Simulation Computing Workshop
Hands-on session

Sayak Chatterjee

UMass, Amherst

May 26, 2023

Plans for the session

- **Setting up the environment for using REMOLL**
  - Prerequisites
  - Local machine (Linux)
  - Ifarm (Jlab computing account is needed to access)
  - Virtual box
- **Visualization of MOLLER experimental setup**
  - Using the REMOLL Graphical User Interface (GUI)
- **Run simulation with REMOLL**
  - How to submit jobs to simulate some events
- **Investigation of the simulated data**
  - Structure of the simulated data
- **Modify the simulation geometry**
  - Add or modify some materials (active or passive)
- **Analysis of the simulated data**
  - Final plots

Setting up the environment to use REMOLL

Linux (Local machine)

Prerequisites: git, cmake > 3.5, Geant4 >= 4.10.00 (>= 4.10.06 recommended), ROOT >= 6.0.0, python

Link of how to: git, cmake, GEANT4, ROOT, python

Once you have all the prerequisites, follow the below steps to download and compile REMOLL

Step 1: **git clone https://github.com/JeffersonLab/remoll directory_name**

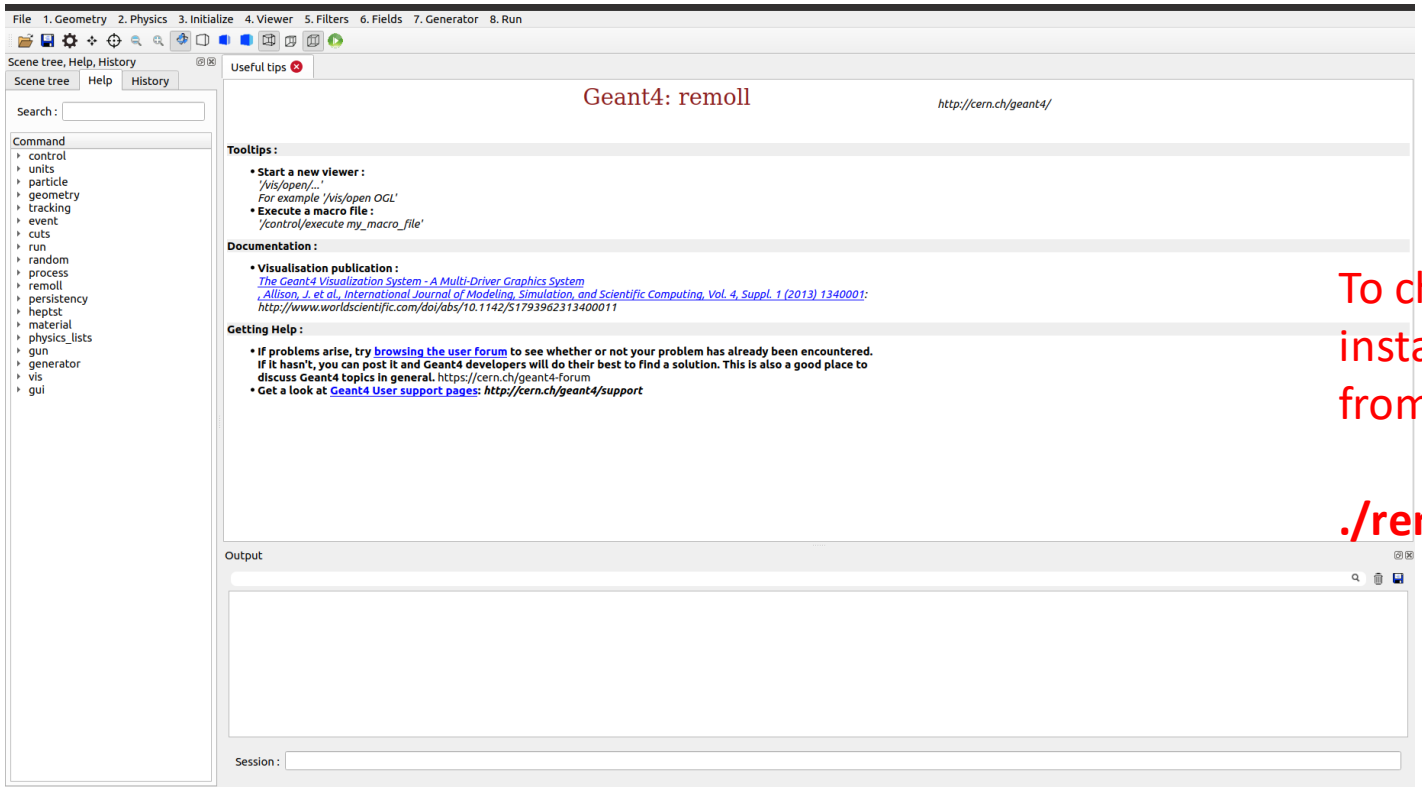
Step 2: **mkdir build**

Step 3: **cd build**

Step 4: **cmake ..**

Step 5: **make –j5**

Step 6: **make install**



To check if you have a successful installation, type the following from your **build** directory:

./remoll

Ifarm (JLab computing account)

How to get an account on ifarm:

- To get a JLab user account, each new user must fill out the Access Registration Form. Be sure to fill out (and remember) the emergency contact information requested - the information will be used for identification later.
