# Oberon Pi Draw User Guide  (PDF version)

Draw is a vector graphics editor. It supports the following graphic elements:

- Lines
- Rectangles
- Circles
- Ellipses
- Captions

It also supports *macros*, which are small drawings that are saved in a library, and then inserted as elements into a graphics document. Macros let you extend the set of graphic elements available to include user-defined custom elements.

---

**OPENING**

To open a graphics document for editing, execute the command Draw.Open. For example:

```
Draw.Open DrawDemo.Graph
```

The command parameter ("DrawDemo.Graph" in this case) specifies the file name of the document.

If a file with that name already exists, then the document stored in that file opens in a new window.

If the specified file does *not* exist, a new empty window opens, with the window title set to the command parameter:

```
Draw.Open Xyzzy.Graphic
```

---

The command parameter for Draw.Open can be a text selection instead of a file name. For example:

```
Draw.Open ^
```

When the file name parameter is a caret character "^", the command uses the current selection as the file name of the document to open.

NOTE – The caret character is commonly known as the "hat" character. It appears on the Pi keyboard as the Shift-"6" key.

---

**COMMANDS**

The graphics editor includes several types of commands:

- Mouse commands
- Menu bar commands
- Command window commands
- Keyboard commands

---

The *mouse* commands in the graphics editor are similar to the standard Oberon mouse commands, but have been redefined to perform the most common draw operations:

- Left set cursors, scroll graphics window
- Middle copy or move graphic elements
- Right select graphic elements

The mouse is also used to draw vertical and horizontal lines in a graphics document.

---

The *menu bar* commands in the graphics editor are used to perform the following operations:

- Manage graphics windows
- Show and hide the drawing grid
- Delete graphic elements
- Restore the graphics document display
- Save a graphics document

---

The *command window* commands in the graphics editor are used to perform the following operations:

- Draw graphic elements
- Change an element's line width or font size
- Insert macros into a graphics document
- Manage macro libraries

To open the graphics editor command window, execute the command "System.Open Draw.Cmd”.

---

The *keyboard* commands are used to perform the following operations:

- Move graphic elements by one pixel
- Delete graphic elements (keyboard shortcut)

---

**SYMBOLS**

The graphics editor displays the following application-specific symbols:

- Draw pointer
- Draw cursor

These are similar to the standard Oberon pointer symbols, but have been redefined to work with graphics elements.

---

The *draw pointer* is an X-shaped pointer symbol. In the graphics editor it is mainly used to set the draw cursor at specific locations in a graphics window. It is also used to select graphics elements.

---

The *draw cursor* is a plus-shaped pointer symbol. It is used to specify the size and position of new graphics elements. It is also used to set the current graphics window as the focus window.

NOTE – In the graphics editor, multiple draw cursors can appear at the same time in a graphics window.

---

**DRAWING**

The graphics editor supports the following operations for *drawing* graphic elements:

- Draw line
- Draw rectangle
- Draw circle
- Draw ellipse
- Draw caption
- Scroll graphics window

---

Most graphic elements are drawn in two steps:

1) Set one or more draw cursors in a graphics window to specify the size and position of the new element.

2) Execute the text command that draws the element.

When more than one draw cursor must be set, the first cursor is set by left-clicking the mouse, while the additional cursors are set by shift-left-clicking the mouse.

NOTE – The Esc key erases all cursors from the document.

---

When an element is first drawn it is automatically selected, and includes a marker which shows its selection status. For details see "SELECTION".

—

**Line**

To draw a *line*, left-click where one end of the line should be, then shift-left-click where the *other* end should be. The two cursors indicate where the line will be drawn.

Next, draw the line by executing the "Curves.MakeLine" command that appears in the Draw command window.

—

**Rectangle**

To draw a *rectangle*, left-click where one corner of the rectangle should be, then shift-left-click where the *opposite* corner should be.

Next, draw the rectangle by executing the "Rectangles.Make" command.

—

**Circle**

To draw a *circle*, left-click where the *center* of the circle should be, then shift-left-click to the left or right of the center, to specify the circle's horizontal radius.

