GEANT4/ROOT HANDS-ON SESSION

Computing Workshop 9/6/2024

ericking@temple.edu

TEMPLE UNIVERSITY

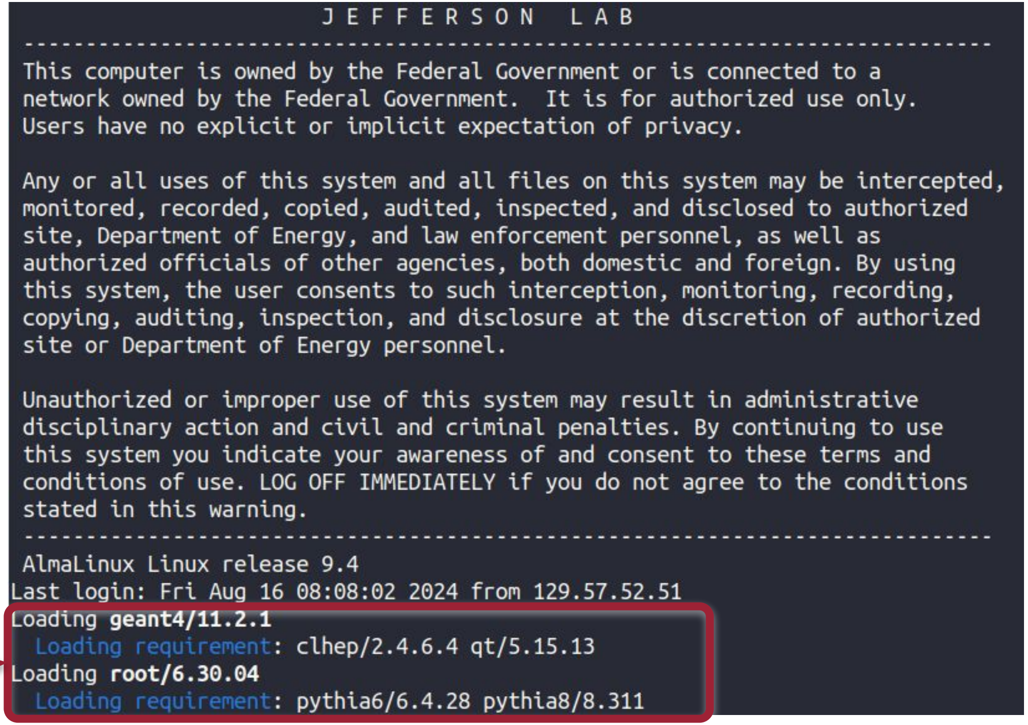
ifarm Setup – Modules

- Hands-on work has been developed with Geant4.11.2.1
- ROOT hands-on work was developed with ROOT 6

Your .cshrc file should contain the following:

```
module use /group/halla/modulefiles
module load geant4/11.2.1
module load root/6.30.04
```

You should see the following successful load messages as seen on the right.



Cloning Hands-on Example from Github

- Make a working directory for yourself

- Clone repository

- Move into cloned directory

- Create a directory named build

- Build your makefile (CMakeLists.txt one directory down)

- Build the application

```
ericking@ericking-G5-5000:~$ mkdir hands-on-session
ericking@ericking-G5-5000:~$ cd hands-on-session/
ericking@ericking-G5-5000:~/hands-on-session$ git clone https://gith
Cloning into 'workshopGeant4Example'...
remote: Enumerating objects: 48, done.
remote: Counting objects: 100% (48/48), done.
remote: Compressing objects: 100% (27/27), done.
remote: Total 48 (delta 22), reused 44 (delta 21), pack-reused 0 (fr
Receiving objects: 100% (48/48), 20.71 KiB | 2.59 MiB/s, done.
Resolving deltas: 100% (22/22), done.
ericking@ericking-G5-5000:~/hands-on-session$ ls
workshopGeant4Example
ericking@ericking-G5-5000:~/hands-on-session$ cd workshopGeant4Examp
ericking@ericking-G5-5000:~/hands-on-session/workshopGeant4Example$ ls
CMakeLists.txt  gui.mac  init_vis.mac  plotNtuple.C  run1.mac  src      workshopCodeBlocks     workshopExample2.mac
GNUmakefile     include  plotHisto.C   README.md     run2.mac  vis.mac  workshopExample1.mac   workshopExample.cc
ericking@ericking-G5-5000:~/hands-on-session/workshopGeant4Example$ mkdir build
ericking@ericking-G5-5000:~/hands-on-session/workshopGeant4Example$ cd build
```

If you've properly cloned you should see a screen something like this.

```
git clone https://github.com/dericking/workshopGeant4andROOT
cd workshopGeant4andROOT
mkdir build
cd build
cmake ../
make
```

After building the application

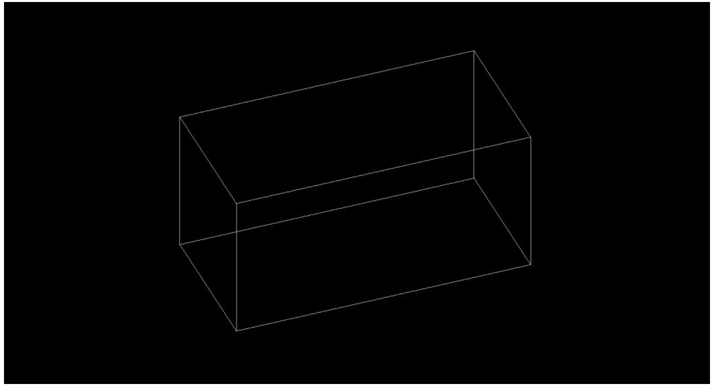
If your application has successfully build you'll see something like the following:

```
ericking@ericking-G5-5000:~/hands-on-session/workshopGeant4Example/build$ make
[  9%] Building CXX object CMakeFiles/WorkshopExample.dir/workshopExample.cc.o
[ 18%] Building CXX object CMakeFiles/WorkshopExample.dir/src/ActionInitialization.cc.o
[ 27%] Building CXX object CMakeFiles/WorkshopExample.dir/src/DetHit.cc.o
[ 36%] Building CXX object CMakeFiles/WorkshopExample.dir/src/DetSD.cc.o
[ 45%] Building CXX object CMakeFiles/WorkshopExample.dir/src/DetectorConstruction.cc.o
[ 54%] Building CXX object CMakeFiles/WorkshopExample.dir/src/EventAction.cc.o
[ 63%] Building CXX object CMakeFiles/WorkshopExample.dir/src/FluxHit.cc.o
[ 72%] Building CXX object CMakeFiles/WorkshopExample.dir/src/FluxSD.cc.o
[ 81%] Building CXX object CMakeFiles/WorkshopExample.dir/src/PrimaryGeneratorAction.cc.o
[ 90%] Building CXX object CMakeFiles/WorkshopExample.dir/src/RunAction.cc.o
[100%] Linking CXX executable WorkshopExample
[100%] Built target WorkshopExample
ericking@ericking-G5-5000:~/hands-on-session/workshopGeant4Example/build$
```

You should now have an executable in your directory called '**WorkshopExample**'

When you execute the application you should get the visualization of an empty World volume.

```
./WorkshopExample
```



Geometry Setup

Open up the file src/DetectorConstruction.cc in your favorite editor

⇒ I've simplified the Geant4 Basic B4c example and cleaned up the Detector Construction code.

To start we:

- Call NIST manager for materials

- Define useful variables

- Construct World volume

```
G4VPhysicalVolume* DetectorConstruction::Construct()
{
  ///////////////////////////////////////////////////////////////////////////////
  // Create instance of NIST manager
  auto nistManager = G4NistManager::Instance();

  ///////////////////////////////////////////////////////////////////////////////
  // Some variables that will tidy up code later A,Z,density,fracMass (maybe show new material)
  G4double a,density,fracMass;
  G4int z;

  ////////////////////////////////////////////////////////////////////////////////
  // Define the Geometry

  ///////////////////////////////////////////////////////////////////////////////
  // World
  auto worldSizeXY = 1*m;
  auto worldSizeZ  = 2*m;
  nistManager->FindOrBuildMaterial("G4_AIR");
  G4Material * defaultMaterial = G4Material::GetMaterial("G4_AIR");
  G4Box * worldS = new G4Box("World",worldSizeXY/2., worldSizeXY/2., worldSizeZ/2.);
  G4LogicalVolume * worldLV = new G4LogicalVolume(worldS,defaultMaterial,"World");
  auto worldPV = new G4PVPlacement(nullptr,G4ThreeVector(),worldLV,"World",nullptr,false,0,fCheckOverlaps);
```

Code above already provided.

Geometry Setup - World Volume (as an example)

- Declare world size

- Use the NIST manager to find our volumes material.

- Assign the Material to a pointer

- Define a solid for the World

- Define a logical volume for the World

- Place the World

- Note: This is the only physical volume that gets a name and is returned at the end of Construct()

```
///////////////////////////////////////////////////////////////////////////////
// Define the Geometry

//////////////////////////////////////////////////////////////////////////////
// World
auto worldSizeXY = 1*m;
auto worldSizeZ  = 2*m;
nistManager->FindOrBuildMaterial("G4_AIR");
G4Material * defaultMaterial = G4Material::GetMaterial("G4_AIR");
G4Box * worldS = new G4Box("World",worldSizeXY/2., worldSizeXY/2., worldSizeZ/2.);
G4LogicalVolume * worldLV = new G4LogicalVolume(worldS,defaultMaterial,"World");
auto worldPV = new G4PVPlacement(nullptr,G4ThreeVector(),worldLV,"World",nullptr,false,0,fCheckOverlaps);
```

G4Box('Solid-Name', halfLengthX, halfLengthY, halfLengthZ)

G4LogicalVolume(Solid , Material , 'LV-Name')

G4PVPlacement(Rotation, Position, LogicalVolume, 'PV-Name', MotherLogVol, false, copy-number, overlapChecker);

Geometry Setup - Let's Stick in Some Detector Material

- Fetch Polyethlene from the NIST manager and assign to a G4Material object.

- Solid (Box): X/Y dimensions 90% of the world size; Z dimension 10cm

- Declare logical volume, please give it the name "detectorLV"

- Place it at a Z position of 85% the total distance in the +Z direction.

Geometry Setup - Let's Stick in Some Detector Material

- Fetch Polyethlene from the NIST manager and assign to a G4Material object.
- Solid (Box): X/Y dimensions 90% of the world size; Z dimension 10cm
- Declare logical volume, please give it the name "detectorLV"
- Place it at a Z position of 85% the total distance in the +Z direction.

```
///////////////////////////////////////////////////////////////////////////////
// HANDS-ON #1: Detector/Calorimeter
// Create Lead-Scintillating Fiber Detector material
// FETCH POLYETHLENE FROM NISTMANAGER
nistManager->FindOrBuildMaterial("G4_POLYETHYLENE");
G4Material * matPE = G4Material::GetMaterial("G4_POLYETHYLENE");

// DEFINE DETECTOR
G4double detLx = worldSizeXY * 0.9;
G4double detLy = worldSizeXY * 0.9;
G4double detPolyLz = 10.*cm;

G4Box * detectorSolid = new G4Box("detector",detLx/2.,detLy/2.,detPolyLz/2.);
G4LogicalVolume * detectorLV = new G4LogicalVolume(detectorSolid,matPE,"detectorLV");
G4double detectorZpos = worldSizeZ/2.0*0.85;
new G4PVPlacement(0,G4ThreeVector(0,0,detectorZpos),detectorLV,"detectorPV",worldLV,false,0,fCheckOverlaps);
///////////////////////////////////////////////////////////////////////////////
```

Your code should look something like this. See codeblock #1 in the repository directory

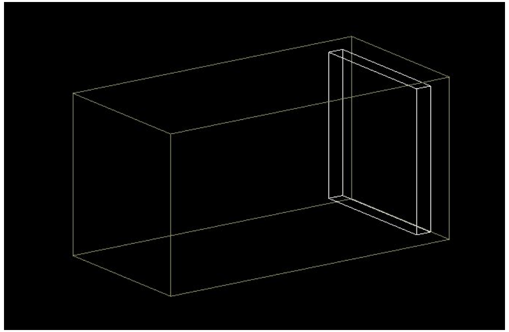
Geometry Setup - Let's Stick in Some Detector Material

