GEANT4 Basics

NPRE 441
04/22/2026
Yifei Jin, PhD, Research Scientist, NPRE
yifeij5@illinois.edu

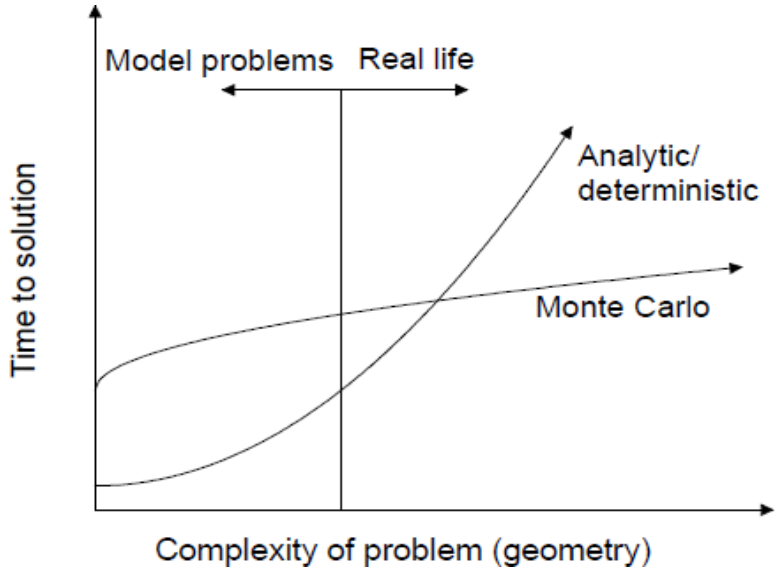
What We Will Cover in This Lecture

- Quick recall
- How to use UI commands to control a simulation application
- Assignment #3 with UI commands: tutorial and demo
- Architecture and Basics of Geant4 simulation applications
  - Open the “black box”

Recall: What is Monte Carlo Method?

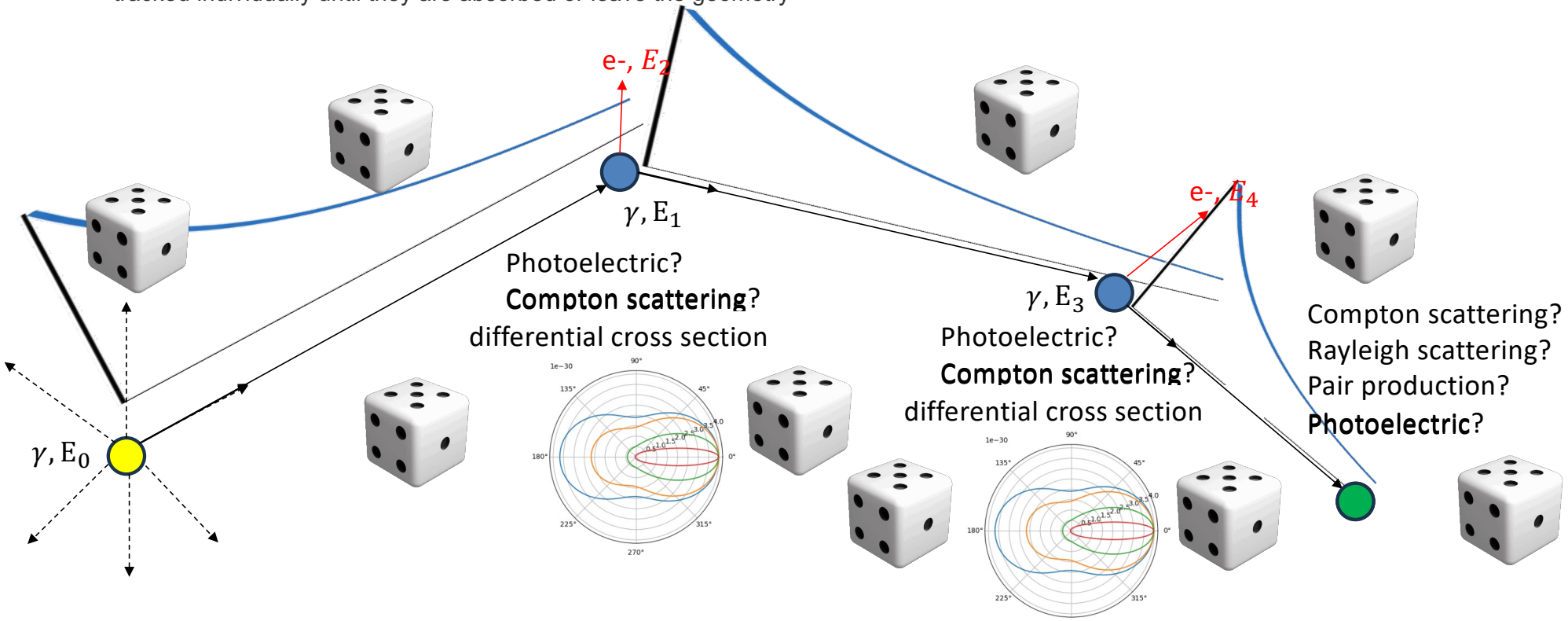
- Numerical solution of a (complex) macroscopic problem, by simulating the microscopic interactions among the components
- Uses random sampling, until statistical convergence is achieved (name after Monte Carlo's casino)

Key takeaway:
Use Random Number to solve Deterministic problem



Recall: What is Actually Happening when using a MC Code to Generate a Random Particle Track?

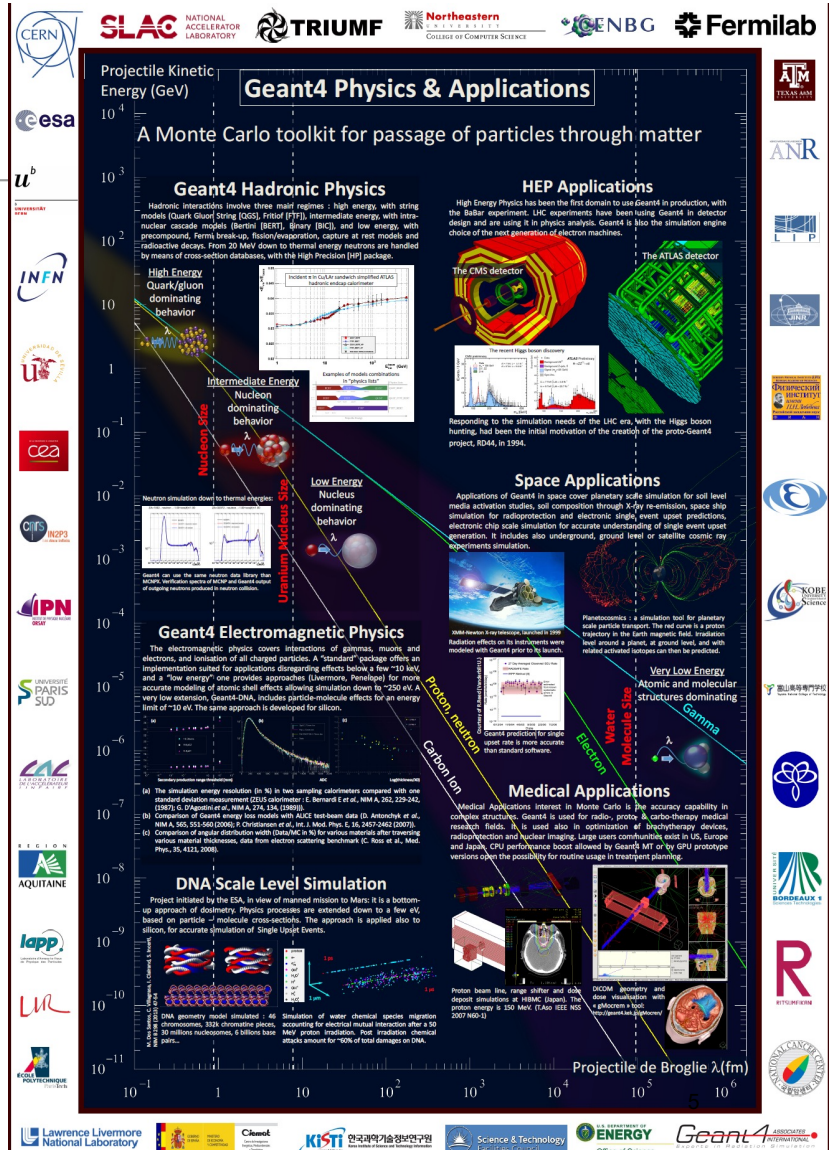
- A particle's trajectory is treated as a sequence of independent "**steps**" and probabilistic events.
- **Distance to Next Collision:** The step length (*s*) is
- **Energy and Angle:** Physics models are used to sample the final state (kinematics)
- **Type of Interaction:** Competing processes are sampled according to their **differential cross-sections**
- **Secondary Production:** The primary particle's properties are updated, and any produced **secondaries** are added to a stack to be tracked individually until they are absorbed or leave the geometry



Recall: What is Geant4?

- Toolkit created by CERN to simulate the passage of particles through matter.
- Designed to make the physics used transparent within the toolkit, handle a wide range of geometries, and enable an easy adaptation of different physics to fit the application.

You can find all you need in
[Book For Application Developers]



How to Use GEANT4?