Next, draw the circle by executing the "Curves.MakeCircle" command.

—

**Ellipse**

To draw an *ellipse*, left-click where the *center* of the ellipse should be, then shift-left-click to the left or right to specify the ellipse’s horizontal radius, and finally shift-left-click above or below the center to specify the ellipse’s vertical radius. The *three* cursors indicate where the ellipse will be drawn.

Next, draw the ellipse by executing the "Curves.MakeEllipse" command.

---

**Caption**

To draw a *caption*, left-click where the caption should be. The cursor indicates where the caption will appear.

Next, type the caption text. As you type, the text is highlighted to show it is still editable – you can use the Backspace key.

When you press the Esc key, or set the draw cursor in a different location, the highlighting disappears to show that the caption is completed.

NOTE – A caption can contain only a single line of text.

—

**Vertical or horizontal line**

To quickly draw a *vertical or horizontal line*, first position the draw pointer where one end of the line should be.

Next, hold down the left mouse button and *drag* the mouse vertically or horizontally to where the other end of the line should be.

When you release the mouse button, the line is drawn between the starting and ending mouse positions.

If you did not drag the mouse perfectly vertically or horizontally, the graphics editor automatically adjusts the new line so it is vertical or horizontal.

—

**Scroll window**

To *scroll* a graphics window, first move the draw pointer into the window.

Next, while pressing on the Shift key, press on the *left* mouse button, and drag the draw pointer in the direction and amount you want to scroll the window.

When you release the left mouse button, the contents of the graphics window scroll in the direction and amount you specified with the draw pointer.

NOTE – Scrolling the window erases the draw cursors.

---

| **Mouse** | **Command** |
|---|---|
| Left Click | set draw cursor |
| Shift Left Click | set additional cursor |
| Left Drag | draw quick line |
| Shift Left Drag | scroll graphics window |

---

| **Element** | **Cursors** | **Command** |
|---|---|---|
| Line | end, other end | Curves.MakeLine |
| Rectangle | corner, opposite corner | Rectangles.Make |
| Circle | center, radius | Curves.MakeCircle |
| Ellipse | center, horiz rad, vert rad | Curves.MakeEllipse |
| Caption | start | - |
| Quick Line | - | - |

---

**SELECTION**

The graphics editor supports the following *selection* operations on graphic elements:

- Select element
- Select multiple elements
- Add element to selection
- Deselect elements

---

Graphic elements must be *selected* before they can be edited.

To select a graphic element, right-click on the element's *selection area*, which is a specific part of the element as it appears in the display.

The selection area for an element varies with its type:

| | |
|---|---|
| Line | left end (bottom when vertical) |
| Rectangle | lower-left corner |
| Circle | bottom |
| Ellipse | bottom |
| Caption | anywhere |
| Quick Line | anywhere |

NOTE – The rule "low or left" identifies the selection area for every element type.

---

The selection area for each graphic element is large enough that you can select an element relatively easily, without having to be overly precise about positioning the mouse.

To see this, try using the mouse to explore the size of the selection area around a single element. Use the Esc key to deselect the element each time it gets selected.

---

When a graphic element is selected, it displays a *selection marker* to indicate its selection status.

These markers vary with the element type:

| | |
|---|---|
| Line | small x on left end (or bottom) |
| Rectangle | small block in lower-left corner |
| Circle | small x on bottom |
| Ellipse | small x on bottom |
| Caption | highlight |
| Quick Line | dotted line |

NOTE – Selection markers always appear in an element's selection area.

---

Note that lines and quick lines use different selection markers, even though these elements are normally indistinguishable in a graphics document.

As long as you use the "low or left" rule to select a line, you will always be able to consistently select both line types. But the difference in selection markers can be unexpected.

NOTE – This becomes a non-issue if you conventionally use quick lines for drawing all vertical and horizontal lines, and regular lines only for drawing angled lines.

---

To select *multiple* elements in a graphics document, first position the draw pointer so it is outside the elements you want to select.

Then press on the *right* mouse button, and drag the draw pointer across the document, creating an imaginary box around the elements you want to select.

