Whizard GUI Walkthrough

Augustinas Malinauskas

September 8, 2015

Abstract

This summer student report also serves as a Whizard GUI walkthrough on which I was working at DESY. WHIZARD is a program system designed for the efficient calculation of multi-particle scattering cross sections and simulated event samples. The GUI allows the user to start using Whizard without a need to know sindarin syntax (language in which Whizard speaks). Based on the input parameters GUI calls Whizard, displays the log file and produced histograms or plots.

Contents

1 Introduction

1.1 About GUI

Whizard GUI (Graphical User Interface) a local website which once started is accessible using any modern browser, GUI is written in JavaScript and is based on Node.js runtime JavaScript engine which allows to call external programs such as Whizard, read file system and most importantly create a server. The GUI could be used accessed on a local machine or in principle from external machine once ports are properly configured.

1.2 Download

To download Whizard GUI use:

```
git clone https://bitbucket.org/whizard/whizard-gui.git
```

Alternatively visit https://bitbucket.org/whizard/whizard-gui

1.3 Prerequisites

The required javascript packages for the GUI are managed by **npm**. The GUI itself is based on **node.js** engine. To install them both on the Debian-like system use:

```
sudo apt-get install npm && nodejs
```

Note: on some systems `nodejs` comes with the name of `node`.

1.4 Installation

To prepare the GUI for use locate the GUI folder containing `configure.js` this script will generate **npm** installation file based on your installed packages. Type:

```
sh configure.sh
```

Now a `package.json` was generated witch contains information about required JavaScript packages in an **npm** format. The following command needs to be executed once to download required packages locally (super user privileges are unnecessary):

```
npm install
```

1.5 Running

To start a GUI locate the folder containing `package.json` and type:

```
npm start
```

The local server will be started on a specified port (by default port 3000). Message in the terminal should confirm that the GUI server was started successfully. Once GUI is started it could be accessed by visiting website with the following address:

```
http://localhost:3000
```

To stop the GUI use `CTRL+C` in the terminal, this will shutdown nodejs as well as any calculations started via GUI. All files of the last calculation remains in `output-whiz/` directory (by default).

2 Usage and Overview

2.1 Limitations

Although Whizard GUI (Graphical User Interface) allows to accomplish simple tasks, it is in early stage and more complicated procedures are unavaialble within the GUI. Nevertheless it contains examples and is good starting point to learn sindarin syntax and see the outcome of the program quickly. It might be useful to look at Whizard Manual: `https://whizard.hepforge.org/manual.pdf`

2.2 First look

Figure 1 shows the main control window as seen once started the GUI. There are some built in examples that could be accessed by clicking *Examples* link in the navigation bar. Choosing an example will automatically fill appropriate input fields.

2.3 Additional notes

By default Whizard assumes that units are in GeV.

3 Basic Settings

3.1 Running

To run Whizard using selected parameters simply click big green *Run Whizard* button. By default Whizard is started by keeping previously calculated relevant integration and simulation data. However, there are other options to choose from available by clicking an arrow on the right to the button, such as:
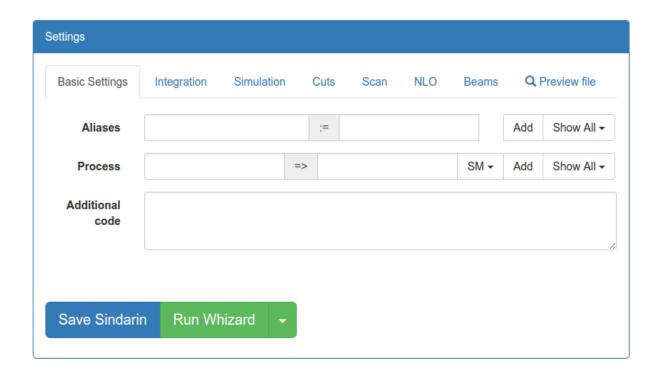
Figure 1: Main Whizard GUI control window

- Run & Rebuild All (`--rebuild`)
- Run & Rebuild Events (`--rebuild-events`)
- Run & Integrate (`--rebuild-grids`)

You can familiarise with these options in Whizard Manual: `https://whizard.hepforge.org/manual.pdf`

3.2 Alias

Alias could be added in the **Basic Settings** section. An example of alias given below, where different particles are separated by :

```
pr = u:ubar:d:dbar:g
 j = u:ubar:d:dbar:g
```

Click *Add* to define a new alias which then could be used anywhere in the GUI.

