JaxoDraw

v. 2.1-0

User Guide

Daniele Binosi, Lukas Theussl,
Christian Kaufhold

2011-10-10

Table of Contents

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.

1 Introduction

1.1 Introduction

- Overview

It is a widely accepted convention today in the scientific community to write scientific papers using the TeX and LaTeX environments. The high quality, publication-style typesetting of LaTeX has made it now a de facto standard, to such an extent that some scientific journals only accept submission of papers in electronic form anymore. The portability goal of TeX has however the drawback that graphical representations are only possible in very rudimentary form (using the LaTeX picture environment or packages using a similar approach). Useful as they are, mostly they are too simple to draw complicated Feynman diagrams as needed in wide parts of theoretical nuclear and particle physics today (see ref. [1] for an introduction to the theory of Feynman diagrams).

This problem has led to the development of more sophisticated programs in the past. The UK List of TeX Frequently Asked Questions lists four possibilities to draw Feynman diagrams in conjunction with LaTeX: Michael Levine's `feynman` [2] bundle; Jos Vermaseren's `axodraw` [3] package which uses Postscript specials and is thus slightly less portable but much more powerful; Thorsten Ohl's `feynmf` [4] package for LaTeX2e which uses `METAFONT` (or `MetaPost`) to combine flexibility and portability; and Norman Gray's `feyn` package. These are all available from the CTAN archives.

Powerful as they are, all these methods have the common drawback that they require some 'hard-coding' from the user side in one or the other programming- or scripting language. There does not exist any graphical user interface (GUI), while modern day drawing programs (like `xfig`) do not include special options or commands that are necessary to draw Feynman diagrams with the same quality as the one achieved by TeX/LaTeX.

A rather advanced application is the `Mathematica` package `FeynArts` [5], which is also capable of producing LaTeX output (using the `feynarts` style package). However, it not only necessitates the huge, commercial program `Mathematica`, but is itself a rather sophisticated application with diagram drawing just a part of its capabilities. And it does not provide a GUI for the interactive drawing of diagrams.

There are only very few programs known to us that allow for an interactive drawing of Feynman diagrams, like A. Laina's program `Xfey` [6], I. Musatov's `FeynmanGraph`, T. Hahn and P. Lang's FeynEdit [10], or A. Santamaria's `JFeynDiagram`. However, all of them present one or the other shortcoming in terms of portability, usability, or output quality.

Our program **JaxoDraw** is an attempt to circumvent all the above mentioned drawbacks as far as possible. The main requirements that we posed on our program were that it should be easy to compile and install, easy to learn and use, produce high-quality output and be freely available (including its dependencies). Furthermore, it should be as operating system independent (or portable) and self contained as possible, with ideally no external dependencies. These requirements lead us immediately to Java as the choice of our programming language [7]. The Java technology is freely available from Sun Microsystems, it is a large-scale project that will not disappear in the near future and it is available for a variety of platforms. A working Java Runtime Environment is the only necessary requirement to run **JaxoDraw**.

As the name suggests, **JaxoDraw** was initially meant to be a graphical user interface for Jos Vermaseren's `axodraw` package, but it may be used independently of it. However, it is in conjunction with `axodraw` that **JaxoDraw** develops its main capabilities because of the possibility of combining the powers of TeX/LaTeX with a modern drawing program. The main design goal of **JaxoDraw** was convenience and ease-of-use, with respect to both compilation/installation as well as every day usage: it should be possible for anybody to draw even complicated Feynman diagrams with just a few mouse clicks, without the knowledge of any programming language. Being written in Java, **JaxoDraw** can be used on any platform where a Java Runtime Environment is installed. This makes it completely

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.

portable, however, some operations, like internal Latex compilation and Postscript preview, require the execution of external commands that are inherently system dependent and are currently only tested under certain operating systems.

This paper attempts to give a complete overview of **JaxoDraw** from installation and usage instructions to documentation issues and possible developments. The information provided in this document applies to the current version `2.0` of **JaxoDraw**, for up-to-date information on the program, please consult the **JaxoDraw** Web-page at `http://jaxodraw.sourceforge.net/`.

1.1.1 Overview

JaxoDraw is a program for drawing Feynman diagrams. It has a complete graphical user interface that allows all actions to be carried out via mouse point-and-click-and-drag operations. Graphs may be exported to Postscript / EPS format and can be saved in XML files to be used for later sessions. One of **JaxoDraw**'s main features is the possibility to create LaTeX source files to draw Feynman diagrams via `latex` and `dvips`. In fact, the original motivation for writing **JaxoDraw** was to create a graphical user interface for J. Vermaseren's `axodraw` package [3].

Some of **JaxoDraw**'s main features are:

- Platform independent
- Easy to compile/install, no external dependencies
- Complete point-and-click graphical user interface
- Pre-defined line styles for common particle representations
- Saving and reading of graphs in XML format
- Export to PostScript, EPS, JPG, PNG and Latex format
- Plugin infrastructure to add custom import and export formats
- Setting of permanent preferences
- Internationalized: currently it comes in English, German, French, Italian and Spanish (the User Guide is only available in English)
- Working with several graphs at a time
- Editing groups of objects
- Importing existing Latex files
- Grid with customizable Size
- Look-and-Feel: All Swing pluggable Look-and-Feels

The home of **JaxoDraw** sources and documentation is: `http://jaxodraw.sourceforge.net/`, this page should contain up to date information on the program. The User Guide of a previous version (`1.1`) has been published in ref. [8]. The User Guide of version `2.0` has been published in ref. [9].

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.

2 Installation

2.1 Installation

You may download **JaxoDraw** in source or binary form (xxx denotes the version number):

File	Description
jaxodraw-xxx-src.tar.gz	A gzipped tar file containing the **JaxoDraw** sources
jaxodraw-xxx-bin.tar.gz	A gzipped tar file containing a pre-compiled binary (.jar java archive)

If you have a Java Developer Kit installed on your system and you want to compile **JaxoDraw** yourself from sources, you may download the src.tar.gz file above. Check the Prerequisites section and the Compiling the sources section below.

If you have a Java Runtime Environment installed on your system (or a Developer Kit which includes the Runtime Environment), you may download the bin.tar.gz file. Check the Prerequisites section and the Running the program section below for information on how to start the program.

2.1.1 Prerequisites

- Compilation and execution of **JaxoDraw** requires an installed and configured Java environment on your system. To execute **JaxoDraw** you need a Java Runtime Environment (jre), while for compilation you need the Java Developer Kit (jdk, which includes the jre). The minimum version to compile and execute the program is Java 5. The most recent version of Java can be obtained from Oracle. Please refer to the Oracle web pages for information on how to install Java on your system. We only support the official java from Oracle. Please only report problems if you encounter them with an official java version.

