RustFig manual

Version 0.9.0-beta.7

Contents

RustFig manual **1**
- Start here . . . 1
- Drawing . . . 1
- Editing . . . 1
- Attributes and helpers . . . 1
- Files and output . . . 1
- Where xfig's own manual went . . . 2

Getting started **2**
- Your first figure . . . 2
- Things worth knowing early . . . 2

The window **2**
- Top strip . . . 2
- Mode panel (left) . . . 2
- Canvas . . . 3
- Indicator panel (bottom) . . . 3
- Depth manager (right, optional) . . . 3
- Dialogs . . . 3

The mouse **3**
- Getting three buttons . . . 3
- What the buttons do . . . 4
- Shift . . . 4
- Ctrl . . . 4
- Markers and tags . . . 4
- Clicking and dragging . . . 4
- Wheel and trackpad . . . 4

Keys **5**
- Mode keys . . . 5
- xfig's menu keys (Meta = Option or Alt) . . . 5
- Platform keys . . . 6
- Other keys . . . 6

Menus **6**
- File . . . 6
- Edit . . . 7
- View . . . 7
- Snap . . . 7
- Object (RustFig) . . . 7
- Help . . . 7

Shapes 7
Circle (radius) 8
Circle (diameter) 8
Ellipse (radii) 8
Ellipse (diameter) 8
Polygon 8
Polyline 8
Lengths while drawing 9
Typed coordinates 9
Box 9
Arc-box (rounded) 9
Regular polygon 10
Arc 10

Splines 10
Approximated spline 10
Closed approximated spline 10
Interpolated spline 10
Closed interpolated spline 11
X-spline 11
Closed X-spline 11
Shape factor on the canvas 11
Changing a spline later 11

Text and pictures 11
Text 11
Local fonts 12
Character map 12
Spell check 13
LaTeX text, typeset on the canvas 13
Picture 13
The picture panel 13

Move, copy, delete 14
Move 14
Copy 14
Delete 15

Scale, align, flip, rotate 15
Scale 15
Align 15
Flip up/down 15
Flip left/right 16
Rotate counter-clockwise 16
Rotate clockwise 16
Array copies 16

Points 16
Move point 16
Add point 16
Delete point 17

Compounds 17
Glue into compound 17
Break compound 17
Open compound 17

Edit a compound . . . 17

Join and split **18**
Join/split lines and splines . . . 18

Chop **18**
Chop . . . 18

Trim mode **18**
Trim, split and extend . . . 18

Tangent and normal **19**
Add tangent/normal to curve . . . 19

Measuring **19**
Measure angle . . . 19
Measure length . . . 20
Measure area . . . 20

Attribute modes **20**
Update attributes . . . 21
Edit attributes . . . 21
Convert line ↔ spline . . . 21
Open/close . . . 21
Add/remove arrows . . . 22

Libraries **22**
Library mode . . . 22
Placing . . . 22
Your own libraries . . . 23
A list of library folders . . . 23
Stop . . . 23
Feynman diagrams . . . 23
Quantum circuits . . . 23
Optics and lab . . . 24
Physics circuits . . . 24
Commands . . . 24
Credits and licences . . . 24

Plotting a curve **24**
Plot . . . 25
The Plot dialog . . . 25
Edit plot . . . 26
Commands . . . 26

The indicator panel **26**
Colours . . . 28
Styles . . . 28

Line styles **28**
One number per line . . . 29
The two defaults . . . 29

Angle Geom **29**

Smart links **30**
Turning it on . . . 30

What attracts lines (targets) . . . 30
What follows (followers) . . . 30
Operations that carry the links . . . 30

Snap and grid **30**
Point Posn and Grid Mode . . . 30
Object snaps (Snap menu) . . . 30

Zoom and view **31**

Files **31**
Autosave and recovery . . . 33
Starting from the command line . . . 34
Opening from the file manager (macOS) . . . 34
The web version . . . 34

Export **34**
The Export panel . . . 34
Exporting in the background . . . 36
Notes . . . 36

Print **36**
The panel . . . 36
Scripts . . . 37

Preferences **37**
LaTeX text . . . 38
Command-line options . . . 38
Coming from xfig . . . 38
Accessibility . . . 38

Automation **38**

Finding help **39**
Command search . . . 39
F1: help for what is under the pointer . . . 39
Mouse buttons . . . 39
What's new . . . 39

RustFig manual

RustFig is a Rust reimplementation of xfig, the classic vector drawing program for figures in papers and documents. It looks and behaves like xfig where that still makes sense, and adds modern conveniences (multi-level undo, a modern Select tool, automation) where xfig's way no longer fits. This manual is part of the program: it works offline and you can search it with the box above.

Start here

- Getting started: draw your first figure in a few minutes.
- The window: menus, mode panel, canvas, rulers, indicator panel, message panel.
- The mouse: xfig uses three mouse buttons; here is how to get them on any computer, and what Shift does.
- Keys and Menus: every shortcut and menu entry.

Drawing

- Shapes: circles, ellipses, boxes, rounded boxes, regular polygons, arcs, polylines and polygons.

- Splines: approximated, interpolated and X-splines, open and closed.
- Text and pictures.

Editing

- Move, copy, delete, Scale, align, flip, rotate.
- Points: move, add and delete the points of a line.
- Compounds, Join and split, Chop, Trim, split and extend, Tangent and normal, Measuring.
- Attribute modes: update, edit, convert, open/close, arrows.
- Libraries, Plotting a curve.

Attributes and helpers

- The indicator panel, Line styles, Angle Geom, Smart links.
- Snap and grid, Zoom and view.

Files and output

- Files: open, save, merge, autosave and recovery.
- Export: PDF, SVG, EPS, PNG and many more formats.
- Print: page setup, preview, printer and batch.
- Preferences and Automation (control channel, MCP, command line).
- Finding help: command search, F1 context help, the mouse cheat sheet, what's new.

Where xfig's own manual went

xfig's Help menu opens an HTML reference, man pages and a How-To guide. In RustFig the Help menu opens this manual instead (Help > RustFig Manual). The command list (Help > RustFig Reference) shows every command with its id and shortcut. The command-line parts of xfig's man pages are in Automation.

Parts of this manual are adapted from xfig's HTML manual (xfig licence, see LICENSE-xfig).

Getting started

Your first figure

1. Pick a mode from the mode panel on the left, for example **Box** (key `B`). The message panel at the top says what to do next, and the mouse indicator shows what each mouse button does right now.
2. Press the left button on the canvas, drag, and release. The box is drawn with the current attributes shown in the indicator panel at the bottom (line width, colours, fill and so on).
3. Pick **Move** (key `M`), click the box and click again where it should go. Or use **Select** and drag it.
4. Change how things look: set an attribute in the indicator panel, then use **Update attributes** (key `U`) on an object, or double-click an object with Select to open its edit panel.
5. Press Option+U or Cmd+Z to undo. RustFig keeps many undo levels.
6. Press Alt+U or Ctrl+Z to undo. RustFig keeps many undo levels.
7. File > Save As writes a `.fig` file. File > Export writes PDF, SVG, EPS, PNG and other formats.

Things worth knowing early

- Most modes have three mouse button functions. See The mouse for how to press the middle and right buttons without a three-button mouse.
- Hold Shift to see Shift variants in the mouse indicator.
- Press Escape or the right button to cancel what you are doing in most modes. Escape again clears the selection, and once more leaves the mode. While drawing a line, Backspace removes the last point.
- Coordinates are in 1/1200 inch (Fig units); you work in inches or centimetres as the units box says.

- The figure is never lost when the program stops unexpectedly: see Files (recovery).

The window

RustFig keeps xfig's arrangement of the main window.

Top strip

- **Menus**: File, Edit, View, Snap, Object (RustFig's own, for commands xfig only has as modes or keys) and Help. See Menus.
- **File name**: the current file; it is marked when there are unsaved changes. Click it to rename the file.
- **Message panel**: what to do next, errors and information. Click it to see the history of messages.
- **Mouse indicator**: three boxes, one per mouse button, say what each does in the current mode and step. Hold Shift to see the Shift meanings. Hovering shows the same as text.
- **Units box**: shows the units (inches or centimetres, a scale); click it for the unit panel (Edit > Set units).

Mode panel (left)

One button per mode: drawing modes first, then editing modes. Hover for the name and key. The same modes are available as keys, see Keys. The mode is highlighted while it is active. On a small screen the panel scrolls vertically (mouse wheel over it); the buttons keep their size.

Canvas

The drawing area with the page (page borders can be shown or hidden, View menu) and optional grid. Rulers at the top and left show the position; the scale follows the zoom. Scroll or use the rulers' arrows to pan; Z and Shift+Z zoom (see Zoom and view).

Indicator panel (bottom)

Buttons for the attributes that new objects get and that the current mode needs: line width, line style, colours, fill, arrows, font and so on. Which buttons show depends on the mode, as in xfig. See The indicator panel.

Depth manager (right, optional)

Lists the depths in use with check boxes to show or hide them (View > Show depth manager).

Choosing a mode puts xfig's sentence for it, with its key, in the message panel (for example "Rectangular BOX drawing (b)"). After opening a file the message reads `Current figure "name" (N objects)`. RustFig's own Trim button is the last of the editing buttons, so every xfig button keeps its place in the grid.

Dialogs

A dialog taller than the window (Global settings on a small screen, for example) gets a vertical scroll bar: scroll with the mouse wheel or drag the bar to reach the rows and buttons below.

The mouse

xfig is drawn with three mouse buttons, and almost every mode gives each button a different job (left: the main action, middle: a variation, right: cancel or another variation). RustFig keeps that exactly. The mouse indicator in the top strip always shows what the three buttons do right now.

Getting three buttons

Mouse	Left	Middle	Right
Three-button mouse	left button	middle button (wheel click)	right button
Trackpad or one-button mouse	click	Option+click	Ctrl+click (or a two-finger click)

Mouse	Left	Middle	Right
Three-button mouse	left button	middle button (wheel click)	right button
Two-button mouse	left button	left and right buttons together (a chord)	right button
Keyboard substitute	click	Alt+click	(none: Ctrl+click is xfig's zoom, see Zoom and view)

The chord works like xfig on a two-button mouse: press both buttons within a moment of each other.

What the buttons do

The three functions of each mode are listed in the mode's section (shapes, splines, move, copy, delete and the other editing topics). Typical patterns:

- Drawing: left adds the next point or starts the shape, middle is a variation (freehand, from the centre) or the final point, right cancels.
- Editing: left acts on the object clicked, middle often acts on a copy or restricts a move to horizontal or vertical, right is a variation or region action.
- Regions: every region action ("tag region", "delete region", "move region", "update region", "delete in region") takes either a rectangle or a lasso. Click two corners for a rectangle, or press and drag to draw a free-hand loop (the lasso); everything inside is used.

Shift

Holding Shift changes a button's job in some modes. While Shift is held, the mouse indicator shows the Shift meanings (likewise Ctrl, and Ctrl+Shift, show those buttons' meanings):

- While drawing a polyline, polygon, spline or arc: Shift+left removes the last point ("del point"). Shift+middle (not for arcs) finishes the object at the last point placed, without adding the clicked point ("finish here", a RustFig addition).
- Box, Arc-box, Picture: Shift+middle draws a square from its centre.
- Edit attributes: Shift+middle shows the object's comments in the message panel.
- Locate object: in every mode that picks an object (Move, Copy, Delete, Edit, Flip, Rotate, Scale, Convert, Align, Break, Open compound, Join/Split, Chop, Add/Move/Delete point, Update, Arrow, Open/close, Tangent), Shift+left click highlights the object under the pointer (its markers turn filled) without acting on it. Shift+left again highlights the next object under the pointer, front to back; after the last one the pointer position itself is marked, and the next Shift+left starts over. A click without Shift then acts on the highlighted object, not on the front-most one. This is how you reach an object hidden under another one. Changing the mode forgets the highlight. Shift+middle and Shift+right do nothing in these modes (xfig locates with every button; RustFig keeps them free for later use), except where xfig gives them their own meaning: in Edit Shift+middle shows the object's comments, and in Rotate and Flip Shift+right sets the centre or anchor, like the right button.

Ctrl

Holding Ctrl (Cmd on macOS) shows xfig's zoom functions: left starts a zoom box, middle pans to the origin, right unzooms (Zoom and view). With Ctrl and Shift in the editing modes, a click on a spline point changes its shape factor (Splines).

Markers and tags

In the editing modes small hollow squares (markers) sit on every object the mode can act on, so you see what to click. Markers stay while a mode waits for its next click (Join's second end, Chop's log, Align's object). Tagged objects show filled markers: objects you tagged with Glue or selected with Select, and the parts of a compound broken with "break and tag". Only Select, Glue and Break tag objects; the other modes act on the object you click and leave the tags as they are, so drawing or editing never tags anything by itself.

Clicking and dragging

Most placing operations work either way: press, drag and release, or click once to start and click again to finish. A click without movement picks the object under the pointer; the pointer picks within a few screen pixels of the line.

Wheel and trackpad

Scrolling pans the canvas, Ctrl+scroll or pinching zooms, and holding Space while dragging pans. On a touch screen, two fingers pan and pinch zooms, and holding one finger still for half a second acts as the right mouse button.

Keys

RustFig understands two sets of keys: xfig's own, and the usual ones of macOS. xfig's set can be switched off in Edit > Global settings ("xfig keyboard shortcuts"). The macOS shortcuts use Cmd (written Ctrl in the tables below); xfig's Meta is the Option key. Where a table says "the physical Ctrl key", it means the Control key, not Cmd.

The menus show one key per entry, xfig's while xfig's keys are on (Edit > Global settings), otherwise the system's; the other one works too, and Help > Keys lists both. For example Undo shows ⌥ U, and ⌘ Z undoes as well. Option and Cmd are written ⌥ and ⌘ (with Control ^ and Shift ⇧), with a thin space before the key, so ⌘ ' (Cmd and the apostrophe key), ⌘ − and ⌘ = stay readable. RustFig understands two sets of keys: xfig's own, and the usual ones of your system. xfig's set can be switched off in Edit > Global settings ("xfig keyboard shortcuts"). xfig's Meta is the Alt key.

Where both sets want the same key (Ctrl+S, Ctrl+C and so on), xfig's meaning wins while xfig's keys are on. The menus show both keys of an entry where it has both, for example `Alt+U / Ctrl+Z`; modifiers are written Ctrl, Alt and Shift, and the key as its symbol (`Ctrl+'` is Ctrl and the apostrophe key, `Ctrl+−`, `Ctrl+=`).

Mode keys

The keys that choose a mode are shown in the mode sections: for example `C` Circle, `B` Box, `L` Polyline, `M` Move, `D` Delete. `W` Trim, split and extend (RustFig's own). Shift selects the variant: `Shift+C` Circle by diameter, `Shift+B` Arc-box. The Ctrl keys xfig uses for modes are:

Key	Mode
Ctrl+S (physical Ctrl)	Scale
Ctrl+C	Copy
Ctrl+A	Add point
Ctrl+E	Edit attributes
Ctrl+R	Rotate clockwise
Ctrl+P	Picture

Key	Mode
Ctrl+G	Measure angle
Ctrl+N	Measure length
Ctrl+M	Measure area

Mode keys do nothing while a menu is open or while you type text.