3.3 Process

GUI allows adding multiple processes, for which individual configuration may be specified. An example processes might be:
Check Whizard manual for details.

```
e+, e- => e+, e-
W+, W- => W+, W-
pr, pr => W+, W-, j
```

4 Integration

This section allows to choose center of mass energy for the process as well as number of iterations and number of Calls per iteration. On the left you see a list of all processes you defined in **Basic Settings** section. For each process you can select different options. Options in this example will be translated in the following sindarin code:

```
process proc_1 = "e+", "e-" => "W+", "W-"
sqrts = 500
integrate(proc_1) {iterations=5:10000:"gw"}
```
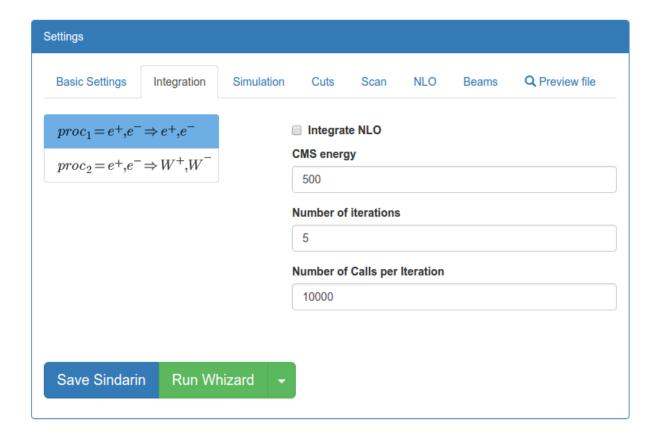

Figure 2: Integration section

5 Simulation

Just as integration, simulation could be configured for each process individually. Text fields such as *Title* or *X-label* accept LaTeXcode enclosed within dollar signs, for example `$code$`. An example of simulation is given in Figure 3. Such GUI configuration results in the following sindarin code:

```
# New histogram
histogram distribution_1 (-1, 1, 2 / 30.) {
$title = " Angular Distribution"
$x_label = "$\cos(\theta)$"
}
analysis = record distribution_1 (eval cos (Theta) ["W+"])
simulate(proc_1) { n_events = 300 }
compile_analysis
```

Which by clicking *Run Whizard* produces the desired distribution.

6 Cuts

This tab allows to apply cuts. An example given in Figure 4. The following configuration is translated to:

```
cuts = all Pt > 100 [collect["W+":"e+"]]
       and all M > 200 [collect[j]]
```

The union field in the Figure 4 collects four-momentum of the particles.

7 Scans

An example of scans would be calculating cross-section as a function of center of mass energy. This section allows to choose subintervals $(\sqrt{s}_{min}, \sqrt{s}_{max})$as well as increments in $\sqrt{s}$ in each subinterval. Scans section in the GUI is shown in figure 5.

8 GUI config file

Config file is located at `public/js/gui.config.js`. It allows to set up some basic configuration such as:

- `WHIZARD_OUTPUT_DIR` - Directory in which Whizard files will be produced.

- `WHIZARD_OUTPUT_SIN` - Directory in which Sindarin files will be produced.

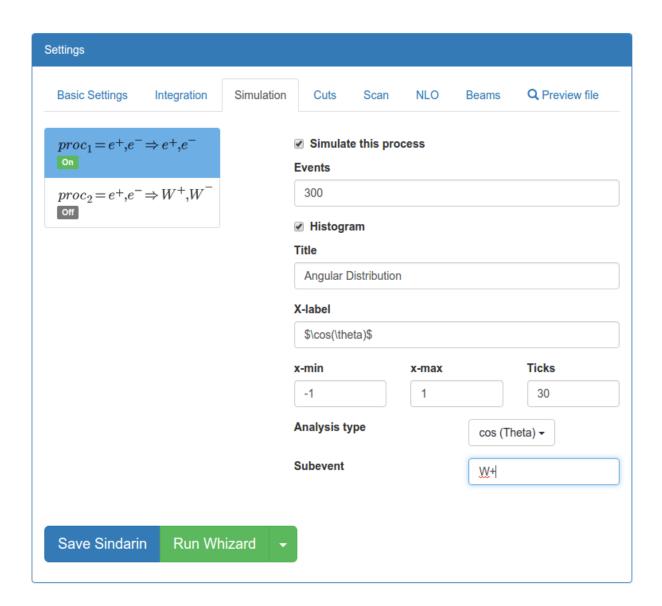- `PORT` - Default port to start local GUI server.

