Montecarlo Methods for Medical Physics

Francesco Longo
(francesco.longo@ts.infn.it)

Summary of the Course

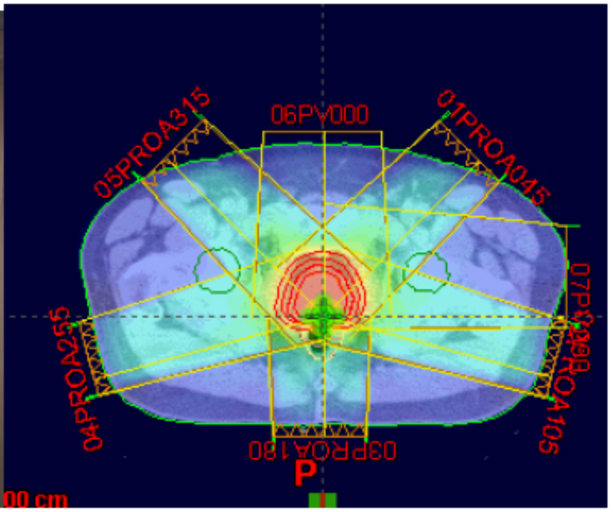
- Part1 (Monday)
  - General (and brief) introduction to Monte Carlo methods
  - Montecarlo methods in Medical Physics
- Part2 (Monday)
  - Introduction to the Geant4 toolkit
- Part3
  - Fundamentals of a Geant4 application (Today)
  - Geometry, Physics, Particle Flux, Scoring needs (Today - …)
- Laboratory (Next weeks)
  - Realisation of an example relevant to Medical Physics

Part 1

Montecarlo in Medical Physics

Montecarlo for Medical Physics

Overview of Medical Physics Applications

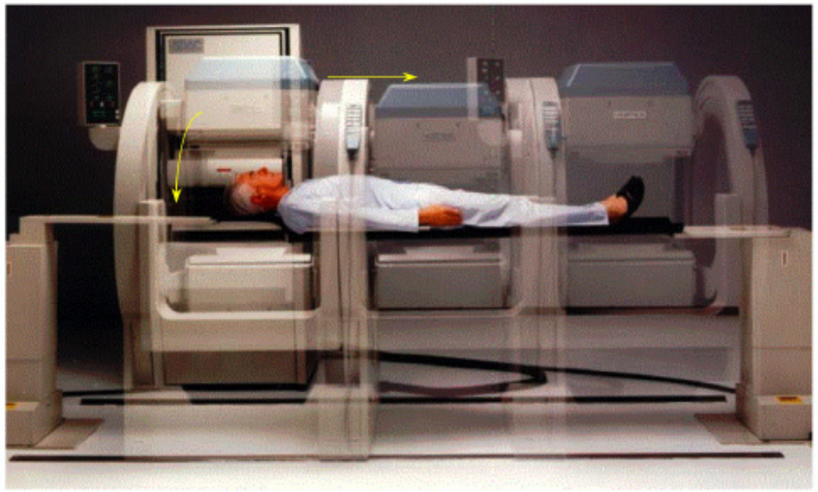


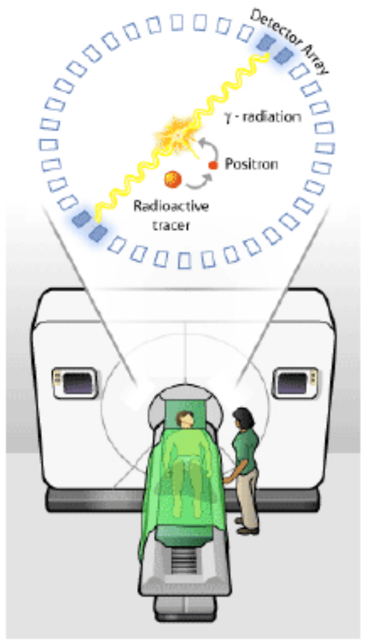
Radiotherapy physics

- external/internal sources and dosimetry
- phantom simulations
- treatment planning

Montecarlo for Medical Physics

Single Photon Emission Computer Tomography (SPECT)

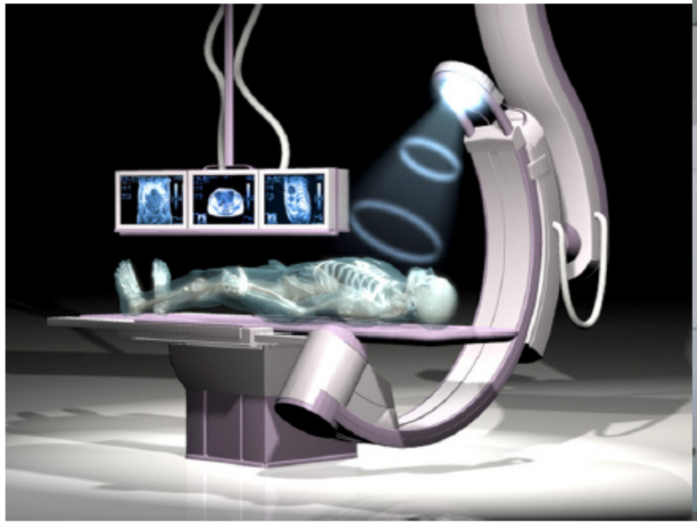




Nuclear medicine

- detectors
- imaging correction
- absorbed dose

Montecarlo for Medical Physics



Diagnostic radiology

- detection systems
- physical quantities
- radiation protection

Part 2

Introduction to Geant4

Technology transfer

Particle physics software aids space and medicine

Geant4 is a showcase example of technology transfer from particle physics to other fields such as space and medical science [. . .].

CERN Courier, June 2002

INTERNATIONAL JOURNAL OF HIGH-ENERGY PHYSICS

CERN COURIER

VOLUME 42 NUMBER 5 JUNE 2002

Simulation for physics, space and medicine

NEUTRINOS
Sudbury Neutrino Observatory confirms neutrino oscillation p5

TESLA
Electropolishing steers superconducting cavity to new record p10

COSMOPHYSICS
Joint symposium brings CERN, ESA and ESO together p15

Geant 4

Strategic vision

OO technology

- Openness to **extension** and **evolution**

new implementations can be added w/o changing the existing code

- Robustness and ease of **maintenance**

protocols and well defined dependencies minimize coupling

Toolkit

A set of compatible components

- each component is **specialised** for a specific functionality
- each component can be **refined** independently to a great detail
- components can be **integrated** at any degree of complexity
- it is easy to provide (and use) **alternative** components
- the user application can be **customised** as needed

Outline of Part2

- General Introduction to G4
  - What is G4 ?
  - Review of user documentation
  - Geant4 as a toolkit
- Basics of OO programming
- Geant4 Kernel and basics of the toolkit
  - Run, Event, Step
  - Particle and Physics processes
  - User classes

Simulation basics

The role of simulation

Simulation plays a fundamental role in various domains and phases of an experimental physics project

- *design of the experimental set-up*
- *evaluation and definition of the potential physics output of the project*
- *evaluation of potential risks to the project*
- *assessment of the performance of the experiment*
- *development, test and optimisation of reconstruction and physics analysis software*
- *contribution to the calculation and validation of physics results*

The scope of Geant4 encompasses the simulation of the passage of particles through matter

- *there are other kinds of simulation components, such as physics event generators, detector/electronics response generators, etc.*
- *often the simulation of a complex experiment consists of several of these components interfaced to one another*

Geant4 simulation toolkit

- Modeling the experimental set-up
- Tracking particles through matter
- Interaction of particles with matter
- Modeling the detector response
- Run and event control
- Accessory utilities *(random number generators, PDG particle information, physical constants, system of units etc.)*

- User interface
- Interface to event generators
- Visualisation *(of the set-up, tracks, hits etc.)*
- Persistency
- Analysis

Flexibility of Geant4

- In order to meet wide variety of requirements from various application fields, a large degree of functionality and flexibility are provided.
- Geant4 has many types of geometrical descriptions to describe most complicated and realistic geometries
  - CSG, BREP and Boolean solids
  - Placement, replica, divided, parameterized, reflected and grouped
  - XML interface
- Everything is open to the user
  - Choice of physics processes/models
  - Choice of GUI/Visualization/persistency/histogramming technologies

Physics in Geant4

- It is rather unrealistic to develop a uniform physics model to cover wide variety of particles and/or wide energy range.
- Much wider coverage of physics comes from mixture of theory-driven, parameterized, and empirical formulae. Thanks to polymorphism mechanism, both cross-sections and models (final state generation) can be combined in arbitrary manners into one particular process.
- Geant4 offers
  - EM processes
  - Hadronic processes
  - Photon/lepton-hadron processes
  - Optical photon processes
  - Decay processes
  - Shower parameterization
  - Event biasing techniques
  - And you can plug-in more

Physics in Geant4

- Each cross-section table or physics model (final state generation) has its own applicable energy range. Combining more than one tables / models, one physics process can have enough coverage of energy range for wide variety of simulation applications.
- Geant4 provides sets of alternative physics models so that the user can freely choose appropriate models according to the type of his/her application.
  - In other words, it is the user’s responsibility to choose reasonable set of physics processes/models that fits to his/her needs.
  - For example, some models are more accurate than others at a sacrifice of speed.

Part 3

Main ingredients of a G4 application

Basic concepts and kernel structure

The main program

To use Geant4, you have to…

- Geant4 is a toolkit. You have to build an application.
- To make an application, you have to
  - Define your geometrical setup
    - Material, volume
  - Define physics to get involved
    - Particles, physics processes/models
    - Production thresholds
  - Define how an event starts
    - Primary 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.