In Help > Keys and the menus xfig's key comes first; the system's key is added only where it differs (a mode letter such as C is xfig's, so it appears once). On a Mac the Meta key is shown as ⌥ and the physical Ctrl key as ^; the words Meta and Alt are not used there.

The complete list, searchable, is in Help > Keys…: every command with its key in your system's scheme and in xfig's scheme, side by side (it shows the keys of the system you are on). Both sets work.

xfig's menu keys (Meta = Option or Alt)

Key	Action
Meta+N, O, M	New, Open, Merge
Meta+S, A	Save, Save As
Meta+X	Export; Shift+Meta+X exports at once with the last settings
Meta+Q	Exit
Meta+U	Undo
Meta+T	Paste
Meta+I	Search/Replace
Meta+K	Spell check
Meta+D	Delete all
Meta+R or Ctrl+L	Redraw
Meta+C	Portrait/Landscape
Meta+B	Show page borders
Meta+Y	Show info balloons
Meta+G	Global settings
Shift+U	Set units
Shift+Z, Z	Zoom in, zoom out
Ctrl+Z (the physical Ctrl key)	Zoom to fit

Meta+F, E, V and H open the File, Edit, View and Help menus; while a menu is open, the underlined letter of an entry picks it (for example i for Zoom In in the View menu). In the Help menu x opens this manual and k the key list. In the View menu c opens the character map (also Ctrl+Shift+M, even while typing).

Platform keys

Key	Action
Ctrl+N, O, S, Shift+S, Q	New, Open, Save, Save As, Quit
Ctrl+E, Ctrl+Shift+E	Export
Ctrl+Z, Ctrl+Shift+Z	Undo, Redo
Ctrl+V	Paste objects (the cut buffer)
Ctrl+=, Ctrl+-, Ctrl+0, Ctrl+1	Zoom in, out, to fit, actual size
Ctrl+' , Ctrl+B	Show grid, show page border
Ctrl+F	Search/Replace

Other keys

- Arrow keys pan the canvas; Home returns to the origin.
- Escape works one level per press: it closes an open panel; otherwise it (1) cancels what is in progress (a shape being drawn is dropped, a move, paste or zoom box stops; typed text is finished), then (2) clears what the mode has collected (Chop's axe list, the located object, the rotate or flip centre, the selection), then (3) leaves the mode, as when RustFig starts. Edit > Select none still selects nothing.
- Backspace (or Delete) while drawing a line, polygon or spline removes the last point, like Shift+left.
- In Text mode: Left, Right, Home, End, Backspace and Delete edit, Enter ends the line and starts the next one below it. Shift extends the selection; word jumps and select all, copy, cut and paste follow the platform: Option+arrows (words), Cmd+arrows (start, end), Cmd+A, Cmd+C, Cmd+X, Cmd+V. Ctrl+arrows (words), Ctrl+A, Ctrl+C, Ctrl+X, Ctrl+V.
- Enter confirms and Escape cancels in most panels.

Menus

Entries xfig has are named as in xfig; RustFig's own entries follow them. An entry greyed out is not available in the current situation (nothing selected, nothing to undo).

File

New, Open, Merge, Save, Save As, Export, Import picture (RustFig), Exit, then the recent files (numbered; pick with the number key). See Files and Export.

Edit

Undo, Paste objects, Search/Replace, Spell check (native app only), Delete all, Global settings, Colours, Set units, Document settings (RustFig), Enter point, then Redo and Copy as picture. xfig has no selection, so there is no Cut, Copy or Select all: the modes act on the object you click, and the cut buffer is filled by Copy mode or Delete mode (right button).

- Undo and Redo have many levels.
- Paste objects attaches the cut buffer to the pointer as an outline, its top-left corner at the pointer. Left places the objects there ("place object"), middle at the position they were cut or copied from ("place at orig posn"), right or Escape cancels ("cancel paste"). One placement is one undo step. If the system clipboard holds a figure copied in another RustFig window, that is pasted instead of the cut buffer; if it holds an image (PNG, for example from a browser or a screenshot), it is pasted as a picture object, saved as `clipboard-<number>.png` next to the figure (in the temporary folder while the figure has no file name) and referenced by that path like any picture in a .fig file. The web version has only the cut buffer. Scripts and MCP give `dx` and `dy` to `edit.paste` to place the objects at once, moved by that offset.
- Search/Replace finds text objects by their text and can replace it.
- Copy as picture (RustFig's addition) puts a picture on the system clipboard, for pasting into other programs: after choosing it, click what to copy — left an object, middle a region (two corners or a loop), right the whole figure; Escape cancels. The mouse indicator shows the three buttons meanwhile.

View

Redraw, Portrait/Landscape, Zoom in, Zoom out, Zoom to fit canvas, Unzoom, Pan to origin, then the switches: Show page borders, Show depth manager, Show info balloons, Show grid, All depths on. See Zoom and view.

Snap

Hold and Release, then the object snaps for the next clicked point: Endpoint, Midpoint, Nearest, Focus, Normal, Tangent, Intersection, Diameter. See Snap and grid.

Object (RustFig)

Commands that xfig has only as modes or keys: Group and Ungroup, Update, Convert, Open/Close, Arrows, Transform (rotate, flip), Arrange (align, distribute, depth order), Measure and Edit Attributes. They act on the selection.

Help

- RustFig Manual: this manual (xfig's "Xfig Reference").
- RustFig Reference (commands): every command with its id and shortcut, filterable. The same ids work in the control channel and MCP.
- Install TeX Gyre fonts: downloads the TeX Gyre fonts from CTAN into RustFig's user data folder (native app; see the font chooser).
- About RustFig: version and licence; the xfig copyright holders.

Shapes

Drawing modes make new objects with the attributes shown in the indicator panel (colours, line width, fill, depth and so on). Change those first, or change the object afterwards with Update. Coordinates snap to the grid and to object snaps (Snap and grid).

In every shape mode you can press, drag and release, or click once and click again to finish. The right button cancels a shape in progress. The mouse indicator reads, for each mode, the words quoted below.

Circle (radius)

Key `C`. Draws a circle from its centre.

- Left, "Circle center": press at the centre, drag out to the radius and release (while dragging the indicator says "set radius").
- Middle, "Ellipse center": the same, but draws an ellipse whose radii follow the pointer's horizontal and vertical distance.
- Right, "circle by 3 points": click three points on the circle.

Circle (diameter)

Key `Shift+C`. Draws a circle from one end of its diameter.

- Left, "Circle diameter": press at one end, drag to the opposite end.
- Middle, "Ellipse corner": draws an ellipse inside the box spanned by the two corners.
- Right, "circle by 3 points": click three points on the circle.

Ellipse (radii)

Key `E`. Same buttons as Circle (radius), but the ellipse is the main shape.

- Left, "Circle center": a circle from its centre.
- Middle, "Ellipse center": press at the centre; the radii follow the pointer. A left or middle click finishes it ("Ellipse corner"), as in xfig, or release the middle button after dragging.

Ellipse (diameter)

Key `Shift+E`.

- Left, "Circle diameter": a circle by its diameter.
- Middle, "Ellipse corner": press at one corner and drag to the opposite corner of the ellipse's box.

Polygon

Key P. A closed shape with straight sides.

- Left, “first point”: click the first point, then each next point. Shift+left removes the last point; Shift+middle finishes at the last point placed without adding the clicked one.
- Middle, “freehand”: start a freehand path; move the pointer and the path follows. Middle again ends it (“final point”). When drawing with straight sides, middle at a point is the final point.
- Right: cancels while drawing.

As in xfig, the middle button does nothing until only the last point is missing, so a polygon always has at least three points; Enter or a double click with fewer keeps you drawing. Shift+left never removes the first point (with one point it adds a point). The polygon closes itself when you finish. Angle Geom can constrain the points (Angle Geom).

Polyline

Key L. An open line of straight segments (the same buttons as Polygon).

- Left, “first point”, then “next point” for each further point.
- Middle, “freehand” to start a freehand line; “final point” ends the line.
- Right, “single point”: before a line is started, enters a single dot; while drawing, cancels.
- Shift+left, “del point”, or Backspace: removes the last point. Escape drops the whole line. Shift+middle, “finish here”: ends the line at the last point placed, without adding the clicked point.
- Shift+left before the first point, “dimension line”: starts a dimension line (xfig’s trigger, `d_line.c`). Click its second point (“final point”; Shift+left removes the first). The result is a compound of the line with arrows at both ends, a box in the middle with the length in the figure’s units and scale, and a tick at each end; the compound’s comment reads “Dimension line: 2.0 in”. The settings are the indicator’s `dimline` group, edited in the “Dimension line” button’s window (shown in Polyline mode; xfig’s “Dimension Line Settings”: line thickness, style and colour; box thickness and colour; the two arrow types, length and width; ticks and their thickness; text font, size and colour; actual length or fixed text; decimals; OK stores, Cancel drops) — (thickness, arrows, ticks, box and text colours, font, decimals, fixed text). When the line is scaled, or an open dimension line is closed again after its main line was edited, the length, box and ticks are recomputed (`object.recompute_dimension`). Command `shape.dimension_line`.

The length in the dimension line follows xfig: in inches it is a whole number with a fraction (for example 2 1/4), otherwise a decimal. The Dimension line button’s settings window shows a live example line (“Dimension line example”) that follows the font, arrow types (pictures as in the arrow menu, or “none” per end) and the other settings. The example draws plain triangle arrowheads.

Lengths while drawing

While a line, box, circle or ellipse is being drawn, or an object moved or copied, the message panel shows the length, dx, dy and angle of the segment from the fixed point, for example “2 points, Length = 1.0 in, dx = 1.0 in, dy = 0.0 in (0.0 deg)”. While a box or rounded box is drawn the message reads “Width = …, Height = …, Factor = …”, and a regular polygon “Length = …, dx = …, dy = … (angle) deg”. With Show Lengths on (command `view.toggle_lengths`, resource `showlengths`) the triangle of dx and dy is drawn at the segment and its two sides are labelled in red.

Other messages follow xfig as well. Moving or adding a point of a line or spline shows both neighbour lengths and offsets (“Len 1 = …, Len 2 = …, dx1 = …”); moving the middle point of an arc adds its radius (“Rad = …”); scaling a box or a compound shows “Width = …, Length = …, Factor = …”. **Show crosshair** (View menu, Edit > Global settings, command `view.toggle_crosshair`, resource `crosshair`, option `–crosshair`) draws full-window red lines through the pointer while a shape is being drawn or an object moved, scaled or reshaped.

Typed coordinates

Command `tool.point` (xfig’s keyboard point entry) gives the next point as text instead of a click: `1.5, 2` (ruler units; fractions as `1–1/2`), `r1, r–0.5` or `r1 0.5` (relative to the current point: the last point of the line, or where

the operation started), `a` for absolute, or polar `2<30` (length and angle in degrees, `2<0.5p` in multiples of pi, `2<1r` in radians; from the current point). The point is used exactly, without snapping.

In the app, **Edit > Enter point...** (Ctrl+K; physical Ctrl+K with xfig's keys) opens a small window with a text field: type a point and press Enter for the left button (the Left, Middle and Right buttons act as that mouse button), and the point goes to the current mode. A bad entry shows its error and stays in the field; the last ten entries are listed under History, click one to type it again.

Box

Key `B`. A rectangle.

- Left, "corner point": press at one corner, drag to the opposite corner.
- Middle, "center point": press at the centre; the box grows around it. Shift+middle makes a square.

Arc-box (rounded)

Key `Shift+B`. A rectangle with rounded corners; the radius is the indicator's Box Curve value. Buttons as for Box.

- Left, "corner point".
- Middle, "center point" (Shift+middle: square).

Regular polygon

Key `Shift+P`. A polygon with equal sides; the number of sides is the indicator's Num Sides.

- Left, "center point": press at the centre, drag to a corner.
- Middle, "one side": press at one end of a side and drag to the other end.

Arc

Key `R`. A circular arc. The indicator's Arc Type chooses open or pie-wedge, and Arrow Mode adds arrowheads.

- Left, "first point": click three points on the arc (first, middle, final).
- Middle, "center point": first the centre, then the start point, a direction point and the end angle ("first point", "direction", "final angle" in the indicator).
- Right: cancels.

Splines

A spline is a smooth curve through or near a series of control points. Fig files have three kinds, each open or closed; all six modes work the same way and differ in the curve they make.

- **Approximated splines** pass near the points (the points pull the curve). Modes **Approximated spline** and **Closed approximated spline**.
- **Interpolated splines** pass through every point. Modes **Interpolated spline** and **Closed interpolated spline**.
- **X-splines** let you choose per point how sharp or smooth the curve is (see below). Modes **X-spline** and **Closed X-spline**. This is the most flexible kind.

Buttons in all six modes: left adds points, middle draws freehand or finishes, right cancels. A closed spline joins its last point to the first. A closed spline needs three points (the middle button is dead until the last one is missing); an open spline needs two, otherwise the message panel says "Not enough points for spline" and you keep drawing. Shift+left removes the last point but never the first; Shift+middle finishes at the last point placed without adding the clicked one. Angle Geom can constrain the points.

Approximated spline

Key `S`.

- Left, “first point”, then each next point.
- Middle, “freehand” to start freehand drawing; middle again, “final point”, finishes.
- Right: cancels.

Closed approximated spline

Key `Shift+S`. As Approximated spline, but the curve closes on itself.

- Left, “first point”.
- Middle, “freehand”.

Interpolated spline

Key `I`. The curve passes through each point you click.

- Left, “first point”.
- Middle, “freehand”.

Closed interpolated spline

Key `Shift+I`.

- Left, “first point”.
- Middle, “freehand”.

X-spline

No key. The curve’s shape at each control point is a number between -1 and 1: 0 makes a sharp corner, 1 a smooth curve that approximates the point, -1 a smooth curve through it. New points are smooth approximations; change them later with Edit attributes (right button on a point of the spline).

- Left, “first point”.
- Middle, “freehand”.

Closed X-spline

No key. As X-spline but closed.

- Left, “first point”.
- Middle, “freehand”.

Shape factor on the canvas

In the editing modes (not the drawing modes), hold Ctrl and Shift and click a spline point: the left button makes the point more approximating (shape +0.04 up to 1), the right button more interpolating (-0.04 down to -1), the middle button cycles through interpolating (-1), angular (0) and approximating (1). The spline becomes an X-spline. The end points of an open spline are refused (“Cannot edit end-points”); a click away from spline points says “Only spline points can be edited”. The indicator shows “More approx / Cycle shapes / More interp” while both keys are held. Undo restores the old shape.

On macOS use Cmd+Shift+click: Ctrl+click is the right button there. On Linux and Windows the keys are Ctrl+Shift.

Changing a spline later

Use Points to move, add and delete control points, and Convert line and spline to turn a polyline into a spline and back.

Text and pictures

Text

Key `T`. Text objects hold one line each.

