`xjpdraw` User's manual, version 4.2

October 6, 1995

Jean-Pierre MERLET

Abstract

We describe the use of a graphic editor, geometry oriented, which run for any workstations in the X-window environment. This editor generates a LaTeX file containing the drawing which can then be included in a LaTeX main file. Texts are written along the LaTeX standards and graphic primitives are described in the Postscript format.

It enables to create various geometric primitives (lines, texts, boxes, polygons, ellipses, ovals, miscellaneous curves and 3D). All this primitives can be easily manipulated.

Furthermore this program enables to create easily drawing on the base of external data.

Chapter 1

Basics

1.1 Introduction

`xjpdraw` was originally written in 1987 under the Suntools windowing system. It was designed mainly to be used inside the PRISME project of INRIA, a project devoted to the study of geometrical problems. It was translated for the X window system in 1991. PRISME members have suggested all along many improvements to the program and since 1991 they have created over 6000 drawings. This program is still undergoing improvements, especially at the requirements of the project's members and is maintained by J-P. Merlet. The current release is the eleventh official version of `xjpdraw`.

Before using `xjpdraw` it is necessary to install it correctly: please see the chapter "Installation". Basically it is necessary to indicate to the program some parameters for a correct run. This is done by creating a "configuration" file and setting the environment variable `JPDRAW` to the location of this file. To create a configuration file see the chapter "Installation". However this creation should have been done by your local system administration staff and you don't have bother with that (at least until you are proficient with the use of `xjpdraw`, afterward you may want to modify some parameters in the configuration file).

You don't need to read this manual before using `xjpdraw` (except the first chapter to understand the role of the basic items you will have to use when creating a drawing)

Any suggestions, comments or bug reports are to be sent at the above E-mail address

Jean-Pierre.Merlet@sophia.inria.fr

`xjpdraw` is a drawing software geometry oriented (meaning that is is intended to be used by people dealing with geometrical problems). It run under the X-window system although in some mode it can be used even with a non-graphic terminal. It is intend to be used in conjunction with the LaTeX text editor and basically produce a LaTeX file encapsulated in a `picture` environment.

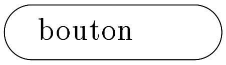
A `xjpdraw` drawing consists of various elements called **"primitives "**. Primitives examples are, for example, lines, texts, rectangles etc... The purpose of this editor is to create and manipulate these primitives with the use of the mouse and some keys, although a drawing can be created without using these devices. The maximum number of primitives is 1500, but this number can be extended if necessary, see page 91.

1.2 Buttons, Menus and Icons

1.2.1 Buttons and menus

In the `xjpdraw` window you will see mainly two different kinds of software facilities which enable to indicate what kind of functionality you want to use. We will have a look at the facility, the best way being to do it in front of the program. To run `xjpdraw` just use the command `xjpdraw` on your workstation. If you cannot start the program ask your local guru. If *you* are the local guru have a look at the error message and go to the chapter "Installation" to figure out what is wrong.

First you have the *buttons* which have the following shape:



To use a button put the hot spot of the mouse inside its box (the border of the button will get highlighted). At this point two things may happen: a *menu* is attached to the button or the button is a stand-alone button. To determine the behavior of a specific button put the hot spot of the mouse inside the button box and press the right mouse button (and kept it pressed).

If a menu is attached to the button (for example try the `FILE` button at the upper left part of `xjpdraw` window) it will appear: you will get a set of boxes oriented along the vertical axis. In each box a text indicates the functionality attached to the box. If no menu appear you have a stand-alone button and to use the functionality associated to the button you just have to click inside the button with the left mouse button. For example try the `Extensions` button in the upper left corner: by clicking in it the name of the button will change to `Standard`. If you click again the name will come back to `Extensions`.

Now assume that you have got a menu. You can move along the menu, putting the hot spot of the mouse in the various boxes (the current boxes will get inverted, (figure 1.1).

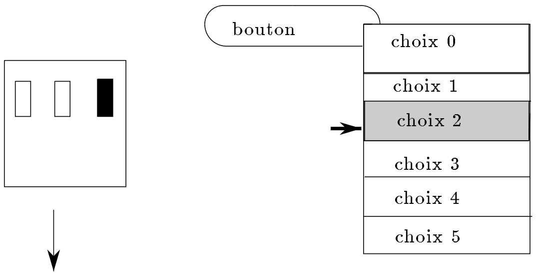


Figure 1.1: Choices in a menu

If the hot spot of the mouse lie inside a box and you release the mouse button the corresponding functionality will be executed. Note that if you have used the middle mouse button instead of the right one you will get a short description of the functionality attached to the box, but this functionality will not be executed.

Remark If you have got a menu the mouse motion enables only to move in the menu. It is not possible to activate another menu without releasing the mouse button If you do not intend to use one of the functionality of a menu put the mouse outside the menu boxes and then release the mouse button.

1.2.2 Icons

1.2.2.1 Principle

Basically icons are similar to button except that their functionalities are described not by text but by images. Another differences is that different functionalities may be attached to one icon. The various functionalities will appear by clicking successively inside the icon or by using some specific keys: different images will then appear. When a functionality attached to an icon is activated the image will get inverted.

1.2.2.2 Example: the Quit icon

By clicking into this icon you will leave `xjpdraw`. To avoid any error the program will ask you to confirm this choice and will indicate if the drawing has been modified without having been saved.

1.2.3 The xjpdraw windows

Sometime during the use of xjpdraw you will have to give file names, numbers, etc.... This is done by using specific windows called *text windows*. These windows are active only if the mouse hot spot lie inside the window. If the program ask you to type some specific string put your mouse inside the window, enter you text and hit the Return key to validate your entry. To cancel the command just delete all the characters in the window and then hit the Return key.

As soon as your are familiar with the use of the buttons, menus, icons you can start your first drawing.

1.3 Running xjpdraw

We have seen that you run xjpdraw by typing the program name. Parameters can be given after the name of the program (see page 91) but they are unnecessary right now.

1.3.1 Presentation

The xjpdraw window is composed of various elements (figure 1.2):

1. a menu panel (on top, horizontal)
2. an icon panel (on the left, vertical)
3. a canvas (where the drawing will be made)
4. an information panel (at the bottom, horizontal) where will be printed information messages about the current state of xjpdraw
5. a text editor panel, emacs like (at the bottom, close to the information panel)
6. a "Cursor" window in which will be indicated in certain modes the current location of the mouse (close to the information panel)
7. a window indicating the current number of primitives (close to the information panel)
8. a text window (at the bottom right), the *file window*, where you will feed data to the program
9. a window called "fast info" (on the upper right, outside the main screen) where xjpdraw will indicate what should be your next move or what the program is doing now
10. a window called "magnet" (just under the "fast info" window), whose role will be explained later.
11. an optional window which will appear only in the slide mode (we will see that later) which indicate the current page number and the total number of pages of the document.

Note that it is absolutely necessary that the "magnet" and "fast info" windows lie completely outside the main screen. xjpdraw will test that and emit a warning message if the location is not correct. Note also that additional windows may appear when running the program.

To use any icon in the panel you have to click in the icon. But you can also select an icon with the mouse in the canvas by pressing the F key (or the equivalent key as defined in the configuration file) indicated in the icon. For example if you want to select the line icon you will press the F3 key.

The canvas has scrollbars on its border enabling to visualize other part of the canvas: the full canvas is an 1000x1000 pixel area which represent a 14.5cmcm by 14.5cmcm paper sheet, so that what you see on the screen is almost double size with what will finally printed on paper.

The information panel will indicate what are the possible commands you may issue in the current state of xjpdraw and as soon as you have started an action the "fast info" window will indicate what should be your next action. The information panel will also print error messages indicating what you have done wrong.

The "emacs" window will be used for correcting text primitives

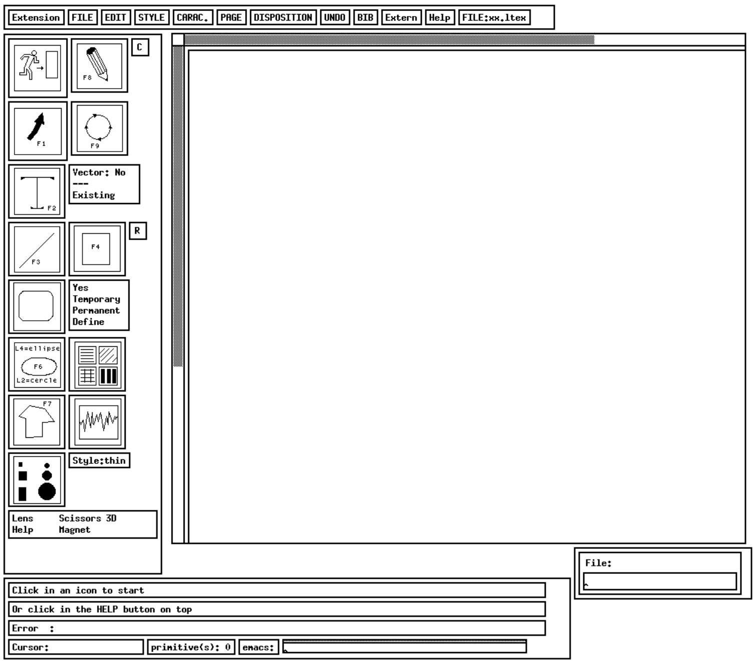


Figure 1.2: The `xjpdraw` window

1.3.2 Geometry of the window

The overall presentation of `xjpdraw` should be similar to the one presented in the figure. If you don't get such presentation you should indicate a correct geometry in your `.Xdefaults` file. My favorite geometry for a SUN workstation is defined by:

```
xjpdraw.Geometry : 1000x870+0+0
```

1.3.3 Key name

Many function keys will be used when running `xjpdraw`. In this manual we will use as name for this function keys the name which were present on some keyboard of SUN workstation i.e. name composed of the (R,L,F) character followed by a number.

In fact these keys can be assigned to any key on your keyboard via the configuration file (see the "Installation" chapter) and any information message appearing in the information panel will display the right name.

1.4 Primitive drawings

The choice of the type of primitive you will create is done through the icons, which display clearly the type of primitive they enable to draw. Sometime it may happen that after having created a primitive the canvas need to be refreshed. In order to do that use the command `Redisplay all` in the `Page` menu. A faster is to use a *shortcut*: many commands of `xjpdraw` can be used just by pressing a key. For this command press the `F5` key (while the mouse is inside the canvas) to redraw. Note that the need to redraw has been drastically reduced starting from version 4.0.

Note that informations will appear in the information panel each time you click inside an icon. These are general informations which describe how to draw the corresponding primitive and the options you have for creating the primitives. Note that when creating a primitive some information on the dimensions of the primitives will be displayed in the "fast info" window.

Before explaining how to create a drawing we have now to explain the management of the files in `xjpdraw`. Reading the following sections is unnecessary if you want just to get a feel of how `xjpdraw` works, and what kind of drawing it can do. In that case you can skip directly to chapter 2 page 13.

1.5 Saving in a file

Assume that you have created a drawing. In this section we will save the drawing in a file and print it. Whenever you save a drawing `xjpdraw` will save the drawing in a file in a specific format. The extension name for this file will be `macdraw`. Don't bother with this file, it is used only to load back the drawing.

As for the drawing `xjpdraw` may use three different ways to save it.

1. in a file where the drawing is described using the (poor) standard graphic capabilities of LaTeX. Saving in that way will generate two files: the file you will include in your LaTeX source file (usually with a .ltex extension) and the .macdraw file

2. in a LaTeX file which will contain all the text primitives and which will contain a specific command for including a Postscript file, the one containing the graphic primitives, which will be automatically created by `xjpdraw`. Three different files will be created: the .macdraw file, a laTeX file and a Postscript file (with a .ps extension)

3. the third way is almost similar to the previous one except that the Postscript will be directly included in the LaTeX file. Consequently only two files are created but there are some restrictions on the size of the Postscript which can be included in this way.

Anyway don't worry: you will have only to include the LaTeX file.

1.5.1 Save in the default file

When running `xjpdraw` a default name `xx` is attributed to the file. We will now save the current drawing in this file. In order to do that we will use the menu attached to the `FILE` button in the menu panel. One of the order of this menu is:

`Save the drawing in Postscript and the text in LaTeX`

Put the mouse in this box, release the button. The information panel and the "fast info" will indicate that the program is currently saving files. Quit `xjpdraw` and you will see that three files have been created: `xx.macdraw`, `xx.ltex`, `xx.ps` (or maybe `xx.ps.Z`).

As told before the `xx.macdraw` contains a description of your drawing in a specific format. To convince you run the command `xjpdraw xx.macdraw`: your drawing will appear on the screen.

Another way to save your file will have be to used the command `Save PS and LaTeX in the file` in the `FILE` menu. Here only the `xx.ltex` and `xx.macdraw` will have been created.

1.5.2 Saving in another file

Evidently it is possible to save your drawing in a file different from the default file. Assume that you want to save your drawing in a LaTeX file called `toto.ltex` (first you have to be sure that no such file exist in the current directory). To save the drawing we will use another command of the `FILE` menu:

`Save the drawing in PS and the text in LaTeX in a file called..`

As soon as you have released the mouse button the mouse hot spot will move to the file window (at the bottom right part of the screen). Give your file name in this window and press the `Return` key. The file name should have an extension like `.ltex` otherwise the program will emit a warning message and add a .ltex extension. Note that you cannot put a .ps or .macdraw extension. `xjpdraw` will have generated three files `toto.ltex`, `toto.ps` et `toto.macdraw`. Note that the current file name will appear in a window at the same level as the menu panel. Sometime the file name cannot fit in this window and you will not see the full file name. If you want to recover the full name put the mouse in the canvas and press the `CTRL-f` key (that is press the CTRL key and simultaneously the 'f' key) and a window will appear with the name of the file in it, together with the name of the directory in which the new file will be saved.

Remark:

Avoid the presence of many different extensions in the file name: `xjpdraw` uses a very simple method to deal with extension names.

Note that to save your file you may have also used the command `Save PS and LaTeX in the file named`. In that case only the files `toto.ltex` and `toto.macdraw` will have been created.

1.5.3 Writing on an existing file

If you try to overwrite an existing file `xjpdraw` will tell you that this file exist and will ask you if you really want to overwrite it. You can either continue or cancel the operation using the corresponding button.

Remark:

The drawing creation date, the user's name together with the directory in which the drawing has been initially created will appear as comment in the saved LaTeX file.

1.6 Automatic backup

A backup file is created very regularly when you run the program: if the base name of the current file is `toto` the backup file will be named `#Ctoto.macdraw` . The backup file enables to recover most of the drawing in case where `xjpdraw` crash (if that happen please send a bug report stating precisely what you were doing at the time of the crash together with the current .macdraw file). Note that you can also get a backup of your drawing by using the files generated by the undo mechanism (see page 50).

The time between two updates of the backup file is automatically adjusted according to the size of the drawing (creating a huge backup file may take some time). But you can also fix this time by using a command in the `EDIT` menu, see page 47.

1.7 On line help

The `Help` button in the menu panel enables to obtain on line either the full manual (if you can visualize a Postscript file) or short summaries of the manual, arranged by topics. By clicking in the "Help" button a new panel will appear which include various buttons (a shortcut to make this window appear is `CTRL-z`). The first button enable to get out of the "Help" mode. The "Manual" button enables to visualize the full manual (according to your particular installation this feature may not be available). The other buttons correspond to various topics. By clicking in them you will get the corresponding chapter of the short help file. If no topic satisfy you, you may try the "A propos" button which will display the whole short help file. In any of the new window that appear after clicking in one of the help button you may use most of the `emacs` command to move around (`CTRL-n,p`: next and previous line, `ESC-v` previous page , `CTRL-v` : next page). In the "A propos" mode you can look for particular words or sentences by using the `CTRL-s` key: another window will appear in which you will introduce the string you are looking for.

1.8 Loading a file

1.8.1 The Load command

The `Load` command from the `FILE` menu enables to load a drawing i.e. to read and display a drawing described by a .macdraw file. The loaded drawing will replace the current drawing (if any). Note that no check is made to insure that you have not modified the initial drawing and omit to save it.

When using this command a new new panel which will appear. In this panel various lists appear displaying the loadable files in the current directory, the directories in the current one and a root directory (denoted by ..).

They are various ways to give the name of the file you want to load. You can use the new panel: click at the appropriate place in the list if you want to load a file or move in another directory (in which case the loadable files list will be updated). Note that the panel will disappear only after clicking in the `Quit` button. Note also the two buttons which enable to hide or delete the backup files (the files starting with #C). Another way is to give the file name in the file window and press the `Return` key. Note that you load a .macdraw file but you may omit the .macdraw extension (except if your file has more than one extension). If instead of using the `Return` key you use a space or a tab the panel will display all the files whose name start with the given chain (completion process).

In the `Find` window you may ask to find a specific file: `xjpdraw` will look for this file recursively starting from the current directory. Note that you may use the "*" character as a match for any set of character (for example to*.macdraw will match toto.macdraw).

`xjpdraw` is also able to read the .macdraw files which have been compressed by `compress` (.Z extension) or `gunzip` (.gz extension). Note that a slide file (see this chapter) cannot be loaded if you are not in the slide mode.

1.8.2 Loading a file at start time

When running `xjpdraw` you can give an extra argument which is the name of a file to be loaded. The .macdraw extension is generally not needed. You can use indifferently:

```
xjpdraw toto                xjpdraw toto.macdraw
```

1.8.3 Album

The `Album` command of the `FILE` menu enables to add a drawing to an existing one. The process is similar to the one used for loading a file. A difference is that the primitives of the added drawing are *associated* (see page 26) meaning that they constitute a group which can be moved, rotated etc..as a whole. A side effect is that it is preferable to dissociate (un group) the existing primitives before using the album. Another difference is that after having selected a file the file panel will disappear. Note that you cannot load a file written in the slide mode if you are not in this mode.

1.9 Saving `xjpdraw` state

If you define an environment variable `JPDRAW_STATE` as `ON` a file called *.jpdraw_state* will be created when you leave `xjpdraw`. This contain the status of `xjpdraw` just before you have left it. This mean that if you run again `xjpdraw` without any argument you will come back exactly in the same state: same loaded file, same disposition of the window etc.. To inhibit this behavior turn the the `JPDRAW_STATE` to `OFF`.

1.10 Printing

1.10.1 Standard graphic possibilities of LaTeX

The LaTeX file saved by `xjpdraw` is encapsulated in a `picture` environment. Therefore it cannot be compiled alone but must be included in another LaTeX file.

As a standard feature LaTeX has got some graphical possibilities although they are poor compared to the performance of the available printer.

In LaTeX version 2.09 the following primitives are available

- rectangles (which can be black)
- circles (with a limited radius, black or hollow)
- texts
- lines and vectors (whose slope should be defined as the ratio of two integers in the range $\pm$ 6 ($\pm$4 for the vectors), being prime.

For example the following file (produced by `xjpdraw`) define the drawing in figure 1.3.

```
\begin{picture}(   3.0,   1.9)( 0.913,11.339)
\thinlines
\large
\put(  2.48, 11.98){\circle{  1.24}}
\put(  3.12, 11.34){\framebox(  0.81,  1.90){}}
\put(  0.91, 11.51){\line(5,4){  2.19}}
\end{picture}
```

Although LaTeX2e provides an improved version of the graphic environment it remains that it is still tedious to describe a drawing in this way.

`xjpdraw` can generate for you a file containing only primitives that are standard in LaTeX and therefore are portable. This behavior is not the default one, so you have to turn `xjpdraw` to the standard mode by clicking in the `Extensions` button on the left of the menu panel. This button shall now indicate `Standard`. In this mode only

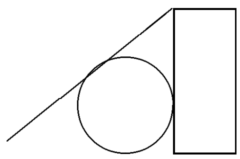


Figure 1.3: A drawing described in LaTeX

drawing fully compatible with LaTeX (version 2.09, not LaTeX2e) can be generated. But to respect this feature you need to save your file with the `Save` or `Save Under` command of the `FILE` menu.

In the early version of `xjpdraw` a TeX package called the *extensions* was provided in a file *lplainext_dvips.tex*,, which extend the graphic possibility of LaTeX by using some Postscript orders. This feature is now obsolete except for one: the rotation order which may be used to rotate texts. You even don't need this last order if we use LaTeX2e and the `rotating` package.

So since version 3.5 it is not recommended to use the `Save` and `Save under` command of the `FILE` menu except when you are in the `Standard` mode.

Most of the time you will save your file using Postscript for the geometric primitives. A side effect is that you may not be able to see your drawing if you use a dvi previewer like `xdvi`. Some dvi previewer may however display some restricted Postscript files. For example if you told `xjpdraw` to use the `epsfig` macros to include the Postscript file by having the following line in your configuration file:

```
# PS \epsffile{%S}
```

a previewer like `xdvik` will be able to draw the figure except for the rotated text.

You can add your figure in your LaTeX source file either by copying the LaTeX file of the drawing or by using the LaTeX command `\input`, as for example `\input{toto.ltex}`.

In a figure environment it would in general be necessary to center the drawing by putting a `\begin{center}` right before the input and a `\end{center}` right after.

1.10.2 The dimensions

The graphic command of LaTeX use an arbitrary unit as dimension unit. The relation between a number in the picture environment and an existing length unit is established by defining the dimension of one picture unit. This is done by defining a dimension value for the variable `unitlength`, as for example in `\unitlength x cm`. The `xjpdraw` drawing assume by default that the value of `unitlength` is 1cm.

We will see that by giving a unitlength value different from 1 cm you will be able to scale your drawing (however without modifying the text size) without modifying the `xjpdraw` source file.

1.10.3 Printing

Sometime you may want to see the look of your drawing on paper i.e. to generate a stand alone LaTeX document which contain only your drawing. There is two commands which may do something in this line.

First if you want to generate only a the stand alone document you may use the command `LaTeX and Postscript verification` (or `LaTeX verification` if you are in the `Standard` mode), which appear in the `FILE` menu. These commands will create the stand alone document `bidon.ltex` which can be compiled. Basically the program generates a LaTeX preamble, then your drawing and finally an end of document. By default the document will be intended for LaTeX version 2.09 but you may easily indicate that you want to use LaTeX2e. If necessary you can customize the preamble and end of document. For these points see the "Installation" chapter. You can now exit `xjpdraw`, compile the file bidon.ltex and then use your favorite dvi driver (preferably `dvips`) to print it.

Another faster way to do that is to use the `Compile and print` command of the `FILE` menu which will perform all the previous operations for you and send directly the output to your printer.

1.11 Tools

The `Tools` menu appear in the icon panel, at the bottom. Different tools are available:

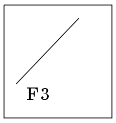
- the `lens` which enable to have an enlarged view of the screen around the hot spot of the mouse, see page 31).
- the `scissor` which enable to cut any primitive (except the text), see page 31.
- the `help` which enable to dismiss or to add the "fast info" window
- the `magnet` which enable to dismiss the "magnet" window. This magnet window enable to enter in a mode where the hot spot of your mouse will be slightly modified by `xjpdraw` according to the rule given for the magnet, see page 31.
- the 3D which will open a panel enabling to create 3D primitives, see page 20.

Chapter 2

Creating primitives

The basics primitives will be created by activating specific mode and this is done by clicking inside some icons of the icons panel.

2.1 Line



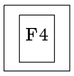
First select the icon, you are now in line mode. If you want to draw vectors click in the `Vector` button in the icon panel. By clicking successively in this button you will get `Vector: yes` (one arrow at the end of your line), `Vector: 2<>` (arrows at the start and end point of your line, with direction opposite to the line), `Vector:2><` (arrows with same direction as the line).

To create a line (or more exactly a segment) put the hot spot of your mouse at the start point, press the left mouse button, drag the mouse to the end point (i.e. move the mouse to this position with the mouse button still pressed) and then release the mouse button. While dragging the mouse you will see your segment.

If you want to modify the style of the segment you may use the `---` button in the icon panel. A new window will appear enabling to choose among the following styles: thin (default), thick, dashed, thick dashed. It enables also to choose among the five different type of arrows for the vector. In page 21 we will explain how to modify the thickness of the line and how to choose among the 25 types of dashed line. Note that changing the line style is valid for any primitives (except the text), and that it is possible to modify the style of existing primitives.

It is possible to enforce that the slope of the line is compatible with the possible slopes of the standard possibilities of LaTeX (see page 9) by drawing the line while keeping the L8 key pressed. This option is very useful for drawing lines at particular angles, like horizontal or vertical lines, because you don't need to be to careful about the location of the hot spot of the mouse.

2.2 Rectangle



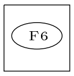
2.2.1 Principle

By clicking on this icon you will be able to draw rectangle or square.

Put the hot spot of the mouse in the canvas, press the left mouse button at the location of one corner of the rectangle, drag the mouse to the opposite corner, then release the mouse button.

If you want to draw a square press the L2 key with the mouse in the canvas. A "C" should appear in the icon panel close to the rectangle icon in place of the default "R". To go back to the rectangle mode press the L4 key.

2.3 Ellipse, Circle, Arc



Using this icon you will be able to draw ellipses. Two creation modes are possible (figure 2.1):

- ellipse defined by its center and a bounding box. Click the left mouse button at the center, then drag the mouse to another position: the ellipse will be enclosed in the rectangle defined by its center and the final position of the mouse. This is the default mode.

- ellipse with horizontal and vertical axis going through 4 points. Click in the first three points, then move the mouse toward the fourth point. The ellipsis is drawn, according to the current position of the mouse. Click to finish.

To switch between these two modes click successively in the icon.

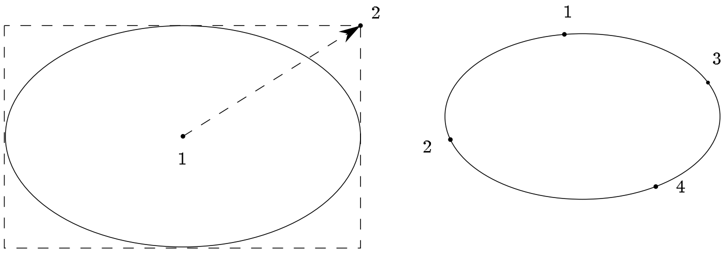


Figure 2.1: The various methods to create ellipses

2.3.1 Circles

This icon can also be used to draw circles. After selecting the icon put the mouse in the canvas and press the L2 key. The ellipse inside the icon will be substituted by a circle. You have various modes available to create circle: to switch between them click successively in the icon (figure 2.2) (left mouse button for going forward, middle mouse button for going backward). The possible modes are:

- 1st mode:we will define first the center of the circle, then a point on the circle. Click at the center, drag the mouse until you get the right circle, then release the mouse button.

- 2nd mode: here you define first a point on the circle then its center. The creation process is the same as the 1st mode

- 3rd mode: we give two points on the circle, the median line between the two points will appear, then we give the center. If the center is on the median line the circle will go through the two initial points. Click first at the two points, then move the mouse toward the desired center (the circle with center at the current position will be drawn) then click at the center.

- 4th mode: circle going through 3 points. Click at the two first points then move the mouse toward the third point (the current circle will be drawn), then click at the third point.

- 5th mode: circle with two lines inside oriented at $\pm$ 45 degree. The creation process is similar to the 2nd mode.

In general the icon image will indicate the current mode. The numbers in the image indicate in which order you have to give the points. To go back to the ellipse mode press the L4 key.

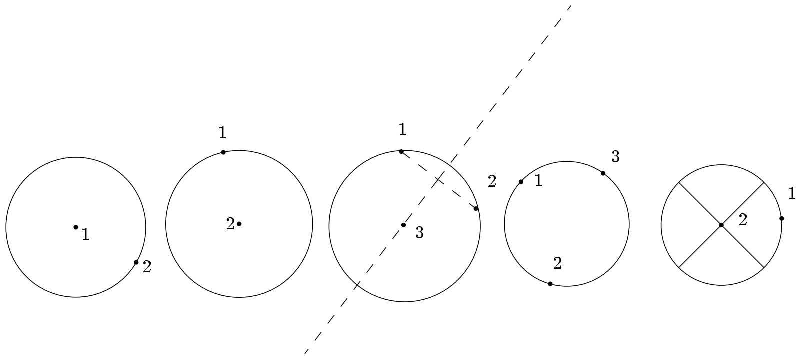


Figure 2.2: The various methods to create circles

2.3.2 Arc of circle

Arc of circle can be created in this mode. Select the third or fourth mode of circle creation. The two first clicked points will be the extremities of the arc. Create your circle, then choose between the two arcs joining the extremal points by using the L6 key (shortest arc) or L8 key (longest arc, figure 2.3). Note that when creating the arc in one of the vector modes you will get arrow(s) at the extremity of the arc.

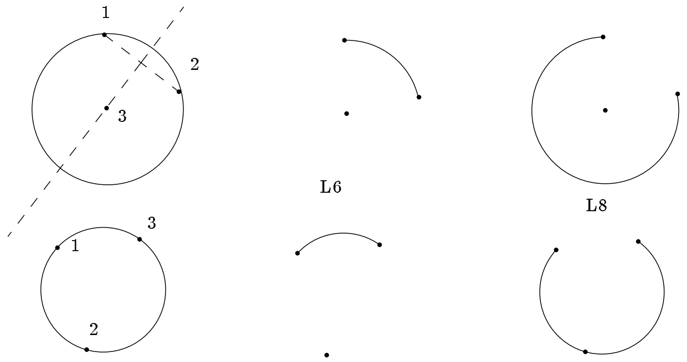
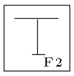


Figure 2.3: The various methods for creating arcs

Remark: in this mode the angles between the x axis and the extreme points of the arc, measured in degree, should be integer. Consequently there may be a slight difference between the two first initial points and those of the final arc. A possible way to overcome this restriction is to create arcs of circle by using the scissor on a circle (see page 31).

2.4 Text



This mode will enable you to add a text to the drawing. First define the character size of your string by using the commands of the `Carac` menu. By default the character size is 10pt and style roman. The given string should be in accordance with LaTeX rules as it will be compiled (consequently complex texts can be introduced, see the example of page 106).