The main program

- Geant4 does not provide the *main*().
- In your *main(),* you have to
  - Construct G4RunManager (or your derived class)
  - Set user mandatory classes to RunManager
    - G4VUserDetectorConstruction
    - G4VUserPhysicsList
    - G4VUserPrimaryGeneratorAction
- You can define VisManager, (G)UI session, optional user action classes, and/or your persistency manager in your *main()*.

User classes

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

Note : classes written in yellow are mandatory.

Describe your detector

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

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
  - Define all necessary processes and assign them to proper particles
  - Define cut-off ranges applied to the world (and each region)
- Geant4 provides lots of utility classes/methods and examples.
  - "Educated guess" physics lists for defining hadronic processes for various use-cases.

Generate primary event

- Derive your concrete class from G4VUserPrimaryGeneratorAction abstract base class.
- Pass a G4Event object to one or more primary generator concrete class objects which generate primary vertices and primary particles.
- Geant4 provides several generators in addition to the G4VPrimaryParticlegenerator base class.
  - G4ParticleGun
  - G4HEPEvtInterface, G4HepMCInterface
  - G4GeneralParticleSource

User Classes needs

- Define material and geometry
  - ➔ G4VUserDetectorConstruction
- Select appropriate particles and processes and define production threshold(s)
  - ➔ G4VUserPhysicsList
- Define the way of primary particle generation
  - ➔ G4VUserPrimaryGeneratorAction
- Define the way to extract useful information from Geant4
  - ➔ G4UserSteppingAction, G4UserTrackingAction, etc.
  - ➔ G4VUserDetectorConstruction, G4UserEventAction, G4Run, G4UserRunAction
  - ➔ G4SensitiveDetector, G4VHit, G4VHitsCollection
  - ➔ G4PrimitiveScorers

Run in Geant4

- As an analogy of the real experiment, a run of Geant4 starts with "Beam On".
- Within a run, the user cannot change
  - detector setup
  - settings of physics processes
- Conceptually, a run is a collection of events which share the same detector and physics conditions.
  - A run consists of one event loop.
- At the beginning of a run, geometry is optimized for navigation and cross-section tables are calculated according to materials appear in the geometry and the cut-off values defined.
- G4RunManager class manages processing a run, a run is represented by G4Run class or a user-defined class derived from G4Run.
  - A run class may have a summary results of the run.
- G4UserRunAction is the optional user hook.

Event in Geant4

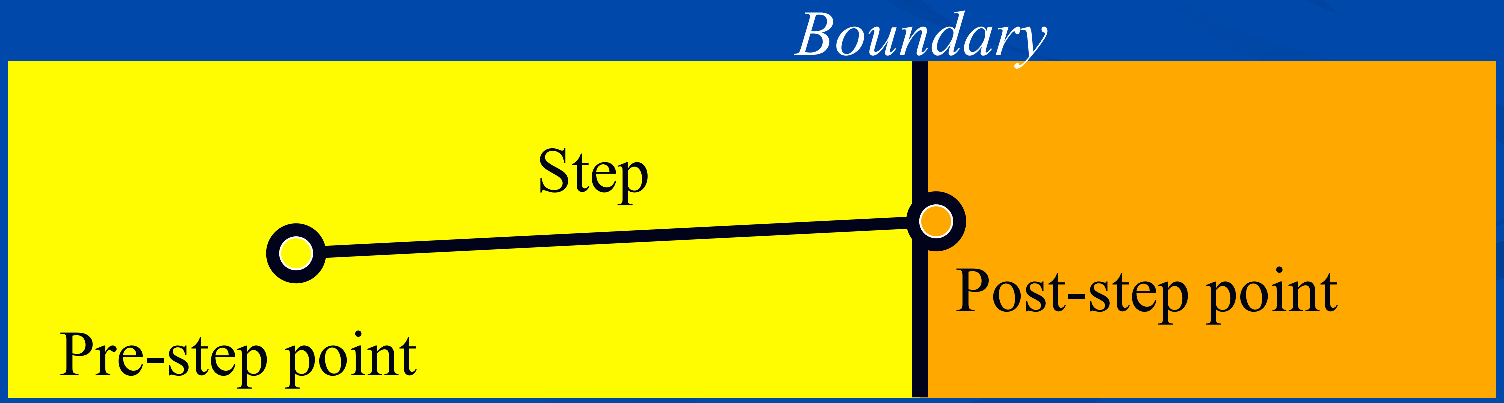
- An event is the basic unit of simulation in Geant4.
- At beginning of processing, primary tracks are generated. These primary tracks are pushed into a stack.
- A track is popped up from the stack one by one and "tracked". Resulting secondary tracks are pushed into the stack.
  - This "tracking" lasts as long as the stack has a track.
- When the stack becomes empty, processing of one event is over.
- G4Event class represents an event. It has following objects at the end of its (successful) processing.
  - List of primary vertices and particles (as input)
  - Hits and Trajectory collections (as output)
- G4EventManager class manages processing an event. G4UserEventAction is the optional user hook.

Track in Geant4

- Track is a snapshot of a particle.
  - It has physical quantities of current instance only. It does not record previous quantities.
  - Step is a "delta" information to a track. Track is not a collection of steps. Instead, a track is being updated by steps.
- Track object is deleted when
  - it goes out of the world volume,
  - it disappears (by e.g. decay, inelastic scattering),
  - it goes down to zero kinetic energy and no "AtRest" additional process is required, or
  - the user decides to kill it artificially.
- No track object persists at the end of event.
  - For the record of tracks, use trajectory class objects.
- G4TrackingManager manages processing a track, a track is represented by G4Track class.
- G4UserTrackingAction is the optional user hook.

Step in Geant4

- Step has two points and also "delta" information of a particle (energy loss on the step, time-of-flight spent by the step, etc.).
- Each point knows the volume (and material). In case a step is limited by a volume boundary, the end point physically stands on the boundary, and it logically belongs to the next volume.
  - Because one step knows materials of two volumes, boundary processes such as transition radiation or refraction could be simulated.
- G4SteppingManager class manages processing a step, a step is represented by G4Step class.
- G4UserSteppingAction is the optional user hook.



Particle in Geant4

- A particle in Geant4 is represented by three layers of classes.
- G4Track
  - Position, geometrical information, etc.
  - This is a class representing a particle to be tracked.
- G4DynamicParticle
  - "Dynamic" physical properties of a particle, such as momentum, energy, spin, etc.
  - Each G4Track object has its own and unique G4DynamicParticle object.
  - This is a class representing an individual particle.
- G4ParticleDefinition
  - "Static" properties of a particle, such as charge, mass, life time, decay channels, etc.
  - G4ProcessManager which describes processes involving to the particle
  - All G4DynamicParticle objects of same kind of particle share the same G4ParticleDefinition.

Extract useful information

- Given geometry, physics and primary track generation, Geant4 does proper physics simulation “silently”.
  - You have to add a bit of code to extract information useful to you.
- There are two ways:
  - Use user hooks (G4UserTrackingAction, G4UserSteppingAction, etc.)
    - You have an access to almost all information
    - Straight-forward, but do-it-yourself
  - Use Geant4 scoring functionality
    - Assign G4VSensitiveDetector to a volume
    - Hits collection is automatically stored in G4Event object, and automatically accumulated if user-defined Run object is used.
  - Use Geant4 native scorers to get specified quantities (dose, energy release, flux, path length, etc.)

User Interface in G4

Geant4 UI command

- A command consists of
  - Command directory
  - Command
  - Parameter(s)

```
/run/verbose 1
/vis/viewer/flush
```

- A parameter can be a type of string, boolean, integer or double.
  - Space is a delimiter.
  - Use double-quotes (“”) for string with space(s).
- A parameter may be “omittable”. If it is the case, a default value will be taken if you omit the parameter.
  - Default value is either predefined default value or current value according to its definition.
  - If you want to use the default value for your first parameter while you want to set your second parameter, use “!” as a place holder.

```
/dir/command ! second
```

Macro file

- Macro file is an ASCII file contains UI commands.
- All commands must be given with their full-path directories.
- Use "#" for comment line.
  - 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 file

```
/control/execute file_name
```

  - hard-coded

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

The main program

To use Geant4, you have to…

- Geant4 is a toolkit. You have to build an application.
- To make an application, you have to
  - Define your geometrical setup
    - Material, volume
  - Define physics to get involved
    - Particles, physics processes/models
    - Production thresholds
  - Define how an event starts
    - Primary 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.

