jfig

Tutorial and User–Guide

Norman Hendrich
University of Hamburg
Department of Informatics

Contents

List of Figures

1 Getting started — jfig in a nutshell

1.1 About jfig

history

Originally a small programming project to learn about object-oriented programming, the first version of *jfig* was written in 1997 as a simple clone of the `xfig` graphics editor. Both programs have since evolved independently, but retain the original user-interface concepts. `xfig` is still a popular 2D-graphics and diagram editor for the X-windows (X11) system. It is available on many Unix and Linux systems, and can also be made to run on Windows (using a rather complex setup). Check `http://www.xfig.org` for documentation and downloads.

jfig is a Java-written 2D-graphics editor based on the FIG file format. It runs on all Java compatible platforms, including PCs or notebooks running Windows 9X/NT/2000/XP, but also Apple Macintosh computers and most Unix or Linux workstations.

features

Just browse this tutorial to get some idea about what kinds of figures you can draw with *jfig*. The editor supports all common drawing primitives, including boxes, circles, ellipses, polylines, arcs, splines, text objects, and embedded bitmap images. Many attributes like colors, line width, line style, and different arrowheads are provided. However, *jfig* is mostly intended for scientific and technical figures. There is no support for 3D-functions, shading, transparency, etc. If you are willing to do a little bit of programming, you can also use *jfig* as a class library to include vector graphics into your own Java applications and to create your own custom graphics editors; see chapter 7 for a few examples.

TEX

Several features, including TEX font and macro support, make *jfig* a popular editor to create figures for TEX and LATEX documents. A companion program called *fig2dev* provides export filters to several popular file formats including Postscript, PDF, LATEX picture environments, and several bitmap formats.

homepage

You can download the current version of the *jfig* editor, documentation, and several tools, including precompiled versions of the *fig2dev* converter program, from the *jfig* homepage, `http://tams-www.informatik.uni-hamburg.de/applets/jfig/`.

license

Please note that *jfig* is released as a *shareware program*: If you use the program regularly, you should register. See the FAQ in section 9 for additional questions regarding *jfig*.

1.2 The right editor for you?

pro

If you are asking yourself why *to use jfig* at all, the following list has some arguments:

- you know and like `xfig` and need a replacement for your Windows-based notebooks or desktop PCs.
- you like TEX or LATEX and want a graphical editor for your figures. For TEX formatted texts overlayed on graphical objects, `xfig` and *jfig* may be your only choices.
- you want a cross-platform graphical editor for use on all your Windows, Unix/Linux, and Macintosh-systems.
- you want a simple and fast editor for technical diagrams and figures, where things like object alignment, arrows, and quick attribute manipulation are more important than fancy fonts, shadows, or alpha-transparency.
- you need a graphical editor with math support. (Naturally, Matlab or Mathematica are fine programs, but they are also somewhat expensive.)

- you need access to a text-based, fully-documented, and simple file format in order to process those files via your own tools and converters. Use *jfig* as the interactive editor to manipulate the figures written by your programs.
- you want to embed a viewer for vector graphics into webpages or into your own Java-based applications.
- you need a custom graphical editor and don't have the time to write one yourself.

contra On the other hand, here are a few reasons *not* to use jfig:

- you want to edit and manipulate very large drawings, say 10.000 objects or more, or you need 3D- or pseudo-3D perspective functions. Please consider using a fully-featured CAD program.
- you often need advanced vector graphics functions like gradient fill-patterns, alpha-transparency, or texts aligned to curves.

  Note that most attributes and fancy rendering tricks could easily be implemented using current Java2D functions, but this would imply a new file format. Also, writing the user-interface code would be a lot of work.
- you need a seemless integration into Windows software like Microsoft Office. Unfortunately, not one of the many vector output formats supported by *jfig* via Java printing or the *fig2dev* converter program is a good choice for integration into MS Office documents. The best option is to embed Postscript or GIF- or PNG-format images.
- you are accustomed to "standard" Windows and Macintosh graphical editors with their object-based user paradigm (first, select an object by clicking, then do something with the object). While the mode-oriented paradigm used by *jfig* is arguably faster to use, you will probably never like it.

1.3 Jumpstart

one-click installation If you have a recent Java virtual machine installed on your computer (namely JDK/JRE 1.4.2 or higher), the quickest way to download, install, and start *jfig* is via the Java Webstart launcher. Just visit the *jfig* homepage,

`http://tams-www.informatik.uni-hamburg.de/applets/jfig/index.html`

read and agree to the software license, and then click on the **download and run jfig** button. This will start your Java virtual machine, download the current version of *jfig2.jar* from our webserver, check the software for integrity via a digital signature, and then ask you whether to run the program.

On Windows, the Webstart launcher will also offer to create a *start menu entry* and *desktop icon* for *jfig*. Once the editor has started, the main window will look similar to the annotated screenshot shown in figure 3.

or manual download Please read section 8.2 on page 101 for a detailed explanation, if you don't like the *security warning* dialog presented by the Java webstart launcher. If in doubt, download the *jfig2.jar* archive file manually. Just visit the *jfig* homepage (see above) and follow the *download* link to go to the *download section*. Use your browser's *save link as* function to download the *jfig2.jar* archive with the current version of the editor.

You will also find links to the different variants of *jfig*, utilities for desktop integration, and precompiled versions of the *fig2dev* converter for Windows and Mac OS platforms. Select and download the software archives that you need. See section 8.3 for a detailed explanation of the available files and section 8.4 for instructions on how to run the editor after the download.

