PenRed Geometry Viewer: User Manual

Vicent Giménez Alventosa

February 15, 2023

1   Introduction

This document describes how to install and use the PenRed Geometry Viewer, which code and pre-compiled bundles can be found in the viewer repository[1].

The Geometry Viewer uses the PenRed geometry libraries to process and show geometries compatible with any PenRed geometry module. This is done calling directly to the *locate* and *step* methods. Therefore, the viewer shows exactly the geometry system which is *"viewed"* by the simulation code.

2   Installation

The viewer itself is isolated from PenRed to achieve compatibility between both packages regardless the PenRed or the viewer version. This is done via a plugin approach, where a shared library is loaded in the viewer to access all the available geometry modules in the PenRed package.

Therefore, regardless the installation method, the Geometry Viewer requires this PenRed shared library to be able to access to the geometry modules. Although this shared library is provided in the pre-compiled bundles, which will be described next, this one may not corresponds to the latest PenRed version. Moreover, as this bundles have been compiled to run on a non specific architectures, the compilation is done without native optimisations.

Therefore, it may be interesting for the user to compile the library itself regardless the installation method. The procedure to follow is explained following.

2.1   Shared Library Compilation

The shared library must be compiled and stored in the same folder as the viewer executable. However, the library compilation is done automatically using the appropiate options of the provided *CMakeFiles* file in the PenRed package.

The flag that must be enabled is the BUILD_C_BINDS. For example, to build it in the *src* folder of the PenRed package, we can run the following commands

```
mkdir build
cd build
cmake -DBUILD_C_BINDS=ON -DWITH_NATIVE=ON -DWITH_DICOM=OFF  \
      -DWITH_MULTI_THREADING=ON -DWITH_MPI=OFF -DWITH_LB=OFF \
```

---

[1] https://github.com/PenRed/GeometryViewer

```
        -DDEVELOPMENT_WARNINGS=OFF ../
 make install
 cd ..
```

Once the build is completed, the library will be stored in the folder

```
/src/bindings/C/viewers/geometry/
```

inside the PenRed package, with a different name depending on the OS. For example, in linux systems, the library is named *libgeoView_C.so*. Notice that, with the previous cmake command, the library is compiled with no DICOM support. Therefore, the viewer will not be able to show DICOM geometries. To enable DICOM geometries, the DICOM flag must be enabled and the corresponding dependencies installed.

Finally, the shared library must be placed in the same folder of the Viewer executable to be loaded properly.

2.2   Pre-Compiled Bundles

The simplest option to use the viewer is downloading the already built packages, which include the executable file, the compiled shared library, a script to run the viewer and the required QT libraries and other dependencies to be able to run the viewer without a QT installation. These bundles can be found in the releases provided in the Viewer repository[2]. Notice that the shared library can be updated with a different PenRed version following the procedure described in the Section 2.1.

If the program runs but is unable to load a geometry, try to compile the geometry viewer library to replace the precompiled one, following the explained procedure in the Section 2.1.

2.3   Viewer Source Compilation

The geometry viewer has been developed using the QT libraries via the QT creator IDE. Therefore, the source code can be imported and built as a QT project using the QT Creator. By default, the required PenRed shared library is compiled automatically when the QT project is built. This one uses the actual master branch of the PenRed repository. However, this step can be disabled via a CMake option named *BUILD_VIEW_SHARED_LIB*. If it is disabled, the library must be compiled from the PenRed source code, as is described in the Section 2.1.

3   Usage

This section will describe how to use the PenRed Geometry Viewer. When the Viewer is started, the main window is presented, as is shown in the Figure 1.

