Geant4 simulation demo

We decided against trying to do a hands on Geant 4 simulation example. These tend to be very time consuming,

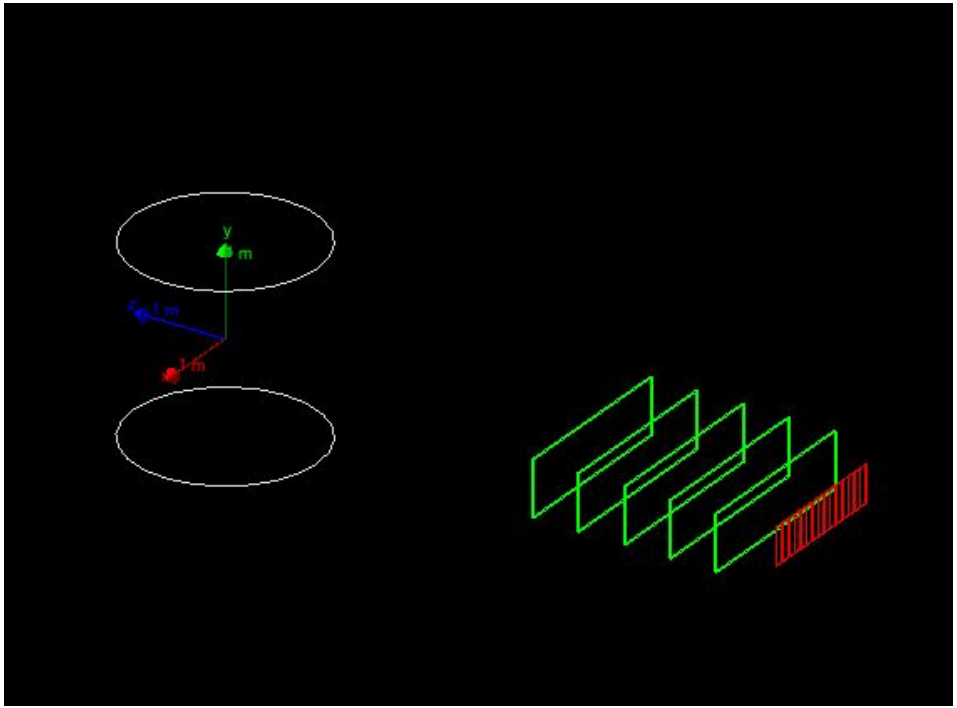
Instead I will do a small demo and illustrate how each part works.

Today’s example

Extend a single arm spectrometer to two arms, including a calorimeter in the second arm.

Goals:

- Explore the code structure
- Run code example and explore code options
- Consider the changes needed to extend this example to include a second arm



Main code components

Program main (tutorial.cc)

Interface directory (include/*.hh)

Source file directory (src/*.cc)

Cmake infrastructure CMakeLists.txt and GNUmakefile)

Runtime input files (.mac)

Let’s inspect the main program

HandsOn3/tutorial.cc

Include directory

ActionInitialization.hh
Analysis.hh
CellParameterisation.hh
DetectorConstruction.hh
EventAction.hh
HodoscopeHit.hh
HodoscopeSD.hh
MagneticField.hh
PrimaryGeneratorAction.hh

Src directory

Essentially the same structure as the interface area

ActionInitialization.cc
CellParameterisation.cc
DetectorConstruction.cc
EventAction.cc
HodoscopeHit.cc
HodoscopeSD.cc
MagneticField.cc
PrimaryGeneratorAction.cc

Mac files

draw.mac
drawSlice.mac
gui.mac
icons.mac
init.mac
init_vis.mac
run1.mac
run2.mac
scoring.mac
Vis.mac

“.mac” is the conventional file ending for Geant4 run-time control files. These can be simple or complex (the ones in my example are complex..)

Before we look through them, lets look at the main program

Ok, lets build the code and run it

```
#Standard compilation method for Geant4 examples

cd HandsOn3

cmake .

make -f Makefile


#and Run

./SLACtut
```

Ok, now the exercise is to add the second spectrometer including a calorimeter.

Lets compare the code changes needed for this

diff -qr HandsOn3 HandsOn3-solution

Now lets look at the details

diff -W 200 --side-by-side HandsOn3/src/DetectorConstruction.cc HandsOn3-solution/src/DetectorConstruction.cc | more
diff -W 200 --side-by-side HandsOn3/src/EventAction.cc HandsOn3-solution/src/EventAction.cc
diff -W 200 -side-by-side HandsOn3/src/HodoscopeSD.cc HandsOn3-solution/src/HodoscopeSD.cc
diff -W 200 -side-by-side HandsOn3/src/HodoscopeHit.cc HandsOn3-solution/src/HodoscopeHit.cc

Ok, lets build the new code and run it

```
#Standard compilation method for Geant4 examples

cd HandsOn3-solution

cmake .

make -f Makefile


#and Run

./SLACtut
```

Links to more Geant4 examples and tutorials

https://geant4.web.cern.ch/docs/tutorials

https://geant4.web.cern.ch/docs/advanced_examples_doc/index

My material is from
https://www.slac.stanford.edu/xorg/geant4/Valencia2021/HandsOn3/

More information - My Mac setup for Geant4

Docker image: You may be able to locally compile Geant4 or use a library built by your experiment (or conda). For this demo, I used a docker image from https://gitlab.mpcdf.mpg.de/rgaida/geant4-docker.

Getting the Qt graphics to work proved to be difficult..

- It doesn't appear to be possible from Jupyter (with the current graphics of Geant)

Tricks needed to set up my laptop to run this example via docker included the following:
(also useful for running on a remote server, such as lxplus at CERN)

- Install XQuartz, and change its settings to "Allow connections from network clients"
- Doing "xhost +localhost" (needed if you stop/restart xquartz)
- Doing "defaults write org.xquartz.X11 enable_iglx -bool true" (likely a one time command)
- Then specifically for Docker: docker run -it --rm --name geant4 -h container -e DISPLAY="host.docker.internal:0" -v $PWD/geant4:/var/geant4/workspace -v /tmp/.X11-unix:/tmp/.X11-unix -t gitlab-registry.mpcdf.mpg.de/rgaida/geant4-docker