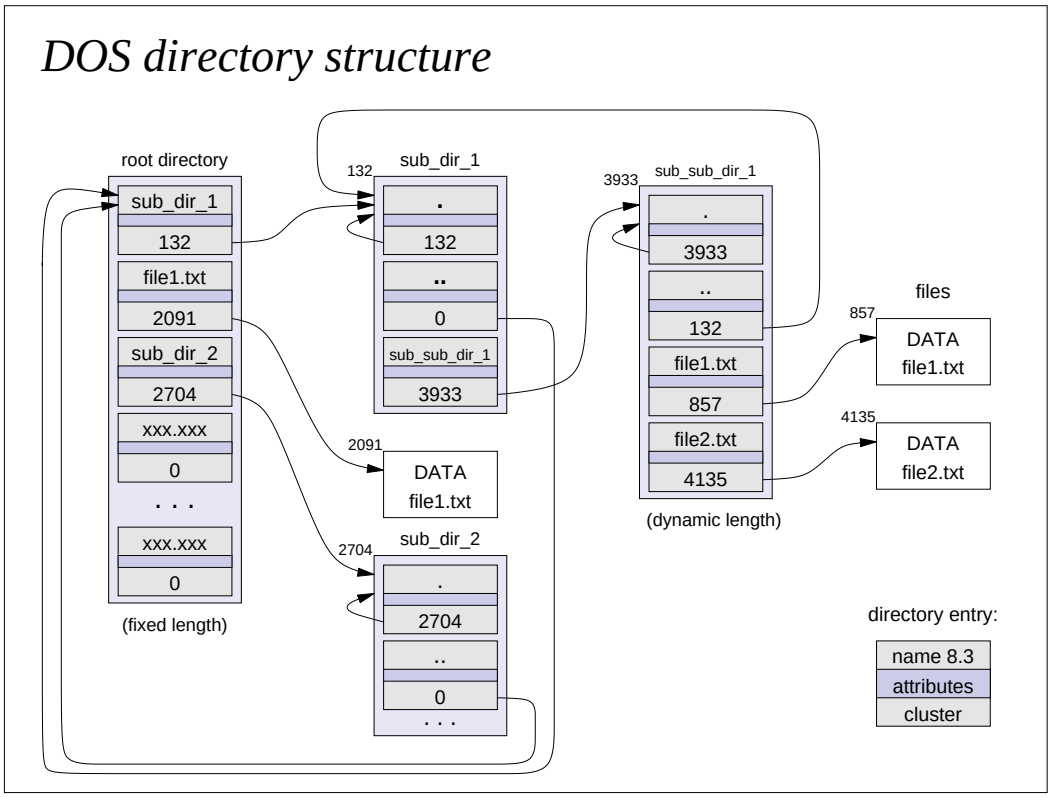

Figure 1: *Example figure: DOS directory structure. The figure shows layered and filled rectangles, splines with arrows, and text objects with several fonts.*

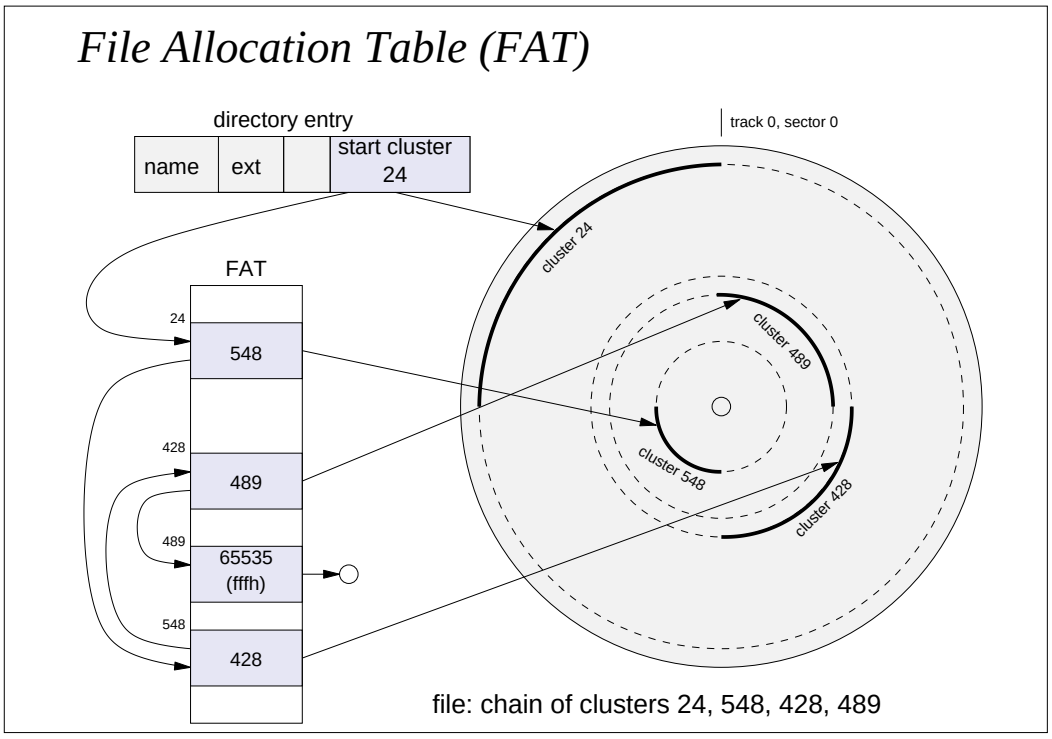

Figure 2: *Example figure: DOS file allocation table. Polylines, circles, arc objects with different line and fill styles are used.*