- Fetch Polyethlene from the NIST manager and assign to a G4Material object.
- Solid (Box): X/Y dimensions 90% of the world size; Z dimension 10cm
- Declare logical volume, please give it the name "detectorLV"
- Place it at a Z position of 85% the total distance in the +Z direction.

```
//////////////////////////////////////////////////////////////////////////////
// HANDS-ON #1: Detector/Calorimeter
// Create Lead-Scintillating Fiber Detector material
// FETCH POLYETHLENE FROM NISTMANAGER
nistManager->FindOrBuildMaterial("G4_POLYETHYLENE");
G4Material * matPE = G4Material::GetMaterial("G4_POLYETHYLENE");

// DEFINE DETECTOR
G4double detLx = worldSizeXY * 0.9;
G4double detLy = worldSizeXY * 0.9;
G4double detPolyLz = 10.*cm;

G4Box * detectorSolid = new G4Box("detector",detLx/2.,detLy/2.,detPolyLz/2.);
G4LogicalVolume * detectorLV = new G4LogicalVolume(detectorSolid,matPE,"detectorLV");
G4double detectorZpos = worldSizeZ/2.0*0.85;
new G4PVPlacement(0,G4ThreeVector(0,0,detectorZpos),detectorLV,"detectorPV",worldLV,false,0,fCheckOverlaps);
//////////////////////////////////////////////////////////////////////////////
```

Your code should look something like this. See codeblock #1 in the repository directory

When you've completed this you can return to your `/build/` directory and recompile by typing `make`



PE=Polyethylene

Geometry Setup - Let's place a radiator in front of the PE

- Fetch lead "G4_Pb" from the NIST manager and assign to a material.
- Solid(Box): Same XY dimensions as PE solid, let this solid be 10cm thick or how ever much you'd like.
- Same as before, declare logical volume, give it whatever name you'd like
- Place it at a Z position directly in front of the PE.

```
////////////////////////////////////////////////////////////////////////////////
// HANDS-ON #2: Radiator
// FETCH LEAD (PB) FROM NISTMANAGER
nistManager->FindOrBuildMaterial("G4_Pb");
G4Material * matPb = G4Material::GetMaterial("G4_Pb");
G4double detLeadLz = 10.*cm;

G4Box * leadPlateSolid = new G4Box("leadPlate",detLx/2.,detLy/2.,detLeadLz/2.);
G4LogicalVolume * leadPlateLV = new G4LogicalVolume(leadPlateSolid,matPb,"leadPlateLV");
G4double leadPlateZpos = worldSizeZ/2.0*0.85 - detPolyLz/2. - detLeadLz/2.;
new G4PVPlacement(0,G4ThreeVector(0,0,leadPlateZpos),leadPlateLV,"leadPlatePV",worldLV,false,0,fCheckOverlaps);
////////////////////////////////////////////////////////////////////////////////
```

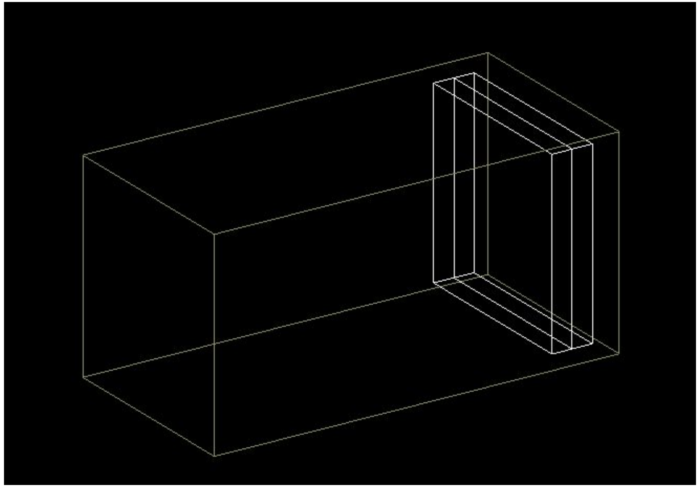
PE=Polyethylene

Geometry Setup - Let's place a radiator in front of the PE

- Fetch lead "G4_Pb" from the NIST manager and assign to a material.
- Solid(Box): Same XY dimensions as PE solid, let this solid be 10cm thick or how ever much you'd like.
- Same as before, declare logical volume, give it whatever name you'd like
- Place it at a Z position directly in front of the PE.

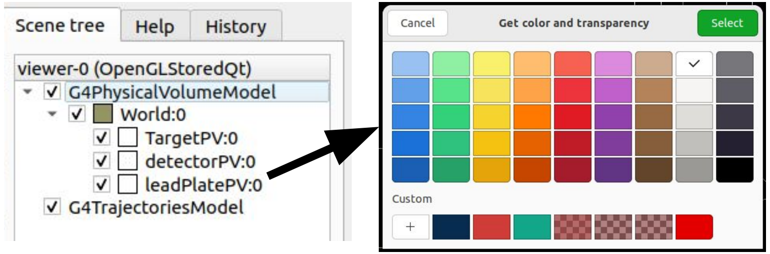
```
///////////////////////////////////////////////////////////////////////////////
// HANDS-ON #2: Radiator
// FETCH LEAD (PB) FROM NISTMANAGER
nistManager->FindOrBuildMaterial("G4_Pb");
G4Material * matPb = G4Material::GetMaterial("G4_Pb");
G4double detLeadLz = 10.*cm;

G4Box * leadPlateSolid = new G4Box("leadPlate",detLx/2.,detLy/2.,detLeadLz/2.);
G4LogicalVolume * leadPlateLV = new G4LogicalVolume(leadPlateSolid,matPb,"leadPlateLV");
G4double leadPlateZpos = worldSizeZ/2.0*0.85 - detPolyLz/2. - detLeadLz/2.;
new G4PVPlacement(0,G4ThreeVector(0,0,leadPlateZpos),leadPlateLV,"leadPlatePV",worldLV,false,0,fCheckOverlaps);
///////////////////////////////////////////////////////////////////////////////
```

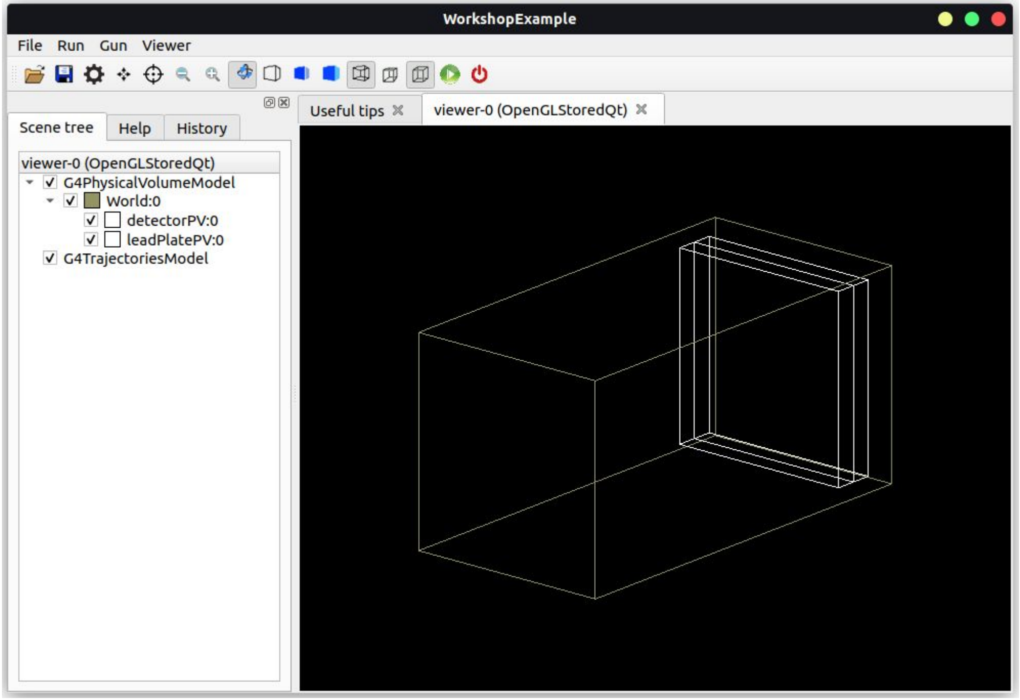
When you're done and you successfully rebuild the application it should look something like this →



Visualization - GUI

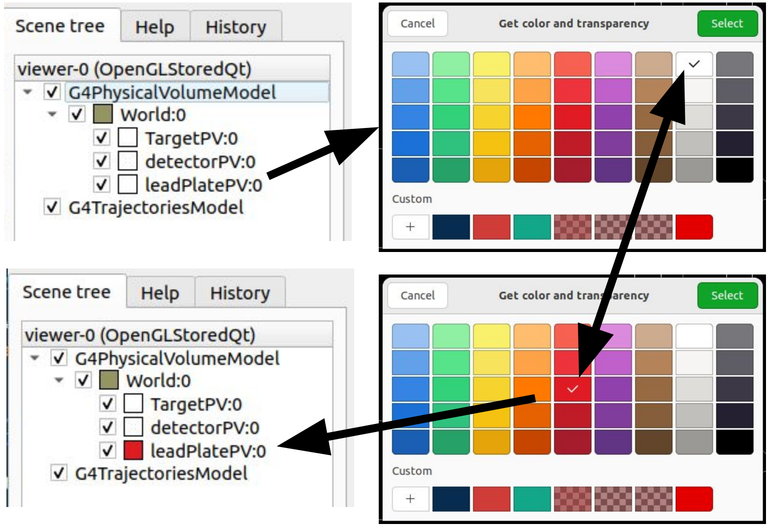
- After you successfully rebuild you have the
- You have two terrible white wireframe boxes together. Let’s do something about that in the GUI.

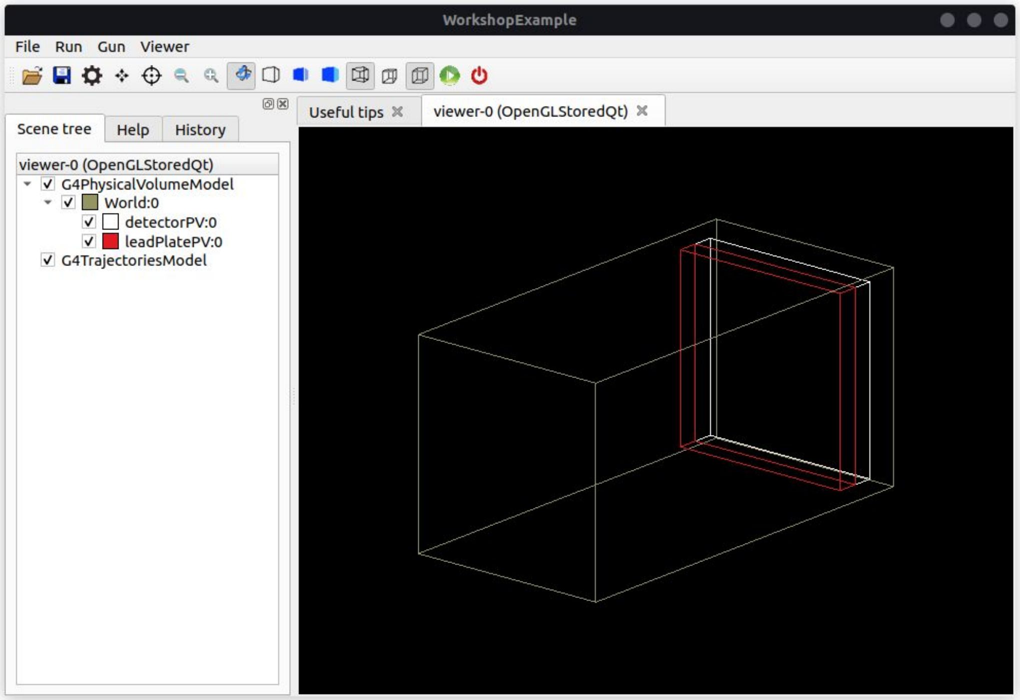




Visualization - GUI

- After you successfully rebuild you have the
- You have two terrible white wireframe boxes together. Let's do something about that in the GUI.