- As an application programmer (Engineers who build a scanner):
  - ➢develops the simulation application by making use of the components provided by the toolkit
  - ➢requires solid knowledge of both the C++ programming language and the simulation toolkit
- As an end-user (Physicians who use the scanner):
  - ➢runs the simulation application with (User interface) UI commands
  - ➢control the program flow of a Geant4 simulation application without using C++ language

As an end-user to perform the Monte Carlo study, you will need **UI Commands**

Control the program flow of a Geant4 simulation application without using C++ language

Basics of UI Command Syntax

- A UI command consists of:
  - command directory
  - command
  - parameter(s)

/run/verbose 1
/gun/particle proton

- A parameter can be a type of string, boolean, integer or double:
  - space is a delimiter
  - use double-quotes (“”) for strings
- A parameter can be omitted. Its default value will be taken in this case

Submitting a Command using the GEANT4

- Geant4 UI commands can be issued in 3 different ways by:
  - (G)UI **interactive** command submission (see more later)
  - **batch** mode using a **macro file** (see more later)
  - **hard-coded** commands in the application (slow):

main()

```
G4UImanager* UI = G4UImanager::GetUIpointer();
UI->ApplyCommand("/run/verbose 1");
```

What is a Macro File ?

- A macro file is an ASCII file that contains UI commands
- All commands must be given with their full-path directories
- Use "#" for comment a line
  - ➢from the first "#" to the end of the line will be ignored
  - ➢comment lines will be echoed if /control/verbose is set to 2
- Macro file can be executed
  - ➢interactively or in other macro files
    - `/control/execute macro_file_name`
  - ➢hard-coded

main()

```
G4UImanager* UI = G4UImanager::GetUIpointer();
UI->ApplyCommand("/control/execute macro_file_name");
```

Batch Mode v.s. Interactive Mode

See the main method of the B1 example:***B1/exampleB1.cc*** The 3 different ways of command submission: hard-coded, batch and interactive.

```
int main(int argc,char** argv)
{
  // Detect interactive mode (if no arguments) and define UI
  //
  G4UIExecutive* ui = 0;
  if ( argc == 1 ) {
    ui = new G4UIExecutive(argc, argv);
  }
  ...
```

Detect batch/interactive mode

```
  ...
  // Get the pointer to the User Interface manager
  G4UImanager* UImanager = G4UImanager::GetUIpointer();
  // Process macro or start UI session
  //
  if ( ! ui ) {
    // batch mode
    G4String command = "/control/execute ";
    G4String fileName = argv[1];
    UImanager->ApplyCommand(command+fileName);
  } else {
    // interactive mode
//    UImanager->ApplyCommand("/control/execute init_vis.mac");
    ui->SessionStart();
    delete ui;
  }
```

batch mode: process macro

interactive mode: start an UI session

Batch Mode v.s. Interactive Mode

SLURM (Simple Linux Utility for Resource Management) is an open-source, highly scalable cluster management and job scheduling system for Linux clusters. It allocates, manages, and monitors compute resources, managing job queues and ensuring fair resource distribution.

- Hard-coded command execution:
  - ➢can be easily identified in the previous slide
  - ➢the C++ code needs to be rebuild after all changes
- Batch mode using a Macro file (submit to SLURM for official run):
  - ➢in batch mode, the commands in the macro file will be executed by processing the macro file using the (hard-coded)
  - ➢*./yourApp macrofile*

```
G4String command = "/control/execute ";
G4String fileName = argv[1];
UImanager->ApplyCommand(command+fileName);
```

- Interactive mode (use to verify geometry):
  - ➢commands are submitted and processed one-by-one through a Geant4 User Interface Session
  - ➢there are several different types of interfaces available:
  - ➢Qt-GUI, GAG-GUI(java based), Xm-GUI (Motif based) or simple C-shell like character terminals (tcsh, csh)

GEANT4 Built-in Commands

- **/run/**: Controls simulation execution.
  - /run/initialize: Initializes the Geant4 kernel.
  - /run/beamOn <N>: Starts a run with *N* events.
  - /run/verbose <level>: Sets the verbosity level for run management.
- **/gun/**: Configures primary particles (requires G4ParticleGun).
  - /gun/particle <name>: Sets particle type (e.g., e-, proton, gamma).
  - /gun/energy <value> <unit>: Sets kinetic energy (e.g., 100 MeV).
  - /gun/position <x> <y> <z> <unit>: Sets the source position.
- **/control/**: Macro and alias management.
  - /control/execute <file>: Executes a macro file.
  - /control/alias <name> <value>: Creates a variable alias.
  - /control/loop <macro> <var> <start> <stop> <step>: Repeatedly executes a macro.
- **/vis/**: Controls Visualization (requires an active driver like OpenGL or Qt).
  - /vis/open <driver>: Opens a visualization window.
  - /vis/drawVolume: Draws the detector geometry.
  - /vis/viewer/set/viewpointThetaPhi <theta> <phi>: Changes camera angle.

Custom Commands

- Creating your own UI command to manipulate some of the properties of your own object (e.g. write a command to be able to `setMaterial [g4 NIST material name]`
- Write your **`Messenger`** class (with defining/adding your **`UIcommand`**-s) to your object and add to your target object (see more at Book For Application Developers)

ShieldMessenger.cc

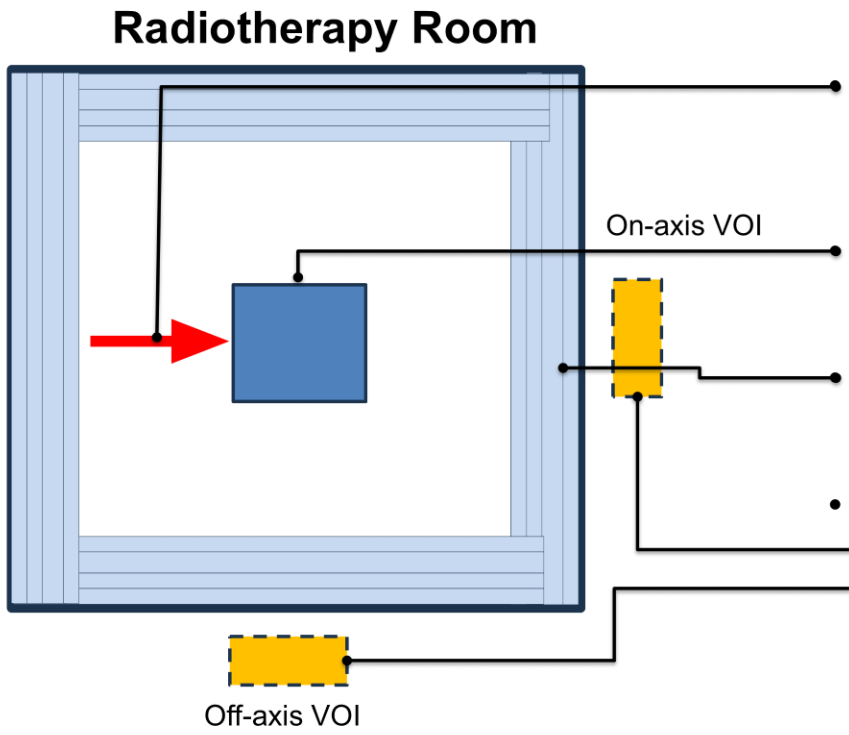
```
G4UIcommand* MakeMaterialCmd(const G4String& path, G4UImessenger* owner)
{
  auto cmd = new G4UIcommand((path + "setMaterial").c_str(), owner);
  cmd->SetGuidance("Set material of shield layer (0-4).");
  cmd->SetGuidance(("E.g.: " + path + "setMaterial 0 G4_CONCRETE").c_str());
  cmd->AvailableForStates(G4State_PreInit, G4State_Idle);

  auto p1 = new G4UIparameter("layerIndex", 'i', false);
  p1->SetGuidance("Layer index (0 to 4)");
  p1->SetParameterRange("layerIndex >= 0 && layerIndex < 5");
  cmd->SetParameter(p1);

  auto p2 = new G4UIparameter("materialName", 's', false);
  p2->SetGuidance("NIST material name (e.g. G4_CONCRETE, G4_Pb, G4_Fe)");
  cmd->SetParameter(p2);

  return cmd;
}
```

shield_custom.mac

```
# Front wall (+Z) — primary beam exit, heaviest shielding needed
/B1/shield/front/setMaterial  0 G4_CONCRETE
/B1/shield/front/setThickness 0 150 cm
/B1/shield/front/setMaterial  1 G4_POLYETHYLENE
/B1/shield/front/setThickness 1 20 cm
/B1/shield/front/setThickness 2 0 cm
/B1/shield/front/setThickness 3 0 cm
/B1/shield/front/setThickness 4 0 cm
```

Recall: Assignment #3

Use Geant4 to simulate particle transport in radiotherapy and analyze how shielding influences the doses in on-axis and off-axis volumes



The Beam

- Monoenergetic beam (e-, gamma, proton, etc.)