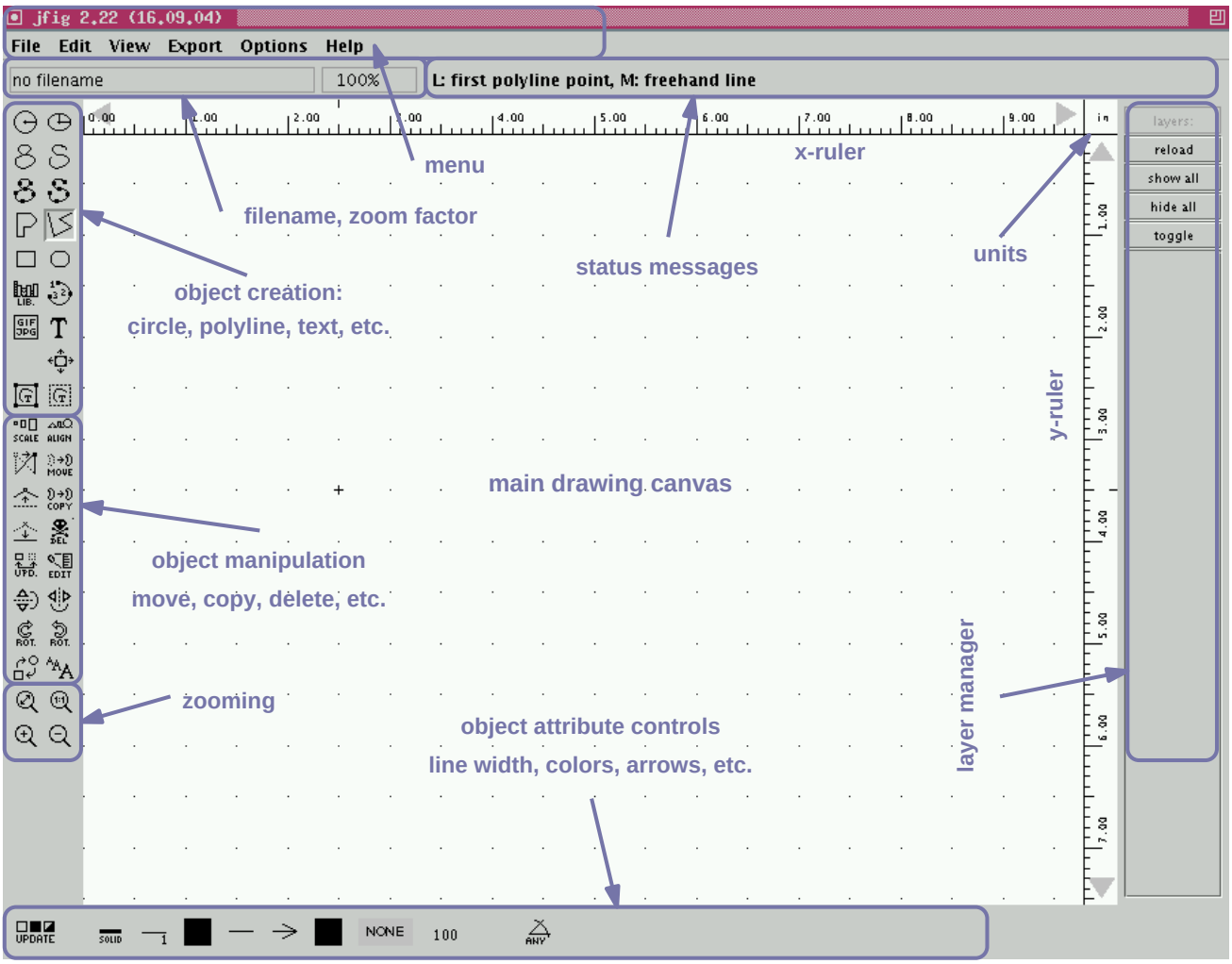

Figure 3: *Screenshot of the main window of the jfig editor. The center of the window is used by the main drawing area, including two rulers that show the current coordinates. Above it are a few controls to indicate the filename, zoom-factor and a short summary of the current editor operation. To the left are the command buttons to select the edit modes, e.g. create polylines, move objects or update object attributes and four zoom-buttons. At the bottom are several controls to specify the current object attributes, e.g. object color, line type, or font selection. The layer-manager to the right controls the visibility of objects via their layer (depth).*

1.4 The user interface

The user-interface of the editor consists of several parts. See figure 3 for a screenshot taken on a Linux system running the *ctwm* window manager. Depending on your operating system and the selected *look and feel*, the editor and window decorations may look different, but the basic layout stays the same:

- the center of the window is used by the **main drawing area**, including two rulers that show the current coordinates; section 2.1 describes the different *zooming* and *panning* commands and the display options lie *grid* and *magnetic grid*.
- the upper part of the window contains the **menu bar** with its *file*, *edit*, *view*, *export*, *options*, and *help* menus. Between menu bar and drawing area lie a few controls to indicate the current *filename* and *zoom factor*, and the **status message** area, which always provides a short summary of the current editor operation.

- the **command panel** (or *mode button panel*) on the left holds two columns of buttons used to select the several *drawing operations* (or *editor modes*). The individual edit operations are described in chapters 3 and 4.
- the **attribute panel** on the bottom of the editor window contains the *attribute buttons* used to select the drawing attributes (color, line width, arrows, fonts). For each edit operation, only the relevant attribute controls are enabled; see section 2.2.
- the **layer manager** on the right side allows you to show or hide objects based on their *layer* attribute; see section 2.3 for an explanation.

1.5 Using the mouse

three buttons

The user interface of *jfig* is optimized for a *three-button mouse*, but you can also use the editor with a two-button mouse (many Windows PCs) or a one-button mouse (Macintosh computers). Sections 2.1.6 and 2.1.7 describe how to use the mouse and how to customize the mouse configuration in detail.

left: L
shift+left: M
cntl+left: R

In the default configuration of the editor, you can hold down the `[shift]`-key and click the left (or only) mouse button instead of clicking the *middle* mouse button. Similarly, you hold down the `[control]`-key and click the left mouse button instead of clicking the *right* mouse button. With a little bit of practice, this works quite well — at least for right-handed persons. In the remainder of this tutorial, a three-button mouse is assumed and the buttons are called *left*, *middle*, and *right* (L,M,R).

new: L
complete: M
cancel: R

The main advantage of using different mouse buttons is that you can select several different functions easily and very quickly. For most drawing commands, pressing the left button will *execute* the command (e.g., creating a new polyline) and the right button *cancels* it, while the middle mouse button is used to *complete* the command or to specify options.

select: L
decrease: M
increase: R

Also, most object attribute controls use all three buttons, left to open a dialog window that allows *selecting* an attribute value, middle to *decrement* the value, and right to *increment* the value. For example, use a *right* click on the *line width button* to change the line-width from 1 to 2, etc.