Visualization Attributes - Hard Coded Example

<u>G4VisAttributes Class</u>

- Constructor is (R,G,B,A)

- or a G4VisAttributeObject

To make the world a wireframe, we:

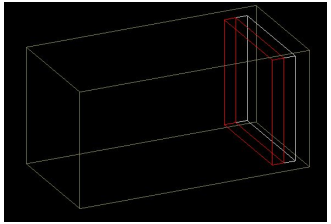
- Declare a new VisAttribute object and give it a color.

- Call the SetForceWireframe() method passing a value of 'true'

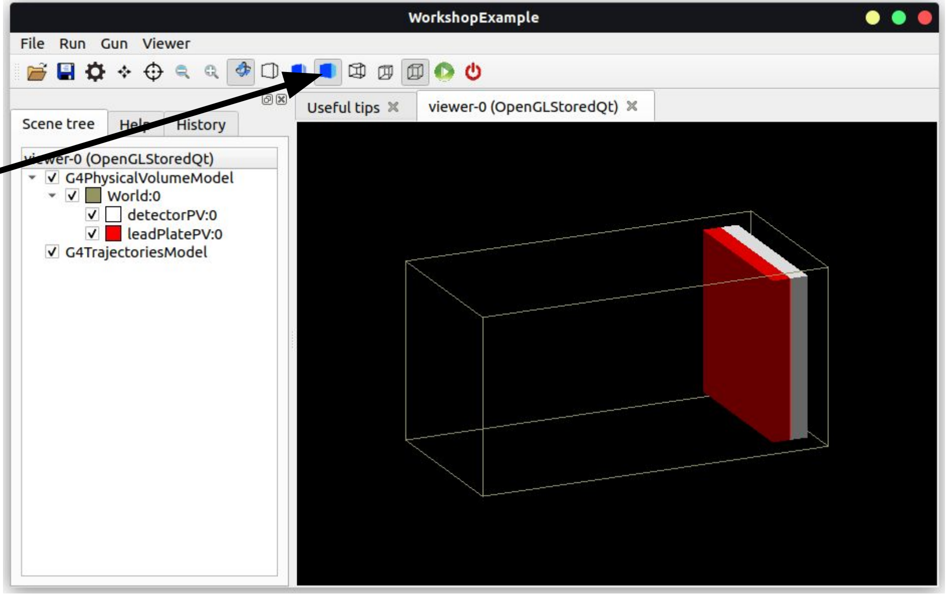
- Call the SetVisAttributes() method on the 'World' Logical volume

```
/////////////////////////////////////////////////////////////////////////////////
// Visualization attributes
//worldLV->SetVisAttributes(G4VisAttributes::GetInvisible());
G4VisAttributes * worldVisAtt   = new G4VisAttributes( G4Colour(149./255.,149./255.,100./255.,1.) );
worldVisAtt->SetForceWireframe(true);
worldLV->SetVisAttributes(worldVisAtt);
```

Visualization Attributes - Now you do it



- Set the color of the radiator to red, or any other color you would like.
- Force wireframe is optional.
  - Without that option you can view it as a solid.
- This isn't wholly necessary here but as you build complicated geometries setting colors is extremely helpful.



You're code should look something like this ⇒

```
///////////////////////////////////////////////////////////////////////////////////
// HANDS-ON #3: Visual Attributes
G4VisAttributes * pbVisAtt   = new G4VisAttributes( G4Colour(255./255.,0./255.,0./255.,1.) );
pbVisAtt->SetForceWireframe(true);
leadPlateLV->SetVisAttributes(pbVisAtt);
///////////////////////////////////////////////////////////////////////////////////
```

G4AnalysisManager ⇒ A Simple Analysis Option

- G4AnalysisManager a simpler option over ROOT Coding in Geant4.
- AnalysisManager ROOT output effectively equivalent of a .csv – can output in root, CSV, or XML
  - While uncomplicated this is useful for simple projects

➢ In this code, histograms and ntuples are created in the RunAction() constructor.
  - AnalysisManager's CreateH1()
  - AnalysisManager's CreateNtuple

➢ Note that you create an n-tuple, then add columns, and then let the AnalysisManager know you've finished.

src/RunAction.cc

```
RunAction::RunAction()
{
  // set printing event number per each event
  G4RunManager::GetRunManager()->SetPrintProgress(1);

  // Create analysis manager -- output type determined by file extension
  auto analysisManager = G4AnalysisManager::Instance();

  analysisManager->SetVerboseLevel(1);
  analysisManager->SetNtupleMerging(true);
  // Note: merging ntuples is available only with Root output

  // Creating histograms
  analysisManager->CreateH1("Edep","Edep in absorber", 110, 0., 330*MeV);
  analysisManager->CreateH1("trackLength","trackL in absorber", 100, 0., 50*cm);

  // Creating ntuple
  analysisManager->CreateNtuple("T", "Edep and TrackL");
  analysisManager->CreateNtupleDColumn("Edep");
  analysisManager->CreateNtupleDColumn("trackLength");
  analysisManager->FinishNtuple();
}
```

G4AnalysisManager ⇒ A Simple Analysis Option [Cont'd]

- At BeginOfRunAction() the application actually opens/creates the file.
- EndOfRunAction() the application closes out the file.
  - Always be sure to close out the file.

src/RunAction.cc

```
void RunAction::BeginOfRunAction(const G4Run* /*run*/)
{
  // Get analysis manager
  auto analysisManager = G4AnalysisManager::Instance();
  // create file at beginning of run
  G4String fileName = "B4.root";
  analysisManager->OpenFile(fileName);
  G4cout << "Using " << analysisManager->GetType() << G4endl;
}
```

```
void RunAction::EndOfRunAction(const G4Run* /*run*/)
{
  // call an instance of the G4AnalysisManager
  auto analysisManager = G4AnalysisManager::Instance();

  // save histograms & ntuple
  analysisManager->Write();
  analysisManager->CloseFile();
}
```

PrimaryActionGenerator – A Quick Review

Code from /src/PrimaryGeneratorAction.cc ⇒

- Default particle set in the constructor
  - If you want to change particles with the default G4 **`/gun/particle`** macro this is the easiest setup.
- GeneratePrimaries() runs at the beginning of each event.

➢ More complicated generators are common. You can:
  - Monte Carlo energy spectrums
  - Monte Carlo vertex positions
  - Generate multiple different particles based on hand-derived kinematics...
  - Etc.

➢ Here, we're just setting a point beam.

```
PrimaryGeneratorAction::PrimaryGeneratorAction()
{
  G4int nofParticles = 1;
  fParticleGun = new G4ParticleGun(nofParticles);

  // default particle kinematic
  auto particleDefinition = G4ParticleTable::GetParticleTable()->FindParticle("
  fParticleGun->SetParticleDefinition(particleDefinition);
  fParticleGun->SetParticleMomentumDirection(G4ThreeVector(0.,0.,1.));
  fParticleGun->SetParticleEnergy(300.*MeV);
}
```

```
void PrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
{
  G4double worldZHalfLength = 0.;
  auto worldLV = G4LogicalVolumeStore::GetInstance()->GetVolume("World");
  G4Box* worldBox = nullptr;

  if (  worldLV ) { worldBox = dynamic_cast<G4Box*>(worldLV->GetSolid()); }

  if ( worldBox ) {
    worldZHalfLength = worldBox->GetZHalfLength();
  } else {
    G4ExceptionDescription msg;
    msg << "World volume of box shape not found." << G4endl;
    G4Exception("PrimaryGeneratorAction::GeneratePrimaries()",
      "MyCode0002", JustWarning, msg);
  }

  // Set gun position
  fParticleGun->SetParticlePosition(G4ThreeVector(0., 0., -worldZHalfLength));
  fParticleGun->GeneratePrimaryVertex(anEvent);
}
```

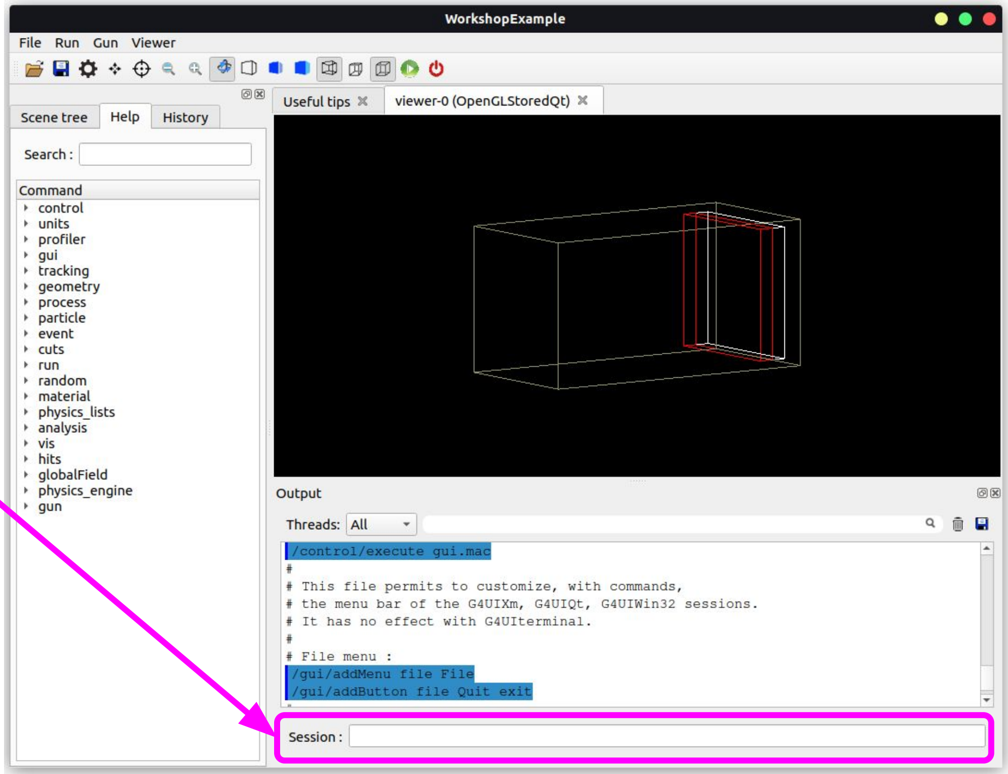
Executing Macro: Run in GUI

Let's run the application in GUI mode:

```
./WorkshopExample
```

In the Session prompt:

```
/run/beamOn 25
```



Executing Macro: Run in GUI

Let's run the application in GUI mode:

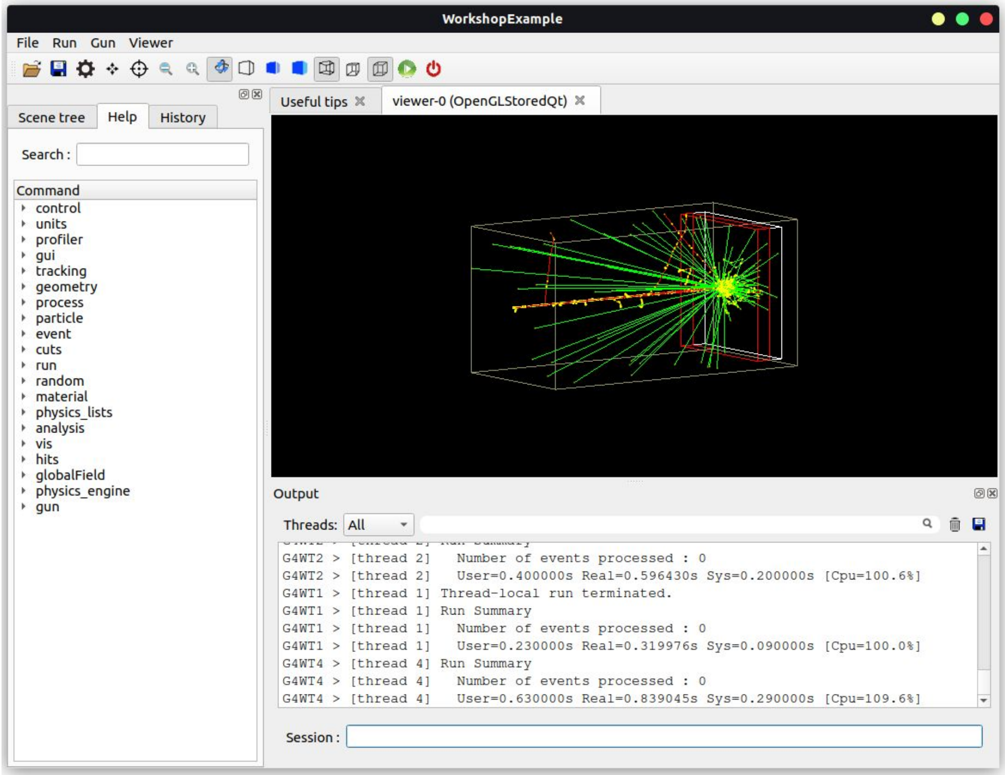
```
./WorkshopExample
```

In the Session prompt:

