00c8b9d8-0

COLLABORATORS

	TITLE : 00c8b9d8-0	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		December 7, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

00c8b9d8-0

1.1 "

```
   $RCSfile: Oberon0.doc $
Description: Documentation for the Oberon-A version of Mössenböck's
             Oberon0 System.

 Created by: fjc (Frank Copeland)
  $Revision: 1.1 $
    $Author: fjc $
      $Date: 1995/07/02 22:57:26 $
```

The Oberon0 System is a very much simplified version of the ETH Oberon System, described in Mössenböck's book "Object-Oriented Programming in Oberon-2". It is intended as a case-study to demonstrate the principles of object-oriented programming discussed in the book. As such it is neither complete, nor very useful as a tool. It is, however, a very good basis for exploration and experimentation in both OOP and Oberon-2.

```
      ~Distribution~~~~~~  Legalistic mumbo-jumbo.
      ~Requirements~~~~~~  Extremely minimal.
      ~Installation~~~~~~  Couldn't be simpler.

      Running Oberon0 from...
      ~Shell~~~~~~~~~~~~~  ...~the~Shell~~~~~
      ~Workbench~~~~~~~~~  ...~the~Workbench~
      ~FPE~~~~~~~~~~~~~~~  ...~FPE~~~~~~~~~~~

      ~OOP~in~Oberon-2~~~  The book.
      ~What~is~Oberon0?~~  The program.
      ~Using~Oberon0~~~~~  The fluffy toy.
      ~Extending~Oberon0~  The T-shirt.

      ~Known~bugs~~~~~~~~  Shock! Horror!
```

Sorry, it was one of those nights. :-)

1.2 Distribution and copyright issues

1.3 What do I need to run Oberon0?

The Oberon0 System requires AmigaOS 2.04 or greater. reqtools.library is required for the screen mode and font requesters. If it is not available, the default public screen is cloned instead.

1.4 Installing Oberon0

The Oberon0 System is provided in source code form only, and must be compiled and linked before it can be used. An AmigaDOS script file named MakeOberon0 can be found in the Oberon0 directory. When executed, this script file will compile all the modules and link them together to create a program named RunOberon0.

To execute MakeOberon0 from the Shell:

- Change directory to OBERON-A:Examples/Oberon0.
- Run the command 'Execute MakeOberon0'.

To execute MakeOberon0 from the Workbench:

- Open the OBERON-A:Examples/Oberon0 drawer.
- Double-click the MakeOberon0 icon.

1.5 Running Oberon0 from the Shell

If you have not already done so, install Oberon0 (see Installation). To start the Oberon0 System, change directory to

```
OBERON-A:Examples/Oberon0
```

and run the program RunOberon0.

1.6 Running Oberon0 from the Workbench

If you have not already done so, install Oberon0 (see Installation). To start the Oberon0 System, open the Oberon0 drawer in the OBERON-A:Examples drawer and double-click the RunOberon0 icon.

1.7 Running Oberon0 from FPE

If you have not already done so, install Oberon0 (see Installation). To start the Oberon0 System, change directory to OBERON-A:Examples/Oberon0 and select RunOberon0 as the the project. Make sure the default settings have been loaded, then click on the 'Run' button.

1.8 Object-oriented programming in Oberon-2

"Object-Oriented Programming in Oberon-2" by Hanspeter Mössenböck is published in English and German editions by Springer-Verlag. While it is not strictly necessary to have the book in order to use the Oberon0 System, it is recommended for its coverage of object-oriented programming techniques in Oberon-2.

1.9 What is Oberon0?

The Oberon0 System is based on the ETH Oberon System, and the original implementation was actually built on top of it. It is basically a pair of editors, one for text and one for graphics, built on a tiled viewer system. It demonstrates a number of techniques in object-oriented programming, including the creation of a framework of classes, extending abstract and semi-finished classes, models, views and controllers, message passing and handling user input.

For a full description of the Oberon0 System, see Chapter 11 of Mössenböck's Object-oriented~Programming~in~Oberon-2.

