GEANT4 Simulation Tutorial

Ryan Bayes

Queen's University @ SNOLAB

January 12, 2026
SNOLAB Orientation Session

What is a Simulation?

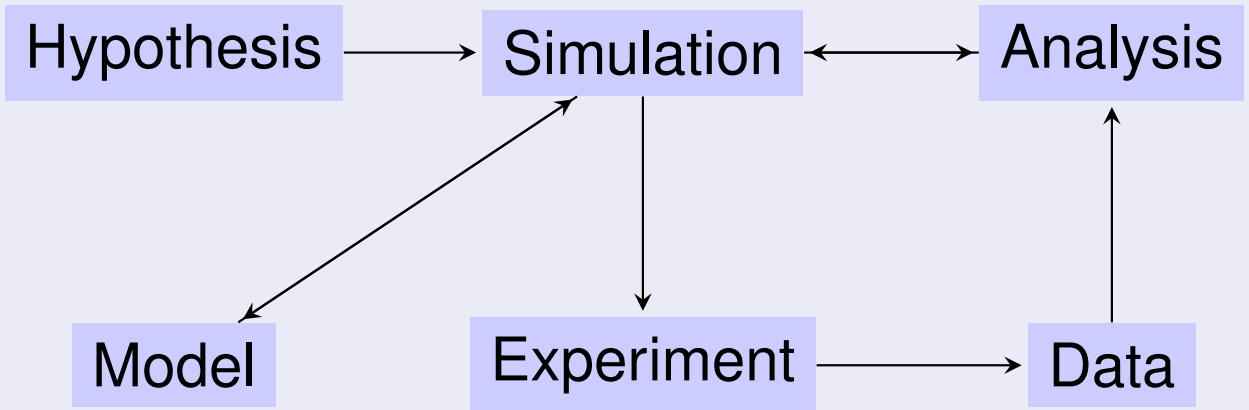
Definition (from Oxford English Dictionary)

1. imitation of a situation or process.
2. the action of pretending; deception.
3. the production of a computer model of something, especially for the purpose of study.

Why Use Simulations?

Simulations inform decisions about complicated systems

- Experimental design.
- Conduct statistical analysis and fitting.
- Validate analysis.
- Inform and test physical models

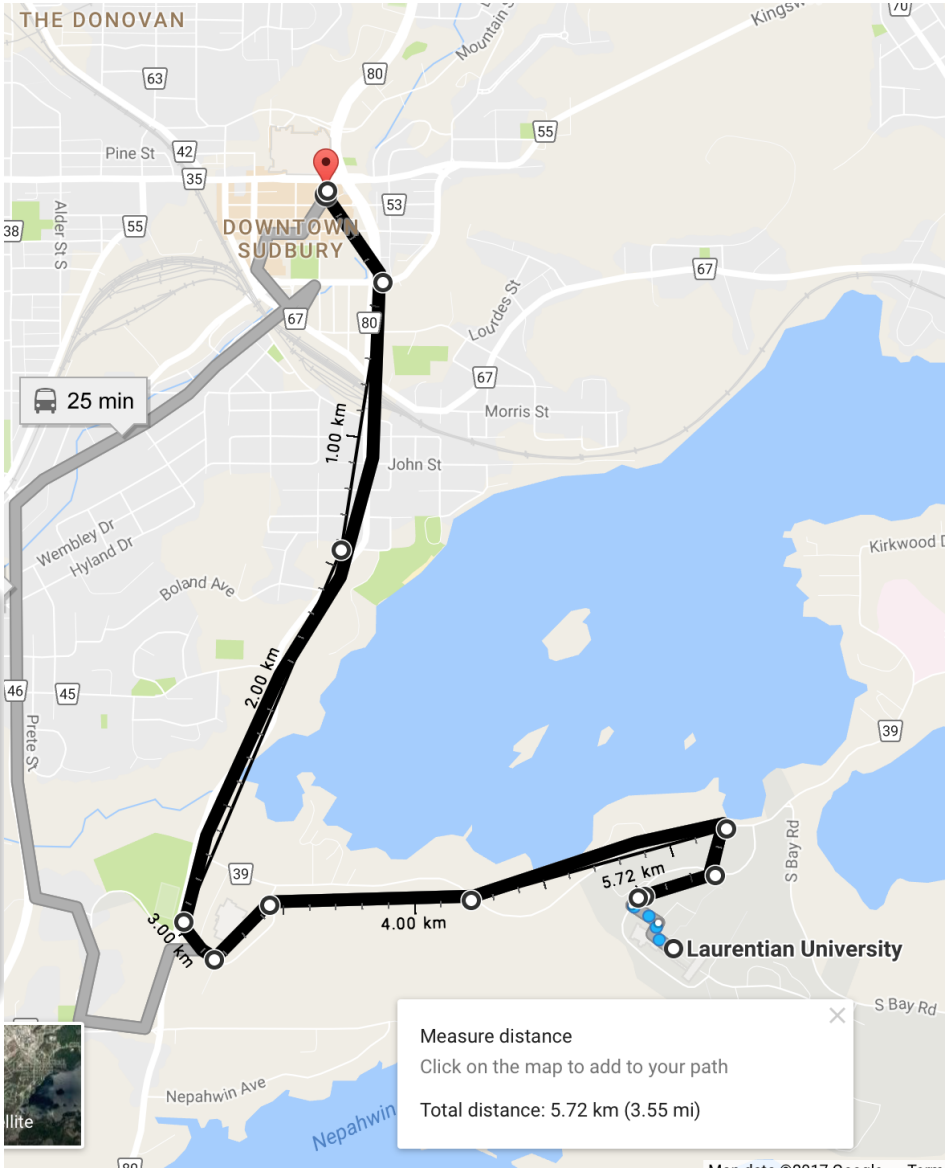


An Example: Bus Scheduling

- Would like to know when bus arrives.
- There are a number of random elements.