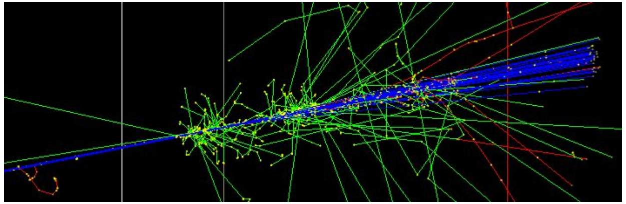
```
/run/beamOn 25
```

Default energy is 300*MeV, let's change that to 3*GeV.

```
/gun/energy 3.0 GeV
/run/beamOn 25
```



Executing Macro: Run in GUI



Let’s run the application in GUI mode:

```
./WorkshopExample
```

In the Session prompt:

```
/run/beamOn 25
```

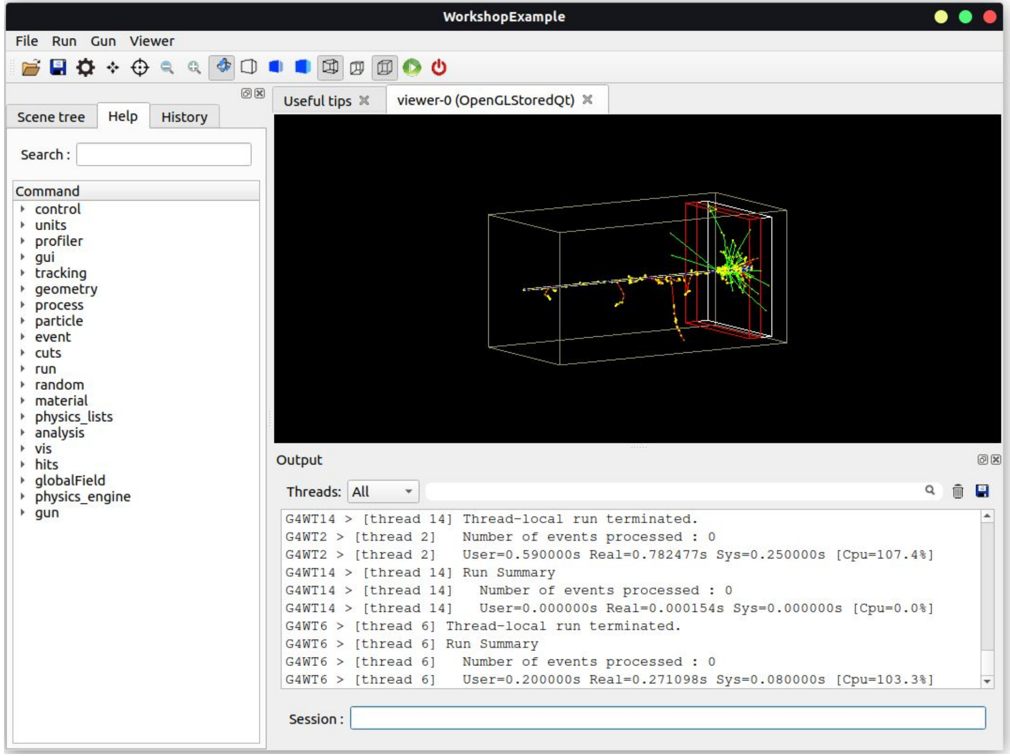
Default energy is 300*MeV, let’s change that to 3*GeV.

```
/gun/energy 3.0 GeV
/run/beamOn 25
```

Let’s change the particle of the gun to mu+

```
/gun/particle mu+
/run/beamOn 25
```

Note the change in color of the primaries as they’re positively charged (see top right img)



Executing Macro: Run beam in Batch Mode

We're going to run the application in 'batch mode' – no GUI.

We'll have a nice output ROOT file that we can quickly look at before proceeding into ROOT…

At your command line:

```
./WorkshopExample -m run2.mac
```

The -m here is specified in WorkshopExample.cc

```
# Macro file for WorkshopExample

/run/initialize

/gun/particle e-
/gun/energy 300. MeV

/run/printProgress 1000
/run/beamOn 100000

```

Any questions while the simulations quickly run?

ROOT file output

We can take a quick look at the root files:

From the command line type:

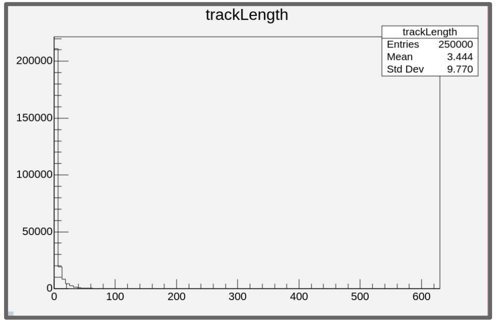
```
root -l B4.root
```

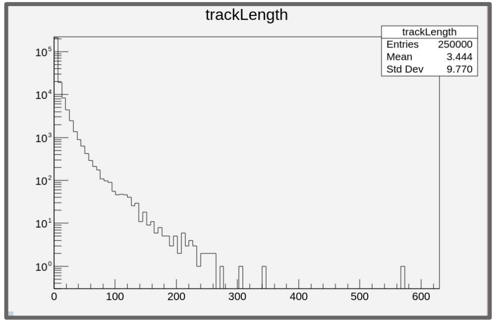
This will open up the ROOT command line

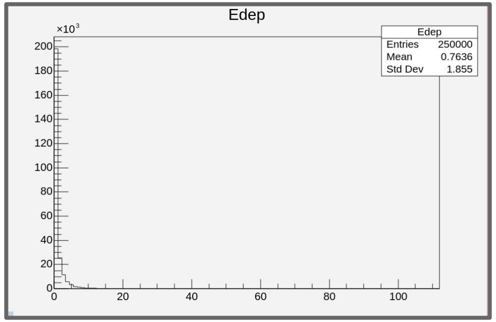
```
T->Draw(“trackLength”)
T->Draw(“Edep”)
T->Draw(“Edep:trackLength”)
```

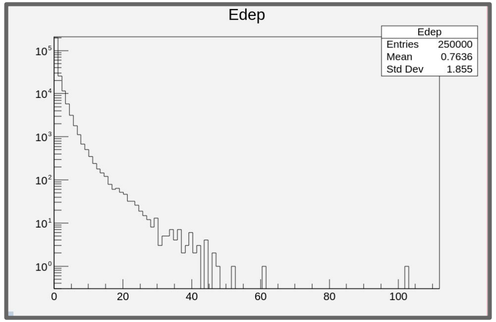
This will draw you histograms of the data collected by G4 into the root file

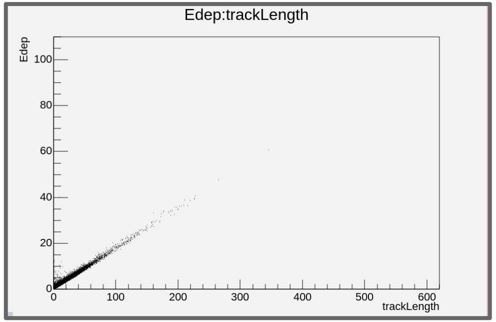
- 2 1D histograms, and 1 2D histogram











Transitioning into ROOT

[We'll start by looking at our G4 Workshop Example ROOT File]

Proper Setup Check

If the instructions on slide 2 were followed then you should be able to access the ROOT binaries

To check this, at the command line:

```
root --version
```

You should see something like the output below

```
[ericking@ifarm2401 workshopGeant4andROOT]$ root --version
ROOT Version: 6.30/04
Built for linuxx8664gcc on Apr 28 2024, 15:46:02
From heads/master@tags/v6-30-04
[ericking@ifarm2401 workshopGeant4andROOT]$
```

Getting Started – Accessing Objects in a ROOT file

- Let's move into the ROOT directory of the repository
- Let's copy our ROOT file here from the G4 simulation for east

```
cp ../build/B4.root ./
```

- Use ROOT to load the file

```
root -l B4.root
```

- This will bring you to the ROOT command prompt

```
[ericking@ifarm2401 workshopGeant4andROOT]$ cd ROOT
[ericking@ifarm2401 ROOT]$ cp ../build/B4.root ./
[ericking@ifarm2401 ROOT]$ ls
B4.root  sampleRootScript
[ericking@ifarm2401 ROOT]$ root -l B4.root
root [0]
Attaching file B4.root as _file0...
(TFile *) 0x3b079a0
root [1]
```

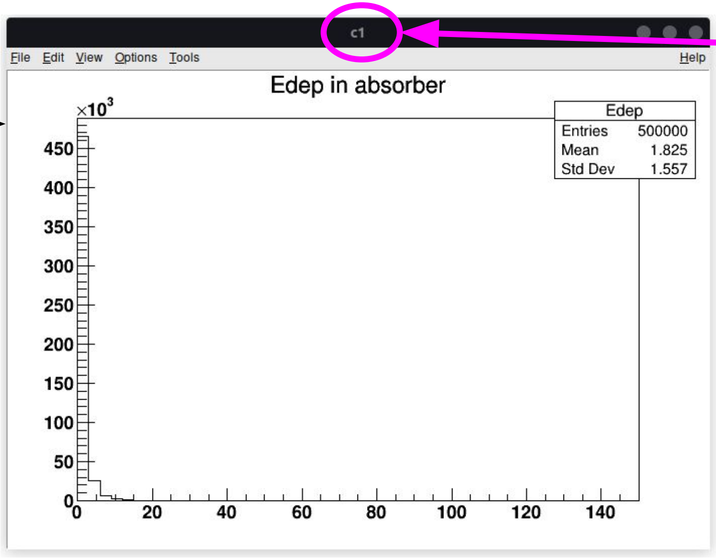
Getting Started – Accessing Objects in a ROOT file

- The file loads with object name `_file0`
- Now we can see the contents of the file.

```
root [1] _file0->ls()
TFile**         B4.root
 TFile*         B4.root
  KEY: TTree    B4;1     Edep and TrackL
  KEY: TH1D     Edep;1   Edep in absorber
  KEY: TH1D     trackLength;1    trackL in absorber
root [2]
```

- Let's draw one of the histograms:

```
Edep->Draw("HIST")
```



Canvas object name is same as title

- This isn't that easy to see. We can modify this plot

```
Edep->SetLineColor(kBlue)
Edep->SetLineWidth(2)
c1->SetLogy()
```