- Left, "position cursor": click where the text should start, then type. Clicking an existing text edits it. Clicking elsewhere finishes the text and starts a new one there.
- Middle, finish text: ends the text being typed. Shift+middle, "paste text" (as in xfig): pastes the system clipboard's text at the cursor of the text being typed. When no text is being typed and the clipboard holds text, middle (shown "paste as text") pastes it as a new text object at the pointer.
- Right: cancels the text being typed; an edited text keeps its old string.
- Enter finishes the line and starts the next one below it, spaced by the indicator's Text Step. Left, Right, Home, End, Backspace and Delete move and edit.
- xfig's Control keys edit the line (`d_text.c`; on macOS the Control key itself, on Linux and Windows those that do not clash with Select all, Cut and Paste): Ctrl-A start of the text (macOS only), Ctrl-E end, Ctrl-B back, Ctrl-F forward, Ctrl-D delete the character at the caret, Ctrl-H backspace, Ctrl-K delete to the end of the text, Ctrl-X delete from the start to the caret (macOS only; Cut elsewhere), Ctrl-W delete the selection.
- **Superscripts and subscripts**: Ctrl-`^` (Ctrl+Shift+6) ends the text so far and starts the next piece at the caret, a quarter of a line higher and at three quarters of the size; Ctrl-_ (Ctrl+Shift+-) does the same lower. The opposite key steps back out again, up to four levels deep; Enter starts the next line below the first piece at the original size. Each piece is a text object of its own, as in xfig. The keys also run through the control channel and `tool.event` as `superscript`, `subscript`, `kill_to_end`, `kill_to_start` and `kill_selection`.
- Clicking in existing text puts the caret at the clicked character (in the text being typed too). Drag to select; double-click selects a word, triple-click the line; Shift+click extends the selection. Shift+Left, Right, Home and End select with the keys.
- Dead keys and input methods work: on macOS Option+E then E types é, on Linux the compose or dead keys do the same; the pending accent is shown at the caret and replaced by the finished character.
- Option+Left and Option+Right jump by words, Option+Backspace and Option+Delete delete a word; Cmd+Left and Cmd+Right go to the start and end; Cmd+A selects all, Cmd+C, Cmd+X and Cmd+V copy, cut and paste through the system clipboard. (A lone tap of Option is still xfig's compose key.)
- Ctrl+Left and Ctrl+Right jump by words, Ctrl+Backspace and Ctrl+Delete delete a word; Ctrl+A selects all, Ctrl+C, Ctrl+X and Ctrl+V copy, cut and paste through the system clipboard.
- Typing or pasting replaces the selection. Any Unicode text works, including input methods (the composition shows underlined at the caret until you commit it); pasted line breaks are dropped.

Local fonts

The font chooser (the Text Font button of the indicator, and the Font row of the edit panel) has a third choice beside PostScript fonts and LaTeX fonts: **Local fonts**, every font installed on your computer that has the Latin letters. Hidden macOS system fonts (names starting with a dot, such as `.SF NS`) are left out, and so are fonts without Latin letters (Arabic, Hebrew, Indic and other scripts, such as Waseem): RustFig lays text out left to right without joining letters, so text in those scripts would not come out right. Each name is drawn in its own font (only the rows you can see are loaded, so a long list stays fast), the search field narrows the list, and the preview line shows the font at the chosen size. The list is empty in the web build and when RustFig runs with the built-in fonts only.

A Fig file knows fonts only as numbers, so a text in a local font also keeps the nearest Fig font as its number (serif: Times, sans-serif: Helvetica, monospace: Courier, with the weight and slant matched) and the font's name in a comment line of the text, `# rustfig-font: Minion Pro Semibold`. xfig and fig2dev read the file as an ordinary Fig text and ignore the comment; your own comments on the text stay. RustFig draws and measures the text in the

installed font, so selecting, editing and the text's box follow it. When a file names a font this computer does not have, the text is drawn in its Fig font and one message on opening lists the missing fonts; saving the file keeps the font name.

Exports: PDF, EPS and PostScript embed the local font (a subset of the glyphs used, as a TrueType/Type 42 font), PNG and the other bitmaps draw it, and SVG names it with its Fig font as the fallback (the viewer must have the font installed). The TeX languages (LaTeX, PSTricks, TikZ, pict2e and the others) and xfig or the C fig2dev reading the file use the Fig font. Scripts use `text.set_local_font` (a font name or `null`, for the selection or `ids`; it also sets the font for new text), `text.local_fonts` (the list) and the `local_font` parameter of `object.set_attributes`.

Character map

View > Character map… (Ctrl+Shift+M, also while typing) opens a window with the characters of the font of the text being typed. Pick a block (Latin-1, Latin Extended, Greek, punctuation, arrows, mathematical operators and more) or type a character or its code (for example `2211`) in the search field; a click inserts the character at the caret. Only characters the current font can draw are listed (its own face or the fallback face; none shows as an empty box), and each button is drawn in that font (PostScript or LaTeX font as set for the text). Characters are greyed out until you are typing in Text mode.

With the PostScript Symbol font (Greek letters, maths: ∑ ∫ ≤ and so on) or ZapfDingbats the window shows that font's own table. Fig stores such texts as bytes in the font's encoding (∑ is code 0xE5 in Symbol), and RustFig stores exactly that, so the PostScript, PDF, SVG and the other exports draw the right glyph. Typing a character the font does not have is refused. Scripts use the `text.insert_char` command.

The indicator panel sets the font, size, justification, angle and flags (hidden, rigid, TeX, LaTeX/PostScript fonts) for new text. A text with the TeX flag holds LaTeX (see *LaTeX text, typeset on the canvas* below). Fonts: the PostScript and LaTeX fonts of xfig map to fonts shipped or substituted by RustFig; the picture on the canvas follows fig2dev's font metrics.

Spell check

Edit > Spell check writes the text of every text object to a temporary file and runs the spell checker of Global settings (`aspell list < %f` by default; if `aspell` is missing, `hunspell -l %f`; `%f` stands for the file). The words it prints open a "Misspelled words" window with a message line ("3 misspelled words found"). Click a word to put it in the **Correction** field, edit it and press **Correct** (or Return): the word is replaced in every text, as Search/Replace would, and leaves the list. **Ignore** drops a word from the list without changing anything, **Recheck** runs the checker again, **Dismiss** (or Escape) closes the window. At most 200 words are listed, and the checker is stopped after 30 seconds. If no spell checker is installed the message says so; set the command in Global settings. Spell check is not available in the web version. Scripts and MCP use `text.spell_check`.

LaTeX text, typeset on the canvas

Set the **TeX** flag (the Fig format calls it the *special* flag, as do xfig's manual and its `-specialtext` option) on a text to write LaTeX in it, for example `$\int_0^1 x^2\,dx$`. With a TeX installation (`pdflatex` or `lualatex`, found on the `PATH` or named in Edit > Global settings) the canvas shows it typeset, as it will appear in the document. View > Typeset LaTeX text turns this off. Every text gets `\usepackage{amsmath,amssymb,color,graphicx}`; the **Preamble** field under *LaTeX text* in Edit > Global settings adds more (for example `\usepackage{siunitx}` or your own `\newcommands`), used when you leave the field (commands `view.set_latex_program`, `view.set_latex_preamble`). The picture keeps a little room around TeX's box, so ink that reaches outside the box (the integral sign below the baseline, italic overhangs) is not cut off; clicking and the selection markers use TeX's box. TeX runs in the background, so the source (for example `$\alpha^2$`) shows until the picture is ready, and stays when TeX is missing, off, or reports an error (the message panel names the error). Objects you place, move or copy show TeX's picture while they follow the mouse; a text you are typing shows its source. The picture is drawn at the text's anchor, justification and angle; the picture is vector, so it stays sharp at every zoom, and clicking and the selection markers follow its size. Editing still works on the source. The file and the exports are not affected. Without TeX, or in the web build, the source shows as in xfig. The LaTeX-type exports

(LaTeX, PSTeX/PDFTeX overlay, pict2e, TikZ, epic, eepic) write it unchanged, without escaping \, $, ^ or _, so LaTeX typesets it; the flag is stored in the `.fig` file.

Picture

No key (xfig key Ctrl+P). Places an image file in a box.

- Left, "corner point": drag a box for the picture; then a file dialog asks for the image. The picture is fitted to the box.
- Middle, "center point": draw the box from its centre (Shift+middle: a square).
- Right, "keep aspect": draw the box with the right button (click and click again, or drag); the picture then keeps its proportions.

In the web version the file dialog is the browser's picker and the picture lives in memory under its file name until the page is closed (see Files); scripts can add one with `picture.store`.

The file name is stored as typed. A relative name is read from the figure's folder, and **Save As** into another folder rewrites relative names so the pictures are still found (Files).

The picture panel

After the box is drawn and the file chosen, xfig's Edit panel for a picture opens; Edit mode (or Edit attributes) on a picture opens the same panel. Its rows:

- **Picture filename**: edit the name and press Reread to load that file (the box keeps its place). **Browse** picks a file. When the file is missing, a red note says so and Browse becomes **Relink**: pick the file's new place. You can also drop a file from the file manager onto the panel. Done also takes a changed name.
- **Type**, **Size (pixels)**, **Colours** (counted up to 256) and **Transparent colour** (yes when the picture has transparent pixels). A small **preview** shows the picture.
- **Width** and **Length** resize the box; **Rotation** (0, 90, 180, 270) turns the picture by changing the order of the box corners, as xfig does; **Flipped about diagonal** mirrors it. **h/w ratio** shows the picture's own ratio next to the box's.
- **Shrink to orig** and **Expand to orig** fit the box to the picture's proportions (shrink the longer side, expand the shorter one); **Use original size** gives the box the picture's size at 80 pixels per inch; **Scale by %** sets it to that percentage of the original size. These change the opposite corner in the panel and on the canvas at once.
- **Edit Image** opens the file in the system's default application (desktop only; an error appears in the message panel if none starts). Reread afterwards to see the change.
- **Import as objects** (PDF pictures only, found by the file's content; an EPS picture cannot be imported, and the button stays grey) replaces the picture by the objects of the PDF's first page, as one compound that fills the picture's box (flipped pictures are transposed as the canvas draws them) at the picture's depth. It uses the same importer as File > Open of a PDF (see Files), so text becomes text and lines become lines; whatever the importer drops or approximates is listed in the message panel. The panel's changes are kept first (one undo step), the import is a second one. When the PDF carries an embedded RustFig figure, the page's vectors are imported anyway.
- **XBM pen colour** appears for XBM pictures.

Reread and Relink are undone by Cancel. The same actions are the commands `picture.resize`, `picture.rotate`, `picture.set_file`, `picture.info`, `picture.edit_image` and `picture.import_objects` (plan G6).

File > Import picture does the same without drawing a box first: the picture goes to the centre of the view at its own size. Pictures can be EPS, PDF, PNG, JPEG, GIF, PCX, PPM, TIFF and others; EPS needs ghostscript (optional) to show on the canvas. The picture is referenced by its file name in the figure; keep it next to the figure file.

Move, copy, delete

xfig has no selection: you choose a mode first (Move, Copy, Delete, …) and then click the object it acts on. RustFig works the same way. To act on an object hidden under another one, Shift+left locates it first (Mouse). The cut

buffer is filled by Copy mode's right button ("copy to cut buf") or Delete mode's right button ("del to cut buf"), and emptied with Edit > Paste objects.

Move

Key `M`. Moves one object, or a compound as a whole.

- Left, "move object": press on an object, drag, release; or click it and click again at the new place.
- Middle, "horiz/vert move": the same, but the move is restricted to horizontal or vertical, whichever is greater.
- Right, "move region": mark a region with two clicks or a drag; every object completely inside moves, placed with the next click.

With Smart links on, lines attached to a moved shape follow. Grid and object snaps apply.

Copy

No key (xfig key Ctrl+C). As Move, but leaves the original.

- Left, "copy object": press, drag, release (or click, click).
- Middle, "horiz/vert copy".
- Right, "copy to cut buf": puts the object in the cut buffer and on the system clipboard (as a figure other RustFig windows can paste, and as a picture for other programs); Edit > Paste objects attaches it to the pointer later (Edit menu).
- While the copy is being placed the buttons read "place object" (left, one copy) and "array placement" (middle): with Num X Copies and Num Y Copies set (indicator fields `num_x_copies` and `num_y_copies`, 0 to 99), the middle button places a grid of copies spaced by the distance you moved, along x and y. With only one of them set to 1 you get a single extra copy that way; with both at 1, one along x and one along y; with larger numbers a columns-by-rows grid without the original's cell (command `edit.copy_array`).

Delete

Key `D`.

- Left, "delete object": click an object.
- Middle, "delete region": mark a region; everything completely inside is deleted.
- Right, "del to cut buf": deletes the object and keeps it on the clipboard (a cut).

Undo brings objects back.

Scale, align, flip, rotate

All of these act on the object you click; a compound is one object. With Smart links on, attached line ends follow.

Scale

No key (xfig key Ctrl+S). Resizes an object in proportion.

- Left, "scale box": press on the object, drag, release. The click picks the nearest of the eight handles of the bounding box. At a corner the fixed point is the opposite corner and the object grows in proportion: the factor is the larger of the width and height ratios of the drag (xfig's `adjust_box_pos`), so the corner may end up away from the pointer on the shorter side. At the middle of an edge only the size across that edge changes (the width for the left and right edges, the height for the top and bottom edges), about the opposite edge (xfig's one-dimension box scale).
- Middle, "scale about center": the same around the centre of the object, with the factor from the point you pressed on (a vertex when the press is near one) to the pointer, again the larger of the two axis ratios.
- Right: cancels.

Align

Key `A`. Aligns the parts of a compound with each other, or one object with another. The indicator's Vert Align and Horiz Align choose how the edges or centres line up.

The seven choices (each of the two indicators): **none**; **start**, **centre**, **end** (left, centre, right or top, centre, bottom: the edges or centres line up with those of the whole group); **distribute centres** (the centres of the objects are evenly spaced, the outer two stay); **distribute edges** (equal gaps between the objects, the outer two stay); **abut** (the objects are put edge to edge, starting from the first). Horizontal and vertical are independent. Command `arrange.align` (`h`, `v`: none, left, center, right, distribute_centres, distribute_edges, abut).

- Left, "align compound": click a compound; its members are aligned with each other.
- Middle, "align canvas": every object in the figure is aligned to the page (the paper size, turned for landscape) by the indicator's Vert Align and Horiz Align (command `arrange.align_canvas`). Distribute and abut do not apply to the canvas.
- Right, "align to first": click the object that stays, then click each object to align to it.

Flip up/down

Key `F`. Mirrors an object about a horizontal line through the clicked point.

- Left, "flip": flips the object. As in xfig, the points are mirrored in place (the first point stays the first, a box is not renumbered from its top left corner) and the changed object is put last in the file, so it draws on top of the others of its depth. Rotating works the same way.
- Middle, "copy & flip": leaves the original and adds the flipped copy.
- Right, "set anchor": marks a point; flips then mirror about the anchor instead of the clicked point. Right again, "unset anchor", removes it. Choosing a mode also removes it.

Flip left/right

Key `Shift+F`. As above but mirrors about a vertical line.

- Left, "flip".
- Middle, "copy & flip".
- Right, "set anchor" / "unset anchor", as above.

Rotate counter-clockwise

Key `Shift+R`. Rotates the clicked object by the indicator's Rotn Angle (set in the indicator panel).

- Left, "rotate object".
- Middle, "copy & rotate": rotates a copy. The indicator's Num Copies (`num_copies`, 1 to 99) makes that many copies, the first turned by the angle, the second by twice the angle and so on, all about the same centre. The Num Copies button appears in the rotate modes (xfig's `I_ROTATE`): click it for a number field, right button increases, middle button decreases.
- Right, "set center": marks the point to rotate about; right again, "unset center", removes it. Without a centre the object turns about the point you clicked, as in xfig.
- Shift+left locates an object (Mouse); Shift+right sets the centre like the right button, as in xfig. To turn the other way use Rotate clockwise.