The Target

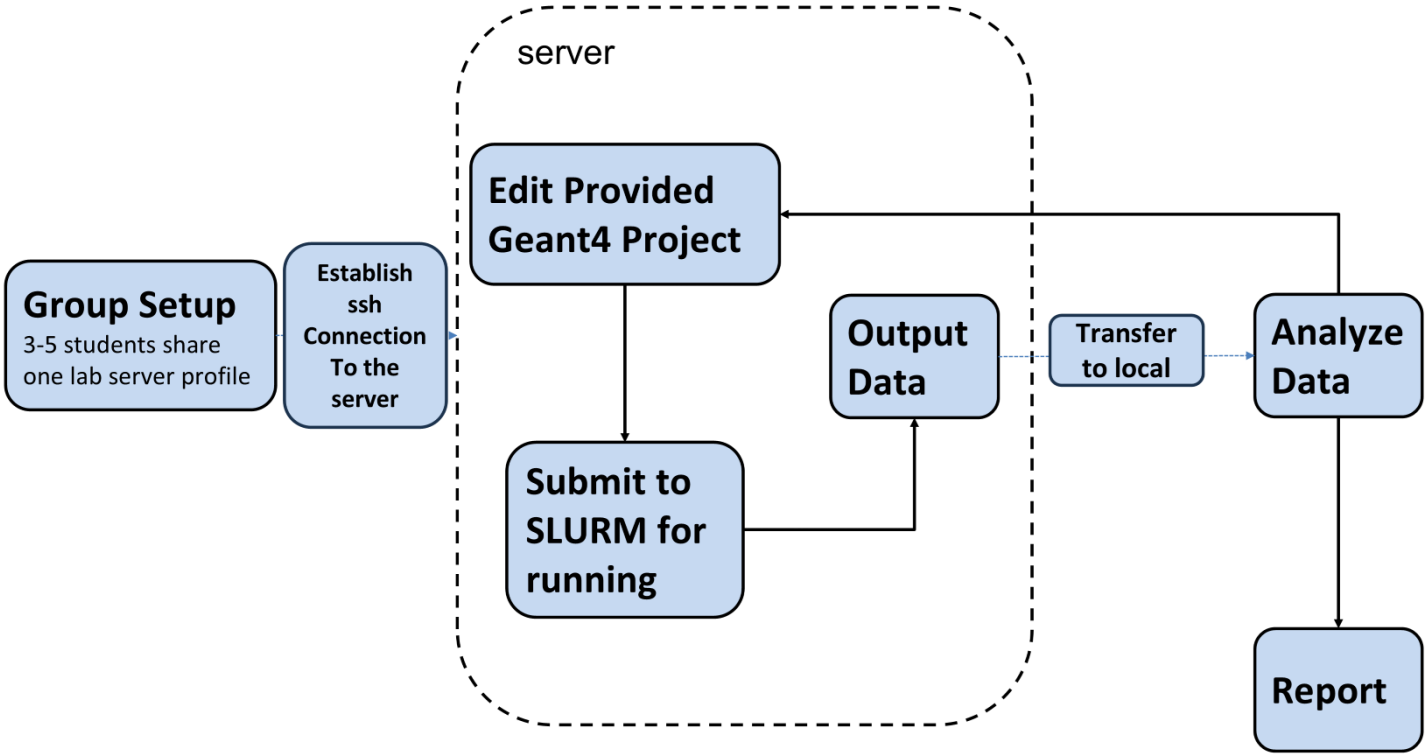
- Fixed tissue-equivalent phantom

The Shielding Wall

- Configurable multi-layer shielding wall

- **The Volumes of Interest (VOIs)**
  - On-axis (captures primary beam)
  - Off-axis (captures scattered radiation)

Workflow


server
Edit Provided Geant4 Project
Group Setup
3-5 students share one lab server profile
Establish ssh Connection To the server
Output Data
Transfer to local
Analyze Data
Submit to SLURM for running
Report

Step 0 — Connect the Server through SSH

- **Step 0.1: Install VcXsrv**
  - Download and install VcXsrv.
  - Launch XLaunch.
  - Select: Multiple windows.
  - Select: Start no client.
  - Check: Disable access control.
  - Click Finish
  - Display number: 0.
  - Finish and keep VcXsrv running.
- **Step 0.2: Set DISPLAY (one-time setup)**
  - Open PowerShell and run:
    setx DISPLAY localhost:0
  - Close PowerShell

Step 0 — Connect the Server through SSH

Step 0.3: Install VS Code and Remote-SSH extension.

Step 0.4: Configure VS Code to use Windows OpenSSH

In VS Code File->Preferences->Settings:

Search for 'Remote.SSH: Path'

Set to:

C:\Windows\System32\OpenSSH\ssh.exe

Restart VS Code **as administrator** completely.

Step 0.5: Configure SSH

In VS code, press Ctrl+Shift+P, search 'Remote-SSH: Connect to Host'. Select ‘Configure SSH Hosts…’

Select C:\Users\<your username>\.ssh\config

```
Host npre441
  HostName <SERVER_IP>
  User npre441-XX
  ForwardX11 yes
  ForwardX11Trusted yes
  SetEnv DISPLAY=localhost:0
```

Step 0 — Connect the Server through SSH

Step 0.6: In VS code, press Ctrl+Shift+P, search 'Remote-SSH: Connect to Host'.

Select host name: npre441 or manually enter npre441-XX@npre441

Select Platform OS: Linux

Enter Password

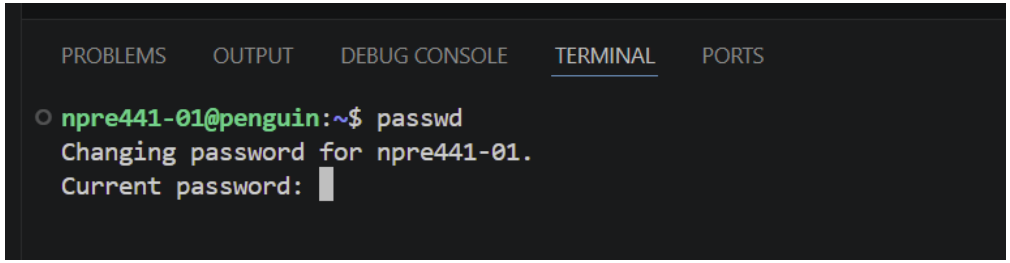
Open a remote terminal (View->Terminal)and run:

echo $DISPLAY

xclock

If a clock appears, setup is successful.

Do this in terminal after your first log-in



Step 1 — Build (ONLY when needed)

- > mkdir -p ~/geant4_course
- > cp -r /shared/npre441/ShieldingProject ~/geant4_course/
- Enter Container:
- > apptainer shell /shared/containers/geant4_course.sif

- > cd ~/geant4_course/ShieldingProject
- > mkdir -p build
- > cd build
- > cmake ..
- > make -j4

```
npre441-01@penguin:~$ apptainer shell /shared/containers/geant4_course.sif
Apptainer> cd ~/geant4_course/ShieldingProject
mkdir -p build
cd build
cmake ..
make -j4
-- Configuring done
-- Generating done
-- Build files have been written to: /home/npre441-01/geant4_course/ShieldingPr
oject/build
Consolidate compiler generated dependencies of target exampleB1
[100%] Built target exampleB1
Apptainer>
```

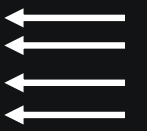
Building and compiling C++ involves transforming human-readable source code into a machine-readable executable file.

Apptainer is an open-source, secure container platform. It packages applications and dependencies into a single, portable, and immutable file, allowing users to run complex software consistently across different environments without requiring root privileges

Step 2 — Edit Macro

Example:
~/geant4_course/ShieldingProject/build/custom_shielding.mac:

```
# --- Front wall (beam exit, primary shielding) ---
/B1/shield/front/setMaterial  0 G4_CONCRETE
/B1/shield/front/setThickness 0 10 cm
/B1/shield/front/setMaterial  1 G4_Al
/B1/shield/front/setThickness 1 10 cm
/B1/shield/front/setMaterial  1 G4_Pb
/B1/shield/front/setThickness 2 10 cm
/B1/shield/front/setThickness 3 0 cm
/B1/shield/front/setThickness 4 0 cm
```



First layer

Second layer

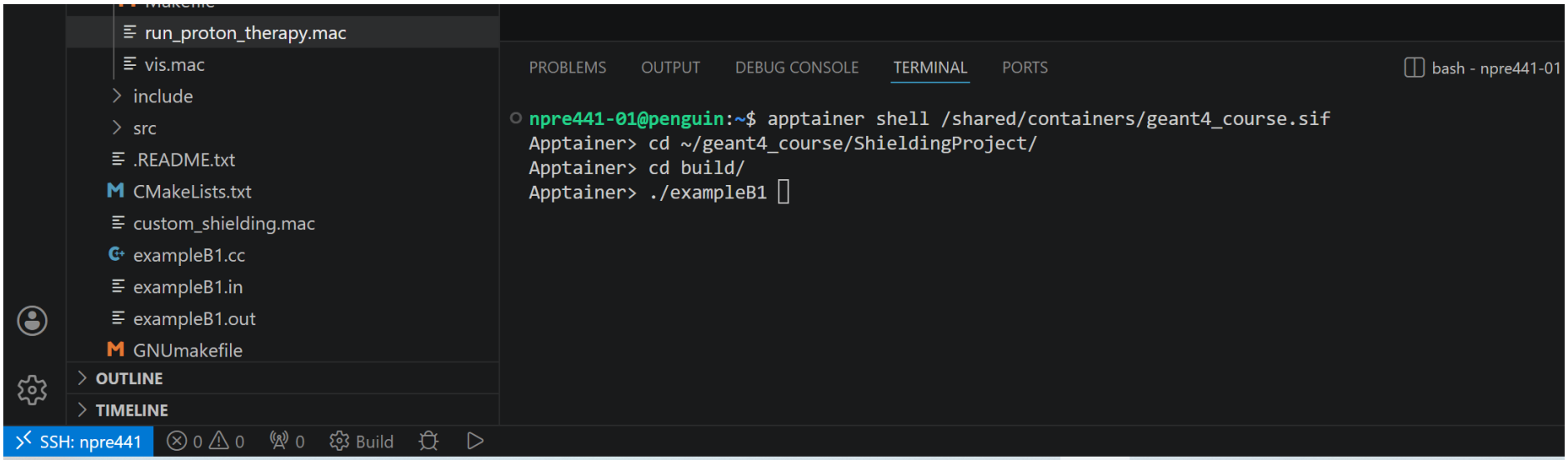
~/geant4_course/ShieldingProject/build/run_proton_therapy.mac:

```
/gun/particle gamma
/gun/energy 10 MeV
/run/beamOn 1000000
```