Getting Started – Accessing Objects in a ROOT file

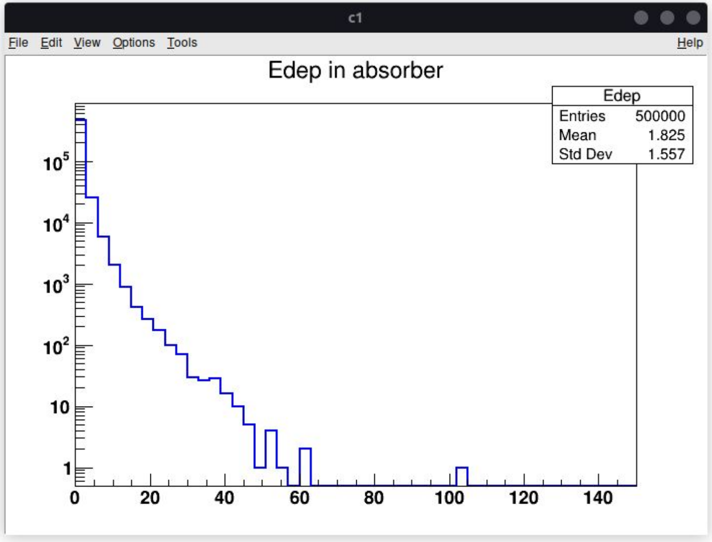
```
root [1] _file0->ls()
TFile**         B4.root
 TFile*         B4.root
  KEY: TTree    B4;1    Edep and TrackL
  KEY: TH1D     Edep;1  Edep in absorber
  KEY: TH1D     trackLength;1   trackL in absorber
root [2]
```

- ➢ The file loads with object name `_file0`
- ➢ Now we can see the contents of the file.
- ➢ Let's draw one of the histograms:

```
Edep->Draw("HIST")
```

- ➢ This isn't that easy to see. We can modify this plot

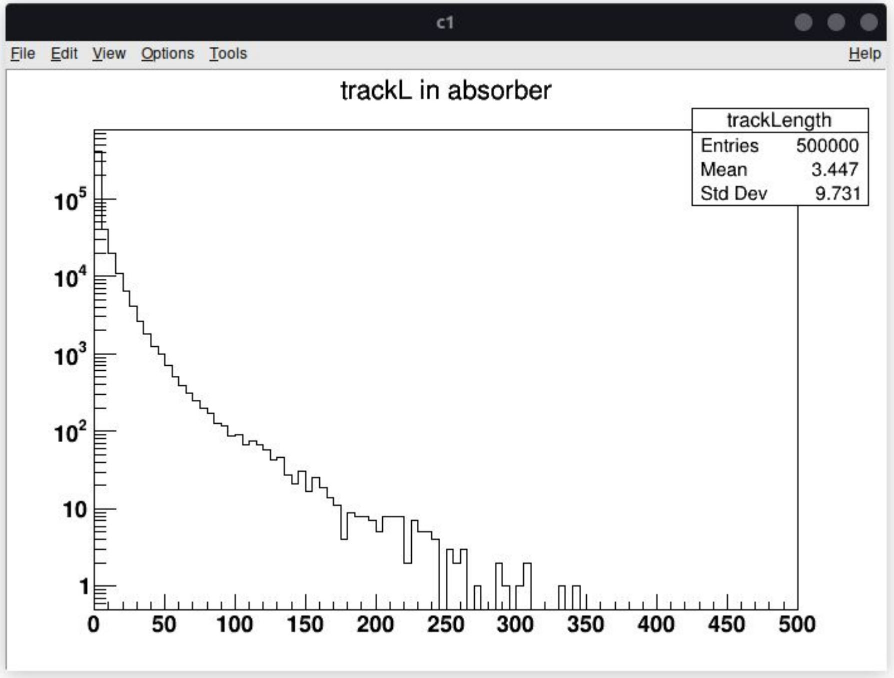
```
Edep->SetLineColor(kBlue)
Edep->SetLineWidth(2)
c1->SetLogy()
```



Getting Started – Accessing Objects in a ROOT file

➢ We will also draw the trackLength histogram:

```
trackLength->Draw("HIST");
```



➢ Note that the canvas retains its property of having a log-y axis but the histograms properties are default.

```
trackLength->SetLineColor(kRed)
trackLength->SetLineWidth(2)
trackLength->Draw("HIST")
```

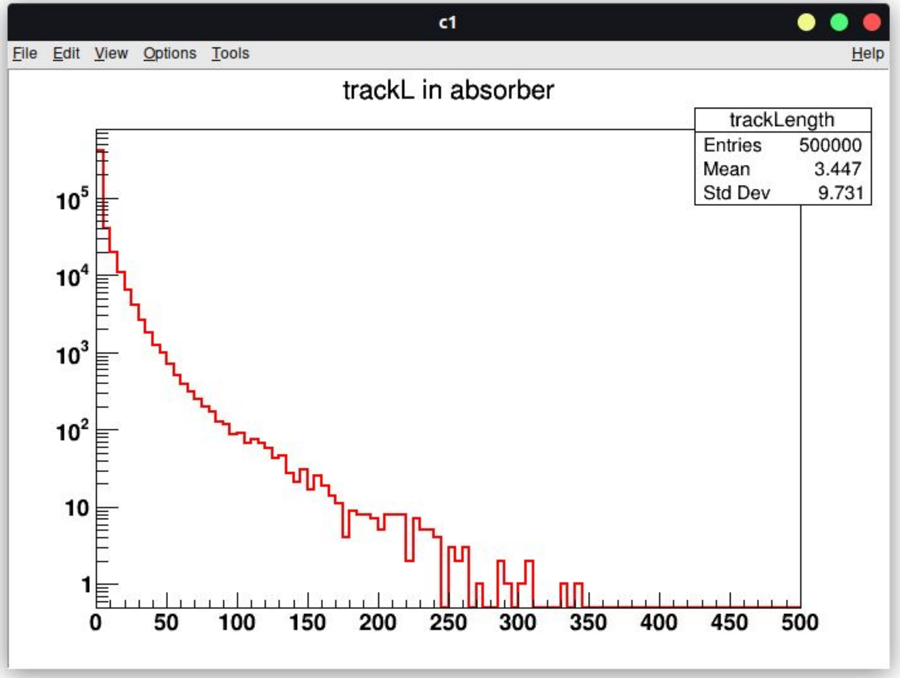
Getting Started – Accessing Objects in a ROOT file

➢ We will also draw the trackLength histogram:

```
trackLength->Draw("HIST");
```

➢ Note that the canvas retains its property of having a log-y axis but the histograms properties are default.

```
trackLength->SetLineColor(kRed)
trackLength->SetLineWidth(2)
trackLength->Draw("HIST")
```



Note: If you closed your canvas then ROOT will create a new one and canvas properties will not be retained.

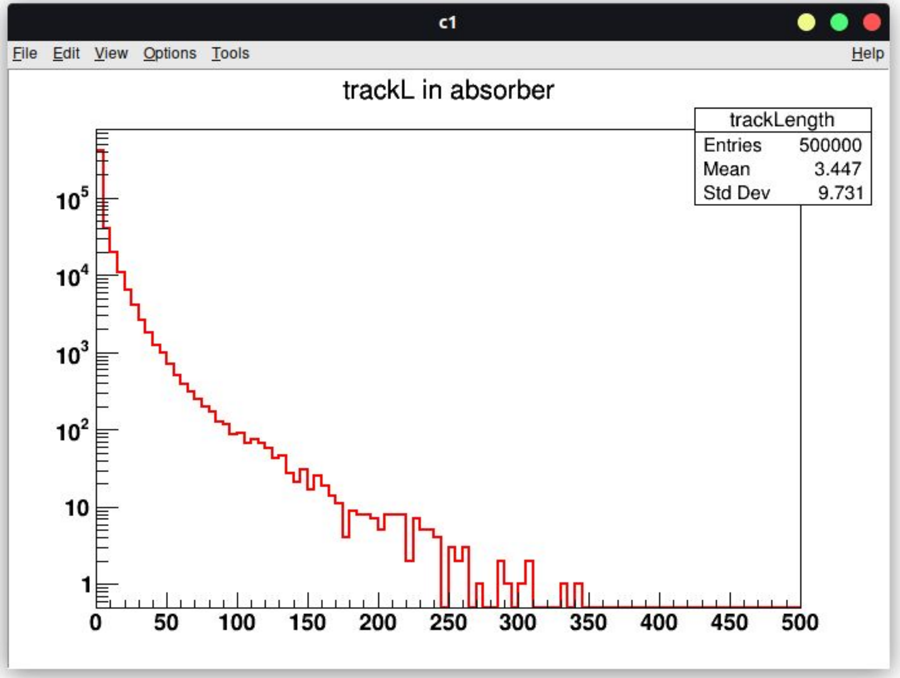
Getting Started – Accessing Objects in a ROOT file

➢ We will also draw the trackLength histogram:

```
trackLength->Draw("HIST");
```

➢ Note that the canvas retains its property of having a log-y axis but the histograms properties are default.

```
trackLength->SetLineColor(kRed)
trackLength->SetLineWidth(2)
trackLength->Draw("HIST")
```



Note: If you closed your canvas then ROOT will create a new one and canvas properties will not be retained.

Getting Started – Plotting from the Data Tree

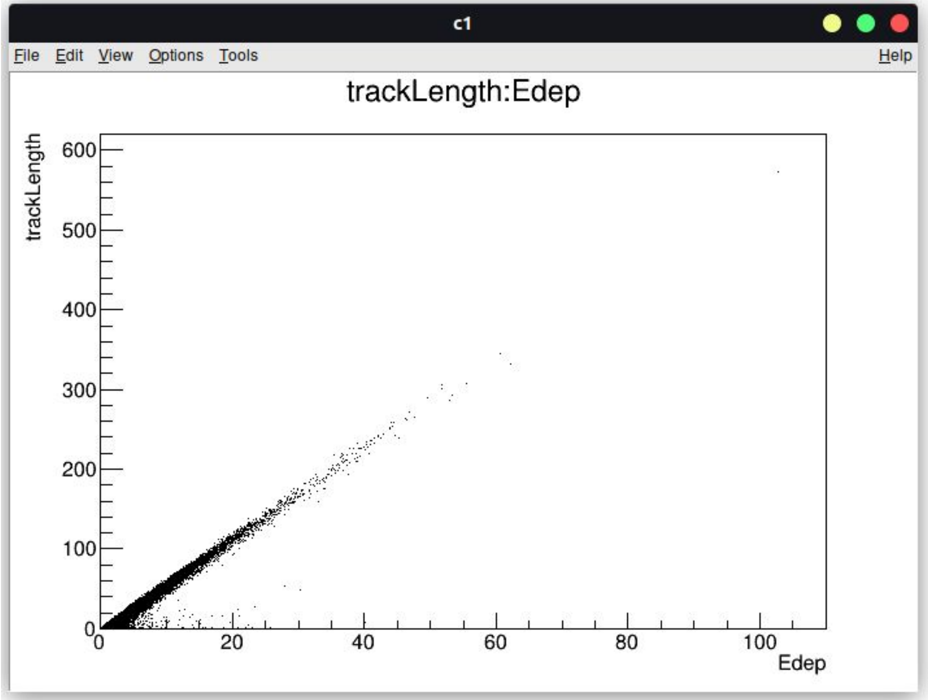
- We can also plot data from the ROOT Tree object:

```
B4->Draw(“trackLength:Edep”,“ ”, “ ”);
```

- Technically what it draws for you the first time is a TGraph–the points on this are fairly accurate.

