SOFIA UNIVERSITY
ST. KLIMENT OHRIDSKI

Framework visualization - updates and progression 09.24.2024

Georgi Petkov

A few reminders

- The framework is a general designed framework to be used to simulating neutrino events in a detector.
- Used in the simulation study of Near Fgd detector.
- It can be run as stand alone application or using scripts (ROOT scripts -> dictionary files are generated at compile time)
- Main third party libs used are:
  1. Genie – neutrino events generation
  2. Geant4 – main simulation engine
  3. CERN Root – data storage and scripts applied

Content

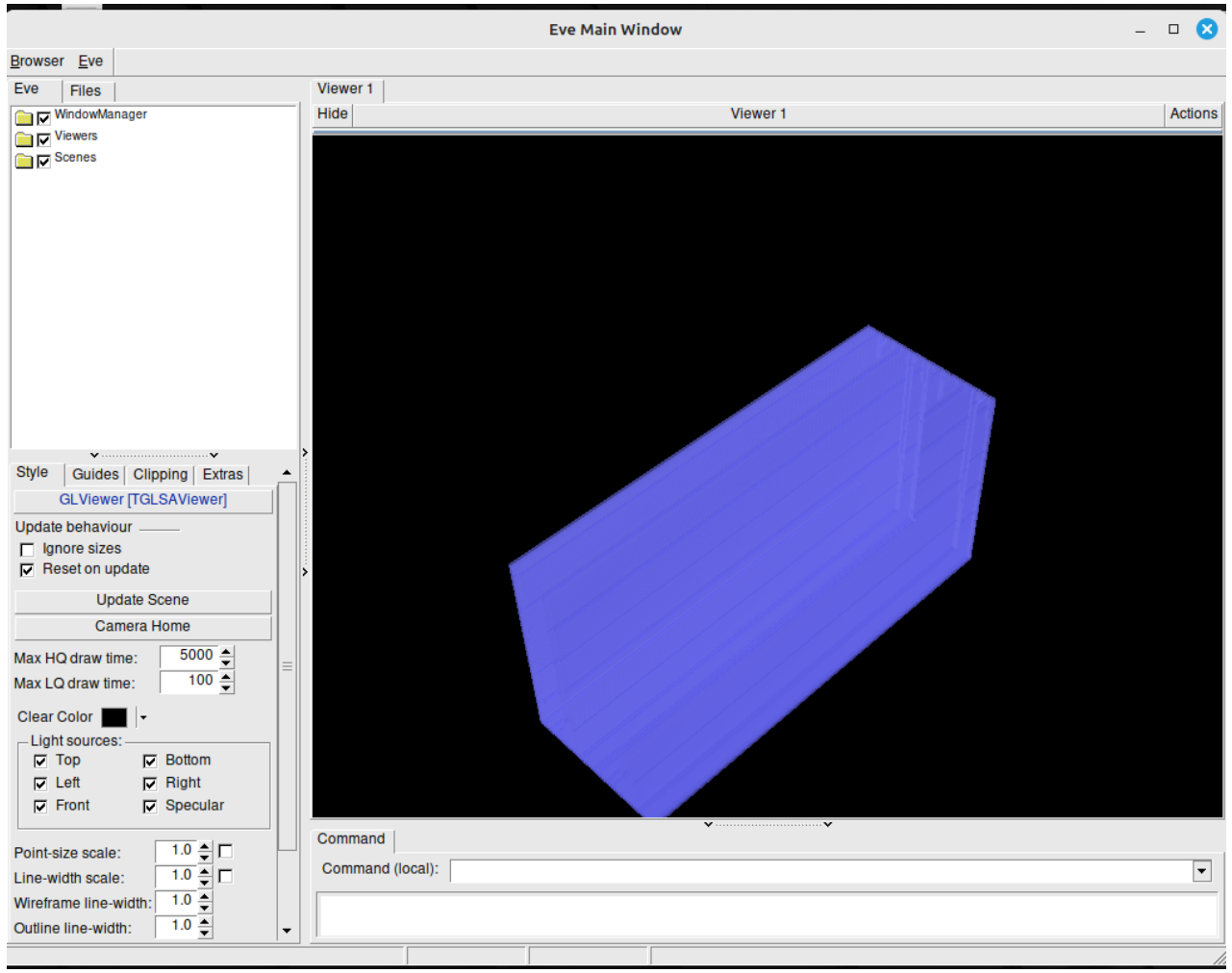
- Visualization of the detector (working volumes)
- Geant4 Visualizations of events
- Cern Root visualization of events
- Data processing and export – third party
- GUI working (under design)

