

FeynGame-2.1 – Feynman diagrams made easy

Robert Harlander,* Sven Yannick Klein and Magnus Schaaf

TTK, RWTH Aachen University, Sommerfeldstr. 16, 52056 Aachen, Germany

E-mail: robert.harlander@rwth-aachen.de,

sven.yannick.klein@rwth-aachen.de, magnus.schaaf@rwth-aachen.de

FeynGame is an open-source software tool to draw Feynman diagrams, but also to get acquainted with their structure. This article reports on a number of new features which have been added to FeynGame since its first release. These include full support of \LaTeX for the line and vertex labels, the possibility to automatically include momentum arrows, new graphical elements, and new pedagogical features. FeynGame is freely available

- as jar file from <https://web.physik.rwth-aachen.de/user/harlander/software/feyngame>
- as source code from <https://gitlab.com/feyngame/FeynGame>

*The European Physical Society Conference on High Energy Physics (EPS-HEP2023)
21-25 August 2023
Hamburg, Germany*

*Speaker

1. Introduction

FeynGame is a Java tool for drawing and playing with Feynman diagrams. The initial idea for its development was to provide a tool for high-school students, teachers, or undergraduate students which allowed them to get familiar with the concept of Feynman diagrams in a playful way. While this is still one of the main purposes of FeynGame, its functionality allows one to use it as a simple, efficient, and flexible drawing tool for Feynman diagrams.

The main feature which distinguishes FeynGame from other drawing tools is that it is based on particle physics models. This means that FeynGame “knows” about the Feynman rules of a particular theory. This information is provided to FeynGame in the form of a *model file*. By default, FeynGame assumes the Standard Model (SM) as the underlying theory, and the user may start from the corresponding *model file* to build *model files* for simpler or more elaborate theories.

This model-based approach has two important consequences. On the one hand, every particle of a particular theory may be given unique attributes, such as the line style or a text label. For example, a gluon may be represented by a red spiral line and the label g , a Higgs boson by a dashed blue line and the label H . This makes the drawing of Feynman diagrams very efficient. Assume that one would like to distinguish top-quark lines in a Feynman diagram from other fermion lines by drawing them a bit thicker and/or in a different color. As opposed to other Feynman diagram drawing tools, there is no need to change the thickness and color of every top-quark line separately: the top-quark line is simply a separate object with well-defined properties.

The second consequence of the model-based approach is that FeynGame can check whether a particular Feynman diagram is compatible with the interactions of the underlying theory. Simply pressing the key `f` is sufficient to validate or disprove the Feynman diagram currently drawn on the canvas corresponds as a physical amplitude.

The experienced particle physicist will scarcely need this feature, of course. Indeed, its main application is educational in the context of the “game mode” of FeynGame. Currently, FeynGame offers one type of game, named InFin (for Initial and Final), where the initial and final state of an amplitude are generated (quasi-)randomly, and the player has to construct a suitable valid (connected) Feynman diagram within the underlying particle model. This game can be configured by the user to a high degree, as will be described in more detail in Section 4.

2. General functionalities

2.1 The main frame

The left part of Fig. 1 shows a screenshot of the so-called *main frame* of FeynGame. The Feynman diagram in the upper part of that window (the *canvas*) has been drawn using a common input device of the computer, such as the mouse or the trackpad, or most conveniently a stylus device. For simplicity, we will refer to the input device as “mouse” in the following.

Below the canvas is a set of tiles, representing the lines (or particles) of the *current model*. Selecting one of these tiles allows one to subsequently add the corresponding line to the canvas by clicking and dragging. Trying this out, the user may soon discover a number of other quite unique features of FeynGame:

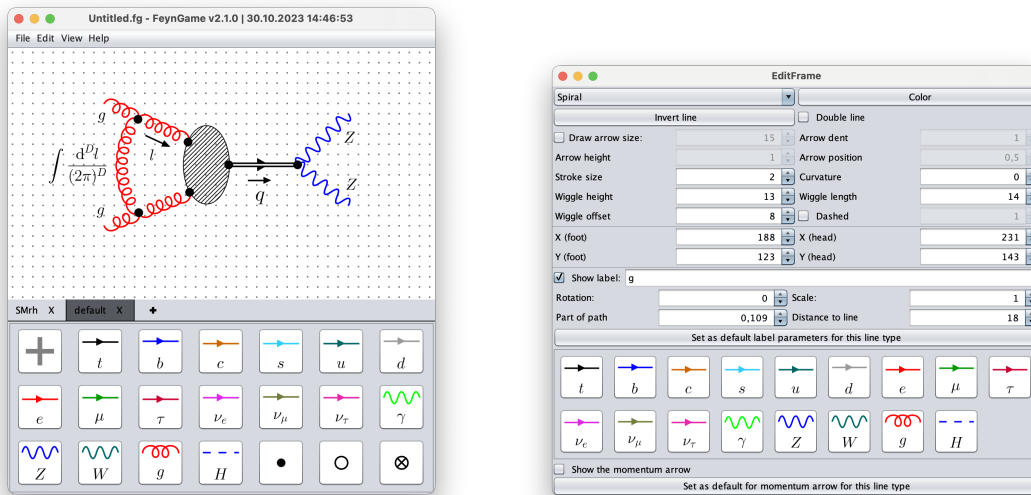


Figure 1: Left: The *main frame* of FeynGame. The tiles in the lower part of the frame represent the *current model*, the diagram itself is displayed on the canvas. The diagram shown here contains elements from the SM (gluons, Z-bosons), but also non-standard elements like the hashed oval shape, and the double-line propagator which can be obtained via the menu items or the *edit frame* (right).