Geant4 Basic Examples

Basic Examples

- The set of basic examples is oriented to "novice" users and covering many basic general use-cases typical of an "application"-oriented kind of development.
- **Example B1**
  - Simple geometry with a few solids
  - Geometry with simple placements (G4PVPlacement)
  - Scoring total dose in a selected volume user action classes
  - Geant4 physics list (QBBC)
- **Example B2**
  - Simplified tracker geometry with global constant magnetic field
  - Geometry with simple placements (G4PVPlacement) and parametrisation (G4PVParametrisation)
  - Scoring within tracker via G4 sensitive detector and hits
  - Geant4 physics list (FTFP_BERT) with step limiter
  - Started from novice/N02 example
- **Example B3**
  - Schematic Positron Emitted Tomography system
  - Geometry with simple placements with rotation (G4PVPlacement)
  - Radioactive source
  - Scoring within Crystals via G4 scorers
  - Modular physics list built via builders provided in Geant4

Basic Examples

- The set of basic examples is oriented to "novice" users and covering many basic general use-cases typical of an "application"-oriented kind of development.
- **Example B4**
  - Simplified calorimeter with layers of two materials
  - Geometry with replica (G4PVReplica)
  - Scoring within layers in four ways: via user actions (a), via user own object (b), via G4 sensitive detector and hits (c) and via scorers (d)
  - Geant4 physics list (FTFP_BERT)
  - Histograms (1D) and ntuple saved in the output file
  - Started from novice/N03 example
- **Example B5**
  - A double-arm spectrometer with wire chambers, hodoscopes and calorimeters with a local constant magnetic field
  - Geometry with placements with rotation, replicas and parameterisation
  - Scoring within wire chambers, hodoscopes and calorimeters via G4 sensitive detector and hits
  - Geant4 physics list (FTFP_BERT) with step limiter
  - UI commans defined using G4GenericMessenger
  - Histograms (1D) and ntuple saved in the output file
  - Started from extended/analysis/A01

Geant4 Extended Examples

Extended Examples

- The set of "extended" examples is covering various use-cases and may require some additional libraries besides of Geant4.
- analysis
  - Histogramming through the AIDA interface
- biasing
  - Examples of event biasing, scoring and reverse-MC-
- common
  - A set of common classes which can be reused in other examples demonstrating just a particular feature
- electromagnetic
  - Specific EM physics simulation with histogramming
- errorpropagation
  - Use of the error propagation utility (Geant4e)
- eventgenerator
  - Applications demonstrating various ways of primary event generation: using Geant4 particle gun, Geant4 general particle source, using interface to HepMC, Pythia
- exoticphysics
  - Exotic simulation applications (classical magnetic monopole, etc...)
- field
  - Specific simulation setups in magnetic field

Extended Examples

- g3tog3
  - Examples of usage of the g3tog4 converter tool
- geometry
  - Specific geometry examples and tools, OLAP tool for detection of overlapping geometries,
- hadronic
  - Specific hadronic physics simulation with histogramming
- medical
  - Specific examples for medical physics applications
- optical
  - Examples of generic optical processes simulation setups
- parallel
  - Examples of event-level parallelism in Geant4 using the TOP-C distribution, and MPI technique
- parameterisations
  - Examples for fast shower parameterisations according to specific models (gflash)

Extended Examples

- persistency
  - Persistency of geometry (GDML or ASCII) and simulation output
- polarisation
  - Use of physics processes including polarization
- radioactivedecay
  - Examples to simulate the decays of radioactive isotopes and induced radioactivity resulted from nuclear interactions
- runAndEvent
  - Examples to demonstrate how to connect the information between primary particles and hits and utilize user-information classes
- visualization
  - Specific visualization features and graphical customisations

Geant4 Advanced Examples

Advanced Examples

- air_shower
- ams_Ecal
- Brachytherapy
- ChargeExchangeMC
- composite_calorimeter
- eRosita
- Gammaknife
- gammaray_telescope
- hadrontherapy
- human_phantom
- iort_therapy
- lAr_calorimeter
- medical_linac
- microbeam
- microelectronics
- nanobeam
- purging_magnet
- radioprotection
- underground_physics
- xray_fluorescence
- xray_telescope

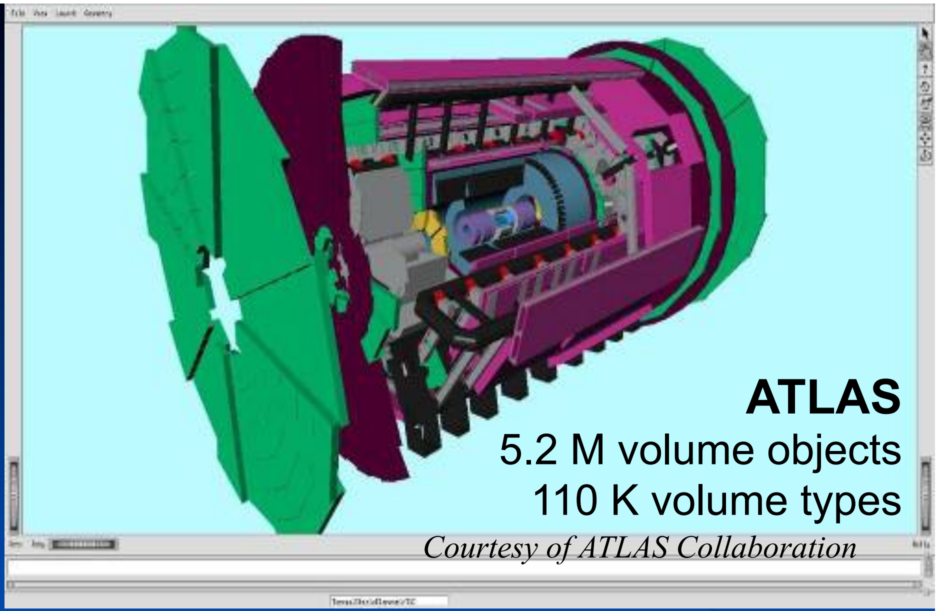
First Homework

- Review G4 web pages
- Find Appropriate documentation
- Find relevant Medical Physics examples
- Define your preferred project
  - Simple geometry
  - Particle distributions
  - Scoring needs

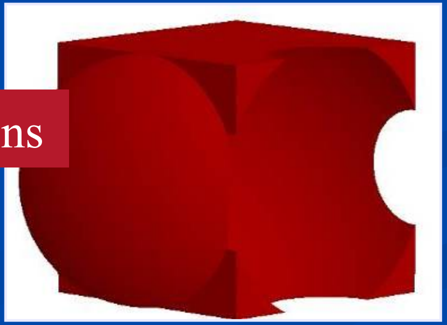
Geometry

Geometry

- Role
  - detailed detector description
  - efficient navigation



- Three conceptual layers
  - Solid: shape, size
  - LogicalVolume: material, sensitivity, daughter volumes, etc.
  - PhysicalVolume: position, rotation
- One can do fancy things with geometry…