When you release the right mouse button, all the elements inside the imaginary box become selected.

---

To *add* an element to the current selection, shift-right-click on the selection area of the unselected element.

---

To *deselect* a graphic element, press the Esc key.

---

| **Mouse** | **Command** |
|---|---|
| Right Click | select element |
| Right Drag | select multiple elements |
| Shift Right Click | add element to selection |

---

## EDITING

The graphics editor supports the following *editing* operations on graphic elements:

- Move element
- Move element by one pixel
- Copy element
- Copy element between windows
- Delete element
- Restore window

These operations work with single or multiple elements.

—

### Move element

To *move* an element in a graphics document, first select the element, then move the draw pointer to the center of the element.

Next, press on the *middle* mouse button, and drag the draw pointer to the location where you want to move the element.

When you release the middle mouse button, the selected element moves to the new location in the graphics document.

—

### Move element by one pixel

To *move* an element by one pixel, first select the element, then use the following keyboard commands to perform the movement:

| | |
|---|---|
| Ctrl-I | up |
| Ctrl-J | left |
| Ctrl-K | down |
| Ctrl-L | right |

NOTE – To use the keyboard commands, you must first set the draw cursor in the graphics document. This sets the document window as the *focus* window.

---

**Copy element**

To *copy* an element in a graphics document, use the same command for *moving* an element, but hold down the Shift key before you press the middle mouse button.

The element moves as before, but leaves behind a copy of the element in its original location.

---

To *copy* an element from one graphics document to another, use the same command for *copying* an element, but drag the cursor from one graphics window to another.

NOTE – This also works with split windows.

—

**Copy caption**

To *copy* a caption from a text document to a graphics document, first select the caption text in a text window, then shift-middle-click where you want the caption to appear in the graphics document.

---

**Delete element**

To *delete* an element from a graphics document, select the element, then execute the menu bar command "Draw.Delete".

---

A keyboard shortcut is defined for the Draw.Delete command:

| | |
|---|---|
| Ctrl-D | delete element |

NOTE – To use the keyboard commands, you must first set the draw cursor in the graphics document. This sets the document window as the *focus* window.

—

**Restore window**

The graphics editor does not redraw the contents of a graphics window after every editing command. As a result, some commands can cause parts of a graphics document to be erased.

To *restore* a document in its window, execute the menu bar command "Draw.Restore".

---

| **Mouse** | **Command** |
|---|---|
| Middle Drag | move element |
| Shift Middle Drag | copy element |
| Shift Middle Click | copy selected text to caption |

---

**SAVING**

To save a graphics document, execute the Draw.Save command that appears in the menu bar of the document's window.

The saved document is stored in a file with the same name as the title of the the document window.

---

The Draw.Save command can also be executed from any document that the command name appears in.

However, before you can use the command this way, you must first specify two things:

1) The graphics document to save
2) The file name to save to

The document is specified by marking its window with the *location marker*. This is a star-shaped symbol which can be placed at any location on the display (including inside a window).

To mark a window for saving, move the Oberon mouse pointer inside the window and press the Ctrl-A key. The star symbol then appears on the screen next to the mouse pointer.

The file name to save to is specified as a parameter following the command name. For example:

```
Draw.Save Hello.Graph
```

---

If a file is saved with the same name as a file that already exists, the Draw.Save command automatically renames the existing file to be a *backup file*. This prevents the existing file from being deleted by the newly-saved file.

A backup file is created by adding the file suffix ".Bak" to the name of an existing file. For example:

```
Hello.Graph
Hello.Graph.Bak
```

To preserve a backup file, you must rename it with a different file name, and no ".Bak" suffix. For example:

```
System.RenameFiles Hello.Graph.Bak => Hello2.Graph ~
```

---

Oberon expects you to be mindful when using the file system (especially when saving a file with a different file name).

The Oberon system will *not* notify you when an existing file gets overwritten by a new file. Instead, it simply creates a backup file so you can recover the overwritten file.