List of G4 internal Material database through NIST
https://geant4-userdoc.web.cern.ch/UsersGuides/ForApplicationDeveloper/html/Appendix/materialNames.html

Step 3 — Visualize Geometry

- In ~/geant4_course/ShieldingProject/build/:



Step 4 — Create SLURM Job

- mkdir –p ~/jobs
- mkdir –p ~/logs **~/jobs/run.slurm**

```
#!/bin/bash
#SBATCH --job-name=geant4_test
#SBATCH --partition=class
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=10
#SBATCH --time=10:00:00
#SBATCH --output=logs/%x_%j.out
#SBATCH --error=logs/%x_%j.err

# Go to your project
cd ~/geant4_course/ShieldingProject/build

# Set threads
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

# Run inside container
apptainer exec \
  /shared/containers/geant4_course.sif \
  ./exampleB1 run_proton_therapy.mac
```

Your job name

Number of CPUs you requested from the server should match your Geant4 config

Time limit

```
# --- Initialize ---
/run/numberOfThreads 4
/run/initialize
```

Your macro file

The command you submitted

```
#!/bin/bash
#SBATCH --job-name=geant4_test
#SBATCH --partition=class
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=10
#SBATCH --time=10:00:00
#SBATCH --output=logs/%x_%j.out
#SBATCH --error=logs/%x_%j.err


# Go to your project
cd ~/geant4_course/ShieldingProject/build


# Set threads
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK


# Run inside container
apptainer exec \
  /shared/containers/geant4_course.sif \
  ./exampleB1 run_proton_therapy.mac
```

Step 5 — Submit&Check Job

- sbatch jobs/run.slurm

```
npre441-01@penguin:~/npre441$ sbatch apptainer_shieldapp.slurm
Submitted batch job 12
```

- squeue -u $USER

```
npre441-01@penguin:~/npre441$ squeue -u $USER
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
                 2     class    hello npre441- PD       0:00      1 (PartitionConfig)
                12     class geant4_t npre441-  R       0:07      1 penguin
```

The job I just submitted, ST=R is running.

If your job is not listed, it's already done.

Output—In ~/logs/

- For each VOI, please extract the following quantities from the output of GEANT 4:

1. Energy deposition in the VOI(s) from:

- Beta particles incident on the volume of interest (VOI)
- Photons
- Protons
- Neutrons

2. Total energy deposition in the VOI(s) from all particles.

3. Volume and Density

A Few Important Notes to Remember

- You ONLY need to build:
  - ➢First time
  - ➢After editing C++ source code

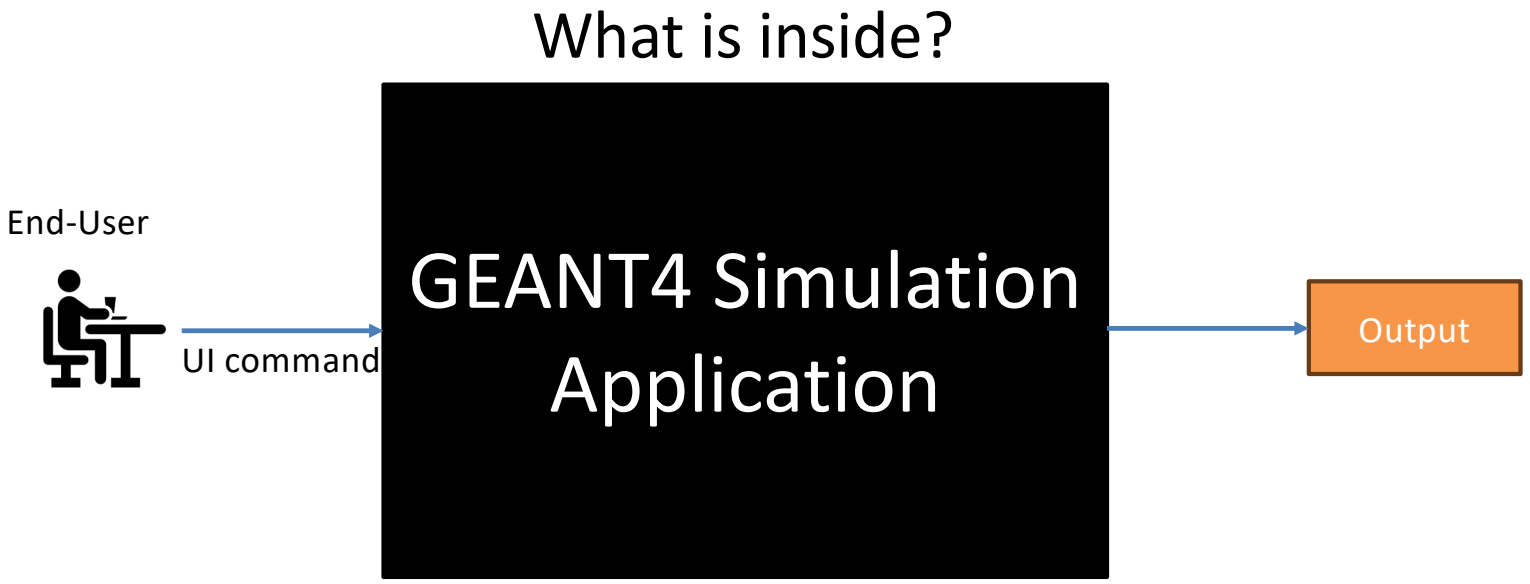
-- Build once, run many times

- If you ONLY change macro files:
  - ➢→ NO rebuild needed

Building and compiling C++ involves transforming human-readable source code into a machine-readable executable file.

- Enter Apptainer for visualization with interactive mode
- Use regular terminal and SLURM for simulation jobs

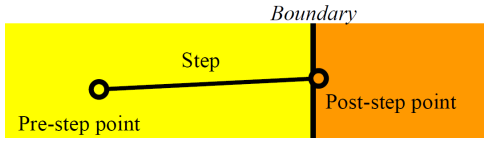
Let's Open the "Black Box"

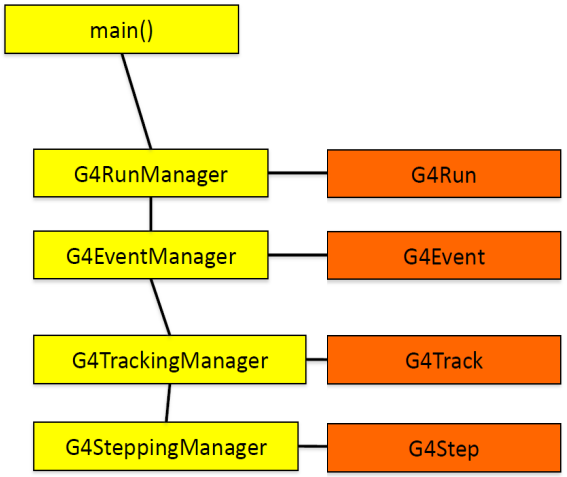


Let's explore the underlying architecture of GEANT4 simulation

Terminology in Geant4

- Run
  - ➢As an analogy of the real experiment, a run of Geant4 starts with “Beam On”.
  - ➢A run consists of one event loop (multiple events).
- Event
  - ➢Basic unit of simulation, stop when no track
  - ➢Primary tracks are generated at the beginning
  - ➢Primary particles and resultant secondary particles are tracked
- Track
  - ➢A snapshot of the current instance. A primary particle can lead to millions of secondary tracks.
  - ➢Deleted when goes out of the “world”, disappears (by e.g. decay), at rest, killed by cut
- Step and Step Point
  - ➢“Delta” information to a track.
  - ➢Has two points, each knows the volume and material
- Cut
  - ➢A production threshold. (measured in length, not energy) that determines the minimum range a secondary particle must be able to travel to be explicitly created and tracked.