  NOTE in particular that **JaxoDraw** does not run or compile with GNU's java compiler gcj which is installed by default on many Linux distributions.
- If you want to compile **JaxoDraw** yourself from sources, you need to install Apache Maven 2 (Maven 3 can be used as well except for building the site, unless the maven 3 version of the site plugin is used).
- In order to profit from the LaTeX export file format, you need (apart from a working LaTeX distribution) an axodraw4j.sty file. This can be obtained from the **JaxoDraw** download web pages, but you have to install it on your system independently of **JaxoDraw** as described in the appendix. The axodraw4j package is almost identical to J. Vermaseren's axodraw package, please refer to the axodraw user guide for documentation on the package.
- If you want to use the Postscript preview option of **JaxoDraw**, you need to specify an external Postscript viewer.

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.

2.1.2 Unpacking the archives

Any of the packages available for download are unpacked with the command

```
tar -zxf jaxodraw-xxx_zzz.tar.gz
```

under Linux or with the `unzip` utility under Windows. Here `xxx` is the version number and `zzz` is either `src` or `bin`. This will create a directory named `JaxoDraw-xxx` (the **JaxoDraw** home directory) in the current directory.

For the `src` distribution the **JaxoDraw** home directory has the following structure:

`src/`	
`assembly/`	Configuration files for the Maven assembly plugin
`changes/`	Configuration files for the Maven changes plugin
`doc/`	Documentation files
`colors/`	Documentation on the use of colors
`legal/`	License files
`main/`	Main program files
`java/`	Java source files
`resources/`	Resources (icons, User Guide, etc.)
`site/`	The **JaxoDraw** web site (XML sources)
`test/`	JUnit test files
`java/`	Java test source files
`resources/`	Test resources

The following directories are not present in the original source distribution, they may be generated by the ant build script (see below) or have to be created manually:

`target/`	Compiled class files and resources. The output of any Maven command goes here.
`target/site/apidocs/`	API specification

2.1.3 Compiling the sources

From version 2.1 on, you have to use `Apache Maven 2` to compile the program (Maven 3 can be used as well except for building the site, unless the maven 3 version of the site plugin is used). Please refer to the `Apache Maven` web pages for documentation on Maven.

To compile the sources and create the binary `.jar` file:

```
mvn package
```

To just compile the sources without creating the `.jar` file:

```
mvn compile
```

To create the distribution archives (`src.tar.gz` and `bin.tar.gz`):

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.

```
mvn -Pdist package
```

In all cases, you may use the `skipTests` profile to accelerate the build, e.g.:

```
mvn -PskipTests package
```

To create the API specification (the javadocs are created in `target/site/apidocs/`):

```
mvn javadoc:javadoc
```

To create the **JaxoDraw** web site (the site is created in `target/site/`):

```
mvn site
```

To remove all generated files:

```
mvn clean
```

2.1.4 Creating the javadoc API specification

Note: This is only needed if you are interested in the structure of the **JaxoDraw** source code, it is not required in order to run the program. We recommend to use Maven as described above.

To create the javadoc API:

```
javadoc -d target/site/apidocs/ -link . \
    -sourcepath src/main/java/ -subpackages net.sf.jaxodraw
```

You may optionally use an additional link parameter like `-link http://download.oracle.com/javase/1.5.0/docs/api/` to link against (in this case) the online java documentation from Oracle.

In any case, the command has to be run twice in order to get the cross-references right. This will create the javadoc API specification in the `target/site/apidocs/` sub-directory.

2.1.5 Running the program

If you compiled the package yourself from sources (see Compiling the sources above), you can start **JaxoDraw** with the command line

```
java -cp target/classes/ net.sf.jaxodraw.JaxoDraw
```

in the distribution home directory, or by

```
java -jar target/jaxodraw-xxx.jar
```

if you want to use the binary `.jar` file.

If you downloaded the pre-compiled binary distribution (`.bin`), just type

```
java -jar jaxodraw-xxx.jar
```

or, depending on your operating system and setup, simply double click on the `jar` file icon.

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.

JaxoDraw recognizes a small number of optional command line arguments (detailed in the Usage section), and you may also append an arbitrary number of xml files to opened at startup, so the full form of the command to start the program is

```
java -jar jaxodraw-xxx.jar [options] [file1.xml file2.xml ...]
```

You should also check the **JaxoDraw** web site if there are any binary installers available for your operating system, a `.exe` Windows self-installer, a `.dmg` disc image for Mac OS X or a `.rpm` package for Linux.

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.

3 Screen Elements

3.1 Screen elements of JaxoDraw

After first execution of the program, the user is presented the graphical user interface as shown in Fig. 1.

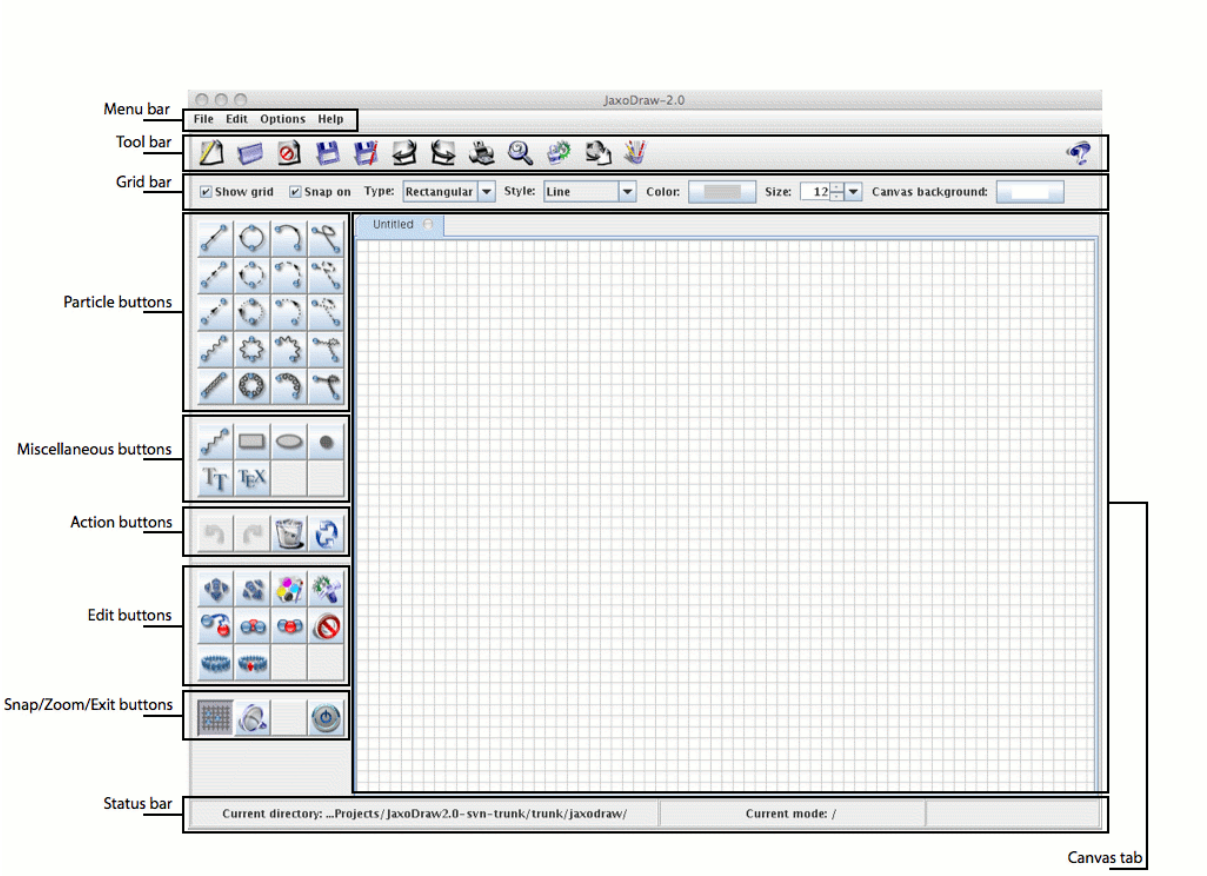

Figure 1: The graphical user interface of **JaxoDraw** at start-up.