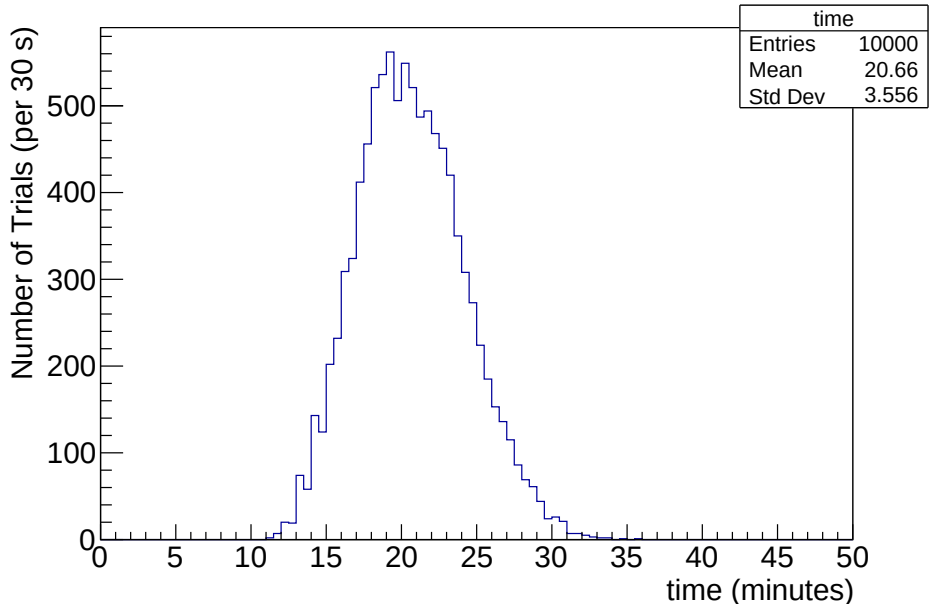
Considerations

- Number of stops with pickups.
- Number of stops at traffic lights.
- Mean speed between stops.



Bus Arrival Distribution

- Minor assumptions made
  - 60 s pickup
  - 35% chance of the bus stopping for pickup
  - Stoplight period is 4 minutes
  - Mean speed is 25 km/h
- Produce a distribution of travel times for 10 000 trials.



Exercise

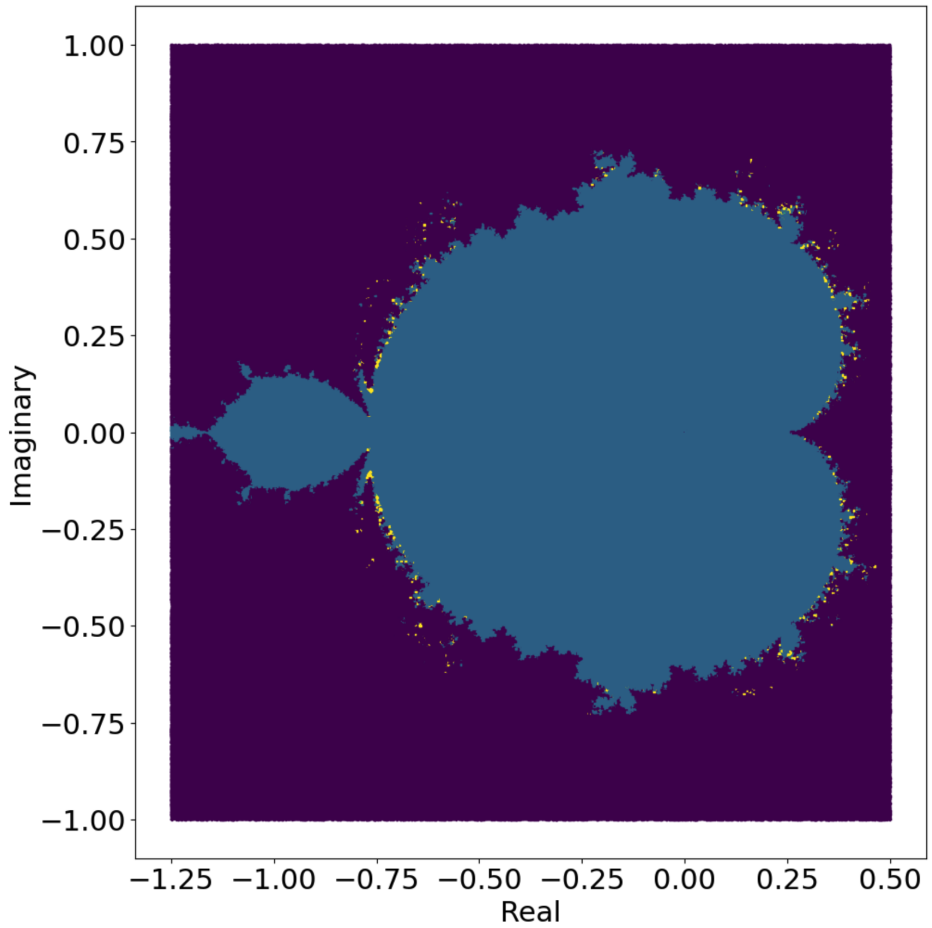
- Copy `/data/ComputerDay2017/Toys/Bus_Schedule.cpp` to your work area.
- Alter speed, stop, and light duration and see how travel time changes.

From Mathematics: The Mandelbrot Set

- Will the iterative expression

$$z_{n+1} = z_n^2 + C$$

where $z \in \mathbb{C}$ be bounded for a given $C \in \mathbb{C}$



Procedure

- Select a point C
- Iterate the expression 50 times
- Calculate the average difference between steps
- Colour code result based on escape "speed"

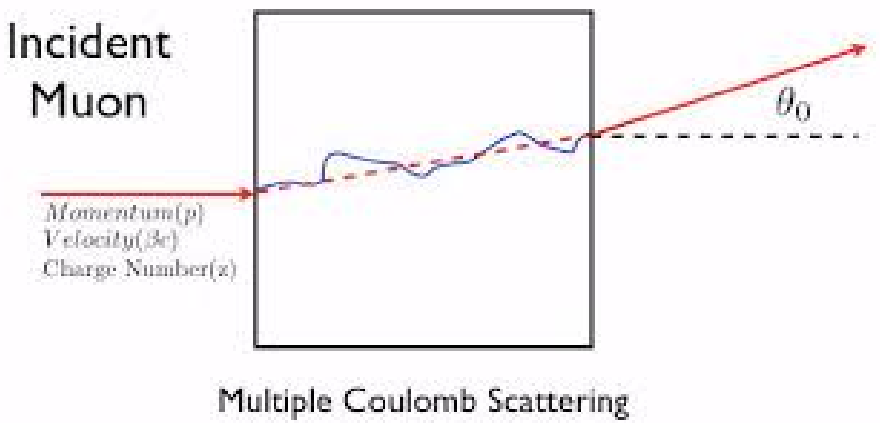
- Jupyter notebook at `/data/ComputerDay2017/Toys/MandelbrotSet.ipynb`