- We can push the draw into a histogram object H:

```
B4->Draw(“trackLength:Edep>>H”,“ ”, “ ”);
```



Getting Started – Plotting from the Data Tree

- We can also plot data from the ROOT Tree object:

```
B4->Draw("trackLength:Edep"," ", " ");
```

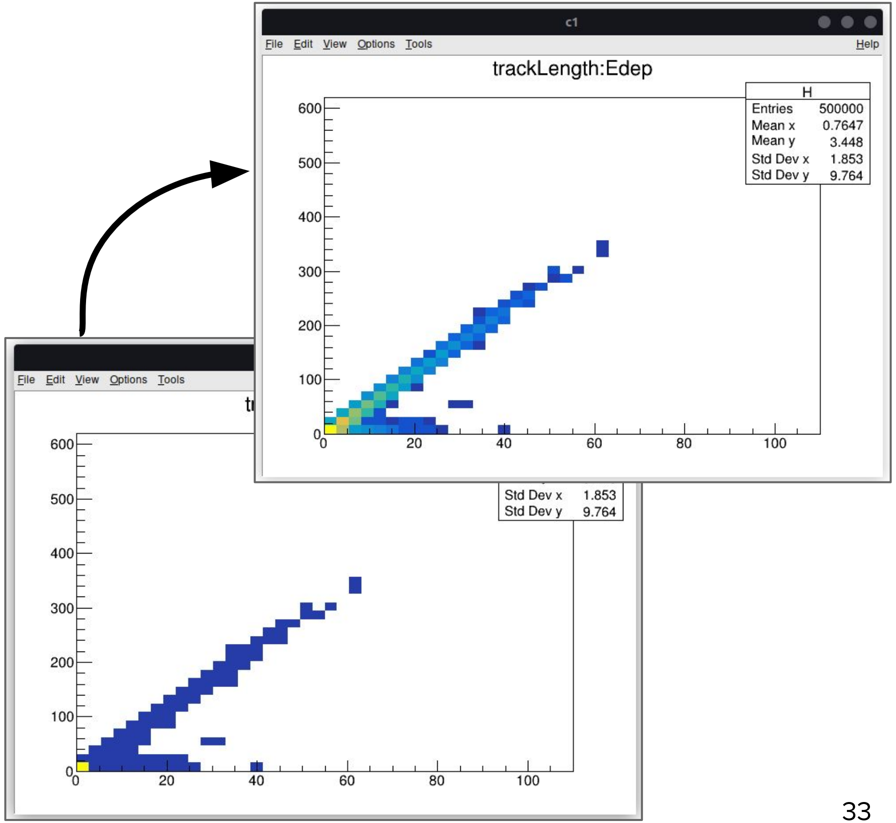
- Technically what it draws for you the first time is a TGraph–the points on this are fairly accurate.

- We can push the draw into a histogram object H:

```
B4->Draw("trackLength:Edep>>H"," ", " ");
```

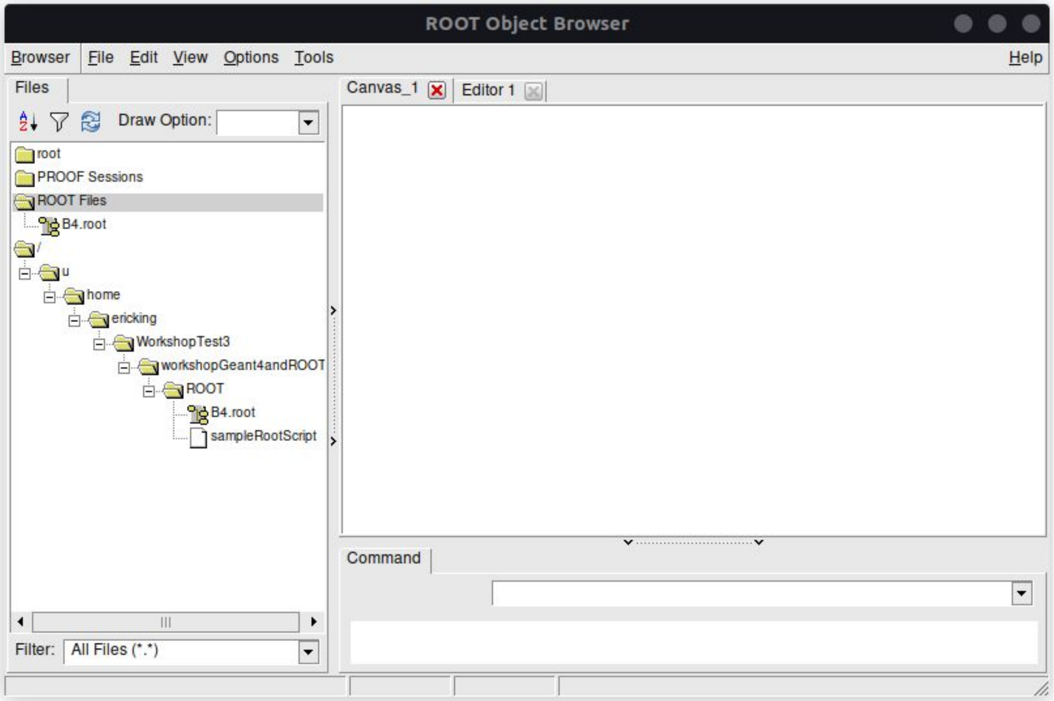
- ROOT seems to want to draw this as a heatmap. Let's at least turn the color map into a log-scale

```
c1->SetLogz()
```



Getting Started – Using the TBrowser

- You can also access the file from a TBrowser using the command line
  - *NOTE: This is cumbersome to use over an X11 connection. On your own machine, it's fine.*

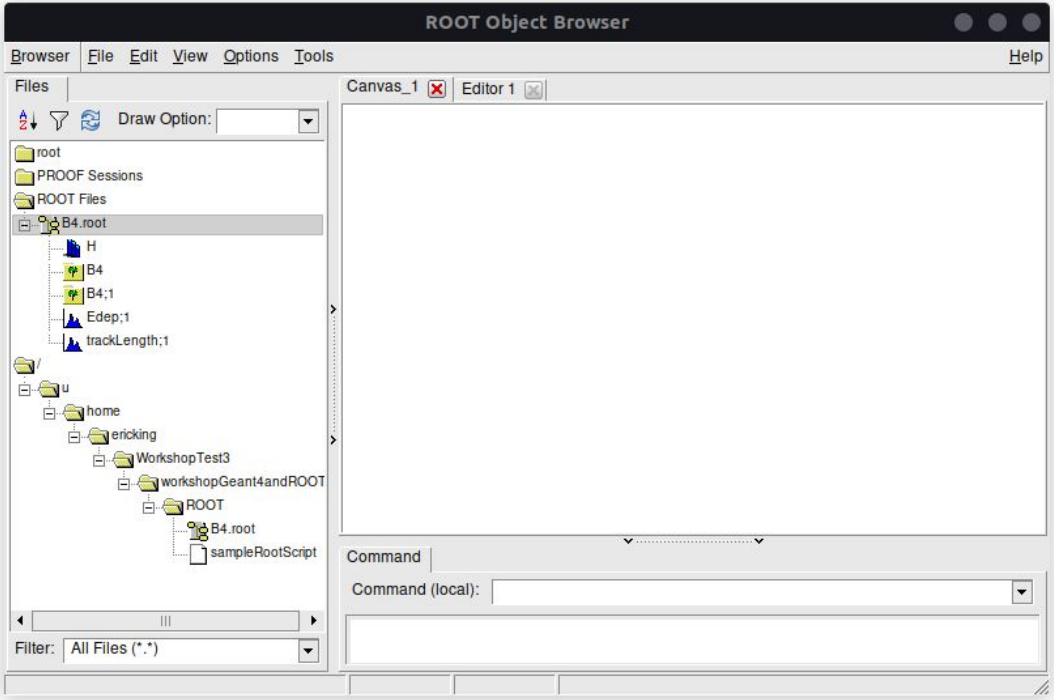


Getting Started – Using the TBrowser

- You can also access the file from a TBrowser using the command line
  - *NOTE: This is cumbersome to use over an X11 connection. On your own machine, it's fine.*

```
TBrowser x;
```

- Click on the ROOT file name, this will expand the file like a directory

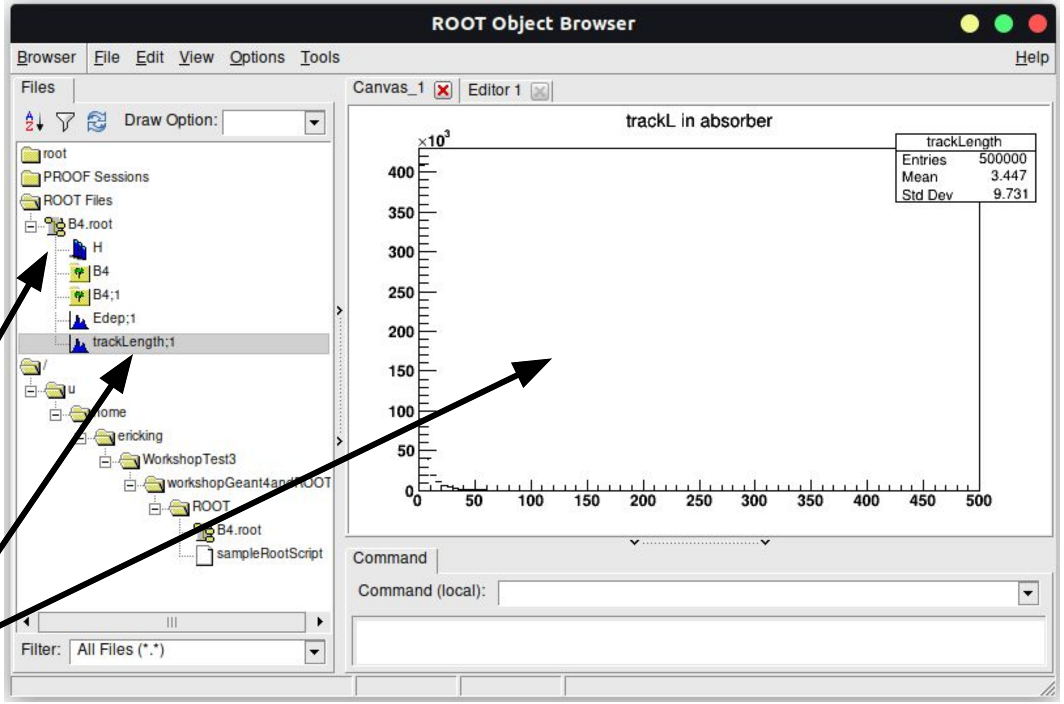


Getting Started – Using the TBrowser

- You can also access the file from a TBrowser using the command line
  - *NOTE: This is cumbersome to use over an X11 connection. On your own machine, it's fine.*

```
TBrowser x;
```

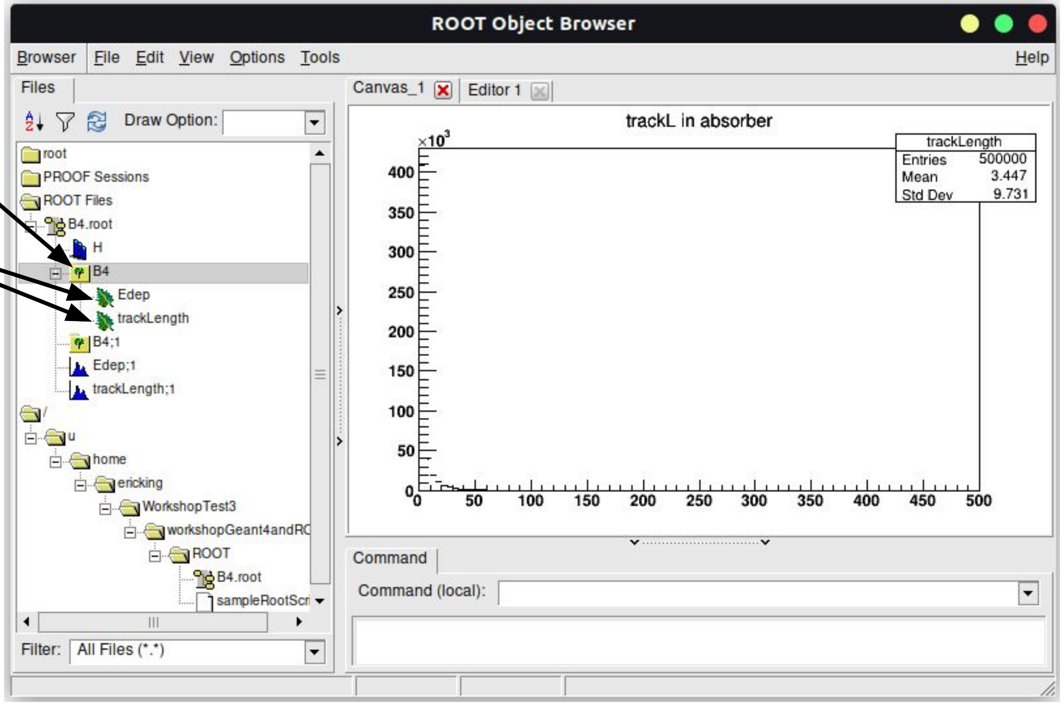
- Click on the ROOT file name, this will expand the file like a directory
- Click on an object and it will display in the window.



