License Agreement

This is a legal agreement between you, the end user, and ProtoView Development Corporation. By opening this envelope you are agreeing to accept ownership of this product and to be bound by the terms of this agreement. It also contains proprietary source code which you are entitled to use only under the terms of this license agreement. If, after reading this agreement, you do not agree with its terms, return the software, manuals and diskettes within thirty days of purchase to the party from whom you received it for a refund. In order to receive a refund, the disk and manuals must be in resellable condition. IF YOU PURCHASE THE SOURCE THE SALE IS FINAL.

ProtoView Software License For InterAct

1. Grant of License. ProtoView Development Corp. grants a limited, non-exclusive license to use one copy of the application development tool called InterAct for the purpose of developing applications for the Windows environment. This copy of ProtoView must exist on a single terminal connected to a single computer (i.e., with a single CPU). You may not network InterAct or otherwise use it for development on more than one computer at the same time.

2. Runtime Distribution License. ProtoView Development Corp. grants you a royalty-free right to distribute copies of the runtime dynamic link libraries for use with applications you have developed using InterAct. InterAct runtime files are listed in exhibit A. These libraries may not be distributed for any other purpose than to accompany software that you have developed using InterAct. You may use InterAct in your specific purpose application programs, in which case ProtoView grants you permission under ProtoView's copyright to use, give away or sell such programs without additional licenses or fees, as long as all copies of these programs bear a valid copyright notice and provided that your program is not merely a set or subset of the or a compilation or development tool or library which includes all or a portion of the , or is otherwise a product that is generally competitive with or a substitute for InterAct. This permission is granted solely for the purpose set forth above, and you are not authorized to use InterAct in any other manner. You may not use InterAct to create a development tool for the AS/400.

3. Copyright. InterAct is owned by ProtoView Development Corporation and is protected by United States copyright laws and international treaty provisions. You may

make one copy of the software for backup purposes. You may not, under any circumstances, copy the manual and other written materials that accompany this software.

4. Other Restrictions. You may not rent or lease InterAct software, but you may transfer the software and accompanying written materials on a permanent basis provided you retain no copies and the recipient agrees to the terms of this Agreement. You may not reverse engineer, decompile, or disassemble this software.

Limited Warranty

ProtoView Development Corp. warrants that the software herein will perform substantially in accordance with the accompanying written materials for a period of thirty days from the date of receipt. Any implied warranties on ProtoView are limited to ninety days. Some states do not allow limitations on duration of an implied warranty, so the above limitation may not apply to you.

Customer Remedies. ProtoView's entire liability and your exclusive remedy shall be, at ProtoView's option, either (a) return of the price paid or (b) replacement of the software that does not meet ProtoView's Limited Warranty and which is returned to ProtoView

with a copy of your receipt. This Limited Warranty is void if failure of the software has resulted from accident, abuse, or misapplication. Any replacement of the software will be warranted for the remainder of the original warranty period or 30 days, whichever is longer.

NO OTHER WARRANTIES. PROTOVIEW DEVELOPMENT DISCLAIMS ALL OTHER WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WITH RESPECT TO THE SOFTWARE AND THE ACCOMPANYING WRITTEN MATERIALS. THIS LIMITED WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS. YOU MAY HAVE OTHERS, WHICH VARY FROM STATE TO STATE.

NO LIABILITY FOR CONSEQUENTIAL DAMAGES. IN NO EVENT SHALL PROTOVIEW OR ITS SUPPLIERS BE LIABLE FOR ANY DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR INABILITY TO USE THIS PROTVIEW PRODUCT, EVEN IF PROTOVIEW HAS BEEN ADVISED OF THE

POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

Exhibit A

PVIDO.DLL - The 16-bit Control Module

PVIDO32.DLL - The 32-bit Control Module

PVIDO.OCX - The 16-bit OLE Control Module

PVIDO32.OCX - The 32-bit OLE Control Module

PGHT.DLL - The 16-bit Support Module

PGHT32.DLL - The 32-bit Support Module

What is InterAct?

Welcome to ProtoView InterAct, the Interactive Diagramming Object - or IDO for short. InterAct is an advanced custom control available as a DLL and ActiveX (OCX). It can be used in dialog editors, such as Microsofts AppStudio, Borlands Resource Workshop, ProtoViews ViewPaint or Microsofts Visual Basic. It can also be created at runtime in C, C++ or Basic programming languages.

InterAct is designed as a tool for displaying data to the user in a graphical format - as a diagram. In addition, it is interactive, meaning that the user can use the mouse and keyboard to modify the diagram. Objects (called entities) can be moved, resized, or have their appearance changed, and lines between objects (called relations) can be added or deleted. The application in which you embed InterAct is alerted to every user action in the diagram, allowing the application to respond to these events. In addition, a full API is exported from InterAct which allows an application to customize it to suit the applications needs.

As additional support, InterAct supplies a complement of built-in dialogs and menus it can display to allow the user to manipulate a diagram. InterAct also provides support

for saving or reading diagrams to disk, cutting and pasting information to the Windows clipboard, and printing diagrams. With this support it is possible to plug InterAct into an existing application and with little effort, have a full-fledged diagramming object. The programmer can then concentrate on the details of integrating the diagram with internal data structures.

The Interactive Diagramming Object

Installing InterAct

Before installing InterAct, be sure to read the license agreement included in the product package or displayed during the installation process.

Perform the following steps to install InterAct:

Floppy disk Install

1. Make sure Windows, Windows NT, or Windows 95 is running.
2. Insert the disk labeled InterAct Disk 1 into drive A or B.
3. For WFWG 3.1 or higher, or Windows NT, select File | Run from the Program Managers menu. In the Command Line edit box of the Run dialog, enter A:\SETUP or B:\SETUP depending on where you inserted the setup disk. Click OK.
4. For Windows95, select Start | Run from the task window. In the Open edit box of the Run dialog, enter A:\SETUP or B:\SETUP depending on where you inserted the setup disk. Click OK.
5. Follow the instructions given in the installation program.

CD-ROM Install

Make sure Windows, Windows NT, or Windows 95 is running.

Insert the CD labeled ProtoView into drive D or whichever drive is your CD-ROM drive.

For WFWG 3.1 or higher, or Windows NT, select File | Run from the Program Managers menu. In the Command Line edit box of the Run dialog, enter D:\SETUP or use another drive letter depending on where you inserted the CD-ROM. Click OK.

For Windows95, select Start | Run from the task window. In the Open edit box of the Run dialog, enter D:\SETUP or use another drive letter depending on where you inserted the CD-ROM. Click OK.

Follow the instructions given in the installation program.

For instructions on how to install InterAct into a programming environment such as Microsofts Developer Studio or Visual Basic, or ProtoViews ViewPaint, see Section III: InterAct Tutorials. These tutorials will also show some of the ways in which InterAct can be programmed.

Product Support

If you have questions about InterAct, you may contact our Technical Support staff.

ProtoView now offers different types of technical support services for you to choose from including 30 days of complimentary telephone support, standard support, and available extended support packages.

When contacting our Technical Support staff, please provide the following information:

- Your product registration number found on the label of each disk that you received with InterAct.
- The version of InterAct you are using and the file dates of any updates you have installed.
- The exact wording of any messages that appeared on your screen.
- The error code returned from the property or method you are having a problem with.
- A brief and concise explanation of the question or problem that you are experiencing.
- How you tried to solve the problem.

Complimentary Technical Support

If you are a registered user, free technical support is available by telephone for 30 days from the date of your first telephone call. (Standard Technical Support is always available at no charge.)

To obtain technical support by telephone, call (609) 655-5000. Technical Support hours are Monday through Friday, 9:00 A.M. to 5:00 P.M. Eastern Standard Time (excluding holidays).

Standard Technical Support

Available at no charge, Standard Technical Support provides all registered users with technical support through the following services:

Bulletin Board Services(BBS)

ProtoView Development maintains a 24 hour Bulletin Board Service. This bulletin board is available for technical questions and problem reports. It also contains updated information about InterAct along with a library of examples and source code that can be downloaded for your own use.

This service is free of charge to all users and can be reached at (609) 655-4411. The

communication parameters are 9600/2400/1200/300,8,N,1.

On-line Support Forum

You can join ProtoView on CompuServe where you can ask questions, download files and communicate with other ProtoView users. To access the ProtoView forum on CompuServe, type GO PROTOVIEW at the CompuServe command prompt.

Internet Support

You can ask questions and download files with ProtoView users on the Internet. To access the ProtoView site use the following addresses:

http://www.protoview.com

ftp.protoview.com

tech@protoview.com

ProtoView Technical Fax

You may also ask about a problem or question by sending a facsimile message. The number for faxing is (609) 655-5353. Be sure to include your name, fax number, telephone number and as much information about the problem as possible. Our technical support personnel will review your questions and fax back a reply.

Extended Support Packages

Several support packages are available at an additional charge. ProtoView offers priority, premier, and pay-as-you-go support options. Call ProtoView sales department at 1-800-231-8588 for additional information about the different support packages available.

Product Training and Consultation Services

Within the United States, ProtoView Development offers the following services for training and consultation.

ProtoView Professional Services

ProtoView Professional Services offers education and training in many areas of Windows programming. Both beginning and advanced courses are available in 3 or 5 day units

ProtoView Professional Services has highly skilled and experienced personnel available for the development of Windows applications and systems. These services can be provided on-site, off-site, full or part time. For more information, PPS can be reached at 770-390-6226 or info@protoview.com.

InterAct File Listing

InterAct will install different components based on the install you purchased and options you select during the install process. Certain components will not be installed if you do not select those options during the installation.

An InterAct DLL setup may install the following components:

1. The InterAct 16-bit dynamic link library (PVIDO.DLL)
2. The InterAct 32-bit dynamic link library (PVIDO32.DLL)
3. Two support DLLs (PGHT.DLL, PGHT32.DLL)
4. The InterAct header file (PVIDO.H)
5. The InterAct DLL import libraries (PVIDO.LIB, PVIDO32.LIB, PVIDO32B.LIB, PVIDO32S.LIB)
6. The InterAct MFC classes (IDOMFC.H)
7. The InterAct OWL classes (IDOOWL.H)
8. The InterAct Diagramming Assistant, a program that allows you to visually design diagrams (INTERACT.EXE)
9. Sample Files

10. InterAct Help Files

An InterAct ActiveX setup may install the following components:

1. The InterAct 16-bit ActiveX library (PVIDO.OCX)

2. The InterAct 32-bit ActiveX library (PVIDO32.OCX)

3. Two support DLLs (PGHT.DLL, PGHT32.DLL)

4. The InterAct Diagramming Assistant, a program that allows you to visually design diagrams (INTERACT.EXE)

5. Sample Files

6. InterAct Help Files

Introduction

The sections in this chapter will give a comprehensive demonstration of InterAct. This tutorial will not cover any of the programming aspects of InterAct - it will simply demonstrate with step-by-step instructions how to manipulate InterAct using the mouse and keyboard. For this tutorial we will use the sample program InterAct Diagramming Assistant installed with InterAct.

Creating Diagrams

To run the InterAct DiagrammingAssistant:

When you installed InterAct, an icon to launch the InterAct Diagramming Assistant was installed. Double-click this icon to begin the InterAct Diagramming Assistant. If you do not have this icon, browse to the directory where you installed InterAct and run the program INTERACT.EXE.