The screen of **JaxoDraw** is divided into six main sections:

The menu bar on top

The tool bar just below the menu bar

The grid bar just below the tool bar

The button panel on the left

The status bar on bottom

The drawing area (the canvas) in the center

The tool-, grid- and status bars may optionally be switched off in the preferences dialog or under the options menu. In the following we will describe each of the above sections in greater detail.

- The menu bar

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.

- The tool bar
- The grid bar
- The button panel
- The status bar
- The canvas

3.1.1 The menu bar

The menu bar contains four main menu bar items: `File`, `Edit`, `Options` and `Help`.

File

In this menu there are the following entries:

New:

Begin a new diagram; this item contains a sub-menu with `New graph` and `New tab` entries; while the first will delete the current graph and begin a new graph on the same canvas, the latter allows to add/remove several canvas tabs (see the section on tabbing for more information).

Open:

Open an existing diagram. This pops up a file chooser dialog where the user may indicate a file (the extension must be `.xml`) that was stored in an earlier session. If the current graph is not saved, the opening will be done in a new canvas tab.

Open recent:

A menu containing the most recently modified files, so they can be rapidly accessed again.

Close:

Close the current tab (except if there is only one left).

Save:

Save the current diagram as a `.xml` file, using the last specified name. If no name was specified, a file chooser menu is popped up. The current graph is then saved in an `xml` file that may be opened in a later session.

Save As:

The same as `Save`, but always pops up a file chooser menu to save the current plot under the chosen name.

Rename tab:

Allows to change the name of the current tab.

Import:

Import an existing LaTeX file. The extension of the file must be `.tex` (see the section on importing LaTeX files for more information).

Export:

Export the current file. This entry pops up a dialog where the user may choose among several export file formats. Possible options are:

- `LaTeX->EPS`: Export the diagram to an Encapsulated Postscript (`.eps`), through a `dvips` command executed on a previous LaTeX compilation.

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.

- `LaTeX`: Export the diagram to a `.tex` text file.
- `PS Portrait`: Export the diagram to a Postscript file (`.ps`) in portrait mode.
- `PS Landscape`: Export the diagram to a Postscript file (`.ps`) in landscape mode.
- `EPS`: Export the diagram to an Encapsulated Postscript file (`.eps`).
- `JPG`: Export the diagram to a JPG/JPEG file (Joint Photographic Experts Group, `.jpg`).
- `PNG`: Export the diagram to a PNG file (Portable Network Graphics, `.png`).

If you have installed any export plugins, additional options may be displayed. Finally, notice that in the export panel there is a button that allows to preview any of the above output formats. Note that in order to preview any of the Postscript exports, you must indicate a Postscript previewer in the preferences dialog (since there is no Java internal Postscript renderer). For previewing output in text format, you may still indicate a preferred text editor but if you do not do so, a Java internal text previewer is used by default.

Print:

Print the current diagram. This opens the standard Java printer dialog where any installed and configured printers are detected automatically. Note that printing to a file should be equivalent to the corresponding `Export - Postscript` option.

Quit:

Exits **JaxoDraw**.

In between the `Print` and `Quit` entries there is a 'recent files' panel, in which the four most recently modified files are displayed, and can be rapidly accessed by a mouse click.

Edit

There are four entries that lead to an immediate action that are:

Undo:

Undo the last operation done on the canvas. The maximum number of steps that **JaxoDraw** keeps track of for undo operations can be configured in the Preferences.

Redo:

Undo the last Undo operation.

Clear:

Clear the canvas by removing all the objects; this command only removes the visible objects from the screen, it does not affect any values associated with the graph.

Paste:

Paste the objects currently in the clipboard to the current canvas (see the clipboard section for information on how to copy and paste objects from the clipboard).

Refresh:

Refresh the canvas by redrawing all the objects.

The other entries of this menu just put the program into the corresponding Edit mode, *i.e.*, little red squares are displayed on certain points of every object (for instance on the end points of lines). When the user clicks on one of these ``handles'', the corresponding edit operation is being carried out on the selected object. Notice that some of the handles of an object may not be active for a particular operation; for example, loops cannot be resized via the handle in the center. These 'inactive' handles have a different color.

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.

Move:

Click on a handle and drag the selected object to a new position.

Resize:

Click on a handle and resize the selected object by dragging. In the case the selected object is a group, just click (no drag) on one of its handles: you will be prompted a panel asking for a scale factor, that will be used to rescale the group.

Duplicate:

Click on one object's handle to duplicate the corresponding object. Drag to move the duplicated object to a new position.

Color:

Change the color of the selected object. If the latter is a filled object, then the color affected by the eventual change will be the filling one. See the section on colors for information about the available colors. Notice that the color can be also changed when editing an object: in this case for filled objects both the line and filling colors are available (with the limitation described in the section on colors).

Edit:

Edit the selected object. This operation will pop-up a panel with all the editable parameters of the chosen object (*e.g.*, its position, size, color(s), etc.), thus allowing for its fine tuning. Operations that are not possible in the interactive drawing (such as removing an arrow from a line, choosing the double line version of an object, etc.) are possible through editing the object.

Delete:

Click on a handle to delete the corresponding object.

Background:

Click on a handle to put the corresponding object in the background.

Foreground:

Click on a handle to put the corresponding object in the foreground.

Select:

Allows to select an arbitrary number of objects so that some editing operations can be carried out on them simultaneously. To select a group of objects, click on the handles of the desired objects. Selected objects will have their handles filled in light gray, clicking again on the handle will unselect the object. The select operation is finished when the right button is clicked (the middle button cancels the operation). There is an alternative (faster) way of selecting objects described in the section on grouping.

Ungroup:

When this operation is chosen, only the handles of group objects will be displayed. When clicking on one of these handles, the corresponding group breaks up in its constituent objects (which may again be groups).

Zoom:

This operation does not display any handles, instead, clicking on the canvas opens a 'dynamical zoom window' (dynamic meaning that it can be dragged over the canvas) with a

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.

magnified view of the region under the cursor. The three mouse buttons give different sizes for the zoom window.