Getting Started – Using the TBrowser

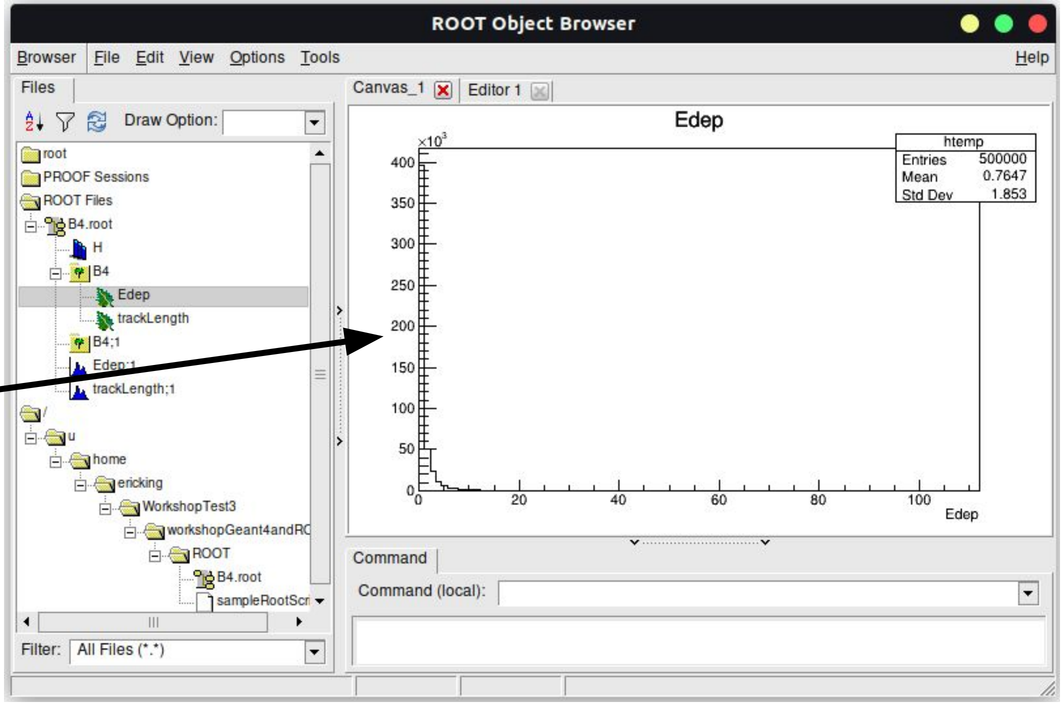
- You can click the TREE object (B4) to expand it

- You'll note that there are two associated Branches



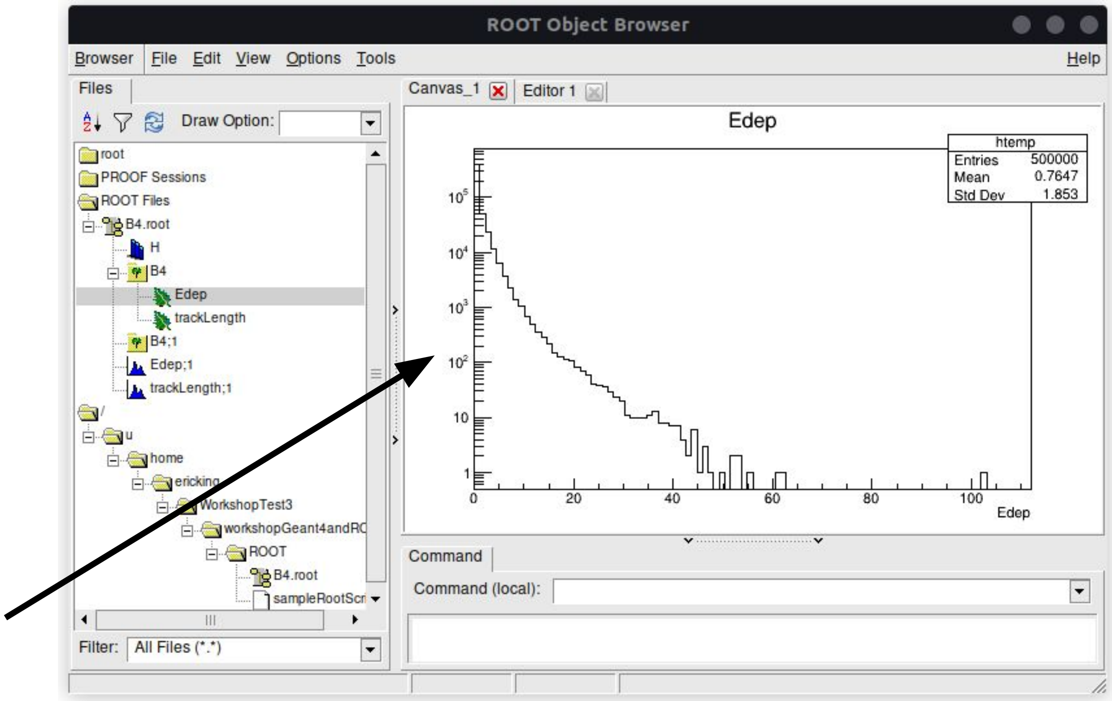
Getting Started – Using the TBrowser

- You can click the TREE object (B4) to expand it
- You'll note that there are two associated TBranches
- Double-clicking on a branch will show you the contained data in the browser window.



Getting Started – Using the TBrowser

- You can click the TREE object (B4) to expand it
- You'll note that there are two associated TBranches
- Double-clicking on a branch will show you the contained data in the browser window.
- Right-click in the margins of the Canvas and a menu will drop down.
  - Select SetLogy (about ⅔ the way down)



Creating a ROOT Macro

You can easily execute a series of commands in ROOT using a curly-bracketed macro of commands.

The macro of commands can then be executed at the terminal command line:

```
root -l  yourMacro
```

Note: I typically save my macros as a .C file since it's generally an easy conversion adding libraries and a few tweaks.

```
{

  // All of your ROOT stuff here.
  // Lines must end in semicolons;

}
```

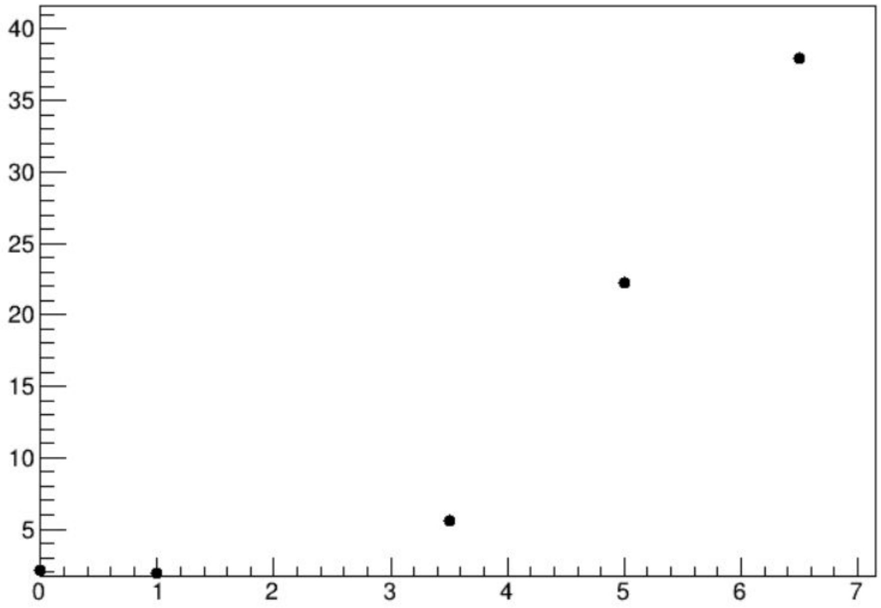
Creating a ROOT Macro: TGraph – Make the Graph

- Create a TGraph object
- Add points to the TGraph
    - SetPoint(n,x,y)
    - Add the points:
        - (0 , 2.1)
        - (1 , 1.9)
        - (3.5 , 5.6)
        - (5 , 22,2)
        - (6.5 , 38.0)
- Set a Marker Style (use 20), and set marker color to blue.
- Draw with option “AP”

```
{

  ///////////////////////////////////////////////////////////////////////
  // Create TGraph
  TGraph * gr = new TGraph();

  ///////////////////////////////////////////////////////////////////////
  // Fill TGraph -- SetPoint(n,x,y)
  gr->SetPoint(0,0.0,2.1);
  gr->SetPoint(1,1.0,1.9);
  gr->SetPoint(2,3.5,5.6);
  gr->SetPoint(3,5.0,22.2);
  gr->SetPoint(4,6.5,38.0);

  ///////////////////////////////////////////////////////////////////////
  // Set a marker style unless you're going to get tiny dots
  gr->SetMarkerStyle(20);
  gr->SetMarkerColor(kBlue);

  ///////////////////////////////////////////////////////////////////////
  // Draw the points -- Options A: Axis, P: Points
  gr->Draw("AP");

}
```

Creating a ROOT Script: TGraph – Execute the macro

Your output should look like the following:



Creating a ROOT Script: TGraph Data Fit

- Let's fit the graph to a second-order polynomial, "pol2" is predefined in ROOT.

```
gr->Fit("pol2")
```

- And lets set an option so our fit parameters show up on our plot.

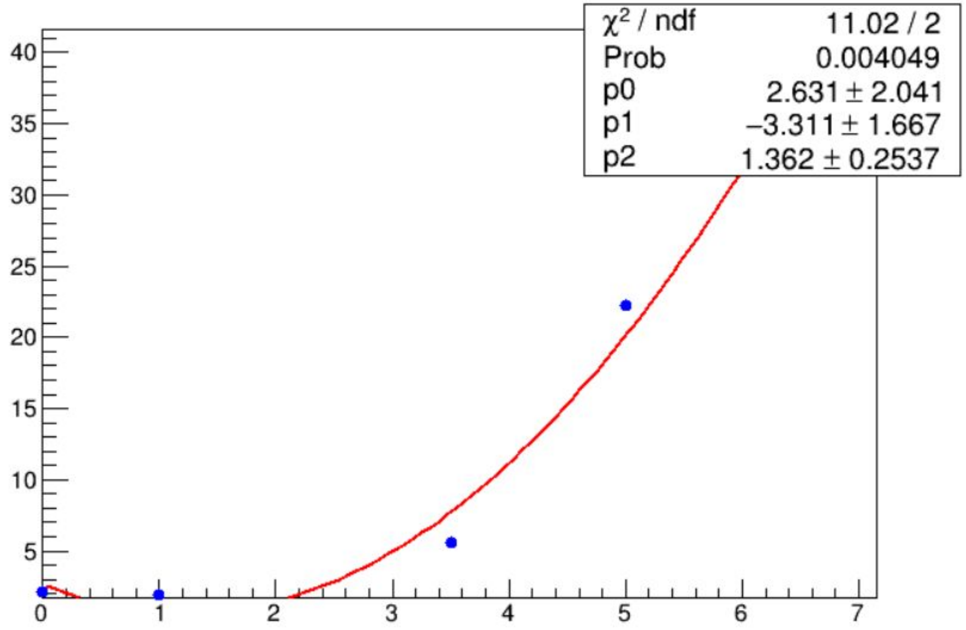
```
gStyle->SetOptFit(1111);
```

```
{

  ///////////////////////////////////////////////////////////////////////
  // Create TGraph
  TGraph * gr = new TGraph();

  ///////////////////////////////////////////////////////////////////////
  // Fill TGraph -- SetPoint(n,x,y)
  gr->SetPoint(0,0.0,2.1);
  gr->SetPoint(1,1.0,1.9);
  gr->SetPoint(2,3.5,5.6);
  gr->SetPoint(3,5.0,22.2);
  gr->SetPoint(4,6.5,38.0);

  ///////////////////////////////////////////////////////////////////////
  // Set a marker style unless you're going to get tiny dots
  gr->SetMarkerStyle(20);
  gr->SetMarkerColor(kBlue);

  ///////////////////////////////////////////////////////////////////////
  // Fit the TGraph--Fit to predefined pol2 function, show fit in box
  gr->Fit("pol2");
  gStyle->SetOptFit(1111);

  ///////////////////////////////////////////////////////////////////////
  // Draw the points -- Options A: Axis, P: Points
  gr->Draw("AP");

}
```