Rotate clockwise

No key (xfig key Ctrl+R). As above, the other way.

- Left, "rotate object".
- Middle, "copy & rotate" (Num Copies as above).
- Right, "set center" / "unset center".

Array copies

In Copy mode the indicator shows **Num X Copies** and **Num Y Copies** (0 to 99; right button increases, middle decreases). With the middle button ("copy array") the copy is placed in that many columns and rows, spaced by the distance between the two clicked points.

Points

These modes edit the points of lines, polygons and splines (a box, arc-box or picture has four corner points).

Move point

Key `Shift+M`.

- Left, "move point": press on a point, drag, release (or click, click). The line follows.
- Middle, "horiz/vert move": the point moves only horizontally or vertically from where it was.
- Right: cancels.

The ends of an open line can be constrained with Angle Geom. Moving a corner of a box stretches the lines attached to its edges (Smart links).

Add point

No key (xfig key Ctrl+A). Inserts a point into a line.

- Left, "break/add here": click on a segment; the new point follows the pointer, with the line drawn through it, until you click again ("place new point") where it should be. Pressing, dragging and releasing does the same in one move. Right cancels.
- Middle, "add at end" (RustFig's addition): click near an end of an open line or spline; the new end point follows the pointer, and the next middle click ("place new point") places it beyond that end — an action is finished with the button that started it. A drag works too. xfig adds at an end with the left button on an end point; in RustFig that is the middle button's job.
- Right: cancels.

Delete point

Key `Shift+D`.

- Left, "delete point": click a point of a line or spline.
- Middle, "delete in region": mark a region; every point of lines and splines inside is deleted.

If the region holds no point, the message panel says so.

Compounds

A compound groups objects so that they act as one: move, copy, rotate and scale treat it as a single object. Compounds can contain compounds.

Glue into compound

Key `G`. Collects objects into a compound.

- Left, "tag object": click an object to tag it (it shows as selected); click it again to untag. Dragging on empty canvas tags everything inside the rectangle.
- Middle, "tag region": click two corners or draw a loop; everything inside is tagged.
- Right, "compound tagged": with two or more objects tagged, makes the compound (the hint shows from the start, as in xfig). While a region is being marked, cancels.

- Shift+left, “locate object”: picks an object under another one; the next left click tags the located object (Mouse).

Break compound

Key `Shift+G`. Takes a compound apart.

- Left, “break compound”: click a compound; its parts become separate objects.
- Middle, “break and tag”: the same, and the parts stay tagged (selected) for a following Glue.
- Right, “break all levels”: breaks the compound and every compound inside it.

Open compound

Key `O`. Edits the inside of a compound without breaking it. After opening, every mode works on the objects inside; the rest of the drawing is gray and cannot be touched.

- Left, “open compound”: click a compound.
- Middle, “open, keep visible”: the same, and the rest of the drawing stays visible in gray as a reference.

A “Close Compound” button appears while one is open; opening several in turn goes deeper. Saving asks whether to save only the visible part or everything. Undo steps through opening and closing too.

Edit a compound

The edit panel of a compound (Edit mode, or Edit attributes, Ctrl+E) has, as in xfig:

- Top left corner and Bottom right corner of the compound’s bounds. Changing a corner moves the compound; its size stays.
- Dimensions (width and height). Changing them rescales the compound about its top left corner; the bottom right corner follows.
- Depths: the smallest and largest depth inside, and the Collapse Depths button, which gives every object inside (also in compounds inside it) the smallest depth. Without that button the depths stay as they are.
- Number of objects, and one text field for each text directly inside the compound.

Changes show on the canvas as you make them; Done is one undo step, Cancel restores the compound. The corners are the compound’s bounds as the export gives them, so they include the width of the lines and the text.

Join and split

Join/split lines and splines

Key `J`. Joins two lines (or two splines) into one, or splits one in two.

- Left, “Join Lines/Splines”: click the end point of the first line; the indicator then reads “Choose next Line” (“Choose next Spline”); click the end of the second. The two become one, the ends meeting at the joined points. The lines may be joined in any of the four ways their ends can meet. Clicking both ends of one object closes it into a polygon or closed spline.
- Middle, “Split Line/Spline”: click a point or segment of a line or spline; it is cut there into two pieces. A polygon, box or closed spline is opened at the cut instead.
- Right: cancels a join in progress.

Arrowheads at the join or cut are removed. The first end picked is marked with a magenta marker. Only lines (or only splines) can be picked in the first step; to cut a line at an arbitrary point without a second object use this mode or Chop.

Chop

Chop cuts objects where other objects cross them, like a knife along a ruler. xfig has the basic mode; RustFig adds trimming, splitting and extending as commands. The indicator panel shows Arc Type: a chopped arc becomes open or pie-wedge pieces.

Chop

Key `x`. xfig's chop: you choose **axe** objects (the knives) and a **log** object (what gets cut). The log is cut at every point where it crosses an axe, and the pieces become separate objects with the log's attributes.

- Left, "Select axe object": click an object to make it an axe; click it again to take it out. Any number of axes can be chosen. Axes are marked in orange (filled markers and a dashed box). Once there is an axe, the object under the pointer that the middle button would cut is marked in green.
- Middle, "Select log object": click the object to cut; it is chopped by every axe at once.
- Right, "Clear axe list": forgets all axes.

Lines, polygons, boxes, rounded boxes, arcs, circles, ellipses, splines and compounds can be logs (a compound chops every member). Lines, polygons, boxes, arcs, circles, ellipses, splines and compounds can be axes. A closed shape needs two crossings to be cut. Circles chop into pie wedges, ellipses into pieces drawn as X-splines, since the Fig format has no elliptical arcs; arrowheads stay only at the original ends. One chop is one undo step.

Trimming, splitting a line at a point and extending a line have a mode of their own: Trim, split and extend.

Trim mode

Trim, split and extend

RustFig's own mode (xfig has none), the button next to Chop in the editing column, key `w`, also in the Tools menu. It acts on the object you click, using every other object in the drawing as the cutting edges. The indicator panel shows Point Posn (the split point snaps to the grid as set there) and Arc Type (a cut arc becomes open or pie pieces).

- Left, "trim piece": removes the piece you click, the part of the object between its crossings with the other objects. The rest stays as separate pieces. If the object has no piece between two crossings, nothing happens and the message line says so.
- Middle, "split at point": cuts a line, polygon, arc or spline in two at the point you click; a closed figure is opened there.
- Right, "extend nearest end": lengthens the end of the line nearest to the click to the nearest boundary, the first other object it meets.

Lines, polygons, boxes, arcs, circles, ellipses, splines and compounds can be clicked; they show markers while the mode is active. Shift+click locates an object, as in every picking mode (Mouse). Each action is one undo step.

The icon shows the idea: a solid line that ends at a cutting edge and goes on as a row of dots, the extension.

The same operations are commands (`tool.trim`, `object.trim`, `object.split_at`, `object.extend`; `object.cut_all` cuts all selected objects against each other), so they can be scripted (Automation). To cut an object at every crossing with chosen axe objects use Chop.

Tangent and normal

Add tangent/normal to curve

Key `n`. Draws a short line touching a curve (a tangent) or crossing it at a right angle (a normal) at the point you click. Useful for marking slopes and construction lines.

- Left, "add tangent": click a point on the curve; a line along the tangent is added, centred on that point.

- Middle, “add normal”: the same with the line perpendicular to the curve.
- Right: nothing.

The curve can be a line, polygon, box, spline, arc, circle or ellipse (also a turned ellipse). The new line is an ordinary three-point polyline in the current line attributes, depth and arrows, so you can change or delete it like any line. It is one undo step.

The length is the indicator’s **Tangent/Norm Length** (in inches or centimetres as the drawing’s units say; the new line is this long in total, centred on the point; start value 1.0; a button between Line Style and Join Style in this mode, steps of 0.1).

When several objects lie under the click, the search order is: ellipses and circles, then lines, then splines, then arcs, the last drawn first. At a corner of a polygon or a line’s vertex the tangent is not defined: the message panel says “singularity, can’t draw tangent/normal”. The message “added tangent” or “added normal” confirms success. On splines the tangent is that of the short curve piece found at HIGH_PRECISION, as in xfig, so at the lowest point of a curve it is horizontal. A click that is not on any curve does nothing.

Measuring

The three measure modes (the last three buttons of the editing column) report in the message panel; they change nothing in the drawing, add no undo step and leave the selection as it is. Lengths and areas appear in the drawing’s units (inches or centimetres, scaled by the Unit/Scale setting), with one decimal; inches show fractions of 1/64 unless the rulers count in tenths. Every number below also comes from the commands `object.measure_angle`, `object.measure_length`, `object.measure_area` and `object.measure_reset` (see Automation).

Measure angle

Key Ctrl+G (the physical Ctrl key). Measures an angle between three points you click, or at a corner of a line or at the centre of an arc.

- Left, “first point”: click the first point of the angle. A rubber band follows the pointer. The next click (“angle tip”) is the corner, then the third click (“final point”, left button) measures the angle at the corner between the two lines and reports “Angle defined by three points: 45.00 Degrees”, between -180 and 180 degrees (the sign says which way it turns).
- Middle, “select & save”: click a vertex of a line or polygon, or one of the three defining points of an arc. The angle is reported and also stored as the **Rotation Angle** (0 to 180 degrees) of the Rotate modes. While the three points are being placed the middle button’s label is “save angle”: the third point clicked with it also stores the angle.
- Right, “select object”: the same without storing. While the points are being placed the right button’s label is “cancel”.

At a polyline corner the report is “Angle at polyline corner: …”; at an arc “Angle at arc center: …” (the angle the arc sweeps, positive or negative by its direction). At the first or last point of an open line there is no corner: “Can’t compute angle at endpoint”. A polygon has a corner at every point, including the first. Escape also cancels the three points. A click on the same point twice is ignored.

Measure length

Key Ctrl+N. Measures the length of lines, polygons, boxes, arcs, circles and ellipses.

- Left, “select object”: click the object; the message panel says for example “Length of polygon is 3.0 in”. For a circle it is the circumference, for an ellipse an approximation of the perimeter, for an arc the length along the arc. A rounded box (arcbox) gives its perimeter less the circumference of a circle with the corner radius, as xfig does.
- Middle, “select & add”: the same, and the length is added to a running total: “Length of box is 2.0 in, accumulated 5.0 in”.
- Right, “reset to 0”: sets the running total to 0 (“length reset to 0”).

Objects the mode cannot measure (text, splines, compounds) show no markers and are not found by a click. Objects on a hidden depth are not found either.

Measure area

Key Ctrl+M. Measures the area of polygons, boxes, arcs, circles and ellipses; lines are not closed, so they have no area.

- Left, "select object": click the object: "Area of box is 2.0 square in". A circle or ellipse gives pi times the two radii; an arc counts as a sector ("sector") for a pie arc and as the segment between arc and chord ("open arc") for an open arc; a rounded box (arcbox) loses the area of a circle with the corner radius, as xfig does. A polygon whose sides cross itself is not measured correctly (as in xfig).
- Middle, "select & add": the same, and the area is added to a running total: "Area of polygon is 2.0 square in, accumulated 3.0 square in".
- Right, "reset to 0": sets the total to 0 ("area reset to 0"). With Shift held the label reads "reset to +-0" and the total is reset to 0 and signed from now on ("signed area reset to 0"): an area is then negative when the polygon's points turn counter-clockwise on the screen (an arc by its direction), so overlapping and opposite parts cancel in the total. The ordinary right button turns the sign off again. Metric areas are given with three decimals, "square cm".

Shift+left locates an object under another one ("locate object", see Mouse); the next plain click measures the located object. Shift+middle does nothing.

Attribute modes

These modes change objects that already exist.

Update attributes

Key `U`. Gives an object the current attributes of the indicator panel (colours, width, style, fill, arrows and so on). Which attributes are applied is set by the Update Control box shown in this mode: all, none or toggled flags, and individual ones.

- Left, "update object": click an object.
- Middle, "update settings": the reverse: the clicked object's attributes become the indicator's current ones.
- Right, "update region": update every object completely inside a region.

Edit attributes

No key (xfig key Ctrl+E). Opens the edit panel for an object with all of its attributes, text, points and comments. The panel is live: every change you make shows on the canvas at once, and Done (Ctrl+Return) keeps the whole edit as one undo step however many changes you made. Cancel (Escape) undoes everything since the panel opened. xfig's Apply button is left out: changes are applied as you make them.

- Line style, cap style, join style and the arrow types are pictorial menus: the button shows the picture of the current value, its pop-up lists every value as a picture (the same pictures as the attribute panel). Every row of the panel has the same height, labels sit on the left and controls in one column, and the point table has headings (#, X, Y, Shape) over its columns.
- Fill style (indicator and edit panel): the edit panel's "Fill style" row is a button showing the current fill; it opens the same choice panel as the indicator (NONE, shades and tints of the fill colour, patterns) and a label beside it says "No fill", "Filled 50 %" or "Pattern 3" as xfig's panel does. For the fill colours black, white and default the menu has no tints, as in xfig (a tint set earlier becomes the full colour).
- Points of a polyline, polygon or spline are a table: edit the X and Y of a row, Insert adds a point before that row, Delete removes it (the object keeps its minimum number of points), Add point appends one. Open Paste points, type or paste a list (one point per pair of numbers: `x y`, `x, y` or `(x, y)`, in the units shown in the

panel) and choose Replace points or Append points. X-splines have a Shape column with each point's shape factor (-1 interpolated, 0 angular, 1 approximated).

- A compound has its own panel, see Compounds.
- Left, "edit object": click an object.
- Middle, "edit Main comment": edits the figure's own comment. Shift+middle on an object shows its comments in the message panel.
- Right, "edit point": on a point of a spline, edits that point's shape factor (end points of an open spline cannot be edited).

Convert line ↔ spline

Key `V`. Turns a polyline into a spline through its points, or a spline into a polyline. Polygons and closed splines go with it. A spline that was turned into a line comes back as the same kind (approximated, interpolated or X-spline) when the next line is turned into a spline; the first conversion makes an interpolated spline. A box becomes a rounded box (arc-box) with the radius of the indicator's Box Curve, and a rounded box becomes a plain box.

- Left, "line/spline, box/arc-box": click the object.
- Right, "open/closed": click a point (vertex) of a line, polygon or spline: a closed object opens there, an open one closes. A polygon opens at the clicked point: the open line starts at that point and ends at the point before it (clicking the first point of a polygon opens it at the second, as xfig does). A click on an edge between points does nothing.

Open/close

No key. Closes an open line or spline (a polyline becomes a polygon) or opens a closed one.

- Left, "open/close": click the object.

Add/remove arrows

Key `Shift+A`. Arrowheads are shaped by the indicator's Arrow Type, Arrow Size and Arrow Mode.

- Left, "add arrow": click near the end of a line, arc or open spline; an arrowhead is added at that end.
- Middle, "delete arrow": removes the arrowhead at that end.
- Right, "reverse arrows": swaps the arrowheads of the object's two ends.

Libraries

A library is a folder of figures (`.fig` files) you can place in your drawing: electronic parts, flags, computers, flowchart blocks, maps and more. RustFig comes with xfig's libraries (about 2,500 figures in 30 libraries) and reads your own from a folder of yours.