Figure 3: Integration section

- UPDATE_TIME - Time in milliseconds to update Whizard computation Log and Plots while Whizard is running.

- USE_GOOGLE_LATEX - Use Google Graph API to produce latex generated images instead of text in the GUI. (ex: "$e^+e^- \Rightarrow W^+W^-$" instead of "e+,e- => W+,W-"). This option required internet connection.

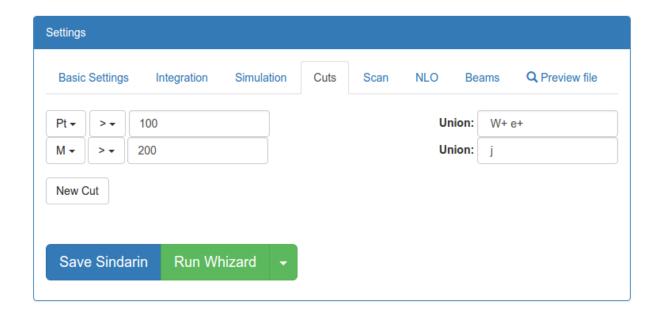Defined directories will be created once the GUI is started.

Figure 4: Cuts section

9 Acknowledgement

I want to thank Christian Weiss for being the first point of contact this summer and guiding my work. I would also like to thank Jürgen Reuter and Bijan Chokoufe Nejad for occasional meetings, and finally to people with whom I had interesting discussions: Fernando Romero López, Thanat Sangkhakrit and Иван Соболев.
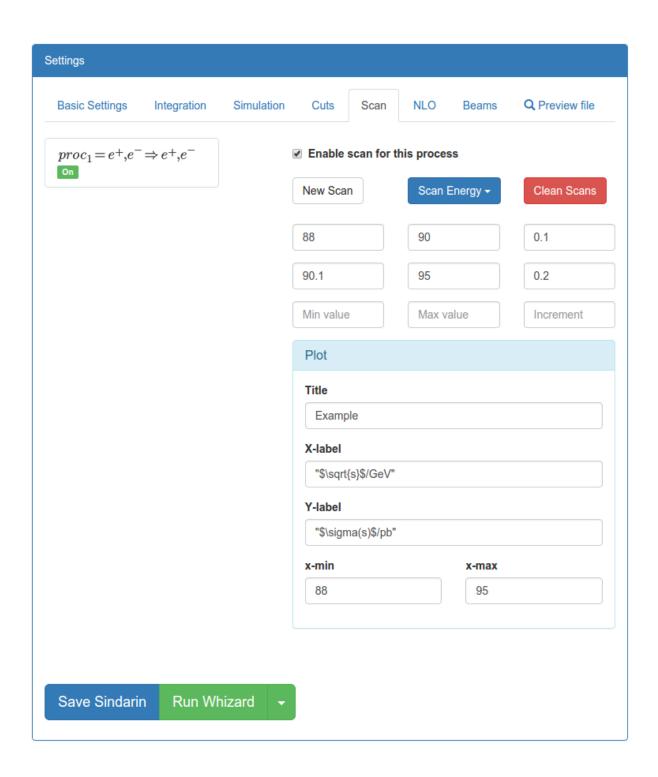
Figure 5: Scans section

References

[1] W. Kilian, T. Ohl, J. Reuter, Eur.Phys.J.C71 (2011) 1742 *arXiv: 0708.4233 [hep-ph]*

[2] M. Moretti, T. Ohl, J. Reuter, *arXiv: hep-ph/0102195*