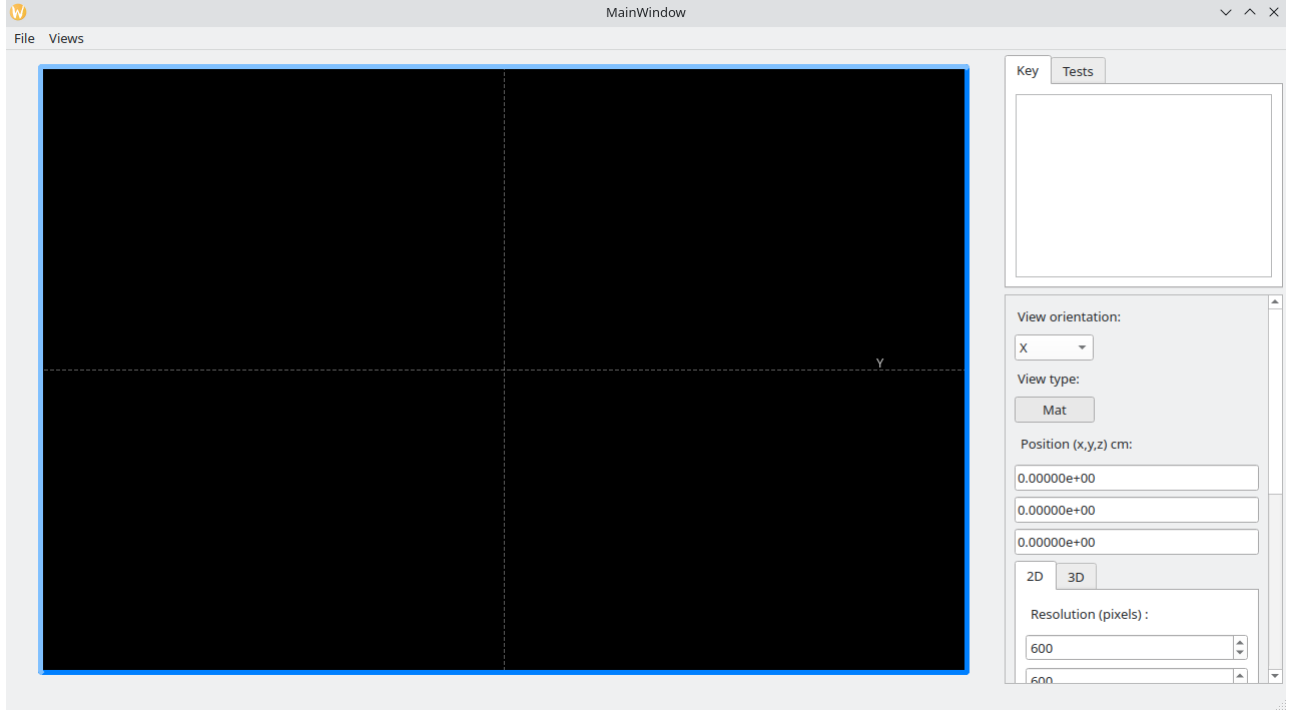[2]`https://github.com/PenRed/GeometryViewer/releases`

Figure 1: Main window of the PenRed Geometry Viewer.

Then, the first step to use the Viewer consists on loading a geometry. This is done using the *Load* submenu inside the *File* menu (Figure 2). To load the geometry, we can specify a quadric or triangular mesh file, or specify a configuration file to load. If we use the configuration file option, we must specify the same parameters as the used in the *geometry* section of the configuration file for the PenRed main program. However, notice that the prefix *geometry* must be removed from the parameters paths. For example, a quadric configuration file example follows,

```
type "PEN_QUADRIC"
input-file "/path/to/file.geo"
processed-geo-file "report.geo"
```
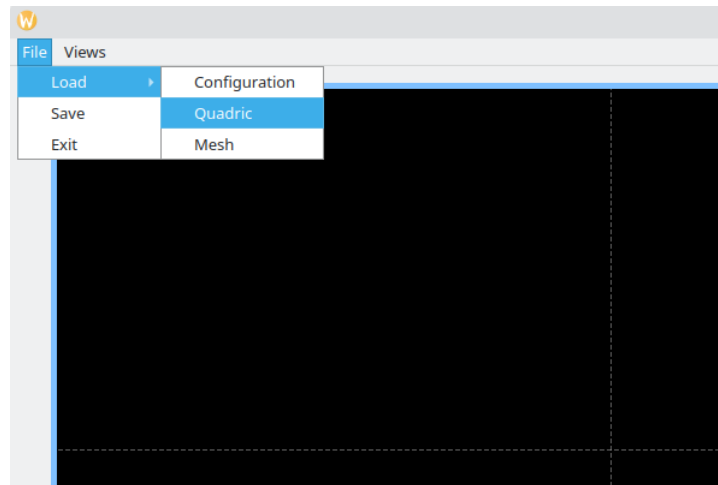

Figure 2: Load geometry procedure.