- Initially, every line is drawn as a straight line. It can then be curved, for example by using the mouse wheel. The end points remain fixed during the curving operation.
- If the end of one line is positioned close to a second line, FeynGame will connect the two through a vertex (no matter if it is compatible with the *current model* or not). The second line will formally be split into two by this operation, such that the vertex actually connects three lines.
- By clicking and dragging the lines on the canvas with the mouse, they can easily be moved (if clicked close to the line’s center), rotated, stretched, or shortened (if clicked close to one of the line’s ends).
- Clicking and dragging a vertex will move the vertex and all the lines connected to it. This makes it very simple to modify the shape of a Feynman diagram.

The completed diagram can then be exported in many formats such as JPG, PDF, or PostScript, which will be in one-to-one correspondence to the image on the canvas (“what you see is what you get”), modulo graphical aids like grid points and *helper lines* which can be activated in the representation on the canvas.¹ It can also be copied to the clipboard and thus simply pasted to other applications such as Keynote or PowerPoint as PNG image (i.e. with transparent background) without the need of saving the image to disk first.

¹The export to PDF on MacOS sometimes does not correctly display the vertex markers. We recommend to export to PostScript and subsequently convert to PDF, if desired.

2.2 The edit frame

The *main frame* is sufficient for quick drawings where all lines have their default attributes. Deviations from the default can be achieved for example with the help of the *edit frame*, which opens by pressing `e`. The content of this window depends on the *active* object (an object can be *activated* by clicking on it in the canvas). For example, the right part of Fig. 1 shows the *edit frame* for one of the external gluon lines of the diagram shown on the left. The *edit frame* allows one to modify all attributes of the active object. For lines, this is the color, the width, the position on the canvas, the curvature, the shape, the label, and much more. For vertex markers, one can also choose a filling pattern, for example.

3. New drawing features

Numerous features and improvements have been added to FeynGame since its first release [1]. We highlight some of the most important ones here. Some of them are exemplified in Fig. 1. A more detailed list will be provided in a forthcoming publication.

L^AT_EX labels. FeynGame will interpret any text labels for lines or other graphical objects entered via the *edit frame* or in the *model file* as L^AT_EX code.² As usual, the canvas will display the compiled L^AT_EX symbols in the same way as they would appear on the exported image.

Momentum arrows. For each line, FeynGame can draw an associated momentum arrow and optionally a momentum label. This feature is toggled by activating the line and pressing `p`, or via the *edit frame*. The latter provides many options for the form and positioning of the momentum arrow and its label.

Shapes. In addition to lines and vertices, the graphical elements “oval” and “rectangle” are available. They can be inserted into the canvas via the menu item `Edit >> Shapes`, and deformed and rotated as usual with the mouse. A full list of changeable attributes (fill color, fill pattern, line color, etc.) is obtained via the *edit frame*.

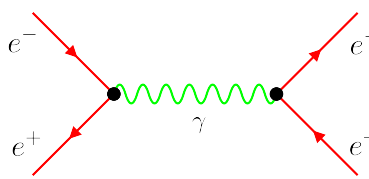
Editing multiple objects. By clicking a tile in the *current model*, the attributes of that particular line type can be modified. This affects all current and subsequently drawn lines on the canvas in the current session. In order to adopt the changes also for future sessions, one can save the *model file* using `File >> Save model file`.

Multiple model files. One can work with several different models at a time and switch between them via tabs which appear between the canvas and the *current model*.

Model from diagram. When loading a diagram that was drawn in an earlier session and then saved to disk, FeynGame can create a model which consists of all the lines and vertices which appear in that diagram. This can be helpful if the *model file* which was used to produce the diagram is unknown, for example because the diagram was produced by someone else. Using the option to load several *model files*, it is then easy to add new lines to the diagram.

Zoom and shift. Diagrams can be moved as a whole on the canvas by pressing `Shift` and click-dragging the canvas. One can also zoom into and out of the canvas with the arrow-up and arrow-down keys, or using the corresponding sub-items in `View`.