1.6 Drawing

step-by-step

This section is intended as a step-by-step explanation of a few important drawing and editing commands. You might want to repeat the steps in the live editor. See chapter 3 for the complete list of commands to *create* drawing objects and chapter 4 for all commands to *edit* existing drawing commands.

edit modes

Unlike many Windows-based graphics editors, the user interface of *jfig* is mode-oriented. This requires some practice to get used to, especially if you are accustomed to "standard" graphics editors with their object-based user paradigm (first, select an object by clicking, then do something with the object), but it allows for very fast drawing. For each of the basic graphical primitives, e.g. circles or polylines, a separate edit mode is provided to create the corresponding objects. The *mode-button panel* on the left side of the editor includes a separate button for most of these edit modes.

status
cancel
undo

At any time, the *status area* on top of the editor indicates the current drawing operations and the actions bound to the three mouse buttons, see below for a few examples. If necessary, select the *menu* ▷ *edit* ▷ *cancel* menu item or type the `[ESC]` key to unconditionally cancel an ongoing edit operation. You can also select the *menu* ▷ *edit* ▷ *undo* menu item to undo any previous edit operation.

preparation To prepare for drawing, select the *menu ▷ view ▷ grid ▷ medium grid* menu item to enable the background grid on the drawing canvas, and the *menu ▷ view ▷ magnetic grid ▷ 1/8 grid* menu-item to enable the magnetic grid, which helps to accurately place and align drawing objects. For details about the grid, see section 2.1.4.

Now, click the *create polyline* button in the *mode button panel* on the left of the window to enter *create polyline mode*. Note that the *status message* changes and lists your options for drawing polylines. The text *L: first polyline point M: freehand line* tells you that clicking the left mouse button will place the first point of a polyline, while clicking the middle button will start the drawing of a freehand line.

a first polyline Now, *click the left mouse* button anywhere to place the first point of the new polyline. The status message changes to *L: next point M: final point R: cancel*, indicating your options: add a new vertex to the polyline via a left click, place the final vertex of the polyline via a middle click, or cancel the operation via a right click. Also, the drawing canvas now uses a rubberband to indicate the current polyline segment. Try adding some intermediate points to your polyline by clicking the *left button* again a few times, then click the *middle button* to place the final point of the polyline.

selecting attributes You can now click the *left button* again to immediately draw the next polyline; click the *right button* to cancel the operation. Note that the editor remains in *create polyline* mode. Move to the attribute panel on the bottom of the editor window and change some line attributes; for example *left click* the *line-color button*, wait for the *select color* dialog and select one of the colors. Now, *right click* the *arrow mode* attribute button (the one left of the *arrow style* button) three times to switch from the *no arrow* setting to the *both arrows* setting. Also select an *arrow style*.

an arrow Move the mouse back to the drawing area and *left click* to place the first point of the next polyline, then use a *middle click* to complete the polyline. It will use the *color* and *arrow style* you selected.

Now, select the *create rectangle button* in the mode button panel to start creating rectangles. Again, the status message changes and lists your options. The attribute controls change, too, and only show the attributes available for rectangles. Use a *left click* to place the first corner of your rectangle. The drawing canvas rubberbanding changes to rectangle mode. Move the mouse and *left click* again to place the opposite corner of your rectangle. The next *left click* will instantly place the first corner of the next rectangle, etc.

Click the *create text button* in the mode button panel to enter *create text* mode. Use a *left click* to place the *base point* of a new text object; the drawing canvas now shows a vertical line to indicate the text cursor. Start typing in the drawing area to enter the actual text for the new text object. You can use the delete, backspace, and cursor keys to edit and navigate in the text. Press the `[return]` (enter) key to finish the text object and start a new text object immediately below the first text object. Move the mouse and *left click* again to place and start a new text object. Play with the attribute controls to select the *text font*, *font size*, and *text alignment* (left, centered, right).

shortcuts As soon as you become familiar with the drawing operations in *jfig*, you will also want to use the *keyboard shortcuts* (*accellerators*) to switch between edit-modes. Selecting an edit-mode via a simple keystroke is much faster than moving the mouse across the screen to click on one of the edit-mode buttons on the far left before moving the mouse right back to the drawing canvas. See section 2.1.8 for details.

In *create text* mode, all keyboard input to the drawing area is used to edit the current text object; therefore, you can neither use the cursor keys for panning the drawing area, nor use the shortcut keys to change the edit mode. Click one of the buttons in the mode button panel or type the `[ESC]` key to leave text mode.

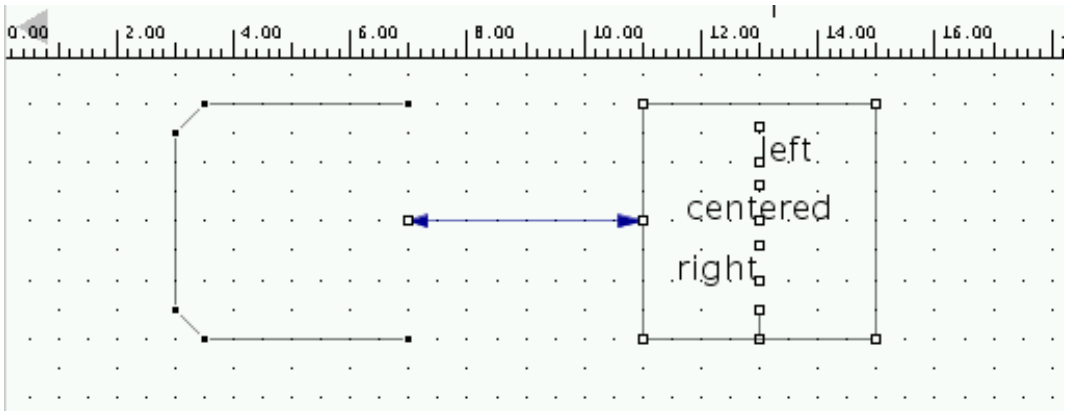

Figure 4: *Selecting objects. In each of the edit object modes, the markers of all editable objects are shown via small open rectangles. Click near a marker to select an object. Selected objects are shown with highlighted, filled markers (here, the polyline on the left).*

1.7 Editing

A variety of *edit modes* allow *modifying* existing objects, that is, to change their position, orientation, structure, or attributes. For example, you can *move*, *copy*, *scale*, *rotate* and *delete* objects. See chapter 4 for the complete list and description of each edit operation.

object selection

To *modify* an object, you must first *select* it. See figure 4 for the example drawing prepared in the previous paragraph. It contains a polyline, an arrow, a rectangle (box), and three text objects with different text-alignment attributes. The short vertical line inside the box indicates the x-position of the three text objects.

markers

To select an object, you *left click* on one of its *markers*; these are the small *outlined markers* at the corners of the box, the ends of the arrow, and the base points of the text objects. The small *filled markers* at the vertices of the polyline on the left indicate an already *selected* object.