To introduce a new text move the mouse hot spot at the desired location of your text (lower left corner), enter your text using the keyboard (you can correct a mistake by using the delete (or backspace) key). You can also copy text from anotehr window by selecting it, putting the mouse in the canvas and pressing the middle button of the mouse. The origin of the text will be the initial position of the mouse.

When the text is defined move the mouse some distance away from the text to indicate that it's finished.

As soon as it's done the text will appear in the `emacs` window and a small box in dashed line will appear around your text. This box is intended to indicate the *real* size of your text. Indeed as your text is a LaTeX command there is no real correspondence between the number of characters in the string and its final size. According to your configuration file the box dimension will indicate:

- the *real* size of the text because in your configuration file you have indicated that each text should be compiled (see the "Installation" chapter). A side effect is that a small delay due to the compilation process will take place as soon as you have entered a new text: the "fast info" window will display the message "Compiling, wait". Starting from version 4.2 this delay has been reduced as a latex is running continuously in background.

- an *estimated* size because the text are not compiled. This estimated size may be far from the mark, especially when using math symbols.

As mentioned, the fact that the text should be or not compiled is indicated in the configuration file. If you want to correct an existing text please see page 30.

You can also enter multi-line text. After each line just press the `Return` key.

Any text beginning by a % character is considered as a comment for the drawing and will not be printed.

The character displayed on the screen should approximatively correspond to the chosen size and style. To make that possible `xjpdraw` uses various fonts which may be not available in your installation. In this case `xjpdraw` will make its best to substitute missing font by existing one but may fail in doing so, in which case you need to indicate in the configuration file which font `xjpdraw` may use (see the "Installation" chapter). You may use the `xfontsel` program to determine which fonts are available in your installation.

The fact that on the screen the character are displayed in a bad size or a bad style (e.g. bold text appearing in roman font) is not important as the size and style will be respected when the drawing will be printed. Evidently you may also incorporate directly in your text style orders.

You may use in your text your own special LaTeX command and still get the exact size of the text on your screen. Indeed when compiling a text `xjpdraw` look for a file called `command.latex` in which any special command has to be defined, and include it in the LaTeX document it uses to find the text size.

If you are not very satisfied with the manner `xjpdraw` display your text remember that you can always use a previewer like ghostview to figure out the final version of your drawing (see page 39).

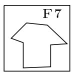
Some special command of LaTeX are treated in a special manner. For example most of the mathematic symbols may be recognized by `xjpdraw`, which will display them approximatively as they will appear in the printed version (if they are followed by a blanck, an _, a backslash or a ' character). For example if your text is `$\theta$` `xjpdraw` will display on the screen $θ$. For this purpose `xjpdraw` uses two special fonts, which by default are:

```
-adobe-symbol-medium-r-normal--14-140-75-75-p-85-adobe-fontspecific
-adobe-times-medium-r-normal--12-120-75-75-p-64-iso8859-1
```

If these fonts are not available in your installation you have to indicate new one in the configuration file. This process will work only if you have no more than 20 special commands in your text and if the special commands are separated by blanks. Note that the `emacs` window displays always your original text.

If you translate text primitives after having created them take care of not putting the origin of the primitive outside the screen otherwise your text will not appear when the drawing will be printed.

2.5 Polygon



Using this icon you can draw polygonal lines or closed polygons.

2.5.1 General polygons

To draw a polygon click with the left mouse button at each vertex of the polygon, and move toward the next vertex (the potential edge will appear on the screen). To finish a polygonal line click with the middle button. To close a polygon simply double-click with the middle mouse button (i.e. press two time the button at a very short interval)

You can cancel the polygon creation by pressing the R5 key and you suppress vertices by clicking on the R4 or L4 key.

Warning: You cannot have more than 30 vertices in a polygon.

2.5.2 Special polygons

2.5.2.1 Arrow head

You may create an arrow head polygon by clicking the polygon icon, then pressing the L3 key with the mouse in the canvas. The image in the icon describes the few vertices you have to give, `xjpdraw` computing the positions of the remaining vertices (figure 2.4). The L10 key enables to come back to the general polygon mode.

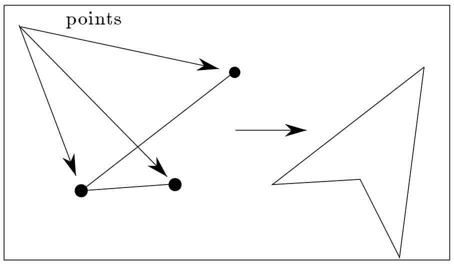


Figure 2.4: Creating an arrow head

2.5.2.2 Regular polygon

After clicking in the polygon icon you may create regular polygons (polygons with edges having the same length) by pressing the L9 key. The distance between the two first vertices will define the length of the edge. Then each time you click on the left button `xjpdraw` will add a new edge to the polygon. Clicking two time will define an equilateral triangle, three time a square, four time a pentagon and so on (figure 2.5. You finish your polygon in the same manner

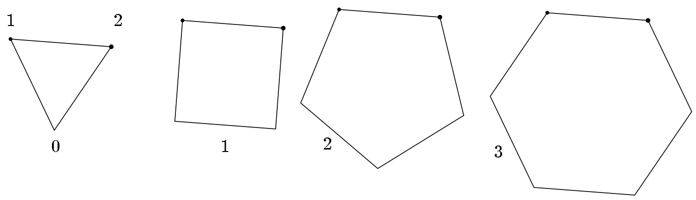
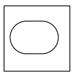


Figure 2.5: Regular polygons. Point 1 and 2 have been introduced and the number at the bottom indicates how many time the left mouse button has been used (note that each time the polygon has been finished with a double-click on the middle mouse button).

as for general polygon. The L10 key enables to come back to the general polygon mode.

2.6 Ovals



By selecting this icon you can draw ovals or part of ovals (i.e. rectangle with rounded corner). By using the keys L1, L2, L3, L4 you may select either a full oval (the default mode) or half oval, quarter of oval or quarter of oval with an arrow. The way you define your oval (figure 2.6) is the following:

- full oval: click at the oval center, drag the mouse. The oval will be included in the rectangle defined by the center and a corner at your current position. Release the mouse button when finished.
- for part of oval: click at the start of the primitive and drag the mouse. For the quarter of oval the current position of the mouse is joined to the start by always rotating anti-clockwise. Note that for the half oval if you are in vector mode two arrows will be added at the extremity.

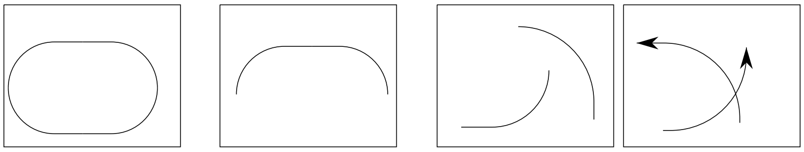
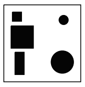


Figure 2.6: The various ovals

2.7 Marks



Marks are predefined primitives and with the possibility of defining your own mark. The shortcut for activating the mark icon is F10.

2.7.1 Default marks

The default marks (figure 2.7) are:

1. black squares: with 2mm et 5mm edge length
2. black rectangle: 2x5 mm dimension
3. black circles: with 0.5, 2 and 5 mm diameter

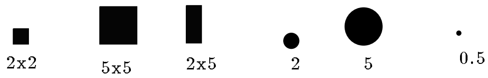


Figure 2.7: Default marks

You select the default mark by successively clicking in the icon (left mouse button for going forward, middle mouse button for going backward). To introduce a mark in your drawing click at a point: it will represent the lower left corner for the squares and the rectangle and the center for the circular marks. You can also select the standard mark by pressing one the key in the range 'A' to 'F'. When you have selected a given mark you can still draw another mark by using a key in the range 'a' to 'f'.

2.7.2 Defining and using marks

When some primitives are to be reproduced many time it may be interesting to define them as marks. In order to do that first draw the primitive you want to define as a mark, then click with the left mouse button in the mark icon until the image in the icon display the word `#define`. Click then close to the primitive you want to define as a mark. Each time you click close to a primitive you define it as a mark (note that you can define up to 10 marks).

To use one of the defined marks click in the mark icon until you get an image icon displaying the word `#use`. Put your mouse at the location of the center of the mark, then press the numerical key corresponding to the mark you have defined (the first mark has number 0 and the tenth number 9).

If you don’t remember the numbering of your mark press the `Return` key when in the `use` mode: the number of the mark will appear on the primitive which has been used to define the marks (if a primitive used to define a mark has been deleted it will be temporarily redisplayed).

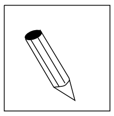
If you want to delete the last defined mark use the key ’-’, if you want to delete them all use the key ’=’.

Another trick with the `use` mode: you can use it to put marks at some specific points of the selected primitive. The process is the following:

1. define your own marks
2. select a primitive (without leaving the `use` mode)
3. press a ’A’ to ’F’ key to put the standard marks (or ’0’ to n for your own mark) on the selected primitive: the mark will be put on the extremities of a segment, on the vertices of rectangles and polygons and at the center of ellipses, circles and ovals.

Note that as soon as you have selected a primitive you cannot use the ’A’ to ’F’ to draw the standard mark and ’0’ to n to draw you own mark at the current mouse position. You need first to cancel the selection of the primitive, which is done by clicking at any point not close to a primitive.

2.8 The pencil



The pencil is used to draw various curves. Four different types of curves can be drawn. To select which mode you will draw use the menu attached to the button with the label ”C” close to the pencil icon. The following sections describe the various modes.

2.8.1 Hand-drawn curves (`Mouse`)

In this mode (which is the default mode) the mouse is used like a pencil. Click at the start point of the curve, then drag the mouse. Every 1/40 of second `xjpdraw` will use the current position of the mouse as a vertex of a polygonal line. Note that you cannot have more than 200 vertices.

2.8.2 Splines

The control point of the spline are defined by clicking with the left mouse button at their location (a control point is displayed with a cross). To draw the spline double-click with the middle mouse button at the last control point. You can cancel the control points by clicking with the middle mouse button. Note that the curve goes through all the control points. After having created a spline you can change the location of the control point, see the ”Resizing” section.

2.8.3 B-splines

The control points and the final one are introduced in the same manner as for the spline. Here the curve does not go through the control points.

2.8.4 Polynomial curves

The n control points are introduced as for the spline and are used to compute a $n-1$ order polynomial which goes through the control points. A polynomial curves will in general be "stiffer" than a spline curve (see figure 2.8.

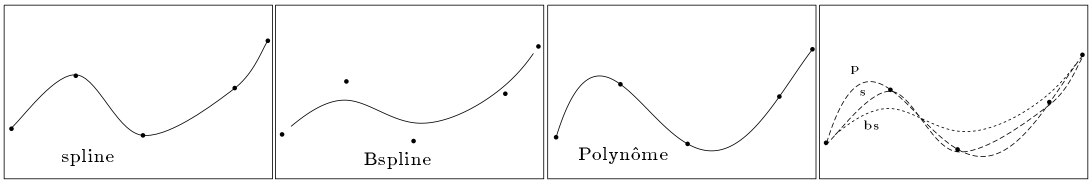


Figure 2.8: Spline, B-spline and polynomial curves obtained with the same control points

2.8.5 Parabola

This mode is used to draw arc of parabola (figure 2.9) either with an horizontal axis or a vertical one. To draw this primitive click at the start point, then drag the mouse. The initial motion, right after having introduced the start point, is used to determine it the tangent at the start point is vertical or horizontal.

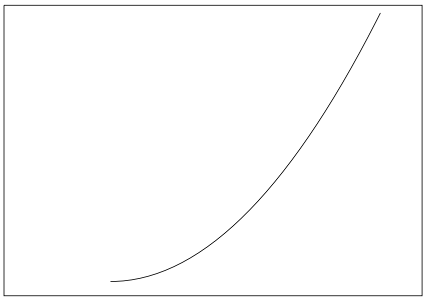


Figure 2.9: Arc of parabola

Note that you may draw arc of parabola (or other complicated curves) whose axis is neither horizontal or vertical by using the "Formula" mode where you will give the symbolic mathematical expression of your curve, see page 43. You can also rotate the arc after creation.

2.9 The 3D tool

The 3D module of the tools menu enables to create sphere, parallelepipeds, cones and objects of revolution. Note that `xjpdraw` does not keep in memory that this object are 3D, so you cannot manipulate them as a whole after having created them. After having selected the 3D tools a new window will appear. The following sections describe the various items of this window.

2.9.1 Choosing the 3D view

The first item of the 3D menu enables to display a new window in which you have first to select the point of view of the 3D object. This is done by using either the two scrollbars which define the value of the two angles enabling to specify the direction of the line of view or by selecting pre-defined point of view like `Perspective` (the default point of view), `Top`, `Face`, `Side`.

After having defined the point of view you have to select a representation type which may be `Shaded facets` (the default mode), `Facets` or `Symbolic`.

In the first modes the 3D object will be represented by planar polygons, each polygon being painted with a gray color depending on the angle of its normal with the line of view (light gray for polygon with the normal being collinear with the line of view). In the second mode the polygons are not painted.

You can increase the number of polygons used to draw a 3D object (therefore getting a better approximation) using a scrollbar of the 3D menu. By default the number of facets is 100 for the spheres and 10 for cylinders, cones. There is no verification on the visibility of the facets.

In the `Symbolic` mode no facets are drawn but the classical representation of 3D object is used, based on a few element, the 3D aspect being illustrated by dashed elements.

2.9.2 Drawing a 3D primitive

After having selected the view point and the representation type you may draw now 3D primitive (figure 2.10. The ways to do that are the following:

- *for sphere*: click at its center then click in another point to define the radius of the sphere.
- *for cube*: click in four points to indicate the center (point 1), its half width (defined by the x variation between point 2 and point 1, its half height (defined by the y variation between point 3 and point 1), its thickness (defined by the distance between point 3 and 4)
- *for cylinder*: click in three points to indicate the center of the cylinder base (point 1), its radius (defined as the x variation between points 1 and 2 and its height (defined as the y variation between point 1 and point 3).
- *for cone*: click in three points to indicate the center of the base of the cone (point 1), the radius of the base (defined as the x variation between points 1 and 2, the height (y variation between points 1 and 3) and the radius of the top of the cone (x variation between points 1 and 3).
- *for object of revolution*: first draw a polygonal line and select it. Then click in the `revolution object` mode and click with the left mouse button to draw. The object you will get is obtained by rotating the polygonal line around a vertical axis (figure 2.11).

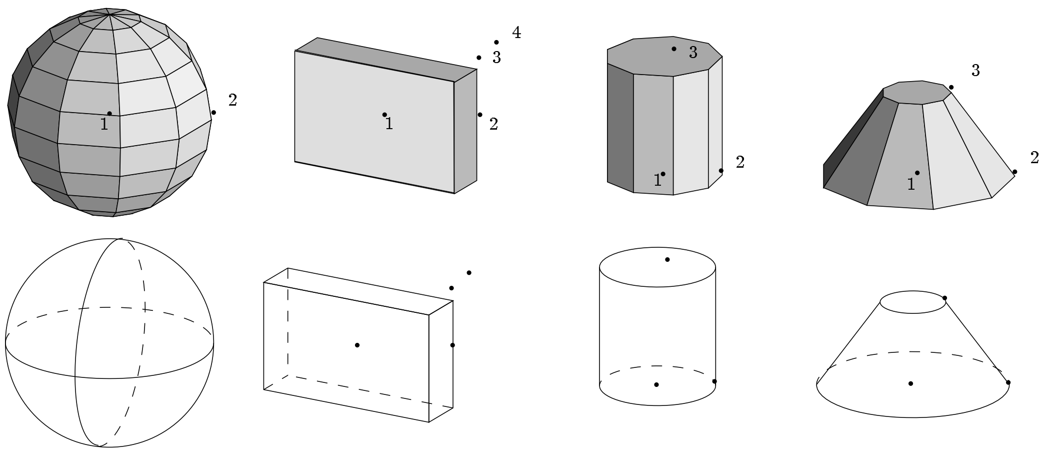


Figure 2.10: The 3D primitives in shaded facets and symbolic mode

2.10 Line style

Clicking in the `---` button close to the line icon will display a new window enabling to choose various line styles. Selecting an item in the list enables to choose a new line style. The following line styles may be selected:

- thin: thin continuous lines

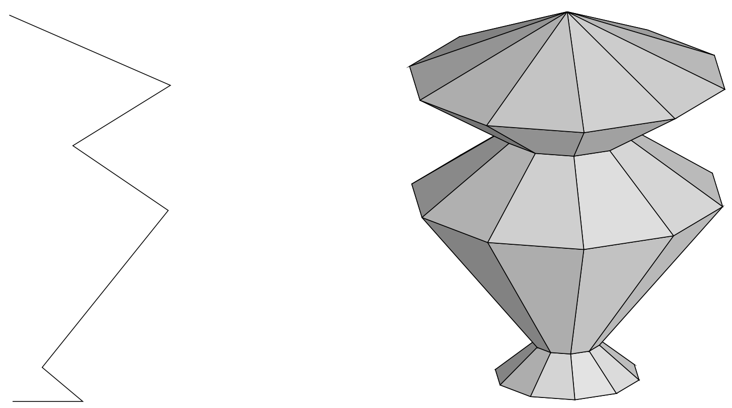


Figure 2.11: Creating an object of revolution.

- thick: continuous thick line. The thickness is fixed by using the command `Change line thickness` appearing in the `Existing` list (see below). The default thickness is 0.7mm.
- dashed: thin dashed lines
- thick dashed: dashed lines with the thickness defined by the `Change line thickness` command (see below). The default thickness is 0.7mm.
- other dashed: when selecting this item a new window will appear enabling to choose among 25 different dashed line styles (figure 2.12). To select a new dashed style put the mouse inside the window (the different dashed lines will only appear when the mouse is present in the window), then get close to the line with the desired dashed style (which should briefly blink), then click with the left mouse button. This window will disappear and `xjpdraw` is now in dashed line mode, with the selected dashed type (a shortcut to open the window of the dashed lines is to use `CTRL-l`).
- other vector: enable to choose in a new window among the 5 different types of arrow head for the vector. The default size of the arrow head is `Medium`.

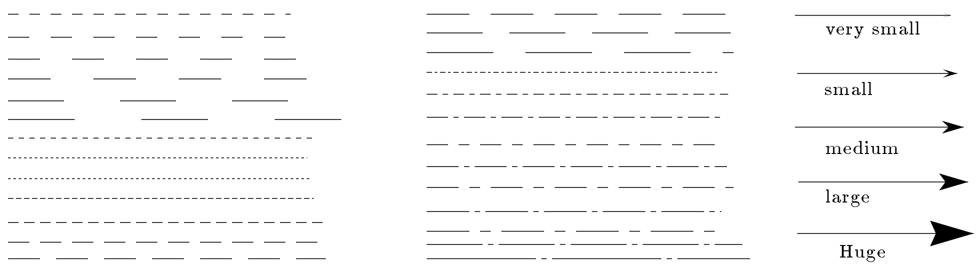


Figure 2.12: The dashed styles and vector heads of `xjpdraw`

Note that for the thick line the thickness of the line on the screen is only an approximate match for the the thickness of the printed line: don't use the border of line as a reference for any other primitive. The current line style is indicated in a small window in the icon panel. It is also possible to get the line style of a primitive by displaying the rulers and selecting the primitive: informations on the primitive will appear in the "fast info" window.

As soon as a line style is selected all the new primitives will have the selected line style. But you can also modify the line style of existing primitives.

2.11 Modifying the line style of a primitive

You can modify the line style of the selected primitive (see the "Selection" chapter, page 25), of the associated primitives (primitives that you have grouped) or of the latest created primitive. To do that first define your new line style, then click in the `Existing` button close to the line style button. A new window with a list will appear. Almost each item of this list indicate on what type of primitive the new line style has to be applied. You will find `for all`, `for the associated`, `for the selected`, `for the last primitive`, `for the dashed` or `for the thick`. Most of the item of this list are self-explanatory except `for the tick` which mean that the new line thickness has to be applied on primitive whose line style is thick.Similarly `for the dash` defined the new dashed style which has to be applied on primitive with a dashed style.

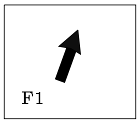
The last item of the list `Line thickness` enables to define the thickness of the future thick lines. A window containing a scrollbar will appear when clicking in this item. The line thickness can be adjusted between 0.1mm to 3 mm. The current value of the line thickness of the thick line is displayed by the scrollbar. Note that if the new line style include the vector mode it may be necessary to apply more than once the request for the change of line style. For example a dashed vector line on which is applied a thin mode will first be transformed into a dashed line then into a thin line.

Chapter 3

Selection and association

The *selection* of a primitive is the operation intended to indicate to `xjpdraw` that an exiting primitive will be the subject of some transformation (e.g. a translation). *Associating* primitives is the operation that will group primitives so that if one member of the group is transformed, then all the members of the group will be submitted to the same transformation.

3.1 Selection



By clicking in this icon `xjpdraw` will move to the selection-association mode. In this mode it is also possible to apply various commands using only some keys as shortcuts (see page 93).

3.1.1 Selecting a primitive

In order to select a primitive put the hot spot of the mouse close to the border of the primitive and click with the left mouse button. If you have proceeded correctly the primitive will briefly blink and the "fast info" window will display the message "Primitive selected".

Note that you can modify the way primitives are blinking by using the `BLINK` command in the configuration file. As soon as a primitive is selected it will be kept selected until the next selection.

Another selection mode is available by using the order `SELECTION_TYPE` in the configuration file. In this new mode newly created primitive will be automatically selected, see the the chapter "Installation". If the canvas rulers are displayed (see page 46) the "fast info" window will display informations regarding the dimensions and feature of the selected primitive. Note that you can also obtain length and angle information related to the current position of the mouse with respect to a given point by using the `measure` tool of the `PAGE` menu.

After having selected a line, circle, ellipse, ovals, or rectangle primitive the center of the primitive will be indicated by a small cross. For general polygons the center of mass will be displayed by such a cross.

Note that to speed up the selection process `xjpdraw` split the screen in rectangular buckets, each primitive having an attached list of bucket. When clicking the mouse button the location of the corresponding bucket is computed and the distance to a primitive is computed only if the bucket is in the list attached to the primitive. In some case the buckets list may have not been updated and it may be impossible to select a primitive. In that case use the `Redisplay all` command of the `PAGE` menu, which will update the buckets list, and start again the selection.

3.1.2 Selecting overlapping primitives

If two primitives overlap it may be difficult to select one of them because `xjpdraw` uses a distance criteria to determine which primitive is selected: in case of similar distance the same primitive will always be selected. In that case press the L3 key continuously and select the primitive: the second closest primitive will be chosen.

To solve this problem you may also change the selection mode to `Selective selection` where only primitives of a certain type may be selected, see page 43.

Note that in the "Selection" mode a selected primitive can be translated by dragging the mouse (not always, see the section "Translating a primitive"). This explain why in some case if you select a primitive with the mouse in the vicinity of the primitive border it can "jump" in a position where one point of the primitive lie exactly at the mouse position.

3.2 Association

The association of primitive is equivalent to select more than one primitive in order to group them. When any member of the group is submitted to a transformation, all the members of the group will be submitted to the same transformation. Associated primitives will have a "+" sign on their border. Primitives can be associated in four different manners.

- *all the primitives in a rectangle or a polygon primitive:* all the primitives which are included in a rectangle or a polygon can be associated by selecting the rectangle or the polygon and then pressing the L8 key.
- *all the primitives in a rectangle or a circle:* without enclosing the primitive to be associated in another primitive. Put your mouse in a location far away from any primitive, press the left mouse button and then drag the mouse: a dashed rectangle will appear. Continue to drag the mouse until all the primitives to be associated lie inside the rectangle, then release the mouse button. All the primitives in the rectangle are now associated. If you use the right mouse button a dashed circle will appear instead of the rectangle.
- *designated primitive:* selecting a primitive by using the middle mouse button instead of the left button will associate the primitive.
- *primitives of the same type:* the command `Multiple association` of the `EDIT` menu enables to associate and dissociate all the primitives of a given type. , A new window will appear with a list of primitive type, click in the desired type item.
- *primitives in a given position with respect to the selected*: the command `Associate everything on..of the selected` of the `EDIT` menu enable to associate every primitives in a given position with respect to the selected primitive (every primitives on the left, on the right, below, etc...).

For dissociating (associating) all the primitives use the command `Dissociate all` (`Associate all`) of the `EDIT` menu. . When in "Selection" mode a shortcut is to press the "D" ("A") key, in the other modes `CTRL-d` and `CTRL-a` should work. To dissociate a primitive you can select it and use the command `Dissociate the selected` of the `EDIT` menu In "Selection" mode a shortcut is to select the primitive and press the "d" key.

The associated primitives will all be transformed for the following operations:

- translation, mirror
- zoom, symmetry
- texture, deletion
- alignment, copy, rotation

Chapter 4

Manipulating primitives

We will describe in this chapter the most frequently used manipulation commands of `xjpdraw`. A full description of all the possible commands is given in the Menus chapter.

4.1 Translating a primitive

Translation can be performed on a primitive or on associated primitives.

4.1.1 Mouse motion

A primitive may follow the mouse motion. First select the primitive by clicking close to its border, then drag the mouse. The primitive will follow the mouse (figure 4.1a).

This will work only if the image of the selection icon display the `with mouse` message (otherwise it displayed `no mouse`). If not, press the L6 key. Note that this command can be customized in the configuration file: you may indicate that the mouse motion will always, never or sometime be used to move primitives (see the "Installation" chapter). The default mode is to enable the translation only when a specific mode is defined by the L6 key: if the message still does not appear verify the assignation of the L6 key in the configuration file. Note that always enabling the translation may have a side effect: when selecting a primitive with the mouse hot spot not exactly on its border, the location of the primitive may change a little bit so that the mouse location is exactly on the border.

Remark: If you translate an associated primitives all the members of the group will also be translated. If you have an important number of primitives the display may be rather slow as `xjpdraw` will display all the associated primitives at all the intermediary positions (the position of the mouse is recorded at regular interval by the X window system). It is therefore better to use the "Large motion" mode.

4.1.2 Large motion

The large motion mode enables to define a point on a primitive and then the new location of this point. First select the primitive by clicking at the desired point, then press the L9 key. Move the mouse at the new location of your point and click: the primitive will be displayed at its new location (figure 4.1b), except if the translation put some part of primitives outside the canvas.

Note that this mode is persistent, meaning that if you click again at a new location the primitive will follow. To cancel this mode you have to press one more time the L9 key. A small window indicates if you are in the large motion mode.

4.1.3 Fine motion

You can also tune very precisely the location of a primitive by selecting it and pressing the R1, R2 keys for moving the primitives along the horizontal axis or the R4, R5 keys for motion along the vertical axis. The amount of motion is one screen pixel (about 0.145 mm).

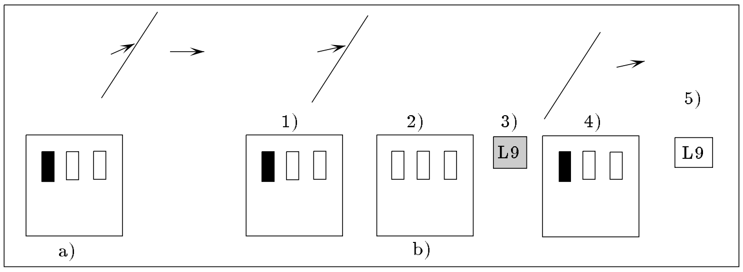


Figure 4.1: how to translate a primitive: a) mouse motion, b) large motion

4.1.4 Precise translation

If you know precisely the amount of translation (in cm) that you want to perform you can use the `Precise translation` command of the `DISPOSITION` menu. After having selected the primitive use the command: `xjpdraw` will ask you the translation values (in cm) in the file window. If the translation is such that primitives will lie outside the canvas it will not be executed.

Warning For a primitive with lines texture (we will see later how to do that) the texture lines are not attached to the primitive. Consequently when translating such a primitive only the border will be translated. To translate the whole group you need to associate the primitive and its texture lines.

4.2 Resizing

You can modify the size of an existing primitive. To do that first select the primitive, then press the L10 key. A small window will indicate that you are now in the resizing mode. To cancel this mode press again the L10 key. The resize mode is persistent but apply only on the primitive which has been selected before running this mode.

If the rulers are displayed (see the `PAGE` menu) the current size of the selected primitive will be displayed. The way you will resize is dependent upon the type of the selected primitive.

4.2.1 Rectangles

Put the mouse close to one of the corner of the rectangle, click on the left mouse button and then drag the mouse. The corner will move to the current location of the mouse. When reaching the correct size release the mouse button.

4.2.2 Lines

Put the mouse close to one extremity of the segment, click and drag the mouse. In this way you can both change the line length and its orientation. If you want to modify only the length of the line without changing its orientation press the `Shift` key before resizing, then drag the mouse until the segment reached the right length and release the mouse. A small window will appear when you press the Shift key and will disappear as soon as you release the mouse button. To change the orientation of a line without modifying its length use the rotation commands. You can also modify a line so that its slope is compatible with the slopes of the standard LaTeX by pressing on the L8 key.

4.2.3 Circles, ellipses, ovals

For the ellipses and ovals you can modify the length of the horizontal or vertical axis. Put the mouse close to the cross which have been displayed on the primitive, click and drag. For the circles you may put the mouse at any location on the border.

4.2.4 Texts

For the text there is no resizing but you can edit an existing text, see page 30.

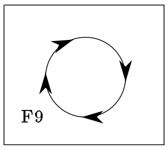
4.2.5 Polygons, B-splines

For the polygons you can either change the position of an existing vertex or introduce a new one. Click and drag: if the mouse location is close to a vertex this vertex will move to the current location of the mouse, otherwise a new vertex will be created (if the number of vertices of the polygon does not exceed 30). This behavior is also valid for any curves (B-spline, polynomial, hand drawing) except for the splines.

4.2.6 Splines

You can change the position of the control points of a spline (indicated by small crosses) by clicking close to them, then dragging. If you are not close from an existing control point the current mouse position is used as a new control point.

4.3 Rotation



Your can rotate primitives (or associated primitives) around any point. Select the primitive, click in the rotation icon (F9). A new window will appear, indicate the rotation angle with the scrollbar and validate the rotation by clicking on `Yes`.