The InterAct Diagramming Assistant

After the InterAct Diagramming Assistant splash screen disappears, the InterAct Diagramming Assistant will open a new window automatically. An InterAct control will completely occupy the window. InterAct will have a grey background and white grid lines evenly spaced. Also notice that a tools palette is floating over the upper-right corner of InterAct. The tools palette is maintained completely by InterAct and provides an advanced interface for modifying a diagram. We will begin using InterAct by

loading a diagram already created.

To load a diagram:

Select the Tools Palette button Options | Load Diagram. With the Common Dialog select the file GTOUR.IDO. InterAct will now load the diagram saved in the GTOUR.IDO file.

The diagram GTour.IDO.

InterAct will restore all the entities and relations in the diagram along with the colors, styles, and graphics of each item. An IDO file also stores all the attributes of a diagram, including background and gridline color, grid properties, and zoom mode.

One of the uses for InterAct is to create diagrams beforehand and load those diagrams when needed. These diagrams can be modifiable or read-only.

To mark this diagram read-only:

Select the menu item Options | Edit Mode. The checkmark will be removed from this

menu option and InterAct is made read-only. Notice that the tools palette disappears automatically when InterAct is marked as read-only. Since the InterAct is read-only and the tools palettes main purpose is to modify the diagram, it is unnecessary to display the tools palette.

Now if you click on any objects or attempt to drag objects to a new location your efforts will be ignored. However, notice that the selection rectangle changes to the last relation, entity or entities clicked. The container application is also notified of these clicks and selection changes so it can provide a response to these events even though the diagram is read-only.

We will restore InterAct to be editable and make some changes to the diagram.

To make InterAct editable:

Select the menu item Options | Edit Mode. The checkmark will be added to this menu option and InterAct made editable. Notice that the tools palette reappears automatically.

For more information about creating and altering diagrams, see Chapter 2: The InterAct Environment.

Modifying Objects in a Diagram

InterAct provides many built-in dialogs for modifying diagrams. Many of these dialogs take the form of property pages - that is, a collection of dialogs which allow customization of common features. It is entirely up to the programmer to decide which of these dialogs to use and which to ignore.

To modify InterActs attributes:

1. Right click on InterActs background, not on an entity or relation. InterActs floating menu will appear.
2. Select the menu item Diagram Colors from the floating menu. This takes us to the Colors property page. This property page allows you to select any of the predefined colors for the InterAct control background or grid lines.
3. Select the color Light Red for the Background Color.
4. Click the Apply button to have this change accepted immediately. InterAct is immediately updated with this new color.

Note: Once you apply changes you cannot cancel these changes.

5. We will now change the colors of InterAct to something less bright. Change the

Background Color back to Light Cyan. Change Grid Line Color to Red.

6. Now display the Grid property page. The main attributes of the Grid property page is the spacing between grid lines.

7. Set the Horizontal Spacing to 25.

Note: To change the value of these fields you can either enter the new value, use the scrollbar on the right edge of the field, or use the up/down arrow keys when focus is in the field.

8. Set the Vertical Spacing to 35.

9. Click OK to have these changes applied to InterAct and close the property window.

Our diagram now has red grid lines on a light blue background. Grid lines can be an important aid in helping users design their diagram. InterAct can automatically snap entities in the diagram to the nearest grid line to make it easier to align entities, even if grid lines are not visible. We will hide grid lines in our Guided Tour diagram.

To hide grid lines:

Select the menu item Options | Grid Lines. The checkmark will be removed from this

menu option and the grid lines will disappear.

We will now modify some of the attributes on an entity in the diagram.

To modify an entitys attributes:

1. Right click in the interior of the entity with the text Start with an Idea. The entitys floating menu will appear.

2. Select the menu item Graphics. This takes you directly to the Graphic property page.

3. Click the button marked Select to select a bitmap file. A common dialog Select BMP File will appear.

4. Navigate to the InterAct directory and select the bitmap GTOUR.BMP. Click OK to accept this file to return to the Graphic property page. Notice that the property page displays a preview of this bitmap.

5. Check on the option Stretch Bitmap.

6. In the Text Placement group, select the radio button Right of Entity. Any text associated with the entity will be placed to the right of the graphic. We will now add that text.

The Graphics Property page.

7. Display the Text property page.

8. Replace the existing text with the text Guided Tour in the field.

9. Click the radio button Left. This will left justify the text.

10. Display the Colors property page.

11. Select Light Cyan as the Background Color.

12. Display the Styles property page.

13. Turn off Border, and set 3D Effect to None.

14. Click the Apply button. These changes will be applied to the entity.

15. Click the Close button.

An entity after we have applied some changes to it.

To modify a relations attributes:

1. Right click on a portion of the line representing the relation originating directly below the entity Guided Tour. The relations floating menu will appear.

Note: Make sure you click on the line itself and not the text associated with the line.

2. Select the menu item Styles. This takes you directly to the Styles property page.

The Styles page of the relation property page.

3. Change Line Thickness to 4.

4. Select White Circle as the Source Arrow.

5. Select Wide as the Destination Arrow.

6. Click the Apply button. These changes will be applied to the relation.

7. Click the Close button.

The relation after we have set some properties on it.

For more information about the available properties, see Chapter 3: Property Pages.

Working with the Tools Palette

In addition to being able to restore entire diagrams at once, another use is the ability to save and restore only a specific palette. A palette contains a collection of entity and relation classes, each describing how an object will look or behave. A palette can also contain other useful information, such as rules which govern how entities can be joined by relations, initial colors of InterAct, and spacing of gridlines. To see this, we will load a palette into InterAct. First, close this window by selecting File | Close. Next, open a new InterAct window by selecting File | New. A fresh InterAct window will be displayed with a tools palette containing a default selection of classes.

To load a new tools palette:

Remove the current palette by selecting File | Reset Palette. All the classes are removed from the palette.

Select the menu File | Load Palette.

Select the file GTOUR.PLT.

InterAct will now load the palette file. Notice the new assortment of buttons on the palette and the new color and grid line spacing of InterAct.

To add an entity from the tools palette:

1. Click on one of the entity symbols on the tools palette.

2. Move the mouse over the InterAct. The mouse cursor will change shape to a hand with a pointer.

3. Click and hold down the left mouse button.

4. Holding the left mouse button down, drag the mouse. A rectangle will be drawn as you drag the mouse to indicate the area which will be occupied by the new entity.

5. Release the left mouse button. A new entity will be added to the diagram.

6. Repeat this procedure to add any number of entities.

To add a relation from the tools palette:

1. Click on one of the relation symbols on the tools palette.

2. Move the mouse over the InterAct. The mouse cursor will change shape to a hand with a pointer.

3. Click and hold down the left mouse button on an entity.

4. Holding the left mouse button down, drag the mouse. A line will be drawn as you drag the mouse to indicate the area which will be occupied by the new relation.

When you move the mouse over a valid target entity the mouse cursor will become inverted.

5. Release the left mouse button when over a valid target entity. A new relation will be added to the diagram between the source and target entities.

6. Repeat this procedure to add any number of relations.

For more information about the tools palette, see Chapter 4: The Tools Palette.

Exploring Classes and Rules

As the previous section showed, loading and using different palettes is an important feature of InterAct. Palettes are just a visual representation of internal classes which InterAct maintains. Each class is simply a series of conditions which determine how an entity will behave and look. Being able to create and maintain these classes is an important part of using InterAct.

Altering an existing class:

Click on the Options button of the Tools Palette. Select the menu item Manage Classes. The Manage Classes dialog will be displayed. This dialog lists all the classes available in InterAct.

The Manage Classes dialog.

In the Entity Classes groupbox, select the class Generic Rectangle and press the Redefine Class button. A property page similar to the one displayed when you edit

properties on an entity is displayed.

Display the Colors property page.

Set the Background Color to Light Blue.

Display the Styles property page.

Set 3D Effect to Shadow.

Click OK. These changes will now be accepted. Notice that all the entities of the class Generic Rectangle now have the light blue background and shadow as a 3D effect.

Creating a new class:

In the Entity Classes groupbox, click the Define New Class button. The Register New Class dialog is displayed.

Enter the Class Name Tour.

For the Class Icon, press the Select button. This allows you to select a bitmap from disk to act as an icon.

Note: Bitmaps displayed on the tools palette should be 16 x 16 pixels in size to appear properly.

Select the bitmap IGTOUR.BMP from InterAct directory.

Defining a new class.

Click Apply. A property page similar to the one displayed when you redefined an entity class is displayed with the caption Register New Class: Tour.

On the Colors property page, set the Background Color to be Green.

On the Styles property page, set the Shape to be Circle.

Click Done to Close the Manage Classes dialog and have these changes reflected on the tools palette. You can now select the newly created class from the tools palette and place objects of that class in the diagram.

Enabling Rules:

Select the menu item Options | Enforce Rules. Rules are now being enforced in the diagram. Select Options | Manage Classes from the Tools Palette. Notice that a button Rules is now visible in the Manage Classes dialog. Click this button to go to the

Manage Rules dialog.

Select all the items in the Diagramming Rules list and click the Delete Rule button.

When asked to confirm the delete select Yes.

Adding a Rule:

1. Click the Add Rule button.

2. In the First Entity list select Tour, the class we defined earlier.

3. In the Relationship list select Generic Relation.

4. In the Second Entity list select Generic Rectangle.

Adding a rule.

5. Click Apply. Click Done to leave the Manage Rules dialog. Click Close to close the Manage Classes dialog.

Enforcing Rules:

InterAct will now enforce any and all rules within a diagram. Select the menu File |

Reset. All the entities and relations will be removed from the diagram. Now add three entities, two of type Tour and one of type Generic Rectangle.

Now if you try to create a Generic Relation between the two Tour entities InterAct will not let you. It will however allow a Generic Relation to connec a Tour entity to a Generic Rectangle. This support is maintained completely by InterAct.

For more information about classes and rules, see Chapter 5: Classes and Rules.

Summary

This concludes the guided tour of InterAct with the InterAct Diagramming Assistant. The tutorial is designed to show some of the basic capabilities of InterAct which a programmer of InterAct can make use of. For tutorials which contain programming samples of InterAct for C, C++ and Visual Basic users, see Section II of this manual.

Overview

InterAct is a diagramming object which contains two basic types of objects - entities and relations.

Entities

Entities are a rectangular region in a diagram which can be manipulated. Entities can have different shapes and colors. Optionally, an entity can contain text and/or a graphic. Entities can be moved and resized throughout a diagram.

Relations

Relations are lines which connect entities. Lines can have two, three or four points which can be positioned, and a color. Optionally, symbols can be added to either or both endpoints, and text can be associated with a line.

Diagrams

Using only these two basic objects, you will be able to create a wide array of detailed diagrams to suite almost any need. Learning how to manipulate these objects using the mouse and keyboard will allow you to create diagrams faster and easier.

How to Modify the InterAct Environment

To select an entity:

Left-click the mouse anywhere within the rectangular region of the entity. A selection rectangle with handles in the corners and on the sides of the entity will appear.

An entity before and after it is selected.

To select more than one entity:

Left-click the mouse anywhere within the InterActs client area, but not within a region occupied by an entity.

Left click to start drawing a selection rectangle.