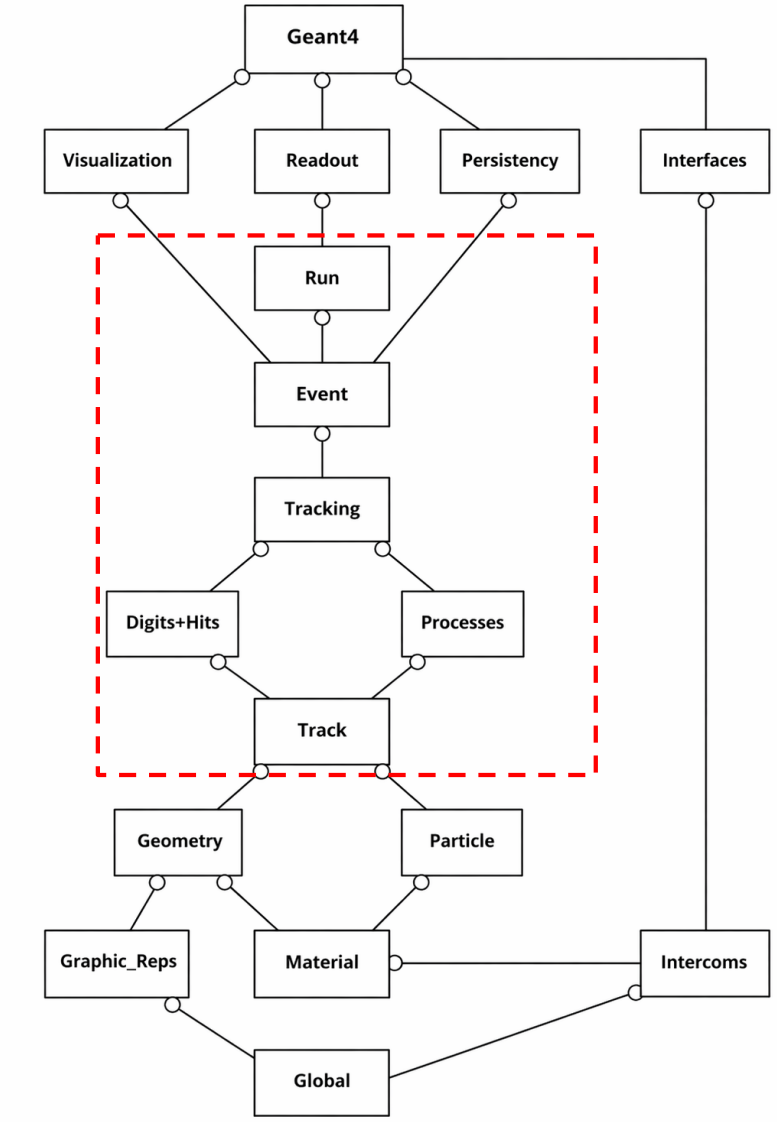




Overall Structure

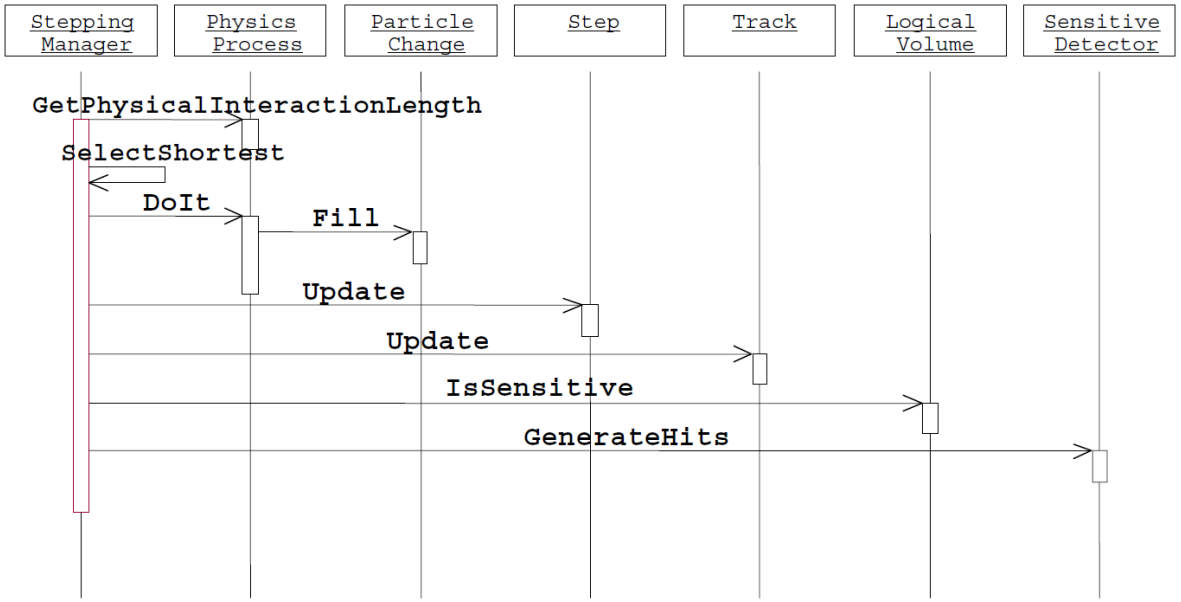
- The Top Level Category Diagram of the Geant4 toolkit.
- No circular dependence.
- Categories at the bottom are used by virtually all higher categories.

- Geant4 Kernel
  - ➢Handles run, event, track, step, hit, trajectory.



Processes in Geant4

- In Geant4, particle transportation is a process as well, by which a particle interacts with geometrical volume boundaries and field of any kind.
- Each particle has its own list of applicable processes. At each step, all processes listed are invoked to get proposed physical interaction lengths.
- The process which requires the shortest interaction length (in space-time) limits the step.



Unit System

• Internal unit system used in Geant4 is completely hidden not only from user's code but also from Geant4 source code implementation.

• Each hard-coded number must be multiplied by its proper unit.
radius = 10.0 * cm; kineticE = 1.0 * GeV;

• To get a number, it must be divided by a proper unit. **G4cout << eDep / MeV << " [MeV]" << G4endl;**

• Most of commonly used units are provided and user can add his/her own units.

• By this unit system, source code becomes more readable and importing / exporting physical quantities becomes straightforward. –

Application Developer and User Classes

To build an application:

- Define your geometrical setup
  - ➢Material, volume
- Define physics to get involved
  - ➢Particles, physics processes/models
  - ➢Cuts (Production thresholds)
- Define how an event starts
  - ➢Primary particle (track) generation
- Extract information useful to you
- You may also want to
  - ➢Visualize geometry, trajectories and physics output –
  - ➢Utilize (Graphical) User Interface
  - ➢Define your own UI commands
  - ➢etc.

Note : You don't have to create an application from nothing. There are plenty of G4 example applications from basic to advance for reference.

The provided shielding project is modified based on example B1.

You can still run a simulation without this, just NO output.

User Classes

A class is a user-defined data type, which holds its own data members and member functions that can be accessed and used by creating an instance of that class.

- main() – Geant4 does not provide *main().*
- Initialization classes
- Use *G4RunManager::SetUserInitialization()* to define. (Invoked at the initialization)
  - G4VUserDetectorConstruction
  - G4VUserPhysicsList
  - G4VUserActionInitialization
- Action classes
- Instantiate in your G4VUserActionInitialization. (Invoked during an event loop)
  - G4VUserPrimaryGeneratorAction
  - G4UserRunAction
  - G4UserEventAction
  - G4UserStackingAction
  - G4UserTrackingAction
  - G4UserSteppingAction

Note : classes written in red are mandatory.

main()

- In your *main(),* you have to
  - ➢ Construct G4RunManager (sequential mode) or G4MTRunManager (multithreaded mode)
  - ➢ Set user mandatory initialization classes to RunManager
    - ❑ G4VUserDetectorConstruction -- Describe your detector
    - ❑ G4VUserPhysicsList -- Select physics processes
    - ❑ G4VUserActionInitialization

- You can define VisManager, (G)UI session, optional user action classes, and/or your persistency manager in your *main()*.

- G4VUserDetectorConstruction -- Describe your detector
- G4VUserPhysicsList -- Select physics processes
- G4VUserActionInitialization

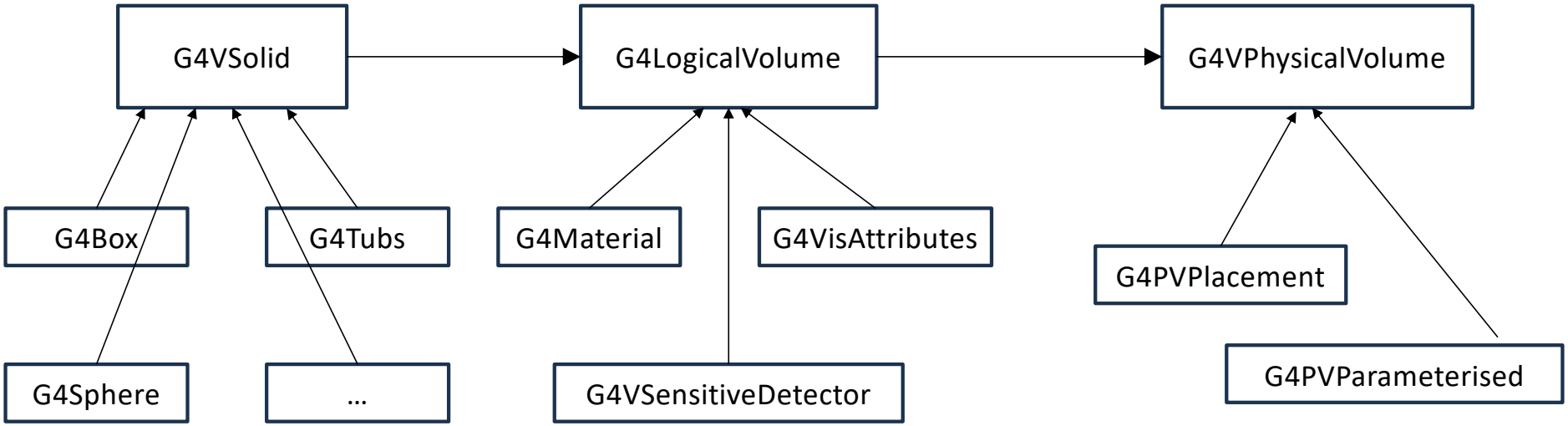
Describe your Detector (Geometry)

- Derive your own class from G4VUserDetectorConstruction abstract base class.
- In the virtual method Construct()
  - Construct all necessary materials
  - Construct volumes of your geometry
  - Construct your sensitive detector classes and set them to the detector volumes
- Optionally you can define
  - Regions for any part of your detector
  - Visualization attributes of your detector elements

Similar to 3D Drawing

Create a Volume

- Three conceptual layers
  - **G4VSolid** -- *shape, size*
  - **G4LogicalVolume** -- *daughter physical volumes, material, user limits, etc.*
  - **G4VPhysicalVolume** -- *position, rotation*



Make it a real detector

Example B1

- All simulation should have a world volume.

