Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Physics Communications 161 (2004) 76–86
Computer Physics Communications
www.elsevier.com/locate/cpc

# JaxoDraw: A graphical user interface for drawing Feynman diagrams ☆

D. Binosi [1], L. Theußl [*,2]

*Departamento de Física Teórica, Universidad de Valencia, E-46100 Burjassot (Valencia), Spain*

Received 23 April 2004; accepted 3 May 2004

**Abstract**

`JaxoDraw` is a Feynman graph plotting tool written in Java. It has a complete graphical user interface that allows all actions to be carried out via mouse click-and-drag operations in a WYSIWYG fashion. 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 code that may be used to generate graphics output, thus combining the powers of TeX/LaTeX with those of a modern day drawing program. With `JaxoDraw` it becomes possible to draw even complicated Feynman diagrams with just a few mouse clicks, without the knowledge of any programming language.

**Program summary**

*Title of program:* JaxoDraw
*Catalogue identifier:* ADUA
*Program summary URL:* http://cpc.cs.qub.ac.uk/summaries/ADUA
*Program obtainable from:* CPC Program Library, Queen's University of Belfast, N. Ireland
*Distribution format:* tar gzip file
*Operating system:* Any Java-enabled platform, tested on Linux, Windows ME, XP, Mac OS X
*Programming language used:* Java
*License:* GPL
*Nature of problem:* Existing methods for drawing Feynman diagrams usually require some 'hard-coding' in one or the other programming or scripting language. It is not very convenient and often time consuming, to generate relatively simple diagrams.
*Method of solution:* A program is provided that allows for the interactive drawing of Feynman diagrams with a graphical user interface. The program is easy to learn and use, produces high quality output in several formats and runs on any operating system where a Java Runtime Environment is available.
*Number of bytes in distributed program, including test data:* 2 117 863

☆ This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author. See address below.
*E-mail addresses:* binosi@ect.it (D. Binosi), theussl@triumf.ca (L. Theußl).
[1] Present address: ECT*, Villa Tambosi, Strada delle Tabarelle 286, I-38050 Villazzano (Trento), Italy.
[2] Present address: TRIUMF, 4004 Wesbrook Mall, BC, Vancouver, Canada, V6T 2A3.

0010-4655/$ – see front matter 
doi:10.1016/j.cpc.2004.05.001

*Number of lines in distributed program, including test data:* 60 000
*Restrictions:* Certain operations (like internal latex compilation, Postscript preview) require the execution of external commands that might not work on untested operating systems.
*Typical running time:* As an interactive program, the running time depends on the complexity of the diagram to be drawn.


*PACS:* 01.30.Rr; 03.70.+k; 07.05.Bx

*Keywords:* Feynman diagrams; LaTeX; Java; GUI

## 1. Introduction

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[3] 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 LaTeX2$\varepsilon$ which uses METAFONT (or MetaPost) to combine flexibility and portability; and Norman Gray's `feyn`[4] package. These are all available from the CTAN[5] 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 (free) drawing programs (like `xfig`[6]) 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.sty` 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 arbitrary 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`,[7] KiwiSoft's (commercial) `FeynDraw`,[8] or A. Santamaria's `JFeynDiagram`.[9] 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,[10] it

[3] http://www.tex.ac.uk/cgi-bin/texfaq2html?label=drawFeyn.
[4] http://www.astro.gla.ac.uk/users/norman/distrib/latex/.
[5] http://www.ctan.org/.
[6] http://www.xfig.org/.
[7] http://www.physics.odu.edu/~musatov/.
[8] http://www.feyndraw.com/.
[9] http://fisteo12.ific.uv.es/~santamar/jfeyndiagram/jfeyndiagram.html.
[10] http://java.sun.com/.

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 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.

The present paper is organized as follows: Section 2 gives a brief overview of the main features of JaxoDraw while Section 3 contains a detailed description of the graphical user interface and usage instructions. A few examples of Feynman diagrams created with JaxoDraw are given in Section 4, in order to illustrate the capabilities of the program, and Section 5 provides some additional information.