Options

The first three entries are:

Add description:

Add a text description to a graph. This will appear as a comment in all output files.

Add LaTeX package:

Allows to specify additional packages that will be included in any LaTeX output via the `\usepackage{}` command. Notice that if you have LaTeX labels that require dedicated packages to show up correctly (*e.g.* the `\square` command from the `amssymb` package), you have to explicitly include them here, otherwise the LaTeX compilation will fail.

Move graph:

This will pop up a small panel that allows to displace the whole graph of the current tab, ie all objects of the graph are moved by the same amount.

The next entries in this menu allow the user to specify how **JaxoDraw** should look and behave.

Look And Feel:

Set the preferred Look And Feel for this session. Notice that the allowed Look And Feels will change according to the OS or the platform from where **JaxoDraw** is executed, and that there might be differences in some layouts, in particular with icons in the tool bar.

Language:

Set the preferred language for the current session. Currently supported languages are English, German, French, Italian and Spanish. Notice that there are minor bugs in some Swing internal components that do not allow full internationalization (see the bugs section). For **JaxoDraw**, this effects only some text fields in the FileChooser and in the ColorChooser dialogs.

Default mode:

Here the user can choose a default return mode, *i.e.*, after each operation, the program automatically returns to this mode.

Vertex types:

Select the type of vertex to be drawn when in `Vertex` mode. The currently available types are dot, circle cross, square, cross and triangle. Choosing one vertex type will change the icon of the Vertex button in the button panel to the corresponding vertex. This menu can be also accessed through right-click on the latter button (see the section on the button panel).

Show Toolbar:

Selects whether or not the tool bar is visible.

Show Statusbar:

Selects whether or not the status bar is visible.

Show Gridbar:

Selects whether or not the grid bar is visible.

Antialias:

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.

Enables/Disables the use of antialias (both on graphics and texts). The graphics quality is usually better with antialiasing turned on. This goes with the cost that graphics rendering may be slower on some machines and you may need to refresh the screen from time to time, especially after a number of editing operations.

Arrow:

Selects whether or not arrows should be drawn by default on all objects that support them.

Grid:

Here the user can choose wether the grid is displayed on the canvas. Note that this only draws the grid points on the canvas, you still have to activate the grid if you want to use it.

Snap:

Activates the grid.

Plugin Manager:

Pops up a dialog that displays the currently installed plugins. Clicking on one of the installed plugins (if any) shows some information about the chosen plugin, and activates the 'Uninstall' button to uninstall it. The 'Install new' button allows you to choose a new plugin to be installed. See the section on plugins.

Preferences:

Pops up a dialog where the user may choose several settings to be saved on a permanent basis. The only required settings that should be filled in are a default Postscript viewer (used for previewing the printer or direct Postscript output), and the paths to your `latex` and `dvips` executables (used for LaTeX->EPS preview). The rest are optional convenience settings (in particular you do not need a default HTML viewer or text editor since Java can render tese foramts natively).

Clicking OK will apply the specified values for the current session without saving them in the system preferences, clicking Save will save the settings without applying them to the current session, the button Clear only clears the text fields of the default previewers, Reset restores all the values to their current default settings and Cancel closes the Preferences dialog without applying any changes.

See the resources section for more information on setting preferences.

Help

In this menu one has:

About:

Provides informations about the **JaxoDraw** version you are running.

System Info:

Provides informations about the system on which you are running **JaxoDraw** (current user, operating system, Java installation). It is a good idea to furnish these informations, together with the `About` ones, when reporting a bug.

User guide:

This entry will pop up a new window with the user guide in HTML format. If a default HTML viewer has been chosen in the Preferences dialog, it will be used, otherwise a Java internal previewer is used by default.

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.

3.1.2 The tool bar

As already noted, the tool bar may be switched on and off from the `Options` menu. When switched on, it contains buttons whose actions are identical to the menu entries `New Graph`, `Open`, `Close`, `Save`, `Save As`, `Import`, `Export`, `Print`, `Paste`, and `User guide`. Furthermore, there is a 'Latex preview' button that does a `LaTeX->EPS` preview, a 'Watch file' button (see the watch file section) and a button to bring up the Preferences.

3.1.3 The grid bar

The grid bar may also be switched on and off from the `Options` menu. It allows to customize various settings of the grid, like grid type, style, size and color.

3.1.4 The button panel

The button panels are located on the left of the screen. Notice that generally, when the user pauses with the cursor over any of the program's buttons, a tool tip for the button comes up (this is also true for the tool bar entries). There are five different panels:

Particle buttons:

There is one button for each particle type: fermion (straight line), scalar (dashed line), ghost (dotted line), photon (wiggled line) and gluon (pig-tailed line); and four object types: lines, arcs, loops and beziers. When one of these buttons is clicked the program goes into the corresponding drawing mode, *i.e.*, no handles are shown on the screen and the user may click on the canvas and then drag (for lines and loops) or click again (for arcs and beziers) to start drawing the corresponding object. Fine tuning of the objects can be achieved, through the `Edit` operation.

Miscellaneous buttons:

There are buttons for drawing blobs (ellipses), boxes, vertices and zig-zag lines, as well as buttons that allow the insertion of Postscript text and LaTeX text into the graph (see the text section for information about adding text to diagrams). When one of these buttons is clicked the program goes into the corresponding drawing mode. Through right-clicking on the vertex button a pop up menu will appear, in which the user can choose the type of vertex drawn when in vertex mode.

Action buttons:

These are the buttons that lead to an immediate action, and correspond to the `Undo`, `Redo`, `Clear`, and `Refresh` entries already described.

Edit buttons:

The action of these buttons (which are `Move`, `Resize`, `Color`, `Edit`, `Duplicate`, `Foreground`, `Background`, `Delete`, `Select`, and `Ungroup`) is equivalent to the ones described in the `Edit` menu panel section. When one of these buttons is clicked the program goes into the corresponding edit mode.

Grid, Zoom and Exit buttons:

The grid button turns on the grid so that the user can choose only certain points on the canvas for placing his objects. Notice that this does not change any objects already present on the screen. The size of the grid can be specified by the user in the `Preferences` item of

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.

the `Options` menu. The zoom button switches into dynamical zoom mode to magnify some section on the canvas. The exit button quits **JaxoDraw**.

3.1.5 The status bar

The status bar may be switched on and off in the Options menu item. If it is switched on, the status bar contains three areas: one to display the current directory, one to display the current drawing mode and one that displays the current coordinates of the cursor on the canvas.

3.1.6 The canvas

This is the main drawing area. After choosing a drawing mode from the button panel, the user may draw the corresponding object by left-clicking and dragging on the canvas.

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.

4 Usage

4.1 Using JaxoDraw

4.1.1 Terminology

Object

An object is the collection of points, with optional associated values, that makes up one entity of a Feynman diagram. Examples are blobs, lines, arcs, boxes, loops, etc. The associated values can be used to change the appearance of an object, like color, line width, photon amplitude, and several other features.