Creating a ROOT Script: TGraph Data Fit

Your output should look like the following:

This isn't a great plot, and we can also fix that…



Creating a ROOT Script: TGraph Data Fit

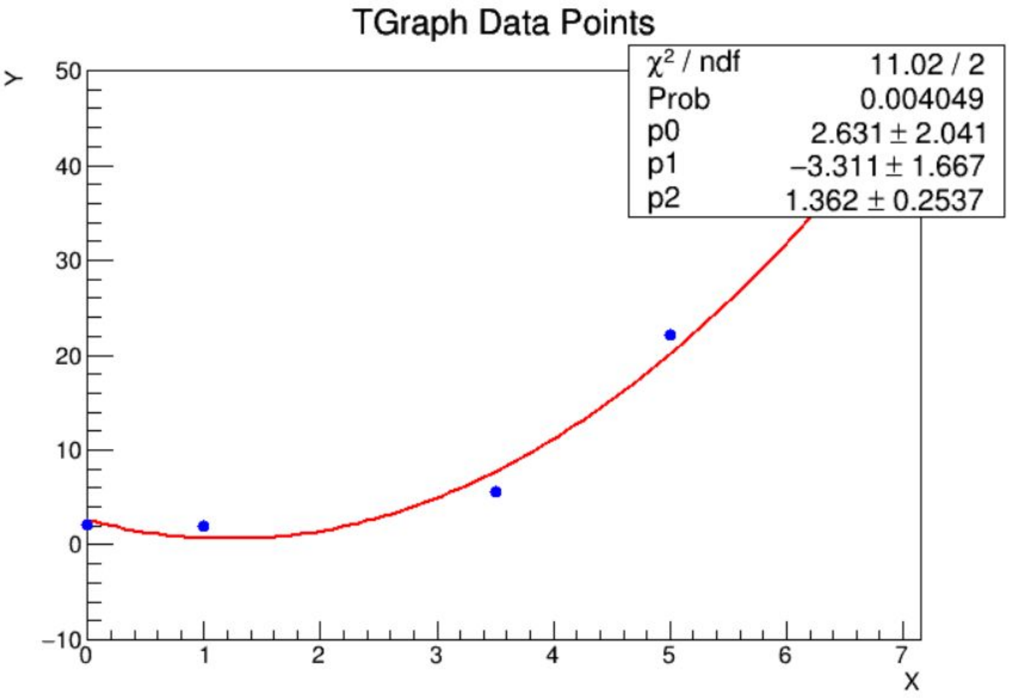
This isn't a great plot, and we can also fix that...

- Set the Y-axis range of the plot
- Add a title to the plot using SetTitle( )
    - Takes a string in the form "Main,X-title,Y-title"

```
///////////////////////////////////////////////////////////////////////
// Fit the TGraph--Fit to predefined pol2 function, show fit in box
gr->Fit("pol2");
gStyle->SetOptFit(1111);

///////////////////////////////////////////////////////////////////////
// Fix the y-axis to look nice
gr->GetYaxis()->SetRangeUser(-10,50);
gr->SetTitle("TGraph Data Points;X;Y");

///////////////////////////////////////////////////////////////////////
// Draw the points -- Options A: Axis, P: Points
gr->Draw("AP");
```

Creating a ROOT Script: TGraph Data Fit

Now, you should have something that looks like the following:



Fitting a Gaussian to Emulated Data

Time permitting, let's load into ROOT

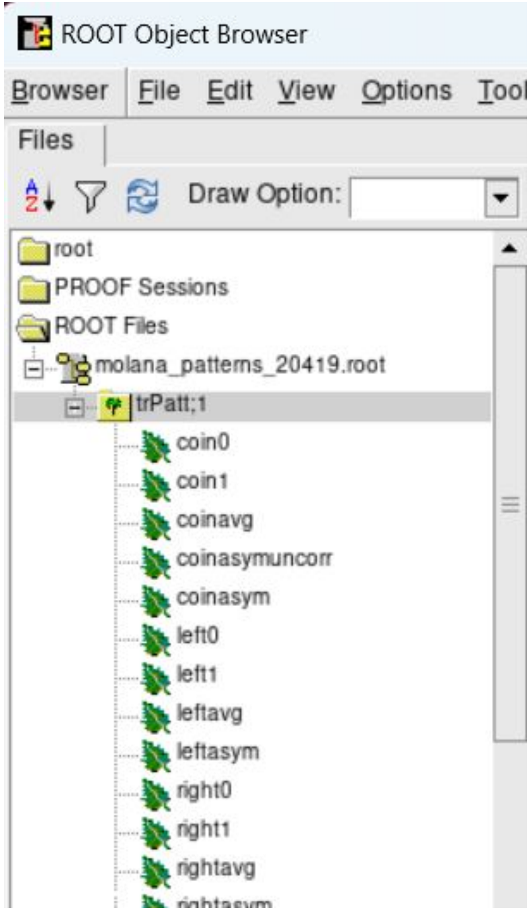
```
root -l molana_patterns_20419.root -e “TBrowser x”
```

This will automatically open up a TBrowser

- Double-click on the ROOT file

- Double-click on the TTree ‘trPatt’

You should see the following:



Fitting a Gaussian to Emulated Data

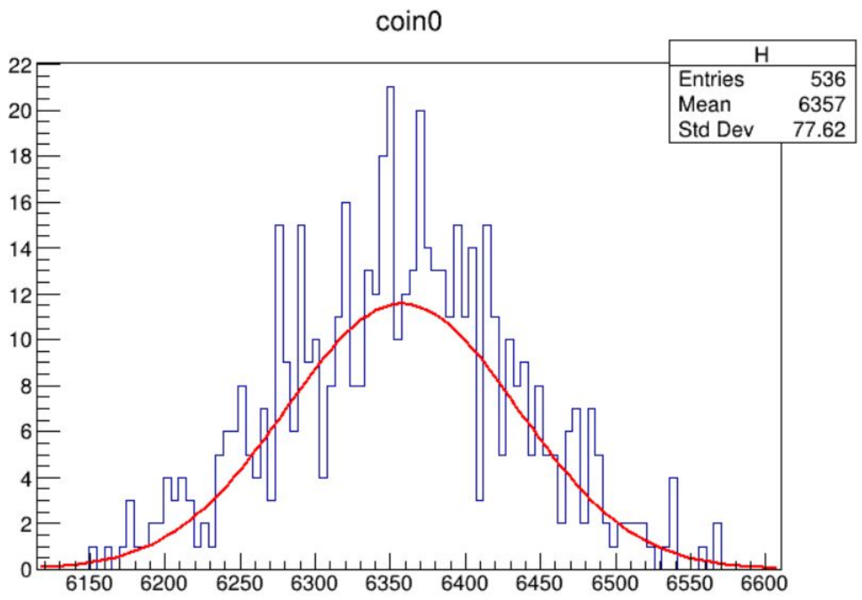
- In the Local Command line, draw the branch named "coin0" and funnel it into a histogram called "H"

```
trPatt->Draw("coin0>>H")
```

- Let's fit a gaussian curve to the data

```
H->Fit("gaus")
```

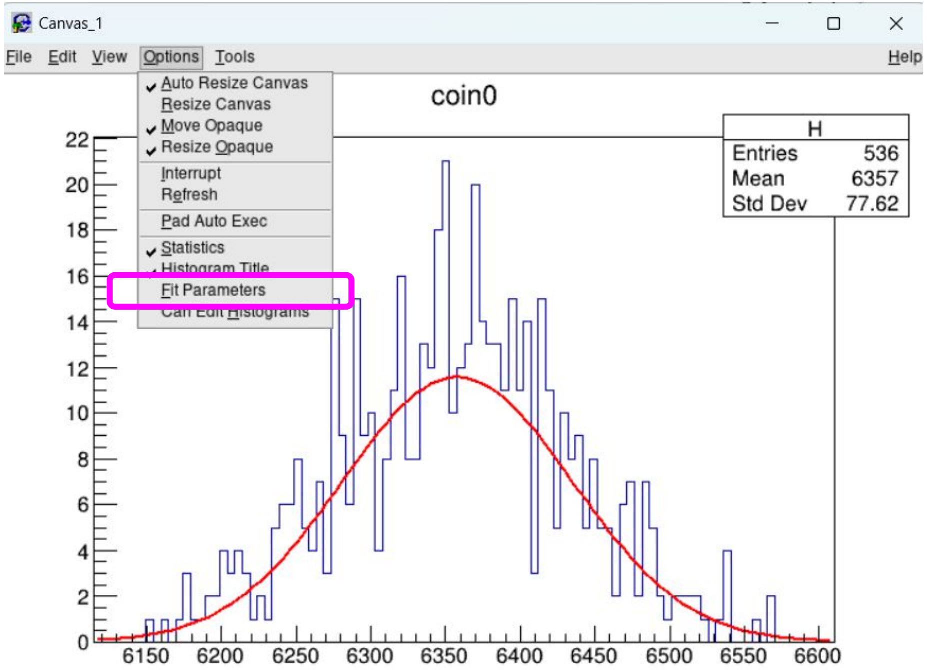
You should get something that looks like the image to the right →



Fitting a Gaussian to Emulated Data

We can add the fit parameters to the plot from the menu:

Options >> Fit Parameters

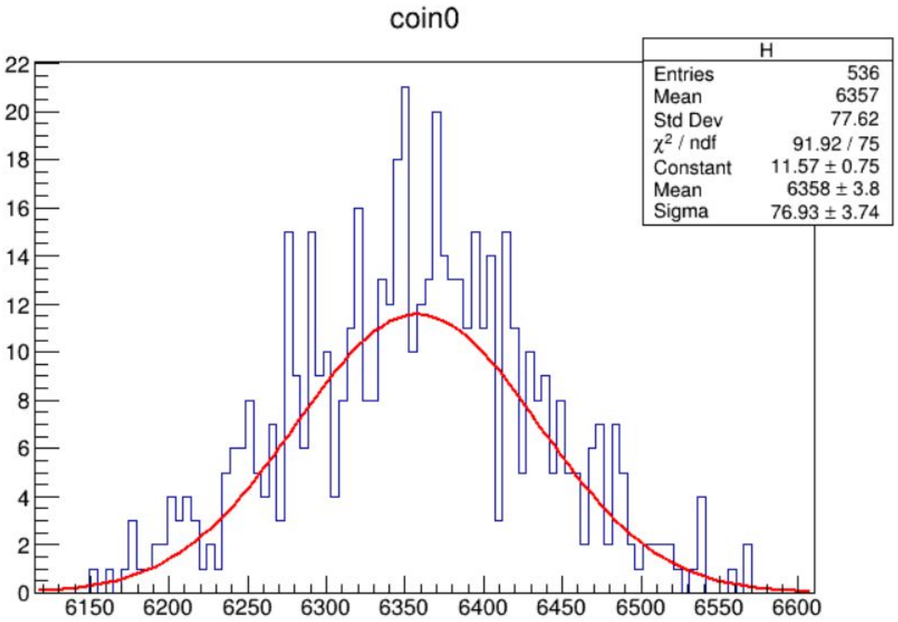


Fitting a Gaussian to Emulated Data

We can add the fit parameters to the plot from the menu:

Options >> Fit Parameters

When your screen updates you should have the Gaussian fit drawn and the parameters for the fit listed.



Note: This ROOT file data comes from the Moller Polarimeter DAQ using a 3-channel emulator to stress-test the dead time and accidentals reporting. There is nothing physical here if anyone was wondering.