When you click into the figure to select an object in one of the edit modes, the editor internally calculates the distance between the mouse-click position and the positions of all object markers, and the object with the smallest distance is used. To reduce the computation complexity, objects further away than two times the current magnetic grid spacing are discarded from this search.

For example, you would click on one of the rectangle corners in the above figure to select the rectangle, while clicking inside the rectangle would (probably) select one of the text objects. Similarly, selecting a polyline (or spline or arc) requires to click on one of the polyline vertices, while clicking on a line segment between vertices will select nothing.

To test this behavior of selecting objects, click the *move object button* in the mode button panel to enter *move object* mode. Again, the status message changes and indicates your options, while the drawing canvas shows all object markers. Move the mouse to one of the object markers and *left click* to select the corresponding object. The rubberband changes to the bounding box of the selected object. Move the mouse to the new position of the object and *left click* again to place the object. Using a *middle click* to pick up an object selects *restricted move mode*, which allows for accurate horizontal and vertical moves. Try moving a few objects.

Now click the *copy object button* in the mode button panel to enter *copy object* mode. Use a *left click* to mark the object to be copied, move the mouse to the target location, and *left click* again to drop the copy. Using a *middle click* to mark the object allows a restricted horizontal/vertical copy operation.

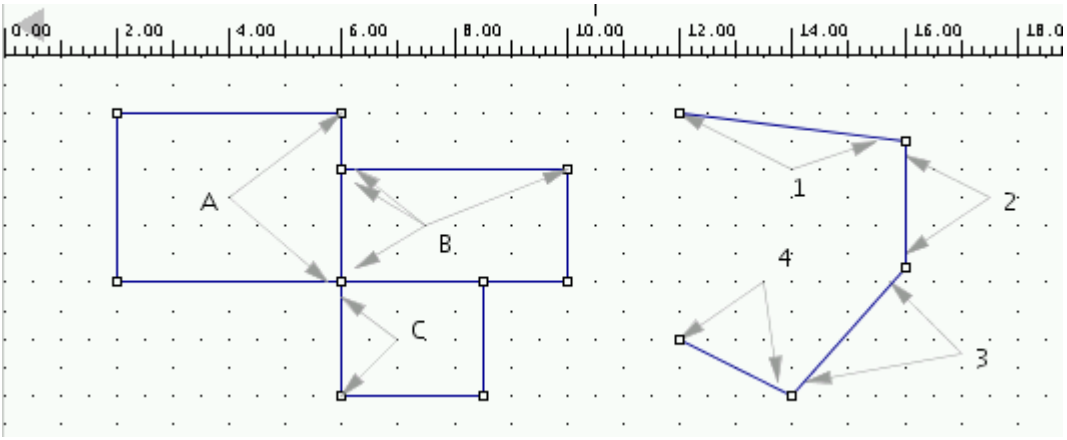

Figure 5: *Selecting abutted objects. When several objects overlap, selecting the right object can be difficult. Try to click on unique points of the target object, or on one of the shared points, but slightly to the inside of the target object. For example, click at the position indicated by the A arrows to select rectangle A, or the 3 arrows file polyline segment 3.*

To *delete* an object select *delete object mode* via the corresponding button, then use a *left click* to delete the marked object. You can also use a *middle click* to specify the first corner and another *left click* to specify the opposite corner of a delete region; all objects fully inside that region will be deleted. Use *menu* ▷ *edit* ▷ *undo* to restore the deleted object(s).

Click the *move point button* to move individual vertices on polyline, spline, or arc objects. Use a *left click* to select both the polyline and the vertex to move, then another *left click* to place that vertex. When used on circles, ellipses, and rectangles, the *move point* works like a scaling operation. Similar point-operations allow inserting or deleting vertices on polylines.

1.8 Loading and saving your figures

save To *save* your current figure, select the *menu* ▷ *file* ▷ *save* menu item from the editor menu or type the `[cntl]+[s]` bindkey. Select the *save as...* menu item from the *file* menu to select the file name before saving the figure. You will also be prompted for a file name when a new figure (which has no file name yet) is first saved.

open Select the *menu* ▷ *file* ▷ *open file...* menu item or type the `[cntl]+[o]` bindkey to open and load an existing figure file into the editor. The *file dialog* window is configured to show all files with the *.fig*, *.jfig*, or *.txt* extensions. You can also load figure files directly from URLs via the *menu* ▷ *file* ▷ *open URL...* menu item. Naturally, the editor will prompt you to save an existing figure file before deleting all objects for the new figure. The editor also maintains a list of four recently opened files in its *file*-menu. Just select one of these menu items to open the corresponding file.

To create a *new and empty figure* either select the *menu* ▷ *file* ▷ *new* menu item or type the `[cntl]+[n]` bindkey. Again, the editor will prompt you to save an existing figure.

merge The *menu* ▷ *file* ▷ *merge file...* and *merge URL* menu items allow loading the selected figure file into the current figure. The merged objects are encapsulated in a compound object, allowing you to easily move the extra objects. Note that you can also load or merge a plain text file. In this case, the editor will automatically convert each line of text from the file into one text object. Use this feature to quickly import short listings into your figures.

quick-open If you begin to use *jfig* frequently, you will probably like the *quick-open* feature. Just press the `[page down]` or `[page up]` keys to open the previous or next figure file in the

current working directory. This allows you to browse through your files very quickly, especially when you follow a naming convention that keeps the figure files sorted. If you have made some changes in the current file, simply press `[cntl]+[s]` to save it, then press `[page down]` to open and edit the next file.