The information provided in this document applies to the current version 1.1-0 of JaxoDraw, for up-to-date information on the program, like updates and new developments, please consult the JaxoDraw Web-page at http://altair.ific.uv.es/~JaxoDraw/home.html.

## 2. Program summary

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 files which make use of J. Vermaseren's axodraw package [3] to create a Feynman diagram via latex and dvips. In fact, the main motivation for writing JaxoDraw was to create a graphical user interface for the axodraw package.

Some of JaxoDraw's main features are:

- Completely portable.
- Easy to compile/install, no external dependencies.
- Complete point-and-click graphical user interface.
- Saving and reading of graphs in XML format.
- Export to Postscript, EPS and Latex format.
- 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.

Note that a rather detailed user guide in HTML format is included in the source distribution (in the bin/JaxoDraw/doc/usrGuide/usrGuide sub-directory). This user guide is accessible from within the program and is also available in Postscript and other formats from Ref. [8].

## 3. Description

### 3.1. Prerequisites

As noted in the introduction, compilation and execution of JaxoDraw requires an installed and configured Java environment. The program was written with the SUN J2SDK developer kit, version 1.4.1_01, using the SUN javac compiler. The Java Developer Kit and Runtime Environment are free software and may be obtained from SUN's web pages.[11] Please refer to SUN's Java documentation for information on how to install and configure the Java environment on your system.

[11] http://java.sun.com.

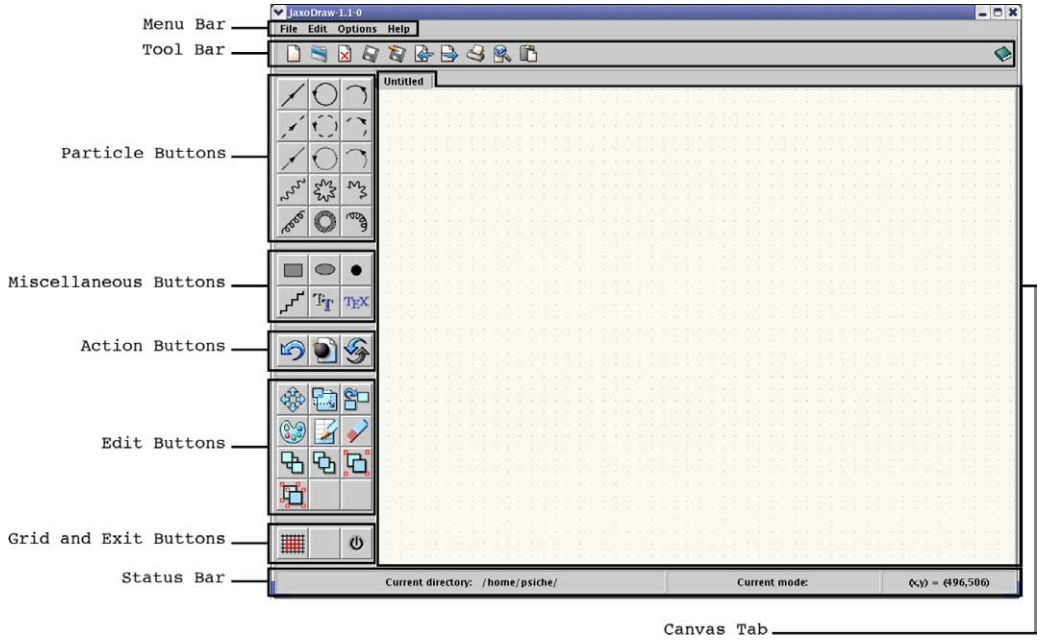


Fig. 1. The graphical user interface of `JaxoDraw` at start-up.

In order to profit from the LaTeX export file format, one needs J. Vermaseren's `axodraw`[12] package. This is distributed along with `JaxoDraw` (with kind permission of the author) but it has to be installed independently of `JaxoDraw`.

For using the Postscript preview option of `JaxoDraw`, one has to specify an external Postscript viewer.