Using this procedure the primitive will rotate around its center (its center of mass for the polygons). If you want to rotate around another point click at the rotation center and press the L9 key before performing the rotation: a small cross indicate the new rotation center. Another way to rotate the primitive is to select the rotation icon, then to drag the mouse with the middle mouse button. The angle of rotation is the angle between the lines going from the rotation center to the initial position of the mouse (first clicked position) and the current one. This angle will be displayed in the rotation window.

You can also indicate a rotation angle in the text window of the panel (angle in degree).

You can also rotate text but the display of rotated text is poor and you will need some special LaTeX package to do that, see the "Installation" chapter. The `End` button enables to dismiss the rotation window.

4.4 Restart

The command `Restart` of the `FILE` menu enable to dismiss the current drawing and start from fresh. You can use this command to make a new drawing although I advise to quit `xjpdraw` and start it again.

4.5 Deleting a primitive

4.5.1 Deleting primitives

First select a primitive and use the `Delete the selected` or `Delete the associated` commands of the `EDIT` menu. A shortcut is to press the R13 key.

4.5.2 The trash can

The deleted primitive are still present in the data structures of `xjpdraw`, only their visibility has been affected. You can see all the deleted primitives by displaying the *trash can* with the `Display the trash can` command of the `EDIT` menu. You can always put back a primitive of the trash can in the current drawing. Select the primitive and use the command `Paste` of the `EDIT` menu. A shortcut is to select the primitive and press the R7 key. The primitive (and all its associated) will disappear from the trash can. You can go back to the current drawing by using the `Hide the trash can` from the menu. Shortcuts to show the trash can in the "Selection" mode is to press the 'a' key (`CTRl-y` in the other modes) and the 'm' key (`CTRL-m` in the other modes) to come back to the current drawing.

The deleted primitives are recorded in the macdraw file, and consequently can be recovered at any time (except if you empty the trash can, see the `EDIT` menu).

4.5.3 Deleting all except ...

To delete a set of primitives in the current drawing but keeping some of of them use the commands `Delete all except the associates` or `Delete all except the selected` of the `EDIT` menu.

4.6 Text editing

To edit a piece of text you can use the `emacs` window. Select the primitive and press the L10 key: note that it is now displayed in the emacs window. Note that a multi-line text is still displayed on one line, the newline character being substituted by a & (normal & character should be entered with a backslash preceding them).

You can move in the `emacs` window using classical emacs command (figure 4.2):

- CTRL-a : move the cursor at the start of the text
- CTRL-b : move the cursor backward one character
- CTRL-d : delete the character preceding the cursor
- CTRL-e : move the cursor at the end of the text
- CTRL-f : move the cursor forward one character
- CTRL-t : invert the two characters before the cursor

You can also use the mouse to put the cursor at a given position in the text by clicking in this position. Long text

emacs:	■qwerty	CTRLa
emacs:	qwerty■	CTRLe
emacs:	qwert■ y	CTRLb
emacs:	qwerty■	CTRLf
emacs:	qwert ■	CTRLd
emacs:	qwetr ■	CTRLt

Figure 4.2: Emacs commands for editing text primitives

will not be displayed in full but the scrollbar over the window enables to display the hidden part: just click with the left mouse button and drag in the desired direction.

Characters are added right after the cursor by typing on the keyboard. To validate the change in the text click in the `emacs` button.

If more than one text is present in the current drawing asking to edit a text will cause the display of a new window. This window contains all the text primitives. Note that each text is followed by the string `%at`, itself followed by the coordinate of the text primitive (in centimeter). The selected primitive is indicated by a `X` after the % sign.

You can move in this window using the classical emacs commands (like `CTRL-n` and `CTRL-p` to go to the next and previous lines) and edit any text in the window. Note that if you have numerous text the window will have scrollbars to enable the visualization of every text primitive. The modifications are validated by clicking in the `End` button of this window.

4.7 Duplicating a primitive

To duplicate the selected primitive use the `Copy` command of the `EDIT` menu or use the R6 key if you are in the "Selection" mode. Note that the new primitive is exactly the same as the initial one, and therefore cannot be distinguished as they overlap.

4.8 Scaled copy

A primitive can be copied with a scaling factor using the command `Copy with scaling` of the `EDIT` menu. First select a primitive and use the command. A new window will appear, with a scrollbar with which you indicate the scaling factor. Each time you move the thumb of the scrollbar the new primitive will be displayed. As soon as you get the desired size validate the change by clicking in the `End` button. You can dismiss this command by clicking in the `Cancel` button. The scaling follow the following rules, according to the type of the primitive:

- text: no change
- line: the start point of the copied segment remains the same as the original and its length is multiplied by the scaling factor.
- rectangle, circle, ellipse, oval: the center of the copy remains the same as the origin and every dimension is multiplied by the scaling factor.
- polygon: the center of mass of the copy remains the same as the original.

Note that if the selected primitive is associated only the selected primitives will be copied with a scaling factor.

4.9 Mirrors, symmetries and zoom

These commands are more precisely described in the Menu chapter.

4.10 Cutting a primitive

To cut a primitive use the scissor tool. Click with the left mouse button close to the primitive you want to cut to select it, then double click (with the middle mouse button) at the cutting point. Note that it is compulsory to first select the primitive.

4.11 Lens

With this tool a small window will appear and move according to the motion of the mouse. It will display, with a scaling factor of 3, the region around the mouse. When using the lens you can also use the R1,R2,R4,R5 key to tune precisely the position of the selected primitive. To make the lens disappear click in any icon.

4.12 The magnet

The magnet is a tool that will attract the current position of the mouse toward another position, according to the magnet mode. The possible modes are:

- `Nearest` : the position is the closest point of the closest primitive.
- `// to near` : only for lines and polygons. It works by pair of points: the first point is the mouse position when starting an action and the second point is the mouse position as soon as the mouse moves. The final position of this point will be such that the segment between the first and second point is parallel to the edge of the closest primitive, the edge being the one the closest to the first point.
- `vertex`: the point will be the closest vertex. A vertex is for the line one of the extremities, for the circles, ellipses and ovals the center and for the rectangle either a vertex or the center.
- `inter`: if the mouse is close to an intersection point between two primitives the assumed position will be this point.
- `Inactive`: the default mode, no change in the mouse position

The magnet is very useful when creating primitives (figure 4.3). Note that, when using the magnet in a active mode, during the creation a primitive may not appear if `xjpdraw` does not find a point which satisfied the desired criterion, you will have to change the mouse position.

Note that at any time you can change the magnet mode by using the `CTRL-n, CTRL-p` key (the mode change in a circular way). This trick can be used for example for creating a line starting in an intersection point and finishing at a vertex of a rectangle: first we put the magnet in `Inter` mode, start our line and then, while keeping the button pressed we use `CTRL-p` to go in `vertex` mode. You can close and open the magnet window by clicking in the magnet tool.

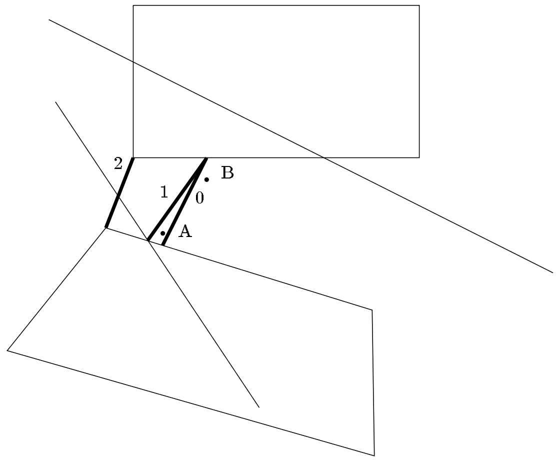


Figure 4.3: Action of the magnet when creating a segment. We first click at A, move the mouse and release the mouse button in B. For line 0 we were in the `nearest` mode, for line 1 we start in the `inter` mode and during the creation we move to the `nearest` mode. For line 2 we were in the `vertex` mode.

Chapter 5

xjpdraw **and Postscript**

In this chapter we will describe the interaction between `xjpdraw`, LaTeX and Postscript.

5.1 Why Postscript?

The idea of separating the text, written as LaTeX commands, and the geometrical primitives, written in a separate Postscript file, may sound strange and unnecessarily complex. Indeed Postscript can also be used to write text. The basic principle underlying `xjpdraw` is that it is more easy to write complex texts, especially mathematics, using LaTeX than by using Postscript. The combination of the two most powerful languages, LaTeX for text edition and Postscript for graphic primitives, was therefore a reasonable choice. Furthermore using Postscript for printing text in a drawing that is intended to be included in a LaTeX main document may have led to some inappropriate discrepancies between the texts appearing in the drawing and the one in the main document.

5.2 How does it work?

To illustrate the combined use of LaTeX and Postscript we will deal with a simple example. We will generate a drawing with a text (the word `toto`) and a line and save this drawing in the default file. `xjpdraw` will generate the three files `xx.ltex`, `xx.macdraw`, and `xx.ps` (or `xx.ps.Z`).

Let's have a look at the LaTeX file `xx.ltex`:

```
% dessin effectue par JpDraw (un programme de JP. MERLET)
% INRIA Sophia-Antipolis,France le: Thu Jun  8 00:18:54 1995
% Directory de creation:/u/merlet
% Createur:merlet
\ifx\cs\tempdima \newdimen \tempdima\fi
\ifx\cs\tempdimb \newdimen \tempdimb\fi
\newcount \jpdrawx
\jpdrawx=100 \tempdima=   1.00cm \tempdimb= 1.0cm
\ifdim \unitlength = 1pt \unitlength=  1.00cm \fi
\ifdim \unitlength > \tempdima \tempdima=\unitlength \fi
\ifdim \unitlength < \tempdima \tempdima=\unitlength \fi
\multiply \jpdrawx by \tempdima \divide \jpdrawx by \tempdimb
\setlength{\unitlength}{\tempdima}

\begin{picture}(   2.0,   2.4)(   1.3,   9.9)
\thinlines
\large
\put( 1.334, 9.889){\special{psfile="`zcat /u/merlet/xx.ps" hscale=\the\jpdrawx \sp vscale=\the\jpdrawx}}
\scriptsize
\put(2.52,10.90){toto}
\end{picture}
\setlength{\unitlength}{1cm}
```

In this file the first lines (which are LateX comments) give some indications on the drawing (creation date, owner, location). The following lines are TeX orders such that a counter `jpdrawx` has a value which is 100 multiplied by the value of the variable `unitlength` if this variable is positive (remember that this variable is used in a picture environment to indicate the correspondence between the unit used in the picture and a real unit length: with a value of `unitlength` of 1 cm the size of our drawing will be 2cm by 2.4cm). The case where `unitlength` is negative will be considered later.

Right after the beginning of the picture environment we have the order to include a Postscript file.

```
\put( 1.334, 9.889){\special{psfile="`zcat /u/merlet/xx.ps"...
```

using the `special` command of LaTeX . Here the inclusion syntax assumed that the dvi driver is `dvips` which understand the `psfile` keyword. Here we have used a little trick allowed by `dvips`. The normal syntax to include the file `xx.ps` will have been:

```
\put( 1.334, 9.889){\special{psfile="/merlet/xx.ps"...
```

But `dvips` enables to apply Unix command on the file: so instead of including the Postscript we include a *compressed* version of `xx.ps` (`xjpdraw` is able to generate directly a compressed version of the Postscript file using the Unix `compress` command, see the "Installation" chapter), therefore reducing the size of the drawing files. The trick is that by using the `zcat` command `dvips` will at run time include the uncompressed version of `xx.ps`.

Now let us look at the final part of the inclusion order. The order `hscale, vscale` are `dvips` order which enable to print a scaled version of the Postscript file. A value of 100 for both order enable to print the Postscript file as it is, a value of 50 will produced the drawing at half size. But note that the value of `hscale, vscale` are defined as the value of the counter `jpdrawx`. Therefore by putting a `unitlength` value different from 1cm (say 0.5 cm) right before inputing the `xx.ltex` file the LaTeX picture **and** the Postscript file will occupy only half the size of the original drawing: consequently the same `xjpdraw` drawing can be used at different scales without changing the drawing itself.

Right after the Postscript file inclusion we have the order to print the string `toto` in a given position.
The Postscript file is the following:

```
%%BoundingBox: 0 0 56 68
% initial value of unitlength : 1.000000
/Ellipse { gsave moveto setgray /savematrix matrix
currentmatrix def currentpoint translate rotate newpath
scale arc closepath savematrix setmatrix stroke grestore 0 setgray} def
/EllipseColor { gsave moveto setgray /savematrix matrix
currentmatrix def currentpoint translate rotate newpath
scale arc closepath savematrix setmatrix fill grestore 0 setgray} def
/EllipseRVB { gsave moveto setrgbcolor /savematrix matrix
currentmatrix def currentpoint translate rotate newpath
scale arc closepath savematrix setmatrix fill grestore 0 setgray} def
/EllipseBRVB { gsave moveto setrgbcolor /savematrix matrix
currentmatrix def currentpoint translate rotate newpath
scale arc closepath savematrix setmatrix stroke grestore 0 setgray} def
/Ligne { gsave moveto currentpoint translate 0 0 moveto
lineto currentpoint stroke moveto closepath grestore} def
/LigneRVB { gsave setrgbcolor moveto currentpoint translate 0 0 moveto
lineto currentpoint stroke moveto closepath grestore 0 setgray} def
/ArrowNoire {gsave 0 setgray moveto currentpoint
translate 0 0 moveto lineto lineto lineto lineto
currentpoint fill moveto closepath grestore 0 setgray} def
/ArrowRVB {gsave setrgbcolor moveto currentpoint
translate 0 0 moveto lineto lineto lineto lineto
currentpoint fill moveto closepath grestore 0 setgray} def
/RectangleNoir {gsave setgray moveto currentpoint
translate 0 0 moveto lineto lineto lineto lineto
currentpoint fill moveto closepath grestore 0 setgray} def
/RectangleRVB {gsave setrgbcolor moveto currentpoint
translate 0 0 moveto lineto lineto lineto lineto
currentpoint fill moveto closepath grestore 0 setgray} def
/Rectangle {gsave moveto currentpoint
```

```
translate 0 0 moveto lineto lineto lineto lineto
currentpoint stroke moveto closepath grestore} def
/OpenPoly {gsave moveto currentpoint translate 0 0 moveto} def
/Li {lineto} def
/ClosePnoir {currentpoint fill moveto closepath grestore 0 setgray} def
/ClosePoly {currentpoint stroke moveto closepath grestore 0 setgray} def
gsave
 /ulx where{pop ulx uly scale} if
 %currentpoint translate
0.3 setlinewidth
[] 0  setdash
 55.899  67.819   0.000   0.000 Ligne grestore
```

Note that this postscript file includes a BoundingBox which indicate the location and dimensions of the drawing. You may also remark that the Postscript file cannot be printed as it is (if you want to do that you will have to insert the sentence %! at the beginning of the file and a showpage at the end). In the following lines of the files some procedures are defined.

The line drawing is done at the last line of the file, using one of the defined procedure:

```
 55.899  67.819   0.000   0.000 Ligne grestore
```

Including the drawing is done by inputing the LaTeX file of the drawing in a main LaTeX source, the use of the Postscript file being transparent for the user (except if the user change the location of the LaTeX, in which case it may be necessary to edit the file to indicate the right location of the Postscript file) and the user will get the drawing of figure 5.1 which has been obtained with the following LaTeX main file:

```
\documentstyle{article}
\begin{document}
\begin{figure}[hbtp]
\begin{center}
\input{xx.ltex}
\caption{The test drawing}
\end{center}
\end{figure}
\end{document}
```

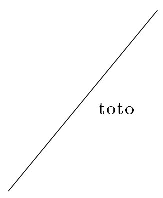


Figure 5.1: The test drawing

Now what happen if the value of unitlength is negative ? The few lines of TeX commands in the picture environment are designed so that in such case the drawing size on the paper (its maximum dimension) will be the absolute value of unitlength. In our example if we affect -4.8cm to unitlength we will get a drawing of size 4cm by 4.8 cm. This can be done simply as xjpdraw is able to compute the size in centimeter of the drawing. Using this knowledge it is very easy to make TeX compute the value of the unitlength, hscale, vscale parameters so that the drawing will fit in a given space.

5.3 The special command

Although the principle of Postscript inclusion described in the previous section may look simple it suffers from a portability problem. Indeed the syntax used is completely dependent upon the dvi driver. In the example we have assumed that the driver `dvips` was used. Indeed the way you include Postscript file may differs from driver to driver. `xjpdraw` propose a simple mechanism to deal with different drivers (see the "Installation" chapter) but some features in a different driver may be such that the Postscript inclusion mechanism will fail, because the Postscript generated by `xjpdraw` is not compatible with the Postscript acceptable by the driver. There is no real solution to this problem and you have simply to figure out what modification has to be done.

5.4 Producing a stand-alone drawing

Having split the drawing in two different files (a LaTeX and a Postscript files) may be a pain if you intend to send a paper including drawings to some third party (for example by e-mail). Indeed you will have not only to send the main document but also all the Postscript files of the drawings.

To solve this problem you may use a feature of the `dvips` driver which enable to incorporate Postscript command directly right after a special command. This is done by using the `Save PS and LaTeX in file..` command of the `FILE` menu. In that case the LaTeX file contains the description of the whole drawing and there is no auxiliary files.

5.5 Textures and texts

We will see in a specific chapter that it will possible to attach to a geometric primitive a gray level or a color texture. Now assume that text primitives have been written on top of a textured primitive. It must be known that in Postscript a textured object completely hide any other primitive which lie below the object and has been described before the object.

This is not a problem with the geometric primitive used in `xjpdraw` (who deal with this kind of problem) but may lead to difficulty when text and graphic are intermixed. For example if in the LaTeX file you include the Postscript file right after all the texts primitives then any text covered by a textured primitive will not appear. This is why the default mode is to include first the Postscript file and then to deal with the text primitive. But this have a side effect: assume that you have on the screen a text primitive which has been covered by a textured primitive and is now hidden. The rule that texts are included after the Postscript file imply that the hidden text will be printed on the paper. The only way to deal with that is to load again the drawing and delete the hidden text primitive. Note that the default mode can be changed by using a command of the `FILE` menu (see page 39).

5.6 Including a Postscript file

A Postscript file generated by an external program (e.g. Maple) can be included in a `xjpdraw` drawing. In that case in order to display the Postscript file `xjpdraw` use a very simple mechanism which is to get an *image* of the Postscript file and include it in the drawing. This lead to an approximated drawing of the file but which is sufficient to make some edition on it.

To generate this image we use the Postscript previewer `gs` which enable to produce a bitmap image of the Postscript file. In order to scale the image it is necessary for the user to indicate the dimension of the Postscript drawing (i.e. the maximum width and height of the drawing). To incorporate the Postscript use the `Include Postscript file` command of the `FILE` menu. You have then to give in the file window the name of the file together with the dimension of the drawing in centimeter (see page 40).

Such Postscript file cannot be edited (you cannot change any of the geometric object in it) but you can translate it and overlap it with `xjpdraw` primitives.

Chapter 6

The menus

In this chapter we describe all the commands available in the menus panel. These menus are `Extension`, `FILE`, `EDIT`, `STYLE`, `Carac`, `PAGE`, `DISPOSITION`, `UNDO`, `BIB`, `Extern`

6.1 Extensions

The button `Extensions` enables to use the limited standard graphic objects of LaTeX or to enable the full use of the Postscript command. The default mode is to use the Postscript commands. If you click in this button the button name will change to `Standard` meaning that from now on you will be able to draw only primitives compatible with LaTeX. Using this mode you can only draw lines with some determined slopes, circles and some specific polygons (the mouse current position is only used to indicate an approximation of the final position of the final position of the primitive). Any other command will be rejected by `xjpdraw` (if you try to use an invalid command `xjpdraw` will ask you if it has to switch back to its full capabilities). The possible LaTeX commands are the one valid for Latex version 2.09. To switch back to the full capability of `xjpdraw` click again in the `Standard` button.

Note that in the user's function mode (see the "User's functions" chapter) there is no verification of the validity of the orders with respect to the graphic possibilities of LaTeX. To save a file which has been drawn in `Standard` mode use the `Save` or `Save under` command of the `FILE` menu. Only a LaTeX file and the `.macdraw` file will be generated.

6.2 The `FILE` menu

6.2.1 New file

This command enables to reinitialize `xjpdraw` for starting a new drawing. If the current drawing has not been saved `xjpdraw` will ask you if it should really start again.

6.2.2 Load

This command enables to load a new drawing. When using this command a new new panel which will appear. In this panel various lists appear displaying the loadable files in the current directory, the directories in the current one and a root directory (denoted by ..).

To choose which macdraw file you want to load (remember that `xjpdraw` only understand this format, not LaTeX or Postscript) they are various ways to give the name of the file you want to load. You can use the new panel: click at the appropriate place in the list if you want to load a file or move in another directory (in which case the loadable files list will be updated). Note that the panel will disappear only after clicking in the `Quit` button. Note also the two buttons which enable to hide or delete the backup files (the files starting with #C). Another way is to give the file name in the file window and press the `Return` key. Note that you load a .macdraw file but you may omit the .macdraw extension (except if your file has more than one extension). If instead of using the `Return` key you use a space or a tab the panel will display all the files whose name start with the given chain (completion process).

You can also indicate that you are looking for a file which is somewhere in the directory or its sub directories by indicating the name of the file in the `Find` text window and pressing the `Return` key. If you don't know the full name of your file you can use the "*" symbol to match an string (for example if you are looking for a `toto-new.macdraw` file you may indicate `toto*` in the Find window). You can also use as first character of your file name the ~ character which will be substituted by the name of your home directory.

If you gave the name of the file in the file window you can omit the `.macdraw` extension if there are no other extension in your file name. Note also that you can load files which have been compressed by `compress` or `gzip` without worrying about the added extension (to load a `toto.macdraw.Z` file, which has been compressed by `compress` just give `toto` as the file name). Note that a slide file (see this chapter) cannot be loaded if you are not in the slide mode.

6.2.3 Save

In this section when a file name has to be given you give it in the text window, followed by a `Return`.

6.2.3.1 Save

Save the drawing using only standard LaTeX commands. This should be done only if you have put `xjpdraw` in its `Standard` mode by clicking in the `Extension` button. The name of the drawing file is the current name.

6.2.3.2 Save under

This command is similar to the previous command except that you have to give a name in the file window, followed by a `Return`. Note that the file name must not have a `.macdraw` or `.ps` extension (if this is the case `xjpdraw` will not save the file and will emit an error message).

If you want to cancel this command just give a empty string as the file name. You can use the ~ character in the first place of the name to indicate your home directory.

6.2.3.3 Save in bitmap file format

This command will generate a bitmap file of your drawing in the file whose name is given in the file window, followed by a `Return`. If you want to cancel this command give an empty string as the file name.

6.2.3.4 Save in Inrimage format

This command is not available in the current version.

6.2.3.5 Save drawing in PS and text in LaTeX

This command will save the drawing in a Postscript file for the geometric objects and in LateX for the text primitives. But the file that you will use in your main document is only the LateX file, see page 33. A shortcut for this command is `CTRL-w`. The file name is the current file name as indicated in the upper part of the `xjpdraw` window (or obtained by using the `CTRL-f` key).

6.2.3.6 Save drawing in PS and text in LaTeX in a file called..

Same command as the previous one but you indicate a file name in the file window followed by a `Return`. If you indicate an empty string the command will be canceled. The ~ character in front of a string will be substituted by the name of your home directory.

6.2.3.7 Save PS and LaTeX in the file

This command enables to save the drawing in a LaTeX file which contain both the text and the Postscript. The file is saved in the current file name.

6.2.3.8 Save PS and LaTeX in file in a file called..

Same command as the previous one but you have to give the file name in the file window followed by a `Return`. An extension different from `.ps` and `.macdraw` should be present. To cancel this command give an empty string as the file name.

The ~ character in front of the file will be substituted by the name of your home directory.

6.2.4 Screen dump

This command is not available in this version.

6.2.5 LaTeX Verification

You can use either `LaTeX verification` or `LaTeX with Postscript verification`. Both commands will generate a complete LaTeX document with name `bidon.ltex` which will contain only your drawing and can be compiled. The difference between the two commands is that the first one generate the drawing using only the `Standard` graphic commands of LaTeX (therefore your drawing should have been done in `Standard` mode) albeit the second one uses the full capabilities of `xjpdraw`, meaning that the geometric object are written in a Postscript file of name `xxx.ps`.

6.2.6 Album

This command will load a drawing and add it to the current drawing. Remember that only `.macdraw` files may be loaded. When using this command a new window similar to the one which appear when using the `Load` command will appear. This window is intended to help you to select the file you want to load. You can also indicate the file name in the file window (you can omit the `.macdraw` extension). You can load `.macdraw` files which have been compressed by `compress` (extension .Z) or `gunzip` (extension .gz). In that case it is sufficient to give the base name of the file.

The primitives of the loaded file will be associated so that you can move them altogether to a new location. It is therefore of good practice to dissociate all the primitives of the current drawing before using this command. Note that you cannot load a file written in slide mode if you are not in this mode.

6.2.7 Put the Postscript at the end

In the produced LaTeX file the Postscript file will be included last. This has the side effect that any text primitive lying over a textured object will not appear on the paper.

6.2.8 Put the Postscript at the beginning

In the produced LaTeX file the Postscript file will be included first. This the default mode of `xjpdraw` because text lying over textured primitives will still appear on the paper. However texts which are hidden on the screen by textured primitives will appear on the paper (you have therefore to delete them).

6.2.9 Read a configuration file

This command enables to read and apply a configuration file (see the "Installation" chapter for the influence of a configuration file). The name of the configuration file followed by a `return` is given in the file window, an empty string cancelling this command.

6.2.10 Compile and print

Using this command a main LaTeX document called `bidon.ltex` containing your drawing will be created, LaTeX compiled and then sent to the printer. This functionality is available only if the `PRINTING` command in the configuration file is present (see the "Installation" chapter).

6.2.11 Compile and preview

Using this command a main LaTeX document called `bidon.ltex` containing your drawing will be created, LaTeX compiled, converted into Postscript and then sent to a Postscript previewer. This functionality is available only if the `GHOST` command in the configuration file is present (see the "Installation" chapter). A freely available Postscript previewer is `ghostview`. A shortcut for this command is `CTRL-e`.

6.2.12 Recompile for the previewer

This command is to be used in conjunction with the previous one. In a first step you use the `Compile and preview` command and keep the Postscript previewer on the screen. This command will generate a new Postscript file of your drawing so that you have only to reload this file to see the most recent version of your drawing. A shortcut for this command is CTRL-x.

6.2.13 Include a Postscript file

This command enables to include in a `xjpdraw` drawing a Postscript file which has been generated by any other software (well almost any...). Give the name of the Postscript file in the file window together with its dimensions (width and height) in centimeters.

To use this functionality it is necessary to have access to the `gs` software and to have updated the `GS` command in the configuration file (see the "Installation" chapter).

Using `gs xjpdraw` will generate a bitmap image of the Postscript file and display it on the screen with its upper left corner at (0,0) (afterward you can translate this primitive). This is a rather lengthy process (the "fast info" window will display what is the current status of the process). Remember also that what you see on the screen is almost twice larger than what will be printed on the paper, therefore the bitmap image may look a little bit strange.

Such included Postscript file will be surrounded by a rectangle on top of which is written `Postscript file`: this rectangle enable to select this kind of primitive. As soon as the image has been displayed on the screen an internal copy will be made, so that redisplaying the drawing will be much more faster. As side effect is that some part of the display may be incorrect if the dimension of the Postscript file exceed the dimension of the visible part of the canvas. It is also necessary that the size of the Postscript file does not exceed the size of the paper sheet represented by the canvas (`xjpdraw` will verify the correctness of the dimensions you gave).

Therefore before including a Postscript file it is necessary to determine its dimensions (i.e. the largest width, height between the visible parts of the drawing). This can be done by looking for the `BoundingBox` of the file. The two first numbers indicate the location of the lower left corner of the drawing, the two following numbers being the coordinates of the upper right corner. Note that the unit in the BoundingBox may be pt (i.e. 2.54/72 cm). However it may be necessary to print the drawing, some Postscript file having no BoundingBox or an incorrect one. If your Postscript drawing is too large you will have to reduce it. To do that you must edit your file and add a Postscript `scale` order, with the scaling factors in x, y before the order. For example:

```
0.5 0.5 scale
```

will produce a half-size drawing. To determine where you have to put the order you must know that generally a Postscript file is separated in two components: a dictionary where Postscript procedures are defined (their names start with a `/`) and the drawing itself. You must put the scale order right after the dictionary and before the drawing.

Sometime some differences may appear between the location of the Postscript drawing on the screen and on the paper (which may be disturbing as you may have put some others primitives at some precise location). The trick is to complete your drawing and then to correct the insertion point of the Postscript file in the LaTeX file until you have the right drawing. For example if you have in your LaTeX file:

```
\put( 1,10){\special{psfile="/u/merlet/postscript_externe.ps"}}
```

you must correct the coordinates (1,10). Another problem is that after having included a drawing with such primitive in the main LaTeX document you get some trouble for printing the pages following the drawing. Well there is no sure way to solve that but you may look at your Postscript file and try to suppress the first line of the file if it start with something like `%!` and any `showpage` appearing at the end of the file.

No edition can be made of the Postscript file itself but any `xjpdraw` primitive can be put over it, enabling to hide part of the drawing, putting some legends on it etc (see the drawing on page 107)...and this special primitive can also be translated. As any other `xjpdraw` drawing you can still increase or decrease the size of the drawing using the process described at page 43.

6.2.14 See all drawings

This command enables to see all the drawings of the current directory. They are displayed in a reduced size, four drawings at a time. A new window enables to switch to the next group of four or to cancel this command. The name of the files is indicated on the screen. Note that `macdraw` files which have been compressed by `compress` or `gunzip` are also displayed.

6.2.15 Save, Read customization file

The overall context of `xjpdraw` (like the presence of rulers, grid, thickness and style of the line) may be saved in a special file and loaded at will to come back to the same context. Note that you can indicate a customization file in your configuration file, so that you will start `xjpdraw` in always the same mode, see the "Installation" chapter.