1.10 Using Oberon0

Mouse control

The Oberon0 System has similar conventions to the ETH Oberon System, in particular the use of a three-button mouse. If you have a three-button mouse attached to your Amiga, all three buttons are recognised and used. If you have the standard two-button mouse, the missing middle button is simulated by holding down the left Alt key while clicking the left mouse button. Whenever the documentation mentions the middle mouse button, it can be assumed to be referring to the left-Alt-LMB substitution as well. There are no standard Amiga menus in the Oberon0 System, and the right mouse button is trapped and interpreted by the Oberon0 System in a non-standard way. The Oberon0 screen can be dragged, scrolled and depth-arranged as normal.

Settings

When Oberon0 is started, the user is presented with a screen mode requester and a font requester. The input to the screen mode requester determines the features of the custom screen opened for the Oberon0 System. If it is cancelled the default public screen is cloned. The font selected becomes the default font. The requesters only appear if reqtools.library is installed on your system. If not, the default public screen is cloned.

Viewers

Oberon0 displays output in one or more Viewers, which are rectangular regions on the screen. All Oberon0 viewers are displayed in a single Amiga backdrop window on the screen opened for Oberon0. All viewers are the same width as the screen, and are arranged vertically in a single column, or track. Each viewer has a menu bar which contains its name and a few standard commands. A viewer can be sized vertically by dragging its menu bar with the left mouse button. A command in the menu bar can be executed by clicking it with the middle mouse button. There are three standard menu commands:

```
  Viewers0.Close -- Closes the viewer and extends any remaining viewers
                    into the vacated space.

  Viewers0.Copy  -- Opens a copy of the viewer, containing the same
                    data as the original.

  Edit0.Store    -- Stores the current contents of the viewer in a file
                    with the same name.
```

Exiting Oberon0

Pressing the Esc key will always cause Oberon0 to halt, even if no viewers are open to receive user input. If all viewers have been closed, pressing the Esc key is the only valid user input.

Editing Text

Text is edited in a text viewer. A text viewer called 'LOG' is automatically opened when the Oberon0 System is started. The main part of a text viewer is divided into a scroll bar along the left edge, and a text area.

Mouse clicks have the following effects:

Mouse button	In text area	In scroll bar
Left	Set caret	Scroll forward
Middle	Execute command	Scroll absolutely
Right	Select + Left = Delete selection + Middle = Copy selection to caret	Scroll to beginning

The caret is a triangular mark that indicates the insertion point, where any characters typed by the user will be inserted in the text. The caret is placed by clicking with the left mouse button. A single character to the left of the caret is deleted by pressing the 'Del' key. The arrow keys do not have any effect on the caret.

Selected text is displayed in reverse video. The delete and copy operations are activated by 'interclicks', where two mouse buttons are pressed together. The right mouse button is used the mark the affected text, then the left or middle button is clicked while the right button is held down. The operation is executed when both buttons are released.

A command takes the form:

```
<module>.<command> {parameter(s)}.
```

For example:

```
Edit0.Open MyFile
```

Clicking on the module or command name with the middle button will cause the named procedure to be executed. Any text on the same line after the command may be treated as a parameter by the command.

The following commands affect text viewers:

```
Viewers0.Close
Viewers0.Copy
Edit0.Open <name>
Edit0.Store
Edit0.ChangeFont <name>.font/<size>
FoldElems0.Insert
GraphicElems0.Insert
```

Text Elements

Elements are objects that can be inserted in a text. Two types of elements are currently implemented: FoldElems and GraphicElems.

A FoldElem is a piece of text that can be hidden from view if desired. A FoldElem is created by selecting a piece of text with the right mouse button and executing the command FoldElems0.Insert. Two marks are inserted in the text before and after the selected text. The marks used in this implementation are the double-chevron characters: '«' and '»'. On my standard US keyboard these are the alt-(and alt-) characters. The left-chevron marks the start of the FoldElem and the right-chevron marks the end. Clicking on the left chevron with the middle button will hide the text and replace it with another mark: '»«'. Clicking on the *right* chevron will reveal the hidden text again.