Visualization of the detector

```
TGeoVolume* fiberYVolume = new TGeoVolume("fiberYVolume",fiberShapeY, fiberCoreMedium);

// Place the fiber core inside its coating
fiberYCoatVolume->AddNode(fiberYVolume, 1 /* One copy*/);

//----------------------------------
//============================================================================

//============================================================================
// Create the fiber hole along Z
//----------------------------------
TGeoCone* fiberHoleZ = new TGeoCone("FZ", GetLength()/2, 0, GetHoleRadius(),0 ,GetHoleRadius());

TGeoRotation* rotZ = new TGeoRotation("rotY",
                        fHoleRotZ.rotateX,
                        fHoleRotZ.rotateY,
                        fHoleRotZ.rotateZ);
rotZ->RegisterYourself();

TGeoCombiTrans* locationZ = new TGeoCombiTrans("locationZ",
                        fHolePositionZ.X,
                        fHolePositionZ.Y,
                        fHolePositionZ.Z,
                        rotZ);
locationZ->RegisterYourself();


// Create the fiber shape - the fiber shape is the contained within the fiber coating
TGeoCone* fiberShapeZ = new TGeoCone("fiberShapeZ", GetLength()/2, 0, GetFiberRadius() - 0.0495 ,0 ,GetFiberRadius() - 0.0495);

// Create the fiber coating-> this is the hole minus the fiber itself
TGeoCompositeShape* fiberZShapeCoat = new TGeoCompositeShape("fiberZShapeCoat","FZ - fiberShapeZ");

// Create fiber coating
TGeoVolume* fiberZCoatVolume = new TGeoVolume("fiberZCoatVolume",fiberZShapeCoat, fiberCladdingMedium);

// Create fiber core volume
TGeoVolume* fiberZVolume = new TGeoVolume("fiberZVolume",fiberShapeZ, fiberCoreMedium);
```

The construction of Sensitive volumes and geometry volumes is done by using quite long pieces of code.
For finding mistakes or mistmaches, we use CERN ROOT visualization -> geometry is exported as root file which can than be opened (by scripts).

Visualization of the detector



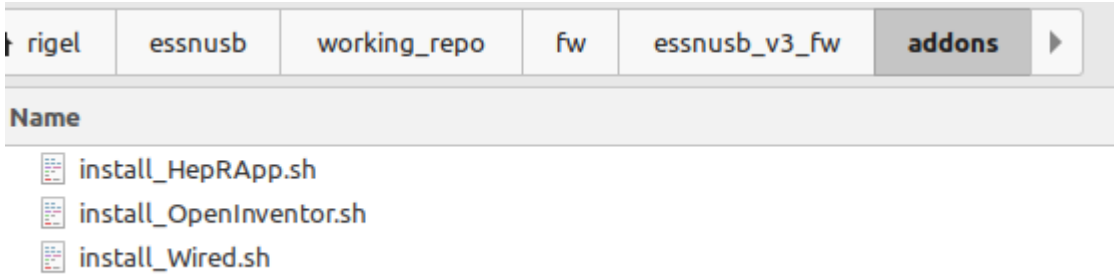
Here is the FGD detector at 980 000 cubes.
The Root Even Window is quite Responsive even at so many elements.

Geant4 Visualization of events

- The framework also does validation that the geometry is correct -> the framework does an export from root geometry to gdml format which is than imported to Geant4. Thus transition does also geometry validation for overlapping of geometry volumes, if there are overlapping volumes, they will be listed and the simulation will stop.
- When successful, one may need to check a given event what are the particle tracks and compare to the output algorithm.