Library mode

Library mode button (the wide button at the bottom of the drawing modes) or `Shift+L`. The library panel opens.

1. **Load a Library**: choose a library in the *Library* menu (`Electronic / Schematic / Classic` is the library `Classic` in `Schematic` in `Electronic`). The *Status* line says how many objects it holds.
2. Click an object: its icon or name is picked, its name shows next to *Selected object*, and its comments (the text xfig shows under *Object comments*) appear below with a preview.
3. **Select object** (or double-click the object) closes the panel and puts the object in hand: it follows the pointer, its top-left corner at the pointer.
4. Place it, as below.

Search: type in the *Search* field to find figures in all libraries at once, not only the loaded one. Every word you type must occur in the figure's name or in its comments (upper and lower case do not matter); the results list each figure with its library, and those found by their comments say so. Click a result: its library is loaded and the figure picked, then proceed as above (double-click, or *Select object*). *Clear* returns to the library's own list. A long result list is cut; type more of the name.

Texts with the TeX flag in a figure, such as `$\lambda/2$` or `$|0\rangle$`, show in the icons and the preview as a quick Unicode approximation (λ⁄2, |0⟩; Greek letters, common symbols, super- and subscripts, fractions); no TeX is run. The figure and what you place keep the LaTeX source.

View switches between *Icon View* (thumbnails; *Icon size* 40 to 120) and *List View* (names). Both are remembered (preferences `icon_view`, `library_icon_size`). *Cancel* closes the panel.

Placing

While an object is in hand, the mouse indicator shows:

- Left, "place object": places a copy at the pointer and stays in library mode, so you can place more. The copy is one compound, selected; each placement is one undo step.
- Middle, "new object": opens the library panel again to choose another object.
- Right, "cancel library": leaves library mode without placing.
- Shift+left, "place and edit": places the object and opens its edit panel.
- Shift+middle, "change draw mode": shows a box the size of the object instead of the object itself (faster on slow screens), and back.
- Shift+right, "place at orig posn": places the object where it is in the library file.

Keys while the object is in hand (xfig's): `r` turns it clockwise, `l` counter-clockwise, `h` flips it left to right, `v` up to down, `s` shrinks it by a tenth, `Shift+S` enlarges it by a tenth (all about the object's centre). `Escape` cancels.

Colours the library figure defines are added to the drawing's colours. Pictures a library figure uses stay in the library folder; the figure keeps pointing at them.

Your own libraries

Your library folder is the preference `library_dir` (default `~/xfiglib`; Preferences, `–library_dir` on the command line). Every folder in it that holds `.fig` files is a library; folders inside a folder are libraries of their own. Copy figures there with any file manager, or use *Copy to library* in the panel: give the figure a name (and a folder inside your library folder if you like); the selected objects, or the whole drawing if nothing is selected, are saved as a `.fig` file there. A comment typed in the figure's comments dialog becomes the *Object comments*.

A list of library folders

Like xfig, `library_dir` may name a plain text file instead of a folder. Each line holds a library folder and, after a space, the name to show for it (`/home/me/electronics  My circuits`); without a name the folder's own name is used. Lines starting with `#` and blank lines are ignored, `~` is your home folder, and a relative folder is taken from the folder the list file is in. Every folder named becomes one library, its sub-folders libraries of their own (`My circuits / Analog`), in the order of the file. *Copy to library* saves into the first folder of the list. The *Source* field at the top of the panel takes a folder or a list file; *Use* checks it, remembers it as `library_dir` and reads the libraries again.

Stop

A big library is read while the panel is open: the library folders are scanned in the background and the icons are drawn a few at a time, so the window stays usable. *Stop* (shown while this goes on) ends it: the scan keeps what it found so far and icons not yet drawn stay blank. *Resume* draws them.

The bundled libraries come with every package: in the `libraries` folder next to the program (Windows zip, Linux tarball), inside the app on macOS (`RustFig.app/Contents/Resources/libraries`), and in

`share/rustfig/libraries` for the `.deb`, `install.sh` and Homebrew. The folders named in the environment variable `RUSTFIG_LIBRARIES` take their place.

In the web version the page serves the bundled libraries as static files (`cargo xtask web` copies `libraries/` into `dist/` and writes the index `libraries.json` there). The panel fetches the index when it opens and each figure the first time it is drawn, so the first view of a big library fills in while you watch ("Loading icons…"); a figure is fetched again only after the page is reloaded. The browser has no folder for your own libraries, so the *Source* field, *Copy to library* and a user library are desktop-only. Search finds figures by name; searching the comment lines needs the figures to have been fetched, so it is left out in the browser. Selecting an object before its preview has appeared can say "still loading, try again": wait for the preview and select again.

Feynman diagrams

The `Feynman` library holds parts for Feynman diagrams, drawn for RustFig: fermion and anti-fermion lines (arrow in the middle, its tip a little past the middle so the arrow sits on the line), scalar (dashed), ghost (dotted), photon (wavy), W/Z boson (heavy wavy), gluon (curly) and graviton lines, each horizontal and one inch long (turn them with the Rotate modes); half-circle arcs of the same kinds for loops; a vertex dot, hatched and grey blobs; momentum arrows; and four complete examples (`example_ee_to_mumu`, `example_compton`, `example_self_energy`, `example_gluon_exchange`). To build a diagram, switch on grid snapping (1/4 inch), place a line part, then place the next part so that its end meets the first one; every line starts and ends on the grid, so parts join exactly. Put a vertex dot on each meeting point. Rotate or flip parts with the editing modes, and scale a line part to make it longer (the waves stretch with it). Label the lines with the Text mode in LaTeX text with the TeX flag (`$e^-$`, `$\gamma$`, `$\mu^+$`); the labels show as LaTeX in a PDF or TikZ export. Open an example as a starting point and change its labels.

Quantum circuits

The `QuantumCircuits` library holds parts for quantum-circuit diagrams in the look of Qcircuit and quantikz, drawn for RustFig: a qubit wire and a classical wire (double line), single-qubit gate boxes (`H`, `X`, `Y`, `Z`, `S`, `T`, `R_x(theta)` and an empty box), a measurement meter, control dot, open control and target, complete `cnot`, `cz` and `swap` parts on two wires, multi-qubit gate boxes over two and three wires, initial-state and ket labels, a barrier and a classical control line, plus four examples (`example_bell`, `example_ghz`, `example_teleportation`, `example_qft`). Qubit wires are one inch long and end on the 1/4-inch grid; the wires of a circuit are 1/2 inch apart, so with grid snapping on the parts line up and join. Labels are LaTeX text with the TeX flag (`$H$`, `$|0\rangle$`): the canvas shows them typeset by TeX (see *LaTeX text* in Text and pictures), and a PDF or TikZ export as LaTeX; change them with the Text mode. Search the libraries for `quantum circuit` to list every part.

Optics and lab

The `OpticsLab` library adds optical-table parts to xfig's own `Optics` library, drawn for RustFig the way physics papers draw setups: a laser, laser beams in xfig's red (solid, with an arrow, dashed, an expanded band and a diverging cone), convex, concave and cylindrical lenses, flat mirrors (plain and with a hatched back, perpendicular to the beam and at 45 degrees), a concave mirror, a dichroic mirror, beam splitter cube and plate, a polarizing beam splitter, half- and quarter-wave plates, polarizers, an iris, a coiled fibre and a fibre coupler, a photodiode, a detector, a CCD camera, acousto- and electro-optic modulators, an optical isolator, a prism, a grating, a cavity of two curved mirrors, a vacuum chamber, a cryostat, a viewport, a flange pair and a sample holder, plus three examples (`example_michelson`, `example_mach_zehnder`, `example_fibre_path`). Every part is one compound whose optical axis runs through its centre at a height on the 1/4-inch grid, and parts in the beam have their beam ports one inch apart on the grid: with grid snapping on, place parts at the same height and they line up; put the red beam parts between them. Rotate the 45-degree parts with the editing modes (they turn a beam from the left upwards), and flip the fibre coupler for the output end. Wave plates carry their labels as LaTeX text with the TeX flag (`$\lambda/2$`). Search the libraries for `optics lab` to list every part.

Physics circuits

The `PhysicsCircuits` library holds lumped circuit elements the way physics papers draw them, filling the gaps of xfig's `Electronic` library: resistors (rectangle and zigzag), a capacitor, an inductor, Josephson junctions (cross in a box and plain cross), a SQUID loop, DC and AC voltage and current sources, a battery, a transmission-line segment, variable versions with an arrow (resistor, capacitor, inductor, Josephson junction), grounds (earth with three bars, chassis, protective earth, analog and digital ground), a transformer and a mutual inductance, a port, a terminal and a wire junction dot, plus three examples (`example_rlc`, `example_transmon`, `example_resonator`). Every part comes in one orientation (two-terminal parts horizontal, grounds pointing down); turn it with the Rotate modes. All two-terminal parts have their two pins exactly one inch apart on the 1/4-inch grid: with grid snapping on, place parts so that pins meet and join them with short wires drawn with the Polyline mode; put a junction dot where three wires meet. A ground has one pin, on top; a transformer has four pins on the grid. The examples carry LaTeX labels with the TeX flag (`$C_c$`, `$E_J$`): change them with the Text mode. Search the libraries for `circuit` to list every part.

Commands

For scripts, the control channel and MCP: `library.list`, `library.source` (what `library_dir` names: a folder, or a list file with its entries), `library.use` (`path`: make a folder or list file the source), `library.search` (`query`, `limit`, `comments`: figures of all libraries by name and comments), `library.objects` (`comments: true` adds each figure's comments), `library.place` (`library`, `name`, `x`, `y` or `orig`, `ops`), `library.hand` (put the object in hand as Select object does), `library.transform`, `library.save`, `library.panel`. See Automation.

Credits and licences

The figures belong to their authors, named in the READMEs in each library folder and in `ASSETS.md`; they are distributed with xfig under the xfig licence (the ERD and GUI libraries carry their own copy of it). The `Charts` library (released under the GPL) and two example drawings with restrictive notices are not bundled.

Plotting a curve

RustFig can draw a graph for you: a function y = f(x), a parametric curve x(t), y(t), or the columns of a data file. The result is ordinary Fig objects, so it prints, exports and scales like the rest of the drawing and opens in xfig. xfig has no such mode; this is an addition.

A plot is **one compound**: the frame, the ticks with their numbers, an optional grid, the axis labels and the curve as polylines. Open the compound (see Compounds) to touch up one part. The plot's settings are kept in the compound's comment (a line starting `rustfig-plot:`), which Fig 3.2 files keep, so a plot can be edited again later, also after saving and opening.

Plot

Key `y`. The last button of the Drawing column, beside the library button; the same as the Edit menu's **Plot curve…** (Ctrl+Shift+P, Cmd+Shift+P on macOS).

- Left, "first corner": click where one corner of the plot's frame goes, as in Box mode. A rubber-band rectangle follows the pointer and the left button now says "final point": click the opposite corner and the Plot dialog opens for that frame. A single click opens nothing. Nothing is drawn until you press **Draw plot** in the dialog; Cancel leaves the drawing as it was.
- The right button, "cancel", drops the frame you have started. The middle button does nothing in this mode.

Edit > **Plot curve…** opens the same dialog without placing a frame: the plot goes to the origin at the default size.

The Plot dialog

Choose what to draw:

- **y = f(x)**: type the expression and the range of x. Samples (default 400) are spread evenly and more are added where the curve bends, so peaks stay sharp.
- **x(t), y(t)**: two expressions in t and the range of t (default 0 to 2π, written `2*pi`).
- **Data file**: a CSV or whitespace separated file (comma, semicolon, tab or blanks). Give the number of the x and y columns, counting from 1; x column 0 uses the row number. Lines starting with # or % and lines that are not numbers (a header) are skipped. Files up to 16 MB are read. This works on the desktop only; on the web use the command `plot.add` with the text as `data`.

The ranges of x and y can be left empty to fit the curve (the y range of a curve with poles ignores a few extreme values, so 1/x does not flatten everything else); the numbers are rounded to tick values. Numbers may be expressions such as `2*pi`.

The dialog has four groups: **Function** (the mode, its fields, samples), **Range** (x and y, each with **Logarithmic**: powers of ten, the range must be positive), **Axes** and **Labels**.

Axes has a small table with the rows **x axis** and **y axis** and the columns **Ticks** and **Numbers**: each box turns the tick marks or the numbers of that axis on or off by itself. **Box** draws the frame around the plot and **Grid** adds dotted lines at the numbered ticks. They are independent: with Box off the ticks still stand on the edges of the (invisible) frame. Plots made before these switches (one "axes" setting) open with Box, ticks and numbers all on, or all off if axes were off.

Labels holds the x and y label, plain text or LaTeX text with the TeX flag when **LaTeX labels** is on (they are then typeset when you export to LaTeX or turn on View > Typeset LaTeX text). **LaTeX numbers** writes the tick numbers as `$...$` texts with the TeX flag too (a power of ten as `$10^{-5}$`, multiples of pi as `$\pi$`). LaTeX labels and numbers are texts in the LaTeX font with the TeX flag set: the canvas typesets them when TeX is installed (View > Typeset LaTeX text), and the LaTeX, pstex and pdftex exports pass them on unchanged.

x axis tick settings and **y axis tick settings** (collapsible, under Axes) set the details of each axis:

- **Manual**: the first tick, the step (or a count of intervals); empty fields stay automatic. Without Manual the ticks are chosen for you, as in plots made earlier.
- **Minor per major** (0 to 9) and the **length** of the ticks (Fig units, 1200 per inch; minor ticks are half as long), their **direction** (in, out or both).
- **Number format**: automatic, a fixed number of **decimals**, **scientific**, or **multiples of pi** (0, pi/2, pi, 3pi/2; the automatic step is then pi/2).
- On a logarithmic axis the step counts decades and the minor ticks stand at 2 to n+1 times a decade.

Too many ticks, a step of zero or less, and similar values are refused with an error. The settings are kept in the plot's comment like the rest.

The line style, thickness, colour, depth and the font of the numbers and labels are the current ones in the attribute panel; set them before you plot.

Expressions use `+ - * / ^` (also × ÷ and **), parentheses, the constants `pi` and `e`, and the functions sin, cos, tan, asin, acos, atan, sinh, cosh, tanh, exp, ln (also log: natural), log10, log2, sqrt, abs, floor, ceil, sign, and with two arguments atan2(y, x), min(a, b), max(a, b), pow(a, b). `-x^2` means -(x ^ 2) and `2^3^2` is $2^{(3}2)$. An expression of up to 500 characters and nested up to 64 deep is accepted; an error names the position.

Where the function is not defined (division by zero, the root of a negative number) or beyond the plot's range, the curve simply stops: there is never a line across a pole such as that of 1/x or tan(x), and parts outside the frame are cut at the frame.

Edit plot

A plot stays one compound that carries the `rustfig-plot:` comment, and every part RustFig generated (the frame, grid lines, ticks, numbers, labels and the curve) carries a marker comment such as `rustfig-plot-part: curve`.