6.2.16 Slide mode, New slide

Refer to the Slide mode chapter.

6.3 The EDIT menu

All the commands of this menu deal with action on existing primitives.

6.3.1 Delete

This section deal with different ways to delete primitives. Note that the deleted primitives are put in the trash can and therefore can be recovered, except if you have emptied the trash can. Note that the deleted primitives are also saved in the `.macdraw` file and therefore can be recovered even after quitting `xjpdraw` (except if you empty the trash can). To delete a set of primitives you have to associate them and delete one member of the group.

6.3.1.1 Delete the selected primitive

This command will delete the selected primitive, meaning that this primitive will be no more displayed on the screen. A shortcut for this command is to press the R13 key.

6.3.1.2 Delete all except the associated

This command will delete all the primitives which are not associated.

6.3.1.3 Delete all except the selected

This command will delete all the primitives except the selected.

6.3.2 Eliminate the hidden primitives

In some cases primitives may be completely covered by other primitives which are in front. Consequently these primitives are not necessary and should be deleted. This can be done using this command where `xjpdraw` uses a crude and lengthy algorithm to figure out what primitives should be deleted.

6.3.3 Copy

6.3.3.1 Copy the selected

This command enables to duplicate the selected primitive. Note that the duplicated primitive is exactly the same as the initial one and therefore both primitives completely overlap. A shortcut for this command is to press the R6 key.
.

6.3.3.2 Copy with scaling

This command enables to create a scaled copy of the selected primitive. See page 31 for the process of choosing the scale factor.

6.3.4 The trash can

The trash can is the place where are put every deleted primitives.

6.3.4.1 Paste

This command is used when you have displayed the trash can and selected a primitive in it. Using this command the selected primitive will be back in the current drawing. A shortcut for this command is to press the R7 key.

6.3.4.2 Show the trash can

This command enables to display the trash can i.e. the place where are put all the deleted primitives. Then, using the `Paste` command you can recover a deleted primitive. If your are in the "Selection" mode a shortcut for this command is to press the "a" key, `CTRL-y` in the other modes.

6.3.4.3 Hide the trash can

This command is used after having displayed the trash can to come back to the current drawing. If your are in the "Selection" mode a shortcut for this command is to press the "m" key, or a `CTRL-m` in the other modes.

6.3.4.4 Empty the trash can

If you have deleted a lot of primitives you may end up with a big `.macdraw` file for a small drawing (indeed the deleted primitives are still saved in the file). To avoid this problem you can reduce the number of primitives by emptying the trash can. A shortcut for this command is `CTRL--`. To get the number of primitives in the trash can use `CTRL-u`.

A small side effect of this command is that the texture eraser (see the "Texture" chapter) may not work (but you can still delete the lines texture one by one).

6.3.4.5 Exchange the visible and invisible

This command enables to switch the trash can and the current drawing: every primitive in the current drawing are deleted and put in the trash can and every primitive in the trash can is undeleted. This command can be useful to create two different drawings from an initial one by splitting the drawing.

6.3.5 Associate-Dissociate

6.3.5.1 Associate all

Using this command all the primitive of the current drawing will be associated. In the "Selection" mode a shortcut for this command is to press the "A" key. In any other mode `CTRL-a` should have the same effect.

6.3.5.2 Associate the primitives of same type

This command enable to associate or dissociate primitives of the same type. When using this command a new window will appear with a list of primitive types and a button to indicate if you want to associate or dissociate. Select first the associate or dissociate mode by clicking in this button, then click in the primitive type.

6.3.5.3 Dissociate all

With this command you dissociate all the associated primitive. After dissociation the drawing will be redisplayed. In the "Selection" mode a shortcut for this command is to press the "D" key, in the other mode `CTRL-d`.

6.3.5.4 Dissociate the selected

The selected primitive will be dissociated. In the "Selection" mode a shortcut is to select the primitive, then press the "d" key.

6.3.5.5 Associate all at...

This command enables to associate all the primitives in a certain position with respect to the selected primitive. You can associate all the primitive on the left, on the right, over, under the selected primitive. A new window will appear with buttons to indicate your choice.

6.3.6 Selective selection

Using this command the selection mode of `xjpdraw` will switch to a restricted selection mode in which only a certain type of primitive may be selected. A new window appear with a list of primitive types. Click in the desired type. A message in the `xjpdraw` canvas will indicate that you are in a restricted selection mode. To go back to the general selection mode click in the "Cancel" item of the list.

6.3.7 Change unitlength value

A standard paper sheet is a 14.5cm by 14.5cm page. By using this command this value is multiplied by the given factor (consequently this is mean to increase the size of the drawing). The value of `unitlength` is saved in the `.macdraw` file and therefore loading such file will update the value of the factor.

This feature is now superseded as a new value for `unitlength` can now be indicated in the main document, see page 47.

6.3.8 Formula

With this command you can display almost every function of the form $y = f(x)$ by giving the analytical form of the function. The function will be displayed as a polygonal line whose vertices correspond to the value of the function at a discrete interval. A new window will appear where you will:

- give the function in symbolic form like:

  ```
  y=sin(x)
  y=0.3*sin(3*(log(x)+1)-2)
  ```

- give the initial and final values for x (`X0`,`X1`)
- give a step Δ for the x value (the function will be computed at $X0$, $X) + \Delta$, $X0 + 2\Delta$ and so on)
- give the location of the initial point ($x = X0$,$y = f(X0)$) on the screen (in centimeter)
- click on `End`

For the symbolic form of the function you can use the following operations: $\times, /, +, -, **$. The following mathematical functions are recognized: *exp*, *log*, *log*10, sin, cos, tan, *atan*, *asin*, *acos*, *sinh*, *cosh*, *tanh*, *sqrt*. The constant π (typed as `pi`) is also recognized.

Figure 6.1 shows the drawing obtained when using the formula mode with:

```
y=1.5*sin(x**2)
```

for `X0=0`, `X1=5`, $\Delta = 0.01$, the initial point being put at (3,10).

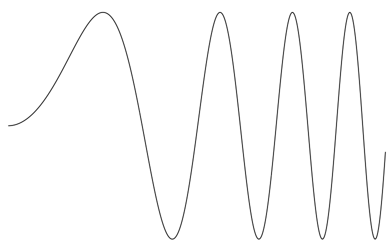


Figure 6.1: Drawing the function $y = 1.5\sin(x^2)$

The formula mode can also be used as a desk calculator by giving only the function form, the value of `X0` (if you don't give this value a zero value will be assumed) and clicking in `End`. But don't give the value of any other parameters because you will need to cancel the mode and start again.

In the formula window you have also a text window where the name of a program file can be given for execution (see the "Programming" chapter).

6.3.9 Align

This command is used to align primitives with respect to a fixed primitive. First you have to associate all the primitives, the first one being the fixed primitive. Then you use this command: a new window will appear with various menu buttons to indicate which type of alignment you will perform. The name of the menu button indicates which part of the moving primitives is to be aligned and which motion will be performed (either horizontal or vertical) and in the menu the various items indicate which part of the fixed primitive is used as reference. When using a command where a margin is mentioned the default value of the margin is 1 mm. You can change this value by using the button `Change the margin`: you have then to give the new value of the margin in the file window. If a value of 0 is entered the items of the align menus will no more include order with a margin.

6.3.10 Put/Strip border

These two commands enable to add (dismiss) a border to the associated primitives which have been textured (in gray or in color) A shortcut for this command is `CTRL-b`.

6.3.11 Smooth a polygon

This command uses the vertices of the selected polygon as control points for a spline curve (figure 6.2).

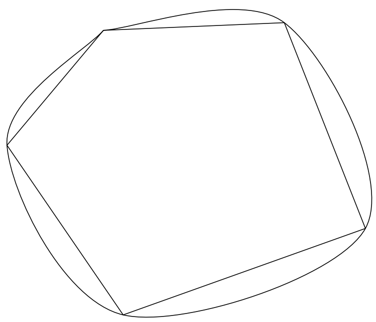


Figure 6.2: A polygon and its smooth version

6.3.12 Create a polygon

This command enables to creates a polygon using the associated lines. The order of the edge of the polygon respect the order in which the lines have been associated. You may either chain the end point of a line to the start point of the next line or, by using the `proximity order`, chain the end point of a line to the closest extremity of the next line (figure 6.3).

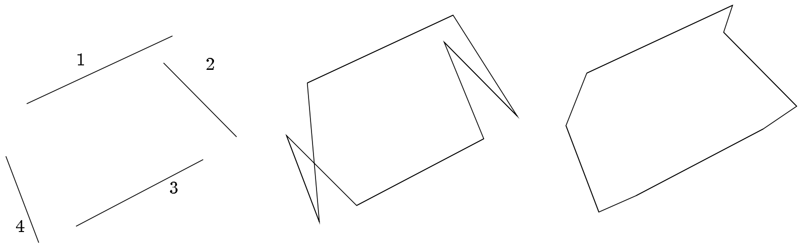


Figure 6.3: The segments are associated in the indicated order. On the right the resulting polygons, without and with proximity order.

6.3.13 Duplicate progressively

To use this command you have first to associate two primitives of the same type. This command will create n primitives of the same type slowly evolving from the first primitive to the second. The number n is to be given in the file window. An example is given in figure 6.4 and in page 108. Note that you cannot duplicate polygons. To cancel this command

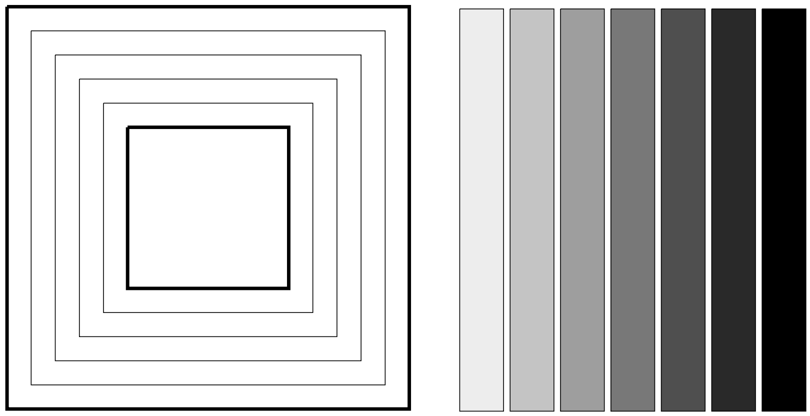


Figure 6.4: Using the `Duplicate progressively` command. The most external and most internal square have been associated and we have asked for a 4 duplications. On the right the light gray rectangle on the left and the black rectangle on the right have been first created.

give an empty string as number.

6.3.14 Add an arrow to the polygon

This command enable to add an arrow vector to the selected polygon. The arrow is added at the closest extremity of the polygon when the polygon is selected. A shortcut for this command is to press the "f" or "F" key when you are in the "Selection" mode. This command works also for lines.

6.3.15 Cut the polygon

Using this command the edges of the associated polygons will be transformed into segments, the polygons themselves being sent to the trash can. The number of newly created primitives is checked so that the maximum number of primitives is not exceeded.

6.4 The `STYLE` menu

This menu enables to select the style of the following text primitives. You can choose among: roman (default), bold, italic, underlined.

. The current style is indicated by a cross. Note that you can always use the LaTeX commands to change the style of a text even if you are in roman style.

6.5 The `Caract.` menu

6.5.1 8pt...24pt

These commands enables to change the size of the text primitives. The default value is 10 pt and the current value is indicated by a cross.

6.5.2 Existing texts

This command enables to change the size and style of a set of text primitives to the current style and size. Before using the commands you should choose you new style and size. The set of text primitives is either:

- `For all`: This command changes the style and style of all the text primitives.
- `For the associated`: This command changes the style and style of all the associated text primitives.
- `For the selected`: only the selected primitive has the new style and size.
- `For the last`: the last defined text primitive is changed.

6.5.3 Compute size

Using this command `xjpdraw` will compute the exact size of all the text primitives by using a LaTeX compilation. The bounding box of the text primitive will be updated. This may be a rather lengthy operation, the "fast window" will indicate the number of primitives remaining to be treated. Starting from version 4.2 this operation is now faster has LaTeX is run from the start in background. This command and the following are installation dependent and may not be available (see the "Installation" chapter).

6.5.4 Compute size of associated

This command is similar to the preceding one except that only the size of the associated primitives will be computed.

6.6 The PAGE Menu

The commands of this menu relates in general to the overall presentation of the drawing on the screen.

6.6.1 Redisplay all

This command enables to redraw all the primitives. A shortcut of this command is top press the F5 key. This command can also be used to reset `xjpdraw`.

6.6.2 Rulers

With this command you will display all around the screen rulers graduated either in pixel or in centimeter.

In this mode many useful information will be displayed. The current position of the mouse is displayed in the "Cursor" window. Various informations on the selected primitive will be displayed in the "fast info" window (dimension, color, line style). A shortcut for this command in the "Selection" mode is to press the "r" key (or `CTRL-r` in any mode).

6.6.3 Size of drawing

This command will display the current size of the drawing in centimeter, pixel.

6.6.3.1 Real size, Screen size

Remember that the full canvas of `xjpdraw` represents a 14.5cmx 14.5cm paper sheet. Therefore the size of an object present on the screen is almost twice the size it will occupy on the paper. The `Real size` command will present on the screen the drawing as it will look like on the paper. The `Screen size` command enables to go back to normal size. Note that a drawing should be saved on in the `Screen size` mode.

6.6.4 Grid

You can use the command to display (hide) a grid in the screen. By default the distance between the grid lines will be one centimeter. A shortcut is `CTRL-q`.

6.6.5 Change the grid size

This command enables to change the distance between the lines of the grid. You have to give the distance in centimeter in the file window.

6.6.6 Change time between checkpointing

At regular time interval `xjpdraw` will makes a backup of the current drawing: it will save a backup `.macdraw` file whose name is the name of the current drawing file name preceded by a #C. The time between the backup is adjusted according to the number of primitives in the drawing (more primitives lead to a higher time interval between the backups). Using this command you can modify the time interval between two backups.

6.6.7 Zoom

This command enables to zoom the current drawing. A new window will appear: in this window you are able to choose a scaling factor using a scrollbar (Figure 6.5). You can zoom along the x axis, the y axis or both (default mode). You can also zoom all the primitives or only the associated one. If you zoom all the primitives the drawing will be centered in the canvas after the zoom. If you zoom only the associated primitives the center of the bounding box surrounding the primitives will be kept at the same position. Note that a zoom along a given axis does not change the topological structure of the primitives (a circle will still be a circle after zooming along the x or y axis). The zoom is validated

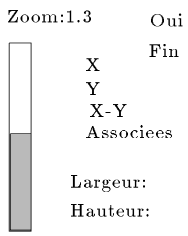


Figure 6.5: The zoom window

by clicking in the `Yes` window. In this mode you can also indicate the final dimension of your drawing by giving its values in centimeter in the text window of the zoom window. Usually the zoom factor is adjusted so that you cannot use a scaling factor which lead to a primitive going out the `xjpdraw` screen. If this happen use the `Put primitive back in the screen` command to put back all the primitives in the screen.

Note that a similar effect can be obtained without modifying the drawing by indicating a new `unitlength` value just before inputing the LaTeX file in the main document like:

```
\unitlength 0.5cm
\input{toto.ltex}
```

This command will reduce the size of the drawing by half. Note however that a too much high value of `unitlength` is not recommended as the size of the line is also increased by a similar factor. If you put a negative value for `unitlength` the maximal dimension of the drawing on your paper will be the absolute value of `unitlength`

6.6.8 Center

6.6.8.1 Center all

The commands `Center all` and `Center the associated` will put the primitive, if possible, in middle of the visible part of the canvas otherwise, for too big drawing, in middle of the full canvas. In the "Selection" mode a shortcut for the `Center all` command is to press the "c" key. In any mode `CTRL-c` should work too.

6.6.8.2 Center the text

To center a text primitive in a graphic primitive you have first to associate both primitives (only them should be associated). Then you can use the `Center text` command (note that it may be necessary to redisplay the drawing to see the centering effect). A text centered in a primitive will always been put in the center of this primitive even if the graphic primitive is moved. Centered text have a small cross sign on their border. Text can be centered only in rectangles, circles, ellipses and ovals. A shortcut for this command is `CTRL-t`. Of course it is possible to cancel the centering of a text.

6.6.9 Get outside primitives in canvas

By using certain `xjpdraw` commands (e.g. translation) some primitives could be sent outside the limits of the canvas. This command will move these primitives back in the screen.

6.6.10 Bounding box without texts

If you does not compile the text primitives and have rather long texts which produce only a small text on the paper (e.g. mathematical symbols) `xjpdraw` may wrongly estimate the size of your drawing and consequently on the paper sheet your drawing may be not well centered. This command told `xjpdraw` to compute the bounding box without taking into account the text. The default mode is to take into account the text primitive. Note this may be needed only if you don't compile your texts.

6.6.11 Measure

With this command a new window will appear indicating the Euclidean, horizontal and vertical distances together with the angle between the current position of the mouse and the last position where the left mouse button has been pressed in the "Selection" mode. This window is dismissed by selecting again this command.

6.7 The `DISPOSITION` Menu

6.7.1 Plan

The commands `Foreground`, `Background` enable to put the selected primitive in front or in rear of the drawing. Not that a selected primitive will appear always in foreground although its real position may be in the background. Use the `Foreground` command to definitely put the primitive in the foreground (in "Selection" mode a shortcut is to press the "p" key). When you have selected a primitive and have used the `Background` command the whole drawing will be redisplayed (a shortcut for this is to press the "s" key in "Selection" mode and `CTRL-s` in the other modes). These commands work also for the associated primitives.

6.7.2 Rotate left/right

The commands `Rotate Left, Right` perform a ± 90 degree rotation of the selected primitive, around its center.

6.7.3 Mirror

Using the mirror commands the selected (or associated) primitives will be drawn as they will be seen in a mirror put in the position defined in the command.

6.7.4 Magnetic grid

The magnetic grid is an invisible grid whose nodes will attract the principal points of primitives when they are created. For example when drawing a segment the start and end point of the segment will be positioned at the closest nodes of the grid, whatever is the current position of the mouse. This can be useful for aligning primitives at creation time. A shortcut for activating/deactivating the magnetic grid is CTRL-i.

By default the distance between the modes of the magnetic grid is 0.1mm but this value can be modified. Primitives which have been created before activating the magnetic grid can also be aligned on the grid afterwards.

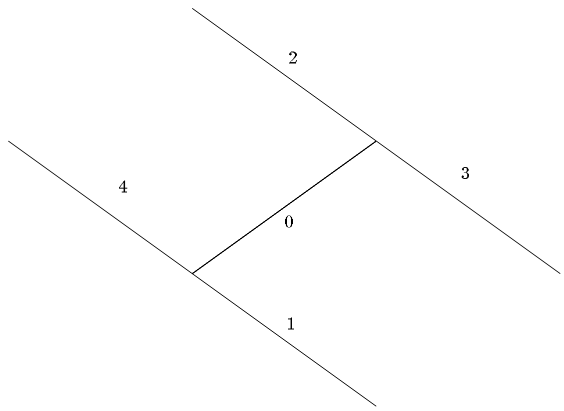


Figure 6.6: Action of the mirror: 0=initial primitive, 1=bottom mirror, 2=top mirror, 3=right mirror, 4=left mirror

6.7.4.1 Align selected on magnetic grid

This command will modify the position of the selected primitive with respect to the grid. The primitive will be modified so that important points lie on the nodes of the grid.

6.7.4.2 Align associated on magnetic grid

Same action except it is applied on all the associated primitives.

6.7.4.3 Align all on magnetic grid

Same action except it is applied on all the primitives.

6.7.4.4 Modify grid

This command enables to modify the distance between the modes of the magnetic grid. Using the scrollbars of a new window you can adjust the distances by giving value either in pixel or in centimeter.

6.7.4.5 Suppress undo

A undo mechanism enables to cancel up to the sixteenth action before reaching the current state of `xjpdraw`. However this mechanism is rather crude and may cause some delay when creating a drawing. This command will render ineffective the undo mechanism.

Note also that this mechanism is automatically suspended as the number of primitives reach a determined threshold defined in the configuration file (see the "Installation" chapter). In this case you can reactivate it by choosing this command (the name in the menu will change to `Activate undo`)

6.7.4.6 Draw in reverse order

Using this command primitives in background (foreground) will be displayed in foreground (background), which can be useful for locating an hidden primitive. Use the `Redisplay all` command to redisplay you drawing in the current order.

6.7.4.7 Perpendicular, tangent lines

Before using these commands you need to select a primitive. The purpose of these commands is to create new segments which are either perpendicular or tangent to the selected primitive at the point where you have selected the primitive.

The length of the new segments have a default value of 1.45 centimeter but this length can be adjusted.

Note that by using these commands you can draw a curve parallel to another one. For example if you have a spline curve defined by control points you will have to draw perpendicular segments at each control points, whose extremity will define the control points of your new curve (trick: to ensure that you use really the extremity of the segments as control points put the magnet in the "vertex" mode before creating your new spline).

6.7.4.8 Line length

This command enables to define the length of the perpendicular and tangent segments. Give the length value in centimeter in the file window.

6.7.4.9 Divide the associated lines

This command enables to split the lines in smaller lines with equal length. Give the number of smaller lines in the file window.

6.7.4.10 Precise translation

This command enables to translate the selected primitive (or the associated primitives) of a specified amount. Give the translation values (x, y, separated by a blank) in the file window.

6.7.5 Symmetries with respect to the associated line

For using this command you need to associate first the line with respect to which you want to complete the symmetry. Then you associate all the others primitives. By using this command all the associated primitives will be transformed to their symmetrical with respect to the line (figure 6.7).

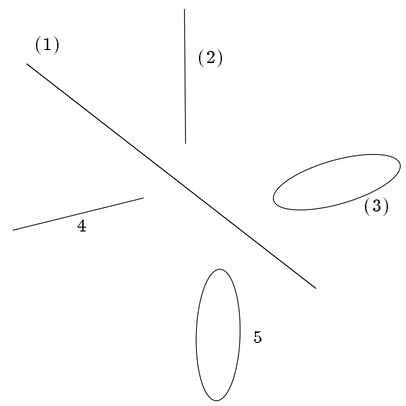


Figure 6.7: We have associated first line 1, then segment 2 and ellipse 3 (just before we have copied primitives 2 and 3). After using the symmetry command we have obtained primitives 4 and 5.

6.8 Undo

This button enables to cancel the previous actions (up to the sixteenth). Just click into the button until you have the desired drawing.

The undo mechanism may sometimes render the time response of `xjpdraw` too important. Consequently it is automatically suspended as the number of primitives reach a determined threshold (fixed in the configuration file). This mechanism can also be suspended (activated) by using the `Suppress undo` (`Activate undo` command of the `DISPOSITION` menu.

6.9 The `BIB` menu

We will see that a user can define complicated graphic objects by writing a C like program. As some of these modules may be of interest for various people they may be grouped in a common directory. Furthermore a user may want to define its own module which are put in another directory. Assume that in the configuration file the location of the common directory is indicated with a `# BIB` command and that the user has a local configuration file (indicated by the environment variable `LOCAL_JPDRAW`) with a `# BIB` command locating its own modules directory. Under that assumption the `BIB` button will display all the module names of these directories and the corresponding programs will be executed when their name is selected.

Note that to be recognized the module name should have an extension `.fi`. In "Selection" mode you can also use of the program of the menu by simply typing in the canvas the number of the module you want to execute (as its appear in the menu, starting at 0) followed by the letter "P".

6.10 The Extern Menu

`xjpdraw` can transmit to an external program the current state of the drawing and then read the result of this program which may define new primitives. For example an external program can be used to compute the intersection of the associated primitives and write the result in a manner such that `xjpdraw` will create a polygonal line corresponding to the intersection.

All the available external programs are defined in the configuration file and the name of this program will appear in the menu attached to the `Extern` button. Provided that the user has respected the program definition the user can apply the selected external module.

The external modules are defined in the configuration file by a line looking like:

```
# EXTERNE "name" program primitive-type input-type output-type
```

`name` is the name of the module as it will be displayed in the `Extern` menu. `program` is the real name of the program. `xjpdraw` will look for such a program first in the directory defined in the configuration file by the command:

```
# LIB_EXTERNE ...
```

then, in case of failure, in the directory defined by the `LIB` command.

`primitive-type` is a keyword which indicate on which type of primitive the program should be applied. The following keywords are possible:

- `all` : All visible primitives should be transmitted to the program.
- `selected`: only the selected primitive is transmitted
- `associates`: only the associated primitives are transmitted

When running an external module `xjpdraw` will write the defined primitives in a file under a specified format. This format is defined by the keyword `input-type` which can be:

- `polygon`: The rectangles, ovals, circles, ellipses, arcs of circles, polygons are written in the file (the text and line primitives are ignored). Each primitive is preceded by the sentence `new polygon` followed by a list of coordinates (in centimeter) defining the primitive (every primitive is approximated by a polygon).
- `typee`: each primitive is preceded by a sentence indicating its type. Texts are preceded by `new text` followed by the coordinates of the start of the text, followed by the text itself between ". Lines are preceded by `new line` followed by the coordinates of the start and end points. Circles are indicated by `new circle` followed by the coordinates of the center and the radius. Arcs of circle are indicated by `new circle_arc` followed by the coordinates of the center, the radius and two angles in degree defining the start and end point (anti-clockwise oriented). Ellipses are introduced by `new ellipses` followed by the coordinates of the center, the lengths along the two main axis and the rotation angle of the ellipse (in radian).

`output-type` is the keyword which indicate under which format the external module will write the result: it take the same value as `input-type`.

The external module must take as argument a file name and output the result on the standard output, in a similar format as the input file. When running an external module `xjpdraw` will write the input data in a file, call the external module with the file name as argument and will redirect the standard output to a file. Then it will read the result file and display the new primitives.

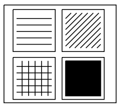
6.11 Help

See page 8.

Chapter 7

Texturing

7.1 Texture



Using this icon we will put texture on primitives. Note that text and ovals primitives cannot be textured (although the color of the border of an oval can be specified) and that associated primitives will also be textured when one member of the group is textured.

7.1.1 How to texture

Before texturing you need to select a primitive, then to activate the icon (note that you can select a primitive without leaving the texture mode). A new window will appear: the icons in the window indicate which type of texture you will apply. You may either choose texture made of lines or gray level and colored texture.

7.1.2 Line textures

Line textures are created by adding new line primitives (this means that you have to take care of the number of primitives). A side effect is that the line texture are not attached to the primitive: if you move or delete the primitive the lines texture will remain at the same place. If you want to move both the primitives and its lines texture you have to associate them.

7.1.2.1 Texture types

You can choose either horizontal, vertical, both or tilted lines to draw a texture. These types appear clearly in the icon images. Choose one of the types and then press the `Yes` button in the icon panel (close to the texture icon) to validate the texture. You can evidently apply different textures on the same primitive.

Standard parameters are defined for these textures: the distance between the lines is 0.5cm and the slope of the tilted lines is 45 degree (figure 7.1). These values can be modified.

7.1.2.2 Create your own texture

To create your own texture click in the icon displaying `votre texture` in the texture window.

In the new window which will appear you will find three scrollbars enabling to modify the texture parameters: one is devoted to modify the distance between the lines of the horizontal texture, the second one the distance between the vertical or tilted lines, the last one for giving the slope of the tilted lines. You can also indicate the number of lines you want, `xjpdraw` updating the texture parameters. When you have fixed all the parameters click in the `End` button. You can also use back the standard value of the parameters by clicking in the `Standard` button.

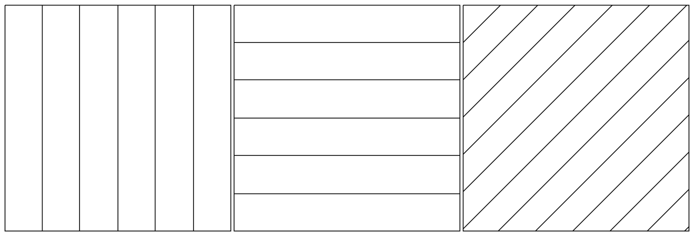


Figure 7.1: Standard line texture in a square

7.1.3 Gray level and color texture

7.1.3.1 Choosing a texture

If you select the black square icon in the texture icon you will be able to apply a gray level or a color texture. In the new window which will appear you will find four scrollbars. The first scrollbar enables to define a gray level texture, the level of gray being defined by the number in the scrollbar, from 1 (black) to 15 (white), see figure 7.2. Note that the white texture will appear a very light gray on the screen (so that you can determine that a texture has been applied on the primitive).

Before choosing a gray level you have to indicate on which element will be applied the texture by clicking in the `Black border` button. You can indeed texture the interior of the primitive while keeping a black border (`Black border` mode, being the default) or texture the interior without having a border (`No border` mode) or texture the border without filling the interior (i.e. changing the color of the border, `Border alone` mode).

After having defined your texture (which is applied during the process as soon as you change the value of the scrollbar) you can validate the texture by clicking in the `End` button.

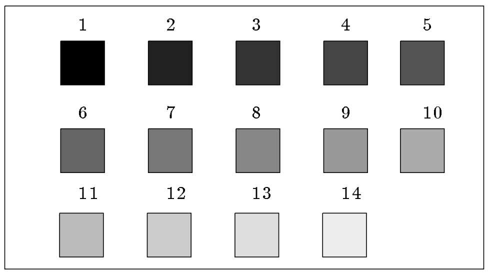


Figure 7.2: Gray level

As for the color texture a similar process is to be used with the three remaining scrollbars which enable to define the red, green, blue components of the color. Note that a shortcut for having directly the gray level window is to be in the "Selection" mode and to press the `CTRL-g` key.

7.1.3.2 Masks

The white texture may be used to hide part of other primitives i.e. to be used as a mask. For example you can create arc of ellipses by hiding with a white polygon some part of an existing primitives, as in figure 7.3 (where the border of the white polygon has not been suppressed to show where the polygon is located).

Note that another way to create arc of ellipse is to use the scissor, see page 31.