Geant4 Visualization of events

- Geant4 has build in visualization drivers, some come with the framework and some need additional installation of software.
- Essnusb v3 comes along with script to facilitate installation (if desired) of the additional packages – the folder name is “addons”



Geant4 Visualization of events

- Drivers:
  1. OpenGL
  2. OpenInventor
  3. Qt3D
  4. VTK
  5. ToolSG
  6. RayTracer

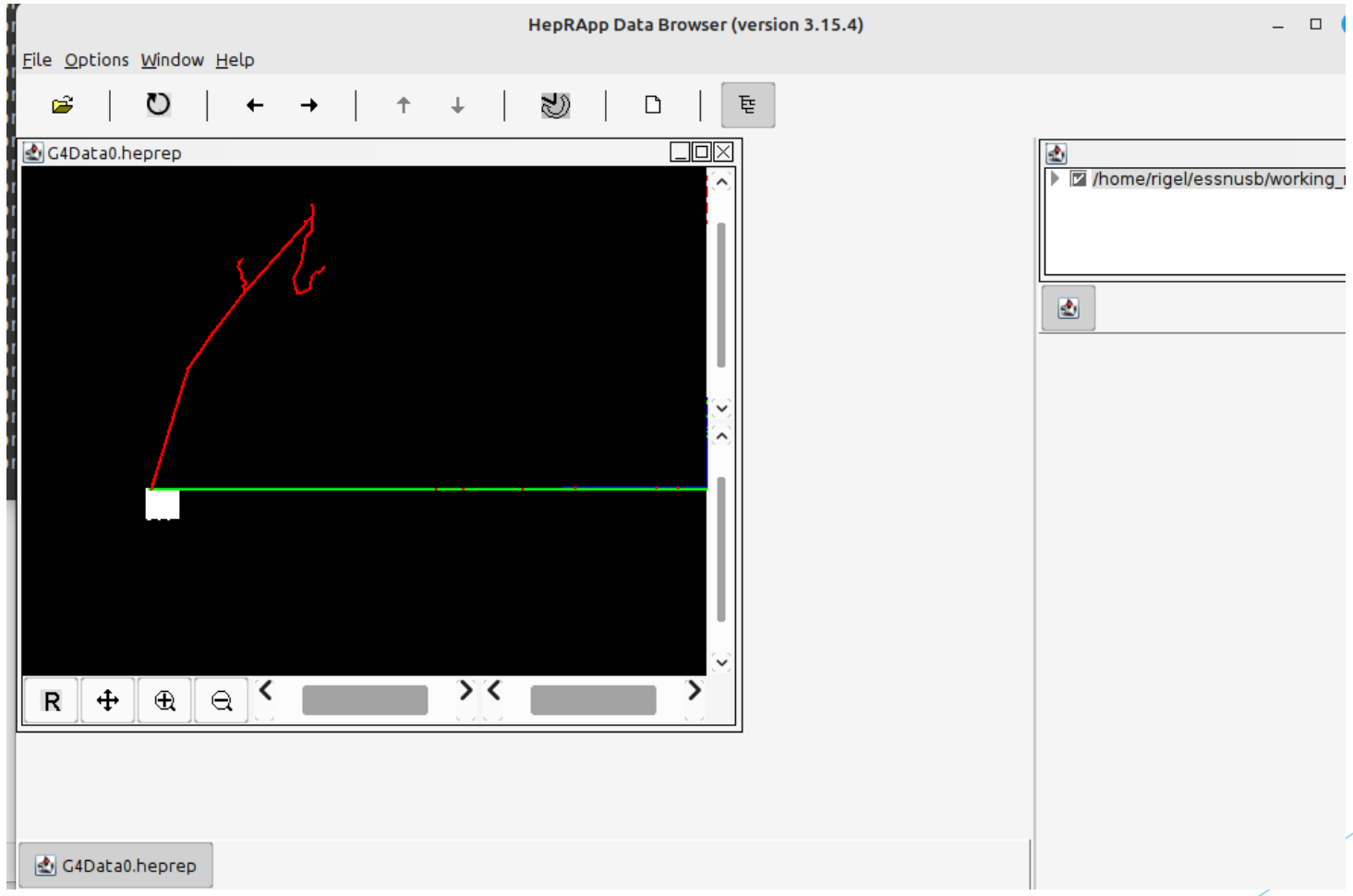
Geant4 Visualization of events

- All visualization is made using Geant4 macros

core > vis > OpenVis.mac

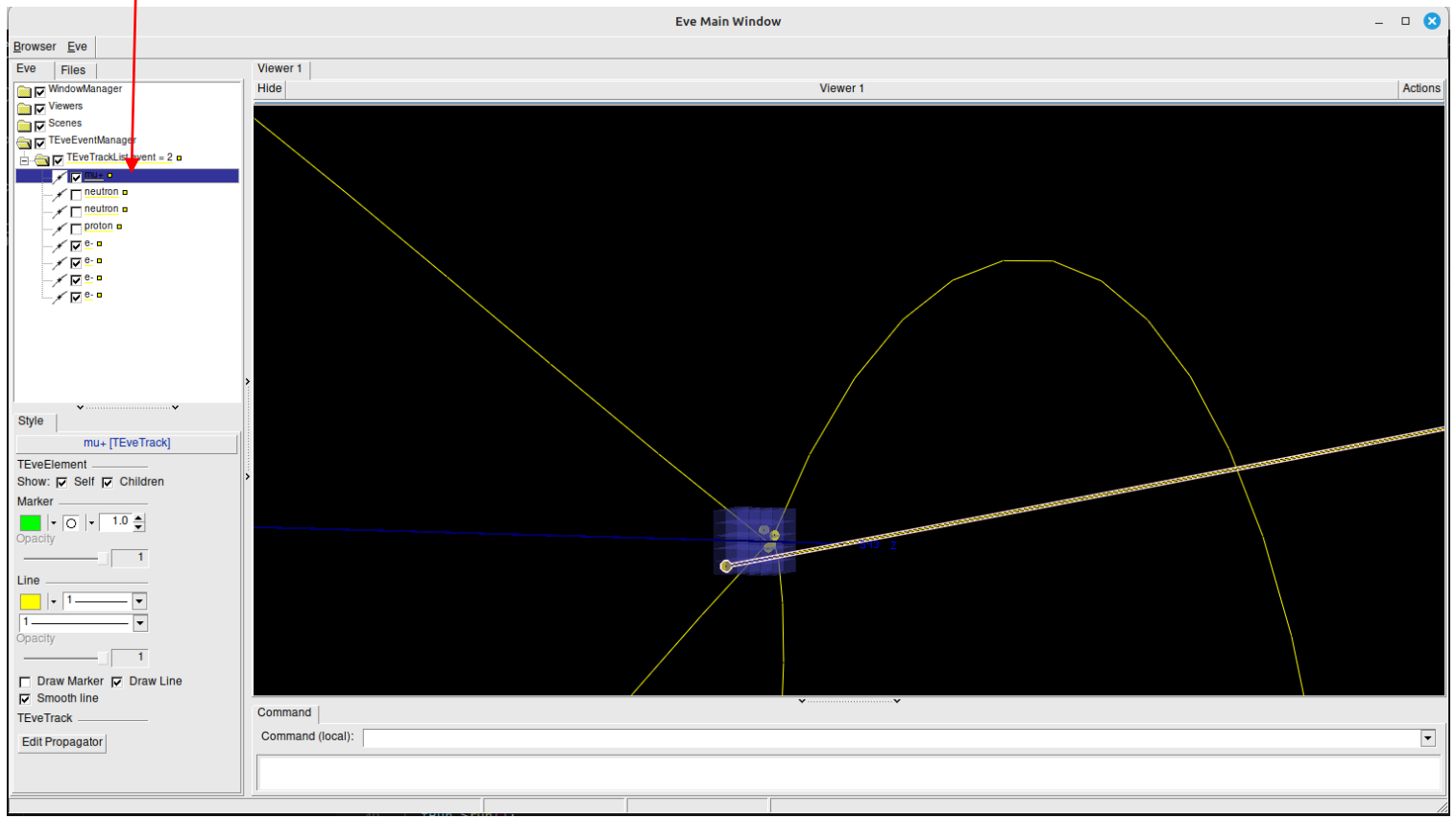
```
/control/verbose  1
/run/verbose 1
/event/verbose    0
/tracking/verbose 0
/run/initialize
#/detector/reflection none

/run/beamOn 5
#
/control/verbose 1
/run/verbose 2
#
# Use this open statement to create an OpenGL view:
#/vis/open OGLIQt 1200x600-0+0
/vis/open OIX 600x600-0+0
#/vis/open TSGQt 600x600-0+0


# For qt
#/vis/open OGL      #(Generic way. For Stored mode if you have define your G4VIS_USE_QT variable)
#/vis/open OGLI     #(for Immediate mode)
#/vis/open OGLS     #(for Stored mode)
#/vis/open OGLIQt   #(for Immediate mode)
#/vis/open OGLSQt   #(for Stored mode)
#   OpenGLImmediateQt (OGLIQt, OGLI)
#   OpenGLStoredQt (OGLSQt, OGL, OGLS)
#   OpenGLImmediateX (OGLIX, OGLIQt_FALLBACK)
#   OpenGLStoredX (OGLSX, OGLSQt_FALLBACK)