As usual, select *menu* ▷ *file* ▷ *quit* or type the `[cntl]+[q]` bindkey to exit the editor. You will be prompted to save any as yet unsaved changes. *quit*

1.9 Printing and exporting

Naturally, you may want to *print* your figures or *export* them to another format to integrate them into your documents. For example, *jfig* is often used to create figures for TEX and LATEX-documents. The Java-based printing functions are selected from the file menu via *menu* ▷ *file* ▷ *print (java native)* and *print (fig2dev)*, while the remaining export functions are available via the *menu* ▷ *export* menu.

At the moment, you have the following options:

- use the **Java printing** functions to access all printers installed on your system. For example, to create Postscript or PDF files, simply install a Postscript printer driver or the Adobe destiller. Then select the *menu* ▷ *file* ▷ *print (java native)* menu item, wait for the *print dialog*, then select the postscript printer, and finally select *print to file*.
- use the **built-in bitmap export** options to create bitmap images ("screenshots") from your figure files. Supported image formats include JPG, PNG, and PPM. If necessary, use external tools (like Photoshop or Gimp) to convert the output images to other image formats.
- create **PDF** files from your figures via the Java-based **iText** library. This feature is only enabled in the registered version of *jfig*.
- install and use the **fig2dev** converter program to create a variety of vector-graphics output formats, including Postscript, PDF, bitmap images, or LATEX picture environments.

Chapter 6 describes the printing and exporting options, including the setup and usage of the *fig2dev* converter in some detail. In general, using the built-in printing functions is easy, but the output quality generated by the Java printing functions may be lower than the output quality of *fig2dev* or *iText*, especially when running an older Java virtual machine. Using JDK/JRE 1.4.2 or higher is recommended for the Java-based printing functions.

1.10 Configuration

You can customize the appearance of *jfig* and many editor settings via entries in *configuration files*. This includes the selection of the editor *look and file* via the corresponding mechanism of the Java Swing user-interface library. Please see section 2.5 on page 38 for an explanation of the concepts and the syntax of the configuration files. An example configuration file is listed in appendix E. *SetupManager*

1.11 Troubleshooting

Given a recent Java virtual machine and a fairly modern computer, *jfig* should run out-of-the-box without any major problems. While four problems are reported quite frequently, all of them have a simple fix: *four typical problems*

- problems in **drawing polylines** (and similar objects) are caused by a misunderstanding of the *three-mouse-buttons* user-interface. Please remember to use the *middle mouse button* to complete the *create polyline* drawing command. Or hold down the *shift*-key and *click the left mouse button* as a replacement for the *middle mouse button*; this should work on most current computers. See sections 2.1.6 and 2.1.7 for tips and details.
- problems in **adding arrowheads** to polylines or arc-objects highlight a bad user-interface design — the default icon for the *arrow-mode none* setting shows a plain line and doesn't indicate the arrows at all. Please remember to select both an *arrow-mode* and an *arrow-type* for your polyline to enable arrows.
- problems with **object rotation** are usually caused by leaving the *rotation angle button* in its default setting of zero degrees. Just select a non-zero rotation angle before trying to rotate an object; see section 4.5 for details.
- problems to run the **fig2dev** converter program are usually due to a wrong setup or version conflicts. Please see section 6.4 for details and tips.

Installation Please consult section 8 for tips about the installation and setup of *jfig* on several Java virtual machines. While the recommended setup is to use a recent version of Java like J2SE 5.0 or JRE 1.4.2, you can also run *jfig* with older or experimental Java virtual machines.

FAQ Last but not least, see the FAQ in section 9 for a list of the most frequently asked questions about *jfig* and some answers. Please check the *known bugs* section at the end of the FAQ before submitting new bug-reports, but don't hesitate to ask for new features or bug-fixes in *jfig*. Basically, the functions in the editor that I use regularly are supposed to work, while the remaining functions receive less care.

2 Concepts

This chapter introduces the concepts underlying the FIG figure file format and describes the user-interface and controls of the editor. The first section 2.1 concentrates on various aspects of the *user-interface*. Please read (at least) the part about *using the mouse* (subsection 2.1.6) before trying to use *jfig*, because the editor expects and uses a *three-button* mouse for most drawing operations. The later sections of this chapter, starting with section 2.2 on page 20, present an overview of the different types of objects and attributes available in *jfig*. The individual editor commands to *create* new drawing objects are the topic of the next chapter 3, while the commands to *edit and manipulate* existing objects are described in chapter 4.

2.1 User-interface

This section introduces the user-interface of the editor:

- **coordinate system**, see subsection 2.1.1
- **units and rulers**, see subsection 2.1.2
- **zooming and panning**, see subsection 2.1.3
- **magnetic grid**, see subsection 2.1.4
- **rendering options**, see subsection 2.1.5
- **using the mouse**, see subsection 2.1.6
- **mouse configuration**, see subsection 2.1.7
- **keyboard shortcuts**, see subsection 2.1.8
- **undo and redo**, see subsection 2.1.9

2.1.1 Coordinate system

2400 dpi

The coordinate system used by both `xfig` and *jfig* is based on cartesian integer coordinates with the origin at the upper left corner of the drawing region and a fixed resolution of 2400 dpi (dots per inch). See figure 6 for an example. In the screenshot, the origin of the coordinate system is exactly in the top left corner of the drawing region, while the upper left corner of the black rectangle is at position (2400,2400) and its lower left corner at position (7200,4800). As indiciated by the dimensioning arrows, the rectangle is exactly two inches wide and one inch high.

range

Internally, *jfig* uses 32-bits signed integer variables to represent point coordinates. This implies that the drawing area is limited by the values ($\pm 2\,147\,483\,647, \pm 2\,147\,483\,647$). Each x- and y-value lies in the range of approximately $\pm 894\,000$ inches or ± 22700 meters while the default resolution of 2400 dpi should be sufficient to allow pixel-accurate positioning of objects on current laser- or ink-jet printers. In theory, the FIG 3.2 format specification also supports other resolutions and the origin in the lower left corner of the drawing region, but these values are not used by any FIG-based tool. For best compatibility with external tools, you might also want to avoid negative coordinates.

jfig assumes a fixed display resolution of 75 dpi. This provides an accetable approximation of the actual display resolution of traditional CRT-type and many LCD-type monitors, but it ignores the exact values provided by many current operating systems (like Windows XP or Linux). Using a zoom-factor of 1.0 therefore results in the display of a figure at magnification 75/2400 (screen resolution vs. internal resolution of the file coordinates). Note that the assumption of a 75 dpi screen is no longer valid for the very high resolution LCD monitors (SXGA+ or better) found on many current notebooks and desktop displays.