- Send an email to helpdesk@jlab.org and ask to create an account on ifarm. You will need to provide your name, your supervisor and instituition name, and also mention that you are part of the moller12gev group under halla.
- Complete the GEN034U training

Login to ifarm:

- Requires two factor authentication (Details here)
- ssh -Y username@scilogin.jlab.org (put your set authentication password)
- ssh -Y ifarm (put your ifarm password)
- cd /work/halla/moller12gev/username
- source /site/12gev_phys/softenv.csh 2.5
- Downloading and compiling of REMOLL is the same as local machine

Virtual Box

- Download and install the VirtualBox Manager
- Download Virtual Machine (11 GB)
- Start the VirtualBox Manager application and from the **File** menu select **Import Appliance**
- Select the Virtual Machine file MOLLER_VM.ova from the file browser
- Click **Next**
- Click **Import** and at the end you should see the virtual machine **MOLLER_VM** displayed on the left side of the VirtualBox application
- Click the virtual machine **MOLLER_VM** and this will boot the Ubuntu virtual machine with all the software installed.

Visualization of the MOLLER experimental setup

Visualization

Using GUI:

Goto following directory by: cd softwares/remoll

Run the command: ./build/remoll

Click menu 1.Geometry and select MOLLER Experiment

Click menu 3. Initialize and select Initialize

Click menu 4. Viewer and select Qt Viewer (Stored) (second item)

Now MOLLER geometry should be accessible

Using commands:

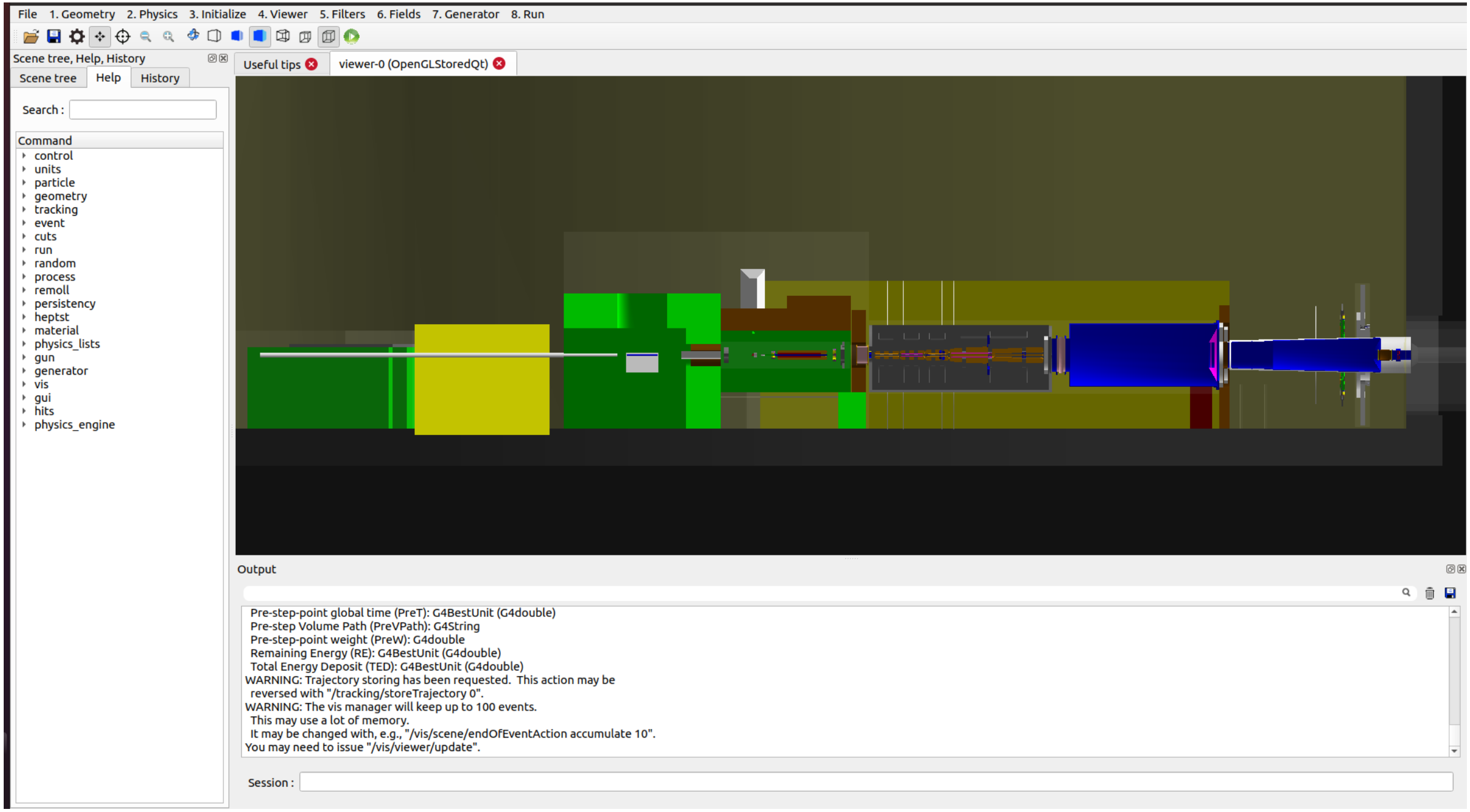
/remoll/geometry/setfile geometry/mollerMother.gdml