7.1.4 Eraser

The eraser enables to suppress the texture of a primitive. To use the eraser click in the icon image containing `gomme` in the texture window, then click in the `Yes` button in the icon panel, close to the texture icon. Note that this will not work for the line texture if you have emptied the trash can: in this case you need to delete individually all the lines.

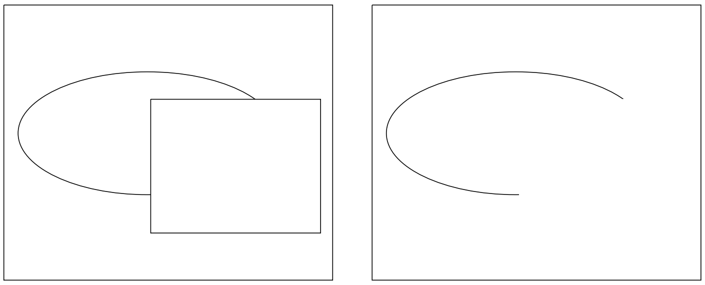


Figure 7.3: Using a white polygon to create arcs of ellipse.

7.1.5 Holes in a texture

If you want to texture a primitive with lines, but you have primitives in it that you don't want to texture, just associate these primitives and apply the texture (Figure 7.4). This will only work for rectangles, ellipses, circles and polygons.

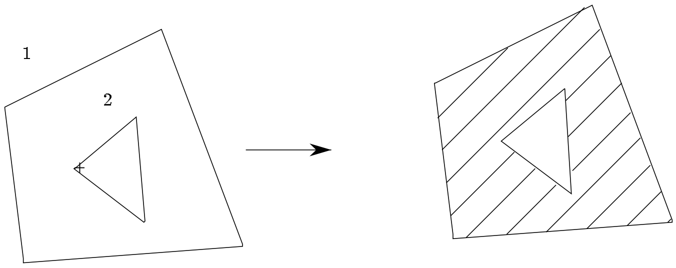


Figure 7.4: Holes in a texture: 2 is associated (note the '+' on its border) then 1 is textured.

7.1.6 Permanent texture

It may be interesting to define a texture which will automatically be applied to the newly created primitives. To do that you have first to go to the `Define` mode, by using the button close to the texture icon. Then you texture a primitive, click in the `Permanent` button. From now on every newly created primitive will have the defined texture. To cancel this option click in the `Temporary` button.

Chapter 8

The slide mode

In this mode it is possible to create multi pages document. Hence it is very convenient to write slides. To create the first page of such a document run `xjpdraw` with the argument `-slide`.

In this mode the presentation of `xjpdraw` is slightly different than in the normal mode. First the canvas represent now a page of 18cm by 23 cm, these values being modifiable through the configuration file. If you want some others values put in your configuration file the orders:

```
# SLIDE_WIDTH 15
# SLIDE_HEIGHT 22
```

which indicate (it is only an example) that the width of your slide is 15 cm and the height 22cm.

To help position the text in the slide mode a grid is drawn in the canvas and the magnetic grid is activated, each node of the grid being drawn in red (the magnetic grid can be deactivated with CTRL-i).

Now you can create any `xjpdraw` primitives as in the normal mode (you can use normal macdraw file through the album, although it may be necessary to reduce their size).

Note that the total number of pages and the number of the current page are indicated in a window at the bottom right of the main window. To avoid manipulation error you cannot exit of this mode without exiting `xjpdraw`.

8.1 The following pages

To create new pages in your document use the command `New Slide` in the `FILE` menu. You can put primitives of one page into another one by copying them in the original page, destroying them, moving to a new page and bringing the primitives back from the trashcan.

8.2 Moving around

To go to a given page you can use the `Goto page` command of the `FILE` menu or, if you are in the Selection mode, you can use the keys `Return` (moving one page forward) or `Backspace` (one page backward).

8.3 Deleting a page

To delete the current page use the command of the `FILE` menu.

8.4 Adding a page

To add a page after the current page use the command of the `FILE` menu.

8.5 Album

When you use the album two cases are possible:

- the album file is a slide document: all the pages of the album file will be added to the document, the first page being included in the current page (consequently if you want to put the album file at the end of your document you have first to create a new page and then use the album in the new page).
- the album file is a normal document: it is simply added to the current page

8.6 Texts

In this mode the text primitives are treated in a different way as in the the normal mode. If you have enabled the computation of the exact size of the text primitives the distance between two characters will be adjusted so that the overall size of the primitive will correspond exactly to its real size (except in the case where this is not possible, especially when using math symbols). The same thing will happen if you are looking at the document in the real size mode.

8.7 Creating a full document

When a slide file is saved `xjpdraw` will create one picture environment for each page. But for creating slides it may be of interest when using the `Verify` command or the `Compile and preview` command to encapsulate each page in a specific environment and use a special style file. This can be done via the configuration file. You have to define via the command `EN_TETE_SLIDE` the start of your document. For example:

```
# EN_TETE_SLIDE  \documentstyle[trans]{article}\n \begin{document}
```

Note the special newline character which shall always be followed by a space. This order enable to define the LaTeX style. Now if each page is encapsulated in a special environment you may use the orders `START_SLIDE` and `END_SLIDE`. The first order is followed by a string which will be put before the picture environment and the second order is also followed by a string which will be put after the picture environment. For example you may have:

```
# START_SLIDE \begin{ltransp}
# END_SLIDE \end{ltransp}
```

So using the `Verify` or the `Compile and preview` command you will get a document looking like:

```
\documentstyle[trans]{article}
 \begin{document}
%some special xjpdraw order

%Start of page 1
\begin{ltransp}
\begin{picture}
votre 1ere page
\end{picture}
\end{ltransp}
%End of page 1
....
\end{document}
```

Note that it is necessary to adjust your `SLIDE_WIDTH` and `SLIDE_HEIGHT` command so that the size of the picture environment is compatible with the LaTeX style.

8.8 Main differences with the normal mode

The main difference is that all the orders dealing with all the primitives (for example `Associate all`) act only on the primitives of the current page. There is however an exception : the trashcan. In the trashcan all the deleted primitives of all the pages are present.

Chapter 9

Language: the user's functions

One of the main drawbacks of the existing drawing softwares is that it is very difficult to interact with them by other mean than using the mouse. One of the main motivation for the creation of `xjpdraw` has been to provide various means to tackle with this problem. We have already seen one of them: the use of the `Extern` command with which an external program may perform a treatment on the existing primitives, `xjpdraw` being able to use the result to create new primitives.

In the two following chapters we will introduce two others means to interact with `xjpdraw`. In the next chapter we will describe a C-like graphical language enabling to *compute* a drawing (this language may also be used to create new primitives and transformation which can be directly called from `xjpdraw`, see the `BIB` menu). The language described in this chapter uses purely numerical data and is called the *user's function*.

In this language a drawing is described by numerical data which are preceded by simple keyword to indicate the type of primitive that has to be drawn. The user's function can, for example, be used to draw complex 2D curves. To do that you have to create a file beginning with:

```
mode polygone
-1 10
```

followed by numerical data defining the x, y coordinates of points of the curve. Then it will be possible to feed this file to `xjpdraw`, and later on to draw axes, grid etc...

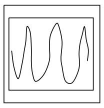
9.1 User's Function

9.1.1 Introduction

Dealing with purely numerical data is a difficult problem when using existing drawing softwares (for example using numerical data describing a planar curve). This problem may be solved in some cases by using specific software like `gnuplot` which enable to treat the data and output a Postscript file which can then be included in a LaTeX main document. But this approach has some drawbacks:

- their lack of generality
- the impossibility to edit the result
- a discrepancy between the text font used in the drawing and the font used in the main document

9.1.2 Using user's function



When you click in this icon you will be able to use a user's function. Note that for the user's function the x axis is defined as horizontal from left to right and the y axis as vertical from bottom to top.

9.1.3 Loading a user's function

To load a user's function you need first to define and select a rectangle primitive which will enclose the drawing defined in the function. This rectangle will be used to scale your drawing, meaning that you don't have to worry about the length units used in the function. After having defined your rectangle click in the function icon. A new window will appear, the *function window*. In this window you have icons: the first one is used to load a user's function, the second one to draw a grid on a new user's function, the third one to define the parameters of the grid. Click in the first icon of this window.

Another window will appear: give the function file name in the text window, followed by a `Return`. Your drawing will appear on the screen.

You can also load a user's function when starting `xjpdraw`. The syntax is:

```
xjpdraw -X x1 -Y y1 -F name [-E xmin xmax ymin ymax][-O x2 y2]
```

where `x1, y1` represent the width and height (in centimeter) of the box which will enclose your drawing. `name` is the name of your function file (a variant for `F` is `Fp` with which the enclosing rectangle will not be visible, but will be present in the trash can).

Optional arguments are:

1. `E xmin xmax ymin ymax` which define the size of the enclosing box in the user's units.
2. `O x2 y2` which indicate the coordinates in centimeter of the upper left corner of the bounding box. The default value of these coordinates are 2.

Another way to load a user's function is provided in the other command language, see the next chapter.

9.1.4 Definition of the user's function

The language defining an user's function uses the concept of *mode*: a mode describes how the following numerical data are to be used for creating a drawing.

A user's function file may start by the number of lines of the file, although this is not necessary if your file start with a mode order. Then the file will contain various mode commands followed by numerical data. We will now describe the various modes.

9.1.4.1 Comment mode

This mode enables to put comments in a user's function. The comment end is indicated by a #.
Example

```
mode comment
This is a useful comment here!
#
```

9.1.4.2 The `mode connecte`

This mode enables to draw polygonal line whose vertices coordinates are written in a file. This command uses in fact various keywords after the word `mode`, indicating the line style of the polygonal line:

standard	vector	dashed
connecte	connecte_vecteur	connecte_pointille
connecte_epais	connecte_vecteur_epais	connecte_pointille_epais

The word `pointille` is used to indicate dashed lines and the word `epais` to indicate thick line. The mode `connecte` is the default mode.
Example: In this example we will draw a polygonal line starting at (0,0), then going through (1,1) and (2,1).

```
mode connecte
0 0
1 1
2 1
```

If a large number of points has to be drawn it is more convenient to use the `mode polygone`

9.1.4.3 The `mode deconnecte`

This mode is used to draw not connected segments. The numerical data following the mode describe the coordinates of the start and end point of each segment like in:

```
mode deconnecte
x1 y1
x2 y2
.. ..
```

As for the `mode connecte` various keywords can be used after the word `mode` to describe the line style of the segment:

standard	vector	dashed
deconnecte	deconnecte_vecteur	deconnecte_pointille
deconnecte_epais	deconnecte_vecteur_epais	deconnecte_pointille_epais

The example given in the `mode connecte` section can also be written as:

```
mode deconnecte
0 0
1 1
1 1
2 1
```

which will lead to the creation of two line primitives instead of one polygonal line.

9.1.4.4 The `mode cercle`

This mode enables to define circles and ellipses in a user's function. The syntax is:

```
mode cercle
x y (coordinates of the center)
rx ry (the radii along the x and y axis)
```

Note that instead of giving the dimension of the primitive in the user's unit you may also give them in centimeters or millimeters (no inches, sorry!) like `0.2mm`. In that case this part of the user's function will not be used to compute the size of the scale factor between the size of the bounding box and the size of the data. It is therefore not possible to have a user's function with only such order. But it is sufficient to enclose them in a rectangle by using the `mode rectangle`.

Here also you have various keywords to describe the line style of the primitive:

standard	dashed
cercle	cercle_pointille
cercle_epais	cercle_pointille_epais
cercle_noir	

Using `cercle_noir` the primitive will have a black interior.

For example to draw a unit circle centered at (0,0) the file will contain:

```
mode cercle
0 0
1 1
```

You can also create ellipses by using this mode. In that case the `rx ry` values are the dimensions of the x and y axis of the ellipse. You may also create rotated ellipses by putting a negative number as the `rx` value, `rx ry` being then followed by the rotation angle in degree. As example to draw an ellipse centered in (0,0), with dimensions 100, 50 rotated anti-clockwise with an angle of 45.5 degree you will have to write:

```
mode cercle
0 0
-100 50 45.5
```

9.1.4.5 The `mode arc`

This mode enables to create arcs of circle. The syntax is:

```
mode arc
x y (coordinates of the center)
radius radius
angle1 angle2
```

where `angle1, angle2` are the angles (in degree and in the range [0,360]) enabling to define the start and end points of the arc. The arc will always go from the start point to the end point in a anti-clockwise manner.

The keyword after the word `mode` may be:

standard	dashed
arc	arc_pointille
arc_epais	arc_pointille_epais
arc_noir	

Example

```
mode arc
1 7
1 1
10 200
```

will define an arc centered at (1,7) of radius 1, starting at a location corresponding to an angle of 10 degree and going at a point with an angle 200 degree. The corresponding arc is presented in figure 9.1. If we have inverted the 10 200 values we will have get the other arc.

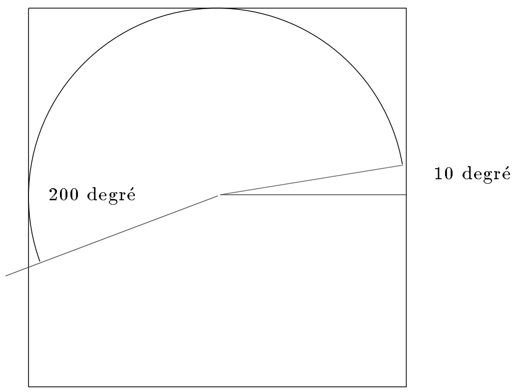


Figure 9.1: An arc of circle defined in an user's function

9.1.4.6 The `mode croix`

This mode enables to draw crosses in the drawing, the dimension of the crosses being given in centimeters or millimeters. The syntax is:

```
mode croix
x y
width height
```

where `x, y` are the coordinates of the cross center. The width and height must be given by number followed by a length unit like `2mm 0.5cm`.

9.1.4.7 The `mode texte`

This mode enables to put a text in the drawing. The syntax is:

```
mode texte
x y
type string
```

`x y` are the coordinates of the lower left corner of the text. `type` is used to indicate the size and style of the text. It's an integer whose unit value indicates the size (0=10pt, 1=12pt, 2=14pt, 3=16pt, 4=18pt, 5=24pt) and the value of the tens the style (0=roman, 1=bold, 2=italic). For example 13 indicate a 16 pt text in bold font. If you indicate a number greater than 100 and lower than 10000 the two first numbers will be a rotation angle for the primitive. To indicate a negative value of the angle add a 1 before the number you will have used for a positive angle. For example 13000 indicates a 10 pt, roman text rotated by -30 degree. If you give a negative value the text will also be underlined.

The string cannot have more than 256 characters and no new line character.

Examples

11 : bold, 12 pt -12: bold, 14pt, underlined
4511 : bold, 12pt, rotated 45 degree
14511: bold, 12pt rotated -45 degree

```
mode texte
0 5
11 here we get a bold, 12 pt string
```

9.1.4.8 The `mode rectangle`

This mode is used to draw a rectangle. The syntax is:

```
mode rectangle
x y
width height
```

where `x y` are the coordinates of the bottom left corner. **Example**

```
mode rectangle
0 0
1 2
```

Various keywords can be used after the word `mode`:

standard	dashed
rectangle	rectangle_pointille
rectangle_epais	
rectangle_noir	

If the dimensions of the rectangle are given with length unit then the two numbers following the `mode rectangle` sentence are the coordinates of the center of the rectangle.

```
mode rectangle
0 0
0.3mm 0.3mm
```

9.1.4.9 The `mode polygone`

This mode enables to create polygonal lines and polygons. Its syntax is:

```
mode polygone
number of vertices type
x1 y1
x2 y2
```

`type` is an integer describing the type and line style of the primitive as indicated in the following table:

type	0	1	2
	thin polygon	thick polygon	dashed polygon
type	10	11	12
	thin polygonal line	thick polygonal line	dashed polygonal line
type	3	13	14..19
	thick dashed polygon	thick dashed polygonal line	dashed polygonal line

Different dashed style is used for type 12 and 14 to 19.
Remember that you cannot have more than 30 points in a polygon.
Example We draw here two polygons, the first one having 5 vertices, connected by thin line, the other one having 4 vertices connected by thick lines.

```
mode polygone
5 0
3 4
5 6
3 2
1 5
2 3
mode polygone
4 1
2 1
2 3
3 2
1 1
```

The number of points in the polygon may not be known in advance. In that case indicate -1 as number of vertices. This is particularly useful to draw a curve as `xjpdraw` will produce as many polygon primitives as necessary.

9.1.4.10 The `Bspline` mode

This mode enables to create b-spline curve by giving the coordinates of the control points. The syntax is:

```
mode bspline
n1 n2
x1 y1
x2 y2
.....
```

where `n1` is the number of control points, `n2` the line style encoded as for the polygons and `xi yi` the location of the control points.

9.1.4.11 The `Spline` mode

This mode is used to draw spline curve. The syntax is similar to the `mode bspline` except for the keyword `bspline` which is substituted by `spline`.

9.1.4.12 The `epaisseur_trait` mode

This mode enables to define the line thickness for the primitive that will follow. The line thickness is given using length unit like in:

```
mode epaisseur_trait 0.1cm
```

where cm may be substituted by mm. You can also indicate the thickness in user's unit by omitting the length unit, like in:

```
mode epaisseur_trait 0.1
```

User's function example

In this example we will draw part of a cone.

```
mode polygone
-1 10
-1.000000 0.000000
-0.750000 -0.935414
-0.500000 -1.224745
-0.250000 -1.369306
0.000000 -1.414214
0.250000 -1.369306
0.500000 -1.224745
0.750000 -0.935414
1.000000 0.000000
mode deconnecte
-1.000000 0.000000
-0.500000 4.000000
1.000000 0.000000
0.500000 4.000000
mode cercle
0 4.000000
0.500000 0.707107
mode texte
0 0
0 here is a cone
```

In this example the `mode polygone` is used to draw an arc of ellipse which is the basis of the cone, the `mode deconnecte` to draw the side of the cone and the `mode cercle` to draw the upper ellipse.

9.1.5 Errors in the user's function

A common mistake when using a user's function is to have a drawing which cannot fit in a rectangle (for example the curve y=1). A good programming habit is to put always a rectangle in the user's function to avoid this type of problem.

`xjpdraw` will check the user's function file at run time and emit error messages if needed. There is two different types of messages:

- non fatal messages: a valid mode has been found but the numerical data cannot be used to create a primitive (e.g. 0 or 1 point in `mode polygone`. This is only a warning but does not impair the creation of the primitive.

- fatal error: `xjpdraw` cannot recover from the error. This may be due to an unknown mode, syntax error in the use of a mode. The drawing will not be created and the line number of the error will be approximatively indicated.

9.1.6 Drawing grids

For every drawing created with a user's function it is possible to draw a grid indicating significant values on the rectangle.

Right after having created your drawing click in the second icon of the function window to create a grid with standard parameters: the distances between two lines of the grid will be 2 cm and the x, y values of the grid lines will be indicated with two decimals.

If these parameters are not satisfactory you can modify them. You have to do that right after having created your drawing. Click in the third icon of the function window. A new window will appear, enabling you to define your own grid (figure 9.2).

In this window you may define:

- the distances between the grid lines, either in the user's unit, in centimeter or by indicating the desired number of lines.

- the format used to write the values of the coordinates along the grid lines. Give the total number of characters and the number of decimals (e.g. 6.2 leads to two decimals and a total of 6 characters, 3.0 indicate no decimals). According to the input data `xjpdraw` will try to figure out an optimal size.
- the x and y coordinates of the first horizontal and vertical lines of the grid (which cannot be lower than the minimum values of the input data, which are displayed into the information panel,right after having created the drawing).
- you can also decide if you want to have grid lines along the x only or along the y axis (and putting them either on the left or on the right). The default mode is to have both grid lines and the y coordinates on the left side of the drawing.
- tiny marks equally spaced between the grid lines on the edge of the enclosing rectangle. By default no such mark will be drawn.

Note that the values given in the file windows should in general be validated by pressing the `Return` key. By default grid lines will be drawn all over the bounding box. You may choose to have only small marks on the edge of the box by clicking in the `No grid` button.

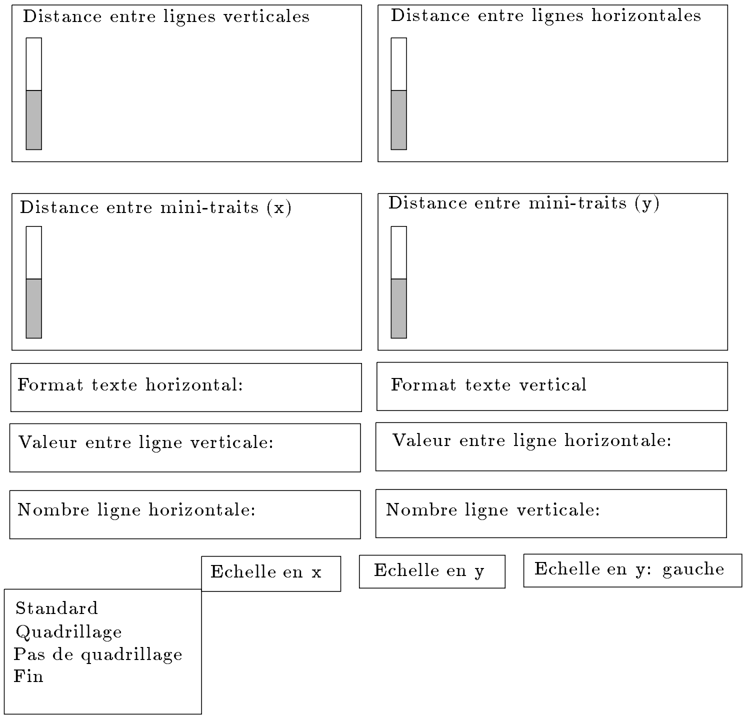


Figure 9.2: The window enabling to adjust the grid parameters

After having loaded a user's function it is possible to load another one without modifying the scaling factor. To do that click in the `No scale change` button before giving the file name. You may select to keep the scaling factor only along the x or y axis.

9.1.7 Fixed scale

Instead of using the automatic scaling mode you can indicate your own scale by giving the minimal and maximal values of the enclosing rectangle in user's unit. You have to do that right before loading a user's function by giving in the corresponding text window four values fixing the minimum, maximum along the x and y axis (e.g. 0 3 1 4 indicates that the extremal values along the x axis are 0, 3 and 1,4 along the y axis). Then you can give the name of your file. Remember that you can also perform this operation when starting `xjpdraw`, see page 60.

9.1.8 Color and gray level

9.1.8.1 Color

You can also specify color in an user's function. This is done by slightly modifying the keyword `mode` which will be substituted by:

```
modeRVB-(red)-(green)-(blue)
```

`(red), (green), (blue)` are integers in the range 0 to 255 which specify the color used for the primitive. With this syntax the interior will be filled with the color, the border being black. If you don't want to draw a border use

```
modeRVB--(red)-(green)-(blue)
```

To put a color onto the border of a primitive without filling it interior use:

```
modeRVB+-(red)-(green)-(blue)
```

9.1.8.2 Gray level

You can also use a gray level texture by substituting `modegris` to the word `mode`, followed by an integer in the range 1 to 100 which indicate the percentage of white in the texture (1 is black, 100 full white). For example with `modegris50` you will have a primitive with a black border, filled with a half gray texture.

If the integer is negative the primitive will have no border. If the integer is preceded by the symbol`+-` only the border will have the color. **Example**

```
mode connecte
0 0
0 9
9 9
9 0
0 0
modegris90 polygone
-1 0
5 5
3 4
4 2
modegris90 rectangle
0 0
1 2
modegris-95 rectangle
2 2
1 2
modegris80 cercle
4 4
1 1
modegris-80 cercle
5 5
0.5 0.5
modegris90 cercle
7 7
0.4 0.7
modegris-90 cercle
8 8
0.4 0.7
```

9.2 Mode summary

mode	1st argument	2nd argument	other arguments
comment	text..#	-	-
connecte	-	-	x,y
connecte_vecteur	-	-	x,y
connecte_pointille	-	-	x,y
connecte_epais	-	-	x,y
connecte_vecteur_epais	-	-	x,y
connecte_pointille_epais	-	-	x,y
croix	x,y center	height, width (mm or cm)	-
deconnecte	x,y start	x,y end	x,y segment
deconnecte_vecteur	x,y start	x,y end	x,y segment
deconnecte_pointille	x,y start	x,y end	x,y segment
deconnecte_epais	x,y start	x,y end	x,y segment
deconnecte_vecteur_epais	x,y start	x,y end	x,y segment
deconnecte_pointille_epais	x,y start	x,y end	x,y segment
epaisseur_trait	thickness[unit]	-	-
cercle	x,y center	x, y dimensions, rotation angle	-
cercle_pointille	x,y center	x, y dimensions, rotation angle	-
cercle_epais	x,y center	x, y dimensions, rotation angle	-
cercle_pointille_epais	x,y center	x, y dimensions, rotation angle	-
cercle_noir	x,y center	x, y dimensions, rotation angle	-
arc	x,y center	x, y dimensions	angle 1, angle 2
arc_pointille	x,y center	x, y dimensions	angle 1, angle 2
arc_epais	x,y center	x, y dimensions	angle 1, angle 2
arc_pointille_epais	x,y center	x, y dimensions	angle 1, angle 2
arc_noir	x,y center	x, y dimensions	angle 1, angle 2
texte	x,y	style text	-
rectangle	x,y lower left corner	width height	-
rectangle_pointille	x,y lower left corner	width height	-
rectangle_epais	x,y lower left corner	width height	-
rectangle_noir	x,y lower left corner	width-height	-
polygone	vertices number(or -1), type	x,y vertices	-
bspline	nb control points, type	x,y vertices	-
spline	nb control points, type	x,y control points	-
epaisseur_trait	line thickness	-	-

Chapter 10

The programming language

A drawing or graphical order can also be defined using a C-like command language. This language uses symbolic variables and has the main instructions of a classical programming language: arrays, loop, conditional expressions but also some graphical instructions. This is this language which is used to create the module which can be found in the BIB menu, see page 50.

For example if you have to make a drawing with waved lines you can define a module which will perform such task. An example of such program is given now:

```
info "Wiggly line, from A to B,in "Selection" mode,Left=abort"
input_mouse 2
%1=distance from %v0 to %v1
%2=atan((%v1[1]-%v0[1])/(%v1[0]-%v0[0]))
%3=0.075 /* amplitude of the wave */
%4=40 /* period of the wave */
%6=pi/(2*%4)+%1/2 /*u2*/
%5=%6-%1
#0=Formule %3*sin(%4*x), x between %5,%6 , step 0.0075 , start %v0
rotate #0 around %v0 , %2
quit
```

With this program we have asked `xjpdraw` to emit an information message as soon as this program is activated (line `info`), then to wait for two clicked points on the screen (line `input_mouse`, then to compute the distance between the two clicked points, to draw a sinusoid starting at the first clicked point and stopping at a given distance. Then the polygonal line is rotated around the first clicked point with an angle such that the end point will be the second point.

As you may see with a few command lines we have been able to defined a rather complex graphical task. We will now see more in details the structure of the language.

10.1 Introduction

Contrary to the user's function the language does not use only numerical data but also has variables. These variables are used to perform computation which are in turn used to compute primitives. The length unit used for the primitive is the centimeter.

The programming language module can be executed in different manners. One possibility is to execute them when starting `xjpdraw`. This is done by running:

```
xjpdraw -F [file name]
```

You can also run `xjpdraw` in silent running mode i.e. without using a graphical display by running:

```
xjpdraw -FI [program file name]
```

You can also indicate parameters to the program by using:

```
xjpdraw -F [program file name][[arguments]]
```

like in:

`xjpdraw -F essai.f[1,2,3] xjpdraw -F essai.f[tata,toto.ltex]`

Another possibility is to execute the file by giving the file name in the `Formula` panel.

10.2 Variables and expressions

10.2.1 Elementary expressions

An elementary expression is a collection of numbers and operators which respect the C language except for the termination (no termination is needed in the language) like, for example `3.14*5.6/(7.8/2)`. The elementary operators which are recognized are:

```
- + / * **
```

The number π is also recognized and it is written as `pi`.

The following mathematical functions are also recognized:

```
exp ceil floor log log10 sin cos tan atan asin acos sinh cosh tanh sqrt rint atan2
```

`rint` return the closest integer of a float. `atan2` take as argument a vector %v (we will see how to define vector in the following section) and compute the arctangent function on the ratio defined by the second component over the first component of the vector.

An elementary operator can be constructed by combining elementary operators like in:

```
3.14*sin(pi/3.0)*cosh(3**3)
```

10.2.2 Variables

In the programming language the name of a variable is usually defined by a % followed by an integer. You can affect to a variable the value of an elementary expression like in:

```
%1=3.14*sin(pi/3.0)*cosh(3**3)
```

But you can also use a variable to construct an elementary expression like:

```
%2=3.14
%1=%2*sin(pi/3.0)*cosh(3**3)
```

You can define an alias to a variable by using the `alias_variable` command followed by a string between quotes and the variable number it will be substituted for, as for example:

```
alias_variable "toto" 0
```

From now on you can use the string {`toto`} as a substitute for the variable `%0` (which can still be used).

10.2.3 Vector

A vector is, in the programming language, an array of two floats. A vector name start by a `%v` and is followed by an integer. The component of a vector can be affected by

```
vector=( [elementary expression], [elementary expression])
```

as, for example:

```
%v1=(3.14*sin(pi/3.0)*cosh(3**3), 3)
%v2=(%10,%11)
```

Note that the right size of this expression is a valid vector. The component of a vector can be recovered and used as an elementary expression. The component are numbered from 0 to 1. For example:

```
%1=%v1[0]
%2=%v1[1]
```

You can also affect an elementary expression to a component of a vector:

```
%v1[1]=%2
```

10.3 Conditional instructions

With the conditional instructions you can execute some part of a program only if some conditions are fulfilled.

10.3.1 Conditions

Condition are written using the following syntax:

```
condition: ( [variable] [operator] [variable])
```

where the following operators are available:

- > : strictly greater
- >= : greater or equal
- < : strictly lower
- <= : lower or equal
- == : equal
- <> : not equal

A particular operator is the negation whose syntax is:

```
(! [variable])
(! condition)
```

which will return 1 if the variable or the condition is equal to 0 and 0 in all the other cases.