But backup files themselves offer you only one chance to recover an accidentally overwritten file. If instead you perform a *second* save on a newly-saved file, this will overwrite the current backup file with a copy of the new file, and the original file will be lost.

NOTE – Whenever you save a graphics file, a message appears in the terminal window, listing the name of the saved file.

---

**PROPERTIES**

The graphics editor supports the following operations for changing the *properties* of certain graphic elements:

- Change line width of a line or rectangle
- Change default line width for lines and rectangles
- Change caption font

Note that the editor does *not* support changing the size of an existing element, or the text in an existing caption.

---

To change the *line width* of a line or rectangle element, first select the element, then execute the command Draw.ChangeWidth. For example:

```
Draw.ChangeWidth 2
```

The command parameter ("2" in this case) specifies the line width in pixels.

The parameter value must be between 1 and 6.

---

To change the *default* line width used for drawing a line or rectangle element, execute the command Draw.SetWidth. For example:

```
Draw.SetWidth 3
```

The command parameter ("3" in this case) specifies the default line width in pixels.

The parameter value must be between 1 and 6. The default value is 1.

NOTE – On rectangle elements, a thick line width will obscure the rectangle's selection marker.

---

To change the *font* of a caption element, first select the element, then execute the command Draw.ChangeFont. For example:

```
Draw.ChangeFont Oberon16.Scn.Fnt
```

The command parameter ("Oberon16.Scn.Fnt" in this case) specifies the file name of the font. The default font is "Oberon10.Scn.Fnt".

---

**MACROS**

The graphics editor supports the following operations for using *macro* elements:

- Draw macro
- Decompose macro

---

*Macros* are small drawings that are saved in a library, and then inserted as elements into a graphics document.

Macros let you extend the set of graphic elements available to include user-defined custom elements.

---

To *draw* a macro element, first left-click where the lower-left corner of the element should be.

Next, draw the element by executing the "Draw.Macro" command. For example:

```
Draw.Macro Shapes Triangle
```

The two command parameters specify a macro library (in this case "Shapes") and a macro element stored in the library ("Triangle").

---

When a macro element is first drawn, it is automatically *selected*.

The selection marker appears as a rectangular dot grid, which appears around the macro element.

The selection area for a macro element is the lower-left corner of its selection marker.

---

NOTE – On some macro elements (such as "Pentagon" in the "Shapes" library), the selection area is not near any of the visible parts of the element. In this case, you must infer the location of the element's selection marker (or instead right-drag the mouse to select the element).

---

Macro elements work like regular graphic elements – they can be selected, moved, copied, or deleted. They can even be used in the definitions of other macro elements.

---

A macro element works as a single unit – its component elements cannot be edited separately.

However, you can *decompose* a macro element back into an editable collection of its component elements.

---

To decompose a macro element, first select the element, then set the draw cursor where you want the decomposed macro element to be.

Next, execute the command "Macros.OpenMacro". For example:

```
Macros.OpenMacro
```

This command copies a decomposed version of the selected macro element to the location specified by the cursor.

In the decomposed copy, all the component elements are automatically selected. To deselect them press the Esc key, then try selecting the component elements individually.

---

**LIBRARIES**

The graphics editor supports the following operations for managing macro libraries:

- Add new macro element to library
- Save library
- Load library
- Unload libraries

---

Macro elements are stored in *libraries*, which contain one or more macros and their assigned macro names.

Macro libraries are stored in files with the file type suffix ".Lib". For example:

Shapes.Lib

---

To *add* a new macro element to a library, first construct the element in a graphics document, then *select* all of its component elements.

Next, while the elements remain selected, create an invisible *bounding rectangle* around them. Left-click where where one corner of the rectangle should be, then shift-left-click where the *opposite* corner should be. The two cursors indicate where the new macro element's bounding rectangle will be.

Next, create the macro element by executing the "Macros.MakeMacro" command. For example:

Macros.MakeMacro Shapes Hexagon

The two command parameters specify the macro library (in this case "Shapes") where the macro element will be stored, and the name of the new macro element ("Hexagon").

---

If the specified library does not exist, a new library is created with the specified name, and the macro element is stored in it.