Keeping the left mouse button depressed, drag the mouse. A rectangular rubber-band

will be drawn on the screen as you drag the mouse.

Encircle entities with the selection rectangle.

Release the left mouse button. Any entities contained within the rectangular rubber-band will be selected. A rectangle will be drawn around all the entities in the group. Notice that only one of the selected entities has track handles surrounding it. This entity is the anchor of the group, and is the main entity in the selected group.

Release the mouse button. The entities within the selection rectangle are now selected.

To add or remove entities to a selection group:

Holding the Shift key down, left-click the mouse on an entity. The entity will be added to the selected group, and have a rectangle drawn around it. If the entity is already

contained in the group, it will be removed from the group and the rectangle around it will be removed.

To move an entity:

Left-click the mouse within the area of a selected entity.

Notice the different shape of the mouse cursor when moved over a selected entity.

Keeping the left mouse depressed, drag the mouse. A shadow image of the entity or entities being moved will be displayed.

Shadows of the selected objects will be drawn as you drag them to a new location.

When you have positioned the entities in their new position, release the mouse button.

The selected entity or entities will be moved.

The selected objects are now moved to their new location.

To resize an entity:

Select an entity. A selection rectangle with handles will be drawn around the entity.

Move the mouse over a handle positioned at a corner or side of the entity (these are called resizing handles). The mouse cursor will change shape when it is moved over the resizing handles.

Notice the cursor shape when moved over an entity corner.

Press the left mouse button.

Keeping the left mouse button depressed, drag the mouse to a new location. As you drag the mouse, a rectangle depicting the entitys new size will be drawn.

A shadow image of the entity will be displayed when dragging to a new size.

Release the mouse button when you have reached the desired size. The entity will be redrawn at the new size.

The entity is now resized.

To connect a relation between two entities:

Select an entity. A selection rectangle with handles will be drawn around the entity.

Move the mouse over the side of the entity, but not over a resizing handle. The mouse cursor will change shape when it is moved over these handles.

Notice the shape of the cursor when moved over the entity side.

Press the left mouse button. This entity is the source entity.

Keeping the left mouse button depressed, drag the mouse to a new location on top of another entity. This entity is the destination entity. As you drag the mouse, a line depicting the relation being created will be drawn. As the mouse is dragged over entities in the diagram, it will change color to let you know you are over a valid target entity.

A shadow of the relation will be drawn as you select a target entity.

Release the mouse button when you have reached the desired destination entity to connect the relation to. A relation will be drawn between the source and destination entities.

The relation is now created between the source and target entities.

To edit the text on an entity:

Double-click the left mouse button on an entity. A window with the entitys current text will appear. Enter the new text. Hit the Enter key to accept this change. Press the Escape key to abort the change.

To select a relation:

Left-click the mouse anywhere along the line of the relation. Selection rectangles will appear at the end- and mid- points of the line. You can click and drag these end-or mid- points to reposition the line.

A relation with selection rectangles at its ends.

To change the shape of a relation:

Select a relation. Red selection markers will be drawn at the end points and midpoints of the line to represent the drag-handles of the line. Some relations will not have midpoints. Move the mouse over a drag-handle and press the left mouse button.

Holding the left mouse button down, drag the handle to a new location. Releasing the left mouse button will move the point to the new location. Pressing Escape while keeping the left mouse button down will abort the operation.

To edit the text on a relation:

Double-click the left mouse button on a relation. A window with the relations current text will appear. Enter the new text. Hit the Enter key to accept this change. Press the Escape key to abort the change.

To add an entity to a diagram:

Click on one of the entity symbols on the tools palette.

Move the mouse over InterAct. The mouse cursor will change shape to a hand with a pointer.

Click and hold down the left mouse button.

Holding the left mouse button down, drag the mouse. A rectangle will be drawn as you drag the mouse to indicate the area which will be occupied by the new entity.

Release the left mouse button. A new entity will be added to the diagram.

Repeat this procedure to add any number of entities.

To add a relation to a diagram:

Click on one of the relation symbols on the tools palette.

Move the mouse over an entity in InterAct. The mouse cursor will change shape to a hand with a pointer.

Click and hold down the left mouse button.

Holding the left mouse button down, drag the mouse. A line will be drawn as you drag the mouse to indicate the area which will be occupied by the new relation.

Drag the mouse over a target entity.

Release the left mouse button. A new relation will be added to the diagram connecting the first entity clicked with the target entity.

Repeat this procedure to add any number of relation.

To delete a relation:

Select a relation. Press the Delete key. The selected relation will be deleted.

To delete an entity or group of entities:

Select one or more entities. Press the Delete key. The entities will be deleted. Any relations pointing to or from the deleted entities will also be removed.

Summary

This chapter has described the keyboard and mouse commands available to manipulate with InterAct. With the mouse and keyboard you can completely customize an InterAct diagram. This chapter has only briefly talked about the tools palette. The next chapter will discuss this component in much more detail.

Introduction

InterAct allows access to built-in property pages which provide access to styles and features of InterAct as well as individual entities and relations.

InterAct Property Pages

InterAct has two property pages - Colors and Grid.

InterAct Colors

Allows selection of colors for InterAct.

InterAct Colors property page.

Background Color

A list of available colors is displayed to choose from. This color is used when drawing InterActs background.

Grid Line Color

A list of available colors is displayed to choose from. This color is used when drawing InterActs grid lines, if grid lines are being displayed.

InterAct Grid

The InterAct Grid property page allows you to set properties which define the gridlines

for the diagram. Spacing, whether snap-to-grid is enabled, and whether gridlines are visible are all defined on the property page.

InterAct Grid property page.

Horizontal Spacing

Allows you to specify the horizontal spacing for grid lines, if grid lines are being displayed. Even if grid lines are not being displayed, the width is used by Snap-To-Grid to snap an entity to the nearest invisible grid line. When the cursor is in this field, the value can be incremented/decremented by pressing the Up Arrow and Down Arrow keys, respectively.

Vertical Spacing

Allows you to specify the vertical spacing for grid lines, if grid lines are being displayed. Even if grid lines are not being displayed, the height is used by Snap-To-Grid to snap an entity to the nearest invisible grid line. When the cursor is in this field, the value

can be incremented/decremented by pressing the Up Arrow and Down Arrow keys, respectively.

Snap to Grid

Toggles if InterAct will snap entities upper-left corner to the nearest grid line intersection. A useful feature for easily arranging entities. Does not depend on Grid Lines being visible.

Grid Lines Visible

Toggles whether InterAct will display grid lines.

Show Printer Page Lines

Toggles whether InterAct will show page lines in a large diagram. Printer page lines are useful if printing a diagram is necessary and the user needs to see the approximate pagination of the print. InterAct must be able to locate the default printer of the workstation for these lines to be drawn correctly.

Entity Property Pages

Entities have four property pages - Text, Colors, Styles and Graphic.

Entity Text

The Text property page allows you to set properties which affect the display of an entity. Of course the most obvious is the actual text displayed by an entity. You can also select the font and orientation of the text. Of equal importance, you can see the ID and Name assigned to the entity. Usually, you will provide the ID or Name when you create the entity, but if you do not supply one, InterAct will create default values. You will also see the class type of the entity.

The Entity Text property page.

Class Type

This is the class of the entity. The class determines what the entity will look like and how it will behave when it is initially created. If the attributes of a class are modified,

the entities of that class will automatically assume the new attributes. If rules are being used InterAct, class name will also be important, as rules refer to how classes can interact.

User ID

A numeric value which uniquely identifies the entity.

User Name

A string which uniquely identifies the entity.

Font

Click this button to select a new font for the entity. Different point sizes and styles can be selected for this font. Each entity can have a different font, if selected.

Left / Center / Right

Select the orientation of the text when aligned in the entity.

Text

The actual text which will be displayed by the entity. The text will be displayed in the selected font and orientation. Text will always be clipped within the region defined by the entity. If the entity is selected to automatically size to accommodate all the text, all

the text will be visible. See the Entity Styles property page for details about the AutoResize option. If the entity is displaying a graphic the text will not be displayed inside the entity. It will be displayed above, below, to the left or right of the entity. See the Graphic property page for an explanation of these options.

Entity Colors

Allows selection of colors for the entity.

The Entity Colors Property Page.

Text Color

A list of available colors is displayed to choose from to use when drawing the entitys text. If the entity is also displaying a border, the border will be drawn in this color as well. See the Entity Styles property page for details about the Border option.

Background Color

A list of available colors is displayed to choose from to use when drawing the entitys

interior. If the entity is selected to be transparent, this color selection will have no effect. See the Entity Styles property page for details about the Transparent option.

Entity Styles

A variety of styles can be applied to each entity that have major effects on the look and behavior of the entity. Any number of styles can be applied, and their effects are cumulative. As an example, if you want to create a Simple Text entity, you would select the styles Transparent, AutoResize to Text, and Not a Valid Relation Source.

The Entity Styles property page.

Cannot be Resized

The entity cannot be resized using the mouse. Attempts to use the drag handles to resize the entity will be ignored.

Note: You can still resize an entity by setting properties on the entity at runtime. This style only prevents the use of the mouse to resize the entity.

Cannot be Moved

The entity cannot be moved from its current location using the mouse.

Note: You can still reposition an entity by setting properties on the entity at runtime.

This style only prevents the use of the mouse to reposition the entity.

Cannot be Deleted

The entity cannot be deleted if the entity is currently selected and the Delete key is

pressed.

Note: You can destroy an entity at runtime by invoking the InterAct method DeleteEntity

method. This style only prevents the use of the Delete key to delete the entity.

Read Only

Normally, double-clicking an entity will allow you to edit the entitys text. This style

suppresses this behavior, creating a read-only entity.

Note: You can still change the text on an entity by setting properties on the entity at

runtime. This style only prevents the use of the mouse to edit the entitys text.

Auto Resize to Text

The entity will automatically resize to encompass the text in the entity. If you stretch

an entity horizontally, the entity will resize horizontally to the closest width, and then resize vertically to include all text. Resizing an entity only vertically will be ignored.

Not a Valid Source if Dragging a New Relation

The entity cannot be used as a source for a relation when using the mouse.

> Note: You can use this entity as a relation source if you add relations at runtime invoking InterAct methods. This style only prevents the use of the mouse to drag a relation from the entity.

Act as a Container

T

he

en

A

tit

co

y

nt

wi

ai

ll

ne

be

r

ha en

ve tity

as .

a

co

nt

ai

ne

r.

A

co

nt

ai

ne

r

is

dr

a

w

n

in

a

sp

ec

ial

w

ay

.

A

ho

llo

w

fr

a

m

e

is

dr

a

w

n

w

hi

ch

ca

n

su

rr

ou

nd

en

titi

es

.

A

ca

pti

on

at

th

e

to

p

of

th

e

fr

a

m

e

is

dr

a

w

n

w

hi

ch

co

nt

ai

ns

th

e

te

xt.

A

ny

en

titi

es

in

si

de

a

co

nt

ai

ne

r

wi

ll

m

ov

e

wi

th

th

e

co

nt

ai

ne

r

w

he

n

th

e

co

nt

ai

ne

r

is

m

ov

ed

.