- **Edit inside**: use Open compound (see Compounds) to move, restyle or delete single parts, or to add your own objects (an arrow, a note, a second curve you drew by hand).
- **Redraw**: while the compound keeps its plot comment, Edit > **Edit plot…** (and Edit mode's left button on the plot) opens the dialog filled with the plot's settings, and **Draw plot** redraws it in its present frame, where

you have moved or scaled it. Redraw replaces only the marked parts. Objects you added are kept, with their changes. A generated part you edited keeps its marker, so it is replaced by the redraw as well; to keep such a part, remove its marker comment first (Edit mode, comments) or draw a copy. Undo brings the old plot back.

- **Break**: Break compound (see Compounds) turns the plot into an ordinary drawing: the plot comment goes away with the compound and the parts lose their markers. It can no longer be edited as a plot.

Commands

For scripts and MCP (see Automation): `plot.add` draws a plot (frame `x`, `y`, `width`, `height`; the settings `mode`, `expr`, `xt`, `yt`, `tmin`, `tmax`, `file`, `xcol`, `ycol`, `data`, `xmin`, `xmax`, `ymin`, `ymax`, `samples`, `axes`, `grid`, `log_x`, `log_y`, `xlabel`, `ylabel`, `latex`), `plot.update` changes some settings of an existing plot, `plot.get` returns them, `plot.panel` and `plot.edit` open the dialog. Hostile or huge input is refused with an error: the expression length, the number of samples (20,000), the points of the curve and the size of a data file are limited.

The indicator panel

The bottom strip holds the attributes that new objects get and that the current mode needs. As in xfig, which buttons appear depends on the mode: a drawing mode shows what that shape has; editing modes such as Update show all of them. Every button works with the mouse like xfig's: the left button opens a menu of choices, the middle button steps to the previous value and the right button to the next (the three words "Menu", "Dec/Prev" and "Inc/Next" under the strip remind you).

Buttons that exist (labels as in xfig):

Button	What it sets
Zoom	the zoom factor
Grid Mode	grid style and spacing (Snap and grid)
Point Posn	how points snap while drawing (Snap and grid)
Depth	the depth (stacking order): lower is in front
Rotn Angle	the angle of the rotate modes
Num Sides	sides of a regular polygon
Vert Align, Horiz Align	alignment of the align mode
Angle Geom	constrains line directions (Angle Geom)
Smart Links	attached lines follow (Smart links)
PenColor, FillColor	line (or text) colour and fill colour; the menu has xfig's standard colours, your own colours, the colours used lately, and **More...** (see Colours below)
Fill Style	none, solid, tints and the pattern fills
Arc Type	open or pie-wedge arcs
Line Width	in 1/80 inch; thickness 0 draws no line
Line Style	solid and the dash styles (Line styles)
Tangent/Norm Length	Tangent and normal
Join Style, Cap Style	corners and line ends (miter, round, bevel; butt, round, projecting)
Arrow Mode, Arrow Type, Arrow Size	arrowheads
Box Curve	corner radius of rounded boxes
Text Just, Text/Ellipse Angle, Text Flags, Text Size, Text Step, Text Font	text settings

Text Flags shows one flag at a time as xfig does (`hidden=off`, `TeX=on`, `rigid=off`); the right or middle button shows the next or previous one, a click opens the panel that sets all three. **Text Size** shows two decimals (`12.00`). The number pop-ups (Depth, Rotation Angle, ...) have a Value field with Cancel and Set.

Text Font opens the font chooser (also in the edit panel's Font row). The switch **PostScript / LaTeX** picks xfig's two lists: the 35 PostScript fonts, or the six LaTeX fonts (Default, Roman, Bold, Italic, Sans Serif, Typewriter).

Every name is drawn in its own font (with only the built-in fonts, every name is drawn in the fallback and tagged "(substitute)"); the current font is marked with a dot and highlighted. Type in **Search fonts** to narrow the list (every word must occur in the name). **Size** sets the Text Size, and the preview line shows sample text in the font under the pointer (or the current font) at that size. Click a name to choose it.

Which fonts are used. Text is drawn with the fonts installed on your system: for each of xfig's 35 PostScript fonts RustFig looks for Times, Helvetica, Courier, Symbol, Zapf Dingbats, Palatino, Bookman, Century Schoolbook and so on (or their URW and TeX Gyre clones), then for the nearest serif, sans-serif or monospaced face, and last for its built-in fonts. The TeX Gyre fonts (metric clones of the PostScript fonts) are used when a TeX installation or system package provides them. If you have none, **Help > Install TeX Gyre fonts…** downloads them from CTAN (about 21 MB), checks the download's SHA-256 checksum, and keeps only the OpenType files and their licence in RustFig's user data folder; a failed or cancelled download leaves nothing behind, and the fonts are used from then on. Setting `FIG2DEV_BUILTIN_FONTS=1` makes RustFig use only its built-in fonts (the same drawing on every machine).

Arrow Size opens xfig's panel: **Absolute Values** or **Multiple of Line Width**, each with Thickness, Width and Length (0.1 to 10000; absolute values are in 1/80 inch). Only the set you chose is used for new arrows; with the multiples (xfig's start state: 1, 4 and 8) an arrow grows with the line width. **Set** (or Return) applies, **Cancel** keeps the old values. "Use current attributes as defaults" keeps the choice for the next start.

Number pop-ups (Line Width, Text Size, Depth, Rotn Angle, Num Sides, copies, Box Curve, Text Step, Zoom and others) work as xfig's: type or drag a value, then **Set** or Return applies it and closes the pop-up; **Cancel** closes it without a change. The Zoom pop-up also has **Integer zoom**: zoom values above 1 become whole numbers when you set them there (rounded) or use Fit (truncated).

Grid Mode has nine choices: none, four square grids and four **isometric** grids (vertical lines and lines at plus and minus 30 degrees; points snap to the crossings, Snap and grid). **Vert Align** and **Horiz Align** have xfig's seven choices: none, start (left or top), centre, end (right or bottom), distribute centres, distribute edges and abut (Transform).

The Update Control box (in Update mode) says which attributes Update applies. Edit > Global settings has "Use current attributes as defaults" to keep the panel's values as the start values for new drawings (Preferences).

Colours

Click PenColor or FillColor for the colour menu: Default, the 32 standard colours, the colours of the drawing and your own colours. Under them are the **recent colours** (the last twelve you used, kept in your settings, not in the drawing) and **More…**.

- **More…** (also Edit > Colours…) opens the Colours window:
  - Pick in the hue/saturation square and the hue strip, or type Red, Green, Blue and Hue, Saturation, Value.
  - Type a hex value (`#rrggbb` or `#rgb`) or an X11 colour name such as `Dark Orange` (case and blanks do not matter; matching names are offered while you type) into **Hex or name**.
  - **Use as pen colour** and **Use as fill colour** make the colour current; a colour that is not one of the 32 standard colours becomes a user colour of the drawing.
  - **Colours in this drawing** lists the user colours (numbers 32 to 543), saved in the Fig file. Click one to edit it, then **Change colour N** (every object that uses it changes), **Delete colour N** (refused while an object uses it) or **Add as new colour**. **Remove unused colours** drops those no object uses.
  - **Pick from screen** takes the colour of any pixel on the screen, also in other programs' windows, and shows it in the window; apply it with **Use as pen colour** or **Use as fill colour** (only then does it become a user colour of the drawing). It is an addition: xfig has nothing like it. On macOS the system's colour sampler (a magnifier) opens: click a pixel to take it, Escape cancels. It needs no permission. On Windows the colour under the pointer is shown live next to the button; click to take it (the click also reaches the window you click on), Escape cancels. On Linux the desktop's colour picker opens through xdg-desktop-portal (Wayland and X11, GNOME and KDE provide it); where no portal runs, the message line says so. In the web app the button is greyed out: the browser's EyeDropper needs JavaScript.
  - **Keep for all drawings** stores the colour under a name in your settings, so it is offered in every drawing.

Scripts and MCP use the `color.*` commands (`color.parse`, `color.new`, `color.edit`, `color.delete`, `color.recent`, ...).

Styles

View › Manage Styles... (Ctrl+Y with xfig's keys on) keeps named sets of attributes in families, as xfig's style manager does, so a look you use again and again is one click away.

- The window shows two lists side by side: **Family** (a name field, **Add**, **Delete**) and **Style** (a name field, **Add**, **Delete**, **Set**). Pick a family to see its styles.
- A style holds the indicator's drawing attributes: depth, line width, style, join, cap, dash length and dot gap, pen and fill colour, fill style, box radius, arrows, the text font, size, angle, justification and flags, and the arc type.
- Set the indicator panel as you want it, pick a family, type a name and press **Add** under Style (a style of the same name is replaced).
- **Set** sets the indicator, so what you draw next looks like the style; objects already drawn are not changed (as in xfig). **Delete** removes the selected family or style. **Close** (or Escape) closes the window.
- Limits: 32 families, 64 styles per family (xfig: 16 and 16), names of 1 to 64 characters.
- Styles are kept in your settings file, for all drawings. They are never written to the .fig file, which stays as xfig reads it.

Scripts and MCP use `style.family_add`, `style.family_delete`, `style.family_rename`, `style.save`, `style.apply`, `style.rename`, `style.delete` and `style.list`, each with a `family` parameter.

Line styles

Line styles are chosen with the indicator's **Line Style** button: solid, dashed, dotted, dash-dotted, dash-double-dotted and dash-triple-dotted.

One number per line

A Fig file keeps only one style value per line, in 1/80 inch: the dash length for the dashed styles, or the gap between dots for Dotted. There are no separate dash, gap and dot lengths in the format (an extension for them was considered and declined, so every Fig reader draws the same line). The rest of the pattern follows from that number v, as fig2dev draws it (and RustFig's canvas and exports do):

Style	Pattern (v is the style value)
Dashed	dash v, gap v
Dotted	dot, gap v
Dash-dotted	dash v, gap v/2, dot, gap v/2
Dash-double-dotted	dash v, gap 0.45v, dot, gap 0.33v, dot, gap 0.45v
Dash-triple-dotted	dash v, gap 0.4v, dot, gap 0.3v, dot, gap 0.3v, dot, gap 0.4v

A dot is a very short dash.

The two defaults

The Line Style panel keeps two defaults for new lines, as xfig: the **Default dash length** (used by the dashed styles) and the **Default dot gap** (used by Dotted). These supply the style value of a new line. To change the value of an existing line, open its edit panel (Edit attributes), which shows the style value with the explanation above.

Angle Geom

Angle Geom (the "Angle Geom" button in the indicator panel) limits the directions of the line segments you draw, so that lines come out horizontal, vertical or at LaTeX's allowed slopes without careful aiming. xfig calls this point positioning for lines.

The button is shown in the modes where it applies: Polyline, Polygon, the spline modes, Move point and Add point. The six choices:

Choice	Directions allowed
Unconstrained	any (the default)
Latex line	the slopes LaTeX's picture environment can draw for lines
Latex arrow	the slopes LaTeX allows for arrows (vectors)
Mount-hattan	horizontal, vertical and 45 degree diagonals
Manhattan	horizontal or vertical only
Mountain	45 degree diagonals only

How it works: the point you click, and the rubber band you see while moving, are placed in the nearest allowed direction from the previous point. When you move or add the end point of an open line, the constraint is taken from its neighbouring point. Points in the middle of a line, and points of closed shapes, are not constrained when moved.

xfig has no Shift key for 45 degree lines (Shift+left removes the last point while drawing), so there is none here: use Mount-hattan.

Angle Geom works together with the grid and Point Posn (Snap and grid).

Smart links

Smart links keep lines attached to shapes. When you move a shape, the ends of lines that touch it move with it, so a diagram's connectors stay connected. Nothing is stored in the file: a line end counts as attached when it lies within a few pixels (45 Fig units, about 0.04 inch) of the shape's outline at the moment you start the operation.

Turning it on

The indicator's **Smart Links** button has three choices:

- **Off** (the default): nothing follows, as in plain editing.
- **Move**: an attached end point moves with the shape.
- **Slide**: the end point moves and the point before it slides so that the last segment keeps its horizontal or vertical direction. Good for right-angle connectors.

The button shows in Move, Copy, Move point, Scale, Flip, Rotate and Align modes.

What attracts lines (targets)

Boxes, rounded boxes and pictures (the edges), compounds (anywhere inside their box), and, within 45 Fig units of the outline: circles, ellipses, arcs, polygons, closed splines and text boxes.

What follows (followers)

Open polylines and open splines. A line attached to two shapes that move together follows at both ends. Only the end points are linked, not points in the middle.

Operations that carry the links

Move, Copy (the attached lines are copied with the shape; the original line's end follows the copy as in xfig), rotate, flip, scale, align, the arrow-key nudge command, and Move point on a box corner (resizing the box stretches the attached lines with its edges; this last one is a RustFig extension). The Move, Copy, Move point and Scale previews show every follower.

Undo reverts the shape and its lines in one step.

Snap and grid

Points you click are moved to a nearby position, so lines meet and shapes line up.

Point Posn and Grid Mode

- **Point Posn** (indicator) sets the step that coordinates are rounded to: any (no rounding), 1/16, 1/8, 1/4, 1/2 or 1 inch (1, 2, 5 mm, 1, 2 cm in metric drawings). xfig starts with 1/16 inch, its "magnet".
- **Grid Mode** (indicator) chooses the grid spacing and style, or none: xfig's nine choices, four square grids and four isometric ones (`indicator.set` with `grid` and `grid_iso`; resource `startgridmode` 0 to 8). On an isometric grid the columns are cos 30 degrees of the spacing apart and every other column is shifted by half a spacing; points snap to those crossings (the spacing is the Point Posn step). The grid is shown on the canvas and points snap to it (View > Show grid, Ctrl+').

Pressing the middle button in the move modes (horiz/vert) constrains the move after snapping.

Object snaps (Snap menu)

The Snap menu arms a snap for the next point you click:

Snap	Snaps to
Endpoint	the end point or vertex of an object
Midpoint	the middle of a segment
Nearest	the nearest point on an object
Focus	the focus points of ellipses
Normal	the foot of a perpendicular from the previous point
Tangent	a tangent point from the previous point
Intersection	where two objects cross
Diameter	the opposite end of a circle's diameter

Hold keeps the chosen snap after each point until you pick **Release**. The Snap Mode box next to the menu says which snap is armed.

Zoom and view

- **Zoom in** and **Zoom out**: the View menu, keys Shift+Z and Z (xfig), or Ctrl+= and Ctrl+-. Ctrl+scroll and pinching also zoom.
- **Zoom to fit canvas** fits the whole drawing (Ctrl+Z with the physical Ctrl key, or Ctrl+0). **Unzoom** returns to actual size (Ctrl+1). **Zoom In** and **Zoom Out** (also `z` and Shift+Z) step by the square root of two as in xfig: 1, 1.41, 2, 2.83, 4, 5.66, 8, 11.3 and so on (Integer zoom rounds above 2). The indicator's Zoom button sets a value (with **Integer zoom**, Cancel and Set; command `view.set_integer_zoom`). Right-click the units box at the top for xfig's **Unit/Scale** panel (command `app.unit_panel`): Ruler units (imperial or metric), **Show fractions** (inches only; off shows tenths), Figure units (**Ruler units** or **User defined** with a name) and Figure scale (**Unity** or **User defined** with a factor); the name and factor are greyed unless user

defined. **Set** (or Return) applies, **Cancel** discards. **View > Show vertex numbers** numbers the points of lines and splines in red, from 0 as xfig does.