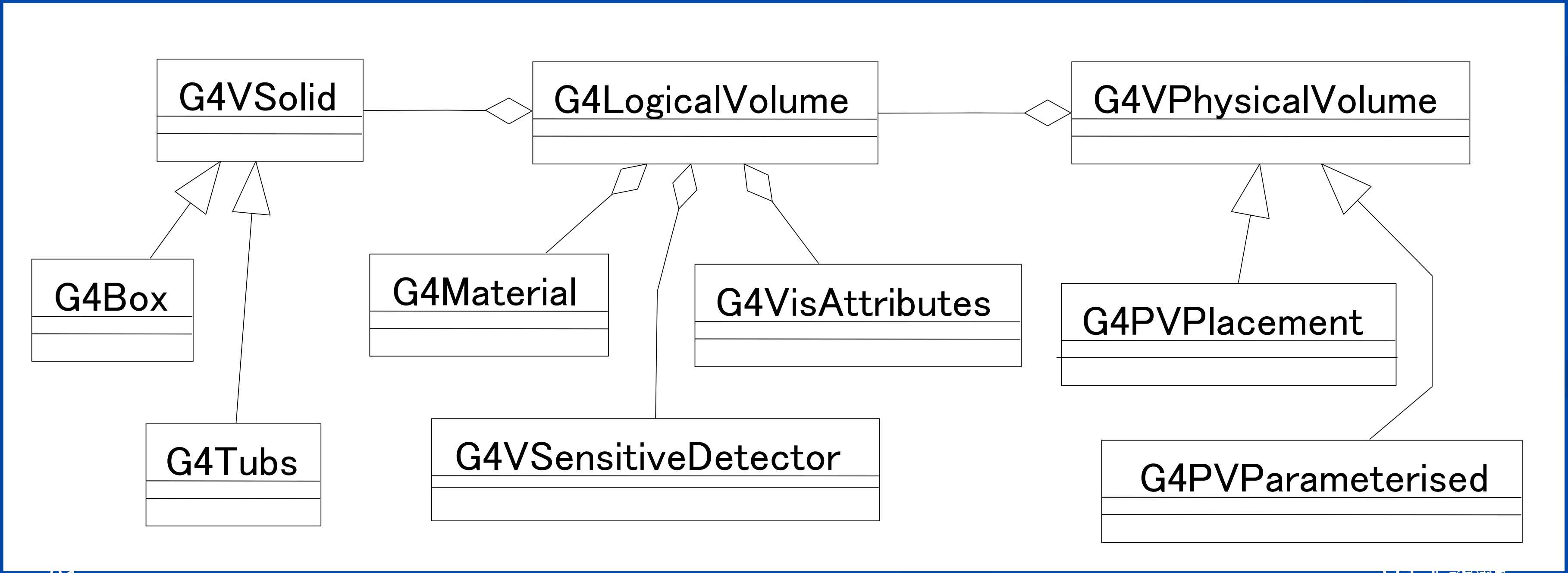


Boolean operations

Geometry - Volumes

Detector geometry

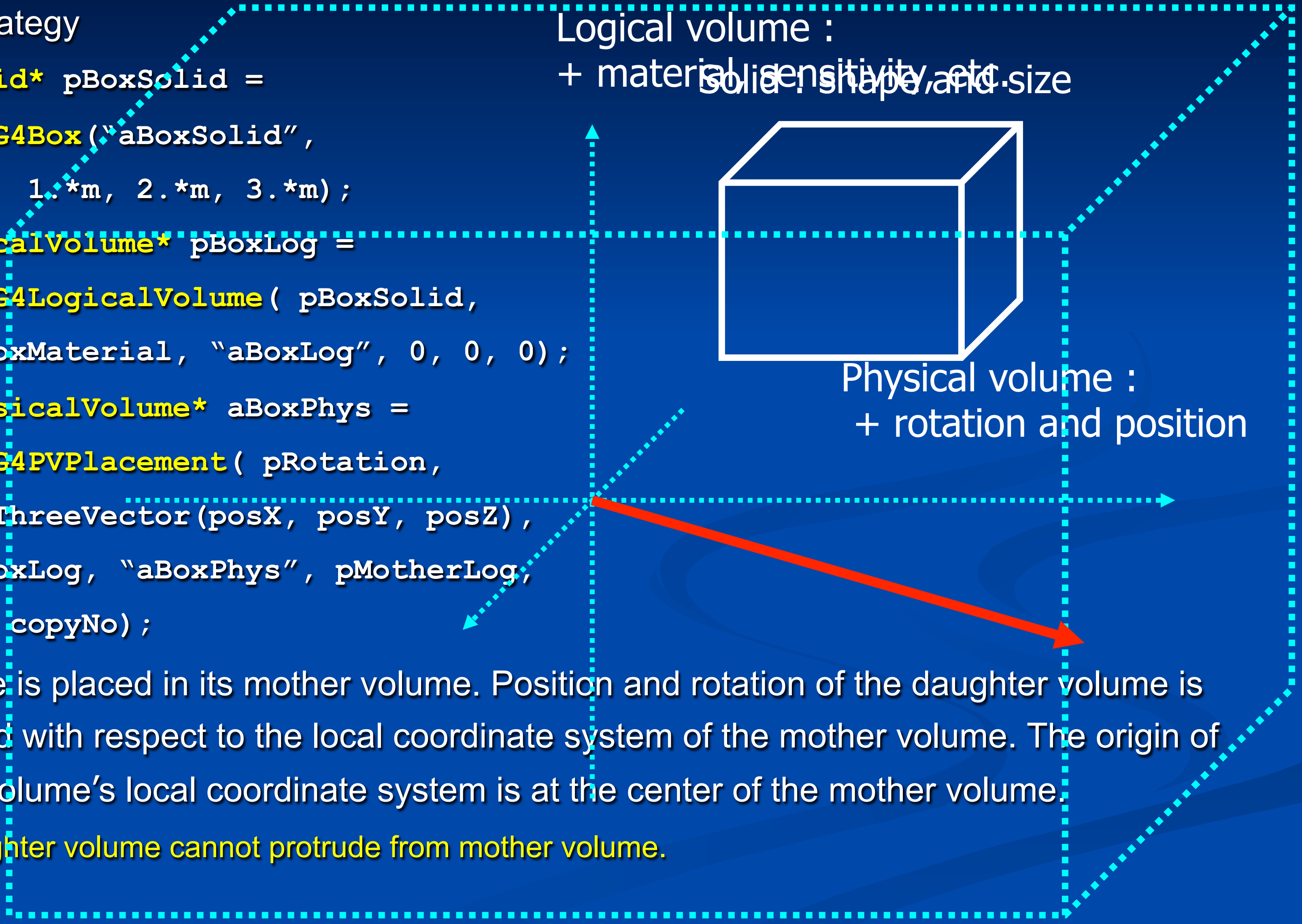
- Three conceptual layers
  - G4VSolid -- *shape, size*
  - G4LogicalVolume -- *daughter physical volumes,*
    material, sensitivity, user limits, etc.
  - G4VPhysicalVolume -- *position, rotation*



Define detector geometry

- Basic strategy

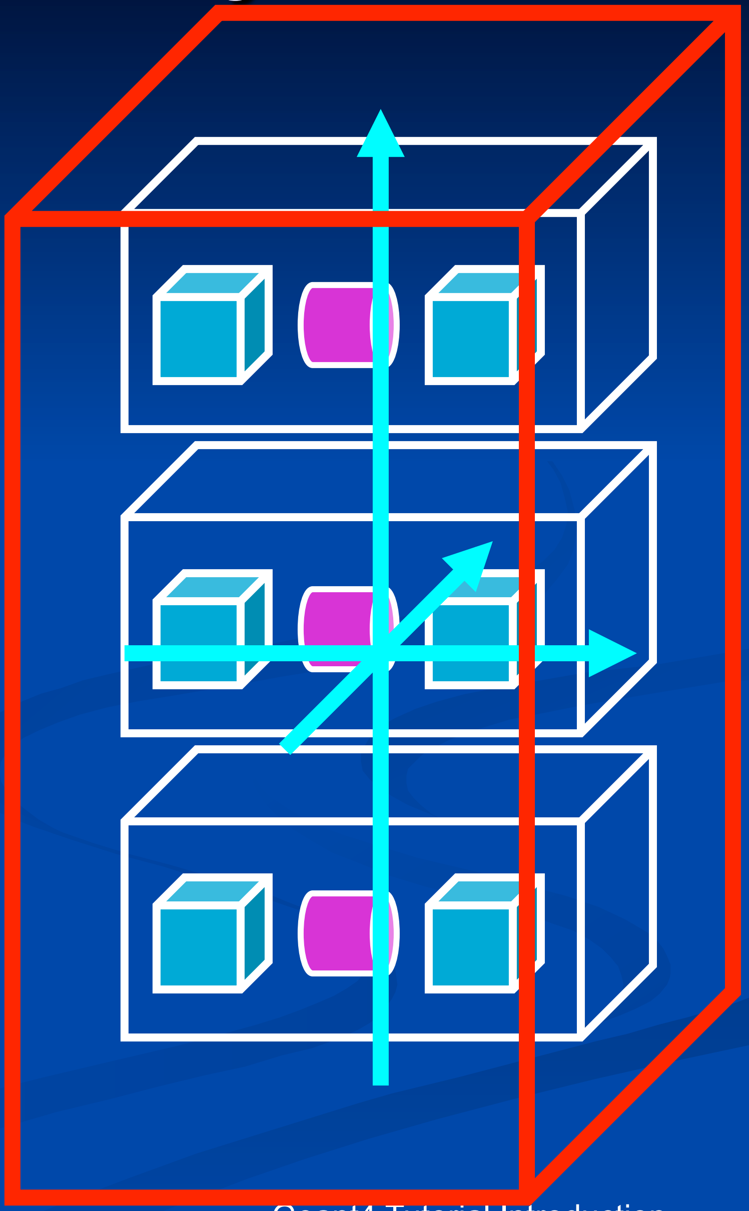
```
G4VSolid* pBoxSolid =
  new G4Box("aBoxSolid",
        1.*m, 2.*m, 3.*m);
G4LogicalVolume* pBoxLog =
  new G4LogicalVolume( pBoxSolid,
    pBoxMaterial, "aBoxLog", 0, 0, 0);
G4VPhysicalVolume* aBoxPhys =
  new G4PVPlacement( pRotation,
    G4ThreeVector(posX, posY, posZ),
    pBoxLog, "aBoxPhys", pMotherLog,
    0, copyNo);
```



- 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 mother volume's local coordinate system is at the center of the mother volume.
  - Daughter volume cannot protrude from mother volume.

Geometrical hierarchy

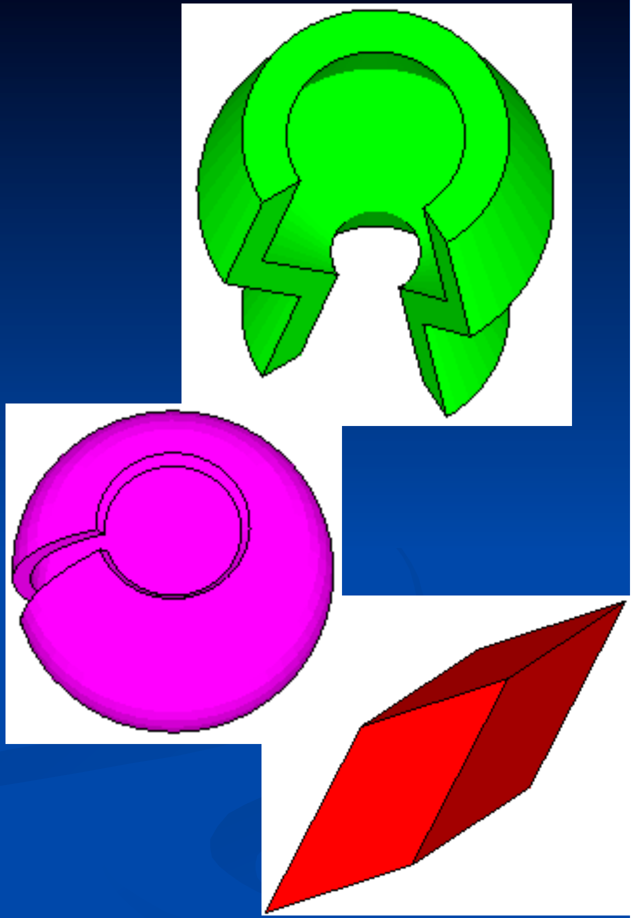
- One logical volume can be placed more than once. One or more volumes can be placed to a mother volume.
- Note that the mother-daughter relationship is an information of G4LogicalVolume.
  - If the mother volume is placed more than once, all daughters are by definition appear in all of mother physical volumes.
- The world volume must be a unique physical volume which fully contains all the other volumes.
  - The world volume defines the global coordinate system. The origin of the global coordinate system is at the center of the world volume.
  - Position of a track is given with respect to the global coordinate system.



Solid and shape

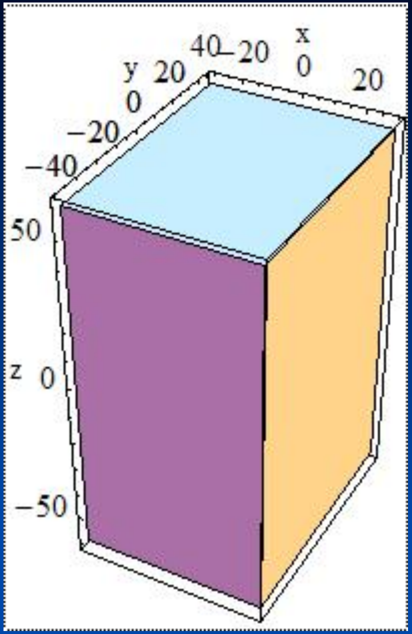
Solids

- Solids defined in Geant4:
  - CSG (Constructed Solid Geometry) solids
    - G4Box, G4Tubs, G4Cons, G4Trd, ...
    - Analogous to simple GEANT3 CSG solids
  - Specific solids (CSG like)
    - G4Polycone, G4Polyhedra, G4Hype, ...
  - Boolean solids
    - G4UnionSolid, G4SubtractionSolid, ...