Permanently Hold Objects

This option is only available if Act as a Container is enabled. Any objects contained within the container can never be removed from the container. If the user attempts to move an entity from the container using the mouse InterAct will force the entity back into the container.

Transparent

The entity will be transparent - allowing objects behind the entity to be seen. This option also affects a graphic which is being displayed. The color white is considered to be the transparent color of a bitmap. If a graphic is being displayed in an entity which has this option selected, the image cannot be stretched. See the Graphics property page for a description of how graphics are treated in an entity.

3D Effect

A variety of 3D effects can add depth to a diagram. Each entity can have no 3D effect or one of the following: Indent, Heavy Indent, Outdent, Heavy Outdent, Shadow. 3D effects can be used in addition to a Border, they are not mutually exclusive.

Border

A thin border will be drawn around the entity. The color for this border is the same as the color selected for Text Color. See this option on the Colors property page. A border can be used in addition to a 3D Effect, they are not mutually exclusive.

Entity Shape

Determines the basic shape of an entity. Choices include: Rectangle, Circle (Elliptic) and Round Rectangle. Border and 3D Effects can be applied to all these shapes. If you are displaying a graphic in an entity, it draws as a Rectangle shape regardless of this selection.

Entity Graphic

Allows selection of a bitmap to be displayed in an entity. If a graphic is displayed in an entity, the image will be stretched to fill the entire area of the entity unless the entity is selected to be transparent. In this case, the image will be shown the same size as when it was created. See the Entity Styles property page for more information about the Transparent option.

The Entity Graphic property page.

Display Bitmap

The name of a BMP file. May also include a full path to the BMP file. If no path is supplied, the BMP file must be able to be located by InterAct, either in the same directory as InterAct or in a directory in the machines path. If the BMP can be loaded, it will be loaded and displayed in the window directly below for you to preview the bitmap.

To remove a bitmap, simply remove the bitmap name from the Display Bitmap field.

Select

Click the button below the Display Bitmap field to select a BMP file on disk. If a BMP file is selected, it will be loaded and displayed in the window. Otherwise the text Click here to select bitmap. will be displayed.

Remove

Removes a graphic from the entity, clearing the Preview window and the Display

Bitmap field.

Text Placement

This affects text when a graphic is being displayed in an entity. The text may not be drawn within the borders of the entity. Instead, you can have the text displayed to the left, right, above or below (the default) of the entity. If you select Centered, the text will be painted on top of the entity.

Stretch Bitmap

Toggles whether the bitmap will be stretched to occupy the entire area of the entity. If this option is off, the bitmap will be centered in the entity, or clipped if the bitmap is larger than the entity. If the style Transparent is selected, Stretch Bitmap has no effect.

Internet

The internet property page allows properties to be set to allow InterAct to initiate an internet session. An entities internet capabilities are initiated on a double-click of the entity. Currently most internet hosts allow a URL command to be passed when the host is started. In the future, other internet hosts will allow a continuous stream of

URL commands to be passed and processed.

The Entity Internet property page.

URL Command

This is the actual command issued to the internet host.

Optional Path for Internet Host

Often the internet host may not be in the path of the system. In this case it is desirable to supply a path. InterAct will try to locate the internet host both with and without this path, and through any other means available to it.

Internet Host

This is the executable which will provide access to the internet.

Execute

This will execute the URL Command immediately. This is primarily meant as a means to test the properties set on this page.

Relation Property Pages

Relations have three property pages - Text, Colors, and Styles. Relations have a special object of text associated with it - called a Line Text. The Line Text is a full-fledged entity in its own right, and can have properties set on it just like a normal entity. However, certain properties in the relation property pages relate directly to this independent Line Text.

Relation Text

The Text property page allows you to set properties which affect the display of a relation. Of course the most obvious is the actual text displayed by the relation, displayed in the Line Text. You can also select the font and orientation of the text. Of equal importance, you can see the ID and Name assigned to the relation. Usually, you will provide the ID or Name when you create the relation, but if you do not supply one, InterAct will create default values. You will also see the class type of the relation.

The Relation Text property page.

Class Type

This is the class of the relation. The class determines what the relation will look like and how it will behave when it is initially created. If the attributes of a class are modified, the relations of that class will automatically assume the new attributes. If rules are being used in InterAct, class name will also be important, as rules refer to how classes can interact.

User ID

A numeric value which uniquely identifies the relation.

User Name

A string which uniquely identifies the relation.

Font

Click this button to select a new font for the relation. Different point sizes and styles can be selected for this font. Each relations Line Text can have a different font, if requested.

Left / Center / Right

Select the orientation of the text when aligned in the relations Line Text.

Text

The actual text which will be displayed by the relations Line Text. The text will be displayed in the selected font and orientation. Text will always be clipped within the region defined by the relations Line Text, unless the Line Text is selected to automatically size to accommodate all the text. See the Entity Styles property page for details about the AutoResize option.

Relation Colors

Allows selection of colors for the relation.

The Relation Colors property page.

Text Color

A list of available colors is displayed to choose from when drawing the relations Line Text. This color will not affect the lines drawn by the relation.

Relation Color

A list of available colors is displayed to choose from when drawing the relations lines and arrow symbols. The relation color will not affect the Line Text.

Relation Styles

A variety of styles can be applied to each relation which affect the look and behavior of the relation. Any number of styles can be applied, and their effects are cumulative.

The Relation Styles property page.

Line Styles

Determines the basic shape of a relation. Choices include: Straight, Three Point, Four Point and Ninety Degree. Straight lines will draw a straight line from the source to the destination entity. The Three and Four Point lines will have 1 or 2 midpoints which can be positioned. Lines will be drawn from the endpoints to the midpoints, and if necessary, between two midpoints. Ninety Degree lines will always draw lines from

the endpoints to the midpoints and between the midpoints at ninety degree angles.

Cannot Move Endpoints

The relations end points cannot be moved with the mouse. Any attempt to drag an endpoint with the mouse will be ignore.

Cannot Move Midpoints

The relations mid points, if any, cannot be moved with the mouse. Any attempt to drag a midpoint with the mouse will be ignore.

Cannot be Deleted

The relation cannot be deleted if the relation is currently selected and the Delete key is pressed.

Read Only

Normally, double-clicking a relation will allow you to edit the relations Line Text. This style suppresses this behavior, creating a read-only relation.

Destination Arrow

A number of different arrow heads are available to display at the relations destination point. Choices include: None, Standard, Narrow, Wide, Swept, Light Circle, and Dark

Circle.

Source Arrow

A number of different arrow heads are available to display at the relations source point.

Choices include: None, Standard, Narrow, Wide, Swept, White Circle, and Dark Circle.

Standard

Swept

Narrow

White Ci

r

c

l

e

d r

e k

C

i

r

c

l

e

Line Thickness

Varies the thickness of the lines being drawn. This can be in the range of 1-5 units

wide. When the cursor is in this field, the value can be incremented/decremented by pressing the Up Arrow and Down Arrow keys, respectively.

Summary

This chapter has explained all the property pages which InterAct can display for itself, and the entities and relations in a diagram. It has explained each option of all the different property pages for each of the objects contained in a diagram.

Overview

The tools palette is an important interface for performing operations on InterAct. With the tools palette you can

- 

Cut/Copy/Pas

t

e

it

e

m

s

t

o

t

h

e

W

i

n

d

o

w

s

C

li

p

b

o

a

r

d

.

.

Z

o

o

m

t

h

e

p

e

r

s

p

e

c

ti

v

e

o

f

l

n

t

e

r

A

c

t.

.

D

i

s

p

l

a

y

H

e

l

p

.

S

e

l

e

c

t

d

e

f

a

u

lt

c

l

a

s

s

e

s

o

f

e

n

ti

ti

e

s

a

n

d

r

e

l

a

ti

o

n

s

t

o

p

a

i

n

t

i

n

I

n

t

e

r

A

c

t.

.

L

o

a

d

/

S

a

v

e

a

d

i

a

g

r

a

m

.

.

L

o

a

d

/

S

a

v

e

a

p

a

l

e

tt

e

.

.

D

i

s

p

l

a

y

a

d

i

a

l

o

g

t

o

m

a

n

a

g

e

c

l

a

s

s

e

s

a

n

d

r

u

l

e

s

.

The tools palette contains the following buttons:

Cut

Delete the selected items and place them in the clipboard. You cannot cut a relation if it is selected. However, if a relation connects two entities which are selected and cut, the relation will also be cut to the clipboard.

Copy

Copy the selected items into the clipboard. You cannot copy a relation if it is selected. However, if a relation connects two entities which are selected and copied, the relation will also be copied to the clipboard.

Paste

Copy items from the clipboard to InterAct. Selected entities which were cut or copied to the clipboard will be pasted into the diagram. If a relation connected any two entities being pasted, then the relation will also be pasted into the diagram.

Zoom In

Zoom the InterAct window for more detail.

Zoom Out

Zoom the InterAct window to see more of the diagram but less detail.

Help

Display the InterAct help file.

Note: If you wish to display a help file which you have defined and not the help file provided with InterAct, you can respond to a notification from InterAct and display your help file instead. See Chapter 10: InterAct Events.

Options

Display a list of options for the InterAct environment. Of particular importance is the ability to load and save diagrams, palettes, and display the ManageClasses dialog.

Note: It is possible to change the text on the Options button and customize the menu which drops down when this button is clicked. See Chapter 10: InterAct Events and Chapter 12: InterAct Methods.

The Speed Buttons

The speed buttons provide a quick way to add entities and relations to a diagram. Clicking on any of the speed buttons adds an entity of the default entity class, relative to the current entity. It also adds a relation of the default relation class from the current entity to the new entity. (See the items Entity Classes and Relation Classes in this section for a description of default classes.) If no entity is currently selected in the

diagram, the new entity is placed in the middle of the current view of the diagram, and no new relation is created.

Entity Classes

The tools palette will display all the available entity classes as buttons, one button per class. Classes usually display a bitmap to convey their use or representation. These buttons can be toggled, meaning that when clicked, they will remain in a depressed position. Only one button of the entity classes will ever be depressed at any time, and one button will always be depressed. This depressed button represents the default entity class. The default entity class is the class which will be added to a diagram when one of the tool palettes speed buttons is clicked.

In addition, when you click one of the buttons on the palette, you can immediately add an entity of the selected class to the diagram using the mouse. Click on one of the entity symbols on the tools palette. Move the mouse over InterAct. The mouse cursor will change shape to a hand with a pointer. Click and hold down the left mouse button. Holding the left mouse button down, drag the mouse. A rectangle will be drawn as you drag the mouse to indicate the area which will be occupied by the new

entity. Release the left mouse button. A new entity will be added to the diagram. Repeat this procedure to add any number of entities.

Relation Classes

The tools palette will display all the available relation classes as buttons, one button per class. Classes usually display a bitmap to convey their use or representation. These buttons can be toggled, meaning that when clicked, they will remain in a depressed position. Only one button of the relation classes will ever be depressed at any time, and one button will always be depressed. This depressed button represents the default relation class. The default relation class is the class which will be added to a diagram when one of the tool palettes speed buttons is clicked. An entity in the diagram must be selected for a relation to be added in this manner. The new relation will be added pointing from this selected entity to the new entity added to the diagram. In addition, when you click one of the buttons on the palette, you can immediately add a relation of the selected class to the diagram using the mouse. Click on one of the relation symbols on the tools palette. Move the mouse over InterAct. The mouse cursor will change shape to a hand with a pointer. Click and hold down the left mouse

button on an entity. This will be the relation source. Holding the left mouse button down, drag the mouse. A line will be drawn as you drag the mouse to indicate the area which will be occupied by the new relation. Release the left mouse button when over a target entity. A new relation will be added to the diagram connecting the source and destination entities. Repeat this procedure to add any number of relation.