- **Pan**: scroll, the buttons at the ends of the rulers (Shift pans five times as far), the arrow keys, or hold Space and drag. **Pan to origin** (Home) returns to the page's corner.
- **Zoom box**: hold Ctrl and click with the left button (Cmd on macOS), then drag and release, or click the opposite corner; the view zooms so the box fills the canvas, its top-left corner at the canvas corner. Right button or Escape cancels. While Ctrl is held the mouse indicator shows "Zoom area / Pan to origin / Unzoom": Ctrl+middle pans to the origin, Ctrl+right returns to 100 %. A zoom box cannot start while a tool is busy. On Linux and Windows Ctrl+click is therefore not the right button; use the real right button, Alt+click for the middle button or the two-button chord.
- **Redraw** (Meta+R, Ctrl+L) repaints the canvas.
- **Portrait/Landscape** turns the page (Meta+C).
- **Show page borders** draws the page's outline in light blue with the paper's name outside its lower-right corner (in multiple-page mode, from the Export or Print panel, the outline of every page); turned off, the whole canvas is paper, as in xfig; **Show depth manager** shows the list of depths with check boxes to hide or show each; **Show info balloons** turns tooltips on and off (the delay is a global setting); **Show line lengths** (Alt+L with xfig's keys) shows the dx / dy triangle with the lengths of its sides while a line is drawn or moved (the message panel always shows length, dx, dy and angle).
- The rulers show the cursor position. The units box shows the current units; click it for Set units (inches or centimetres, a scale for the drawing).
- **Document settings** has the units, paper size and orientation.

Files

RustFig's own formats are xfig's **Fig** files (`.fig`, Fig 3.2 on writing) and `.rustfig` (a lossless JSON version of the same drawing). RustFig reads Fig 1.3 to 3.2, so old figures open too; warnings appear in the message panel where an old file needed repairs.

- **New** starts an empty drawing; **Open** opens a file; **Merge** adds a file's objects to the current drawing. If the drawing has unsaved changes, RustFig asks first.
- **Open with preview…** and **Merge with preview…** (File menu, commands `app.open_panel` and `app.merge_panel`; native versions only) show xfig's file panel: the files of the **Current Dir** that match the **Filename Mask** (blank-separated patterns with `*`, `?` and `[set]`; the default is `*.fig *.rustfig *.xfig *.fig.gz *.fig.Z *.pdf *.svg *.svgz`), the **Directories** to go into (`..` goes up, **Home**, **Show Hidden**, **Rescan**), and the **Filename** field (Return opens or merges it; **Browse…** uses the system's own file dialog). Clicking a file previews it on the right: the figure drawn small (special LaTeX text as a quick Unicode approximation, `$\lambda/2$` as λ/2, as in the library previews), its size (in the open figure's units), the number of objects and the **Figure comments**. **Open** loads the file as File > Open does (asking first about unsaved changes); **Merge** adds it as one compound, moved by the **Figure Offset** X and Y, each in Inches, cm or Fig units (`file.merge` with `offset_x`, `offset_y` and `unit`). Scripts can ask for the list and the preview with `file.list_dir` and `file.preview`.
- **Open** also takes a PDF, SVG or PNG exported with **Embed the figure** (Export) and opens the figure inside it, under a `.fig` name next to the file; the same goes for dropping such a file on the window.
- **Open** and **Merge** also take any other **PDF** and make editable Fig objects from one of its pages (plan G5; `file.open` / `file.merge` with `page`, default 1; `file.pdf_pages` counts the pages). A PDF with several pages first asks which page. The page's top-left corner is the figure's origin. Open names the figure like the PDF with `.fig` (Save never overwrites the PDF); Merge adds the page as one compound. What the import makes:
  - **Lines and shapes**: straight paths become polylines and polygons, a rectangle a box, a full axis-aligned ellipse an ellipse; curves are flattened into polylines (one Fig unit of tolerance, so very curved drawings have many points). Stroke width, dashes (dashed or dotted), caps, joins, and stroke and fill colours become Fig attributes; a colour that is not one of Fig's 32 becomes a new user colour (at most 512). A filled shape is filled solid (fill style 20, full saturation of its colour); a shape with holes becomes separate shapes. Pattern and gradient fills are drawn in grey.
  - **Text**: glyph runs with readable Unicode become text objects in the nearest of Fig's 35 PostScript fonts

(Times, Helvetica, Courier and the others by name, bold and italic kept), with size and angle; a word set glyph by glyph is joined with the text next to it. The PDF's own font is not kept, so line breaks and widths may differ a little. Text whose characters cannot be read (Type 3 fonts, unusual encodings) becomes filled outlines.
  - **Images** become picture objects; the pixels are written as PNG files `NAME-pPAGE-N.png` next to the PDF (and kept in memory), so the figure saved there shows them.
  - **Dropped or approximated**: clipping to a convex shape cuts the lines it clips; clipping to a concave shape, transparency, blend modes and soft masks are ignored, with one message each in the message panel (and the `warnings` of the command). Rotated images take their bounding box. A PDF with more than 200,000 shapes keeps the first 200,000. Encrypted or damaged PDFs give an error and leave the figure as it was.
  - A PDF exported with **Embed the figure** still opens as the figure inside it; `vectors: true` imports the page instead. PDFs from other programs come out as drawn: expect many small objects.
  - In the browser the page is always page 1 and the images stay in memory.
  - A PDF already placed as a picture can be turned into objects in place with **Import as objects** in the picture's Edit panel (plan G6, `picture.import_objects`; see Pictures).
- **Open** and **Merge** also take any other **SVG** or **SVGZ** file and make editable Fig objects from it (plan G7; `file.open` / `file.merge`, also with `file.open_data` / `file.merge_data` in the browser). The picture is drawn at 96 pixels per inch (a width in `mm`, `cm`, `in` or `pt` is honoured) with its top-left corner as the figure's origin; Open names the figure like the SVG with `.fig`, Merge adds it as one compound. What the import makes:
  - **Lines and shapes**: straight paths become polylines and polygons; a rectangle becomes a box, a rectangle with equal round corners an arc-box, a circle or an ellipse (also turned) an ellipse; curves are flattened into polylines (one Fig unit of tolerance). Transforms, `use`, CSS styles and `viewBox` are resolved first. Stroke width, dashes, caps, joins, and stroke and fill colours become Fig attributes (new colours as user colours, at most 512); an open path with a fill stays an open polyline, filled as Fig fills those. Gradients are drawn in the average of their stop colours and patterns in their ink colour, solid, with a message each.
  - **Text** with a simple layout (one style per run, no text on a path) stays text: position, size, angle and left/centre/right alignment (`text-anchor`) are kept and the font family is mapped to the nearest of Fig's 35 PostScript fonts (Times, Helvetica, Courier and the others by name or `serif`, `sans-serif`, `monospace`; bold and italic kept). Other text (on a path, vertical, mixed styles in one run) becomes filled outlines.
  - **Images** become picture objects: embedded (`data:`) images and image files in the SVG's own folder (not elsewhere, no `..`); the pixels are written as PNG files `NAME-N.png` next to the SVG (and kept in memory). An SVG inside an SVG is imported in place.
  - **Dropped or approximated**: clip paths, masks, filters, opacity, underlines and letter offsets are ignored, with one message each in the message panel (and the `warnings` of the command); holes in filled shapes become separate shapes; rotated images take their bounding box. A drawing with more than 200,000 shapes keeps the first 200,000, and documents nested more than 256 levels deep are refused. Damaged files give an error and leave the figure as it was.
  - An SVG exported with **Embed the figure** still opens as the figure inside it; `vectors: true` imports the drawing instead. Other PNG files give "no embedded figure".
- **Save** and **Save As** write the file. When they replace an existing file, the old one is kept as `NAME.bak` (xfig's `write_bak`; switch it off with the `write_bak` preference or `-nowrite_bak`).
- **Open** of a name that does not exist yet starts an empty figure with that name (xfig's "new file"); the first Save creates the file. While a compound is open (Compounds) a query asks whether to save the visible part or all.
- **Import picture** places an image file in the drawing (Text and pictures).
- The most recent files are listed at the end of the File menu and open with their number key.
- Click the **file name** in the top strip to rename the file.
- **Save As** into another folder keeps pictures working: relative picture names are rewritten so they still point at the same files (xfig's `relative_path`); absolute names stay as they are. Undo after the save brings back the old names; save again to rewrite them.
- **Digitize...** (Alt+Z) saves the points of the selected polylines and splines as text for a plotting program, xfig's Digitize the other way round. Name the files by prefix, starting number and suffix (`PREFIX_NNNN.SUFFIX`,

four digits, as xfig's example line shows); **New file for each line** writes one numbered file per object, **Append to one file** adds all lines to a single file (a blank line between lines, added to the file if it exists). A suffix of `csv` writes comma separated values with an `x_fig,y_fig` header, any other suffix one `x y` pair per line. Without calibration the numbers are figure units (1/1200 inch). Tick **Calibrate to data units** and give two reference points of a scanned plot (position in the figure and the value there) to get the plot's own units, with optional logarithmic axes. Boxes, arcs and pictures are not lines and are skipped. Command `file.digitize`; native versions only.

- **New window…** starts a second RustFig, optionally on a file (xfig's New xfig button); native versions only.
- **Autorefresh mode** (View menu, off at start, kept in the preferences): once a second RustFig looks at the open file and reloads it when another program changed it. If the figure has unsaved edits you are asked first: Reload (your edits are lost) or Keep my version. Saving from RustFig itself never triggers it. Native versions only.
- **Print** (Meta+P): the Print panel with page preview (Print).

Autosave and recovery

While you work, RustFig writes a recovery copy of any drawing with unsaved changes a few seconds after the last change. If the program stops without saving (a crash, power loss, a killed process), the next start offers to restore each drawing it finds, or discard it. After a normal save or exit the copy is removed. Restore a drawing, check it and save it where you want.

In the browser the copy goes to the browser's local storage (as Fig text, drawings up to 2 MB). Reload the page and RustFig offers to restore it or discard it; saving the drawing or starting a new one removes the copy. Clearing the site's data, or a private window, loses it, so download (File > Save) what matters.

Saved versions. Each time you save, RustFig keeps the file it replaces in the recovery folder. File > Saved versions… lists the last versions of the open file with their times; Restore puts one back as the open drawing (save it to keep it). Global settings > Saved versions kept sets how many (default 5, 0 switches it off, at most 50). Versions are kept only where the recovery folder is available (native app). The commands are `file.versions` and `file.restore_version`.

Starting from the command line

Running `rustfig FILE.fig` opens a file at start. See Automation.

Opening from the file manager (macOS)

Double-clicking a `.fig` file in Finder, choosing Open With > RustFig, `open -a RustFig FILE.fig` or dropping a file on the Dock icon opens it, whether RustFig is already running or is started by it. As with File > Open, RustFig asks before it discards unsaved changes; several files sent at once open one after the other.

The web version

RustFig also runs in a browser (WebAssembly). Build it with `cargo xtask web` (needs `trunk`: `cargo install --locked trunk`) and serve the `dist/` folder with any static web server, or run `cargo xtask web --serve` and open http://127.0.0.1:8080. The page shows "Loading RustFig…" until the program has started.

In the browser there are no file paths. **File > Open** shows the browser's file picker (a Fig file, a PDF or SVG to import, or a PDF, SVG or PNG export with the figure embedded); you can also drop such a file on the window. **File > Save** and **Save As** download the figure as a file named after the open one (`figure.fig` for a new figure; the browser decides the folder), and **File > Export** downloads each exported file. The menu leaves out what only makes sense on the desktop: Open and Merge with preview, Digitize, Capture Screen, New window, Saved versions and Exit.

File > Merge (Alt+M) shows the same picker and adds the picked figure as one compound, as on the desktop.

Pictures in the browser. Picture mode, Import picture and the picture panel's Browse/Relink show the browser's picker too (a figure's pictures can also be dropped on the picture panel). The browser gives RustFig the picture's

bytes, not a place on disk, so RustFig keeps them in memory under the file name for as long as the page stays open, and the figure stores only that name, as always. After you reload the page, or when you open a `.fig` that names pictures, the pictures are empty boxes with their names until you pick them again (Reread in the picture panel, or Relink: pick the file with the same name). Saving the figure downloads only the `.fig`; keep the picture files next to it to see them on the desktop. Exports made in the browser (File > Export) do not contain these pictures yet. EPS, PDF and compressed pictures need the desktop version: the browser build reads PNG, JPEG, GIF, BMP, PNM and TIFF.

Libraries in the browser. Library mode works with the bundled libraries, which the page serves as files next to the program (see Libraries). The Source line reads "bundled libraries" (there is no folder to choose). - **Capture Screen…** (File menu, command `app.capture`; xfig's Screen Capture button; not in the browser) grabs a window or an area of the screen into a picture. RustFig's window steps aside (it is minimised) while the system's own selector runs: on macOS `screencapture` (drag an area, press Space to pick a window, Escape cancels; RustFig needs the Screen Recording permission in System Settings > Privacy & Security), on Linux the desktop's Screenshot portal, on Windows the whole screen (no selector). Then a Save dialog asks for the PNG file name (default `<figure>_<seconds>.png`, as xfig named its captures, next to the figure); the image is written there and added to the drawing as a picture object (like **Import Picture…**) at the centre of the view, ready to move and resize. Cancelling the selector or the Save dialog adds nothing.

Export

File > Export (Meta+X, Ctrl+E) writes the drawing in another format. To print, use File > Print (Print).

The Export panel

- **Language** (format): the list holds xfig's export languages. Vector formats keep vectors: PDF, SVG, EPS and PostScript, the LaTeX and TeX-family languages (LaTeX picture, epic, eepic, PiCTeX, TikZ, PSTricks, pict2e, pstex and pdftex pairs, and others), MetaPost, METAFONT, CGM, HPGL, DXF, EMF, Gerber, Tk and more. Bitmap formats: PNG, GIF, JPEG, TIFF, PPM, PCX, XBM, XPM. Entries still being ported are greyed with their plan task.
- **Asymptote** (`asy`, a RustFig addition, not in xfig's fig2dev): the figure as an Asymptote program; `asy -f pdf figure.asy` (or `-f eps`, `-f svg`) turns it into a picture, with texts set by LaTeX as in the other TeX languages, so the TeX fonts of the Fig fonts must be installed (TeX Live's `collection-fontsrecommended`). Every object kind, arrowheads, line styles, colours, shades, tints and fill patterns are kept; the options are those of the TeX languages (border, magnification, fonts).
- **Magnification** scales the output (percent). It is the figure's own setting and is saved in the .fig file when you export. **Fit to Page** sets the magnification so that the figure fills the paper (a half inch of margin all round); **Figure size** shows the size of the result in the figure's unit (inches or cm) and follows the magnification.
- The panel shows only the rows a language can use, as xfig's does:
  - **Border** (points; pixels for bitmaps): blank space around the figure. xfig's `export_margin` resource is its start value.
  - **Background**: a colour behind the figure (the default is none). Grid: **Minor** and **Major** spacing draw a light grid under the figure (PostScript, PDF and bitmaps).
  - **EPS/PS preview** (EPS and PostScript): none, ASCII (EPSI), colour TIFF or monochrome TIFF.
  - **PDF page mode** (PDF): a whole page of the paper size instead of a figure-sized PDF. The paper rows below are greyed unless it is on.
  - **Paper size**, **Orientation**, **Justification**, **Pages** (single or multiple, with **Overlap**) and the **Offset** X and Y (in, cm or Fig units): the figure's page settings (PostScript and PDF page mode). They are stored in the .fig file, as in xfig.
  - **HPGL** (language HPGL/2 or IBM-GL): **Issue PCL Command to use HPGL** (fig2dev `-k`) and **Use Specified Font** (`-F`, xfig's `toggle_hpgl_pcl_switch` and `toggle_hpgl_font`) with the paper size, orientation and justification rows; the export passes the figure's paper size and orientation as xfig does (`-z`, `-P`).