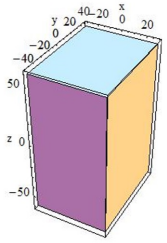
```
G4double world_sizeXY = 1.2*env_sizeXY;
G4double world_sizeZ  = 1.2*env_sizeZ;
G4Material* world_mat = nist->FindOrBuildMaterial("G4_AIR");

auto solidWorld = new G4Box("World",                           // its name
  0.5 * world_sizeXY, 0.5 * world_sizeXY, 0.5 * world_sizeZ);  // its size

auto logicWorld = new G4LogicalVolume(solidWorld,  // its solid
  world_mat,                                       // its material
  "World");                                        // its name

auto physWorld = new G4PVPlacement(nullptr,  // no rotation
  G4ThreeVector(),                           // at (0,0,0)
  logicWorld,                                // its logical volume
  "World",                                   // its name
  nullptr,                                   // its mother  volume
  false,                                     // no boolean operation
  0,                                         // copy number
  checkOverlaps);                            // overlaps checking
```

G4VSolid

G4LogicalVolume

G4VPhysicalVolume

```
G4Box(const G4String& pName,
            G4double  pX,
            G4double  pY,
            G4double  pZ)
```

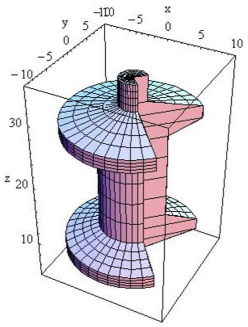


In the picture:
`pX = 30, pY = 40, pZ = 60`

by giving the box a name and its half-lengths along the X, Y and Z axis:

`pX`	half length in X	`pY`	half length in Y	`pZ`	half length in Z

```
G4Polycone(const G4String& pName,
                 G4double  phiStart,
                 G4double  phiTotal,
                 G4int     numZPlanes,
           const G4double  zPlane[],
           const G4double  rInner[],
           const G4double  rOuter[])
```

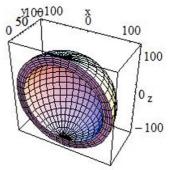


In the picture:
`phiStart = 1/4*Pi, phiTotal = 3/2*Pi, numZPlanes = 9, rInner = { 0, 0, 0, 0, 0, 0, 0, 0, 0}, rOuter = { 0, 10, 10, 5 , 5, 10 , 10 , 2, 2}, z = { 5, 7, 9, 11, 25, 27, 29, 31, 35 }`

where:

`phiStart`	Initial Phi starting angle
`phiTotal`	Total Phi angle
`numZPlanes`	Number of Z planes
`numRZ`	Number of corners in r,Z space
`zPlane`	Position of Z planes, with Z in increasing order
`rInner`	Tangent distance to inner surface
`rOuter`	Tangent distance to outer surface
`r`	r coordinate of corners
`z`	Z coordinate of corners

```
G4Sphere(const G4String& pName,
               G4double  pRmin,
               G4double  pRmax,
               G4double  pSPhi,
               G4double  pDPhi,
               G4double  pSTheta,
               G4double  pDTheta )
```

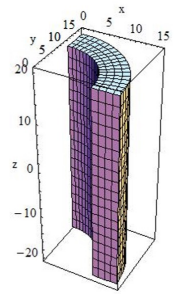


In the picture:
`pRmin = 100, pRmax = 120, pSPhi = 0*Degree, pDPhi = 180*Degree, pSTheta = 0 Degree, pDTheta = 180*Degree`

to obtain a solid with name `pName` and parameters:

`pRmin`	Inner radius
`pRmax`	Outer radius
`pSPhi`	Starting Phi angle of the segment in radians
`pDPhi`	Delta Phi angle of the segment in radians
`pSTheta`	Starting Theta angle of the segment in radians
`pDTheta`	Delta Theta angle of the segment in radians

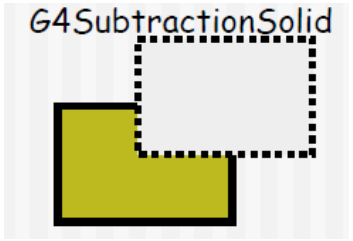
```
G4Tubs(const G4String& pName,
             G4double  pRMin,
             G4double  pRMax,
             G4double  pDz,
             G4double  pSPhi,
             G4double  pDPhi)
```

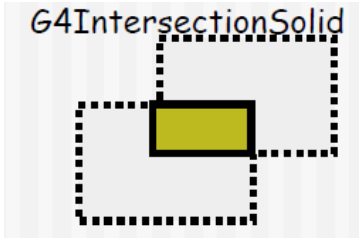


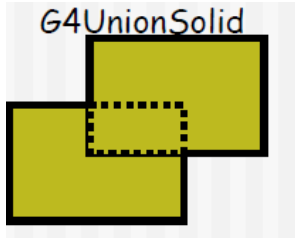
In the picture:
`pRMin = 10, pRMax = 15, pDz = 20`

giving its name `pName` and its parameters which are:

`pRMin`	Inner radius	`pRMax`	Outer radius
`pDz`	Half length in Z	`pSPhi`	Starting phi angle in radians
`pDPhi`	Angle of the segment in radians		







Find more in [Book For Application Developers]

Example B1 -- Envelope

```
G4Material* env_mat = nist->FindOrBuildMaterial("G4_WATER");
```

```
// Envelope
//
auto solidEnv = new G4Box("Envelope",                    // its name
  0.5 * env_sizeXY, 0.5 * env_sizeXY, 0.5 * env_sizeZ);  // its size

auto logicEnv = new G4LogicalVolume(solidEnv,  // its solid
  env_mat,                                     // its material
  "Envelope");                                 // its name

new G4PVPlacement(nullptr,  // no rotation
  G4ThreeVector(),          // at (0,0,0)
  logicEnv,                 // its logical volume
  "Envelope",               // its name
  logicWorld,               // its mother  volume
  false,                    // no boolean operation
  0,                        // copy number
  checkOverlaps);           // overlaps checking
```

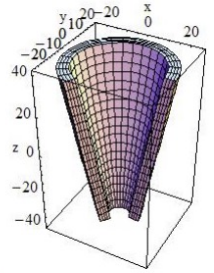
Example B1 – Shape 1

```
// Shape 1
//
G4Material* shape1_mat = nist->FindOrBuildMaterial("G4_A-150_TISSUE");
G4ThreeVector pos1 = G4ThreeVector(0, 2*cm, -7*cm);

// Conical section shape
G4double shape1_rmina =  0.*cm, shape1_rmaxa = 2.*cm;
G4double shape1_rminb =  0.*cm, shape1_rmaxb = 4.*cm;
G4double shape1_hz = 3.*cm;
G4double shape1_phimin = 0.*deg, shape1_phimax = 360.*deg;
auto solidShape1 = new G4Cons("Shape1", shape1_rmina, shape1_rmaxa,
  shape1_rminb, shape1_rmaxb,
  shape1_hz, shape1_phimin, shape1_phimax);

auto logicShape1 = new G4LogicalVolume(solidShape1,  // its solid
  shape1_mat,                                        // its material
  "Shape1");                                         // its name

new G4PVPlacement(nullptr,  // no rotation
  pos1,                     // at position
  logicShape1,              // its logical volume
  "Shape1",                 // its name
  logicEnv,                 // its mother  volume
  false,                    // no boolean operation
  0,                        // copy number
  checkOverlaps);           // overlaps checking
```

Cone or Conical section:

Similarly to create a **cone**, or **conical section**, one would use the constructor

```
G4Cons(const G4String& pName,
             G4double  pRmin1,
             G4double  pRmax1,
             G4double  pRmin2,
             G4double  pRmax2,
             G4double  pDz,
             G4double  pSPhi,
             G4double  pDPhi)
```



In the picture:
`pRmin1 = 5, pRmax1 = 10, pRmin2 = 20, pRmax2 = 25, pDz = 40, pSPhi = 0, pDPhi = 4/3*Pi`

giving its name `pName`, and its parameters which are:

`pRmin1`	inside radius at `-pDz`	`pRmax1`	outside radius at `-pDz`
`pRmin2`	inside radius at `+pDz`	`pRmax2`	outside radius at `+pDz`
`pDz`	half length in Z	`pSPhi`	starting angle of the segment in radians
`pDPhi`	the angle of the segment in radians		