/run/initialize

/vis/open OGL

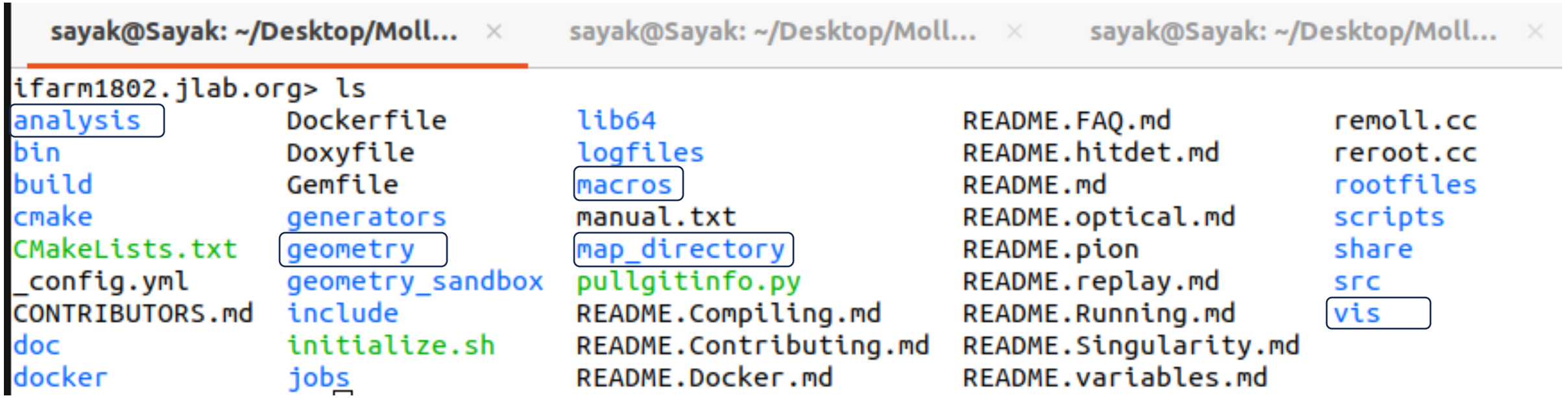
/control/execute vis/vis.mac

Visualization



Running simulation

Where to find what?



geometry -> Contains all the GDML files for the individual sub-systems

Position of the different sub-systems: geometry/positions.xml

Geometry for the sensitive volumes: geometry/mollerParallel.gdml

map_directory -> Contains the magnetic field files

macros -> Different macros to run simulations

analysis -> Contains various example analysis scripts

vis -> Different macros for visualization

Simulation in batch mode

- Simulate multiple events according to input macro file to generate the root output
- Input macro file provide what physics models, event types to generate, geometry, magnetic field, output destination, and number of events to simulate
- Standard input files are available in macros directory remoll/macros
- Hands-On-Remoll directory has three such files that we will use HandsOn_run_moller.mac, HandsOn_run_ep.mac HandsOn_kryptonite.mac
- Copy these files to you VM or to ifarm remoll/macros
  - For VM, just open a browser in your VM and download the files
  - For ifarm (from local machine to ifarm):
    scp -r file_name username@login1.jlab.org:/u/home/username/
  - For ifarm (from ifarm to local machine):
    scp -r username@login1.jlab.org:/u/home/username **directory_at_the_local_machine**

Standard input macro file (.mac)

HandsOn_run_moller.mac

```
1 #  Hands-On tutorial moller generator macro file
2
3
4 # This must be called before initialize
5 /remoll/geometry/setfile geometry/mollerMother.gdml
6 # Parallel world geometry is optional - detector 28 (the primary detector array's idealize vacuum detector) is included in this parallel world now.
7 /remoll/parallel/setfile geometry/mollerParallel.gdml
8
9 /remoll/physlist/parallel/enable
10
11 # This must be explicitly called
12 /run/initialize
13
14 /remoll/printgeometry true
15
16 /control/execute macros/load_magnetic_fieldmaps.mac
17
18 # Raster and initial angle stuff
19
20 /remoll/oldras true
21 /remoll/rasx 5 mm
22 /remoll/rasy 5 mm
23
24
25 /remoll/evgen/set moller
26 /remoll/evgen/thcommin 80.0 deg
27 /remoll/evgen/thcommax 100.0 deg
28
29 /remoll/beamene 11 GeV
30
31 /remoll/beamcurr 85 microampere
32
33 # Make interactions with W, Cu, and Pb
34 # realistic rather than pure absorbers
35 #/control/execute macros/HandsOn_kryptonite.mac
36
37 ##disable all detectors,
38 /remoll/SD/disable_all
39 /remoll/SD/enable 28
40 /remoll/SD/enable 47
41 /remoll/SD/print_all
42
43 /process/list
44
45 # Specify random number seed
46 #/remoll/seed 123456
47
48 /remoll/filename remollout_Moller_gen_2k.root
49 /remoll/target/print
50 /run/beamOn 2000
```