10.3.2 Combination of conditions

More than one condition can be combined using logical operator to build a new condition.

```
condition: condition [operator] condition
```

where the possible operator are:

- `&&` : logical and
- `||` : logical or

10.3.3 If

A conditional instruction can be build using the `if` instruction. The syntax is the following:

```
if ( condition ) bloc
     instructions to be executed
end_bloc
```

as in the following example:

```
if ((%1 <0) && (%2>=0))bloc
if (%0>10) bloc
%1=%1+10
end_bloc
end_bloc
```

where a conditional instructions is embedded in another conditional instructions.

10.4 Loop

Loop are used to repeat a bloc of instructions until some condition is satisfied.

10.4.1 For

The `for` command can be used to define a loop. Its syntax is:

```
for [variable] from [variable] ; condition ; step [variable] bloc
instructions to execute
end_bloc
```

An example follow:

```
for %2 from %10 ; (%2 <= %11) ; step %20 bloc
for {za} from {za1}; ({za} < {za2}) ; step 0.1 bloc
%1=%1+10+pi
end_bloc
end_bloc
```

The first loop will be executed while the variable %2 is not strictly greater than the variable %11, the variable %20 being added to %2 each time the program reach the end of the loop.

10.4.2 While

Another loop command is `while` whose syntax is:

```
while condition bloc
        instructions to execute
end_bloc
```

as in the following example:

```
%0=0
while (%0 <5) bloc
        print(%0,"\n")
        %0=%0+1
end_bloc
```

which will print on the screen the list of the five first integers.

10.5 Branching instructions

It is possible to interrupt the sequential treatment of the program and to do a branching at some specific part of the program. First you have to define a *label* and put it at the location where you may want to branch. A label is a string with only letters followed by a ":". To do the branching use the `goto` command followed by the label string between quotes, like in:

```
deb:

while (%0 <5) bloc
        print(%0,"\n")
        %0=%0+1
end_bloc

if (%1==1)bloc
        %1=%1+1
        %0=0
        goto "deb"
end_bloc
```

10.6 Characters string

A character string is included between quote. Some character or sequence of character may have a special meaning like: `\n \r \t \b \\`

The concatenation operator "." can be used to build variable strings like:

```
%0=10
"toto".%0
```

The resulting string is `toto10`. If the variable is not an integer the closest integer will be appended to the file name.

10.7 Printing

Printing variable and strings on the standard output can be done using the `print` command whose syntax is:

```
print([character string])
print([character string],[character string])
print([character string],[variable],[character string])
print([character string],[vector],[character string])
print([primitive number])
```

Note that some sequences have a special meaning:

- `\n` : newline
- `\r` : line feed
- `\t` : tab
- `\b` : backspace

Examples

```
print("hello world!\n")
print("variable 0 value:",%0,"\n")
print("toto".%0,"\n")
```

The last form of the `print` instruction is used to print some data on a primitive.

10.7.1 Printing format

The way data are printed can be controlled using format similar to the one used in the C programming language and defined with the command `format` followed by a string defining the format. For example:

```
%0=1.23456
format "%4.1f"
print(%10,"\n")
```

will print "1.2" in the window.

10.8 Line style

The command `set style` is used to set the line of the primitive which will be created from now on. Its syntax is:

```
set style [thin][thick][dashed][thin dashed][thick dashed]
set style [vect][double_vect]
```

10.9 Inputing values

In a program you can ask to the user some numerical data. This is done by using the command `input_value` which has to be affected to a variable or a vector:

```
%10=input_value
%v0=input_value
```

Note that `xjpdraw` will ask for this value in the file window.

10.10 Inputing points

Instead of using fixed variable as input to a program you may ask to the user to introduce point with the mouse, whose coordinates may be used as vector. This is done using the command `input_mouse` followed by the number of points the program will wait for. Every instruction of the program before the command is executed, then `xjpdraw` will emit a message asking to input points with the mouse.

You have then to go in the "Selection" mode and click at the desired position with the left mouse button. A point can be canceled by clicking with the middle mouse button. The full program can be canceled with the right mouse button. The coordinates of the input points are put in the vector `%v0,%v1` and so on.

As soon as the number of points reach the specified value `xjpdraw` resume the execution.

10.11 Information message

Message can be emitted in the information window using the command `info` followed by a string between quotes. It is good practice to put an `info` command before an `input_mouse` command in order to inform the user how much points he has to introduce.

10.12 Creating primitives

Primitives may be labeled using a # followed by an integer. This label has to be affected whence creating the primitive.

10.12.1 Texts

Texts are created by:

```
text at [vector] [string] size [value]
#[value-label]=text at [vector] [string] size [value]
```

where `value` indicates the size of the text (8, 10, 12, 14, 16, 18, 24). and `vector` the coordinates of the lower left corner of the text.

10.12.2 Circles, Ellipses, Arcs of circle

Circles can be defined using the following syntaxes:

```
circle center [vector] radius [value]
circle toward [vector] [vector] [vector]
ellipsis center [vector] radius [value1] , [value2]
```

In the first case the first `vector` define the center. In the second case the circle going through the three points whose coordinates are defined by the three vectors is drawn. The third syntax enables to draw ellipse, the width and height of the ellipse being defined by the values.

It is possible to affect to a variable the radius of a circle by:

```
[variable]=radius [primitive number]
```

Arc of circle can be drawn using the following syntax:

```
arc center [vector] radius [value] from [value] to [value]
```

The last two values define the two angles in degree which are necessary to define an arc.

Another possibility is to define the arc by its center and the coordinates of its two extremities, always going anti-clockwise.

```
arc center [vector] at [vector] to [vector]
```

10.12.3 Rectangle

The syntaxes are:

```
rectangle from [vector] to [vector]
rectangle from [vector] length [vector]
```

In the first case we give the coordinate of two corners of the rectangle. In the second case you give the position of the upper left corner and the width and height as component of the vector.

10.12.4 Line

The possibles syntaxes are:

1. `line from [vector] to [vector]`: the start and end points coordinates are defined

2. `line from [vector], vector [vector], length [value]`: the start point, a unit vector and a length are defined

3. `line from [vector] [side] perpendicular to [primitive number], length[value]`: the start point, and length are specified. The line will be perpendicular to the primitive defined in the command. As two such lines may be defined `side` defines the desired perpendicular, `left` or `right`.

4. `line from [vector] parallel to [primitive number], length [value]`: the start point and length are defined. The line will be parallel to the primitive defined in the command.

10.12.5 Polygon

To create a polygon put the coordinates of the vertices in successive vectors (for example from `%v0` to `%v10`) and use the command:

```
polygon from %v0 to %v10
```

10.12.6 Curves

10.12.6.1 Spline

To create a spline put the coordinates of the control points in successive vectors (for example from `%v0` to `%v10`) and use the command:

```
spline from %v0 to %v10
```

You can also use directly the vertices of an existing polygon as control point as in:

```
#1=polygon from %v0 to %v10
spline #1
```

10.12.6.2 Formula

You can draw a curve of type $y = f(x)$ using the command:

```
Formule f(x), x between x1,x2 , step [value] , start [vector]
```

as in:

```
%v0[0]= 2
%v0[1]= 10
Formule sin(x), x between 1,10 , step 0.1 , start %v0
quit
```

This program will draw the curve $\sin(x)$ for x from 1 to 10, with a step of 0.1, the first point being drawn on the screen at (2,10).

10.13 3D primitives

3D primitives can be drawn with the programming language.

10.13.1 Defining the view

Before drawing a 3D primitive it is necessary to define the point of view. This is done by defining the features of a 3D sphere that will contain all the 3D primitives and the 2D circle which correspond to this sphere. This is done by using the `view` command as in:

```
view ( 0 0 0 7) ( 6 10 10)
```

which indicate that the sphere of center (0,0,0) with radius 7 will correspond to a circle centered on the screen at (6,7) with radius 10cm.

10.13.2 3D circle

A circle in 3D is defined by:

```
cercle3d x1 y1 z1 radius r1 (psi1 teta1)
```

where `x1, y1, z1` are the coordinates of the center, `r1` its radius and `psi1, teta1` are the Euler's angles in degree defining the normal to the circle (`psi1` is the rotation angle around the z axis, `teta1` the rotation angle around the new x axis)

For example:

```
cercle3d %82 0 0 radius %81 (0 90)
```

define a circle in the $x - z$ plane.

10.13.3 3D Sphere

The syntax is:

```
sphere x1 y1 z1 radius r1 style [number]
```

where `style` indicates the type of display: 0 for shaded facets, 1 for facets, 2 for a symbolic representation. `x1, y1, z1` are the coordinates of the center and `r1` its radius.

10.13.4 3D block

A 3D block can be defined with:

```
cube expr expr expr radius expr expr expr style [number]
```

where `expr` define elementary expressions. The three first expressions define the center, the next three the height, length and width and `style` indicates the type of representation as mentioned for the 3D sphere.

10.13.5 3D Point

To draw curves in 3D space you can use the command `point3d` which return the 2D coordinates of the point:

`[vector]=point3d ( x1 y1 z1)`

where `x1, y1, z1` are the 3D coordinates.

10.14 Texturing primitives

You can add a texture to a created primitives by:

`texture [primitive number] type y v1 x v2 at v3`

`type` has a value 1 for a texture with vertical lines, 1 for horizontal lines, 3 for vertical and horizontal lines and 4 for tilted lines. In that case `v1` is the distance between the horizontal lines and `v2` between the vertical lines. If `type` has a value of 5 you will get a gray level texture and `v3` is the gray level starting from 1 (black) to 100 (white).

10.15 Manipulating primitives

10.15.1 Deleting primitives

You can delete a primitive with:

`destroy [primitive number]`

10.15.2 Associating, dissociating

You can associate or dissociate all the primitives with the commands `associate-all`, `dissociate-all`. You can also do that for specific primitives with `associate [primitive number]` et `dissociate [primitive number]`

You can also manipulate primitives which have been associated before running the program. This is done by the attribution of a primitive number to each associated primitives:

`#[number]=&[i]`

where `i` indicates the ith primitive, according to the order in the association process, starting from 0 (the first associated primitive has number 0).

10.15.3 Translation, rotation

You can translate a primitive with:

`translate #[primitive number] [vector]`

where `vector` is the translation vector. You can rotate a primitive with:

`rotate #[primitive number] around [vector] , [value]`

where `vector` is the coordinates of the rotation center and `value` the rotation angle in radian.

10.15.4 Compilation and texts size

The command `compile_all_text` enables to compile the texts to get their exact size.

10.15.5 Copying a primitive

You can copy a primitive with:

`#[primitive number]= copy #[primitive number1]`

which copy the primitive `primitive number1` into primitive `primitive number`. You can also copy a primitive with a scale factor with:

`#[primitive number]= copyh #[primitive number1] [value]`

where the scale factor is given by `value`.

10.15.6 Mirrors

You can apply the mirror on a primitive by:

```
lower-mirror #[primitive number]
upper-mirror #[primitive number]
right-mirror #[primitive number]
left-mirror #[primitive number]
```

10.15.7 Centering

You can center all the primitives or the associated primitives with the commands `Center`, `Center-associate`.

10.16 Program with parameters

You can indicate parameters to a program at start time by:

```
xjpdraw -F [program file name][[argument0,argument1]]
```

as in:

```
xjpdraw -F formule.f[0,2,toto]
```

10.16.1 Declaring arguments

Although it is not necessary you can indicate in the program file the number of arguments of a program:

```
prog([argument number])
```

This is not compulsory but in that case `xjpdraw` will verify the number of arguments and will protest in the program if an invalid argument is used.

10.16.2 Using arguments

The arguments value are put in special variables starting with a $ followed by the argument number, starting at 0.

10.16.3 Unitlength

You can set the variable `unitlength` with:

```
set unitlength [number]
```

10.16.4 Comments

Every character between `/*` et`*/` is a comment.

10.16.5 Loading and saving a file

To load a `.macdraw` file use the command:

```
load [character string]
```

To save a file use:

```
save [character string]
```

which will save a LaTeX file and a Postscript file. To save the Postscript in the LaTeX file use:

```
save_included [character string]
```

For example:

```
save "toto.ltex"
save $0
load "tata"
```

These commands can be used to modify a set of file. For example consider the following program:

```
load $0
set unitlength $2
save $1
quit
```

which is run by:

```
xjpdraw -FI zoom.f[toto,toto.ltex,0.5]
```

The file `toto.macdraw` is loaded, `unitlength` is set to 0.5 and then the file is saved to produce a new set of `toto` file. With that the size of the drawing is reduced by two. here we have the silent running mode of `xjpdraw`, enabling to do the job from a simple terminal.

10.16.6 Quitting

To quit a program use:

```
quit
```

10.17 Loading a user's function

You can load a user's function using the same syntax as for loading the function at start time, like in:

```
load-function "-X 10 -Y 10 -F toto"
```

which will load the user's function `toto` in a 10 by 10 rectangle. The possible syntaxes are:

```
load-function "-X [expr] -Y [expr] -F [string]"
load-function "-X [expr] -Y [expr] -F [string] -O [expr] [expr] -F [string]"
```

as in:

```
load-function "-X %12 -Y %12 -O %10 %11 -F toto".%0
```

You can also test for the existence of a user's function (see `exist`). The minimum and maximum values of the function are put in the variable `function_max_x,function_min_x`, `function_max_y,function_min_y`.

You can set the parameters of the grid which will be drawn around the user's function:

```
grid x = [expr]
grid y = [expr]
format x [string]
format y [string]
```

The two first commands are used to set the distance between the vertical, horizontal lines of the grid. The last set of command is used to define the format of the text associated to the grid lines.

Creating the grid is done by using the command `put-echelle`. You can ask for a full (reduced) grid by using the command `echelle-longue` (`echelle-courte`).

If you want to load different user's function without modifying the scale factor you may use the command `no_scale_change`. To come back to the automatic scaling mode use the same command.

10.18 Miscellaneous

10.18.1 Resetting

For the creation of a new file use the command `new`. This command has the same behavior of the command `New file` of the `FILE` menu.

10.18.2 Distance between two points

To compute the distance between two points use:

```
[variable]=distance from [vector] to [vector]
```

10.18.3 Center of a primitive

To get the location of the center of a primitive use:

```
[vector]=center [primitive number]
```

10.18.4 Intersection of two lines

You can compute the location of the intersection of two lines with:

```
[vector]=intersection [primitive number] [primitive number]
```

If there is no intersection both components of the vector will be set to -1.

10.18.5 Start and end point of a line

The coordinates of the start point and end point of a line can be obtained via:

```
[vector]=start [primitive number]
[vector]=end [primitive number]
```

10.19 Existence of a file

You can test if a file exist by:

```
[variable]= exist [string]
```

The variable will be set to 1 if the file exists , to 0 otherwise. For example:

```
%10=exist "toto"
%11=exist "toto".%2
```

10.20 Ordering the Postscript and LaTeX file

By default the Postscript file is included before any other command in the LaTeX document produced by `xjpdraw`. You can modify this behavior by using `ps_after_latex`. The standard behavior will be obtained by using `ps_before_latex`.

10.21 Program examples

10.21.1 Mouse interaction

```
/*---------------------------------------
This program will draw a model of linear actuator.
Two points are given with mouse, the first one to indicate the
base of the actuator, the second for the mobile part extremity.
---------------------------------------*/

/* information */

info "Linear actuator. In ‘‘Selection’’ mode use right to abort"
/*input points */
input_mouse 2
```

```
%1=1.5

/* diameter of ram */
%2=%1/8
%12=%1/10

%v2[0]= %v1[0]-%v0[0]
%v2[1]= %v1[1]-%v0[1]
%5= sqrt(%v2[0]*%v2[0]+%v2[1]*%v2[1])
%v20[0]=%v2[0]/%5
%v20[1]=%v2[1]/%5

%v5[0]=%v2[1]
%v5[1]=%v2[0]

%9=atan2(%v5)
%6=%9*180/pi

/* actuator's body */

%v10[0]=%v0[0]+%1*%v20[0]-%2*cos(%9)
%v10[1]=%v0[1]+%1*%v20[1]+%2*sin(%9)

%v11[0]=%v0[0]-%2*cos(%9)
%v11[1]=%v0[1]+%2*sin(%9)

%v12[0]=%v0[0]+%2*cos(%9)
%v12[1]=%v0[1]-%2*sin(%9)

%v13[0]=%v0[0]+%1*%v20[0]+%2*cos(%9)
%v13[1]=%v0[1]+%1*%v20[1]-%2*sin(%9)

line from %v10 to %v11
line from %v11 to %v12
line from %v12 to %v13



%v14[0]=(%v0[0]+%v0[0]+%1*%v20[0])/2
%v14[1]=(%v0[1]+%v0[1]+%1*%v20[1])/2

/* actuator's beam */


%v15[0]=%v14[0]-%12*cos(%9)
%v15[1]=%v14[1]+%12*sin(%9)

%v16[0]=%v14[0]+%12*cos(%9)
%v16[1]=%v14[1]-%12*sin(%9)

line from %v15 to %v16
line from %v14 to %v1

quit
```

An example of use of this program is presented in figure 10.1.

10.21.2 Example of use of `for` loop

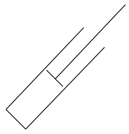


Figure 10.1: An example of use of the actuator program

```
/*------------------------
This program will draw a circle with segments originating at the
center and finishing at equally distributed points on the circle.
Points are drawn at equal interval on the segments.
-------------------------*/
for %0 from 0; (%0 < 360); step 10 bloc
       %10=%0*pi/180
       %1=5*cos(%10)
       %2=5*sin(%10)
       %v0=(6,8)
       %v1=(6+%1,8+%2)
       line from %v0 to %v1
       for %20 from 0; (%20 < 5); step 0.5 bloc
              %11=%20*cos(%10)
              %12=%20*sin(%10)
              %v10=(6+%11,8+%12)
              circle center %v10 radius 0.02
       end_bloc
end_bloc
circle center %v0 radius 5
quit
```

This program produces the output of figure 10.2.

10.21.3 Example of use of Formule

```
/* ---------------------
This program will draw a sine curve between to points introduced
with the mouse (the points should be close)
----------------------- */

info "Wiggly line from A to B, in ‘‘Selection’’ mode use right to abort"

input_mouse 2

%1=distance from %v0 to %v1
%2=atan((%v1[1]-%v0[1])/(%v1[0]-%v0[0]))

%3=0.2 /* k1 */
%4=10 /*k2*/
%6=pi/(2*%4)+%1/2 /*u2*/
%5=%6-%1

#0=Formule %3*sin(%4*x), x between %5,%6 , step 0.05 , start %v0

rotate #0 around %v0 , %2

quit
```

The output of this program is presented in figure 10.3.

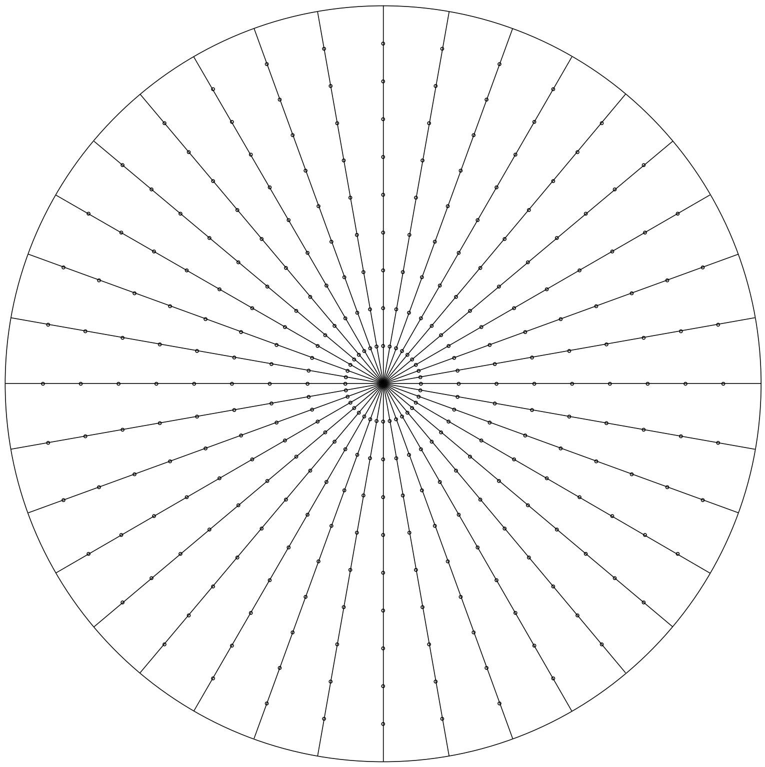


Figure 10.2:

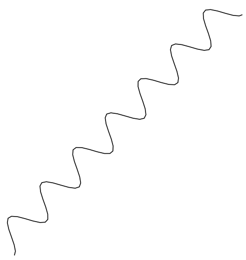


Figure 10.3:

10.21.4 More examples

```
/*----------------------------------------
Constructing a pencil of circles with xjpdraw using two points
given with the mouse.

Two points %v0, %v1. are given with the mouse and the segment S1
connecting them is drawn.
The middle point M is computed and the middle line S2 is drawn
Two points P1,P2 are chose on S1 and one point P3 on S2.
The circle C going through P1,P2,P3 is drawn. Two point Q1,Q2 are chosen
on the circle and we compute the two circles normal to C and going through
the points Q1, Q2 whose centers are on S1. We draw the segments tangent of
these circles going through M and we show that they have the same
length as predicted by the theory
----------------------------------------*/

/* information */

info "Constructing a pencil of circles"
/*the initial points  */
input_mouse 2


%80=atan((%v1[1]-%v0[1])/(%v1[0]-%v0[0]))
/* middle point of v0, v1 */
%v2[0]=(%v0[0]+%v1[0])/2
%v2[1]=(%v0[1]+%v1[1])/2

/* segment S1 */
#1=line from %v0 to %v1

/* S2=middle line of S1 */
#2=line from %v2 left perpendicular to #1 , length 3
#3=line from %v2 right perpendicular to #1 , length 3

/*v3: a point on S1 */
%v3[0]=(%v2[0]+%v1[0])/2
%v3[1]=(%v2[1]+%v1[1])/2

/*v4: another point on S1 */
%v4[0]=(%v2[0]+%v0[0])/2
%v4[1]=(%v2[1]+%v0[1])/2

/*v6: a point on S2 */
%v5=end #2
%v6[0]=(%v2[0]+%v5[0])/2
%v6[1]=(%v2[1]+%v5[1])/2

/* circle C going through v3,v4,v6 */

#4=circle toward %v3 %v4 %v6

/* %v7 C center, %8 C radius */
%v7= center #4
%8=radius #4

/*#5,#6 two segments starting from C center */

#5=line from %v7 , vector (1,1) , length %8
#6=line from %v7 , vector (1,-1) , length %8
```

```
/*#7,#8 two segments perpendicular to the two previous segments */

%v9=end #5
%v10= end #6
#7=line from %v9 , vector (1,-1) , length 7
#8=line from %v10 , vector (1,1) , length 7

/* intersection of #7,#8 with S1 which will be the centers of the two
circles normal to C */

%v11=intersection #1 #7
%v12=intersection #1 #8

/* these circles radii */

%10=distance from %v9 to %v11
%11=distance from %v10 to %v12

/* drawing these circles */

circle center %v11 radius %10
circle center %v12 radius %11

/* drawing the tangents to these circles, going through v2 */

%13=distance from %v2 to %v11
%14=sqrt(%13*%13-%10*%10)
%15=atan(%10/%14)
%v30=(cos(%15+%80),sin(%15+%80))

set style thick
#9=line from %v2 , vector %v30 , length %14

%23=distance from %v2 to %v12
%24=sqrt(%23*%23-%11*%11)
%25=atan(%11/%24)
%v31=(cos(%25+%80),sin(%25+%80))

set style thick
#10=line from %v2 , vector %v31 , length %24

/* destroying the primitives used during the construction */

destroy #5
destroy #6
destroy #7
destroy #8

/* writing some texts */

%v40[0]=%v2[0]-0.3
%v40[1]=%v2[1]+0.3
text at %v40 "A" size 10

%v41=end #9
%v42=end #10

%v43[0]=%v41[0]
%v43[1]=%v41[1]
```

```
%v44[0]=%v42[0]
%v44[1]=%v42[1]
text at %v43 "B1" size 10
text at %v44 "B2" size 10

%v50=end #2
%v60=end #3


text at %v50 "Distance AB1",%14 size 10

%v51[0]=%v50[0]
%v51[1]=%v50[1]-0.5
text at %v51 "Distance AB2",%24 size 10

%v61[0]=%v60[0]-3
%v61[1]=%v60[1]-0.5
text at %v61 "Building pencil of circles with xjpdraw"
size 10
set style thin
quit
```

An example of drawing produced by this program is given in figure 10.4.

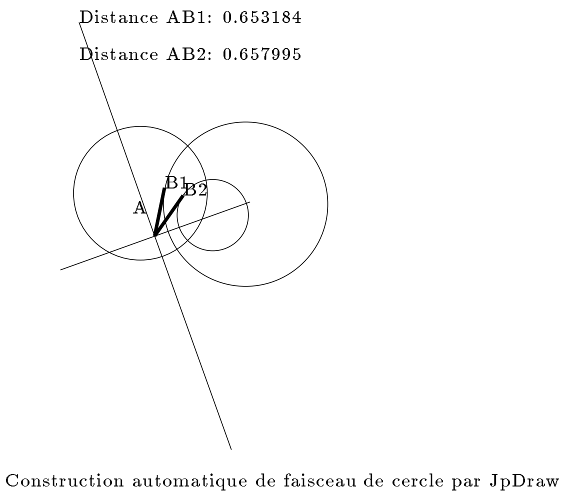


Figure 10.4:

10.21.5 3D examples

```
/*----------------------------
Splitting a 3D cube in smaller cubes
-----------------------------*/
view ( 0 0 0 15) ( 6 10 5)

%20=5

%10= -7.5+%20/2
%11= 7.5-%20
for %0 from %10 ; (%0 <=%11) ; step %20 bloc
       for %1 from %10 ; (%1 <= %11) ; step %20 bloc
```

```
            for %2 from %10 ; (%2 <= %11) ; step %20 bloc
                   cube %0 %1 %2 radius %20 %20 %20 style 2
            end_bloc
      end_bloc
end_bloc
quit
```

This program lead to the figure 10.5.

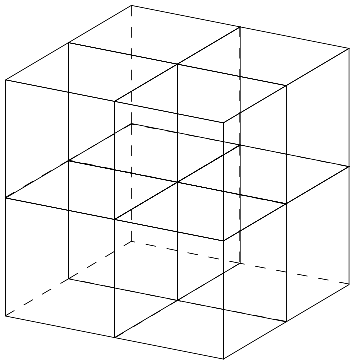


Figure 10.5:

10.21.6 Connoly surface

```
/*--------------------------
The Connoly surface is the surface generated by the intersection
of two fixed spheres with all the spheres having a contact point
with both spheres (imagine that you take all the spheres which can
roll on the two fixed spheres). For one rolling sphere the intersection
is two circles (if the intersection exists).
--------------------------*/
view ( 0 0 0 7) ( 6 10 10)
alias_variable "r1" 10
alias_variable "r2" 11
alias_variable "x1" 0
alias_variable "x2" 1
alias_variable "r" 20
alias_variable "D" 40
alias_variable "d1" 50
alias_variable "d2" 51
alias_variable "R" 61
alias_variable "xc" 21
alias_variable "yc" 22
alias_variable "zc" 23
{x1}=1.5
{x2}= -{x1}
{r1}=1
{r2}=1
/* the fixed spheres */

set style thick
sphere {x1} 0 0 radius {r1} style 2
sphere {x2} 0 0 radius {r2} style 2
set style thin
```

```
{r}=1.5

/* A1, A2 */
%30=({r1}+{r})**2
%31=({r2}+{r})**2

/* D */

{D}=2*{x1}

/* d1, d2, D=d1+d2 */

{d2}=({D}*{D}-%30+%31)/(2*{D})
{d1}={D}-{d2}

{xc}={x2}+{d1}

/* R^2 =%60*/

%60= %30-{d1}*{d1}

if ( %60 < 0 ) bloc
       print("no solution\n")
       quit
       end_bloc
/* R = %61 */

{R}=sqrt(%60)


/* we draw the intersection for %70 rolling spheres */

%70=5

for %71 from 1; (%71 < %70) ; step 1 bloc
       {yc}={R}*cos(2*pi*%71/%70)
       {zc}={R}*sin(2*pi*%71/%70)
       sphere {xc} {yc} {zc} radius {r} style 2
end_bloc

%v0[0]={d1}
%v0[1]={R}

%80=atan2(%v0)

%81={r1}*sin(%80)
%82={r1}*cos(%80)

%82={x1}-%82
cercle3d %82 0 0 radius %81 (90 90)

if ( %60 > {r} ) bloc
       %v1[0]={d1}
       %v1[1]={R}-{r}
       %85=atan2(%v1)
       end_bloc
if (%60 <= {r})bloc
       %85=0
       end_bloc
```

```
for %40 from %85; (%40 < %80) ; step 0.1 bloc
        %41={d1}*cos(%40)+{R}*sin(%40)
        %41= -%41
        %42={d1}*{d1}+{R}*{R}-{r}*{r}
        %45= -%41-sqrt(%41*%41-%42)
        %81=%45*sin(%40)
        %82=%45*cos(%40)
        %82={x1}-%82
        cercle3d %82 0 0 radius %81 (90 90)
        end_bloc


%v0[0]={d2}
%v0[1]={R}

%80=atan2(%v0)

%81={r2}*sin(%80)
%82={r2}*cos(%80)

%82={x2}+%82
cercle3d %82 0 0 radius %81 (90 90)


if ( %60 > {r} ) bloc
        %v1[0]={d2}
        %v1[1]={R}-{r}
        %85=atan2(%v1)
        end_bloc
if (%60 <= {r})bloc
        %85=0
        end_bloc

for %40 from %85; (%40 < %80) ; step 0.1 bloc
        %41={d2}*cos(%40)+{R}*sin(%40)
        %41= -%41
        %42={d2}*{d2}+{R}*{R}-{r}*{r}
        %45= -%41-sqrt(%41*%41-%42)
        %81=%45*sin(%40)
        %82=%45*cos(%40)
        %82={x2}+%82
        cercle3d %82 0 0 radius %81 (90 90)
        end_bloc
```

A Connoly surface is drawn in figure 10.6.

