Kasten in der aktiven Farbe (festgelegt
mittels FrontColour()) in das aktuelle Fenster.
```

1.4 circle

```
  SYNTAX
Circle(x, y, Radius)

  STATEMENT
Zeichnet den Umriß eines Kreises an der Position x,y mit der Größe
des angegebenen Radius.
```

1.5 cls

```
  SYNTAX
Cls(Colour)

  STATEMENT
Löscht das gesamte Ausgabeziel in angegebenen Farbe.
```

1.6 copybitmap

```
  SYNTAX
CopyBitMap(BitMapID, SourceX, SourceY, DestX, DestY, Width, Height)

  STATEMENT
Kopiert den angegebenen Bereich - SourceX, SourceY beschreiben die
linke obere Ecke; Width, Height die Breite und Höhe des Bereichs -
aus der BitMapID auf das Auagabeziel (Output). Mit DestX und DestY
können Sie die x und y Position, an der der kopierte Bereich
eingefügt werden soll, angeben.
```

```
Diese Funktion ist 100% OS-freundlich und beinhaltet automatisches
Clipping (Abschneiden der über den Fensterrand hinausstehenden
BitMap-Teile).
```

1.7 cursx

```
  SYNTAX
Result.w = CursX

  FUNCTION
Gibt die X-Position des Text Cursors im aktuellen Ausgabeziel
zurück.
```

1.8 cursy

```
  SYNTAX
Result.w = CursY

  FUNCTION
Gibt die Y-Position des Text Cursors im aktuellen Ausgabeziel
zurück.
```

1.9 drawingfont

```
  SYNTAX
DrawingFont(FontID)

  STATEMENT
Ändert den aktuellen Zeichensatz auf die angegebene FontID. Alle
neuen Texte werden mit dem neu eingestellten Zeichensatz dargestellt.

FontID MUSS ein gültiger IntuiFont Zeiger sein. Sie können die
NFont Library benutzen, um irgendeinen Font zu laden und den FontID()
Zeiger an diese Funktion zu übergeben.
```

1.10 drawingmode

```
  SYNTAX
DrawingMode(Mode)

  STATEMENT
Ändert den Zeichenmodus für Grafikausgaben:

  Hier folgt eine kurze Liste gültiger Modi:

    #JAM1       = 0 : für Textausgabe, läßt den Hintergrund transparent
    #JAM2       = 1 : für Textausgabe, gibt Text in der Hintergrundfarbe
```

```
                    aus
    #COMPLEMENT = 2 : Grafikausgabe im XOR Modus
    #INVERSVID  = 4 : Benutzung in Verbindung mit JAM2, es invertiert
                      Hintergrund- und Vordergrundfarben bei der
                      Textausgabe

Hinweis: Diese Konstanten sind in der AmigaOS Resident Datei enthalten.
```

1.11 drawingoutput

```
  SYNTAX
DrawingOutput(RastPort)

  STATEMENT
Stellt die Zeichenausgabe (Output - Ausgabeziel) auf den angegebenen
RastPort ein. Nach dieser Einstellung werden alle Zeichenfunktionen
auf diesem RastPort ausgeführt. Sie müssen einen gültigen RastPort
Zeiger angeben.

Sie können folgende Funktionen benutzen, um einfach RastPorts von
Objekten zu erhalten:

    + WindowRastPort()
    + ScreenRastPort()
    + BitmapRastPort()

Beispiel:

  DrawingRastPort WindowRastPort()  ; alle Zeichenausgaben finden im
                                    ; Fenster statt
```

1.12 drawingrastport

```
  SYNTAX
*RastPort = DrawingRastPort()

  STATEMENT
Gibt den Wert des aktuell benutzten Ausgabe-RastPorts zurück
(für fortgeschrittene Programmierer).
```

1.13 ellipse

```
  SYNTAX
Ellipse(x, y, RadiusX, RadiusY)