Elements of Simulation

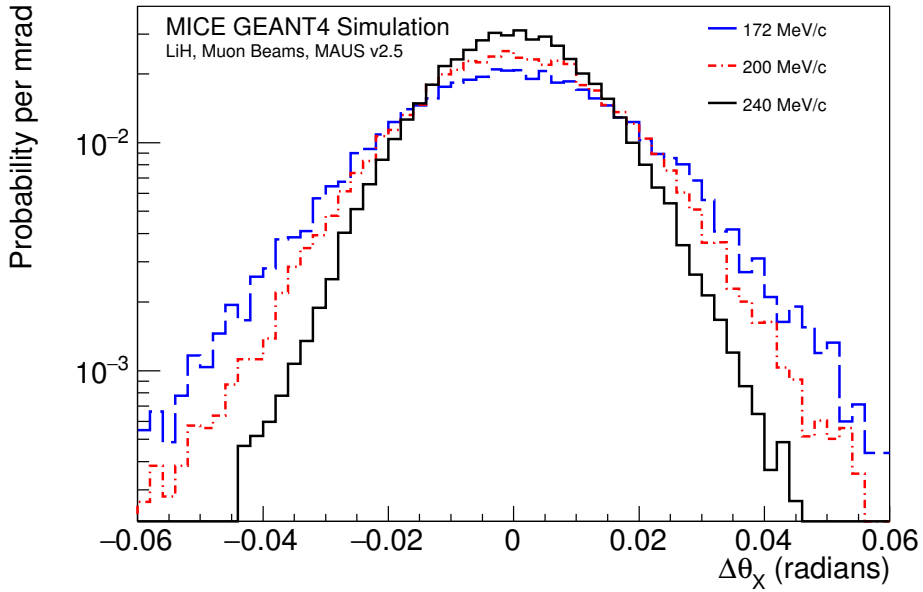
- Model
  - Geometry
  - Initial assumptions
  - Physics
- Initial state; sometimes dictated by random element
- Progression of the state;
  - with time or position
  - the dynamics of the system
  - random elements
- Final state;
  - What we are interested in
  - Need to dictate what form the output takes

A Physics Example: Particles Scattering in Matter

- Particles passing through a volume will change direction from Coulomb interactions with atoms.
- Many interactions will take place along path.



Multiple Coulomb Scattering



- R. Hoch et al. Muon tomography Algorithms for nuclear threat detection, Studies in Computational Intelligence, 214, 225-231 (2009).
- Bogomilov, M. et al, “Multiple Coulomb Scattering of muons in Lithium Hydride” Phys.Rev.D 106 (2022) 9, 092003.

Random Numbers

We use random selections every day

Computer "random numbers" are not random

- Are in fact "pseudo"-random.
- Are generated in a predicable way from an (integer) seed.
- Long periods between repetition.

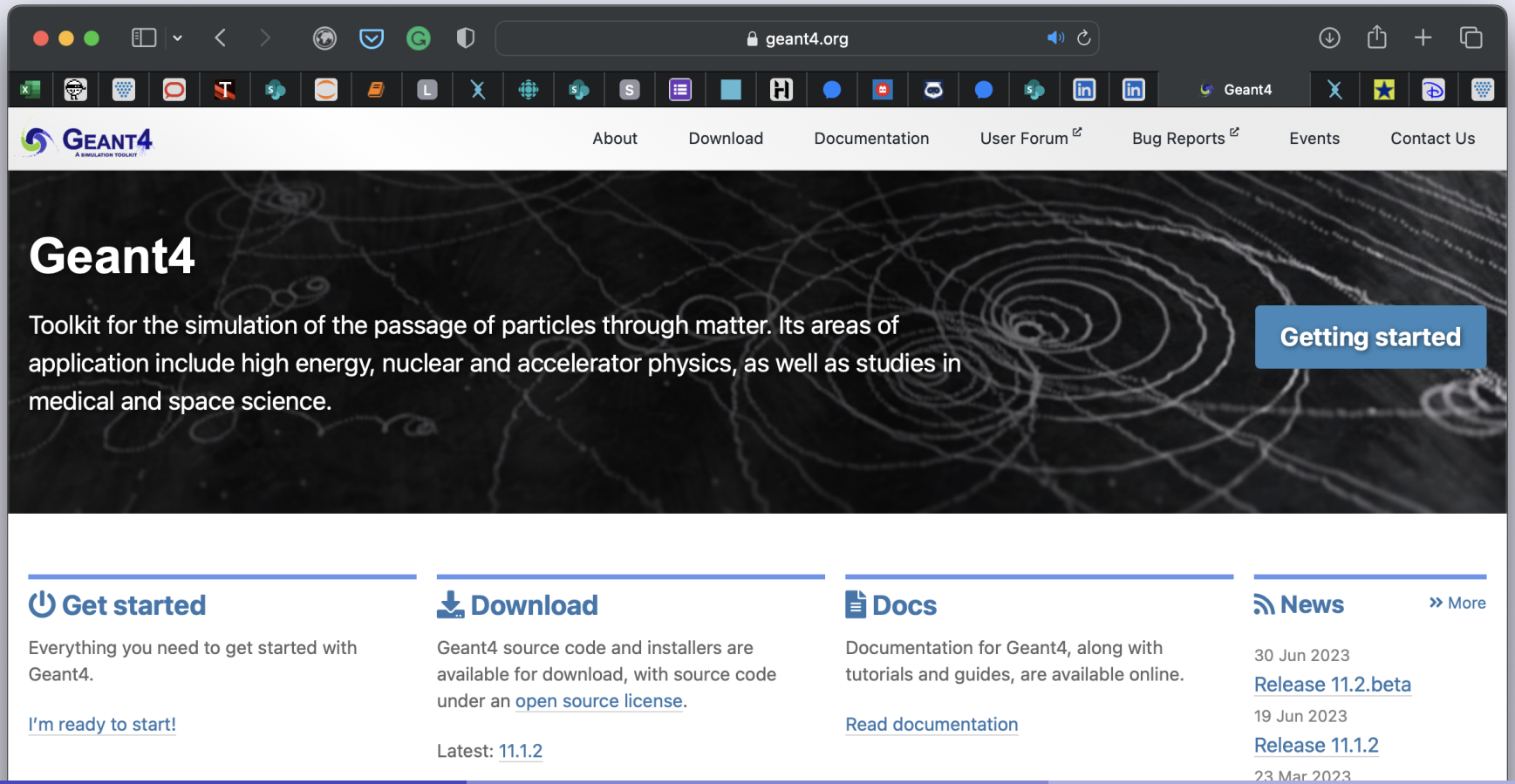
Common Elements for Particle Physics Experiments