### 3.2. Execution

The file README in the distribution home directory gives some instructions on how to compile and run the program on different platforms. In general, using Sun's `javac` compiler, the source is compiled with

```
javac -d bin/ src/JaxoDraw/*.java
```

and the program may then be started with the command

```
java -cp bin/ JaxoDraw/JaxoDraw
```

in the distribution home directory. Supposing that Java is installed on the system, this will work on any platform. The current version `1.1-0` supports a number of command line parameters that may be viewed with the –`help` option. Furthermore, if a graph was saved in an earlier session, it may be read in directly on the command line by supplying the file name as an argument (the extension of the file has to be `.xml`).

### 3.3. The layout of the graphical user interface

After first execution of the program, the user is presented the graphical user interface as shown in Fig. 1. The screen of `JaxoDraw` is divided into five main sections: the menu bar on top, the tool bar just below the menu bar, the button panel on the left, the status bar at the bottom and the drawing area (the canvas) in the center. In the following we will describe each of the above sections in greater detail.

#### 3.3.1. The menu bar

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

[12] http://www.nikhef.nl/~form/FORMdistribution/axodraw/.

*File.* Apart from the standard entries (`New`, `Open`, `Close`, `Save`, `Save as`, `Export`, `Print`, `Quit`), there is an `Import` item, that allows to import existing LaTeX files (see Section 3.6). The `New` item contains a sub-menu with `New graph` and `New tab` entries, the latter allows to add/remove several tabs to the canvas (see Section 3.4.4 for information on tabbing). The `Export` menu item pops up a dialog where different export formats may be chosen which may also be previewed by clicking the `Preview` button (note that for Postscript previews, you need to specify an external Postscript viewer in the `Preferences` menu).

*Edit.* There are four entries that lead to an immediate action (`Undo`, `Clear`, `Refresh` and `Paste`) while the others (`Move`, `Resize`, `Copy`, `Color`, `Edit`, `Delete`, `Background`, `Foreground`, `Group`, `Ungroup`) 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 and middle points of lines). When the user clicks on one of these handles, the corresponding edit operation is being carried out on the chosen object. See Section 3.4.3 for information on Group/Ungroup, and Section 3.4.5 for information on copying to/pasting from the clipboard.

*Options.* There is first an item `Add description` that allows the addition of a text description to a graph (this will appear as a comment in all output files). The `Add LaTeX package` entry allows the specification of additional packages that will be included in any LaTeX output via the `\usepackage{}` command. Then the user may choose a Look and Feel and the language for the current session, the type of vertex to be drawn when in Vertex mode, whether the tool- and status-bar are visible, whether to use antialiasing and whether arrows should be drawn by default on all objects that support them. Finally, the Preferences item pops up a dialog where the user may choose several settings to be saved on a permanent basis. This is detailed in Section 3.5.

*Help.* The `About` and `System info` items will give some information about the current version of `JaxoDraw` and the current system, respectively, while the User guide entry will pop up a new window with the user guide in HTML format.

*3.3.2. The tool bar*

The tool bar may be switched on and off in the Options menu. If it is switched on, the tool bar contains buttons whose action is identical to the corresponding menu entries: `New Graph`, `Open`, `Close`, `Save`, `Save As`, `Import`, `Export`, `Print`, `Paste`, and `User guide`. Furthermore, there is a "quick-build" icon that does a `Latex -> EPS` preview.

*3.3.3. The button panel*

The button panel on the left of the screen is divided into the following subsections:

*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 the three object types: lines, arcs and loops. 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 to start drawing the corresponding object. A double line version of all the objects (with a customizable separation between the two lines) is also available, through editing the objects.

*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. 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 the vertex mode (the currently available types are dot, circle cross, square, cross and triangle).

*Action buttons.* These are the buttons that lead to an immediate action: `Undo` and `Clear` have the same effect as the corresponding entries in the `Edit` menu, while the `Refresh` button leads to a redrawing of the screen. This is especially useful if antialising is used.