Customizing the Tools Palette

The tools palette is a major interface with InterAct. With the tools palette you can easily add entities and relations to a diagram and manage all the different classes available within a diagram. You can move the tools palette anywhere on the screen, not just clipped within InterActs window.

Summary

This chapter has introduced InterActs tool palette. The tools palette is an optional component for interacting with InterAct. With it, the user can quickly and easily add entities and relations to a diagram and manage all the classes used in the diagram.

Introduction

InterAct allows you define classes and rules which affect its built-in behavior. Classes describe how particular entities and relations will look and behave and rules govern when particular relations can be used to connect entities.

Entity Classes

These describe how a particular entity will look and behave when created. For example, if you want all of your entities to appear with black text on a white background with a shadow 3D effect, you can define a class with these attributes. Whenever an entity of that class is created it will automatically be created with these attributes. In addition, if you change the definition for a class, all the entities of that class will automatically inherit that change.

Relation Classes

These describe how a particular relation will look when created. For example, if you want all of your relations to appear red with a wide arrow shape, you can define a class with these attributes. Whenever a relation of that class is created it will automatically be created with these attributes. In addition, if you change the definition for a class, all

the relations of that class will automatically inherit that change.

Rules

Rules define when it is legal to have a relation connect two entities. For example, if you have two entities, one of a class called Person and the second from a class called Job, you can define a rule which allows a relation class Works to be connected from Person to Job. This can be described with the notation:

Object <Action> Object.

In our example, the rule would be Person <Works> Job.

With only this single rule, InterAct would prevent the creation of a relation of class Works from a class Jobs to a class Person, or from Person to Person or class Job to Job. Rules and classes are an important component for allowing InterAct to validate the visual selections made by a user.

What do classes and rules provide for your diagram? When representing internal data structures in a visual way, it is often necessary to restrict the types of objects that can be displayed, and how they can interact. Rules and classes provide the mechanism for this type of validation.

Also, when you change an attribute on a class, all the entities or relations of that class will automatically inherit the new attributes. This provides a convenient method to make global changes in a diagram without having to iterate through all the objects in the diagram.

Entity Classes

You can define an entity class by selecting the option Manage Classes from the tools palette Options button. The Registered Classes dialog will appear which allows you to manage all your entity classes.

In the Entity Classes group is a list of all the available classes.

How to Define a New Entity Class

Click the Define New Class button in the Entity Classes group. The Register New Class dialog will appear.

Enter the name of the new class in the New Class Name field. The class name must be unique. You can also select a class icon in the Class Icon combobox. The class

icon is displayed on the Tools Palette. InterAct supplies a set of predefined class icons, or you can press the Select button to select your own bitmap graphic to represent the class icon.

Note: Bitmaps displayed on the tools palette should be 17 x 17 pixels in size to appear properly.

When satisfied with your selections, click the Apply button. Click the Cancel button to abort the class definition.

If you click the Apply button, the dialog Register New Class: <classname> will appear. This dialog has a series of property pages allowing you to customize the look of your new entity class. When finished customizing these details, click the OK button to accept the new class or click the Cancel button to abort the process.

How to Modify an Entity Class

Select a class in the Entity Classes group list. Click the Modify Class button. You will be taken to the Redefine Class: <classname> dialog. This dialog contains a series of property pages allowing you to customize the look of your existing entity class. When finished making changes, click the OK button to accept the changes or click the Cancel

button to abort the changes.

If you click OK, any entities of this class in your diagram will automatically be updated with the new changes.

How to Delete an Entity Class

Select a class in the Entity Classes group list. Click the Delete Class button. You will be prompted to make sure you want to delete the selected class. Click Yes to delete, No to abort.

How to Select a Default Entity Class

Click on the upper-left corner of the Entity Classes button. By holding the left mouse button down, a list of all the available classes will be displayed under the button. Keeping the left mouse button depressed, move the mouse cursor over the desired class you wish to make the default class. The classes will be highlighted as you move the mouse cursor over them. Once you have selected the desired entity class, release the mouse button. If you wish to abort the class selection, press the Escape key.

Relation Classes

You can define a relation class by selecting the option Manage Classes from the tools palette Options button. The Registered Classes dialog will appear, which allow you to manage all your Relation Classes.

How to Define a New Relation Class

Click the Define New Class button in the Relation Classes group. The Register New Class dialog will appear.

Enter the name of the new class in the New Class Name field. The class name must be unique. You can also select a class icon in the Class Icon combobox. The class icon is displayed on the Tools Palette. InterAct supplies a set of predefined class icons, or you can press the Select button to select your own bitmap graphic to represent the class icon.

Note: Bitmaps displayed on the tools palette should be 16 x 16 pixels in size to appear

properly.

When satisfied with your selections, click the Apply button. Click the Cancel button to abort the class definition.

If you click the Apply button, the Register New Class: <classname> dialog will appear. This dialog contains a series of property pages allowing you to customize the look of your new relation class. When finished supplying these details, click the OK button to accept the new class or click the Cancel button to abort the process.

How to Modify a Relation Class

Select a class in the Relation Classes group list. Click the Modify Class button. You will be taken to the Redefine Class: <classname> dialog. This dialog contains a series of property pages allowing you to customize the look of your existing relation class. When finished making changes, click the OK button to accept the changes or click the Cancel button to abort the changes.

If you click OK, any relations in your diagram of this class will automatically be updated with the new changes.

How to Delete a Relation Class

Select a class in the Relation Classes group list. Click the Delete Class button. You will be prompted to make sure you want to delete the selected class. Click Yes to delete, No to abort.

How to Select a Default Relation Class

Click on the upper-left corner of the Relation Classes button. By holding the left mouse button down, a list of all the available classes will be displayed under the button. Keeping the left mouse button depressed, move the mouse cursor over the desired class you wish to make the default class. The classes will be highlighted as you move the mouse cursor over them. Once you have selected the desired relation class, release the mouse button. If you wish to abort the class selection, press the Escape key.

Rules

Press the Rules button on the Manage Classes dialog to display the Manage Rules dialog. The Manage Rules dialog allows you to manage all your rules by adding new rules, modifying existing rules or deleting rules.

In the center of the dialog is the Diagramming Rules list which contains all the Rules in effect for the diagram.

How to Add a Rule

To add a new rule, press the Add Rule button. The Define New Rule dialog will be displayed.

You can select the source entity class, relation class, and destination entity class that compose the rule. When satisfied, click the Apply button.

How to a Delete Rule

Select one or more rules in the Diagramming Rules list. Click the Delete Rule button. You will be prompted to make sure you want to delete the selected rule. Click Yes to delete, No to abort.

How to Modify a Rule

Select a rule in the Diagramming Rules list. Click the Redefine Rule button. Or double-click on a rule in the Diagramming Rules list. The Redefine Rule dialog will be displayed, allowing you to modify the existing rule.

Enforcing Rules

Often, it is not desirable to have rules enforced in a diagram. This level of support can be toggled by using the property RulesEnforced. See Chapter 11: InterAct Properties

for a description of this property. By default, rules are not enforced in a diagram.

Summary

This chapter has introduced the concepts of Entity Classes, Relation Classes, and Rules. With these features of InterAct, very sophisticated interactions can be achieved with little effort.

Introduction

This tutorial will show how to use InterAct using Visual C++ 4.x. This tutorial will show how you can:

- register InterAct in the Developer Studios Component Gallery
- add InterAct to a dialog and set properties on InterAct
- programmatically add entities and relations to a diagram
- respond to mouse clicks within the InterAct environment

If you want a tutorial regarding how to use InterAct using the mouse and keyboard, see Chapter 1: A Guided Tour of InterAct.

Adding InterAct to an Application

Creating a Generic Program

Choose the File | New menu item to invoke the New dialog. Choose Project Workspace from the list and press OK. The New Project Workspace dialog is now visible. Choose MFC AppWizard (exe) from the Type list box. Enter idodemo in the Name edit box, and press the Create button. On the MFC AppWizard - Step 1 dialog, choose Dialog Based as the type of application and press the Next button. On the MFC AppWizard - Step 2 dialog, select the OLE Controls checkbox and press the Finish button. On the New Project Information dialog, press the OK button.

Installing InterAct in Microsofts Developer Studio

The 32-bit InterAct ActiveX can be installed into Microsofts Developer Studios Component Gallery. If the InterAct ActiveX is properly registered it will automatically be added to the component gallery.

Note: VC++ 4.0 did not automatically add registered controls to the component gallery. You must explicitly add InterAct to the component gallery. To add InterAct to the Component Gallery, choose the menu item Insert | Component to invoke the Component Gallery dialog. From the OLE Controls tab page, choose the ProtoView InterAct Object icon, and press the Insert button. Press the OK button on the Confirm Class dialog. Press the Close button on the Component Gallery dialog.

To add the InterAct ActiveX to an application you must be sure to insert InterAct from the Component Gallery into your application. In Visual C++ select Insert | Component to invoke the Component Gallery dialog. Click the OLE Controls page of the page of the Component Gallery.

Highlight InterAct and click the Insert button. VC++ will confirm that you want to add the InterAct ActiveX to your project. Including the InterAct ActiveX will add several

source and header files to your project. Click OK to confirm the addition of these files and click Close to close the Component Gallery.

Adding InterAct to a Dialog

From the resource page of the Project Workspace window, double-click the entry IDD_IDODEMO_DIALOG.

The Project Workspace window.

Delete the TODO: Place dialog controls here. static text field. Stretch the new dialog so that it is a bit bigger and move the OK and Cancel buttons so they are still on the right edge of the dialog.

Select the InterAct icon from the control palette and place the control on the dialog.

Note: If InterAct does not appear on your control palette see the section Installing the IDO in Microsofts Developer Studio.

Size the InterAct control so that it takes up most of the dialog.

InterAct when designing in AppStudio.

InterAct will be drawn as a simple rectangle with gridlines drawn at even intervals.

Setting Properties on InterAct

Double click on the InterAct control to invoke the properties window. InterAct has several properties which can be set. Switch to the Control property page.

Check off the property Tools Palette. In the Grid Properties group, set H Grid Spacing to 30 and V Grid Spacing to 25. Set ScrollBars to Both.

The InterAct Properties sheet.

Close the property page. These properties will now be applied to InterAct. Notice that InterAct is now displaying the grid lines with the specified spacing.

InterAct after the properties have been set.

Programming InterAct

Most applications will want to load their own diagram into InterAct. We will now demonstrate how to dynamically create a diagram in InterAct.

Adding Elements to a Diagram

Add a button to the dialog under the Cancel button. Double-click the new button and set the Caption to Diagram. Close the property window to accept this change.

Invoke ClassWizard by pressing CTRL+W. Select the Member Variables tab page. Select IDC_IDOCTRL1 from the Control IDs list box and press the Add Variable button. On the Add Member Variable dialog box, enter m_ido in the Member variable name edit box. Press the OK button.