Geometry files

Enable physics list

Initialization

Magnetic field

Raster settings

Event generators

Beam parameters

Setting Kryptonite material

Detector list

Output file

events to be simulated

Modifying the Input macro file

- Simulation output file size can be reduced, if you know what sensitive detectors you want to record data

Be

- We can first disable all the detector
- disable all detectors, /remoll/SD/disable_all
- Then enable ones you want

  /remoll/SD/enable 28
  /remoll/SD/enable 470

- You can control what data you want to record for each detector

  /remoll/SD/detect lowenergyneutral 28
  /remoll/SD/detect secondaries 28
  /remoll/SD/detect boundaryhits 28

- Enable kryptonite feature

  /remoll/kryptonite/enable

- Then you can either turn certain materials into kryptonite

  /remoll/kryptonite/add Tungsten
  /remoll/kryptonite/add Copper

- You can also turn certain volumes into kryptonite by giving the name of the GDML solid shape

  /remoll/kryptonite/volume name_of_the_GDML_solid_shape

How to run the simulation

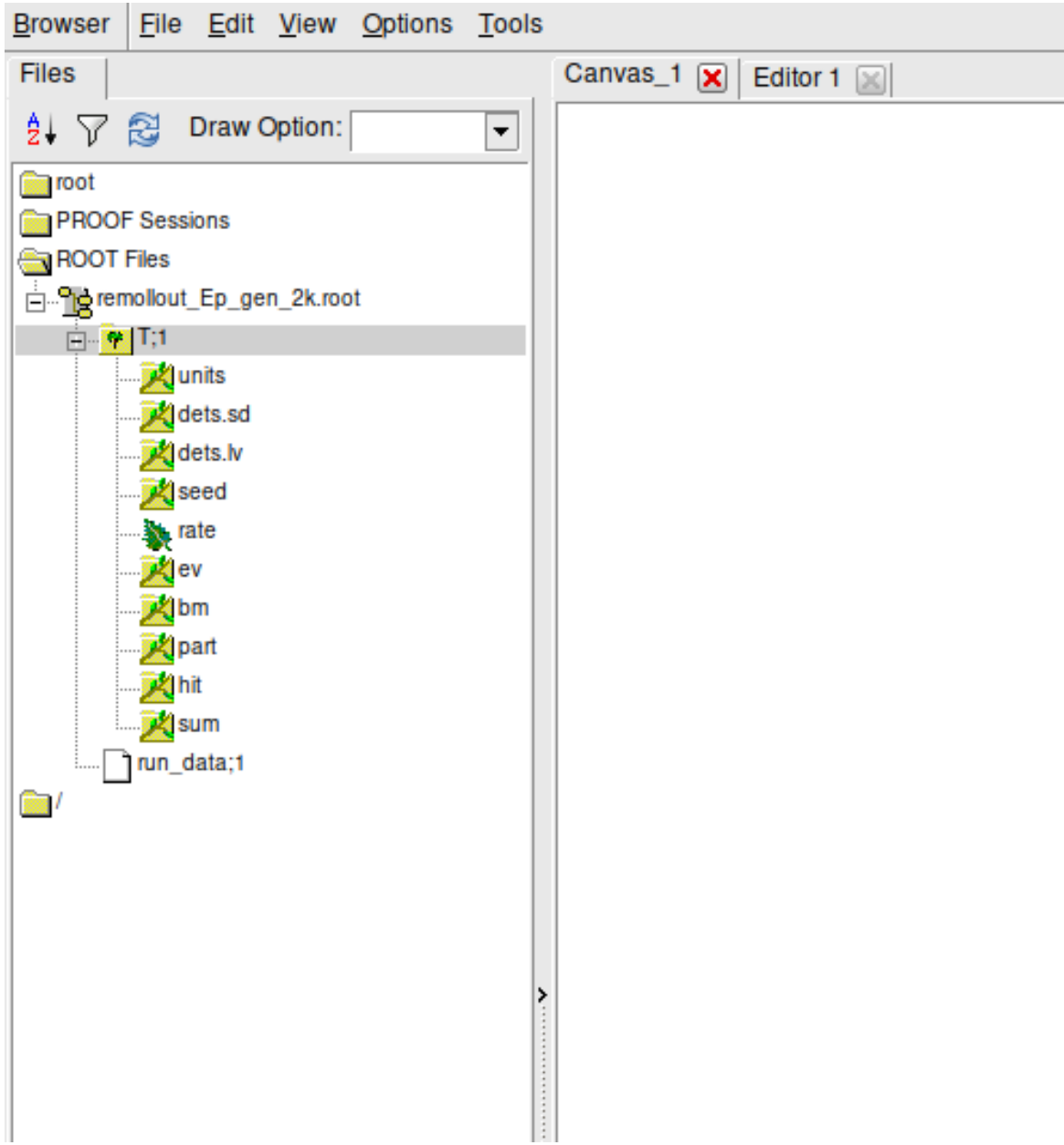
- Goto remoll directory
- ../build/remoll destination+macro file name
- This will start the simulation in batch mode
- We will do following simulations
    a. Run 2000 moller electrons events
    b. Run 2000 ep elastic electron events
- The output root files from these two jobs are also available at Hands-On Materials (Rootfilesremollout_Moller_gen_2k.root and remollout_Ep_gen_2k.root)
- Let us now try to see how the simulated data look like!