*Edit buttons.* There are a number of buttons (`Move`, `Resize`, `Copy`, `Color`, `Edit`, `Delete`, `Foreground`, `Background`, `Group` and `Ungroup`)

whose actions are self-explanatory. When one of these buttons is clicked the program goes into the corresponding edit mode. In particular the `Edit` button 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 their fine tuning. Operations that are not possible by interactive drawing (such as removing an arrow from a line, changing the angular extension of an arc, or choosing the double line version of a drawn object) may be carried out through editing the object.

*Grid and exit button.* The grid button turns on the grid so that the user can choose only certain points on the canvas for placing his objects. This does not change any objects already present on the screen. The exit button quits `JaxoDraw`.

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).

### *3.3.4. The status bar*

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

### *3.3.5. 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.

## *3.4. 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 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.

### *3.4.1. Colors*

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.

### *3.4.2. 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, color, and the font of the text object. A set of Greek characters is available via a syntax that is derived from the corresponding LaTeX commands:

| | | | |
|---|---|---|---|
| $\alpha$ | \alpha | $\tau$ | \tau |
| $\beta$ | \beta | $\upsilon$ | \upsilon |
| $\gamma$ | \gamma | $\phi$ | \phi |
| $\delta$ | \delta | $\chi$ | \chi |
| $\epsilon$ | \epsilon | $\psi$ | \psi |
| $\zeta$ | \zeta | $\omega$ | \omega |
| $\eta$ | \eta | $\vartheta$ | \vartheta |
| $\theta$ | \theta | $\varphi$ | \varphi |
| $\iota$ | \iota | $\Gamma$ | \Gamma |
| $\kappa$ | \kappa | $\Delta$ | \Delta |
| $\lambda$ | \lambda | $\Theta$ | \Theta |
| $\mu$ | \mu | $\Lambda$ | \Lambda |
| $\nu$ | \nu | $\Xi$ | \Xi |
| $\xi$ | \xi | $\Pi$ | \Pi |
| $o$ | o | $\Sigma$ | \Sigma |
| $\pi$ | \pi | $\Phi$ | \Phi |
| $\rho$ | \rho | $\Psi$ | \Psi |
| $\varsigma$ | \varsigma | $\Omega$ | \Omega |
| $\sigma$ | \sigma | | |

Notice 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 $A^{\mu}_{\nu}$ 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.

*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, the user has to explicitly include it in the corresponding diagram through the "`Add LaTeX package`" entry of the Options menu. 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. When the cursor is moved over this icon, the corresponding input text is displayed in the upper part of the canvas. Notice that the LaTeX text string will automatically be put between $ signs, so the text will always be in LaTeX math mode. In edit mode, the user may choose the size and color of the LaTeX font, and the alignment with respect to the current position of the text object.

### 3.4.3. Grouping

`JaxoDraw` allows for the grouping of objects so that they can be moved, resized and edited in a single operation. Objects may be grouped together by two different means:

*Using the Group button:* When the Group button in the edit button panel is clicked, the program goes into group mode, i.e., the user can click on the handles of those objects that he wants to group together. Chosen objects will have their handles filled in light gray, clicking again on the handle will take the object out of the group. The grouping operation is finished when the right button is clicked, the middle button cancels the operation.

*Using the "faint box" method:* Whenever the user clicks the middle mouse button on the canvas and starts 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 group mode, i.e., without having the Group button pressed.

### 3.4.4. 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 tab` entry. These operation are also presented in a pop-up menu if the user right-clicks on a tab. Objects from one graph may be copied to other graphs, using the `Copy` and `Paste` entries.

### 3.4.5. Copying to and pasting from 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, right-click on the canvas and drag the mouse: there will appear the faint box described in the previous subsection. Once the right button is released, all objects en-

tirely located inside this box will be copied to the clipboard (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 `Options` menu, or right click on the canvas tab and choose the `Paste` entry from the pop-up menu).