Group

Different objects may be collected to form a group, such that they may be moved/copied/edited together.

Graph

A graph is the collection of objects (lines, arcs, ...), together with titles, comments, layout options, etc. drawn to display the Feynman diagram.

Parameters

Parameters are the settings of symbols, line styles, colors, fonts, etc. used to define graphs and the display of the active objects.

Handles

When the program goes into Edit mode (any mode that allows the modification of any parameters of any object), little red squares are displayed on certain points of every object (for instance on the end points of lines). When the user clicks on one of these handles, the corresponding edit operation is being carried out on the chosen object.

Canvas tab (Tab)

Canvas tabs (Tabs) are different drawing areas that allow the user to work on several graphs at a time.

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.

Clipboard

A virtual canvas tab that is used as a buffer to copy objects from one tab to another.

4.1.2 Execution

The most convenient way to start **JaxoDraw** depends on your operating system and on how you installed the program. See the installation section for generic instructions on compiling and running **JaxoDraw**.

If you want to compile **JaxoDraw** yourself from sources, you have to use the `ant` script `build.xml` to create the executable `jaxodraw.jar` file (see the section on ant). The binary distributions already contain a pre-compiled `jaxodraw.jar` file, which can then be executed with the command

```
java -jar jaxodraw-xxx.jar
```

in the current directory (`xxx` is the version number). Supposing you have a Java Runtime Environment installed and configured on your system, this will work on any platform.

4.1.3 Command line parameters

The current version JaxoDraw-2.0 supports the following command line parameters:

`--version`

Prints out the version number of **JaxoDraw**.

`--help`

Prints out some usage info on the standard output.

`--info`

Prints out some information about your system.

`--convert`

Used to convert a number of **JaxoDraw** xml files (given as parameters on the command line) to axodraw4j tex files (and vice versa) without the need of bringing up the user interface. Use like eg:
`java -jar jaxodraw-xxx.jar --convert test1.xml test2.tex`
converts `test1.xml` to `test1.tex` and `test2.tex` to `test2.xml`.

`-verbose`

Turns on verbose error messaging (default in the current version).

`-debug`

Same as `-verbose`.

`-quiet`

Turns off verbose error messaging.

`-nosplash`

Doesn't show the splash window at start up (default is to show it).

By default, all parameters starting with '`--`' do not pop up the graphical user interface of **JaxoDraw**.

Furthermore, if you have saved an XML file with a JaxoGraph in an earlier session, you may read in this graph directly on the command line by supplying the file name as an argument (the extension of the file has to be `.xml`).

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.

4.1.4 Drawing

Drawing Feynman diagrams with **JaxoDraw** is pretty easy and self-explaining. The program has been designed with the main strategy to be easy to use. In particular, if you are familiar with the `xfig` program, you will have little problems to get used to **JaxoDraw**. In general, to draw an element of a Feynman diagram, you first choose the drawing mode by clicking on the corresponding button in the button panel, and then draw the object by left mouse-clicking and dragging on the canvas. Drawn objects may then be moved/resized or edited by choosing the corresponding button in the edit button panel and then clicking on one of the handles specifying the object.

A few things to note:

- Arcs and triangular vertices are drawn in a three-click process, beziers in a four-click process. Any of the points may then be moved using the resize button.
- Any operation that changes any attribute of an object (move, resize, edit, ...) will automatically put the object in the foreground.

4.1.5 Setting resources

JaxoDraw allows the permanent setting of preferences via the `Preferences` menu. If you press the "Save" button for the first time in the `Preferences` dialog, the current settings are stored in a specific preferences file. The exact name and location of this file depends on your operating system. It is read automatically every time **JaxoDraw** is started. Normally, you should not edit this file manually, but use the `Preferences` menu dialog of the graphical user interface. See the `Preferences` menu item of the menu bar for more information on the items that may be saved on a permanent basis.

4.1.6 Colors

There are two color spaces that may be chosen in the `Preferences` menu: the `colordvi` space that restricts colors to the ones defined by the `colordvi` LaTeX class; and the 'complete' space, that lets you choose any color. If you use the complete space and do a LaTeX export, the colors will be adjusted to the 'closest' `colordvi` values. The following information applies to the `colordvi` space.

In the current version of **JaxoDraw**, the user may choose from a set of 84 colors that are presented in a convenient color chooser panel if the user clicks an object in color mode. The colors include all the 68 colors defined by the `colordvi` LaTeX class (on a standard TeTeX distribution, these may be found in `/usr/share/texmf/tex/plain/dvips/colordvi.tex`) and 16 gray scales. If figures with color are produced via the `latex -> dvips` command of **JaxoDraw**, these colors will be used as defined in the `colordvi` style file. For direct Postscript output, we have tried to reproduce as closely as possible the RGB values of these colors, but since there are no complete RGB specifications (for free), the output will not be exactly the same as in the LaTeX case.

As a reference, there are two files in the source distribution of **JaxoDraw**, that illustrate the differences. The `latexcolor.ps` file in the `JaxoDraw/doc/` directory gives a collection of all the colors present in `colordvi` as produced by `latex -> dvips`. The file `pscolor.ps` in the same directory gives the corresponding collection as produced by direct Postscript output.

Notice that when using `Blob`, `Box` or `Triangle` objects, if the fill color is a gray scale, the line color will always be black. If you want a different line color, the fill color must not be a gray scale (but it can be the `Gray` color). This is done to mimic the behavior of the `axodraw` LaTeX style.

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.

4.1.7 Text

There are two ways of entering text in **JaxoDraw**: Postscript text mode and LaTeX text mode. Even though they may be used at the same time in a graph, they will appear mutually exclusive in any derived output (a warning message is displayed if a Postscript export/preview is attempted with some LaTeX text present in the graph, and vice versa).

Postscript text mode

When entering the Postscript text mode, the user may enter a text string that will appear directly on the screen and in any direct Postscript output (i.e., also in any printer output). It will not appear in any output created via `latex -> dvips`. In edit mode, the user may choose the text size and font of the text object. A set of Greek characters is available via a syntax that is derived from the corresponding LaTeX commands:

α	\alpl	λ	\laml	υ	\upsi	Λ	\Lambda
β	\beta	μ	\mu	ϕ	\phi	Ξ	\Xi
γ	\gamr	ν	\nu	χ	\chi	Π	\Pi
δ	\dell	ξ	\xi	ψ	\psi	Σ	\Sigma
ϵ	\epsi	o	o	ω	\omeg	Φ	\Phi
ζ	\zeta	π	\pi	ϑ	\varl	Ψ	\Psi
η	\eta	ρ	\rho	φ	\varp	Ω	\Omega
θ	\thel	ς	\vars	Γ	\Gamr		
ι	\iota	σ	\sigr	Δ	\Dell		
κ	\kapp	τ	\tau	Θ	\Thel		

Note that no $ signs are necessary for these commands (any $ signs will appear verbatim on the screen). If the user enters a string starting with a " \" that is not recognized as a valid Greek letter, it will be replaced by a question mark "?". Super- and subscripts are available in some rudimentary form via a syntax that is again derived from the corresponding LaTeX syntax, *i.e.*, for A^{μ}_{ν} you would type `A^{\mu}_{\nu}`. Note again that no $ signs are necessary and that the brackets are always required even if there is just one character as an argument. Curly brackets are implemented via a syntax that is again inspired by LaTeX: `\{` and `\}`.