- Large number of instances to reproduce experiment
- Particle definitions
  - Charge
  - Stability
  - Mass
  - Spin
- Interactions with matter
  - Propagation
  - Energy loss; discrete and continuous
  - Multiple scattering
  - Secondary production
- Material definitions
  - Optical properties
  - Electro-Magnetic fields
  - Chemical structure
  - Electron configuration

GEANT 4

- Many particle physics processes are applicable to many applications.
- Have been collected in a standardized package.

See www.geant4.org



Structure of a GEANT Simulation

There is a common structure

- Create the world volume
  - Define a detector geometry
- Define the Physics lists
  - What happens to your particle?
- Define the primary event
  - Particle type, direction, energy etc.
- Set the output
  - ROOT, Ntuple, Visual, ASCII

Example: GEANT/examples/basic/B2/B2a

- Log into "science2.snolab.ca"; i.e. `ssh -CY username@science2.snolab.ca`
- Copy the example

  `cp -r /data/ComputerDay2017/geant4.10.04.p01/examples/basic/B2/B2a ~/.`
- Open `exampleB2a.cc` in `vim` or `emacs`
- Find functions associated with processes on last page
- Attempt to build code:

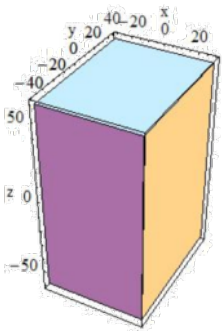
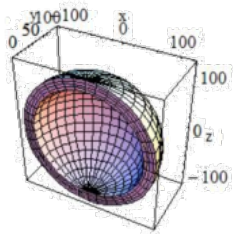
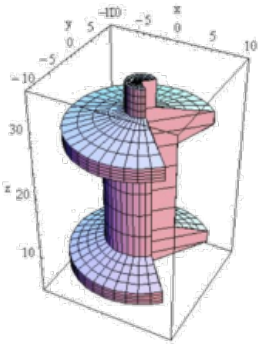
```
>source
/data/ComputerDay2017/geant4.10.04.p01-install/bin/geant4.sh
>mkdir ~/B2a-build
>cd ~/B2a-build >cmake -DGeant4_DIR=/data/ComputerDay2017/
geant4.10.04.p01-install/lib64/Geant4-10.4.1/ ~/B2a; make
```

Example: GEANT/examples/basic/B2/B2a

```
int main(int argc,char** argv)
{
  // Choose the Random engine

  G4Random::setTheEngine(new CLHEP::RanecuEngine);

  // Construct the default run manager

#ifdef G4MULTITHREADED
  G4MTRunManager * runManager = new G4MTRunManager;
#else
  G4RunManager * runManager = new G4RunManager;
#endif

  // Set mandatory initialization classes

  runManager->SetUserInitialization(new B2aDetectorConstruction());

  G4VModularPhysicsList* physicsList = new FTFP_BERT;
  physicsList->RegisterPhysics(new G4StepLimiterPhysics());
  runManager->SetUserInitialization(physicsList);

  runManager->SetUserInitialization(new B2ActionInitialization());

  // Initialize G4 kernel

  runManager->Initialize();

#ifdef G4VIS_USE
  // Initialize visualization
  G4VisManager* visManager = new G4VisExecutive;
```

Defining the Geometry

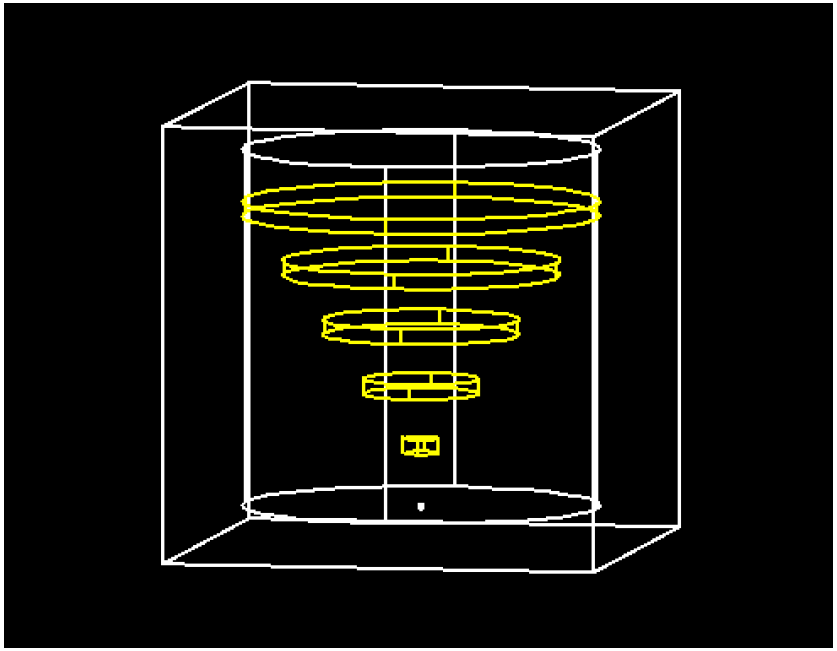
- Composed of primitive solids
  - Cubes
  - Spheres
  - Cylinders
  - Prisms... etc.
- Combinations of these solids
  - Unions
  - Subtractions
  - Arrays (parametrized volumes)
- Tessellated solids; Rotated solids