Add a member variable to the project in Class Wizard.

Switch to the Message Maps tab page, select IDC_BUTTON1 from the Object IDs list box and BN_CLICKED from the Messages list box. Press the Add Function button.

Add a function to a button in Class Wizard.

Press OK on the Add Member Function dialog. Press Edit Code on the ClassWizard dialog.

```
void CidodemoDlg::OnButton1()
{
m_ido.ResetDiagram();

// add two entities connected with a relation
m_ido.AddEntityFromClass(1, , Generic Rectangle,
   First Node, 100, 100, 50, 50);
m_ido.AddEntityFromClass(2, , Generic Rectangle,
   Last Node, 200, 100, 50, 50);
m_ido.AddRelationFromClass(0, Link, Generic Relation,
        , 1, , 2, );

// disable InterActs popup menu for entities
m_ido.SetPopupMenu(IDOMENU_ENTITY, FALSE);
}
```

Code for adding objects to a diagram.

At the top of the file, add a helper header file provided with InterAct to define the contants used by InterAct.

```
#include "idodef.h"                    // InterAct contstant header file
```

The button Command1 will clear the diagram by calling the InterActs ResetDiagram method and then add two entities and a relation connecting these entities.

The method AddEntityFromClass takes several parameters. The first two allow you to uniquely identify the entity we are about to add either with a number or a string. In this example we are just supplying a number. Next is an entity class name. The class Generic Rectangle is a default class available with a new InterAct control. Classes are simply an easy way to specify how an entity will look and behave without having to supply all these details when you create the entity. Next, text which will be displayed by the entity is passed. Finally, the last four digits are the coordinates of the new

entity.

The method AddRelationFromClass also takes several parameters. Like the AddEntityFromClass method, the first two allow you to uniquely identify the relation we are about to add either with a number or a string. In this example we are just supplying a string. Next is a relation class name. The class Generic Relation is a default class available with a new InterAct control. Next, the ID or name of a source and destination entity are needed. Note that we are passing the IDs assigned to the entities created with the AddEntityFromClass methods.

Compile and run the program by selecting the menu items Build | Build idodemo.exe and Build | Execute idodemo.exe. Press the button Diagram. A simple diagram of two entities connected by a relation will be created. One entity will contain the text First Node and the other will contain the text Last Node. You can use the mouse and keyboard to move and resize the entities. Stop the program by clicking the OK button and return to the programming environment.

InterAct at runtime with a diagram.

Responding to Events

Next we are going to respond to a click event to query the diagram. We will determine which entity was clicked and retrieve the text of that entity. We will then display that text to the user in a message box.

Add an Event Procedure

Invoke ClassWizard again by pressing CTRL+W. Select the Message Maps tab page. Select IDC_IDOCTRL1 from the Object IDs list box and ClickEntity from the Messages list box. Press the Add Function button..

Adding an event with Class Wizard.

Press OK on the Add Member Function dialog. Press Edit Code on the ClassWizard dialog.

```
void CIdodemoDlg::OnClickEntityIdoctrl1(long MouseStatus)

{

// TODO: Add your control notification handler code here



CEntity Entity;

CString Text;



if ( MouseStatus == IDO_RIGHTCLICK )

{

 Entity = m_ido.GetNotifyEntity();

 Text = Entity.GetText();



 MessageBox((LPSTR)Text.GetBuffer(0), Entity Text, MB_OK);

}

}
```

Code to respond to a click event.

At the top of the file, add another header for the Entity object.

```
#include "entity.h"                    // Entity object header
```

Recompile and run the program. Click the Diagram button. The program output will appear the same but by right-clicking on an entity, a message box with the entitys text will appear.

InterAct when the entity First Node is clicked.

Conclusion

This concludes the Visual C++ tutorial of InterAct. We have shown just some of the basic features and capabilities of InterAct. When using InterAct in Visual C++ you will use these same four techniques:

- Insert InterAct from the Component Gallery into your application.
- Place InterAct on a form and set properties for InterAct.
- Add code to invoke methods and properties of InterAct to create or alter diagrams.
- Respond to events from InterAct to perform special actions.

Introduction

This tutorial will show how to use InterAct using Visual Basic 4.0 32-bit. This tutorial will show how you can:

- add InterAct to a Visual Basic project
- add InterAct to a form and set properties on it
- programmatically add entities and relations to a diagram
- respond to mouse clicks within the InterAct environment

If you want a tutorial regarding how to use InterAct using the mouse and keyboard, see Chapter 1: A Guided Tour of InterAct.

Installing InterAct in Visual Basic

Adding InterAct to the default Visual Basic Project:

To add InterAct to the default project, you must first open the default project. Choose the File | Open Project menu item to invoke the Open Project dialog. Select the project Auto32ld.vbp from your Visual Basic directory and press the OK button. Now when you create a new project as described under Create a New Project, InterAct will automatically be available to that project. Just follow the steps beginning at Adding InterAct to a Form.

Create a New Project:

Start a new project by choosing the menu File | New Project. To use InterAct in Visual Basic, InterAct must be added to your project. The ActiveX can either be added to each project that uses InterAct or added to the default project. By adding the ActiveX to the default project, the InterAct ActiveX is automatically loaded whenever you start Visual Basic or create a new project.

Adding InterAct to a Visual Basic Project:

If InterAct is not part of the default project and you wish to add it to an existing or a new

project, choose the Tools | Custom Controls menu item to invoke the Custom Controls dialog to add the InterAct ActiveX file.

Adding InterAct in Visual Basic

Select ProtoView InterAct Control from the Available Controls list box and press the OK button. If ProtoView InterAct Control is not visible in the list, use the Browse button to locate and register the PVIDO32.OCX control.

The InterAct icon, , is now displayed in the Toolbox. InterAct behaves like any other control. It can be dragged from the Toolbox to a form, its properties can be retrieved or set, and event procedures can be created.

Adding InterAct to a Form

Select the InterAct icon, , from the Toolbox and draw the InterAct control on the form. Resize InterAct so that it takes up most of the form.

InterAct during design mode.

InterAct when drawn will be a simple rectangle with gridlines drawn at even intervals.

Setting Properties on InterAct:

Right click on InterAct and choose Properties from the floating menu. InterAct has several properties which can be set. Switch to the General property page.

Check off the property Tools Palette. In the Grid Properties group, set H Grid Spacing to 30 and V Grid Spacing to 25. Set ScrollBars to Both.

InterAct Properties sheet

Click the OK button. These properties will now be applied to InterAct. Notice that

InterAct is now displaying the grid lines with the specified spacing.

InterAct after the properties have been set.

Programming InterAct

Most applications will want to create their own diagrams in InterAct. We will demonstrate how to dynamically create a simple diagram in InterAct.

Adding Code to Command1:

Add a button to the form. By default it will be called Command1. Double click on the button to add an event procedure to the command1_click event. Add the following code to the button Command1:

```
Rem clear any objects from the diagram
ido1.ResetDiagram

Rem add two entities and a relation
ido1.AddEntityFromClass 1, , Generic Rectangle, First Node, 100, 100, 50, 50

ido1.AddEntityFromClass 2, , Generic Rectangle, Last Node, 200, 100, 50, 50

ido1.AddRelationFromClass 0, Link, Generic Relation, , 1, , 2,

Rem disable the InterActs default menu for entity objects
ido1.PopupMenu (IDOMENU_ENTITY) = FALSE
```

The button Command1 will clear the diagram by calling the InterActs ResetDiagram method and then add two entities and a relation connecting these entities.

The method AddEntityFromClass takes several parameters. The first two allow you to uniquely identify the entity we are about to add either with a number or a string. In this example we are just supplying a number. Next is an entity class name. The class Generic Rectangle is a default class available with a new InterAct control. Classes are simply an easy way to specify how an entity will look and behave without having to supply all these details when you create the entity. Next, text which will be displayed by the entity is passed. Finally, the last four digits are the coordinates of the new entity.

The method AddRelationFromClass also takes several parameters. Like the AddEntityFromClass method, the first two allow you to uniquely identify the relation we are about to add either with a number or a string. In this example we are just supplying a string. Next is a relation class name. The class Generic Relation is a default class available with a new InterAct control. Next, the ID or name of a source and destination entity are needed. Note that we are passing the IDs assigned to the entities created with the AddEntityFromClass methods.

Run the program. Press the button Command1. A simple diagram of two entities

connected by a relation will be created. One entity will contain the text First Node and the other will contain the text Last Node. You can use the mouse and keyboard to move and resize the entities. Stop the program and return to the programming environment.

InterAct after the Command1 button is clicked.

Responding to Events

Invoke the Code Window by left double-clicking on InterAct. Choose ClickEntity from the Procedure listbox.

Add the following code to the InterActs ClickEntity event:

```
Dim E1 As Entity

Dim Text As String



If (MouseStatus = IDO_RIGHTCLICK) Then



 Rem retrieve the entity which was just clicked

 Set E1 = ido1.GetNotifyEntity



 Rem get the text from the clicked entity

 Text = E1.Text



 Rem display the text

 MsgBox (Text)



End If
```

Run the program now. Press the button Command1. Our simple diagram of two entities connected by a relation will be created. Now right click the entity marked First Node. The program output will respond with a message with the text First Node.

InterAct when the entity First Node is clicked.

Conclusion

This concludes the Visual Basic tutorial of InterAct. We have shown just some of the basic features and capabilities of InterAct. When using InterAct in Visual Basic you will use these same three techniques:

- Add InterAct to a specific Visual Basic project or the default Visual Basic project.
- Place InterAct on a form and set properties for InterAct.
- Add code to invoke methods and properties on InterAct to create or alter diagrams.
- Respond to events from InterAct to perform special actions.

Introduction

This tutorial is meant to show how to use the InterAct DLL as a custom control using ProtoGen+. This tutorial can be used by any programmer wishing to use C or C++ and InterAct and does not rely on any capabilities provided solely by ProtoGen+.

This tutorial will show how you can:

- add InterAct as a component to ViewPaints control palette
- add InterAct to a dialog and set styles on InterAct
- programmatically add entities and relations to a diagram
- respond to mouse clicks within the InterAct environment

If you want a tutorial regarding how to use InterAct using the mouse and keyboard, see Chapter 1: A Guided Tour of InterAct.

Adding InterAct to an Application

Creating a Generic Program

Start a new project by choosing File | New Project. In the Application Setup Assistant, enter idodemo as a Project Name and choose a Directory to store the project. Next, choose the Code Generation Language your computer supports. This tutorial will select 16bit Ansi. In the Options group, make sure only Standard Menu is checked. Click Create to accept these options.

The ProtoGen+ Application Setup Assistant dialog.

Adding the InterAct header and library Files

When you compile an application which contains InterAct you must include the InterActs header and import library. From the PG+ main menu, select Project | Manage Files. This takes you to the Manage Application Files dialog.

The ProtoGen+ Manage Application Files dialog.

Select the entry Include Files in the File Type listbox. Navigate to the directory which contains the include file pvido.h. Highlight this file and click the Select button.

Select the entry Library Files in the File Type listbox. Navigate to the directory which contains the include file pvido.lib. Highlight this file and click the Select button.

Click Done. All the necessary support files for InterAct have now been added.