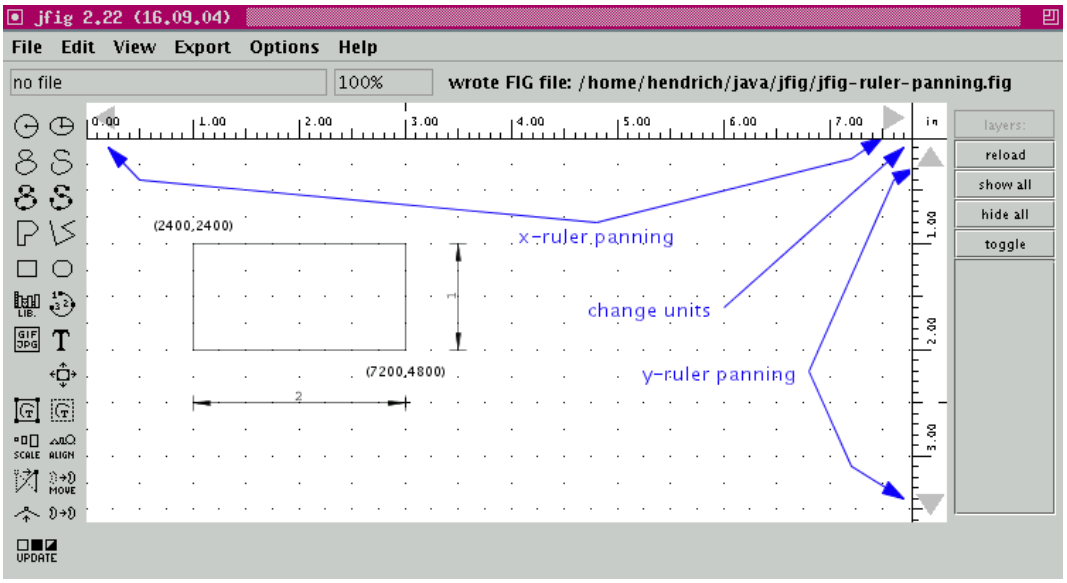

Figure 6: *The coordinate system used by jfig is based on integer coordinates with the origin at the upper left and a fixed resolution of 2400 dpi. Therefore, the upper left corner of the rectangle is at position (2400,2400) and its lower left corner at (7200,4800) — the rectangle is exactly two inches wide and one inch high. Use a left mouse button click in the 'units' region in the upper right corner of the drawing canvas to toggle the units display in the rulers and grid between inches, millimeters, and xmm. Click the small triangles inside the rulers for panning the figure.*

2.1.2 Units and rulers

units As explained above, the coordinate system is based on integer coordinates at a fixed resolution of 2400 dpi. Therefore, *inches* are the basic *units* for calculating and labeling the *rulers* and the *background grid* on the drawing canvas. However, *jfig* also supports two additional settings for the *rulers* and *grid* which can be helpful when preparing figures based on the standard metric units. The following three settings are possible:

inches
- *inches* (also called *imperial* units in `xfig`): given the internal resolution of 2400 dpi and the assumed display resolution of 75 dpi, one inch corresponds to excactly 32 pixels at zoom-factor 1.0.

xmm
- *xmm*: this setting is provided for backwards compatibility with existing `xfig` figures and corresponds to the *metric units* in `xfig`. A conversion factor of 80 is used instead of the "real" conversion factor of $2400/25.4 \approx 94.488$; resulting in a relative error of $(94.488 - 80)/80 \approx 18\%$.

mm
- *mm*: this setting gives a better approximation to real millimeters than the *xmm* setting. A conversion factor of 96 is used instead of the "real" conversion factor; resulting in a relative error of $(94.488 - 96)/96 \approx 1.6\%$.

While the *mm* units imply a much smaller relative error to actual millimeters than the *xmm* units, neither of them are exact. In hindsight, it would probably have been better to use the best possible integer approximation to millimeters in *jfig* from the start, instead of yet another poor scaling. However, most figures created with *jfig* end up as illustrations in standard documents, where exact scaling is of no concern.

When mixing different units in one figure, *inches* and *xmm* are the best combination, because many grid points are the same in both units, so that object alignment is possible.

changing the units

Use a left-button mouse click on the top-right corner of the drawing canvas (where the two rulers meet) to *toggle the units* used by the editor in the order *inches* ▷ *xmm* ▷ *mm*. The rulers and the grid will change to indicate the new units. You can specify the *jfig.units* property in your `.jfigrc` configuration file (see section 2.5) to select the units used by the editor at program start. Possible values recognized by the editor are: *inches*, *millimeters* or *xfig millimeters*.

Note that the editor will not automatically adjust the units when loading a figure file, because most users tend to stay with their favorite set of units anyway. However, when trying to load a figure file prepared using a different units setting, several objects in the loaded figure file will appear off-grid. In such cases, simply change the units until the objects appear on-grid again.

If you really need pixel-accurate output, keeping with *inches* is the best option, because most output devices like laser-printers are based on inches anyway with typical resolutions of 600 dpi or 1200 dpi. It is also possible to achieve an (almost) exact output from *xmm* or *mm* units using the corresponding magnification factor during printing or exporting.