- Solids used to define logical volumes
- Each logical volume assigned a material
  - Atoms, molecules, compounds
  - Solids, liquids, or gases.
- Logical volumes placed in a mother volume by defining a physical volume
- All volumes have a mother (except the world volume).
- Defined as a class inheriting from "G4VUserDetectorConstruction"

Example: B2aDetectorConstruction

```
G4double targetLength =  5.0*cm; // full length of Target

G4double trackerLength = (fNbOfChambers+1)*chamberSpacing;

G4double worldLength = 1.2 * (2*targetLength + trackerLength);

G4double targetRadius  = 0.5*targetLength;   // Radius of Target
targetLength = 0.5*targetLength;             // Half length of the Target
G4double trackerSize   = 0.5*trackerLength;  // Half length of the Tracker

// Definitions of Solids, Logical Volumes, Physical Volumes

// World

G4GeometryManager::GetInstance()->SetWorldMaximumExtent(worldLength);

G4cout << "Computed tolerance = "
       << G4GeometryTolerance::GetInstance()->GetSurfaceTolerance()/mm
       << " mm" << G4endl;

G4Box* worldS
  = new G4Box("world",                                    //its name
              worldLength/2,worldLength/2,worldLength/2); //its size
G4LogicalVolume* worldLV
  = new G4LogicalVolume(
                worldS,   //its solid
                air,      //its material
                "World"); //its name

//  Must place the World Physical volume unrotated at (0,0,0).
//
G4VPhysicalVolume* worldPV
  = new G4PVPlacement(
                0,               // no rotation
                G4ThreeVector(), // at (0,0,0)
                worldLV,         // its logical volume
                "World",         // its name
                0,               // its mother  volume
                false,           // no boolean operations
                0,               // copy number
                fCheckOverlaps); // checking overlaps

// Target

G4ThreeVector positionTarget = G4ThreeVector(0,0,-(targetLength+trackerSize));

G4Tubs* targetS
  = new G4Tubs("target",0.,targetRadius,targetLength,0.*deg,360.*deg);
fLogicTarget
  = new G4LogicalVolume(targetS, fTargetMaterial,"Target",0,0,0);
new G4PVPlacement(0,               // no rotation
                  positionTarget,  // at (x,y,z)
                  fLogicTarget,    // its logical volume
                  "Target",        // its name
                  worldLV,         // its mother volume
                  false,           // no boolean operations
```



Exercise

- Open `src/B2aDetectorConstruction.cc`
- Try to change position of planes in detector volume.
- Add additional plane volumes within detector.

Sensitive Volumes

- GEANT needs to know what you want to measure
- A subset of volumes are defined to be "sensitive".
- A special class can be defined from "G4VSensitiveDetector".

Example: src/B2TrackerSD.cc

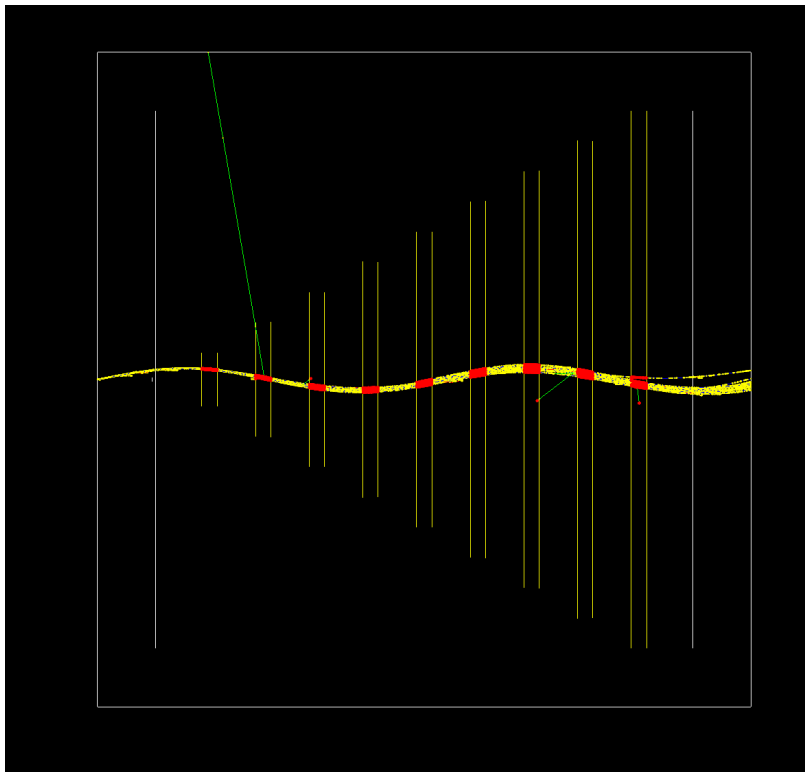
```
void B2TrackerSD::Initialize(G4HCofThisEvent* hce)
{
  // Create hits collection

  fHitsCollection
    = new B2TrackerHitsCollection(SensitiveDetectorName, collectionName[0]);

  // Add this collection in hce

  G4int hcID
    = G4SDManager::GetSDMpointer()->GetCollectionID(collectionName[0]);
  hce->AddHitsCollection( hcID, fHitsCollection );
}

//....oooOOOOOOooo........oooOOOOOOooo........oooOOOOOOooo........oooOOOOOOooo......

G4bool B2TrackerSD::ProcessHits(G4Step* aStep,
                                G4TouchableHistory*)
{
  // energy deposit
  G4double edep = aStep->GetTotalEnergyDeposit();

  if (edep==0.) return false;

  B2TrackerHit* newHit = new B2TrackerHit();

  newHit->SetTrackID  (aStep->GetTrack()->GetTrackID());
  newHit->SetChamberNb(aStep->GetPreStepPoint()->GetTouchableHandle()
                                               ->GetCopyNumber());
  newHit->SetEdep(edep);
  newHit->SetPos (aStep->GetPostStepPoint()->GetPosition());

  fHitsCollection->insert( newHit );

  //newHit->Print();

  return true;
}

//....oooOOOOOOooo........oooOOOOOOooo........oooOOOOOOooo........oooOOOOOOooo......

void B2TrackerSD::EndOfEvent(G4HCofThisEvent*)
{
  if ( verboseLevel>1 ) {
     G4int nofHits = fHitsCollection->entries();
     G4cout << "\n-------->Hits Collection: in this event they are " << nofHits
            << " hits in the tracker chambers: " << G4endl;
     for ( G4int i=0; i<nofHits; i++ ) (*fHitsCollection)[i]->Print();
  }
}

//....oooOOOOOOooo........oooOOOOOOooo........oooOOOOOOooo........oooOOOOOOooo......
```