  - **GIF transparent color**: none, the background, or a colour of the figure. **Image quality** (JPEG). **Resolution** and **Smoothing** (bitmaps).

- **Preview** shows a small picture of the result, made again when you change the figure or an option (not for very large figures).
- The values of these rows are kept for each language in the settings file (`[rustfig.export.<language>]`, see Preferences), never in the drawing; **Reset to xfig defaults** puts the language's values back (no border, no grid, no preview, no smoothing).
- The same values are available to scripts as `export.settings`, and `file.export` with `"panel": true` uses them.
- **Export** writes the file next to the figure with the chosen name. **Export now** (Shift+Meta+X) repeats the last export without the panel.
- **Export on save**: when checked, every Save also rewrites the exported file, so the PDF in your document is always current.
- **Embed the figure** (PDF, SVG and PNG; off at start, kept per language like the other rows): stores the figure's Fig text inside the file, so you can edit the picture you sent around. In a PDF it is an attached file `figure.fig` (PDF readers list it among the attachments); in an SVG it is a `<metadata>` element; in a PNG a compressed text chunk (`rustfig-fig`). Other programs show the file as usual and ignore the payload. **File > Open** (or dropping the file on the window) opens the embedded figure under a `.fig` name, so Save never overwrites the export. The embedded figure includes depths that are switched off and the figure's comments, so leave the option off for files you publish. `file.export` takes `"embed": true`; Export on save keeps it.

Exporting in the background

Export from the panel runs on a background thread, so the window stays responsive. A progress window shows a spinner and the seconds elapsed, with **Cancel**. Cancel is checked between steps: a driver that is already running finishes, but its result is discarded and no file is written. When the export is done an "Export finished" notice names the file and offers **Show in folder**. The commands `app.cancel_export` and `app.show_export_folder` do the same from scripts. **Export with current settings** (Shift+Alt+X, command `app.export_now`; xfig's ExportFile) exports again without opening the panel: next to the figure, with the panel's language and settings. `file.export` (commands, MCP, the command line) is synchronous. The web app downloads the file at once.

Notes

- Text uses fig2dev's font metrics; LaTeX text (the TeX flag) is typeset by TeX in the TeX languages and shown as written on the canvas.
- Pictures in the figure are embedded, linked or converted as the format allows; EPS pictures on the canvas and in bitmaps need ghostscript (optional).
- The same exports are available from the command line: `rustfig-cli convert in.fig out.pdf`, `rustfig-cli formats`, or as `fig2dev` (Automation).

Objects that share a depth are drawn in xfig's order in every export, as on the canvas: arcs, compounds, ellipses, lines, splines, then texts, each in the order of the file. (The C fig2dev leaves that order to its system's sort routine, so where objects of one depth overlap, its output can differ.)

Print

File > Print (Meta+P, Ctrl+P) opens the Print panel. RustFig makes a PDF of the page exactly as fig2dev lays it out and shows your operating system's own print dialog for it: you choose printer, copies, page ranges and the printer's options there (the macOS print panel, the Windows print dialog, the desktop's print dialog on Linux).

The panel

- **Magnification** and **Fit to Page** (a half inch of margin all round); **Figure size** follows the magnification. The magnification, **Paper size**, **Orientation**, **Justification** (centred or flush left) and **Pages** (single or multiple, with **Overlap**) are the figure's own settings, stored in the .fig file as in xfig and shared with the Export panel.
- **Background** colour, **Grid** (minor and major spacing), **Layers**: print all, or only the active ones (depths not switched off in the depth manager), optionally with the page sized by them.
- **Preview** shows the page as it prints, made again when you change an option; with several pages, step with the arrow buttons. Very large figures are not previewed.
- **Print FIGURE** opens the system's print dialog with the page as the preview shows it; if you cancel the dialog nothing is printed. **Add to Batch** keeps the figure's PDF in the batch (with the settings it has now), **Print Batch** sends all figures as ONE print job (their PDFs joined into one multi-page PDF, one print dialog; cancel keeps the batch), and **Clear Batch** forgets them. The batch lives in memory until RustFig quits.
- Shift+Meta+P (Quick print) prints with the kept settings without the panel; if figures wait in the batch it prints the batch instead, as xfig does.

The panel's values are kept in the settings file. RustFig has no printer name or job options of its own: the dialog lists the printers the system knows.

- macOS: the standard print panel (it also saves as PDF). Windows: the print dialog; the PDF then goes to the chosen printer through the program registered for PDF files, so one must be installed, and the dialog's copies and page ranges are not passed on. Linux: the xdg-desktop-portal print dialog (needs a portal with a Print backend, such as the GTK or KDE one). Web: the PDF loads in a hidden frame and the browser's print dialog opens for it; if the browser refuses that (some PDF viewers do), the PDF opens in a new tab instead, where you print it with the viewer (allow pop-ups).

Scripts

`print.settings`, `print.pdf` (the PDF as base64), `print.run`, `print.quick`, `print.batch_add`, `print.batch_clear`, `print.batch_print`. Without a front end (command line, MCP) the engine has no print system attached and printing reports that.

Preferences

Edit > Global settings holds RustFig's preferences; they are saved when you change them and loaded at every start (the panel shows where).

The upper part is xfig's panel, with the same rows. **Ok** applies every row you changed; **Cancel** (or closing the window) throws the changes away. Each row is a command (`prefs.set`, the View toggles) and is kept in the preferences:

- **Autorefresh figure**: reload the figure when its file changes on disk (View > Autorefresh mode).
- **Track mouse in rulers**: the markers in the rulers follow the pointer (resource `trackCursor`).
- **Show page borders**, **Show depth manager**, **Show info balloons** and the **balloon delay**: the View menu's switches.
- **Show line lengths**: the length triangle while drawing lines (View > Show line lengths).
- **Show vertex numbers**: numbers the points of lines, polygons and splines on the canvas (in red, from 0) (View > Show vertex numbers, resource `shownums`).
- **Allow negative coords**: when off, points are kept right of and below the origin and the view moves to the origin (`allownegcoords`).
- **Draw axis lines**: pink lines through the origin (`showaxislines`).
- **Freehand drawing resolution**: the least distance, in Fig units (1/1200 inch), between two points of a freehand stroke (`freehand_resolution`, default 25).
- **Recently used files**: 0 to 9 entries in File > Open recent (`max_recent_files`).
- **Maximum image colors**: kept for pictures (`max_image_colors`; 0 is no limit); RustFig shows pictures in full colour, so it has no effect yet.

- **Image editor**: the program Edit Image starts (`image_editor`); `%f` stands for the picture file, and without `%f` the file is added at the end. The default (`open` on macOS, `start` on Windows, `xdg-open` on Linux) uses the system's own program for the file type.
- **Spell checker**, **HTML Browser**, **PDF Viewer**: kept for the features that use them (`spellcheckcommand`, `browser`, `pdf_viewer`). `%f` stands for a file with the drawing's texts; the spell checker prints the misspelt words, one per line. The default is `aspell list < %f` (xfig's was `spell %f`, a Unix program most systems no longer have); `hunspell -l %f` works the same way. If `aspell` is not installed, `hunspell -l %f` is tried; the Edit menu's Spell check uses it (see Draw text).

The lower part is RustFig's own:

- **xfig keyboard shortcuts**: on by default; xfig's Meta and Ctrl keys (Keys).
- **Button style**: 3D (xfig's bevelled buttons) or flat.
- **Theme**: System (follows the system's light or dark setting), Light or Dark. The drawing's paper stays white in every theme.
- **UI scale**: 75 % to 200 % (a slider in steps of 5), for small or large screens. Both are kept for every start (`app.theme`, `app.ui_scale`).
- **Units and page**: opens Document settings.
- **xfig resources**: every X resource and command-line option of xfig under its xfig name (for example `startfontsize`, `startlinewidth`), with its value, default and what RustFig does with it. A value set here is kept for every start; xfig's command-line options override it for one run.
- **Use current attributes as defaults**: the indicator panel's values become the start values for new drawings.
- **Your colours**: colours added here appear in every colour menu of every drawing.

The colour theme follows the system's light or dark setting unless you choose Light or Dark above.

Document-specific settings (units, paper, orientation, comments) are saved in the drawing itself: see Edit > Document settings and Zoom and view.

LaTeX text

Under **LaTeX text** in Edit > Global settings: **Typeset LaTeX text on the canvas** (View > Typeset LaTeX text), the **TeX program** (empty: `pdflatex`, then `lualatex`, from the `PATH`; the line below names the one in use) and the **Preamble** added after `\usepackage{amsmath,amssymb,color,graphicx}`. Each field is saved, and the texts typeset again, when you leave it. See Text and pictures.

Command-line options

RustFig starts like xfig: `rustfig [options] [file]`, and every option of xfig's manual page is accepted, so `rustfig -metric drawing.fig` does what `xfig -metric drawing.fig` does. An option changes the setting for that run only; your saved preferences stay as they are. Options take their value as the next word (`-startfontsize 16`) and may be shortened as long as the start is unique among xfig's options (`-me` for `-metric`). `-help` lists them all.

- Units and page: `-metric` (or `-centimeters`), `-inches` (or `-imperial`), `-landscape`, `-portrait`, `-paper_size A4`, `-magnification 80`.
- Start attributes: `-startfontsize`, `-startlinewidth`, `-startfillstyle`, `-startgridmode 0-3`, `-startposnmode 0-4`, `-startpsFont`, `-startlatexFont`, `-latexfonts`, `-specialtext`, `-hiddentext`, `-rigidtext`, the `-startarrow...` options.
- View: `-zoom 1.5`, `-showpageborder`, `-userscale`, `-userunit`, `-geometry 1000x700+50+50` (window size and position).
- Export: `-exportLanguage pdf`, `-export_margin` (the border before any is kept), `-correct_font_size`. The Export panel's values are kept per language in the `[rustfig.export.<language>]` tables of the settings file.
- Libraries: `-library_dir FOLDER` (your library folder or a list file naming several folders, default `~/xfiglib`; Libraries), `-icon_view` / `-list_view`, `-library_icon_size N` (40 to 120).
- Files: `-write_bak` / `-nowrite_bak` (keep the old file as `NAME.bak` when saving; on by default).

- `-update file.fig ...` rewrites old files in the current Fig format (the original stays as `file.fig.bak`) and quits without opening a window.
- `-scale_factor 2` scales every file by 2 as it is read (opening, merging and `-update`): positions, sizes and arrowheads, not font sizes or line widths. With `-update` it rescales the files for good.

Coming from xfig

On the first start RustFig takes the recent files of xfig's `~/.xfigrc` into its File menu (and xfig's `max_recent_files`). On Linux, `install.sh` also installs an `xfig` command that starts RustFig, when no xfig is installed; with Homebrew, `brew install rustfig --with-xfig` does the same (it cannot be installed next to Homebrew's `xfig`). Scripts that call `xfig file.fig` then open RustFig. - Options that only make sense on X11 (`-display`, `-visual`, `-iconic`, colours and fonts of the toolkit) are accepted and ignored; RustFig prints a note to the terminal. An unknown option stops the start with a message. - Resources in `~/.Xresources` or `~/.Xdefaults` (`Fig.startfontsize: 14`) are read too, and fill in settings your preferences file does not have.

Accessibility

Every button, check box, field and ruler has a name that screen readers (VoiceOver, Narrator, Orca) announce through AccessKit; the mode and attribute buttons read as their xfig tooltips, the colour swatches by colour name. The canvas reads as a summary of the drawing, for example "Canvas: 3 objects: polyline, box, text 'Hello'" (the first eight objects, then "and N more"). A test checks that no interactive widget of the main window has an empty name; the check with a real screen reader is manual.

Automation

Everything RustFig does is a **command** with an id (for example `shape.rectangle`, `transform.rotate`, `tool.chop`). The menus, keys and mode buttons all run commands, and so can programs.

- **The command list**: Help > RustFig Reference shows every command with its id, parameters and shortcut. It is also the reference for the other routes.
- **Control channel**: start with `rustfig --control 7979 FILE.fig` and send JSON lines to `127.0.0.1:7979`: `{"id":1,"method":"engine.execute","params":{"command":"shape.rectangle","params":{"x":600,"y":600,"w` Other methods list commands, run user-interface commands such as `help.open`, report the window state or take a screenshot. `--background` opens the window without taking the keyboard focus.
- **MCP**: `rustfig-cli mcp` is a Model Context Protocol server over standard input and output, so an AI assistant can draw and edit figures with the same commands, and render the result.
- **Command line**: `rustfig-cli convert in.fig out.pdf` converts files, `rustfig-cli info FILE` describes one, and `rustfig fig2dev ARGS` (or running the program under the name `fig2dev`) is fig2dev's own command line (`fig2dev -L eps in.fig out.eps`), whose options are documented in fig2dev's manual page; `fig2dev -h` lists the general options, `fig2dev -h -L eps` adds those of one language.

Coordinates in commands are Fig units: 1200 per inch, 450 per centimetre in metric drawings, y growing downwards. Mutating commands are atomic and one undo step each.

To open a manual page from a script: `{"method":"ui.run","params":{"command":"help.open","params":{"topic":"chop.`

Finding help

Four helpers in the Help menu and on the keyboard.

Command search

Help > Command search… (`help.command_search`, `Ctrl+Shift+K`; on macOS ⌘ ⇧ K) opens a small window with a text field. Type part of a command's name, its id or its menu path ("arrange", "export", "edit.undo"): the matching commands are listed with their keys, best match first. Commands that are not available now (nothing

selected, nothing to undo) are greyed. The arrow keys move through the list, Enter (or a click) runs the highlighted command, Escape closes the window. The window finds the commands of the menus and also RustFig's own dialogs, such as Keys… and Global settings…

`Ctrl+K` is not used for the search: in xfig it opens the keyboard panel (Enter point, `app.keyboard_entry`), and RustFig keeps that key.

F1: help for what is under the pointer

Press `F1` while the pointer is over a mode button, over an attribute button in the indicator panel (each opens the page of its own topic, for example Line Style opens Line styles and Zoom opens Zoom and view), over the depth manager, the rulers or the units box (The window), or with a dialog open. The manual opens at the page that explains it. Elsewhere F1 opens the manual's first page.

Mouse buttons

Help > Mouse buttons… (`app.mouse_sheet`) lists every mode with what its left, middle and right button do, taken from the mouse-function indicator itself (so it always agrees with it), and how to press the middle and right buttons without a three-button mouse: on macOS Option+click is the middle button and Ctrl+click (or a two-finger click) the right one; on Linux and Windows Alt+click or both buttons together is the middle button; on a touch screen a long press is the right button. See also The mouse.

What's new

Help > What's new… (`app.whats_new`) lists what changed in this version. The page opens by itself once after RustFig is updated (not on the very first start); RustFig remembers the version it last showed it for in the settings file.