LaTeX text mode

When entering the LaTeX text mode, the user may enter a text string that will appear only in the LaTeX output file and any files created from it via `latex -> dvips`. Like that all the commands known to LaTeX in math mode are available to the user. Notice that if a math symbol requires a dedicated LaTeX package to be displayed (such as the amssymb

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.

package, etc.), the user has to include it explicitly through the ``Add LaTeX package'' entry of the Options menu (otherwise the LaTeX compilation will fail producing an ``Undefined control sequence'' error). The position of the text will be marked on the screen by an icon that identifies it as a LaTeX text object. This icon does not appear in direct Postscript or printing output. If the icon is rolled over by the cursor, the corresponding LaTeX text is displayed in a pop-up window on top of the canvas. Note that the LaTeX text string will automatically be put between $ signs, so the text will always be in LaTeX math mode. If you want a normal font in LaTeX text mode, you should use `{\rm }`. Note also that your input here is the only possible source of errors in your LaTeX source code. If you get any LaTeX compilation errors, check your LaTeX text objects first. In edit mode, the user may choose the LaTeX font size and the alignment with respect to the current position of the text object. You can also specify rotation angles for LaTeX texts (rotations are implemented using the LaTeX `pstricks` package).

4.1.8 Grouping

JaxoDraw allows the grouping of different objects into a single one so that they can be moved, resized and edited at the same time. Grouping groups is also allowed, and the hierarchy is always respected. To build a group of objects, first select the desired objects, then right click and select 'Group' from the drop-down menu.

Apart from grouping objects using the `Select` mode, one can use also the ``faint box'' method. At any time, it is possible to click the right mouse button on the canvas and by dragging, there will appear a faint gray box. Once the button is released, all objects entirely located inside this box will be grouped together. This method may be used also when the program is not in select mode, *i.e.*, without having the `Select` button pressed.

4.1.9 Tabbing

It is possible to work with several graphs at a time by using the tabs of **JaxoDraw**. At the first start-up, the program just contains one tab ("Untitled"), you may add tabs with the `New tab` entry in the `File` menu and tabs may be closed with the `Close` entry. These operation are also presented in a pop-up menu if the user right-clicks on a tab. Notice that new tabs sharing the same name with existing opened tabs, will be automatically numbered. Objects from one graph may be copied to other graphs, using a copy and paste procedure treated in the next section.

4.1.10 Using the clipboard

It is possible to copy objects from one canvas to another one, by first copying them to the clipboard, and pasting the clipboard content into the desired canvas tab afterwards. To copy objects to the clipboard, select them (using the Select button or the 'faint box' method) and choose 'Edit - Copy' (notice that the previous clipboard content will be lost). Next click on the canvas tab were you want to paste the clipboard content and then click on the paste icon of the tool bar (alternatively you can use the `Paste` item of the `Option` menu, or right click on the canvas tab and choose the `Paste` entry from the pop-up menu).

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.

4.1.11 Importing LaTeX files

JaxoDraw allows to import existing LaTeX files, even if they were not originally created by **JaxoDraw**. However, only commands that are known to **JaxoDraw** are actually recognized (that are commands that are used when exporting to a LaTeX file), unknown commands will be silently ignored. The commands known to **JaxoDraw** are:

`\ArrowArc`	`\ArrowArcn`	`\ArrowLine`	`\CArc`
`\CBox`	`\CCirc`	`\COval`	`\CTri`
`\DashArrowArc`	`\DashArrowArcn`	`\DashArrowLine`	`\DashCArc`
`\DashLine`	`\GBox`	`\GCirc`	`\GlueArc`
`\Gluon`	`\GOval`	`\GTri`	`\Line`
`\PhotonArc`	`\Photon`	`\SetColor`	`\SetWidth`
`\Text`	`\usepackage`	`\Vertex`	`\ZigZag`

(see the `axodraw` user guide for documentation on these commands [3]). In addition, the following LaTeX commands are required:

`\documentclass`	`\begin{document}`	`\begin{picture}`

If these ones are not found in the file to be imported, the import process is abandoned with a warning message.

Due to the specific algorithm internally used by **JaxoDraw** to draw triangles, the commands `\GTri` and `\CTri` will work only when the imported LaTeX file has been generated from **JaxoDraw** itself. Otherwise they will be ignored. Moreover, because of rounding errors in the export/import routines, sometimes the imported graphs may be slightly different from the original ones (possible differences being the number of wiggles/windings of photon/gluon objects and the position of objects). These differences are usually of one unit. Lines that start with `%%\JaxoComment` will be read in as a description of the graph while lines of the form `%%\JaxoScale{scale}` allow to read in a floating point value scale factor which is used internally by **JaxoDraw** to convert Java coordinates into LaTeX coordinates. If it is not given, the scale factor defaults to 1. Finally lines of the form `%% \JaxoDrawID:` are identifiers used by **JaxoDraw** to reconstruct some objects.

4.1.12 Watch file mode

The WatchFile mode is switched on and off via a button in the tool bar. When switched on, a postscript preview will only export the current graph into a temporary file, but will not open it with the current default postscript viewer. This has the advantage that if you have already a postscript viewer window open, no new window will pop up, you can just redraw the graph in the old window. Note that for this to work, you need to switch on the WatchFile mode **after** your first preview operation, otherwise no postscript window will be opened in the first place.

4.1.13 Plugins

Version 2.0 of **JaxoDraw** introduced a plugin architecture, in order to draw some functionality out of the **JaxoDraw** core. This makes it easy to use optional features, like export to uncommon formats, while keeping the size of the main program at a minimum.

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.

Plugins are installed (and un-installed) using the Plugin Manager panel accessible from the Options menu. Once a plugin is installed, **JaxoDraw** will automatically recognize it at start-up and the corresponding functionality will be available for the session (eg export to some other file format).

Please check the **JaxoDraw** web site for a list of available plugins, at the time of this writing, there were plugins available for export to PDF and SVG format.

Note that it is also possible to write your own plugins for new export/import formats. A tutorial for doing that is available on the **JaxoDraw** web site.

4.1.14 Comments and bug reports

Please send your comments, questions or bug reports to the `jaxodraw-discuss` mailing list:

jaxodraw-discuss@lists.sourceforge.net

When reporting bugs, you should be as specific as possible about the problem so that we can easily reproduce it. Include some information about your operating system, the Java version and the version of **JaxoDraw** that you are using. Include for instance the output of the

```
java -jar jaxodraw-xxx.jar --info
```

and

```
java -jar jaxodraw-xxx.jar --version
```

commands (this information is also available under the Help menu of the graphical user interface). You should also try to run **JaxoDraw** in debug mode:

```
java -jar jaxodraw-xxx.jar -debug
```

and check for any relevant information.