For example:

```
«This is the text revealed.»
^
```

```
  |
  +-- click here to hide the text.

  »«
  ^
  |
  +-- click here to reveal the text.
```

The text inside the FoldElem can be edited normally. The FoldElem can be deleted without deleting the text by deleting the two marks. Note that not all fonts contain the glyphs for the double-chevrons, which may make it difficult to detect the presence of FoldElems.

A GraphicElem is a graphical figure that can be displayed along with the text. A GraphicElem is created by placing the caret with the left mouse button and executing the command GraphicElems0.Insert. An empty GraphicElem will be inserted at the caret. It can then be edited by clicking on it with the middle mouse button. This will open a viewer which will allow the graphic to be edited.

Editing Graphics

At present the only graphical figures implemented are Rectangles, which are implemented in the module Rectangles0. In order to draw a Rectangle, you must first execute the command Rectangles0.Set. A Rectangle can then be drawn in a graphics viewer by dragging the left mouse button. The rectangle is drawn between the points at which the button is pressed and released.

At present there is no way to edit or delete a Rectangle once it is drawn.

The command GraphicElems0.Update must be executed to save any changes in the figure, and redraw the figure in the text in which it is embedded.

1.11 Extending Oberon0

It should take you about five minutes to realise that Oberon0 is basically a toy. What you also need to realise is that it is an educational toy. Its usefulness lies not in what you can do with it, but in what it can teach you.

I have deliberately resisted any urge I might have had to 'improve' Oberon0. That would defeat the whole purpose. Similarly, I recommend that you don't bother sharing any improvements you may make; let each person discover it for themselves. You will get the most benefit from exploring, experimenting and improving the system for yourself.

So I will just limit myself to suggesting a few things you may find it worthwhile to look into.

- Modify the code for deleting text so that the rest of the viewer

isn't simply redrawn. See if you can work in a block move somehow to preserve the existing lines that don't change.

- Add some more shapes for the graphics editor. Arcs, ellipses, Bezier curves, polygons, etc.

- Add the ability to move, edit and delete objects in the graphics editor.

See also Adding~Modules and the List~of~Modules.

1.12 Known bugs

- If you edit a text and save it, a temporary file containing the old text is left in the current directory. This is a problem with module Files.

- If the font you choose as the default does not contain the double chevron glyphs ("«" and "»"), then you won't be able to see FoldElems. They are still there, but a bit difficult to work with :-).

- If you delete the character marking the end of a FoldElem, then try to expand or contract it, the Guru will come visiting.

1.13 Adding Modules

If you wish to extend the Oberon0 System by adding a module to it, you must do two things: add the module to the import list of the main module RunOberon0, and register the module and its types and commands with module Kernel.

To register the module, types and commands, you use the RegisterModule, RegisterType and RegisterCommand procedures from module Kernel. The necessary statements should be added to the Register procedure in the RunOberon0 module. See RunOberon0.mod.

1.14 List of Modules

The Oberon0 System consists of the following modules:

```
OS~~~~~~~~~~~~~  Hides the details of the operating system.
Viewers0~~~~~~~  The basic Viewer class.
Shapes0~~~~~~~~  The base class for all graphical shapes.
AsciiTexts~~~~~  Simple unformatted ASCII text.
GraphicFrames0~  Displaying and editing graphics.
Texts0~~~~~~~~~  Texts with formatting and embedded elements.
Rectangles0~~~~  The Rectangle class.
TextFrames0~~~~  Displaying and editing texts.
FoldElems0~~~~~  Embedded elements for text folding (hiding).
```

```
GraphicElems0~~  Embedded elements for graphics.
Oberon0~~~~~~~~  The main event loop.
io~~~~~~~~~~~~~  Simple formatted text I/O.
Edit0~~~~~~~~~~  A toolkit for editing text.
Draw0~~~~~~~~~~  A toolkit for editing graphics.
RunOberon0~~~~~  The main module.

Some supporting modules are also required:

AmigaSupport~~~  Interface to the Amiga operating system.
Display~~~~~~~~  The Oberon0 display.
Fonts~~~~~~~~~~  The Project Oberon Fonts module.
InputPO~~~~~~~~  The Project Oberon Input module.
Types~~~~~~~~~~  The Oberon System Types module.
Modules~~~~~~~~  The Project Oberon Modules module.
```