Note: In future editions of ProtoGen+, the Workbench will automatically detect the IDO and add these support files.

We will now create a main window dialog which will contain InterAct. Right click on the main window. Choose Select Main Window from the floating menu. Click the New Dialog button in the Main Window Dialog screen. Double click the entry MainWindowDlg in the Select New Dialog Resource list. We are now in ViewPaint.

Installing InterAct in ProtoViews ViewPaint

If you have installed InterAct in ViewPaint prior to this tutorial, you do not need to perform these steps again. You need only install InterAct with ViewPaint once for it to be available for all new projects. You can skip to the step Adding InterAct to a Dialog.

If you havent installed InterAct in ViewPaint yet, select the Add and Remove Custom Controls menu item from the Option menu. This presents the Manage Custom Controls dialog.

Manage Custom Controls dialog

If you are using the 16-bit ProtoGen+ Workbench, navigate to the directory that contains the PVIDO.DLL file in the Directories list box and select the PVIDO.DLL file in the Files list box. Click on the Include button. The PVIDO.DLL will show up in the Current List. Click on the OK button to close the dialog. ViewPaint now knows where to find the 16-bit InterAct dynamic link library and "PVIDO" will show up in the list of

available controls.

If you are using the 32-bit ProtoGen+ Workbench, navigate to the directory that contains the PVIDO32.DLL file in the Directories list box and select the PVIDO32.DLL file in the Files list box. Click on the Include button. The PVIDO32.DLL will show up in the Current List. Click on the OK button to close the dialog. ViewPaint now knows where to find the 32-bit InterAct dynamic link library and "PVIDO32" will show up in the list of available controls.

Customizing the Tools Palette:

To include the InterAct icon on the ViewPaint tools palette, right-click on an existing or empty control icon. This presents the Customize Workbench Toolbox dialog.

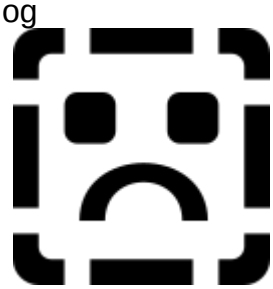
Customizing the ViewPaint tools palette.

Depending on whether you are using the 16- or 32-bit ProtoGen+ Workbench, set the Tool Class to PVIDO or PVIDO32, respectively, and set Button Graphic to PVIDO.BMP.

Click the Save button to permanently keep the InterAct control on ViewPaints tools palette. Click the OK button to close the Customize Workbench Toolbox dialog.

Note: You do not need to include the InterAct icon on the tool palette. Once it is installed into ViewPaint, it is always accessible through the tool palettes Control List.

Adding InterAct to a Dialog



Select the InterAct icon, , from the Tools Palette and draw the InterAct control on the dialog. Resize InterAct so that it takes up most of the dialog.

Designing InterAct in ViewPaint.

When drawn in ViewPaint, InterAct will be a simple rectangle with gridlines drawn at even intervals.

Setting Styles on InterAct

Right-click on InterAct. This takes you to the InterAct Control Styles dialog. Here you can apply standard Windows styles to the InterAct control. Make sure the following styles are applied:

· Visible

- Border
- Tabstop
- Horizontal Scroll
- Vertical Scroll

Click OK to return to ViewPaint. Add a button to the dialog. By default it will be called IDC_BUTTON1 and have the text Button. Right-click the button to display its styles dialog. Change the Field Text to Build Diagram. Click OK.

Select File | Save and Exit to save this dialog. Enter MainView in the field Enter Resource Name and click OK. Select the file idodemo.dlg as the dialog resource file and click OK.

We are now in the Main Window Dialog screen. Make sure MainView is selected and click Select. The main window of our application has now been selected and contains InterAct.

Programming InterAct

Most applications will want to create and manipulate their own diagrams in InterAct.

We will demonstrate how to dynamically create a simple diagram in InterAct. First,

generate code for our main window dialog by clicking the Express Generate button

on the ProtoGen+ top toolbar.

Right-click on the main window background and select Main Window Messages from

the floating menu. Highlight WM_INITDIALOG in the Selected Messages list and click

Edit Code. We will be brought into the source code at the correct location.

Add code to WM_INITDIALOG:

Add the following code to the dialog initialization routine.

```
// hide the tools palette
idoSetToolsPalette(GetDlgItem(hWnd, IDC_IDO1), FALSE);

// make editable
idoSetEditMode(GetDlgItem(hWnd, IDC_IDO1), TRUE);

// set grid width
idoSetGridWidth(GetDlgItem(hWnd, IDC_IDO1), 25);

// set grid height
idoSetGridHeight(GetDlgItem(hWnd, IDC_IDO1), 20);

// hide InterAct default menus when items are right-clicked
idoSetPopupMenu(GetDlgItem(hWnd, IDC_IDO1),
   IDOMENU_ENTITY, FALSE);
```

This code disables use of the InterActs built-in tools palette and menu for entity objects and sets the width and height of the grid lines in InterAct. The one parameter common to all these functions is a window handle which uniquely identifies InterAct. You can retrieve the window handle by using the Windows API function GetDlgItem( ). Return to the Workbench.

Adding code to a button:

Right-click on the button and choose Edit Button Code from the floating menu to respond to the click event. Add the following code to the button IDC_BUTTON1 notification response:

```
// reset the diagram
idoResetDiagram(GetDlgItem(hWnd, IDC_IDO1));

// add an entity identified as 1
idoAddEntityFromClass(GetDlgItem(hWnd, IDC_IDO1),
  1, , Generic Rectangle, First Node, 100, 100, 50, 50);

// add an entity identified as 2
idoAddEntityFromClass(GetDlgItem(hWnd, IDC_IDO1),
  2, , Generic Rectangle, Last Node, 200, 100, 50, 50);

// connect entity 1 to entity2 with a relation link
idoAddRelationFromClass(GetDlgItem(hWnd, IDC_IDO1),
  0, Link, Generic Relation, , 1, , 2, );
```

The button Build Diagram will clear the diagram by calling InterActs ResetDiagram method. The only parameter is a window handle which uniquely identifies InterAct. You can retrieve the window handle by using the Windows API function GetDlgItem( ). Next we add two entities and a relation connecting these entities

The method AddEntityFromClass takes several parameters. The first is the window handle of InterAct. The next two allow you to uniquely identify the entity we are about to add either with a number or a string. In this example we are just supplying a number. Next is an entity class name. The class Generic Rectangle is a default class available with a new InterAct control. Classes are simply an easy way to specify how an entity will look and behave without having to supply all these details when you create the entity. Next, text which will be displayed by the entity is passed. Finally, the last four digits are the upper left corner and width and height of the new entity.

The method AddRelationFromClass also takes several parameters. Like the AddEntityFromClass method, the first is the window handle of InterAct and the next two allow you to uniquely identify the relation we are about to add either with a number or a string. In this example we are just supplying a string. Next is a relation class name.

The class Generic Relation is a default class available with a new InterAct control.

Next is text to be displayed with the relation. We are not supplying any. Next, the ID or name of a source and destination entity are needed. Note that we are passing the IDs assigned to the entities created with the AddEntityFromClass methods.

Run the application:

Compile and execute the program by clicking the Generate and Compile Source button

on the PG+ top toolbar. Press the button Build Diagram. A simple diagram of two entities connected by a relation will be created. One entity will contain the text First Node and the other will contain the text Last Node. You can use the mouse and keyboard to move and resize the entities. Exit the program and return to the ProtoGen+ programming environment.

InterAct after the Add button is clicked.

Responding to Events

InterAct will send notification of key events as both notifications and messages to its parent dialog. We will respond to the message sent when an entity is clicked. Right-click on the background of the main window. Select Main Window Messages from the floating menu. Select Message Category | Custom Messages. Select the message IDOM_N_CLICK_ENTITY from the Custom Messages list. This message will now appear in the Selected Messages list. Highlight the message IDOM_N_CLICK_ENTITY in the Selected Messages list and click the button Edit Code. We will be brought into the source code at the correct location.

Note: If this message does not appear in your list, quit your PG+ Workbench and copy the file MSG.TAB from the InterAct directory to your PG+ directory and restart PG+.

```
case IDOM_N_CLICK_ENTITY :

            //Regen_IDOM_N_CLICK_ENTITY

            {
                ENTITY entity;


                switch(lParam)
                {
                    case NOTIFY_CLICK_RIGHT :

   // Get Clicked Entity
   // if successful, print the entities text to the screen
```

```
                if(idoGetNotifyEntity(GetDlgItem(hWnd, wParam), &entity))
                  MessageBox(hWnd, idoEntityGetText(&entity),
"Entity Text", MB_OK);
                break;
            }
        }

        //Regen_IDOM_N_CLICK_ENTITY
        break;
```

This code allows us to respond to the mouse click event on an entity. For further information, we evaluate the contents of the lParam, responding only when the entity is right clicked and ignoring all other clicks on entities. In response to this event we wish to display a message box with the text of the clicked entity. For this we call idoGetNotifyEntity. If this function succeeds, information about the entity will be stored in the variable entity. We can then use idoEntityGetText to retrieve the text associated

with that entity.

Run the application:

Compile and execute the program by clicking the Generate and Compile Source button

on the PG+ top toolbar. Press the button Build Diagram. Our simple diagram will be created. Right-click on the background of on of the entities. A message box will appear with the text of that entity.

InterAct when an entity is right clicked.

Conclusion

This concludes the ProtoGen+ tutorial of InterAct. We have shown just some of the basic features and capabilities of InterAct. When using InterAct in ProtoGen+ you will use these same four techniques:

- Add the InterAct header and library files to your project.
- Install InterAct as a custom control in ViewPaint.
- Place InterAct on a dialog and set styles for the it.
- Add code to call functions which invoke methods and properties of InterAct to create or alter diagrams.
- Respond to events from InterAct through notifications and messages to perform special actions.

Introduction

InterAct is available as a 16 and 32-bit DLL or a 32-bit ActiveX. In order to keep the manual manageable and the programming interface consistent, the following reference section can be used by both the DLL and ActiveX. Where appropriate, if differences exist, notes will describe the difference between the DLL and ActiveX.

InterAct is manipulated from a program through invoking methods or setting properties. In addition, InterAct can fire events to its parent, notifying the parent of key events.

A property is usually a value that can be retrieved or set. An example of a property is whether to display grid lines in a diagram. Properties will usually be accessed with Get/Set functions. For the ActiveX, some properties can also be set when designing InterAct in a programming environment, such as Microsofts AppStudio, Visual Basic or Borlands Delphi.

A method is a function that tells InterAct to perform an action. For example, you might call a method telling InterAct to save the current diagram to a file, passing a valid filename and optional path to InterAct. Methods cannot be invoked when designing InterAct in a programming environment, such as Microsofts AppStudio, Visual Basic or

Borlands Delphi, and can only be called at runtime.

InterAct has many properties that affect its appearance and many methods to add, manipulate and remove objects from a diagram.

Identifying Objects

Every object in the diagram needs to be uniquely identified to allow actions to be taken on specific items. There are two ways you can identify an object - by a name or an ID. IDs are stored as a C data type long, and can have a value between 0 and 32000. Names are a character string of up to 99 characters. An object can be identified by both a name and an ID.