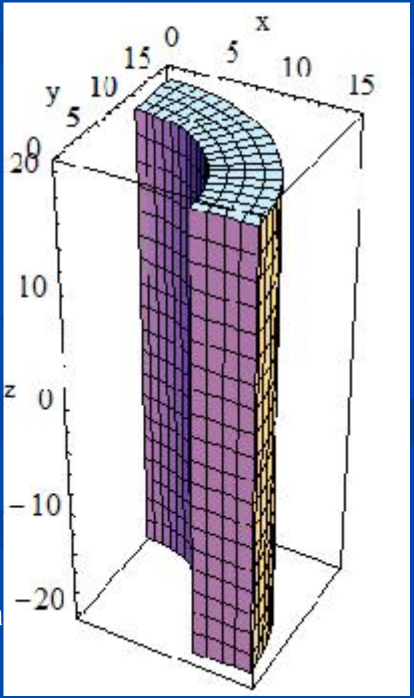


CSG: G4Box, G4Tubs

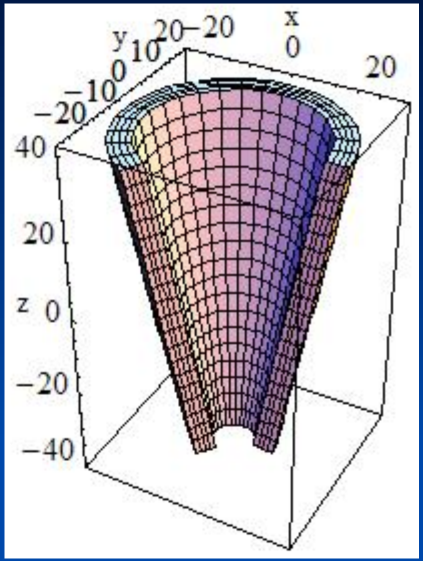
```
G4Box(const G4String &pname,    // name
            G4double half_x,    // X half size
            G4double half_y,    // Y half size
            G4double half_z);   // Z half size
```



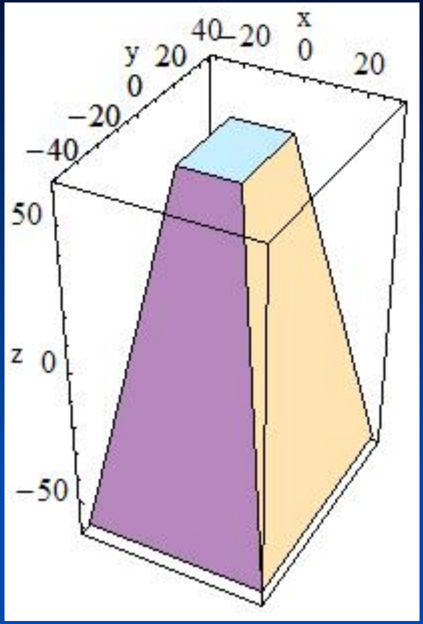
```
G4Tubs(const G4String &pname,   // name
             G4double  pRmin,   // inner radius
             G4double  pRmax,   // outer radius
             G4double  pDz,     // Z half length
             G4double  pSphi,   // starting Phi
             G4double  pDphi); // segment angle
```



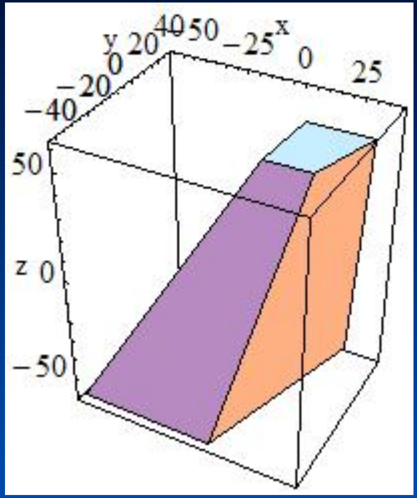
Other CSG solids



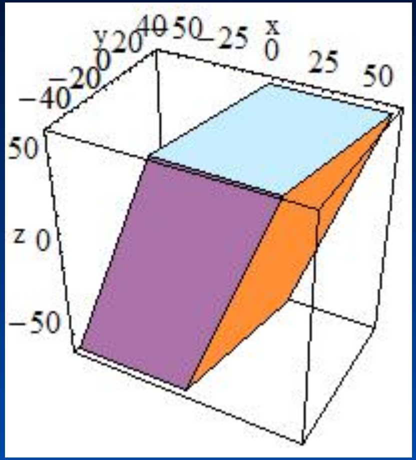
G4Cons



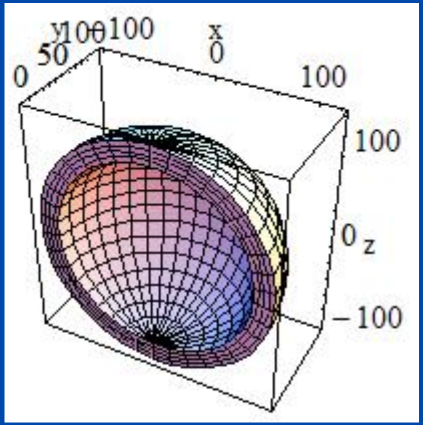
G4Trd



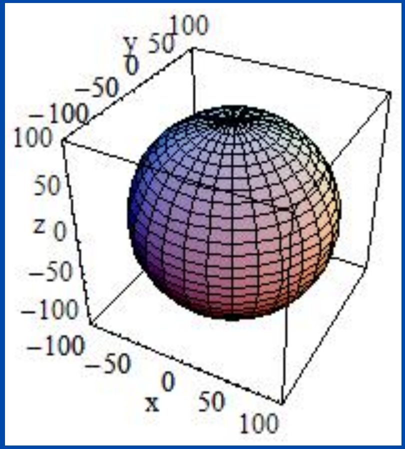
G4Trap



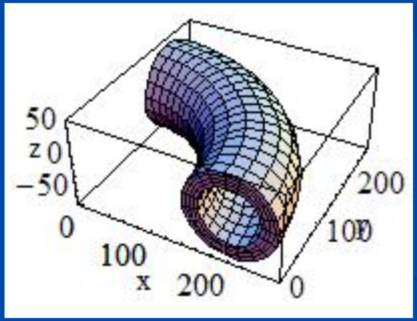
G4Para (parallelepiped)



G4Sphere



G4Orb (full solid sphere)