If you are having problems with the LaTeX compilation process, also include detailed information about your LaTeX distribution, the version of `dvips`, your Postscript viewer and any other information that may be relevant.

We will try to make all messages of general interest available on our Web-site

`http://jaxodraw.sourceforge.net/`

Please check these pages and also the FAQ and Known problems sections of this document before reporting any bugs.

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.

5 Known Problems

5.1 Known problems and limitations

This section gives a list of bugs and limitations that were known at the time of first publication of JaxoDraw-2.0. Please check the Bugs section of our Web page for an updated version of this document. Note that not all points are necessarily real bugs, we regard this just as a collection of features that do not work exactly the way we would like to.

5.1.1 Bugs

- When using IBM's Runtime Environment, the program may be executed and works fine for most parts but presents some peculiarities: the layout of pop-up windows is not always the same and XML output serializes the bounding boxes of objects that are explicitly marked as transient in the source code. This has been reported to us for version number `1.4.1` of IBM's SDK. This appears to be an incompatibility between SUN's and IBM's Runtime Environments. On the other hand, the program compiles fine with IBM's `jikes` compiler (tested with version `1.13`).
- Some Swing internal components do not (yet) allow full internationalization (see this URL for some explanation, or check Sun's bugs pages, bug # 4195173). For **JaxoDraw**, this effects only some text fields in the FileChooser, ColorChooser and Print dialogs, where the text will always be given in English.
- If a LaTeX text object is present at the edge of a graph then the bounding box of a "`LaTeX -> EPS`" export is most probably not correct. This manifests itself by a "broken" text, *i.e.*, a thin white line is crossing the text. Basically, there is no way for us to estimate the extension of a text after latex compilation, so the only workaround is to adjust the bounding box by hand.
- When drawing a line or loop and releasing the mouse outside the drawing area, the object will still be drawn correctly. However, the same does not work for arcs and beziers: clicking once outside the drawing area will make the object disappear. This is a fundamental Java limitation and cannot be fixed in general.
- On windows, plugins cannot be un-installed from the Plugin Manager. They get removed from the current session but remain in the plugin cache, so next time JaxoDraw is started they get loaded again. The plugin jar has to be removed manually from the plugin installation folder to uninstall the plugin permanently.
- Running JaxoDraw-2.0-0 on a 64bit java installation, the program first executes fine and works for most parts, but trying to draw a gluon line gives a crash dump. This is evidently a bug in the java virtual machine 64bit version, so there is nothing we can do about it. The only workaround is to install a 32bit version and use this to run JaxoDraw. See bug #2561249 for more details.

5.1.2 Wish list

Any feature request or suggestions should be filed at our SourceForge Request page.

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.

6 Documentation

6.1 Documentation

This section gives a list of hints and tricks as well as a list of frequently asked questions that were known at the time of first publication of **JaxoDraw-2.0**. Please check the `Reference` section of our Web page for an updated version of this document.

6.1.1 Tips and tricks

- Note that Postscript files produced by `Export - EPS` are considerably larger in size than the same files generated via `latex -> dvips`. This is due to the way how Java handles the Postscript printing internally. Keep this in mind if you want to include a bunch of small figures in your document: it is then probably preferable to use the LaTeX output.
- It is a good idea to use the refresh button from time to time, especially if there are a lot of objects on the screen and if you are using antialiasing.
- Instead of producing EPS figures and including them into your document, you may as well cut-and-paste the LaTeX output of **JaxoDraw** into your own LaTeX source code. Like that you may avoid the proliferation of numerous Postscript files to be distributed with your source code. Note however that you will have to include axodraw4j in the header of your LaTeX file (put `\usepackage{axodraw4j}` somewhere before `\begin{document}`) and you will probably have to distribute the `axodraw4j.sty` file along with your source code because it is not part of any standard LaTeX distribution.
- To add multiple arrows to objects (such as loops, for example), or to draw arrows on objects that do not support them (photon and gluon objects in particular), use fermion lines with very small length (5 points should do the job). Use the edit menu, to give the arrow the direction you need and then move it to the wanted location.
- For Mac OS X users: note that there are some Mac specific quirks (mostly due to Apple's Java implementation), please read the file `README_Mac` in the `MacOSX` subdirectory of the source distribution for some further information.

6.1.2 FAQ

Why does my text not appear in the LaTeX output?

Postscript text mode and LaTeX text mode appear mutually exclusive in any derived output (see the text section). If you want your text to appear in the LaTeX output, use the TeX icon to enter your text.

Why is there this funny fcolorbox command in the Latex text file generated by JaxoDraw?

The `\fcolorbox` command was introduced because otherwise `dvips` does not recognize the bounding box coordinates correctly for the conversion to EPS. This is a workaround that we used ourselves when working with `axodraw.sty`, if you cut and paste the LaTeX output into your own documents, or if you don't want an EPS, you may erase the `\fcolorbox` command.

When I try to do a Latex-EPS export or preview, I get the following error message: java.io.FileNotFoundException: Jaxo_tmp.tex (Permission denied), and no files are created. What does that mean?

Apparently your file permissions are set such that you don't have write access to the **JaxoDraw** home directory. `Jaxo_tmp.tex` is a temporary file that is created during the

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.

LaTeX compilation process (among others). To check that, go one directory up from the **JaxoDraw** home directory and type:

```
ls -dl JaxoDraw-$VERSION
```

(replace $VERSION by the version of **JaxoDraw** that you have installed). This should give you a '`drwx`' in the beginning of the line. If this is not the case, type

```
chmod u+rwx JaxoDraw-$VERSION
```

and try again.

I compiled JaxoDraw myself from sources. If I start the program as described in the user guide everything is fine, but when I try to run it from a different directory than the home directory, I get a "NoClassDefFoundError". How can I run JaxoDraw from any location I like?

If you are not using the `jar` executable and would like to run **JaxoDraw** from a different location than the program's home directory, you have to tell the java interpreter where it can find the executable `.class` files. This is done with the `-classpath` (or `-cp`) option.

Example: Suppose you have put the **JaxoDraw** home directory `JaxoDraw-1.2` into your home directory `$HOME`. Going into the `JaxoDraw-1.2/` directory and typing

```
java JaxoDraw/JaxoDraw
```

will work as expected, but typing

```
java JaxoDraw-1.2/JaxoDraw/JaxoDraw
```

gives you the above error. What you need to type instead is

```
java -cp JaxoDraw-1.2 JaxoDraw/JaxoDraw
```

(note the space between the classpath variable and the executable). An alternative is to create the `jar` executable (see the installation section) which may be executed from everywhere with the `-jar` option:

```
java -jar JaxoDraw-1.2/JaxoDraw.jar
```

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.