- Initializes a "HitCollection"
- Describes how to process hits in volumes.
- Sensitive volumes designated in Geometry

```
void B2aDetectorConstruction::ConstructSDandField()
{
  // Sensitive detectors

  G4String trackerChamberSDname = "B2/TrackerChamberSD";
  B2TrackerSD* aTrackerSD = new B2TrackerSD(trackerChamberSDname,
                                            "TrackerHitsCollection");
  // Setting aTrackerSD to all logical volumes with the same name
  // of "Chamber_LV".
  SetSensitiveDetector("Chamber_LV", aTrackerSD, true);

  // Create global magnetic field messenger.
  // Uniform magnetic field is then created automatically if
  // the field value is not zero.
  G4ThreeVector fieldValue = G4ThreeVector();
  fMagFieldMessenger = new G4GlobalMagFieldMessenger(fieldValue);
  fMagFieldMessenger->SetVerboseLevel(1);

  // Register the field messenger for deleting
  G4AutoDelete::Register(fMagFieldMessenger);
}
```

Electromagnetic Fields

- Defined as part of the geometry
  - `ConstructSDandField()`
- Uniform field can be generated with
  - `G4GlobalMagFieldMessenger`
- Set the field value in the UI with
  - `/globalField/setValue vx vy vz unit`
- Non-uniform fields needs to be set using
  - `G4MagneticField *magfield;`



- Register with
  - `auto fieldBuilder = G4FieldBuild::Instance();`
  - `fieldBuilder->SetGlobalField(magfield);`
  - `fieldBuilder->ConstructFieldSetup();`

Physics List Definition

- Lists define different types of interactions
  - Electromagnetic
  - Hadronic
  - Decays
  - Transportation
  - Optical
  - Photolepton_hadron
  - Parametrisation
- Not all processes are valid or required for every situation
- Most lists are predefined as instances of "G4VModularPhysicsList" class.
- All processes need to be registered with run manager.

Primary Events

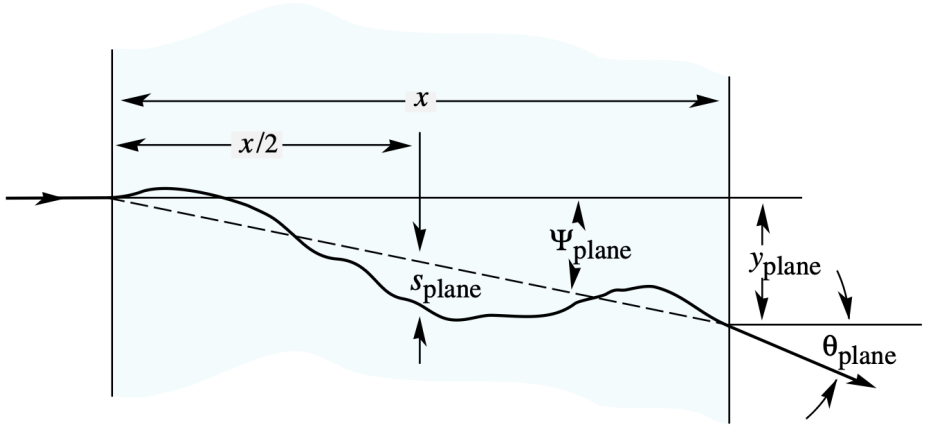
- Must be defined as class inheriting from "G4VUserPrimaryGeneratorAction"
- Defines the particle type, polarization, and initial momenta.
  - Protons, neutrons, pions, photons, muons, electrons all predefined.
  - Can also make your own particles.
- Special generators can be defined and referenced here.
  - Long baseline neutrino generators (GENIE)
  - Cosmic ray generators (CRY)
  - Particle decay distributions.
  - Other spectral generators.

Example: see "B2PrimaryGeneratorAction.cc"

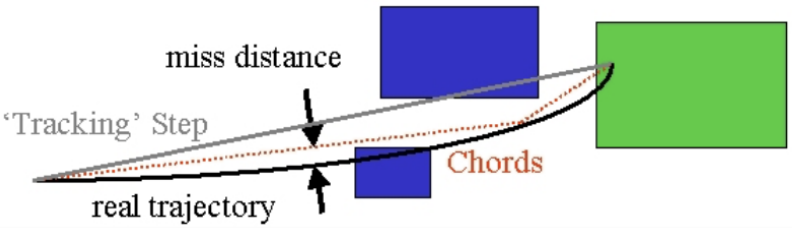
- Change position of event vertex
- Change momentum of outgoing particle
- Can also be altered in the UI or in macros

Particle transport

- Particles step through geometry
  - Path made in discrete steps
  - Steps are dynamic
  - Will break at material boundaries
- Discrete interactions accounted at the end of a step
  - Particle emission
  - Energy loss (i.e. Compton scattering, bremsstrahlung)
- Appropriate approximations taken for continuous changes
  - Multiple scattering
  - Ionization energy loss
  - Magnetic field interpolation



Representation of multiple scattering (From “Passage of Particles Through Matter”, pdg.lbl.gov, December 1, 2025)



Approximation of track curvature by GEANT4 in a magnetic field (From “G4 Documentation” section on Detector Definition and Response: Electromagnetic Field)

Progressing the Simulation

Actions taken every Event

- Could write out the state of an event to a root tree
- Could produce output that looks like data (i.e digitized hits).