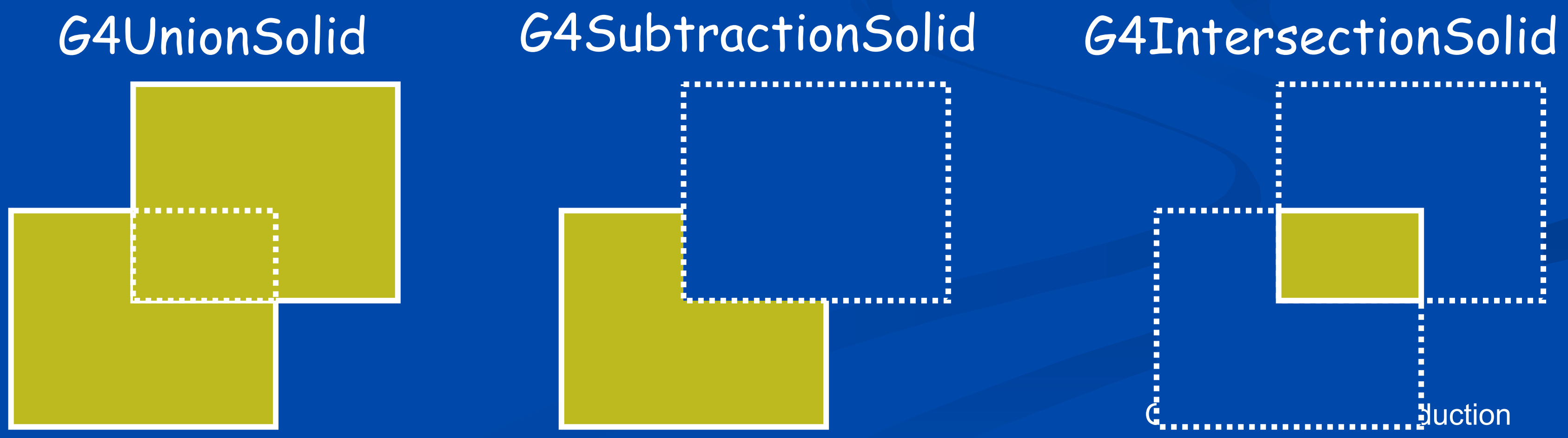


G4Torus

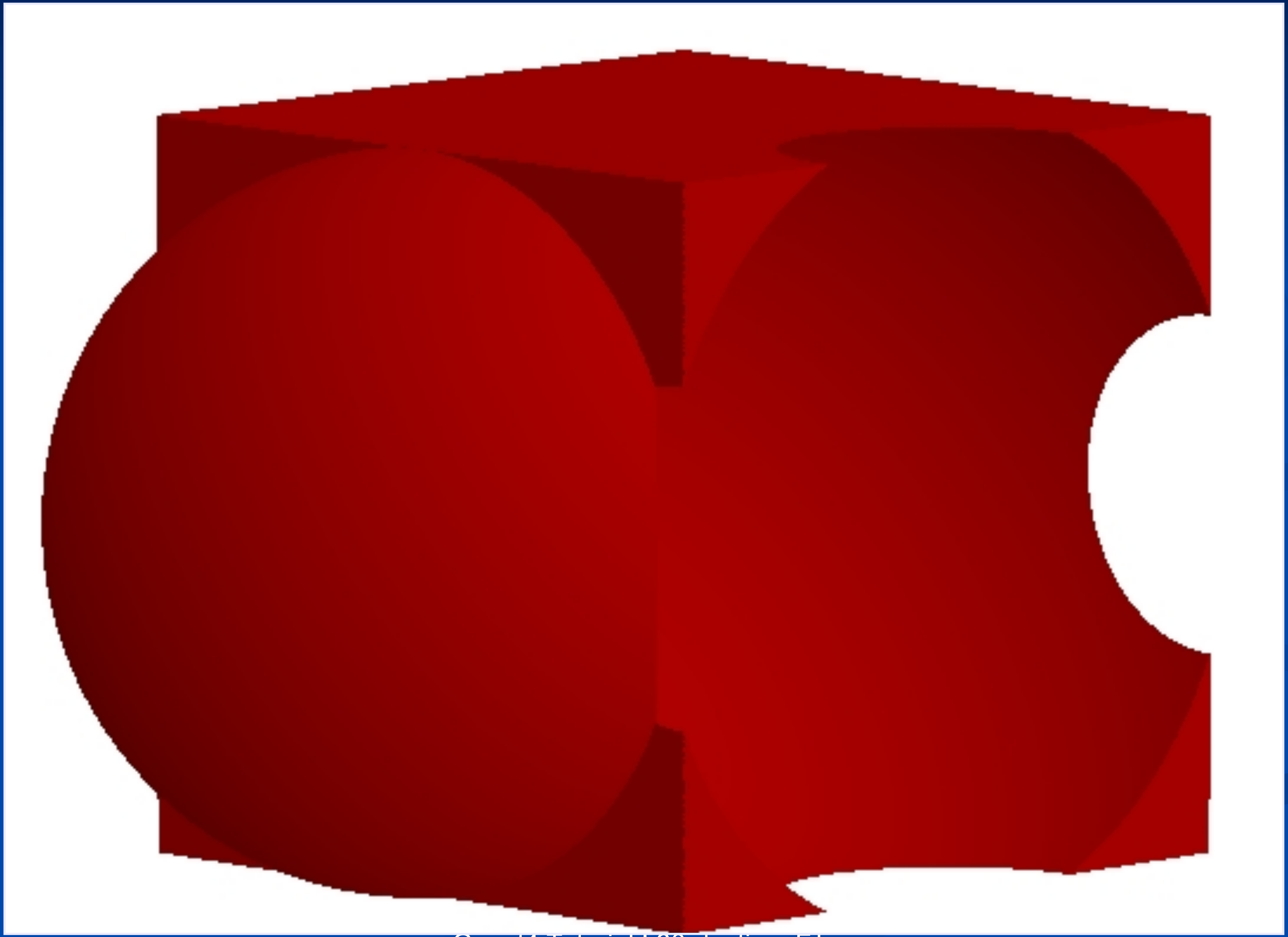
Consult to <u>Section 4.1.2 of Geant4 Application Developers Guide</u> for all available shapes.

Boolean Solids

- Solids can be combined using boolean operations:
  - `G4UnionSolid, G4SubtractionSolid, G4IntersectionSolid`
  - Requires: 2 solids, 1 boolean operation, and an (optional) transformation for the 2nd solid
  - 2nd solid is positioned relative to the coordinate system of the 1st solid
  - Result of boolean operation becomes a solid. Thus the third solid can be combined to the resulting solid of first operation.
- Solids to be combined can be either CSG or other Boolean solids.
- Note: tracking cost for the navigation in a complex Boolean solid is proportional to the number of constituent CSG solids



Boolean solid



G4LogicalVolume

G4LogicalVolume

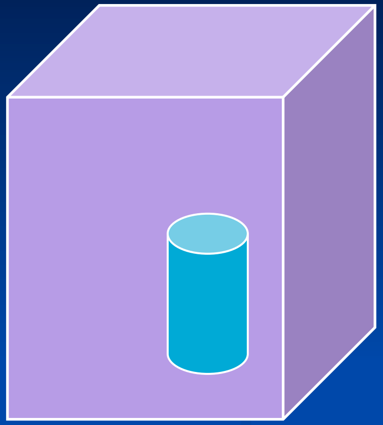
```
G4LogicalVolume(G4VSolid *pSolid,
                G4Material *pMaterial,
                const G4String &name,
                G4FieldManager *pFieldMgr=0,
                G4VSensitiveDetector *pSDetector=0,
                G4UserLimits *pULimits=0);
```

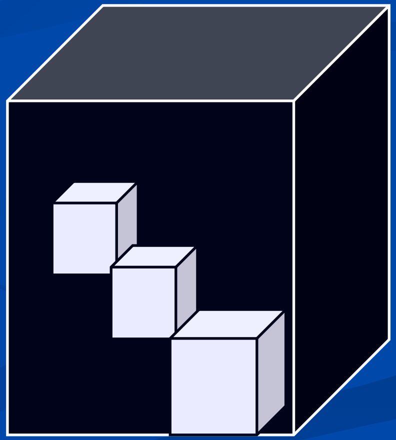
- Contains all information of volume except position and rotation
  - Shape and dimension (G4VSolid)
  - Material, sensitivity, visualization attributes
  - Position of daughter volumes
  - Magnetic field, User limits, Region
- Physical volumes of same type can share the common logical volume object.
- The pointers to solid must NOT be null.
- The pointers to material must NOT be null for tracking geometry.

Physical volume

Physical Volumes

- Placement volume : it is one positioned volume
  - One physical volume object represents one “real” volume.
- Repeated volume : a volume placed many times
  - One physical volume object represents any number of “real” volumes.
  - reduces use of memory.
  - Parameterised
    - repetition w.r.t. copy number
  - Replica and Division
    - simple repetition along one axis


placement


repeated

Physical volume

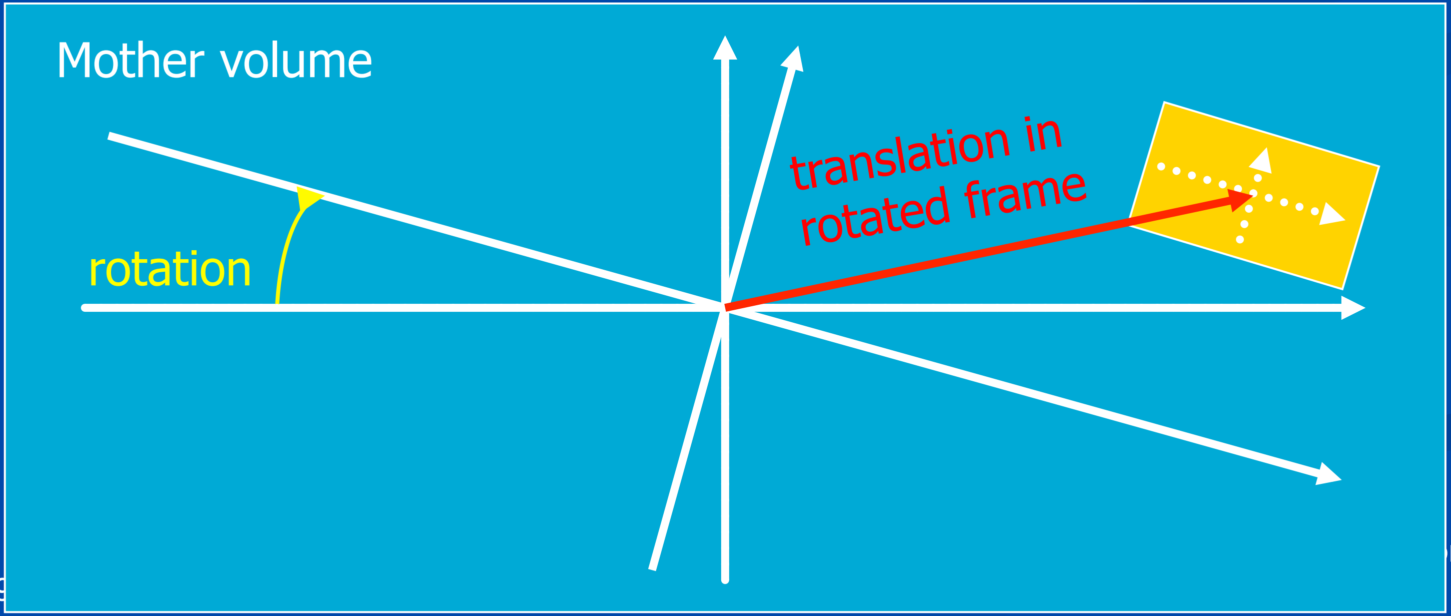
- G4PVPlacement 1 Placement = One Placement Volume
  - A volume instance positioned once in its mother volume
- G4PVParameterised 1 Parameterized = Many Repeated Volumes
  - Parameterized by the copy number
    - Shape, size, material, sensitivity, vis attributes, position and rotation can be parameterized by the copy number.
    - You have to implement a concrete class of G4VPVParameterisation.
  - Reduction of memory consumption
- G4PVReplica 1 Replica = Many Repeated Volumes
  - Daughters of same shape are aligned along one axis
  - Daughters fill the mother completely without gap in between.