10.21.7 Loading a user's function

```
/* load various user's function with  name toto followed by a number.
The existence of the files is tested and they are put in square
with a grid */


/* dimension of the square */
%10=0.8
%11=0.5
%12=4
%13=0.8
%14=1
```

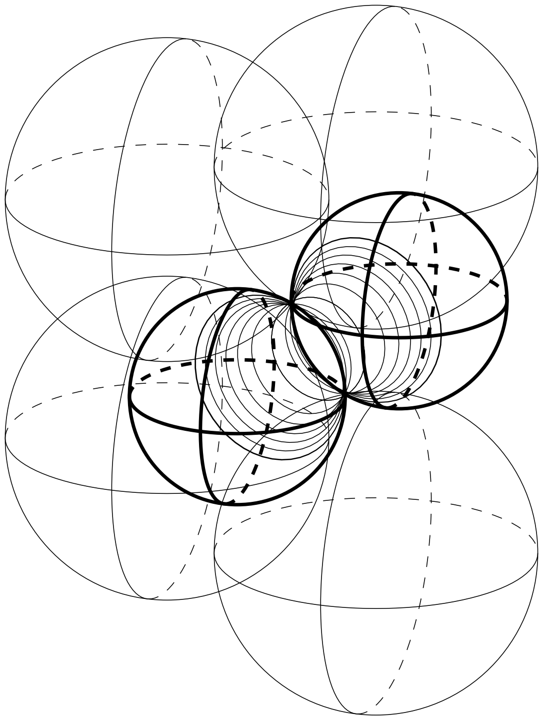


Figure 10.6:

```
%20=0

format "%5.0f"

for %0 from 10; (%0 < 350);step 59 bloc
	%1=exist "toto".%0
	if (%1==1)bloc
		load-function "-X %12 -Y %12 -O %10 %11 -F toto".%0
		echelle-courte
		grid x = 1
		grid y = 1
		format x "3.1"
		format y "3.1"
		put-echelle
		%v0[0]=%10+1
		%v0[1]=14.5-%11-1
		text at %v0 %0 size 10
		%20=%20+1
		%10=0.8+%20*(%12+%13)
		if (%20 > 2) bloc
			%20=0
			%10=0.8
			%11=%11+(%12+%14)
		end_bloc
	end_bloc
end_bloc

quit
```

Chapter 11

Arguments

In this chapter we will describe the various arguments that you can give when starting `xjpdraw`, together with a summary of the keys which are used as shortcuts.

11.1 Arguments

11.1.1 A .macdraw file

If you give only one argument it will be considered as the name of a `.macdraw` file. The `.macdraw` extension is necessary only if there is more than one extension.

`xjpdraw toto` for loading `toto.macdraw` but `xjpdraw toto.ps.macdraw` for loading `toto.ps.macdraw`

11.1.2 A user's function

You can also indicate a user's function as argument with the syntax:

```
xjpdraw -X x1 -Y y1 -F name [-E xmin xmax ymin ymax][-O x2 y2]
```

where `x1, y1` are the width, height of the enclosing rectangle (in centimeter) and `name` is the file name of the user's function. If you use `-Fp` instead of `-F` the enclosing rectangle will not be drawn (but will be present in the trash can). Optional argument are:

- `-E` where you define the minimum and maximum values of the box in the user's unit
- `-O` where you define the location of the upper left corner of the box (in centimeter). The default values are (2,2).

11.1.3 A program

To run a program use:

```
xjpdraw -F [program file name]
```

You can also give arguments to a program:

```
xjpdraw -F [program file name][[argument0,argument1]]
```

as in:

```
xjpdraw -F formule.f[0,2,toto]
```

11.1.4 Memory

By default the maximum number of primitives is 1500. You can modify this value by running:

```
xjpdraw -M [number}
```

where `number` indicates the maximum number. When loading a `.macdraw` with more than 1500primitives `xjpdraw` will automatically perform a memory allocation. This is not the case when using the album.

11.1.5 Slide mode

You start the first page of a slide mode document by running `xjpdraw` with the argument `-slide`.

11.1.6 Verification of a configuration

If you use the argument `-check` `xjpdraw` will verify the validity of your configuration file. A diagnosis will be printed on the standard output (see page 103).

11.2 Keys summary

Remember that to use a shortcut the mouse should usually be in the canvas.

key	mode	action
L2	rectangle	enforce the drawing of a square
L2	ellipsis	to draw circles
L3	polygon	draw arrow-like polygons
L3	selection	selection of the covering primitive
L4	rectangle	enforce the drawing of a rectangle
L4	polygon	delete last vertex
L4	ellipse	to draw ellipsis
L5	polygon	suppress the creation of a polygon
L6	circle	delete the biggest arc on the circle
L8	rectangle	associate the primitives inside the rectangle
L8	polygon	associate the primitives inside the polygon
L8	line	enforce the slope of a line
L8	circle	delete the smallest arc on the circle
L9	selection	big motion for a primitive
L9	polygon	enforce the creation of a regular polygon
L9	rotation	to indicate the rotation center
L10	polygon	for general polygons
L10	selection	resize
F1	-	select the selection icon
F2	-	select the text icon
F3	-	select the line icon
F4	-	select the rectangle icon
F5	-	redraw
F6	-	select the ellipsis icon
F7	-	select the polygons icon
F8	-	select the pencil icon
F9	-	select the rotation icon
F10	-	select the mark icon
R1	selection	left one-pixel translation
R2	selection	right one pixel translation
R4	selection	up one-pixel translation
R4	polygon	delete last vertex
R5	selection	down one-pixel translation
R6	-	copy
R7	-	paste
R13	-	delete

In the "Selection" mode some shortcuts are available.

key	action
a	show the trash can
A	associate all
c	center all
d	dissociate the selected
D	dissociate all
f	put an arrow on the selected primitive
h	help
m	hide the trash can
n	normal to the selected point at the current point
p	put selected in foreground
q	grid
r	show or hide the rules
s	put selected in background
t	center the text in the associate
w,W	save macdraw file
0-9P	execute program [0-9]
CTRL-f	indicate the file name and the directory

Outside the "Selection" mode some shortcuts are available.

key	action
CTRL-a	associate all
CTRL-b	put/dismiss the border on the associates
CTRL-c	center all
CTRL-d	dissociate all
CTRL-e	compile and preview
CTRL-f	display file name
CTRL-g	show the gray texture window and select the most recent primitive
CTRL-i	activate/deactivate the magnetic grid
CTRL-l	show the various dashed line
CTRL-m	hide the trash-can
CTRL-n	next item the magnet list
CTRL-o	indicate the drawing size
CTRL-p	previous item the magnet list
CTRL-q	show or hide the grid
CTRL-r	show or hide the rules
CTRL-s	selected in background
CTRL-t	center the associate text in the associate primitive
CTRL-u	give the number of primitives in the trash can
CTRL-w	save the .macdraw file
CTRL-x	recompile for the previewer
CTRL-y	show the trash can
CTRL-z	on-line help
CTRL–	empty the trash can

Chapter 12

Installing xjpdraw

To run `xjpdraw` it is necessary to define a *configuration file*. But first of all when you get `xjpdraw` you get also some auxiliary file namely:

```
header.latex
header_end.ltex
lplainext.tex
lplainext.psm
doc-en-ligne
doc-en-ligne-english
```

These files should be put in a specific directory. The role of these files are:

- header.latex: a preamble of a LaTeX document. When using some commands (for example the `Verify`, `Compile` commands) `xjpdraw` will produce a complete LaTeX document by copying this file, adding then first the LaTeX file of the drawing and then the file `header_end.ltex`. This file is not necessary: by default `xjpdraw` will use as a preamble

  ```
  \documentstyle{article}
  begin{document}
  ```

  if you are running LaTeX 2.09 and

  ```
  \documentclass{article}
  begin{document}
  ```

  if you are running LaTeX 2e.

- header_end.ltex: this file contains the end of a LaTeX document. It is not compulsory to have defined such file, the default value being

  ```
  end{document}
  ```

- lplainext.tex,lplainext.psm: a set of TeX macros which are now only used to rotate text. If you are using LaTeX 2e this file is not compulsory.

- doc-en-ligne-english, doc-en-ligne : an on-line help file either in french or in english. Although not compulsory this file may be useful.

When installing `xjpdraw` the following process should be followed:

1. put all the above file in a specific directory e.g. `Jpdraw`.
2. in the file `header.latex` modify the access path to `lplainext..` (to be substituted by `Jpdraw`). According to the version you are using for running LaTeX you may have to modify the preamble in this file.
3. in the file `lplainext.tex` modify the access path to the file `lplainext.psm` (to be substituted by `Jpdraw`)
4. update the configuration file

12.1 The configuration file

To locate the configuration file xjpdraw use the environment variable JPDRAW, it is therefore necessary to put in this variable the full name of the configuration file, with its access path.

In the configuration file all the commands start with a # followed by a keyword. The keywords are:

```
BIB
BLINK
COMMAND_FILE
COMPILE_LATEX
CONTEXT
DVI->PS
EN_TETE_SLIDE
END_SLIDE
EXTERNE
FOLLOW_MOUSE
FONT
FONTMATH
FULL_MANUAL
GHOST
GHOSTVIEW
GHOST_UPDATE
GS
LANGUAGE
LATEX_COMMAND
LATEX_VERSION
LIB
LIB_EXTERNE
LOCAL_JPDRAW
MANUAL
PRINTING
PS
PSEXTERNE
ROTATE_ORDER
SELECTION_TYPE
START_SLIDE
UNDO_DIR
UNDO_NUMBER
```

The role of this keywords are summarized here:

Name	Role
BIB	the access path to the programming module
BLINK	to control the blinking of the primitives
CONTEXT	the name of a context file to be loaded when starting
COMMAND_FILE	LaTeX commands file to be added when compiling
COMPILE_LATEX	to indicate if the text are compiled
DVI->PS	to indicate he program which convert the .dvi file into Postscript
EN_TETE_SLIDE	the start of a complete document in slide mode
END_SLIDE	the end of the environment encapsulating each page of a slide document
EXTERNE	to deal with extern program
FOLLOW_MOUSE	to control the motion of the primitive in the "Selection" mode
FONT	screen fonts
FONTMATH	the fonts which are used to display special symbols
FULL_MANUAL	to indicate the full manual file name
GHOST	the program used to visualize Postscript file
GHOST_UPDATE	the program used to update the Postscript file
GHOSTVIEW	the program used to visualize the manual Postscript file
GS	the program used for generating the image of external Postscript file
LATEX_COMMAND	the command used to run LaTeX
LIB	the access path to some files
LIB_EXTERNE	the access path to external programs
LANGUAGE	the language used in xjpdraw
MANUAL	the name of the on-line help
PRINTING	for compiling and printing
PS	syntax to include a Postscript file in LaTeX
PSEXTERNE	the command used to include external Postscript file
ROTATE_ORDER	the command used for rotating text
SELECTION_TYPE	to validate how primitives are selected
START_SLIDE	the beginning of the environment encapsulating each page of a slide document
UNDO_DIR	the directory used for the UNDO command
UNDO_NUMBER	maximum number of primitives before canceling the undo command

Another type of command in the configuration file is the one related to the key used as shortcuts. These keys are called R1,..,R15, L1,..,L10, F1,..,F9, and in the configuration file it is possible to affect these keys to any key on the keyboard. Comments can be added in the configuration file by putting them between /* and */. Remember that you also read a configuration file after starting xjpdraw, by using the command Read a configuration file of the FILE menu.

12.1.1 BIB

If you have written programming modules (see the "Programming" chapter) you may want to be able to use them whenever needed. To do that you can put them in a specific directory with an extension name .fi. If you indicate the path of this directory with this command, like:

```
# BIB /u/miracle/merlet/Util_xjpdraw
```

then the menu attached to the BIB button will display the list of the available programs and you may use any of them.

12.1.2 BLINK

When selected a primitive will blink. The blinking process is controlled with this command. The keyword BLINK is followed by a number which indicate on which primitive the blink definition will be applied:

0	text
1	line
2	rectangle
3	circle, ellipse
4	oval
5	polygon

For primitive of type 0 to 4 the number that follow the type is the number of times that the primitive will blink. For example:

```
# BLINK 0 4
```

indicate that the text primitive will blink 4 times when they are selected. For type 5 primitive two numbers are used: the first one for polygon with less than 30vertices, the second one for bigger polygons:

```
# BLINK 5 20 2
```

The default values are:

0	6
1	20
2	11
3	4
4	4
5	11, 2

12.1.3 COMMAND_FILE

This optional command is followed by the name of a file which contain LaTeX commands. This file will be added each time either a text is compiled or a full LaTeX document is generated. This is especially useful if you use very often some special LaTeX command when typing texts.

12.1.4 COMPILE_LATEX

The purpose of this command is to indicate if text primitives are compiled after their input in order to get their exact size. This command is followed by a number which should either 0 (the texts are not compiled) or 1 (the texts are compiled using the command defined by the command `LATEX_COMMAND`) as in:

```
# COMPILE_LATEX 1
```

12.1.5 CONTEXT

This command is followed by a file name which is a context file. The context will be loaded and used when starting `xjpdraw`. See the `File` menu for informations about context.

12.1.6 DVI->PS

This command is used to define which type of dvi driver will be used to generate Postscript file from the dvi file, enabling `xjpdraw` to generate Postscript file which are compatible with the driver. The syntax is:

```
DVI->PS [program name]
```

where `[program name]` is the name of the driver and should be either `dvi2ps, dvips or dvitops`. Note that the use of `dvips` is recommended.

12.1.7 EN_TETE_SLIDE

Used for the slide mode. A complete document may start with a special style given by this command. A good example is:

```
# EN_TETE_SLIDE  \documentstyle[trans]{article}\n \begin{document}
```

12.1.8 END_SLIDE

indexEND_SLIDE Each page of a slide may be encapsulated in a special environment. The string following this order will be put at the end of each page.

12.1.9 EXTERNE

`xjpdraw` can output a data file describing the current drawing which can be used by an external program, producing in turn some drawing data that are read by `xjpdraw` to be incorporated in the current drawing, see page 51 for more details.

12.1.10 FOLLOW_MOUSE

This command manage the behavior of `xjpdraw` when being in the "Selection" mode you have selected a primitive and are currently dragging the mouse.

The command:

```
# FOLLOW_MOUSE always
```

has the effect that the selected primitive will follow the mouse motion. The command:

```
# FOLLOW_MOUSE never
```

has the effect that the selected primitive will be kept stationary.

The command:

```
# FOLLOW_MOUSE variable
```

will have the following behavior: by default the primitive will not move, but if you press the L6 key the primitive will follow the mouse motion. This is the default mode.

The "Selection" icon displays the current behavior: `no mouse` means that the primitive does not move otherwise the message `with mouse` is displayed.

12.1.11 FONT

To display text primitive `xjpdraw` uses some specific font according to the style and size of the text. You may indicate which font are used by using this command which should be followed by one of the keyword `STANDARD, GRAS , ITAL` with a number indicating the font size. For example you may write:

```
# FONT GRAS10 -adobe-times-bold-r-normal--8-80-75-75-p-47-iso8859-1
```

The default fonts are:

STANDARD8	-adobe-times-medium-r-normal–8-80-75-75-p-44-iso8859-1
STANDARD10	-adobe-times-medium-r-normal–12-120-75-75-p-64-iso8859-1
STANDARD12	-adobe-times-medium-r-normal–12-120-75-75-p-64-iso8859-1
STANDARD14	-adobe-times-medium-r-normal–14-100-100-100-p-74-iso8859-1
STANDARD16	-adobe-times-medium-r-normal–17-120-100-100-p-84-iso8859-1
STANDARD18	-adobe-times-medium-r-normal–18-180-75-75-p-94-iso8859-1
STANDARD24	-adobe-times-medium-r-normal–24-240-75-75-p-124-iso8859-1
GRAS8	-adobe-times-bold-r-normal–8-80-75-75-p-47-iso8859-1
GRAS10	-adobe-times-bold-r-normal–12-120-75-75-p-67-iso8859-1
GRAS12	-adobe-times-bold-r-normal–12-120-75-75-p-67-iso8859-1
GRAS14	-adobe-times-bold-r-normal–14-100-100-100-p-76-iso8859-1
GRAS16	-adobe-times-bold-r-normal–17-120-100-100-p-88-iso8859-1
GRAS18	-adobe-times-bold-r-normal–18-180-75-75-p-99-iso8859-1
GRAS24	-adobe-times-bold-r-normal–24-240-75-75-p-132-iso8859-1
ITAL8	-adobe-times-medium-i-normal–8-80-75-75-p-42-iso8859-1
ITAL10	-adobe-times-medium-i-normal–12-120-75-75-p-63-iso8859-1
ITAL12	-adobe-times-medium-i-normal–12-120-75-75-p-63-iso8859-1
ITAL14	-adobe-times-medium-i-normal–14-100-100-100-p-73-iso8859-1
ITAL16	-adobe-times-medium-i-normal–17-120-100-100-p-84-iso8859-1
ITAL18	-adobe-times-medium-i-normal–18-180-75-75-p-94-iso8859-1
ITAL24	-adobe-times-medium-i-normal–24-240-75-75-p-125-iso8859-1

The keyword `JPDRAWFONT` can be used to define the font used for `xjpdraw` buttons and menus. The default value is:

```
#FONT JPDRAWFONT
-misc-fixed-bold-r-normal--13-100-100-100-c-70-iso8859-1
```

Note that it is necessary to choose carefully this font in order to get a correct presentation of `xjpdraw`.

12.1.12 FONTMATH

Some mathematical symbols written in LaTeX will be displayed as appearing on the paper. To do this we use two mathematical fonts which are by default:

```
# FONTMATH -adobe-symbol-medium-r-normal--14-140-75-75-p-85-adobe-fontspecific
# FONTMATH -adobe-times-medium-r-normal--12-120-75-75-p-64-iso8859-1
```

Using `FONTMATH` you can specify other fonts.

12.1.13 FULL_MANUAL

This command is used to define the full name of the `xjpdraw` manual Postscript file as in:

```
# FULL_MANUAL /u/merlet/Jpdraw/manuel.ps
```

To visualize this manual the command GHOSTVIEW must be defined.

12.1.14 GHOSTVIEW

With this command you specify the name of a program enabling to display a Postscript file:

```
# GHOSTVIEW /usr/bin/gnu/ghostview
```

12.1.15 GHOST

In this command you describe the commands for compiling a LaTeX file, to convert the dvi file in Postscript and then to display the Postscript file. It is used with the `Compile and visualize` command of the `FILE` menu. An example is:

```
# GHOST dvips -f * >bidon.ps;/usr/bin/gnu/ghostview bidon.ps
```

Here the dvi file (obtained after running the LaTeX compilation, which is done in background, therefore never give a latex order in this command) is converted into Postscript using `dvips` and then displayed on the screen using `ghostview`.

12.1.16 GHOST_UPDATE

After using the `Compile and visualize` command of the `FILE` menu you may not exit from `ghostview`. Keeping it in background you may modify your drawing and just update your Postscript file by the command by the command `Recompile for the previewer`. This command will use the sentence defined in this command for updating the Postscript file as in:

```
# GHOST_UPDATE dvips -f * >bidon.ps&
```

Never put a latex order in this command as latex is run directly by `xjpdraw`.

12.1.17 GS

For including external Postscript file `xjpdraw` uses the `gs` program to convert the Postscript into a bitmap which is then displayed on the screen. This command enables to define the `gs` command as in:

```
# GS /usr/bin/gnu/gs
```

12.1.18 LANGUAGE

This command enable to use `xjpdraw` in two different languages,french or english. You can use either

```
# LANGUAGE french
```

or

```
#LANGUAGE english
```

The default value is french.

12.1.19 LATEX_COMMAND

This command enables to define with which program the text are compiled like:

```
# LATEX_COMMAND latex
```

This command is useful only if the command COMPILE_LATEX has been set to 1.

12.1.20 LATEX_VERSION

This command enable to define the version of LaTeX you are using. The possible value are:

```
LATEX_VERSION latex2.09
LATEX_VERSION latex2e
```

12.1.21 LIB

The command `LIB` enable to indicate the directory where are located the auxiliary files:

```
# LIB /u/merlet/Jpdraw
```

12.1.22 LIB_EXTERNE

This command defines the location of the directory where are located the external programs that `xjpdraw` may execute, see page 51 for more details.

12.1.23 MANUAL

This command defines the location and name of the on-line help, like in:

```
# MANUAL /u/merlet/Jpdraw/doc-en-ligne-english
```

12.1.24 PRINTING

This command is used for compiling and printing of a drawing. It is used by the `Compile and print` command of the `FILE` menu. For example:

```
# PRINTING latex %;dvips *
```

The % character will be substituted by the name `bidon.ltex` and the * character by `bidon`.

12.1.25 PS

Unfortunately there are no standard way to include a Postscript file in LaTeX, the operation being dependent on the dvi driver. For example using dvi2ps the syntax is:

```
\special{ps:plotfile [Postscript file]}
```

The command `PS` enables to define the syntax of the inclusion order for your own driver. For example the syntax for dvi2ps will be described by:

```
# PS \special{ps:plotfile %s}
```

The %s string will be substituted by the name of your Postscript file. You can also use two float defined by %f which will be substituted by the width and height of your drawing. For example for the `dvitops` driver we will have:

```
# PS \special{dvitops: import %s %fcm %fcm}
```

For the `dvips` program you may use:

```
# PS \epsffile{%s}
```

if your are using the `epsfig` macros. An alternate way to use dvips without any need of macros is to use the following syntax:

```
# PS \special{psfile="`zcat %s" hscale=\the\jpdrawx \sp vscale=\the\jpdrawx}
```

In that case some nice feature of `dvips` are used. First the Postscript file generated by `xjpdraw` will be compressed as the inclusion order enables to include compressed files. In fact these files will be uncompressed on run time by dvips using the `zcat` command. . Next we will use the `hscale, vscale` command accepted by `dvips` to scale the drawing. Indeed the LaTeX file generated by `xjpdraw` will take into account the value of the LaTeX `unitlength` command to scale the drawing. If the value of this command is 1cm the drawing will be included as it was written. But a `unitlength` value of 0.5cm will lead to a half size drawing. Furthermore if the value of `unitlength` is negative then the drawing will be scaled so that its maximum dimension will correspond to the absolute value of `unitlength`. For example if your original drawing has a size of 4 by 6 cm and you have defined `unitlength` as -3cm then the size of the drawing on your paper will be 2 by 6 cm.

An important option of the `PS` command is the use of %S instead of %s. In that case not only the name of the Postscript will be written but also its absolute path, therefore enabling to move the LaTeX file without worrying with the location of the Postscript file.

Note also that the syntax defined by the `PS` command should be compatible with the driver defined in the `DVI->PS` command.

12.1.26 PSEXTERNE

The purpose of this command is similar to the purpose of the `PS` command except that it is used for including the external Postscript file.

12.1.27 ROTATE_ORDER

This command enables to define the commands which are used to rotate text. If you have not defined text and have rotated text in your drawing `xjpdraw` will use the order of the TeX package `lplainext.tex` which is included with `xjpdraw`. But using this command you can define another syntax. For example:

```
\begin{rotate}{%d} %s \end{rotate}
```

enable to use the `rotating` package of LaTeX2e. The following convention are used:

- %r will be substituted by the rotation angle in radian
- %R will be substituted by minus the rotation angle in radian
- %d will be substituted by the rotation angle in degree
- %D will be substituted by minus the rotation angle in degree
- %s will be substituted by the text to rotate

12.1.28 SELECTION_TYPE

The default mode for selecting a primitive is to click near its border. But another mode can be used with this order: every newly created primitives will be automatically selected. This mode is obtained with the order:

```
# SELECTION_TYPE creation_and_icon
```

The default mode is obtained with:

```
# SELECTION_TYPE icon
```

12.1.29 START_SLIDE

Each page of a slide may be encapsulated in a special environment. The string following this order will be put at the beginning of each page.

12.1.30 UNDO_DIR

The undo mechanism of `xjpdraw` is managed by writing temporary `.macdraw` file in a directory. This order defines the directory which is used, by default the value is /tmp and the default value may also be described by:

```
# UNDO_DIR /tmp
```

The name of the undo files is:

```
undo[number]_[user's name].macdraw
```

12.1.31 UNDO_NUMBER

The undo mechanism of `xjpdraw` is automatically canceled whenever the number of primitives is too high. This command enable to define the threshold on the number of primitives.

```
# UNDO_NUMBER 200
```

12.1.32 Keys

The special function keys (R,L,F keys) are defined using the following syntax:

```
#[key type][key number] [code] "[name]"
```

For example:

```
# R6 51 "Insert"
```

will affect the R6 key to the key whose code (KeyCode) is 51, i.e. the key name `Insert` on most Sun workstation keyboard (to determine the KeyCode of the keys of your keyboard you can use the `xev` program or the command `xmodmap -pk`). Note that the on-line help will take into account the name attached to a key during the display of an help manual.

12.1.33 Verifying a configuration file

If you want to verify a configuration file run:

```
xjpdraw -check
```

`xjpdraw` will check what you have defined in your configuration file and if some errors have been found will try to propose an alternate solution. The diagnosis is printed on the standard output. A special treatment is reserved to the key assignation: the proposed keycode will be printed together with the corresponding `KeySymb`. By looking at the file:

```
/usr/include/X11/keysymdef.h
```

you will then be able to find if the definition of the key name is correct.

12.1.34 Example

```
# LANGUAGE english
# LIB /u/merlet/Jpdraw/Configurateur
# BIB /u/merlet/Util_xjpdraw
# DVI->PS dvips
# PS \special{psfile="`zcat %S" hscale=\the\jpdrawx \sp vscale=\the\jpdrawx}
# R1 31 "<-"
# R2 35 "->"
# R4 27 "^"
# R5 34 "\/"
# R6 51 "Insert"
# R7 101 "Ins"
# R13 119 "End"
# L1 8 "Stop"
# L2 10 "Again"
# L3 32 "Props"
# L4 33 "Undo"
# L5 56 "Front"
# L6 58 "Copy"
# L7 79 "Open"
# L8 80 "Paste"
# L9 102 "Find"
# L10 104 "Cut"
# BLINK 0 4
# BLINK 1 25
# BLINK 2 6
# BLINK 3 3
# BLINK 4 20
# BLINK 5 20 2
# PRINTING latex209 %;dvips *
# GS /usr/bin/gnu/gs
# GHOSTVIEW /usr/bin/gnu/ghostview
# FULL_MANUAL /u/Jpdraw/manual.ps
# GHOST latex209 %; dvips -f * >bidon.ps;/usr/bin/gnu/ghostview bidon.ps&
# GHOST_UPDATE latex209 %; dvips -f * >bidon.ps&
# PSEXTERNE \special{psfile="%S" hscale=\the\jpdrawx \sp vscale=\the\jpdrawx}
# FONTMATH -adobe-symbol-medium-r-normal--14-140-75-75-p-85-adobe-fontspecific
# FONTMATH -adobe-times-medium-r-normal--12-120-75-75-p-64-iso8859-1
# MANUAL /u/merlet/Jpdraw/Configurateur/doc-en-ligne-english
# COMPILE_LATEX 1
# LATEX_COMMAND latex209
# UNDO_NUMBER 200
# EXTERNE "Associated Intersection" intersection associates polygon polygon
# LIB_EXTERNE /u/Jpdraw
```

12.2 Local configuration file

You may add your own BIB library by defining a local configuration file which contain only BIB commands. The path and name of your local configuration file is given with this command.

Chapter 13

Conclusion

You will probably encounter troubles when running `xjpdraw`. Well be indulgent and remember that this program was originally designed to draw rectangles and horizontal lines...And remember:

-Year: A period of three hundred and sixty-five disappointments.

-43rd Law of Computing:

Anything that can go wr

xjpdraw Segmentation violation – Core dumped

-Scott's second Law:

When an error has been detected and corrected it will be found to have been wrong in the first place.

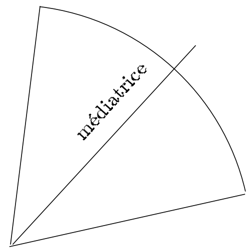
Corollary

After the correction has been found in error it will be impossible to fit the original quantity back into the equation.

Bonne chance!

13.1 Some drawing examples

13.1.1 Rotated text

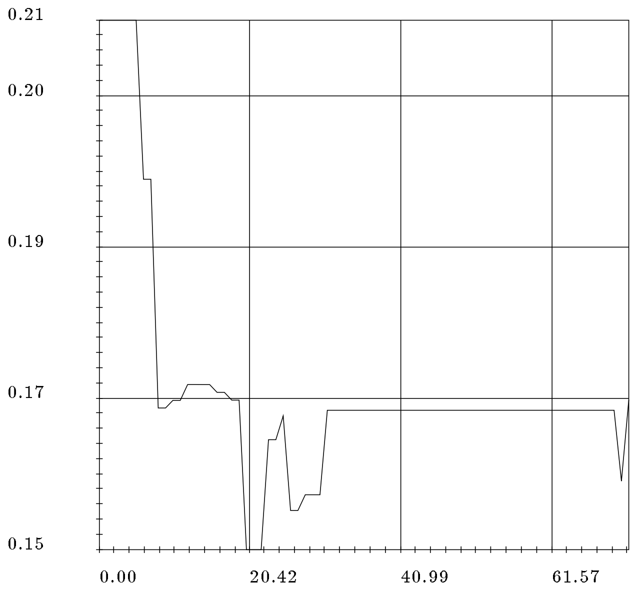


13.1.2 Drawing a 2D curve

635 couples of points have been computed during a robotics experiment and put in a file `y.7` which start with the instruction `mode polygone -1 10`. We have run:

xjpdraw -X 9 -Y 5 -F y.7

which has drawn the curve. The grid has been then defined using the simple window of the function icon.