```

Geant4 Visualization of events - HepRapp



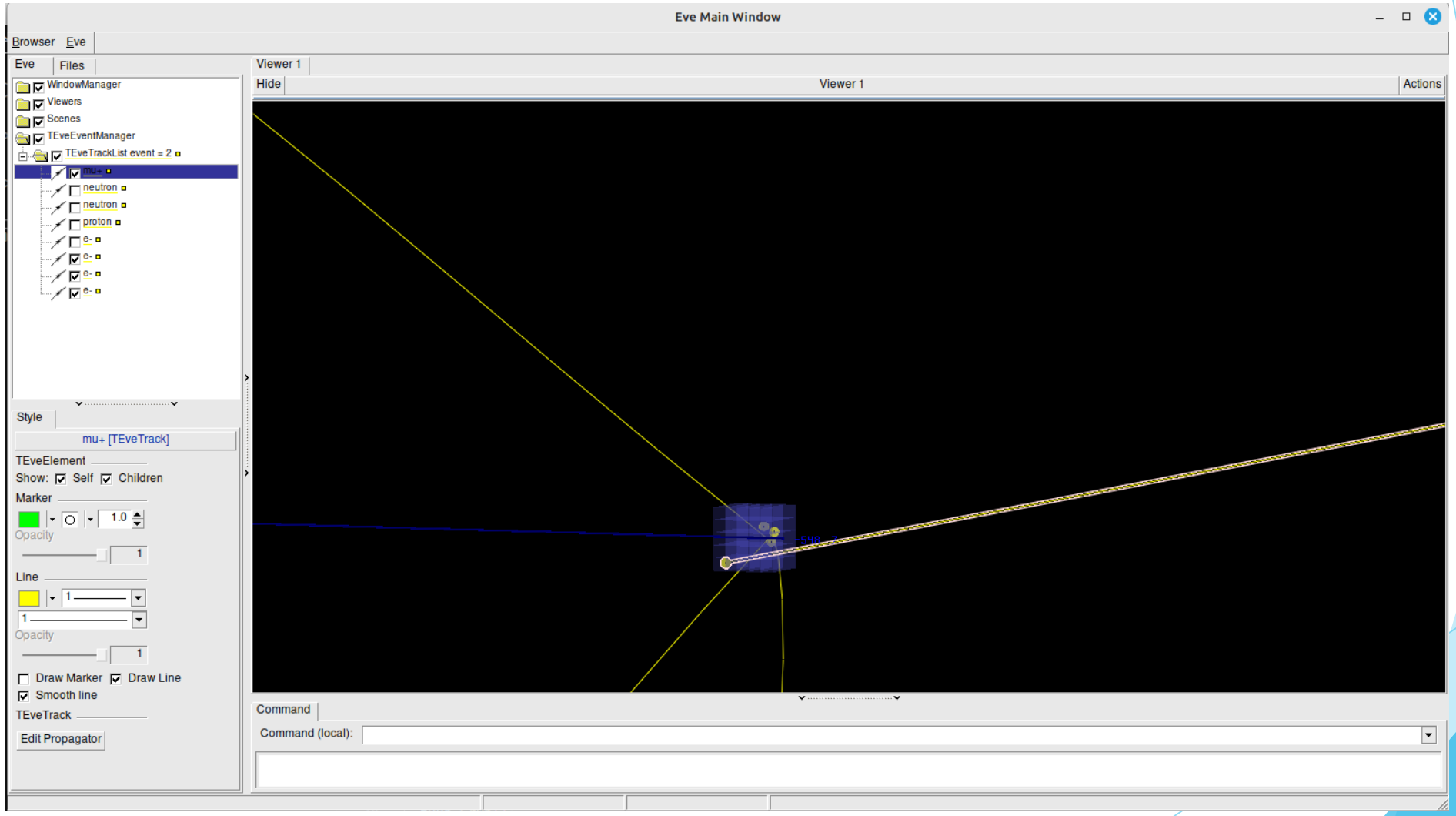
Cern Root visualization of events

They can be turned on/off for better visualization



After running the simulation Track data is exported and then For each event the tracks can Be visualized.s