2.1.3 Zooming and panning

panning

There are five different ways for *panning* (or *scrolling*) the current figure:

- *view menu*: select one of the menu items in the *menu* ▷ *view* menu to move the figure around (see figure 7). Selecting the *pan origin* menu item resets the transformation so that the origin (0,0) of the coordinate system lies at the upper left corner of the drawing canvas.
- *cursor keys*: type one of the standard cursor keys on the main drawing area to move the figure in the opposite direction; same as you would in any standard text editor. Hold down the `[shift]` key to increase the amount of scrolling from 1/20th to 1/5th of the currently visible area. For example, `[shift]+[cursor down]` key to move the figure one big step up. Note that bindkeys are disabled during the *create text* operation.
- *ruler buttons*: left-click one of the small gray triangles in the x- and y-rulers to scroll your figure, similar as you would with standard scrollbars.
- *ruler dragging*: use mouse-dragging inside the x-ruler or y-ruler to drag the figure proportionally. To avoid accidental scrolling, the editor rejects a dragging distance of less than five pixels. Still, this mode gives you the best control over panning distance.
- *mouse-wheel scrolling*: you can also use the mouse-wheel for scrolling and zooming your figure. See the description below (page 14).

zooming

Use the editor menu, bindkeys, or the extra *zoom buttons* on the left of the main editor window to change the *zoom factor* of the drawing canvas:

- *view menu*: select one of the menu items in the *menu* ▷ *view* menu to change the drawing canvas zoom (see figure 7).

  zoom-fit

  The *zoom fit* command changes the zoom-factor (and translation) so that all objects in the current figure are just visible, while the *zoom region* command allows you to interactively select the region of interest (see below).

  The *zoom in* command increases the zoom-factor by a factor of $\sqrt{2}$, while the *zoom out* decreases the current zoom by that factor. Therefore, any two zoom-in or zoom-out operations will double or halve the zoom factor.

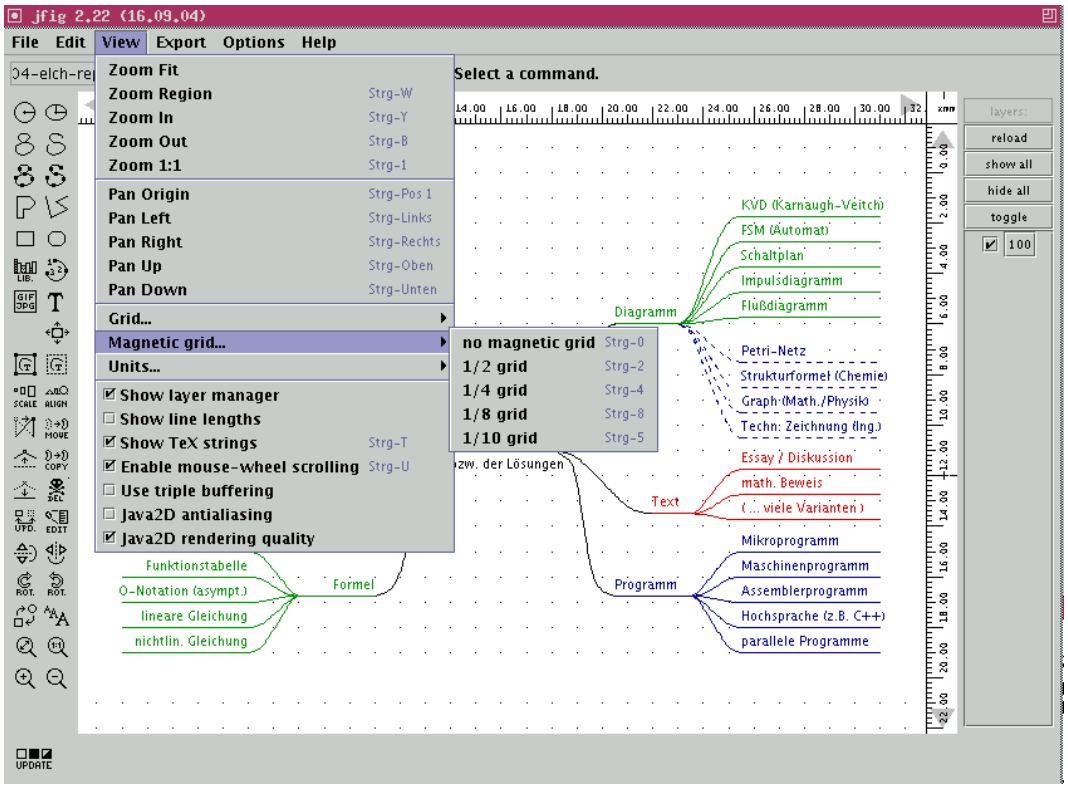

Figure 7: *The view menu includes several functions for zooming and panning and is used to control the display options of the editor like grid, magnetic grid, or Java2D rendering. See the text for details.*

Finally, the *zoom 1:1* menu item restores the zoom-factor to a value of 1.0 or 100%, so that the figure is shown at its original size (assuming a 75 dpi display resolution).

- the *zoom buttons* in the mode button panel allow quicker access to the most important zoom operations than the menu-items. There are four buttons (from top to bottom) for *zoom 1:1*, *zoom in*, *zoom out*, and *zoom region*.
- You can also type the following *bindkeys* for quick zooming; but remember that bindkeys are disabled during a *create text* operation:

  for a *zoom fit* type `[f]`.

  for a *zoom region* type `[cntl]+[w]`.

  for a *zoom out* type `[z]` or `[cntl]+[b]`.

  for a *zoom in* tpye `[shift]+[z]` or `[cntl]+[y]`.

  for a *zoom 1:1* type `[cntl]+[1]`.

The *zoom region* command expects you to interactively select the region to zoom into. After selecting the command, click the left mouse button on the first corner of the region that you want to zoom into, then click the left mouse button on the opposite corner of that region.

mouse-wheel panning

Finally, you can also use the mouse-wheel for zooming and panning, if you prefer to do so (may take some time to get used to). You also need at least Java 1.4.0 because older versions of Java didn't include mouse-wheel support. To enable mouse-wheel scrolling, set the values of the two properties `jfig.gui.Editor.EnableMouseWheelSupport` and `jfig.gui.Editor.MouseWheelPanning` to `true` in your `.jfigrc` configuration file. If enabled, you can switch mouse-wheel scrolling on and off interactively in the editor via the *menu* ▷ *view* ▷ *enable mouse-wheel scrolling* menu item.