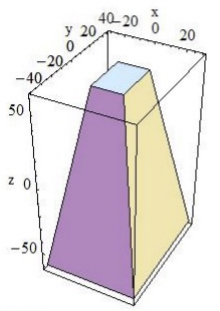
Example B1 – Shape 2

```
// Shape 2
//
G4Material* shape2_mat = nist->FindOrBuildMaterial("G4_BONE_COMPACT_ICRU");
G4ThreeVector pos2 = G4ThreeVector(0, -1*cm, 7*cm);

// Trapezoid shape
G4double shape2_dxa = 12*cm, shape2_dxb = 12*cm;
G4double shape2_dya = 10*cm, shape2_dyb = 16*cm;
G4double shape2_dz  = 6*cm;
auto solidShape2 = new G4Trd("Shape2",  // its name
  0.5 * shape2_dxa, 0.5 * shape2_dxb, 0.5 * shape2_dya, 0.5 * shape2_dyb,
  0.5 * shape2_dz);  // its size

auto logicShape2 = new G4LogicalVolume(solidShape2,  // its solid
  shape2_mat,                                        // its material
  "Shape2");                                         // its name

new G4PVPlacement(nullptr,  // no rotation
  pos2,                     // at position
  logicShape2,              // its logical volume
  "Shape2",                 // its name
  logicEnv,                 // its mother  volume
  false,                    // no boolean operation
  0,                        // copy number
  checkOverlaps);           // overlaps checking

// Set Shape2 as scoring volume
//
fScoringVolume = logicShape2;
```

Trapezoid:

To construct a **trapezoid** use:

```
G4Trd(const G4String& pName,
            G4double  dx1,
            G4double  dx2,
            G4double  dy1,
            G4double  dy2,
            G4double  dz)
```



In the picture:
`dx1 = 30, dx2 = 10, dy1 = 40, dy2 = 15, dz = 60`

to obtain a solid with name `pName` and parameters

`dx1`	Half-length along X at the surface positioned at `-dz`
`dx2`	Half-length along X at the surface positioned at `+dz`
`dy1`	Half-length along Y at the surface positioned at `-dz`
`dy2`	Half-length along Y at the surface positioned at `+dz`
`dz`	Half-length along Z axis

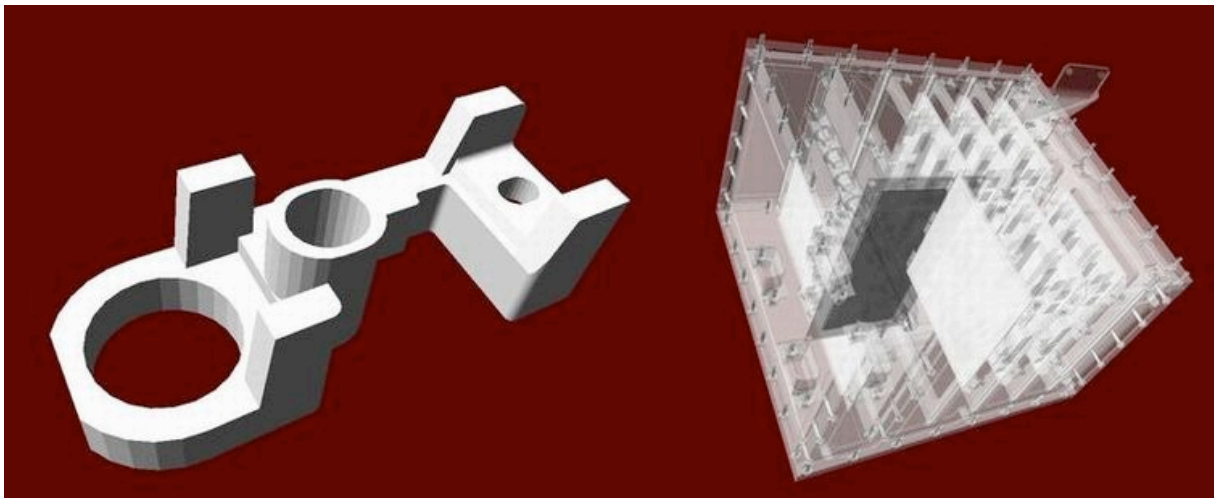
Example B1 – Shape 3

- Now let’s try to add a new shape to it.

Complex Volumes from STL

• Class G4TessellatedSolid can be used to generate a generic solid defined by a number of facets (`G4TriangularFacet`).

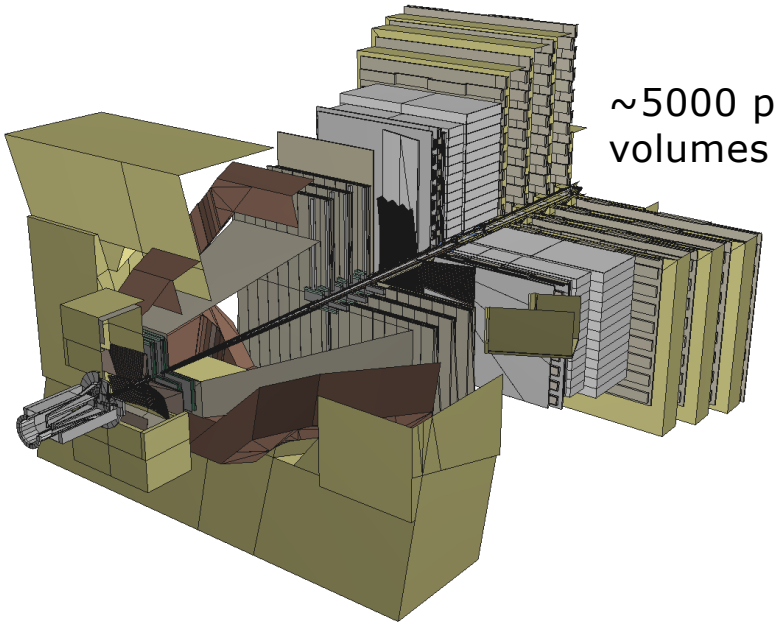
➢Defined by three vertices, which shall be supplied in ***anti-clockwise*** *order* looking from the outside of the solid where it belongs.



Find more in Book For Application Developers

Complex Volumes from GDML

- CAD geometries generated through STEP-Tools can be imported through the GDML reader:
  - parser.ParseST("stFile", WorldMaterial, GeomMaterial);
- Tool like FastRad allow for importing CAD STEP files and directly convert to GDML

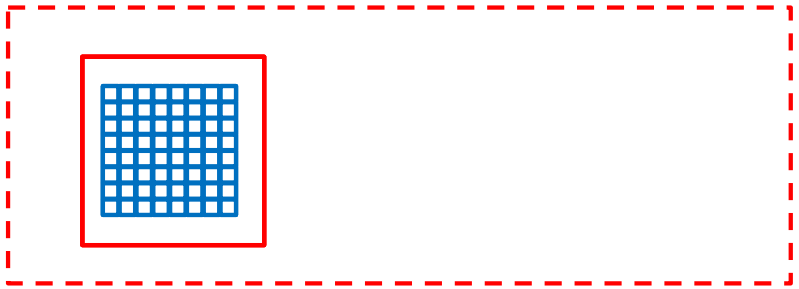


Geant4 LHCb geometry imported in Root through GDML

Find more in Book For Application Developers

Geometry Hierarchy

- World volume is the root volume of the hierarchy
- Mother and daughter volumes (similar to Assembly and Parts)
  - A volume is placed in its mother volume
  - Position and rotation of the daughter volume is described with respect to the local coordinate system of the mother volume
  - The origin of the mother's local coordinate system is at the center of the mother volume
  - Daughter volumes cannot protrude from the mother volume
  - Daughter volumes cannot overlap
  - One or more volumes can be placed in a mother volume
  - If the logical volume of the mother is placed more than once, all daughters appear by definition in all these physical instances of the mother



```
// Envelope
//
auto solidEnv = new G4Box("Envelope",                    // its name
  0.5 * env_sizeXY, 0.5 * env_sizeXY, 0.5 * env_sizeZ);  // its size

auto logicEnv = new G4LogicalVolume(solidEnv,  // its solid
  env_mat,                                     // its material
  "Envelope");                                 // its name

new G4PVPlacement(nullptr,  // no rotation
  G4ThreeVector(),          // at (0,0,0)
  logicEnv,                 // its logical volume
  "Envelope",               // its name
  logicWorld,               // its mother  volume
                            // no boolean operation
                            // copy number
                            // overlaps checking
```

```
// Shape 1
//
G4Material* shape1_mat = nist->FindOrBuildMaterial("G4_A-150_TISSUE");
G4ThreeVector pos1 = G4ThreeVector(0, 2*cm, -7*cm);

// Conical section shape
G4double shape1_rmina =  0.*cm, shape1_rmaxa = 2.*cm;
G4double shape1_rminb =  0.*cm, shape1_rmaxb = 4.*cm;
G4double shape1_hz = 3.*cm;
G4double shape1_phimin = 0.*deg, shape1_phimax = 360.*deg;
auto solidShape1 = new G4Cons("Shape1", shape1_rmina, shape1_rmaxa,
  shape1_rminb, shape1_rmaxb,
  shape1_hz, shape1_phimin, shape1_phimax);

auto logicShape1 = new G4LogicalVolume(solidShape1,  // its solid
  shape1_mat,                                        // its material
  "Shape1");                                         // its name

new G4PVPlacement(nullptr,  // no rotation
  pos1,                     // at position
  logicShape1,              // its logical volume
  "Shape1",                 // its name
  logicEnv,                 // its mother  volume
  false,                    // no boolean operation
  0,                        // copy number
  checkOverlaps);           // overlaps checking
```

```
// Shape 2
//
G4Material* shape2_mat = nist->FindOrBuildMaterial("G4_BONE_COMPACT_ICRU");
G4ThreeVector pos2 = G4ThreeVector(0, -1*cm, 7*cm);

// Trapezoid shape
G4double shape2_dxa = 12*cm, shape2_dxb = 12*cm;
G4double shape2_dya = 10*cm, shape2_dyb = 16*cm;
G4double shape2_dz  = 6*cm;
auto solidShape2 = new G4Trd("Shape2",  // its name
  0.5 * shape2_dxa, 0.5 * shape2_dxb, 0.5 * shape2_dya, 0.5 * shape2_dyb,
  0.5 * shape2_dz);  // its size

auto logicShape2 = new G4LogicalVolume(solidShape2,  // its solid
  shape2_mat,                                        // its material
  "Shape2");                                         // its name

new G4PVPlacement(nullptr,  // no rotation
  pos2,                     // at position
  logicShape2,              // its logical volume
  "Shape2",                 // its name
  logicEnv,                 // its mother  volume
  false,                    // no boolean operation
  0,                        // copy number
  checkOverlaps);           // overlaps checking
```