Opening the root file generated after simulation & understanding the data structure

- Open a terminal using Alt+Crtl+T
- Goto ~/softwares/remoll
- Now we will open the root file using
- ./build/reroot HandsOn_remollout_10k.root

  reroot is root compiled with remoll libraries
- Type new TBrowser()

  This will open the browser to see the data inside the root file
- Click on the tree elements (branches) to see the data in the form of histograms
- We can also access the data using the root command line:
  - T->Print()
  - T->Draw ("hit.p","hit.pid==11 && hit.det==28")
  - T->Draw("hit.r","(hit.pid==11 && hit.det==28)*rate",”hist”)
  - Etc...



Particle PID: Link

Some examples on how to visualize the data using the ROOT command line!

Plot hit.vz (start Z vertex) of electrons hitting detector 28:

```
T->Draw("hit.vz","hit.pid==11 && hit.det==28")
```

Plot hit.p (momentum) of electrons hitting detector 28:

```
T->Draw("hit.p","hit.pid==11 && hit.det==28")
T->Draw("hit.p/GeV","hit.pid==11 && hit.det==28")
```

Plot hit.p (momentum) of electrons hitting detector 28 weighted by the rate so higher rate events gets higher weights and lower rate events gets a lower weight:

```
T->Draw("hit.p","(hit.pid==11 && hit.det==28)*rate",”hist”)
```

Plot hit.r (radius of hit) of electrons hitting detector 28 weighted by the rate so higher rate events gets higer weights and lower rate events gets a lower weight:

```
T->Draw("hit.r","(hit.pid==11 && hit.det==28)*rate",”hist”)
```

Plot 2D histogram to see correlation between momentum and radius at the detector 28:

```
T->Draw("hit.r:hit.p","(hit.pid==11 && hit.det==28)*rate",””)
T->Draw("hit.r:hit.p","(hit.pid==11 && hit.det==28)*rate",””)
```

How do we automate these steps? Use a root script (basicRootScript.C -> basic root script that we'll be using)

ROOT scripting: Defining histograms

basicRootScript.C is our template script, we will create histograms of hit radius, xy 2D distribution and source vertex of these hits

Declare 1D histograms for radius and source vertex:

```
TH1D *r;
TH1D *sourceZ;
TH1D *rRate; //for rate weighted radial distribution
```