Actions taken every Run

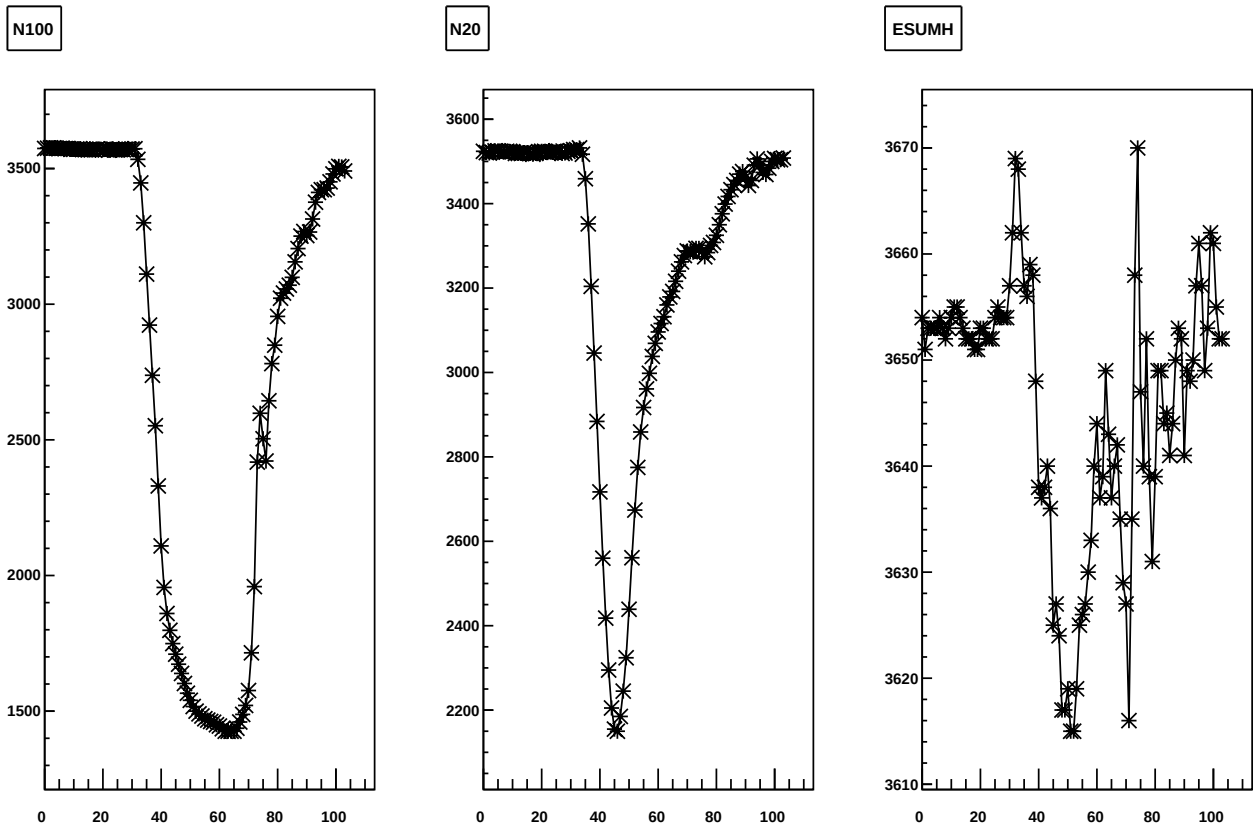
- Store a tree
- Write a histogram

Exercise: see `src/B2EventAction.cc`

- Open file
- Find `BeginOfEventAction` and `EndOfEventAction`
- Move event comment from End to beginning of event.
- Print position of vertex at End of event.

Digitization

- Make results look like data
- Largely detector dependent
  - Depends on the details of readout electronics
- Virtual class exists in GEANT4
  - G4VDigitizerModule



Generating Readable Output: Example B4a

- Geant can write results directly to ROOT

Examine Example B4a

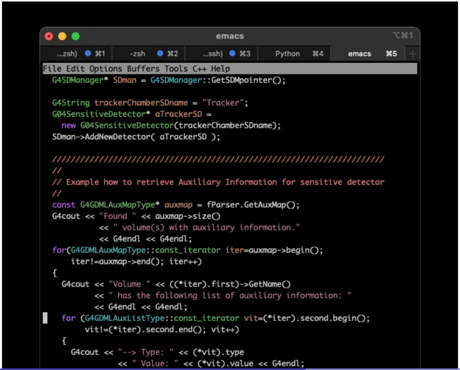
- Execute: `cp -r /data/ComputerDay2017/geant4.10.04.p01/examples/basic/B4/B4a ~/.`
- Histograms and Trees created in `B4a/src/B4RunAction.cc`
- Histograms and Trees filled in `B4a/src/B4aEventAction.cc`

```
B4RunAction::B4RunAction()
 : G4UserRunAction()
{
  // set printing event number per each event
  G4RunManager::GetRunManager()->SetPrintProgress(1);

  // Create analysis manager
  // The choice of analysis technology is done via selectin of a namespace
  // in B4Analysis.hh
  auto analysisManager = G4AnalysisManager::Instance();
-UU-:----F1  B4RunAction.cc   31% L48    (C++//l Abbrev) --------------------
  //

  // Creating histograms
  analysisManager->CreateH1("Eabs","Edep in absorber", 100, 0., 800*MeV);
  analysisManager->CreateH1("Egap","Edep in gap", 100, 0., 100*MeV);
  analysisManager->CreateH1("Labs","trackL in absorber", 100, 0., 1*m);
  analysisManager->CreateH1("Lgap","trackL in gap", 100, 0., 50*cm);

  // Creating ntuple
  //
  analysisManager->CreateNtuple("B4", "Edep and TrackL");
  analysisManager->CreateNtupleDColumn("Eabs");
  analysisManager->CreateNtupleDColumn("Egap");
  analysisManager->CreateNtupleDColumn("Labs");
  analysisManager->CreateNtupleDColumn("Lgap");
  analysisManager->FinishNtuple();
}
```