Notice that the simulation parameters specified in the *geometry* section, as the absorption energies per body, are not required to visualise the geometry. If an error is produced during geometry load, it will be shown in the console where the Viewer has been executed.

In the following sections the parameters to navigate within the geometry are described. This can be divided in common parameters, which are maintained even if the view plane is changed between 2D and 3D renders, and specific, which are independent between 2D and 3D renders.

3.1 Common parameters

Once the geometry is load, the user can move the camera inside the geometry using the interface or the shortcuts described following and shown in the Figure 3:
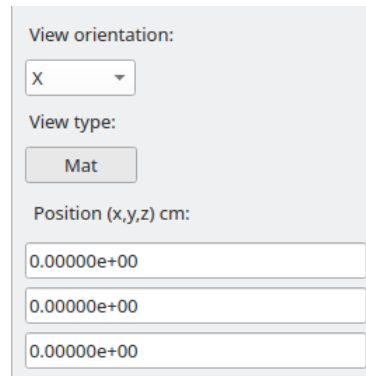

Figure 3: Common parameters.

- **Mat**: Change the render between material and body view. The keys with the colours assigned to each material and body identifiers can be found in the *Key* tab, next to the *Load* tab. Both views can be switched also using the m key.

- **View orientation**: This menu, which is labelled with a **X** in the Figure 3, allows the user to change the 2D plane where the geometry is rendered (*X, Y* or *Z*) or changing to a 3D view. It can be changed also selecting the render window and pushing the x, y or z keys respectively.

- **Position coordinates**: These values show the actual (x,y,z) position of the camera, in cm. In addition, the values can be edited to move the camera to the desired point. Also, to move the camera left, right, top or down, the user can use the keys a, d, w and s respectively or, in the 2D view, the key arrows.

3.2 Specific parameters

All parameters and options in the Section 3.1 are common to 2D and 3D views within the same render window. However, some parameters are specified independent for 2D and 3D render even if are specified for the same render window. For this reason, they are included in both, the 2D and the 3D parameters tabs. These parameters are listed following and shown in the figure 4:

Figure 4: Specific 2D parameters.

- **Resolution**: Specify the number of pixels, in the horizontal and vertical axis respectively, to be used to render the images. Notice that the more you use, the slower the render will be.

- **Pixel Size**: Specify the pixel size to be used. Increasing the pixel size will cause an effect of zoom out, while decreasing it will cause a zoom in effect. This value can be changed for the selected render window using the keys − and + respectively.

3.2.1 Specific 3D parameters

Finally, another set of parameters are specified and used only for 3D rendering. These ones are shown in the Figure 5 and described below:

Figure 5: Specific 3D parameters.

- **Camera(ρ, θ, ϕ)**: Specify the camera position in cylindrical coordinates taking as origin the common **position** parameter. Therefore, the camera always looks at the specified **position** in the common parameters. All values can be edited manually. Also, the θ angle can be changed using the up and down key arrows and the ϕ angle using the left and right arrows.

- **Camera(x, y, z)**: Same as the **Camera(ρ, θ, ϕ)** parameters but in Cartesian coordinates.

Finally, the **Resolution** and **Pixel Size** parameters behaviour is the same as the explained in the Section 3.2.

3.3   Test

The test option tab exposes an experimental feature which can be used to check the consistency of the geometry. As the 2D renders are performed using the *locate* geometry method, the test will use calls to the *step* method from each pixel to their neighbours to check the geometry consistence. If the material or body pixel value after a *step* call does not coincide with the obtained with the locate method, an error will be added to the list and showed in the *test* text box.

3.4   Views

It is possible to add and remove viewports using the *View* menu in the top left of the program window. Notice that, when multiple viewports are used (4 as maximum), the values shown in the parameters correspond to the active viewport. This one is remarked with a blue contour. Take into account that, when we delete a viewport using the *View* menu, the selected viewport to be removed is the actual active viewport.

References