Declare 2D histograms for XY distribution:

```
TH2D *hXY;
TH2D *hXYrate; //for rate weighted XY distribution
```

Let's define their parameters and create them initHisto() routine:

```
r = new TH1D("r","radial distribution;r[mm]",200,500,1500);
sourceZ = new TH1D("sourceZ","initial vertex for hit ;z position [mm]",5000,-5500,-3500);
hXY = new TH2D("hXY","2D hit distribution;x [mm];y [mm]",200,-2100,2100,200,-2100,2100);
```

Define rate weighted histograms:

```
rRate = new TH1D("rRate","rate weighted distribution;r[mm]",200,500,1500);
hXYrate = new TH2D("hXYrate","rate weighted 2D hit distribution;x [mm];y [mm]",200,-2100,2100,200,-2100,2100);
```

ROOT scripting: Filling the histograms, scaling & plotting

Let's fill these histograms with data from the Tree in at the processOne(...) routine

```
r->Fill(hit->at(j).r);
sourceZ->Fill(hit->at(j).vz);
hXY->Fill(hit->at(j).x,hit->at(j).y);
rRate->Fill(hit->at(j).r,rate);
hXYrate->Fill(hit->at(j).x,hit->at(j).y,rate);
```

Scale rate weighted histograms if we have used chain of root files (more than one root filed linked) in the void scale() routine:

```
r->Scale(1./nFiles);
sourceZ->Scale(1./nFiles);
hXY->Scale(1./nFiles);
rRate->Scale(1./nFiles);
hXYrate->Scale(1./nFiles);
```

Let's create few canvases to visualize the data:

```
TCanvas *c1 = new TCanvas(); c1->Divide(1,2);
c1->cd(1);   r->DrawCopy();
c1->cd(2);   rRate->DrawCopy();
```

ROOT scripting: Saving plots as a root file

Output written in this step can be accessed later in a root file “basicRootScript.root”:

```
r->Write();
sourceZ->Write();
hXY->Write();
rRate->Write();
hXYrate->Write();
```

Output written in this step can be accessed later in a root file “basicRootScript.root”
This file name is set in the routine:

```
void initHisto()
string foutNm = Form("basicRootScript.root");
```

You can access the saved histograms using the command:

```
root basicRootScript.root
or
./build/reroot basicRootScript.root
```

Executing the analysis script

Load the script BasicRootScript.C:

```
.L  Analysis/basicRootScript.C
```

Execute the script:

```
basicRootScript("HandsOn_remollout_10k.root")
```

Modification of REMOLL geometry

Handling the GDML files

- Main geometry is kept in geometry/mollerMother.gdml parent file
- The parent or mother GDML file has set of daughter volumes containing different regions of the experiment
- A parallel GDML geometry is kept for intercept type sensitive detectors to observe particles crossing certain z-location
- Let us try to create a new sensitive detector of radius 600 mm with detector id of 470 located at 4600 mm in hall coordinates.
    - We use the parallel world to implement it

Steps to implement new sensitive detector in the geometry using GDML

Positioning : In the positions.xml file:

```
<position name="TestSensDetVirtualPlane_pos" z="4600.0" unit="mm"/>
```

Create the solid in the mollerParallel.gdml:

```
<tube name="TestSensDetVirtualPlane_solid" startphi="0" deltaphi="360" aunit="deg" rmax="600" rmin="0"
z="1" lunit="mm"/>
```

Create the logical volume:

```
<volume name="TestSensDetVirtualPlane_log">
  <materialref ref="G4_Galactic"/>
  <solidref ref="TestSensDetVirtualPlane_solid"/>
  <auxiliary auxtype="SensDet" auxvalue="planeDet"/>
  <auxiliary auxtype="DetNo" auxvalue="470"/>
</volume>
```