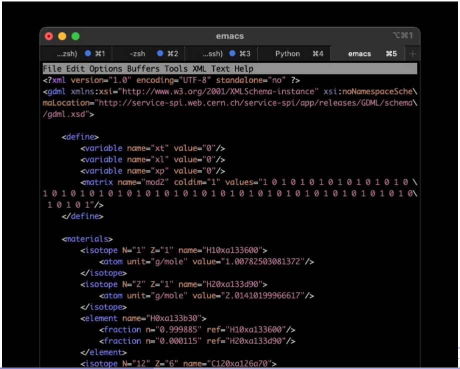
```
void B4aEventAction::EndOfEventAction(const G4Event* event)
{
  // Accumulate statistics
  //

  // get analysis manager
  auto analysisManager = G4AnalysisManager::Instance();

  // fill histograms
  analysisManager->FillH1(0, fEnergyAbs);
  analysisManager->FillH1(1, fEnergyGap);
  analysisManager->FillH1(2, fTrackLAbs);
  analysisManager->FillH1(3, fTrackLGap);

  // fill ntuple
  analysisManager->FillNtupleDColumn(0, fEnergyAbs);
  analysisManager->FillNtupleDColumn(1, fEnergyGap);
  analysisManager->FillNtupleDColumn(2, fTrackLAbs);
  analysisManager->FillNtupleDColumn(3, fTrackLGap);
  analysisManager->AddNtupleRow();
```

Geometry Persistency; GDML Running

- GDML provides a way to arbitrarily define the Geometry at run-time
- Allows to replicate many of the behaviours of GEANT native geometry
  - Primative solids
  - Parametrized solids
  - Sensitive volumes
  - Electric and Magnetic Fields



```
File Edit Options Buffers Tools C++ Help
  G4SDManager* SDman = G4SDManager::GetSDMpointer();

  G4String trackerChamberSDname = "Tracker";
  G04SensitiveDetector* aTrackerSD =
    new G04SensitiveDetector(trackerChamberSDname);
  SDman->AddNewDetector( aTrackerSD );

  ///////////////////////////////////////////////////////////////////////
  //
  // Example how to retrieve Auxiliary Information for sensitive detector
  //
  const G4GDMLAuxMapType* auxmap = fParser.GetAuxMap();
  G4cout << "Found " << auxmap->size()
            << " volume(s) with auxiliary information."
            << G4endl << G4endl;
  for(G4GDMLAuxMapType::const_iterator iter=auxmap->begin();
      iter!=auxmap->end(); iter++)
  {
    G4cout << "Volume " << ((*iter).first)->GetName()
           << " has the following list of auxiliary information: "
           << G4endl << G4endl;
    for (G4GDMLAuxListType::const_iterator vit=(*iter).second.begin();
         vit!=(*iter).second.end(); vit++)
    {
      G4cout << "--> Type: " << (*vit).type
                << " Value: " << (*vit).value << G4endl;
```



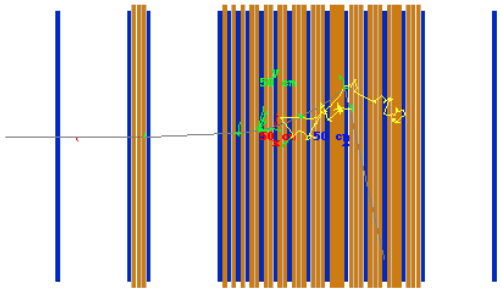
```
File Edit Options Buffers Tools XML Text Help
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<gdml xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSche\
maLocation="http://service-spi.web.cern.ch/service-spi/app/releases/GDML/schema\
/gdml.xsd">

    <define>
        <variable name="xt" value="0"/>
        <variable name="xl" value="0"/>
        <variable name="xp" value="0"/>
        <matrix name="mod2" coldim="1" values="1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 \
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0\
 1 0 1 0 1"/>
    </define>

    <materials>
        <isotope N="1" Z="1" name="H10xa133600">
            <atom unit="g/mole" value="1.00782503081372"/>
        </isotope>
        <isotope N="2" Z="1" name="H20xa133d90">
            <atom unit="g/mole" value="2.01410199966617"/>
        </isotope>
        <element name="H0xa133b30">
            <fraction n="0.999885" ref="H10xa133600"/>
            <fraction n="0.000115" ref="H20xa133d90"/>
        </element>
        <isotope N="12" Z="6" name="C120xa126a70">
```

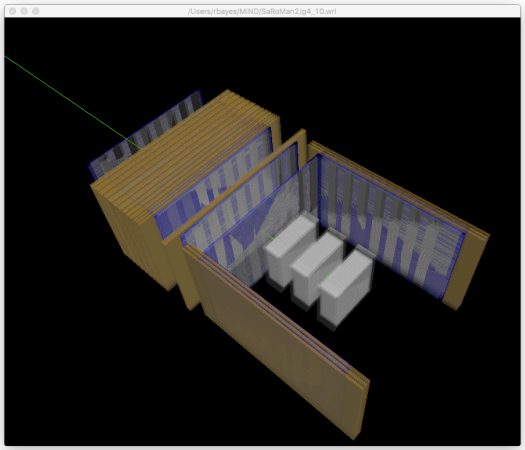
Executing the Program

- Interactive Mode
  - Necessary for OGL viewers
  - A Qt gui is available.
  - i.e. `./B2a-build/exampleB2a`
- Macro file (.mac files)
  - Requires a messenger to parse the macro
  - See `B2aDetectorMessenger.cc` and `init_vis.mac`
- Batch execution
  - All run information contained in the executable
  - See `exampleB2a.in`

Detector Visualization

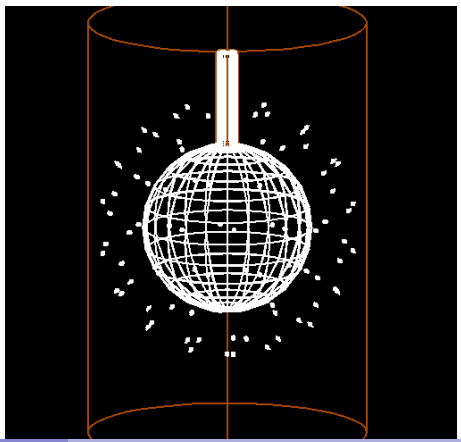
- Open GL
- Needs X11 support (Unix)





- VRML representation
- Requires third party viewer to use

- HepRApp
- Java viewer available



Further Comments

- Many things not mentioned
- For any given task look for examples.
- There are further functionalities to explore
- Feel free to try new things.