7 axodraw4j

7.1 Installing axodraw4j.sty

From version 2.0 on, **JaxoDraw** uses its own LaTeX style file `axodraw4j`. This package is derived from J. Vermaseren's `axodraw` style file, which was used in earlier versions by **JaxoDraw**. `axodraw4j` is supposed to be completely backward compatible with `axodraw`, ie any graphs written for `axodraw` will also be processed correctly by `axodraw4j`. There are just a few additions for the drawing of Bezier curves and the resizing of arrows.

We do not want to promote `axodraw4j` for general usage yet, because it is still in an incomplete state at this point. However, you need to install `axodraw4j` if you want to use the LaTeX-EPS export feature of **JaxoDraw**. For documentation on the package please consult the original `axodraw` user guide by J. Vermaseren.

Note: It is not necessary to install `axodraw4j` in order to run **JaxoDraw**. You will just not be able to use the LaTeX/LaTeX - EPS export options but you may still generate direct Postscript output of your Feynman diagrams.

In the current version of **JaxoDraw** we distribute a modified version of J. Vermaseren's `axodraw` package (with kind permission of the author) in the distribution home directory. You have to install `axodraw4j.sty` such that LaTeX can find it on your system. This appendix describes how to do that.

First get the `axodraw4j.sty` file from the **JaxoDraw** distribution home directory. `axodraw4j.sty` is largely backward compatible with `axodraw` i.e. you will be able to compile all you old tex files with the new style file.

Please refer to the `axodraw` user guide for a detailed documentation of the package. The documentation is available from the axodraw web site.

The documentation for `axodraw4j.sty` is available from the **JaxoDraw** web site. We shall only outline here how you make `axodraw` available on your system and how you use it with the LaTeX output from **JaxoDraw**.

7.1.1 Linux instructions

For installation, you have two options: if you intend to use `axodraw4j` just for yourself on a multi-user platform, you may install it locally; if you want to make it available for all the users on the system, you should do a global installation. Note that you will need root privileges for a global installation.

Installing axodraw4j locally

The easiest way to use axodraw4j is to put the `axodraw4j.sty` file in the same directory as your LaTeX source file (like the one produced by **JaxoDraw** via the Export -> LaTeX command). This is usually the same directory where you execute the program (but note that you cannot execute **JaxoDraw** from a different directory in this case). You can then run `latex` on your source file as usual, the style file will be found because the current directory is by default in the `TEXINPUTS` search path.

An alternative (better) way is to put the style file in a special directory (this is particularly useful if you have several style files which are not part of your standard LaTeX distribution). Let's say you put it into the directory `latex/` in your home directory. You then have to set the `TEXINPUTS` variable to this path. If you are using bash, just do

```
export TEXINPUTS=$HOME/latex//:
```

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.

(the `//:` at the end tells LaTeX also to look into sub-directories of this path). If you want to make that permanent, you should put this line into your `.bash_profile` file. You will then be able to start **JaxoDraw** from any directory, independently of the location of `axodraw4j.sty`.

Installing `axodraw4j` system-wide

Installing `axodraw4j` system wide is very easy. Just put the style file somewhere in the global search path of your LaTeX distribution (for TeTeX, typically `/usr/share/texmf/tex/latex/misc/`) and update the TEX database with

```
mktexlsr
```

(you will have to be root for doing this). Note that there is also an rpm package of `axodraw4j` available on the **JaxoDraw** downloads page, which may be used on Redhat-like systems.

Finally, always check the **JaxoDraw** web site for eventual binary installers.

7.1.2 Windows instructions

Under Windows, you have to do basically the same as under Linux. First put your `axodraw4j.sty` file into the MikTeX search tree (somewhere under `/texmf/tex/latex/`, replace the slash by a backslash!) and update your database with the command `mktexlsr` in the `/texmf/tex/miktex/bin` directory.

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.

8 Miscalleneous

8.1 Miscalleneous

8.1.1 History

Date	Release
1.9.2003	Released JaxoDraw-1.0
10.9.2003	Released JaxoDraw-1.0-1 (Bugfix)
15.3.2004	Released JaxoDraw-1.1-0
5.7.2004	Released JaxoDraw-1.2-0
20.6.2005	Released JaxoDraw-1.3-0
29.6.2005	Released JaxoDraw-1.3-1 (Bugfix)
24.3.2006	Released JaxoDraw-1.3-2 (Bugfix)
20.11.2008	Released JaxoDraw-2.0-0
26.07.2009	Released JaxoDraw-2.0-1 (Bugfix)

See the file `CHANGELOG` in the distribution home directory for a complete list of changes in a particular release.

8.1.2 License

```
Copyright (C) 2003-2011 Daniele Binosi and Lukas Theussl
Copyright (C) 2007-2008 Christian Kaufhold

JaxoDraw is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

JaxoDraw is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA

A copy of the GNU General Public License can be found in the
file GNU-LICENSE that is distributed along with this program.
```

8.1.3 Credits

We are grateful to Prof. Arcadi Santamaria for numerous helpful remarks and moral support during the development of **JaxoDraw**.

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.

We also acknowledge Prof. Jos Vermaseren for his kind permission to use and distribute a modified version of his `axodraw.sty` style file along with earlier versions of **JaxoDraw**.

Finally, many thanks to all the people who tested **JaxoDraw** and sent us bug reports and suggestions.

Prev Home Next

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.

9 Bibliography

9.1 Bibliography

1. M. Veltman.
   Diagrammatica : The Path to Feynman Diagrams.
   Cambridge University Press, 1994.
2. M. J. S. Levine.
   A LaTeX graphics routine for drawing Feynman diagrams.
   Comput. Phys. Commun., 58:181-198, 1990.
3. J. A. M. Vermaseren.
   Axodraw.
   Comput. Phys. Commun., 83:45-58, 1994.
4. T. Ohl.
   Drawing Feynman diagrams with FX340-1 and Metafont.
   Comput. Phys. Commun., 90:340-354, 1995.
5. T. Hahn.
   Generating Feynman diagrams and amplitudes with FeynArts 3.
   Comput. Phys. Commun., 140:418-431, 2001.
6. A. Laina.
   Xfey, a Feynman diagram editor.
   Comput. Phys. Commun., 111:217-242, 1998.
7. K. Arnold, J. Gosling, and D. Holmes.
   The Java(TM) Programming Language (3rd Edition).
   Addison-Wesley, 2000.
8. D. Binosi and L. Theußl.
   JaxoDraw: A graphical user interface for drawing Feynman diagrams.
   Comput. Phys. Commun., 161:76-86, 2004.
9. D. Binosi, J. Collins, C. Kaufhold and L. Theussl.
   JaxoDraw: A graphical user interface for drawing Feynman diagrams. Version 2.0 release notes
   Comput. Phys. Commun., 180:1709-1715, 2009.
10. T. Hahn and P. Lang
    FeynEdit - a tool for drawing Feynman diagrams
    Comput. Phys. Commun., 179:931-935 , 2008.

©2011, Daniele Binosi, Lukas Theussl, Christian Kaufhold • ALL RIGHTS RESERVED.