  STATEMENT
Zeichnet den Umriß einer Ellipse an der Position x,y mit den beiden
Radien RadiusX und RadiusY.
```

1.14 frontcolour

```
  SYNTAX
FrontColour(Colour)

  STATEMENT
Definiert die standardmäßige Zeichenfarbe für alle Grafikfunktionen
und die Textanzeige.
```

1.15 line

```
  SYNTAX
Line(x1, y1, x2, y2)

  STATEMENT
Zeichnet eine Linie in der aktiven Farbe (festgelegt mit FrontColour())
im aktuellen Ausgabeziel.
```

1.16 locate

```
  SYNTAX
Locate(x,y)

  FUNCTION
Positioniert den Textcursor an der angegebenen Position
(für DPrint()).
```

1.17 obtainbestpen

```
  SYNTAX
Result.w = ObtainBestPen(r, g, b, precision)

  STATEMENT
Ermittelt die Farbnummer, die am besten zu den angegebenen Parametern
passt. Auf einem öffentlichen Bildschirm (public screen) mit freien
Farben wird eine neue Farbnummer mit den (r,g,b) Werten reserviert.
Der 'precision' (Präzision) Parameter gibt an, wie die Funktion ihre
Suche durchführen soll.

  Gültige Werte von 'precision' (gefolgt von der Systemkonstante):
    Exact = -1 (#PRECISION_EXACT)
    Image =  0 (#PRECISION_IMAGE)
    Icon  = 16 (#PRECISION_ICON)
    Gui   = 32 (#PRECISION_GUI)

Hinweis: Sie MÜSSEN für alle reservierten Farben die Funktion
ReleasePen(Result) benutzen, bevor Sie Ihr Programm beenden.
Andernfalls werden diese Farben nie freigegeben und damit dem
Verzeichnis der freien Farben des Bildschirms zugeordnet (sehr
schlecht).
```

1.18 plot

```
  SYNTAX
Plot(x,y)

  STATEMENT
Zeichnet einen Punkt in der aktiven Farbe (festgelegt mittels
FrontColour()) in das aktuelle Fenster.
```

1.19 point

```
  SYNTAX
Colour.w = Point(x, y)

  STATEMENT
Gibt die Farbnummer an den Koordinaten (x,y) des aktuellen
Ausgabeziels zurück.
```

1.20 printtext

```
  SYNTAX
PrintText(String$)

  STATEMENT
Zeigt den angegebenen String auf dem eingestellten Ausgabeziel an der
mit Locate() festgelegten Position und in der mit FrontColour()
definierten Farbe an. Sie können TextStyle() benutzen, um den Ausgabe-
Modus (fett, kursiv, unterstrichen) zu wechseln und DrawingMode(),
um die Zeichenart des Textes zu ändern.
```

1.21 releasepen

```
  SYNTAX
ReleasePen(#Pen)

  STATEMENT
Teilt dem System mit, dass dieser Stift (Farbe) nicht mehr von
Ihrem Programm benutzt (locked) wird. Damit können andere Programme
diese Farbe wieder nutzen. Sie müssen diese Funktion aufrufen, wenn
Sie mit ObtainBestPen() Farben reserviert haben.
```

1.22 textlength

```
  SYNTAX
Length.w = TextLength(x, y)

  STATEMENT
```

Ermittelt die Länge des angegebenen String in Pixeln bezogen auf
das aktuelle Ausgabeziel. Der Hauptvorteil dieser Funktion liegt
darin, dass Sie die tatsächliche Länge von beliebigen Strings
in einem beliebigen Zeichensatz (auch nicht-proportionale)
ermitteln können.

1.23 textstyle

SYNTAX
TextStyle(Style)

STATEMENT
Legt den Stil für zukünftige Textausgaben fest.

Die erlaubten Stil-Werte sind:

0 = keiner
1 = unterstrichen
2 = fett
4 = kursiv

Sie können diese Werte bei Bedarf mischen (fett+kursiv ist 2+4 = 6).