### 3.5. Setting resources

`JaxoDraw` allows the permanent setting of preferences via the Preferences item in the Options menu. The first group of settings are the default HTML viewer (to view the `JaxoDraw` documentation in HTML format), a default text editor (used for previewing LaTeX text output) and a default postscript viewer (used for previewing the printer or direct postscript output). Note that it is necessary to specify a default postscript viewer in order to view postscript files from within `JaxoDraw` because Java does not have an internal possibility to render postscript files. Contrary to that, if no default HTML viewer or text editor is specified, previews will still be possible with the Java internal HTML and text rendering mechanisms which will be used by default.

Apart from default values for all the items present in the Options pull-down menu, the user may also choose the initial screen size of `JaxoDraw` and specify default values for the size of the grid, the line width, the angular extent of arcs and the amplitude of photon and gluon objects.

If the `Save` button is pressed for the first time in the Preferences dialog, a corresponding preferences file called `.Jaxorc` will be created in the user's home directory. This file is read automatically every time `JaxoDraw` is started. It is in an XML format that may be edited manually in principle, the preferred way of editing being the above mentioned Preferences menu dialog of the graphical user interface.

### 3.6. 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 (the latter being commands that are used when exporting to a LaTeX file), unknown commands will be silently ignored. The commands known to `JaxoDraw` are:

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

(see the `axodraw` user guide[13] for documentation on these commands [3]). Finally, 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. When reading a LaTeX file generated by `JaxoDraw` itself, 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 that start with `%%\JaxoDrawID` are used by `JaxoDraw` to gather extra information for identifying objects. 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

[13] http://www.nikhef.nl/~form/FORMdistribution/axodraw/.

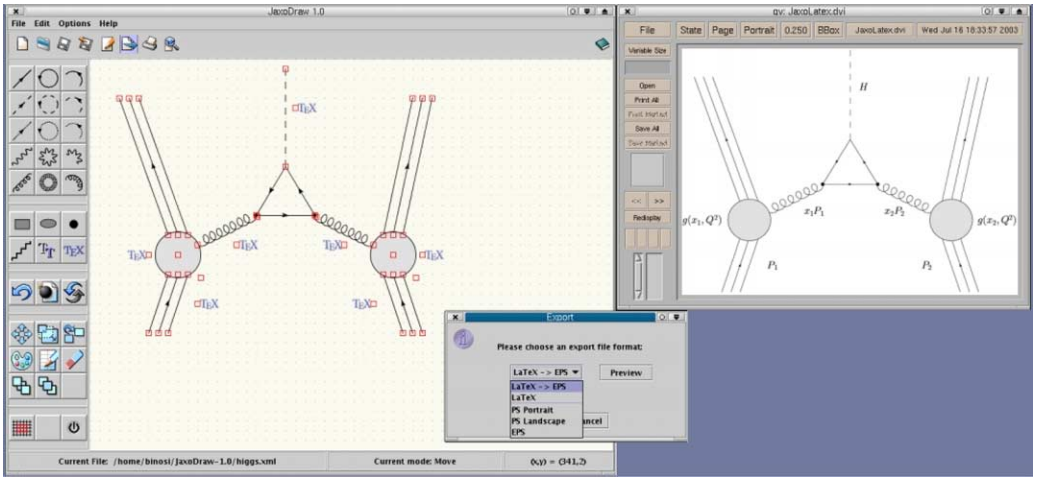


Fig. 2. Drawing a Higgs production mechanism.