G4PVPlacement

G4PVPlacement

```
G4PVPlacement(G4RotationMatrix* pRot,   // rotation of mother frame
             const G4ThreeVector &tlate, // position in rotated frame
             G4LogicalVolume *pDaughterLogical,
             const G4String &pName,
             G4LogicalVolume *pMotherLogical,
             G4bool pMany, // 'true' is not supported yet…
             G4int pCopyNo, // unique arbitrary integer
             G4bool pSurfChk=false); // optional boundary check
```

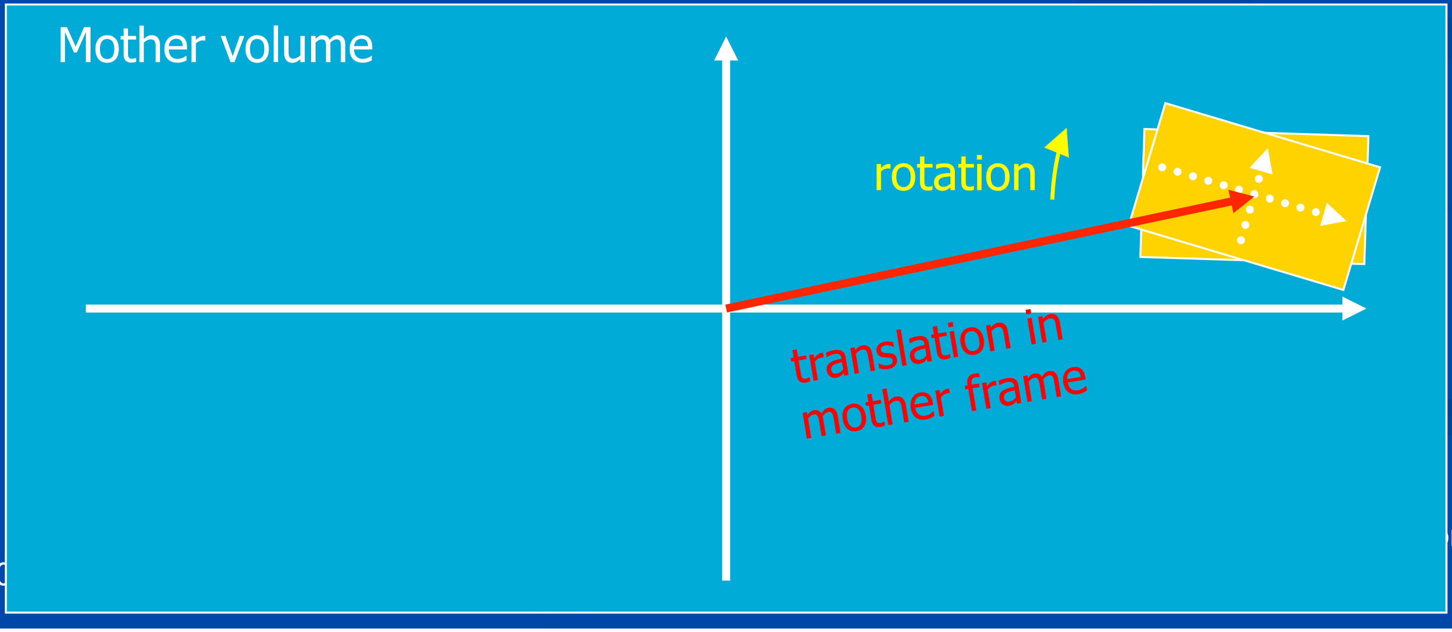
- Single volume positioned relatively to the mother volume.



Alternative G4PVPlacement

```
G4PVPlacement(
  G4Transform3D(G4RotationMatrix &pRot, // rotation of daughter frame
                const G4ThreeVector &tlate), // position in mother frame
  G4LogicalVolume *pDaughterLogical,
  const G4String &pName,
  G4LogicalVolume *pMotherLogical,
  G4bool pMany, // ‘true’ is not supported yet…
  G4int pCopyNo, // unique arbitrary integer
  G4bool pSurfChk=false); // optional boundary check
```

- Single volume positioned relatively to the mother volume.



Parameterized volume

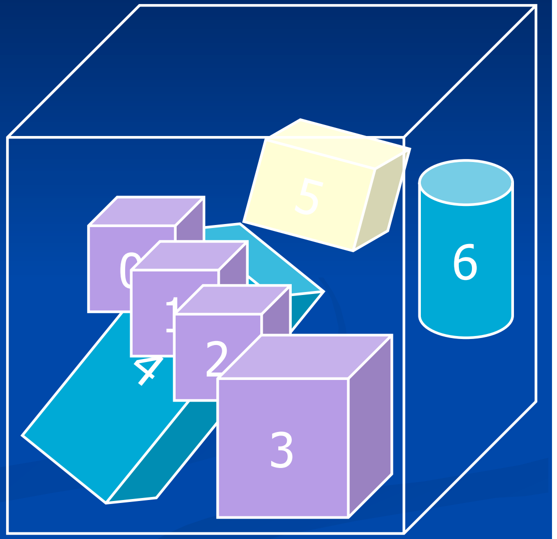
G4PVParameterised

```
G4PVParameterised(const G4String& pName,
                  G4LogicalVolume* pLogical,
                  G4LogicalVolume* pMother,
                  const EAxis pAxis,
                  const G4int nReplicas,
                  G4VPVParameterisation *pParam
                  G4bool pSurfChk=false);
```

- Replicates the volume `nReplicas` times using the parameterization `pParam`, within the mother volume `pMother`
- `pAxis` is a suggestion to the navigator along which Cartesian axis replication of parameterized volumes dominates.
  - kXAxis, kYAxis, kZAxis : one-dimensional optimization
  - kUndefined : three-dimensional optimization

Parameterized Physical Volumes

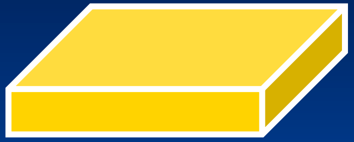
- User should implement a class derived from G4VPVParameterisation abstract base class and define following as a function of copy number
  - where it is positioned (transformation, rotation)
- Optional:
  - the size of the solid (dimensions)
  - the type of the solid, material, sensitivity, vis attributes
- All daughters must be fully contained in the mother.
- Daughters should not overlap to each other.
- Typical use-cases
  - Complex detectors
    - with large repetition of volumes, regular or irregular
  - Medical applications
    - the material in animal tissue is measured as cubes with varying material



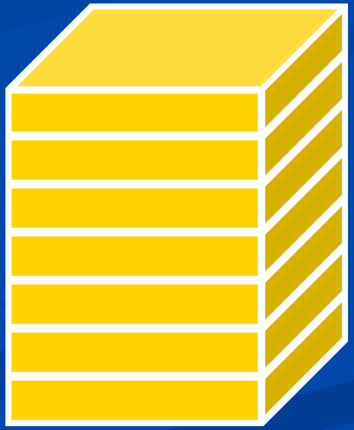
Replicated volume

Replicated Volumes

- The mother volume is completely filled with replicas, all of which are the same size (width) and shape.
- Replication may occur along:
  - Cartesian axes (X, Y, Z) – slices are considered perpendicular to the axis of replication
    - Coordinate system at the center of each replica
  - Radial axis (Rho) – cons/tubs sections centered on the origin and un-rotated
    - Coordinate system same as the mother
  - Phi axis (Phi) – phi sections or wedges, of cons/tubs form
    - Coordinate system rotated such as that the X axis bisects the angle made by each wedge



a daughter logical volume to be replicated



mother volume

G4PVReplica

```
G4PVReplica(const G4String &pName,
            G4LogicalVolume *pLogical,
            G4LogicalVolume *pMother,
            const EAxis pAxis,
            const G4int nReplicas,
            const G4double width,
            const G4double offset=0.);
```

- `offset` may be used only for tube/cone segment
- Features and restrictions:
  - Replicas can be placed inside other replicas
  - Normal placement volumes can be placed inside replicas, assuming no intersection/overlaps with the mother volume or with other replicas

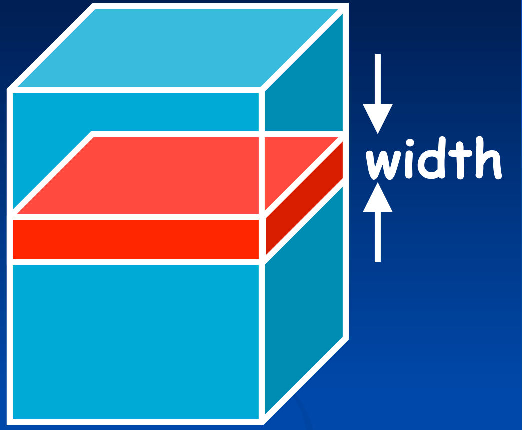
Replica - axis, width, offset

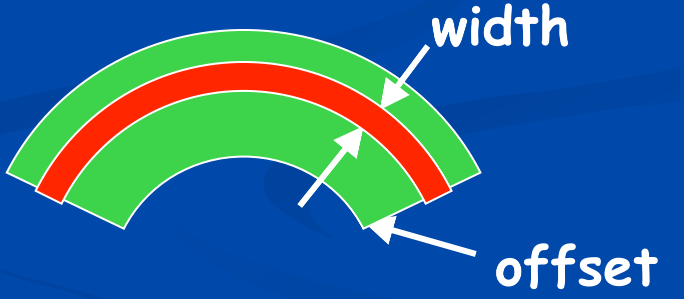
- Cartesian axes - **`kXaxis, kYaxis, kZaxis`**
  - Center of n-th daughter is given as