Example B1 – Place two more Envelopes

- G4VUserDetectorConstruction -- Describe your detector
- G4VUserPhysicsList -- Select physics processes
- G4VUserActionInitialization

Select Physics Processes

- Geant4 does not have any default particles or processes.
  - Even for the particle transportation, you have to define it explicitly.
- Derive your own concrete class from G4VUserPhysicsList abstract base class.
  - Define all necessary particles, processes, cut-off ranges
- **Primarily, the user's task is choosing a "pre-packaged" physics list**, that combines physics processes and models that are relevant to a typical application use-cases.
  - If "pre-packaged" physics lists do not meet your needs, you may add or alternate some processes/models.
  - If you are brave enough, you may implement your physics list.

Select Physics Processes

- **Some "standard" EM physics constructors:**
  - `G4EmStandardPhysics` - standard, for "higher energy physics"
  - `G4EmStandardPhysics_option1` - for HEP, fast but not precise settings
  - `G4EmStandardPhysics_option2` - for HEP, experimental
  - `G4EmStandardPhysics_option3` - for medical and space science applications
  - `G4EmStandardPhysics_option4` - most accurate EM models and settings
- **Some "low energy" EM physics constructors:**
  - `G4EmLivermorePhysics`
  - `G4EmLivermorePolarizedPhysics`
  - `G4EmPenelopePhysics`
  - `G4EmDNAPhysics`

Example B1

```
// Physics list
auto physicsList = new QBBC;
physicsList->SetVerboseLevel(1);
runManager->SetUserInitialization(physicsList);
```

QBBC

- Hadronic Component
- Electromagnetic Component
- Decay Component
- Neutron tracking cut
- Recommended Use Cases
- Related Physics Lists

https://geant4.web.cern.ch/documentation/pipelines/master/plg_html/PhysicsListGuide/reference_PL/QBBC.html#

- G4VUserDetectorConstruction -- Describe your detector
- G4VUserPhysicsList -- Select physics processes
- G4VUserActionInitialization

Mandatory User Action Class: Generate Primary Event

- Derive your concrete class from G4VUserPrimaryGeneratorAction abstract base class.
- Geant4 provides several generators:
  - G4GeneralParticleSource
    - Define radioactivity
  - G4ParticleGun
    - We can scratch a digital phantom with a random number generator

Mandatory User Action Class: Generate Primary Event

- Derive your concrete class from G4VUserPrimaryGeneratorAction abstract base class.
- Geant4 provides several generators:
  - G4GeneralParticleSource
    - Define radioactivity
  - G4ParticleGun
    - We can scratch a digital phantom with a random number generator

Example B1

Set Particle type, energy, direction

```
rimaryGeneratorAction::PrimaryGeneratorAction()

G4int n_particle = 1;
fParticleGun  = new G4ParticleGun(n_particle);

// default particle kinematic
G4ParticleTable* particleTable = G4ParticleTable::GetParticleTable();
G4String particleName;
G4ParticleDefinition* particle
  = particleTable->FindParticle(particleName="gamma");
fParticleGun->SetParticleDefinition(particle);
fParticleGun->SetParticleMomentumDirection(G4ThreeVector(0.,0.,1.));
fParticleGun->SetParticleEnergy(6.*MeV);
```

Get Geometry Information

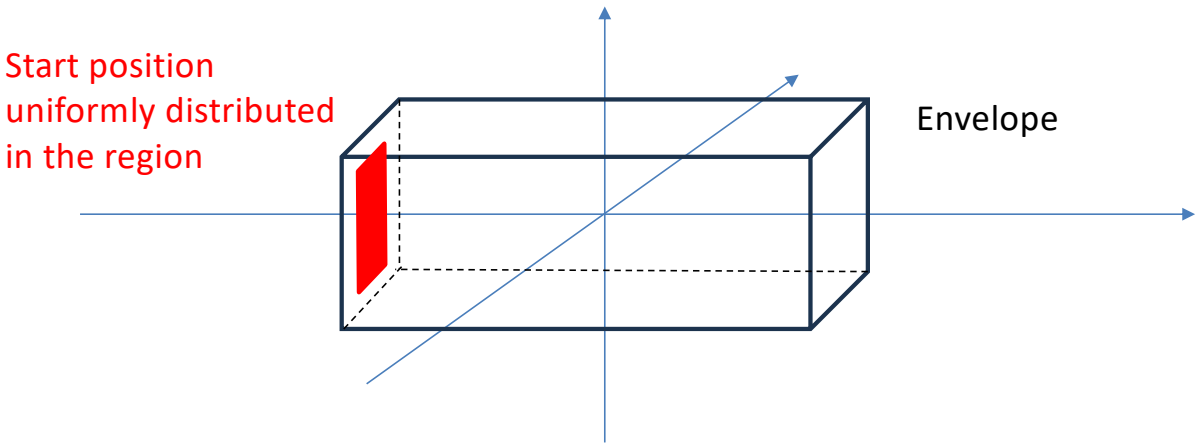
```
G4double envSizeXY = 0;
G4double envSizeZ = 0;

if (!fEnvelopeBox)
{
  G4LogicalVolume* envLV
    = G4LogicalVolumeStore::GetInstance()->GetVolume("Envelope");
  if ( envLV ) fEnvelopeBox = dynamic_cast<G4Box*>(envLV->GetSolid());
}

if ( fEnvelopeBox ) {
  envSizeXY = fEnvelopeBox->GetXHalfLength()*2.;
  envSizeZ = fEnvelopeBox->GetZHalfLength()*2.;
}
```

```
G4double size = 0.8;
G4double x0 = size * envSizeXY * (G4UniformRand()-0.5);
G4double y0 = size * envSizeXY * (G4UniformRand()-0.5);
G4double z0 = -0.5 * envSizeZ;

fParticleGun->SetParticlePosition(G4ThreeVector(x0,y0,z0));

fParticleGun->GeneratePrimaryVertex(anEvent);
```

Set particle position based on geometry

Example B1

```
G4double size = 0.8;
G4double x0 = size * envSizeXY * (G4UniformRand()-0.5);
G4double y0 = size * envSizeXY * (G4UniformRand()-0.5);
G4double z0 = -0.5 * envSizeZ;

fParticleGun->SetParticlePosition(G4ThreeVector(x0,y0,z0));

fParticleGun->GeneratePrimaryVertex(anEvent);
```



Optional User Action Classes

- G4UserRunAction
  - ➢void BeginOfRunAction(const G4Run*) --Define histograms
  - ➢void EndOfRunAction(const G4Run*) --Analyze the run, Store histograms
- G4UserEventAction
  - ➢void BeginOfEventAction(const G4Event*) -- Event selection
  - ➢void EndOfEventAction(const G4Event*) -- Output event information
- G4UserStackingAction
- G4UserTrackingAction
  - ➢void PreUserTrackingAction(const G4Track*)
    - ❑Decide trajectory should be stored or not
    - ❑Create user-defined trajectory
  - ➢void PostUserTrackingAction(const G4Track*) --Delete unnecessary trajectory
- G4UserSteppingAction
  - ➢void UserSteppingAction(const G4Step*) --Kill / suspend / postpone the track

Extract Useful Information

- Given geometry, physics and primary track generation, Geant4 does proper physics simulation “silently”.
  - ➢You have to do something to extract information useful to you.
- Three ways:
  - ➢Built-in scoring commands
    - ❑Most commonly-used physics quantities are available.
  - ➢Use scorers in the tracking volume
    - ❑Create scores for each event
    - ❑Create own Run class to accumulate scores
  - ➢Assign G4VSensitiveDetector to a volume to generate “hit”.
    - ❑ Use user hooks (G4UserEventAction, G4UserRunAction) to get event / run summary
- You may also use user hooks (G4UserTrackingAction, G4UserSteppingAction, etc.)
  - ➢ You have full access to almost all information