Create the physical volume that will be placed within the simulation:

```
<physvol name="TestSensDetVirtualPlane_phys">
    <volumeref ref="TestSensDetVirtualPlane_log"/>
    <positionref ref="TestSensDetVirtualPlane_pos"/>
</physvol>
```

Simulation & Analysis

A small simulation to get started with REMOLL

- Run simulation in batch mode to produce 2000 moller events with realistic materials
- Modify input macro file to change some realistic materials into Kryptonite
- Run simulation in batch mode to produce 2000 moller events with some materials converted to Kryptonite
- Run remoll in batch mode:

  ./build/remoll macros/HandsOn_run_moller.mac
- Open macros/HandsOn_run_moller.mac in a text editor
- Remove # to uncomment the line /control/execute macros/HandsOn_kryptonite.mac
- Change the output root file name to remollout_Moller_gen_2k_kryptonite.root
- Run remoll in batch mode:

  ./build/remoll macros/HandsOn_run_moller.mac

The output root files are available to download from here

Analysis of the simulated data

- We will look at Photons (pid=22) hitting the sensitive detector (det id 470) we created previously using GDML
- Use RootScript.C as a template script and create a root script relevant for above study
- Declare 1D & 2D histograms for radius, source vertex and XY distributions

```
TH1D *r;
TH1D *sourceZ;
TH1D *rRate; //for rate weighted radial distribution
TH2D *hXY;
TH2D *hXYrate; //for rate weighted XY distribution
```

- Defining the histograms

```
r = new TH1D("r","Det470 radial distribution;r[mm]",200,0,600);
rRate = new TH1D("rRate","Det 470 rate weighted distribution;r[mm]",200,0,600);
hXY = new TH2D("hXY","2D hit distribution;x [mm];y [mm]",200,-600,600,200,-600,600);
hXYrate = new TH2D("hXYrate","rate weighted 2D hit ditribution;x [mm];y [mm]",200,-600,600,200,-600,600);
sourceZ = new TH1D("sourceZ","initial vertex for hit ;z position [mm]",10000,-5300,8000);
```

- Cuts

```
if(hit->at(j).pid!=22) continue;
if(hit->at(j).det != 470) continue;
```

Analysis of the simulated data

- Fill the histograms

```
r->Fill(hit->at(j).r);
sourceZ->Fill(hit->at(j).vz);
hXY->Fill(hit->at(j).x,hit->at(j).y);
rRate->Fill(hit->at(j).r,rate);
hXYrate->Fill(hit->at(j).x,hit->at(j).y,rate);
```

- Create a canvas

```
Double_t w = 600;//width
Double_t h = 600;//height
TCanvas *p1 = new TCanvas("TCan_sourceZ","Source Z Canvas",w,h);
```

Analysis of the simulated data

- Use the function: gStyle->SetOptStat("nemr"); to format histogram stat box information: n-name, e-events, m-mean, r-rms

- Draw the vertex histogram on the p1 Canvas :

```
sourceZ->DrawCopy();
```

- You can save the canvas as an image to formats including pdf or png:

```
p1->SaveAs("TCan_sourceZ.png");
```

- Canvas with multiple pads:

```
TCanvas *p2 = new TCanvas("TCan_rate_xy","Radial and XY hits",w,h);
p2->Divide(2,2);
p2->cd(1);
r->DrawCopy();
p2->cd(2);
rRate->DrawCopy();
p2->cd(3);
hXY->DrawCopy();
p2->cd(4);
hXYrate->DrawCopy();
```

Analysis of the simulated data

- How to run the analysis script
  - We will first run the script RootScript.C with root file remollout_Moller_gen_2k.root
  - Give a unique name at string

    foutNm = Form("RootScript_real.root");
  - Load the script RootScript.C

    .L analysis/RootScript.C
  - Execute the script

    RootScript(“remollout_Moller_gen_2k.root”)

So now you have all the plots or at least you know how to make your required plots, so the next task is to explain them!!!