If another macro element with the same name already exists in the specified library, the new macro definition replaces the existing one.

---

NOTE – The bounding rectangle created for a new macro element gets used as the element's selection marker.

---

To *save* a macro library, execute the command "Macros.SaveLibrary". For example:

```
Macros.SaveLibrary Shapes
```

The command parameter specifies the macro library (in this case "Shapes") to save.

The command writes this library to the corresponding library file ("Shapes.Lib").

---

NOTE – If you do not save a macro library after adding new macro elements to it, the added elements are not saved, and remain available only during the current Oberon session.

---

To *load* a macro library, execute the command "Macros.LoadLibrary". For example:

```
Macros.LoadLibrary Shapes
```

The command parameter specifies the macro library (in this case "Shapes") to load into memory from the associated library file ("Shapes.Lib").

---

To *unload* all macro libraries from memory, execute the following command:

```
Graphics.UnloadLibraries
```

NOTE – When the macro libraries are unloaded, any changes made to the libraries will be lost, unless they are first saved with the Macros.SaveLibrary command.

---

**DRAWING GRID**

The graphics editor supports the following operation for managing the contents of a graphics window:

- Show or hide the drawing grid

---

The *grid* is the pattern of dots that appears in the background of every graphics window. It is used to align the graphic elements in a graphics document.

To show or hide the grid, execute the menu bar command "Draw.Grid". For example:

```
Draw.Grid
```

---

## MOUSE COMMANDS

The *mouse commands* in the graphics editor are similar to the standard Oberon mouse commands, but have been redefined to perform the most common draw operations.

---

| **Mouse** | **Command** |
|---|---|
| Left Click | set draw cursor |
| Shift Left Click | set additional cursor |
| Left Drag | draw quick line |
| Shift Left Drag | scroll graphics window |
| | |
| Middle Drag | move element |
| Shift Middle Drag | copy element |
| Shift Middle Click | copy selected text to caption |
| | |
| Right Click | select element |
| Right Drag | select multiple elements |
| Shift Right Click | add element to selection |

---

## TEXT COMMANDS

The *text commands* in the graphics editor appear in two places:

- Graphics window menu bars
- Graphics editor command window

To open the command window, execute the command "System.Open Draw.Cmd".

---

| Name | Command |
|---|---|
| Draw.Grid | show/hide drawing grid |
| Draw.Delete | delete element |
| Draw.Restore | restore document display |
| Draw.Save | save document |
| | |
| Rectangles.Make | draw rectangle |
| Curves.MakeLine | draw line |
| Curves.MakeCircle | draw circle |
| Curves.MakeEllipse | draw ellipse |
| | |
| Draw.SetWidth | set default line width |
| Draw.ChangeWidth | change line width |
| Draw.ChangeFont | change caption font |
| Draw.Macro | draw macro element |
| | |
| Macros.MakeMacro | create new macro element |
| Macros.OpenMacro | decompose macro element |
| Macros.LoadLibrary | load macro library |
| Macros.SaveLibrary | save macro library |
| Graphics.UnloadLibraries | unload macro libraries |

---

## KEYBOARD COMMANDS

The *keyboard commands* in the graphics editor are used as move commands, and (in the case of Ctrl-D) and as a keyboard shortcut for the menu bar command Draw.Delete.

---

| Key | Command |
|---|---|
| Ctrl-I | move element up by one pixel |
| Ctrl-J | move element left by one pixel |
| Ctrl-K | move element down by one pixel |
| Ctrl-L | move element right by one pixel |
| | |
| Ctrl-D | delete graphic element |

---

## LIMITATIONS

The graphics editor has the following limitations:

- No printing or exporting of graphics documents
- Bit-mapped (not vector) drawing
- No autoscrolling of graphics windows
- No selection outside current window view
- No resizing of shapes after they are drawn
- Text elements limited to one-line captions
- No undo

---

## CREDITS

The graphics editor was developed by Niklaus Wirth (NW).

The graphics editor software changes and Oberon Pi Draw User Guide were developed by Richard Gleaves (RG).