When adding a new entity or relation programmatically, you can assign either a name or an ID to the new item. In the first sample below, a new entity is given the ID of 100 and the name Object A. You do not have to supply both a name and an ID. If you supply neither, InterAct will provide a default name and ID.

```
// We have supplied both a name and an ID.
IDO.AddEntityFromClass(100, Object A, Rectangle, , 10, 10, 50, 50);

// We have supplied only an ID.  InterAct will supply a default name.
IDO.AddEntityFromClass(100, , Rectangle, , 10, 10, 50, 50);

// We have supplied only a name.  InterAct will supply a default ID.
IDO.AddEntityFromClass(0, Object A, Rectangle, , 10, 10, 50, 50);

// We have supplied neither a name or an ID.  InterAct will supply a default value for
both.
IDO.AddEntityFromClass(0, , Rectangle, , 10, 10, 50, 50);
```

Add objects using the ActiveX methods.

```
HWND hIDO = GetDlgItem(hWnd, IDC_IDO1);

// We have supplied both a name and an ID.
idoAddEntityFromClass(hIDO, 100, Object A, Rectangle, , 10, 10, 50, 50);

// We have supplied only an ID.  InterAct will supply a default name.
idoAddEntityFromClass(hIDO, 100, , Rectangle, , 10, 10, 50, 50);

// We have supplied only a name.  InterAct will supply a default ID.
idoAddEntityFromClass(hIDO, 0, Object A, Rectangle, , 10, 10, 50, 50);

// We have supplied neither a name or an ID.  InterAct will supply a default value for
both.
idoAddEntityFromClass(hIDO, 0, , Rectangle, , 10, 10, 50, 50);
```

Adding objects using the ANSI C functions.

InterAct Architecture

InterAct is a combination of three distinct objects - the InterAct window itself, and a collection of entities and relations. InterAct has a list of properties and methods which can be invoked, and each entity and relation also has a list of properties and methods which can be invoked. You will always have access to the InterAct object and its list of properties and methods. You must request access to one of the entities or relations from InterAct before interfacing with these objects. For example, if you want to place some text on an entity, you must first get a reference to that entity.

```
ENTITY entity;



if(idoGetEntity(hWndIdo, 25, NULL, &entity))
    idoEntitySetText(&entity, New Text);
if(idoGetEntity(hWndIdo, 0, Object A, &entity)
    idoEntitySetText(&entity, New Text);
```

DLL example.

```
IDO.GetEntity(25, NULL).SetText(New Text);

IDO.GetEntity(0, Object A).SetText(New Text);
```

ActiveX example.

In the DLL example code, a method and a property must be invoked. The method idoGetEntity takes four parameters:

1) A window handle to identify the InterAct object.

2) An ID to uniquely identify an entity. Pass 0 (zero) if you are going to supply a unique name.

3) A string to uniquely identify an entity. Pass NULL if you are supplying a unique ID.

4) The address of an entity object which will receive information about the entity we are selecting.

The InterAct DLL will return a reference to this entity. Next, the property

idoEntitySetText is invoked. This property takes two parameters:

1) The address of an entity reference object.

2) A string to assign to the identified entity.

For the ActiveX, you are really calling a method and property at once. IDO.GetEntity is a method which returns an entity object. This method takes two parameters:

1) An ID to uniquely identify an entity. Pass 0 (zero) if you are going to supply a unique name.

2) A string to uniquely identify an entity. Pass NULL if you are supplying a unique ID.

You can now set a property on the entity object. The Entity object exposes the property Text. Entity.SetText takes a string to assign to the entity. You could also perform this task on the ActiveX in two steps:

```
EntityObject = IDO.GetEntity(25, NULL);

EntityObject.SetText(New Text);
```

Using Fonts

InterAct allows you to set fonts on entities and relations. This font is used to display text for that entity or relation. For both the DLL and ActiveX, a font object is used to describe a font. For the DLL, a structure IDO_FONT is used. IDO_FONT has the following members:

```
char        szFontName
int   cSize
BOOL        bBold
BOOL        bItalic
BOOL        bUnderline
BOOL        bStrikeout
```

With these members you can describe a font to be used by an entity or relation.

For the ActiveX an OLE Font object is used. Each entity or relation object has a font property which you can get or set. The OLE font object has the following properties:

STRING	Name
CURRENCY	Size
BOOL	Bold
BOOL	Italic
BOOL	Underline
BOOL	Strikethrough
short	Weight
short	Charset

With these members you can describe a font to be used by an entity or relation.

Using Pictures

InterAct allows you to display pictures within entities or associate a picture with a class displayed on the tools palette. These pictures must reside on disk as a BMP file. InterAct will store a path to the file and attempt to load the picture using that path. If this fails, InterAct will attempt to open the file anywhere on disk where it can search. The directories seached will include the current directory, the Windows system directory, and any directory in the PATH environment variable.

Programming with the DLL

Support Components

When programming with the DLL, you need to include the header file PVIDO.H and link with the import library PVIDO.LIB if you are using the 16-bit DLL. If you are using the 32-bit DLL you must link with PVIDO32.LIB for Visual C++, PVIDO32B.LIB for Borland C++, or PVIDO32S.LIB for the Symantec compiler.

Function API

Normally a custom control in a DLL like InterAct exposes a single Window Procedure and a list of vendor-defined messages which tell the control to perform a specific action. However, a message-based architecture is limiting, since a message can allow only two parameters of information to be passed when the message is sent. This leads to situations where structures of data are passed when a message is called and not just single values. In order to get around this limitation and avoid the need for cumbersome structures, InterAct also exports a wide range of functions.

This set of functions exposed by the DLL can be called by an application. Just like sending a message, many of these functions will require passing a window handle as

the first parameter to identify InterAct you wish to perform the action on. If an invalid window handle is passed, the invoked function will not have any effect.

You can get a window handle from an existing InterAct control in a dialog by using GetDlgItem(), or saving the result of CreateWindow() if you dynamically create the InterAct control. See your Windows API help file for information about these two functions.

```
HWND hIDO;



// get the window handle to InterAct
hIDO = GetDlgItem(hWnd, IDC_IDO);



// set properties on InterAct, using the window handle as a parameter
idoSetPaletteVisible(hIDO, FALSE);
idoSetGridHeight(hIDO, 25);
idoSetGridWidth(hIDO, 25);
```

ANSI C example of accessing an InterAct object using a Window handle.

When referring to entities or relations, a structure is used instead of a window handle. These are declared as types ENTITY and RELATION, respectively. When setting properties or invoking methods on these objects, you do not need a window handle to InterAct. You need only pass a pointer to these structures as the first parameter.

```
HWND hIDO;

ENTITY entity;




// get the window handle of InterAct

hIDO = GetDlgItem(hWnd, IDC_IDO);



// retrieve a reference to the entity with the ID of one

// return that reference in the variable entity

idoGetEntity(hIDO, 1, NULL, &entity);



// set the text on the entity

// we need pass only the address of the entity object

idoEntitySetText(&entity, Text.);
```

ANSI C example of using an ENTITY structure.

If you really wish to use the messages and not the functions exported from the DLL, you should look at the PVIDO.H header file which defines all the messages and look at CIFACE.C, a source file from the InterAct DLL which shows how the exposed functions are packed into corresponding messages.

Some DLL functions return a long pointer to a string (LPSTR). This string is a temporary pointer returned by the DLL. You must immediately copy the contents of this string into your own allocated character array. You must never copy anything into this internal string array or save this pointer.

Using the DLL in C++

You can use the message and function API in C++ just as you would in ANSI C. The InterAct DLL also supports the MFC and OWL class libraries.

For MFC, include the header file IDOMFC.H. Four classes are defined: CIDO, CEntity, CRelation and CIDORule. For OWL, include the header file IDOOWL.H. Four classes are defined: TIDO, TEntity, Trelation and TIDORule. All four classes are defined inline so you do not need to link with any libraries.

All the standard function APIs are available in the C++ wrappers. Where functions

exist to retrieve an entity or relation object, these functions have been overloaded to accept or return CEntity and CRelation objects, or TEntity and TRelation objects.

Some functions which accept long pointers to strings have been overloaded to accept CString objects.

For functions which require a hWnd to InterAct, drop the prefix ido from the function and drop the required window handle parameter. For entity or relation functions, drop the prefix idoEntity and idoRelation from the function and drop the required pointer to a ENTITY or RELATION object. See the sample below:

```
ENTITY entity;

HWND hIDO;




// get a window handle to InterAct

hIDO = GetDlgItem(hWnd, IDC_IDO);



// get a reference to an entity

idoGetEntity(hIDO, 1, NULL, &entity);



// set the text on the entity

idoEntitySetText(&entity, Text);
```

ANSI C example using the Message API.

Now the same sample using the MFC wrappers.

```
CEntity * pEntity;

CIDO *pIDO;



// create a new instance of an entity C++ object

pEntity = new CEntity();



// get a reference to the InterAct object

pIDO = (CIDO *)GetDlgItem(IDC_IDO);



// retrieve a reference to an entity object from InterAct

pIDO->GetEntity(1, NULL, pEntity);



// set the text on the entity object

pEntity->SetText(Text);
```

```
// delete the entity C++ object

delete pEntity;
```

MFC example using the C++ classes.

What do I need to ship with my finished Application?

If you are developing a 16-bit application, you need to ship PVIDO.DLL. If you are planning to display the property pages for InterAct, entities or relation, you also need to ship PGHT.DLL.

If you are developing a 32-bit application, you need to ship PVIDO32.DLL. If you are planning to display the property pages for InterAct, entities or relation, you also need to ship PGHT32.DLL.

Programming with the OCX

Support Components

When using the ActiveX, the programming environment, such as Microsofts Developer Studio, Visual Basic or Borlands Delphi, will be able to query the ActiveX and create a programming interface for the ActiveX automatically. Visual C++ will generate a class header and source file, while Delphi will generate a Pascal unit file, for each object defined in the InterAct ActiveX. These files will have functions for each property and method defined for each object.

Visual Basic is able to read and store all the constants defined by the InterAct control. Other environments, such as Visual C++ and Delphi, do not. For Visual C++ InterAct provides an additional header file called IDODEF.H with these defines. For Delphi, IDODEF.PAS is included. Be sure to include these header files to define these constants.

Using the ActiveX in C++

At the time of printing, Visual C++ does not declare constants used by InterAct in a header file. These constants are used throughout InterActs properties and methods

and must be declared in order to be used. InterAct therefore supplies a header file called IDODEF.H. This header file contains the declarations for these constants and must be included in the C++ application.

Using the ActiveX in Delphi

At the time of printing, Delphi does not declare constants used by InterAct in a pascal unit file. These constants are used throughout InterActs properties and methods and must be declared in order to be used. InterAct therefore supplies a header file called IDODEF.PAS. This header file contains the declarations for these constants and must be included in the C++ application.

What do I need to ship with my finished Application?

If you are developing a 16-bit application, you need to ship PVIDO.OCX. If you are planning to display the property pages for InterAct, entities or relation, you also need to ship PGHT.DLL.

If you are developing a 32-bit application, you need to ship PVIDO32.OCX. If you are planning to display the property pages for InterAct, entities or relation, you also need to ship PGHT32.DLL.

Introduction

InterAct will fire events to its parent notifying it of key events. These events allow you to provide sophisticated responses to InterAct. Events which a programa he