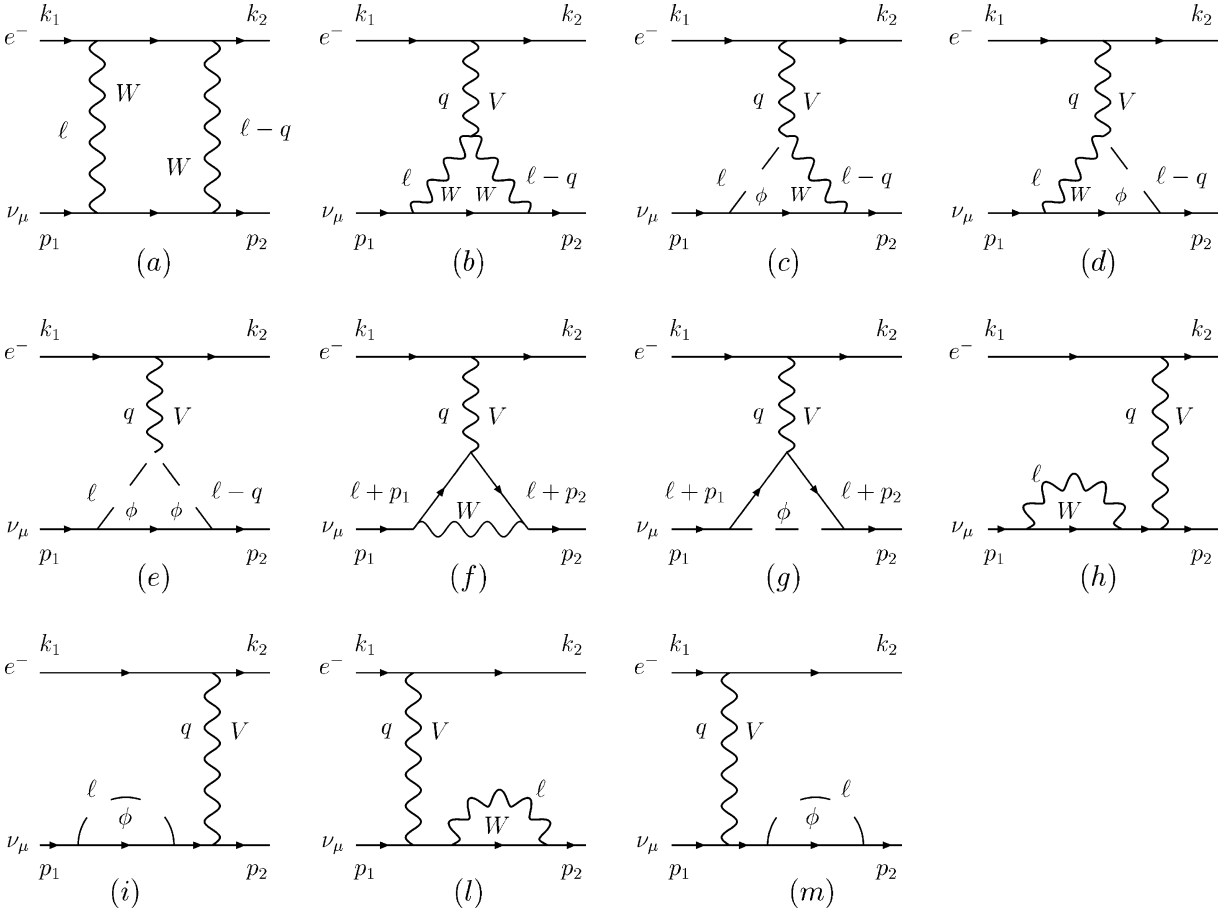


Fig. 3. A number of possible $e^-\nu$ scattering diagrams.