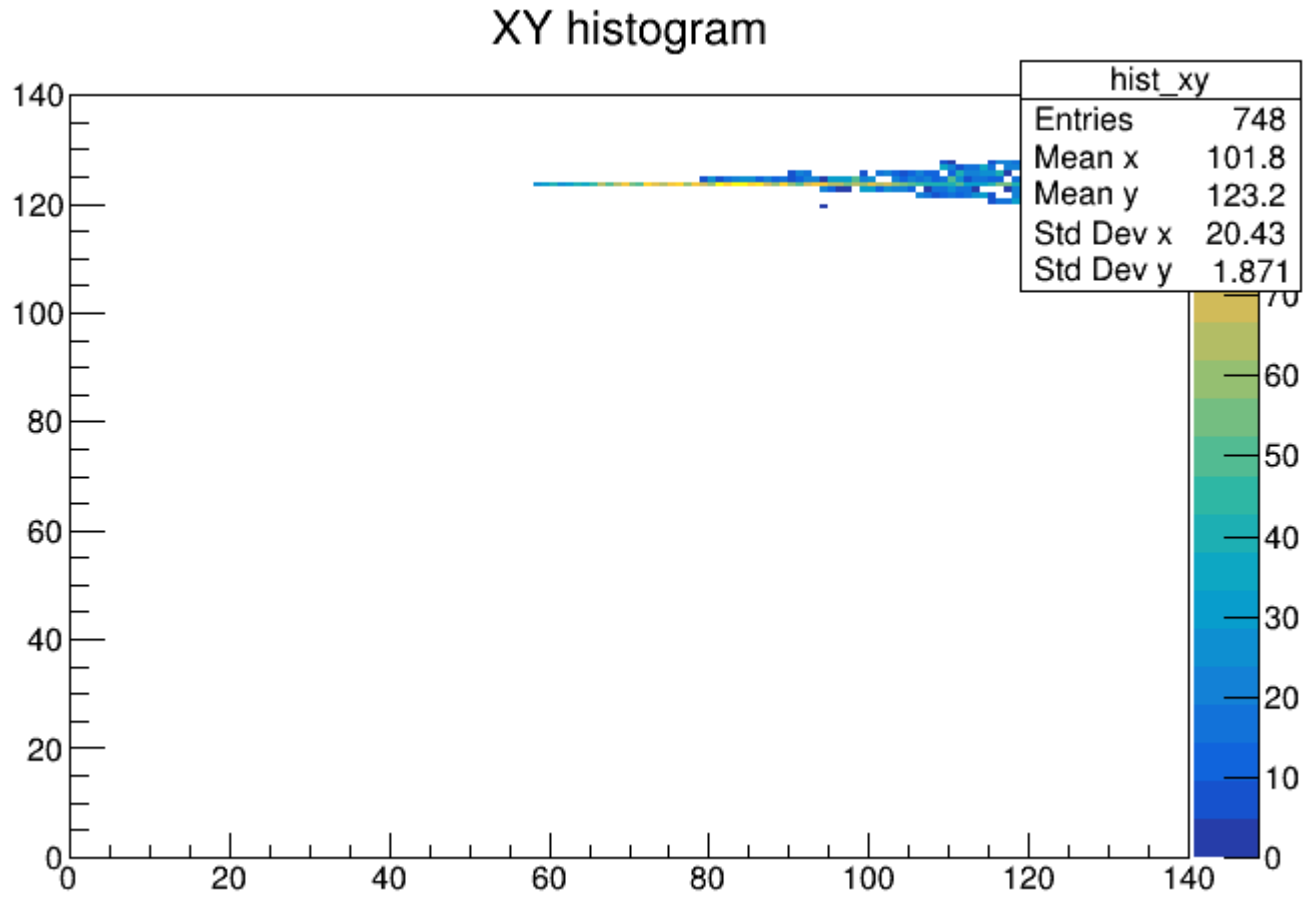
Cern Root visualization of events



Data processing and export – third party

- When the simulation is done, data is exported.
  The user can decide if this data can be used by a third party tool to do some other types of visualization of data processing.

Data processing and export – third party



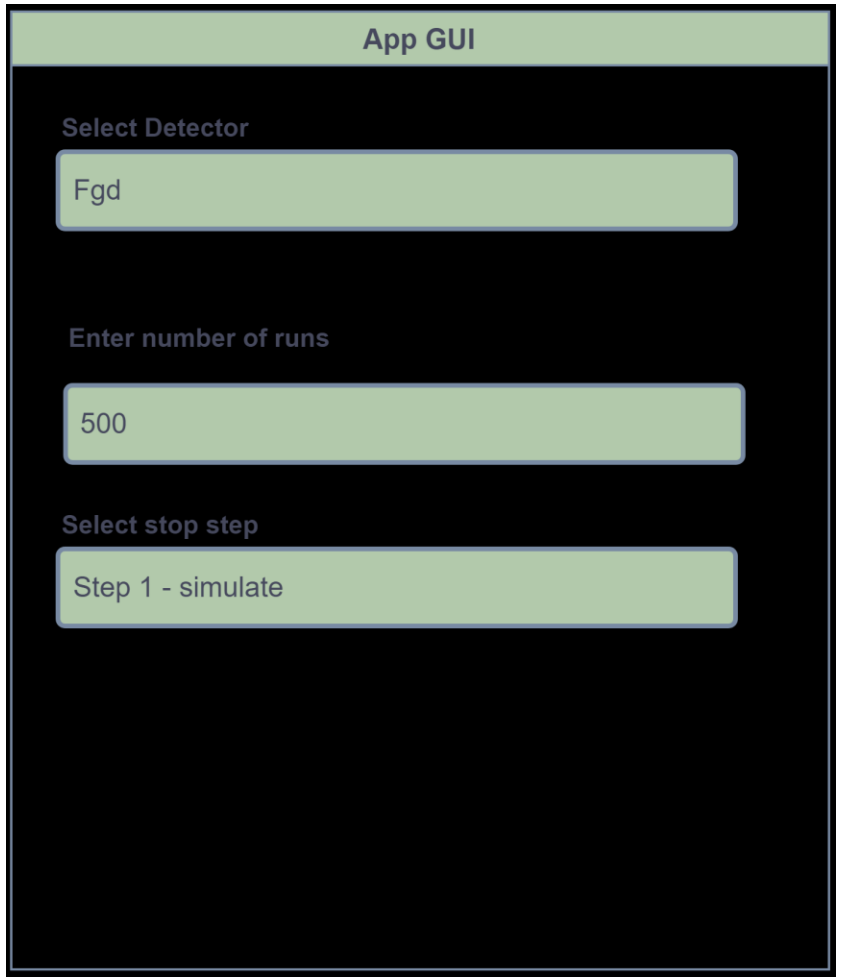
This is using cern root 3d histogram
A XY view of the detector, showing
The photons registered by the
photomultiplier.

Cern Root visualization of events –a few issues left to resolve

- The event manager has to navigate throught the events
- The tracks are not always display as expected (some additional points are added)

NOTE: The visualization is not still a 100% reliable

GUI working (under design)



For more easy working with the framework, a GUI can be added
So that users just add what they
Are interested in not to worry
About setting config pats.

Drawbacks: For each detector added
The GUI may have to be changed
depending on the requirements

Questions?