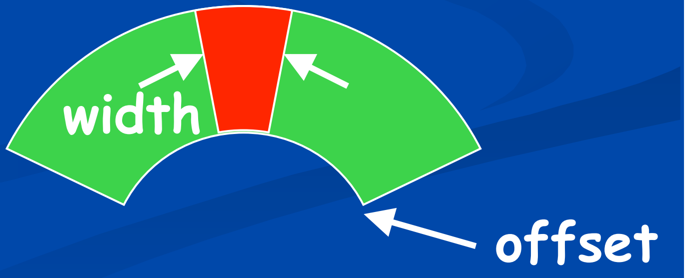
    `-width*(nReplicas-1)*0.5+n*width`
  - Offset shall not be used
- Radial axis - **`kRaxis`**
  - Center of n-th daughter is given as

    `width*(n+0.5)+offset`
  - Offset must be the inner radius of the mother
- Phi axis - **`kPhi`**
  - Center of n-th daughter is given as

    `width*(n+0.5)+offset`
  - Offset must be the starting angle of the mother

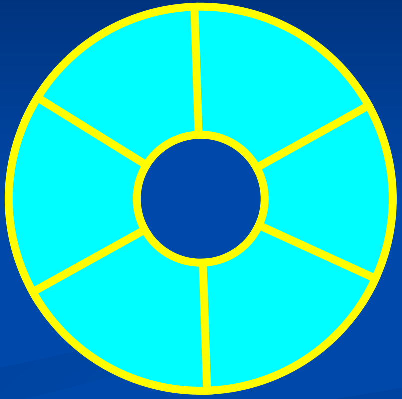


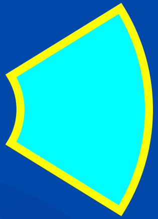




G4PVReplica : example

```
G4double tube_dPhi = 2.* M_PI * rad;
G4VSolid* tube =
   new G4Tubs("tube",20*cm,50*cm,30*cm,0.,tube_dPhi);
G4LogicalVolume * tube_log =
   new G4LogicalVolume(tube, Air, "tubeL", 0, 0, 0);
G4VPhysicalVolume* tube_phys =
   new G4PVPlacement(0,G4ThreeVector(-200.*cm,0.,0.),
            "tubeP", tube_log, world_phys, false, 0);
G4double divided_tube_dPhi = tube_dPhi/6.;
G4VSolid* div_tube =
   new G4Tubs("div_tube", 20*cm, 50*cm, 30*cm,
        -divided_tube_dPhi/2., divided_tube_dPhi);
G4LogicalVolume* div_tube_log =
   new G4LogicalVolume(div_tube,Pb,"div_tubeL",0,0,0);
G4VPhysicalVolume* div_tube_phys =
   new G4PVReplica("div_tube_phys", div_tube_log,
   tube_log, kPhi, 6, divided_tube_dPhi);
```





Define detector geometry

- Basic strategy

```
G4VSolid* pBoxSolid =
  new G4Box(“aBoxSolid”, 1.*m, 2.*m, 3.*m);
G4LogicalVolume* pBoxLog =
  new G4LogicalVolume( pBoxSolid, pBoxMaterial,
                              “aBoxLog”, 0, 0, 0);
G4VPhysicalVolume* aBoxPhys =
  new G4PVPlacement( pRotation,
    G4ThreeVector(posX, posY, posZ), pBoxLog,
    “aBoxPhys”, pMotherLog, 0, copyNo);
```

Geometry - Materials

Definition of material

Definition of Materials

- Different kinds of materials can be described:
  - isotopes <-> G4Isotope
  - elements <-> G4Element
  - molecules, compounds and mixtures <-> G4Material
- Attributes associated to G4Material:
  - temperature, pressure, state, density
- Prefer low-density material to vacuum

- Single element material

```
double density = 1.390*g/cm3;
double a = 39.95*g/mole;
G4Material* lAr =
 new G4Material("liquidArgon",z=18.,a,density);
```

Material: molecule

- A Molecule is made of several elements (composition by number of atoms)

```
a = 1.01*g/mole;
G4Element* elH  =
   new G4Element("Hydrogen",symbol="H",z=1.,a);
a = 16.00*g/mole;
G4Element* elO  =
   new G4Element("Oxygen",symbol="O",z=8.,a);
density = 1.000*g/cm3;
G4Material* H2O =
   new G4Material("Water",density,ncomp=2);
G4int natoms;
H2O->AddElement(elH, natoms=2);
H2O->AddElement(elO, natoms=1);
```

Material: compound

- Compound: composition by fraction of mass

```
a = 14.01*g/mole;
G4Element* elN  =
   new G4Element(name="Nitrogen",symbol="N",z= 7.,a);
a = 16.00*g/mole;
G4Element* elO  =
   new G4Element(name="Oxygen",symbol="O",z= 8.,a);
density = 1.290*mg/cm3;
G4Material* Air =
   new G4Material(name="Air",density,ncomponents=2);
G4double fracMass;
Air->AddElement(elN, fracMass=70.0*perCent);
Air->AddElement(elO, fracMass=30.0*perCent);
```

Material: mixture

- Composition of compound materials

```
G4Element* elC  = …;   // define “carbon” element
G4Material* SiO2 = …;  // define “quartz” material
G4Material* H2O = …;   // define “water” material

density = 0.200*g/cm3;
G4Material* Aerog =
   new G4Material("Aerogel",density,ncomponents=3);
Aerog->AddMaterial(SiO2,fractionmass=62.5*perCent);
Aerog->AddMaterial(H2O ,fractionmass=37.4*perCent);
Aerog->AddElement (elC ,fractionmass= 0.1*perCent);
```

Element with user defined abundance

- An element can be created according to user defined abundances

- Ex. Create an enriched Uranium for nuclear power generation

```
G4Isotope* isoU235 =
      new G4Isotope("U235", iz=92, ia=235, a=235.0439242*g/mole);
 G4Isotope* isoU238 =
    new G4Isotope("U238", iz=92, ia=238, a=238.0507847 *g/mole);

 G4Element* elenrichedU =
    new G4Element("enriched U", symbol="U" , ncomponents=2);
 elenrichedU->AddIsotope(isoU235, abundance=80.*perCent);
 elenrichedU->AddIsotope(isoU238, abundance=20.*perCent);

 G4Material* matenrichedU=
    new G4Material("U for nuclear  power generation" , density= 19.050*g/
     cm3 , ncomponents = 1 , kStateSolid );
 matenrichedU>AddElement( elenrichedU , fractionmass = 1 );
```

NIST Material Database in Geant4

NIST Elements and Isotopes

```
Z    A      m       error     (%)         Aeff
============================
14   Si  22   22.03453        (22)                      28.0855(3)
         23   23.02552        (21)
         24   24.011546       (21)
         25   25.004107       (11)
         26   25.992330       (3)
         27   26.98670476     (17)
         28   27.9769265327 (20)      92.2297 (7)
         29   28.97649472     (3)      4.6832 (5)
         30   29.97377022     (5)      3.0872 (5)
         31   30.97536327     (7)
         32   31.9741481      (23)
         33   32.978001       (17)
         34   33.978576       (15)
         35   34.984580       (40)
         36   35.98669        (11)
         37   36.99300        (13)
         38   37.99598        (29)
         39   39.00230        (43)
         40   40.00580        (54)
         41   41.01270        (64)
         42   42.01610        (75)
```

- Natural isotope compositions
- More than 3000 isotope masses are used for definition

NIST materials in Geant4

```
=====================================
###   Elementary Materials from the NIST Data Base
=====================================
 Z Name  ChFormula        density(g/cm^3)  I(eV)
=====================================
1   G4_H    H_2              8.3748e-05       19.2
2   G4_He                    0.000166322      41.8
3   G4_Li                    0.534            40
4   G4_Be                    1.848            63.7
5   G4_B                     2.37             76
6   G4_C                     2                81
7   G4_N    N_2              0.0011652        82
8   G4_O    O_2              0.00133151       95
9   G4_F                     0.00158029       115
10  G4_Ne                    0.000838505      137
11  G4_Na                    0.971            149
12  G4_Mg                    1.74             156
13  G4_Al                    2.6989           166
14  G4_Si                    2.33             173
```

- NIST Elementary Materials
  - H to Cf
- NIST Compounds
- HEP and Nuclear Materials
  - Ex. liquid Ar $PbWO_4$

```
====================================
###   Compound Materials from the NIST Data Base
====================================
 N Name      ChFormula        density(g/cm^3)  I(eV)
====================================
13  G4_Adipose_Tissue            0.92            63.2
            1     0.119477
            6     0.63724
            7     0.00797
            8     0.232333
           11     0.0005
           12     2e-05
           15     0.00016
           16     0.00073
           17     0.00119
           19     0.00032
           20     2e-05
           26     2e-05
           30     2e-05
4  G4_Air                        0.00120479     85.7
            6     0.000124
            7     0.755268
            8     0.231781
           18     0.012827
2  G4_CsI                        4.51            553.1
           53     0.47692
           55     0.52308
```

How to use

- Do not need anymore to predefine elements and materials
- Main new user interfaces:

```
G4NistManager* nist = G4NistManager::Instance();

G4Element* elm = manager->FindOrBuildElement(“symb”, G4bool iso);

G4Element* elm = manager->FindOrBuildElement(G4int Z, G4bool iso);

G4Material* mat = manager->FindOrBuildMaterial(“name”, G4bool iso);

G4Material* mat = manager->ConstructNewMaterial(“name”,
                        const std::vector<G4int>& Z,
                        const std::vector<G4double>& weight,
                        G4double density, G4bool iso);

G4double isotopeMass = manager->GetMass(G4int Z, G4int N);
```

UI commands

```
  /material/nist/printElement   --- print defined elements
  /material/nist/listMaterials  --- print defined materials
```