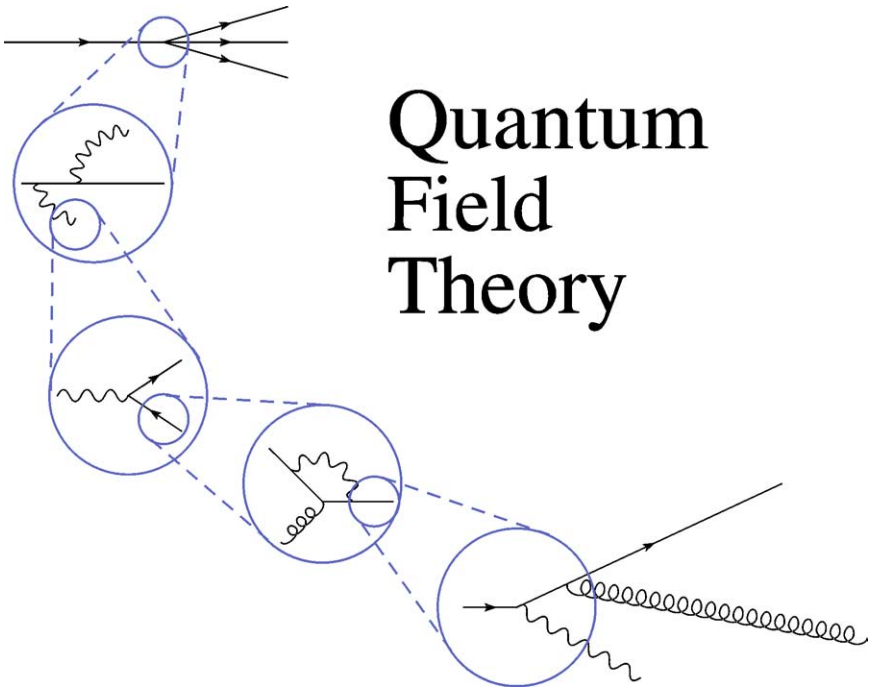


Fig. 4. A real world cover art design!

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).

## 4. Examples

This section gives a few examples of Feynman diagrams created with `JaxoDraw`. This is done for illustrative purposes to show some of the capabilities of the program.

Fig. 2 is a screenshot that shows the editing session of a Feynman diagram. The window on the left is the working area where the user draws the graph. Note that the LaTeX labels cannot be rendered directly on the screen, they are indicated by little blue "TeX" icons, the corresponding LaTeX text is shown in a pop-up window on top of the screen if the user moves the mouse over an icon. The window on the right shows the final Postscript output as generated by the internal Postscript preview mechanism.

Fig. 3 shows a number of Feynman diagrams that were all created in one editing session and exported as a single `eps` file. Note that it is also possible to export each diagram separately.

Fig. 4 finally illustrates that `JaxoDraw` can be used for creating more than just technical Feynman diagrams. This drawing was inspired by the cover of the book by Peskin and Schroeder [9].

## 5. Additional information

The program and HTML manual are available from the CPC Library. For latest information on the program you should consult the `JaxoDraw` home page at http://altair.ific.uv.es/~JaxoDraw/home.html, in particular, consult the Bugs section for an updated list of problems and known bugs (and solutions if available), and for instructions on how to report a bug. There is also a Docs section on this page that gives some practical information (tips and tricks) for using `JaxoDraw`.

The source code of `JaxoDraw` is completely documented using Java's `javadoc` utility. The file `README` contains information on how to create the `javadoc` API.

## Acknowledgements

We are grateful to Professor Arcadi Santamaria for numerous helpful remarks and moral support during the development of `JaxoDraw`. We also acknowledge Professor Jos Vermaseren for his kind permission to use and distribute his `axodraw` style file along with `JaxoDraw`. Finally, we are indebted to all the people who sent us bug reports and suggestions.

## References

[1] M. Veltman, Diagrammatica: The Path to Feynman Diagrams, Cambridge Univ. Press, Cambridge, UK, 1994.

[2] M.J.S. Levine, A LaTeX graphics routine for drawing Feynman diagrams, Comput. Phys. Commun. 58 (1990) 181–198.

[3] J.A.M. Vermaseren, Axodraw, Comput. Phys. Commun. 83 (1994) 45–58.

[4] T. Ohl, Drawing Feynman diagrams with Latex and Metafont, Comput. Phys. Commun. 90 (1995) 340–354.

[5] T. Hahn, Generating Feynman diagrams and amplitudes with FeynArts 3, Comput. Phys. Commun. 140 (2001) 418–431.

[6] A. Laina, Xfey, a Feynman diagram editor, Comput. Phys. Commun. 111 (1998) 217–242.

[7] K. Arnold, J. Gosling, D. Holmes, The Java(TM) Programming Language, third ed., Addison–Wesley, Reading, MA, 2000.

[8] D. Binosi, L. Theußl, JaxoDraw: A graphical user interface for drawing Feynman diagrams, http://arxiv.org/abs/hep-ph/0309015.

[9] M.E. Peskin, D.V. Schroeder, An Introduction to Quantum Field Theory, Addison–Wesley, Reading, MA, 1995.