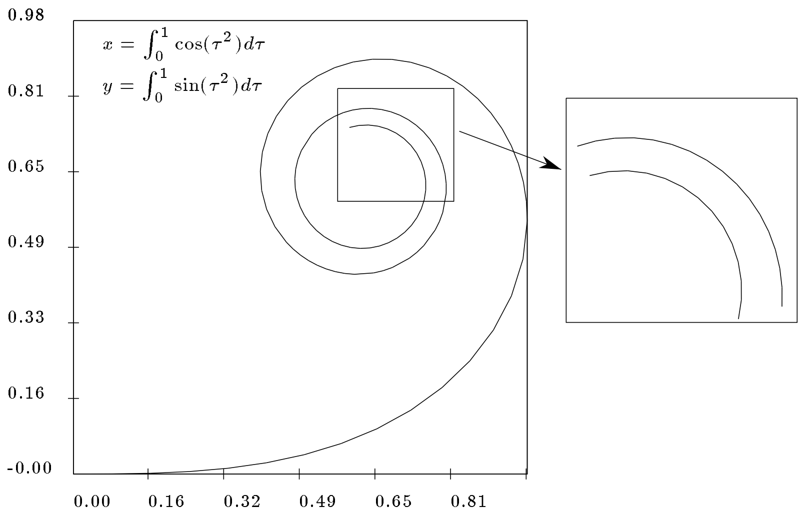


13.1.3 Maple

Using Maple we have drawn the clotoïd defined by:

$$x = \int_0^t \cos(\tau^2) \qquad y == \int_0^t \sin(\tau^2)$$

The `Formula` mode of `xjpdraw` does not permit to display such function but it was easy to get the coordinates of the points from the drawing in Maple to create a user's function.

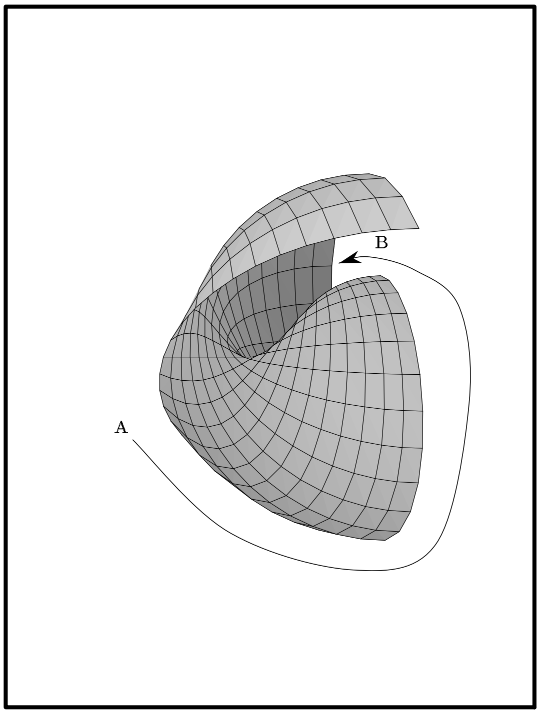


To get a zoom in a local zone we have drawn a square around the zone of interest, associated all the primitives in the zone (by selecting the rectangle and pressing on L8), copied the associated primitives using L6. Then we have asked for a zoom on the associates using the `Zoom` command in the `PAGE` menu.

We may have also used directly the Postscript output of Maple. For example to draw the surface:

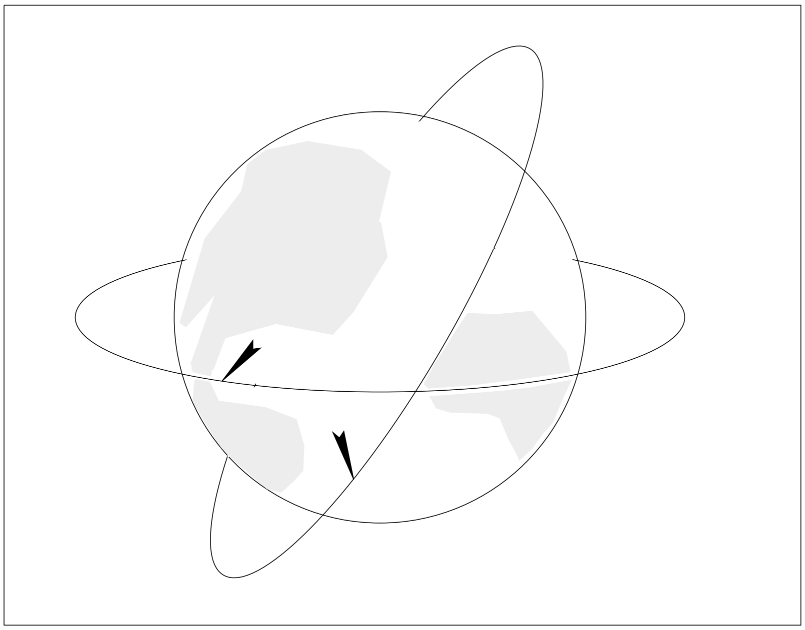
$$\begin{aligned} X &= x\sin(x)\cos(y) \\ Y &= x\cos(x)\cos(y) \\ Z &= x\sin(y) \end{aligned}$$

with $x \in [0, 2\pi]$,$y \in [0, 2\pi]$.



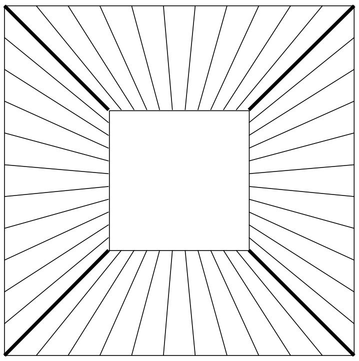
Here we have directly used the Postscript file produced by Maple and then simply add the spline curve going from A to B.

13.1.4 Rotations, satellite and mask



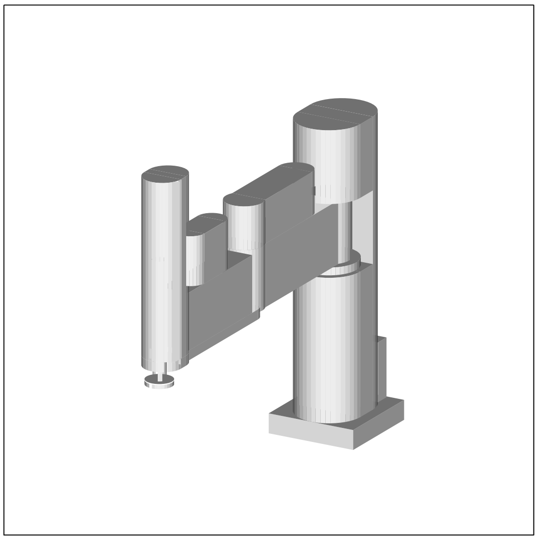
13.1.5 Playing with duplicate

For this drawing we have first drawn the external rectangle, then we have copied it with a scale factor (`EDIT` menu) to generate the internal rectangle. We have then drawn the four lines joining the four vertices of the rectangles, using the "vertex" mode of the magnet. Each pair of such lines have been associated and then we have used the `Duplicate progressively` command to generate all the other lines.



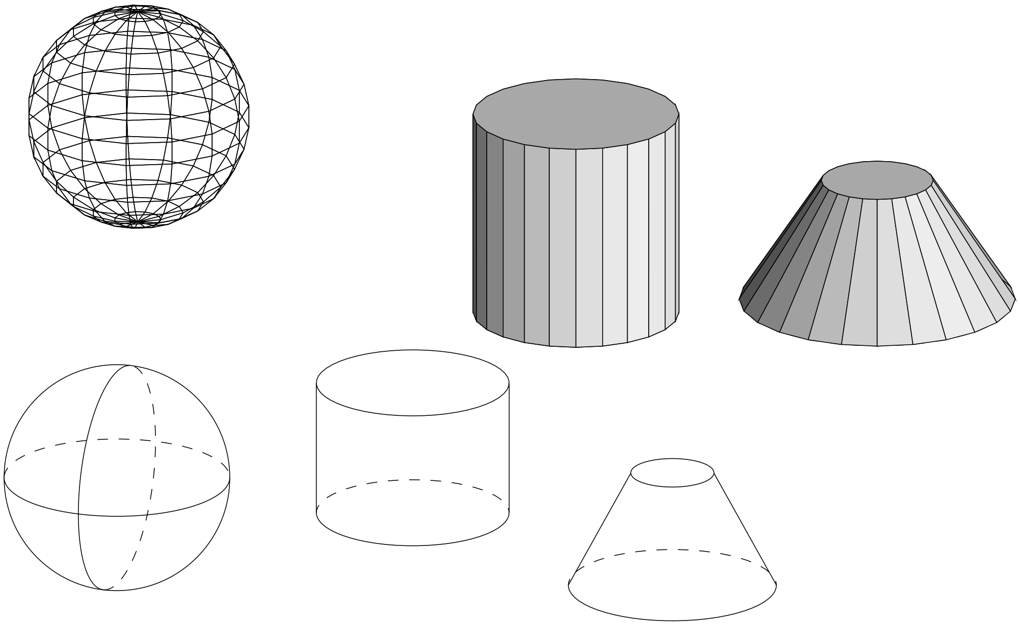
13.1.6 Using a CAD system

The output of a CAD system has been used to make this drawing.



13.1.7 3D Modules

Here are some 3D primitives.



13.2 Errors

Sorry this part is in french....

13.2.1 Messages au lancement

`# should be followed by instruction`
Dans le configurateur on a trouvé un # non suivi d'une commande
`Attention:la variable d'environnement JPDRAW indique un fichier..`
Votre variable d'environnement JPDRAW pointe sur un mauvais fichier configurateur
`Bad argument to BIB in local configuration file`
Le configurateur local contient un # sans argument
`Bad local configuration file`
Le configurateur local contient une commande inconnue
`Bad number after UNDO_NUMBER in configuration file`
UNDO_NUMBER doit être suivi d'un nombre
`Erreur:ces fonts ne sont pas a definir`
Vous avez défini des fonts de nom incorrect dans le configurateur
`Erreur:chaine trop longue dans header_end.ltex:on l'ignore`
Le fichier header_end.ltex est incorrect
`Erreur:dans header_end.ltex (probablement il manque un CR)`
Le fichier header_end.ltex est incorrect
`Erreur:dans header_end.ltex:ignore (il manque un % dans le special)`
Le fichier header_end.ltex est incorrect
`Erreur:dans le fichier de configuration, ligne`
Vérifier la ligne
`Erreur:hdcp-> chaine trop longue (>255)-> pas de copie d'ecran:`
Erreur dans le script hdcp
`Erreur:mauvais fichier de font`
Dans le configurateur on a définit une fonte incorrecte
`Erreur:Pas de script hdcp pour copie d'ecran`
L'opération `copie écran` n'est pas possible car pas de script hdcp pour copie d'ecran dans la directory `LIB` du configurateur. Mettez en un
`Erreur:Pas de script hdcpc pour copie ecran couleur`
L'opération `copie écran` n'est pas possible car pas de script hdcp pour copie d'ecran dans la directory `LIB` du configurateur. Mettez en un
`Error: argument to BEZIER <=0`
L'argument de BEZIER doit être positif
`Error in BLINK argument`
Le premier argument de BLINK doit être compris entre 0 et 5
`Error in BLINK`
L'instruction BLINK doit être suivi de deux ou de trois nombres
`Error in BLINK...must be followed by an int`
BLINK doit être suivi d'un entier
`Error in BLINK 5:must be followed by two int`
BLINK 5 doit être suivi de deux entiers
`Error in configuration file, bad argument to BIB`
BIB doit être suivi d'une chaîne
`Error in configuration file, F key`
F doit être suivi d'un nombre et d'une chaîne, il manque le nombre
`Error in configuration file, L key`
L doit être suivi d'un nombre et d'une chaîne, il manque le nombre
`Error in configuration file, missing F key name`
F doit être suivi d'un nombre et d'une chaîne, il manque la chaîne
`Error in configuration file, missing L key name`
L doit être suivi d'un nombre et d'une chaîne, il manque la chaîne
`Error in configuration file, missing R key name`

R doit être suivi d'un nombre et d'une chaîne, il manque la chaîne

`Error in configuration file, R key`

R doit être suivi d'un nombre et d'une chaîne, il manque le nombre

`Error in CONTEXT`

La commande CONTEXT du configurateur doit être suivie d'un nom de fichier

`Error in CONTEXT: bad file`

Le nom suivant la commande CONTEXT n'est pas le nom d'un fichier existant

`Error in` $DVI->PS$

$DVI->PS$ doit être suivi par une chaîne

`Error in FOLLOW_MOUSE: invalid chain`

FOLLOW_MOUSE doit être suivi de always, never, variable

`Error in FOLLOW_MOUSE`

FOLLOW_MOUSE doit être suivi d'une chaîne

`Error in FULL_MANUAL`

FULL_MANUAL doit être suivi du chemin d'accès au manuel

`Error in GHOSTVIEW`

GHOSTVIEW doit être suivi du nom de la commande ghostview

`Error in LANGUAGE`

LANGUAGE doit être suivi de la chaîne "french" ou "english"

`Error in UNDO_DIR`

UNDO_DIR doit être suivi du nom du répertoire où est effectué la sauvegarde des fichiers d'Annuler

`Error in UNDO_NUMBER`

UNDO_NUMBER doit être suivi du nombre maximum de primitives neutralisant l'ordre ANNULER

`Error: no argument to BEZIER`

BEZIER doit être suivi par une chaîne

`Error: unknown argument to` $DVI->PS$

L'argument de $DVI->PS$ n'a pas été reconnu

`Error:Bad argument to PS`

L'argument de PS est incorrect

`Error:LIB not followed by argument`

LIB doit être suivi par le nom du répertoire des modules

`Error:no argument to PS`

PS doit être suivi par une chaîne

`LANGUAGE should be french or english in configuration file`

`Local libraries should contain only # BIB`

`Negative number after UNDO_NUMBER in configuration file`

UNDO_NUMBER doit être suivi d'un nombre positif

`Too many local libraries, ignoring`

Le configurateur local contient trop de libraries

`Unknown command in configuration file`

Une commande inconnu a été trouvée

`Warning: no name associated to key`

Une chaîne de caractère doit être associée au nom d'une touche.Vérifier le configurateur

`You have to indicate a configuration file for running xjpdraw..`

Avant de lancer `xjpdraw` il faut que la variable d'environnement JPDRAW indique le configurateur

`Error in EXTERNE (name)`

voir syntaxe de EXTERNE

`Error in EXTERNE (input)`

voir syntaxe de EXTERNE

`Error in EXTERNE (type input)`

voir syntaxe de EXTERNE

`Error in EXTERNE (output)`

voir syntaxe de EXTERNE

`Error in EXTERNE (type output)`

voir syntaxe de EXTERNE

`MINITEL should be followed by 4 int`
La commande minitel doit être suivi de 4 entiers indiquant la taille de la fenêtre
`Error in SELECTION_TYPE`
cette commande doit être suivie de `icon` ou `creation_and_icon`
`Error in SLIDE_HEIGHT`
cette commande doit être suivi d'un nombre indiquant la hauteur d'un transparent
`Error in SLIDE_WIDTH`
cette commande doit être suivi d'un nombre indiquant la largeur d'un transparent

13.2.2 Messages généraux

`Annulation non disponible (repertoire inconnu)`
Les fichiersd'annulation n'ont pu être crées, modifier le UNDO_DIR de votre configurateur
`Annulation non disponible (trop d'objet)`
Il y a trop de primitive pour annuler l'opération
`Annulation non disponible (vous l'avez choisi)`
Vous avez supprimméla possibilité d'annulation, voir le menu PAGE
`Attention: erreur probable dans votre texte`
Votre texte contient une erreur latex
`Attention:cannot open font`
Problème à l'ouverture d'une fonte, l'affichage seule sera incorrecte
`Attention:je ne peux ouvrir des fonts`
Certaines fonts du configurateur n'existe pas
`Attention:Le nombre de primitive est tres proche du nombre maximum`
`xjpdraw` approche du nombre maximum de primitive, essayer de vider le presse-papier
`Attention:Un point d'un polygone est hors de la page`
un polygone sort du cadre, ramener le avec la commande ramener les primitives hors-cadre
`Attention:vous n'avez pas mis d'extension, j'ai rajoute l'extension .ltex`
Lors de la commande sauver vous avez indiqué un fichier sans extension. `xjpdraw` a rajouté l'extension .ltex
`Erreur:Associer seulement le texte et la primitive`
Pour centrer un texte il faut que seul le texte et la primitive où l'on veut le mettre soit associés
`Erreur:attention au nombre maxi de primitives!`
L'opération va créer trop de primitives
`Erreur:Cannot allocate so much primitive, sorry`
Vous avez indiqué un chiffre trop gros dans l'option -M
`Erreur:Cannot find image file`
Lors de l'incorporation d'un fichier Postscript externe il y a eu un problème de création d'image, vérifier si `gs` arrive à lire votre fichier
`Erreur:Cannot incorporate postscript file,GS not defined`
Vous avez demandé d'incorporer un fichier Postscript externe mais la commande GS du configurateur n'est pas défini. Modifier votre configurateur
`Erreur:Ce n'est pas un texte`
L'opération nécessite que la primitive soit un texte
`Erreur:Desole je ne sais pas creer de spline avec ces points de controle`
Le programme de calcul du spline a échoué. Modifier la position des points de contrôle.
`Erreur:help indisponible, verifier MANUAL dans le configurateur`
Vous avez demandé l'aide en ligne mais dans le configurateur l'instruction MANUAL est incorrecte. Vérifier le configurateur
`Erreur:il faudrait qu'il y ait une primitive selectionne!`
Cette opération nécessite de sélectionner une primitive
`Erreur:Il faut associer les polygones a decouper`
Vous avez demandé de découper les polygones en arêtes, mais il faut associer les polygones que vous voulez couper
`Erreur:il faut au moins deux points pour un polygone`
Vous avez tenté de créer un polygon à un seul sommet
`Erreur:Il faut d'abord definir une texture pour la rendre permanente!`
Vous avez essayé de rendre une texture permanente mais vous ne l'avez pas défini. Utiliser le bouton définir et recommencer
`Erreur:il faut selectionner une primitive..`
Cette opération nécessite de sélectionner une primitive

`Erreur:Il n'y a pas de texte a centrer`
Il n'existe pas de primitive de texte dans les associées. La commande centrer un texte le nécessite
`Erreur:Impossible de couper ici`
Le ciseau ne peut pas couper la primitive en ce point
`Erreur:Impossible de couper les textes`
Vous avez essayé d'utiliser le ciseau sur un texte. Utilisez plutôt le bouton L10 suivi de emacs
`Erreur:Impossible de tout couper, trop de primitives seraient crees`
Vous avez demande de couper les polygones associés mais le nombre de primitive qui seraient obtenu après coupure va être trop important
`Erreur:installation incorrecte(pas de header):probleme avec VERIFICATION!`
On n'a pas trouvé de fichier préambule dans le répertoire `LIB`. Les commandes de vérification du menu FICHIER ne fonctionneront pas
`Erreur:Je ne peux pas creer de perpendiculaire ici`
Il n'existe pas de perpendiculaire à la primitive au point sélectionné
`Erreur:Je ne peux pas creer de tangente`
Il n'existe pas de tangente à la primitive au point sélectionné
`Erreur:je ne peux pas ecrire dans cette directory ou trop de fichiers ouverts`
On ne peut pas sauver dans le répertoire courant
`Erreur:Je ne peux pas mettre une fleche sur ce polygone trop petit`
Le polygone est trop petit pour que l'on puisse calculer sa tangente en ce point
`Erreur:Je ne sais centrer que dans un rectangle ou un cercle`
Vous avez demandé de centrer un texte dans un primitive qui n'est pas un rectangle ou un cercle
`Erreur:Je ne sais pas centrer dans un rectangle tourne`
Pour centrer un texte dans un rectangle il ne faut pas que le rectangle soit tourné
`Erreur:La selectionnee n'est pas un polygone`
L'opération demandé impose que la primitive sélectionnée soit un polygone
`Erreur:La selectionnee n'est pas un texte`
L'opération demandé impose que la primitive sélectionnée soit un texte
`Erreur:le fichier n'existe pas`
Le fichier demandé n'existe pas
`Erreur:Le point n'est pas sur la primitive selectionnee`
Le point de coupure pour le ciseau doit être proche de la primitive que l'on veut couper
`Erreur:le polygone a deja le nombre maximum de point`
On ne peut pas créer un nouveau sommet au polygone
`Erreur:Must be a number >1`
Le nombre que l'on doit fournir doit être > 1
`Erreur:Must give a number`
On attend un nombre dans cette opération
`Erreur:No ellipse created`
Problème lors de la création d'une ellipse par 4 points, les 4 points ne sont pas en position générale
`Erreur:No such file`
Le fichier n'a pas été trouvé
`Erreur:nombre primitives dessin album > 1500,essayer de vider le presse-papier`
L'ajout de l'album ferait que le dessin aurait trop de primitives
`Erreur:Not a polygon`
L'opération se fait uniquement sur un polygone
`Erreur:Not possible on an external postscript file`
Opération impossible sur un Postscript externe
`Erreur:Operation impossible sans Extensions`
Opération impossible sans les extensions, passez en mode Extension en cliquant dans le bouton de la fenêtre des menus
`Erreur:Pas assez de point de controle pour la creation d'un Bspline`
Impossible de créer un B-spline avec les points indiqués
`Erreur:Pas assez de point de controle pour la creation d'un spline`
Impossible de créer un spline avec les points indiqués
`Erreur:Pas d'associees`
L'opération nécessite qu'il y ait des primitives associées
`Erreur:pas de changement de forme pour cette primitive!`

On a essayé de changer les dimensions d'un primitives pour laquelle c'est impossible

`Erreur:Pas de fichier de ce nom`

Le fichier demandé n'existe pas

`Erreur:Pas de marque pour cette touche!`

Vous avez appuyé sur un touche chiffre mais le il n'existe pas de marque associé à cette touche

`Erreur:Pas de polygone associe`

L'opération demandée impose qu'il y ait des polygones associés

`Erreur:pas de premier plan pour ce type`

La primitive sélectionnée ne peut pas être mis au premier plan

`Erreur:pas de primitive selectionnee`

Cette opération nécessite de sélectionner une primitive

`Erreur:Pas de primitive selectionnee`

L'opération demandée impose qu'il y ait une primitive sélectionnée

`Erreur:Pas de rotation des postcripts externes`

Il est impossible de faire tourner un postscript externe

`Erreur:pas de texture pour ce type de primitive`

On ne peut pas texturer la primitive sélectionnée

`Erreur:Pas de zoom pour les poscripts externes`

On ne peut pas zoomer un postscript externe. Vous pouvez seulement faire un zoom de toute la figure en changeant le unitlength

`Erreur:Pas d'homothetie sur ce type de primitive`

Copier avec homothétie n'est pas possible sur ce type de primitive

`Erreur:Plus de place pour definir une nouvelle marque!`

Vous avez déjà défini 10 marques, qui est le nombre maximum

`Erreur:Probleme d'ouverture de fichier`

Le fichier demandé n'existe probablement pas ou vous avez trop d'applications ouvertes

`Erreur:Probleme possible pour la compression du .ps`

La commande compress n'a pas fonctionné, compressez votre fichier à la main

`Erreur:seul les fichiers d'extension .macdraw sont lisibles`

On a essayé de lire un fichier d'extension différente de .macdraw

`Erreur:The command GS is not defined, cannot include postscript`

Vous avez demandé d'inclure un fichier postscript externe mais la commande GS n'a pas été défini dans le configurateur. Modifier le configurateur

`Erreur:The number must be >0`

On doit fournir un nombre > à 0

`Erreur:trop de caractere!`

On a essayé d'introduire un texte trop long

`Erreur:Trop de ligne de texture`

La texture va créer trop de lignes

`Erreur:trop de point dans le trace au crayon`

On a fait un tracé à la souris trop longtemps

`Erreur:trop de point d'intersection pour cette ligne de texture!`

la coupure de la texture s'est mal faite

`Erreur:Trop de point pour creer le polygone`

Les associées définissent plus de 30 points, nombre de sommets maximum pour un polygone

`Erreur:trop de point pour trouver le polynome`

On a donné trop de point de contrôle pour un polynôme

`Erreur:Trop de primitive pour couper`

On ne peut pas créer de primitives en coupant car cela créerait trop de primitives

`Erreur:Type incompatible pour la duplication`

L'opération dupliquer impose que les primitives associées soit du même type

`Erreur:Une primitive hors cadre,je ne peux pas centrer.`

Le dessin est trop gros pour que l'on puisse le centrer, utilisez le zoom

`Erreur:Vous devez associer au moins 2 primitives, la 1ere etant la fixe`

Pour aligner il faut associer les primitives à aligner

`Erreur:Vous devez d'abord associer le texte puis la primitive`

Pour centrer un texte dans une primitive associez les deux éléments

`Erreur:Vous devez d'abord selectionner un rectangle ou un polygone`

On peut associer les primitives internes seulement à un rectangle ou à un polygone

`Erreur:Vous devez d'abord selectionnez la primitive a texturer`

Pour texturer une primitive il faut d'abord la sélectionner

`Erreur:Vous devez d'abord selectionnez la primitive a tourner`

Pour faire tourner une primitive il faut d'abord la sélectionner

`Erreur:Vous n'avez pas donne soit la formule,soit X0, soit X1, soit le pas!`

Pour tracer une courbe analytique il faut donner: la gamme sur le x et le pas sur le x

`Erreur:vous ne pouvez pas mettre un angle nul ici`

pas d'angle nul pour une texture inclinée

`Erreur:vous ne pouvez pas sauver dans un fichier d'extension .macdraw`

On a tenté de sauver le dessin dans un fichier d'extension .macdraw. Il faut donner une autre extension

`help indisponible, verifier MANUAL dans le configurateur`

`Impossible pour les postscript externes`

Opération non permise

`Je ne trouve pas le fichier image`

L'incorporation d'un postscript externe passe par la création d'une image ppm, qui na pu être crée

`La commande GS n'est pas defini, je ne peux rien faire`

Pour cette opération il est nécessaire d'utiliser GS, vérifier votre configurateur

`Probleme durant le calcul de la taille du textte`

Le calcul des dimensions du texte s'est mal passé

`Probleme lors du calcul des dimensions`

Le calcul des dimensions du texte s'est mal passé

`Vous devez associer les lignes`

Pour découper des lignes il faut les associer d'abord

`Vous devez donner un nombre`

`Zoom trop grand`

Après un tel zoom la taille du dessin excédera les dimensions de la fenêtre

13.2.3 Messages en mode Fonction utilisateur

`Attention:mode connecte ou deconnecte sans point`

les modes deconnecte ou connecte doivent être suivis de points

`Attention:mode spline sans argument`

pas de point de contrôle pour un spline

`Attention:pas de texte dans un mode texte`

le mode texte doit être suivi d'un texte

`Attention:polygone a moins de deux points`

un mode polygone doit être suivi d'au moins deux points

`Attention:rayons non egaux en mode arc, je prend le premier comme rayon`

le mode arc doit être suivi des coordonnées du centre du cercle puis de 2 valeurs identiques qui sont le rayon du cercle

`Attention:style incorrect en mode texte, je met 10 pt`

le mode texte est suivi d'un nombre qui indique le style. Le nombre trouvé ne correspond pas à un style connu

`Erreur:Trop de point pour un spline : je tronque`

On a donné trop de point de contrôle pour définir un spline

`Erreur:Vous devez d'abord selectionner le rectangle ou vous allez placer le trace`

Avant de tracer une fonction il faut sélectionner le rectangle dans lequel se fera le tracé

`Erreur:coordonnees incorrectes en mode polygone`

des coordonnées sont incorrectes dans le mode

`Erreur:d'argument:definition incorrecte pour une fonction`

les arguments de `xjpdraw` au lancement sont incorrects

`Erreur:dans -X (compris entre 0 et 14.5)`

L'argument de -X doit être compris entre 0 et 14.5

`Erreur:dans -Y (compris entre 0 et 14.5)`

L'argument de -Y doit être compris entre 0 et 14.5

`Erreur:dans la position du texte`

vérifier les coordonnées de ce mode texte

`Erreur:dans le centre en mode cercle`

vérifier les coordonnées qui suivent le mode cercle

`Erreur:dans le rayon en mode cercle`

Vérifier le rayon, deuxième ligne après le mode cercle

`Erreur:dans origine d'une fonction (dessin hors cadre 14.5x14.5)`

les arguments -O, -X, -Y doivent conduire à un dessin inclu dans un rectangle de 14.5 cm au maximum

`Erreur:le format des echelles est incorrect`

Vérifier le format des échelles que vous donnez pour la fonction

`Erreur:le format donne n'est pas correct,je le remplace par 6.2`

Le format des textes est incorrect

`Erreur:mauvais centre en mode arc`

Vérifier les coordonnées du centre qui suivent le mode cercle

`Erreur:mauvais parametre d'angle en mode arc`

Vérifier les angles du mode arc

`Erreur:mauvais rayon en mode arc`

Vérifier le rayon du cercle du mode arc

`Erreur:mauvais type en mode polygone`

Vérifier le nombre qui suit le mode polygone

`Erreur:mauvaise hauteur ou largeur en mode rectangle`

Vérifier les dimensions du mode rectangle

`Erreur:mode inconnu`

Un mode du fichier fonction est inconnu

`Erreur:pas de parametre en mode rectangle`

Vérifier les paramètres du mode rectangle : on doit avoir origine et largeur et hauteur

`Erreur:trop de points dans un polygone (mettez -1 dans le nombre)`

Le nombre indiqué de points d'un polygone ne correspond pas au nombre de points fournis

`Erreur:trop de points de controle dans un spline`

On a fourni trop de point de contrôle dans un spline

13.2.4 Messages en mode Programmation

`Erreur:Pas de point souris en mode invisible!`

Vous êtes en mode invisible et dans le programme vous avez demandé l'introduction de point à la souris

`Erreur:Trop de points souris`

On a demandé trop de point souris

Index

T

U

V

Contents