²FeynGame uses the Java library [JLaTeXMath](#) for that purpose.



$$\begin{aligned}
 & e^2 u_{\alpha_1}(\vec{p}_1) \bar{v}_{\beta_1}(-\vec{p}_2) (-iQ_e \gamma_{\beta_1 \alpha_1}^{\mu_1}) \\
 & \times i \left[\frac{-g_{\mu_1 \nu_1}}{(-p_2 + p_1)^2 + i\epsilon} + (1 - \xi_\gamma) \frac{(-p_2 + p_1)_{\mu_1} (-p_2 + p_1)_{\nu_1}}{((-p_2 + p_1)^2)^2} \right] \\
 & \times (-iQ_e \gamma_{\delta_1 \gamma_1}^{\nu_1}) v_{\gamma_1}(-\vec{q}_1) \bar{u}_{\delta_1}(-\vec{p}_2 + \vec{q}_1 + \vec{p}_1)
 \end{aligned}$$

Figure 2: When drawing the diagram on the left with FeynGame, it will produce the amplitude on the right.

Performance improvements. A number of issues related to CPU efficiency or graphical representations have been improved or resolved.

4. New educational features

So far, we have described new graphical features of FeynGame. However, also the pedagogical part has been updated significantly.

4.1 Amplitude

Provided that the *model file* contains the required information on the Feynman rules, FeynGame can produce the mathematical expression for the amplitude of a particular (valid) Feynman diagram. Pressing will check the diagram for validity, and if this is passed, it will open a dialog box where the user can choose between seeing the formula for the amplitude, or copying this formula in L^AT_EX code to the clipboard which can simply be pasted into any L^AT_EX document.³ The default model contains all the required information for this. One can also ask FeynGame to display the momentum routing through the diagram.

For example, if one uses the default model to draw the diagram for $e^+e^- \rightarrow e^+e^-$ via the s -channel exchange of a photon shown in the left part of Fig. 2, and subsequently asks FeynGame for the amplitude, the L^AT_EX code will compile to the expression shown in the right part of Fig. 2.⁴

4.2 Level Generator

The initial and final states generated in the InFin game mode are not generated on the fly, but are read by FeynGame from a control file, the so-called *level file*. FeynGame comes with a default version of the *level file*, which is based on the SM. It lists a number of possible valid processes from which FeynGame randomly picks one and displays its initial and final state particles on the canvas. The player is asked to construct a connected Feynman diagram which mediates the process. If the task is completed, FeynGame picks another pair of initial and final states from the *level file*.

If one wants to assume a different underlying particle model, one needs to supply FeynGame with a corresponding *level file*. Since constructing such a file by hand can be quite tedious, in particular if the number of processes to be played should be larger than just a few, FeynGame

³Note that the expression for the amplitude can be quite long and extend beyond the width of the screen. Also, FeynGame only inserts the Feynman rules and does not attempt any simplifications on the resulting expression.

⁴The linebreaks were inserted manually.

provides a *Level Generator*. Given a *model file*, the maximal and minimal number of particles in the initial and final state, and the maximal number of loops, it uses the diagram generator `qgraf` [2, 3] to generate the requested number of processes and puts them into a *level file*. We refrain from a more detailed description of the level generator due to the page limitations of these proceedings. It will be provided in a forthcoming publication; the interested reader is welcome to contact us beforehand.

5. Plans for the future

As pointed out above, the features described in this paper are a collection of just some of the most important changes since the original release of FeynGame, and the interested reader is invited to try out the new version. In addition, many other features are planned, in preparation, or even already contained in preliminary versions. Among the latter is the automatic visualization of `qgraf` output, which should be a very useful feature for debugging code that calculates Feynman diagrams. There are also more sophisticated versions of `InFFin`, where, in addition to the given initial and final state particles, also specific internal lines have to be incorporated in the final Feynman diagram. Future releases of FeynGame should also allow for editing groups of lines in a Feynman diagram simultaneously.

6. Conclusions

We have presented a number of new features which are available in the latest version of FeynGame and hope that they will be helpful to the physics community in producing high-quality publication-level Feynman diagrams in an efficient way. We also hope that FeynGame helps to convey the structure of Feynman diagrams to particle physics novices.

Any feedback or requests for help or new features are welcome and can be sent directly to the authors, or submitted via the `gitlab` repository.

Acknowledgments. We would like to thank all users of FeynGame for their feedback. Special thanks go to Jakob Linder for pointing out a number of issues, Lars Bündgen and Erik de la Haye for contributing to the FeynGame development, and Maximilian Lipp for designing FeynGame’s structure in such a flexible way.

References

- [1] R. V. Harlander, S. Y. Klein, and M. Lipp, *FeynGame*, *Comput. Phys. Commun.* **256** (2020) 107465, [arXiv:2003.00896](https://arxiv.org/abs/2003.00896) [[physics.ed-ph](https://arxiv.org/abs/2003.00896)].
- [2] P. Nogueira, *Automatic Feynman graph generation*, *J. Comput. Phys.* **105** (1993) 279–289.
- [3] P. Nogueira, *Abusing qgraf*, *Nucl. Instrum. Meth. A* **559** (2006) 220–223.