Intorduction to the Geant4

南開大學
Nankai University

徐 音
2014-8-12

Outline

- Basic concepts
- Geant4 kernel
- Describe Your Detector
- Physics Processes
- Primary Particles
- Optional User Action
- Geant4 UI
- Visualization

Email:
xuyin@nankai.edu.cn

Toolkit+User application

- Geant4 is a toolkit
  - i.e you cannot “run” it out of the box
  - You must write an application, which uses Geant4 tools
- Consequences
  - You must provide the necessary information to configure your simulation
  - You must choose which Geant4 tools to use

Email:
xuyin@nankai.edu.cn

What can a simulation package or toolkit do ?

- A Package provides ‘general’ tools to undertake (some or all) of the key tasks:
  - tracking, and geometrical propagation
  - modelling of physics interactions, visualization, persistency
- and enable you to describe your setup’s
  - detector geometry,
  - radiation source,
  - details of sensitive regions

Email:
xuyin@nankai.edu.cn

Basic concepts

- What you **MUST** do:
  - Describe your **experimental setup**
  - Provide the **primary particles** input to your simulation
  - Decide which **particles** and **physics models** you want to use and the precision of your simulation (cuts to produce and track secondary particles)
- You may also want
  - To interact with Geant4 kernel to **control** your simulation
  - To **visualise** your simulation configuration or results
  - To produce **histograms, tuples** etc. to be further analysed

Email:
xuyin@nankai.edu.cn

Basic concepts

- Gean4 kernel
  - Run, event, track, step
  - Trajectory, classes to define particle
  - Tracking and processes
  - Application states

- User application

Email:
xuyin@nankai.edu.cn

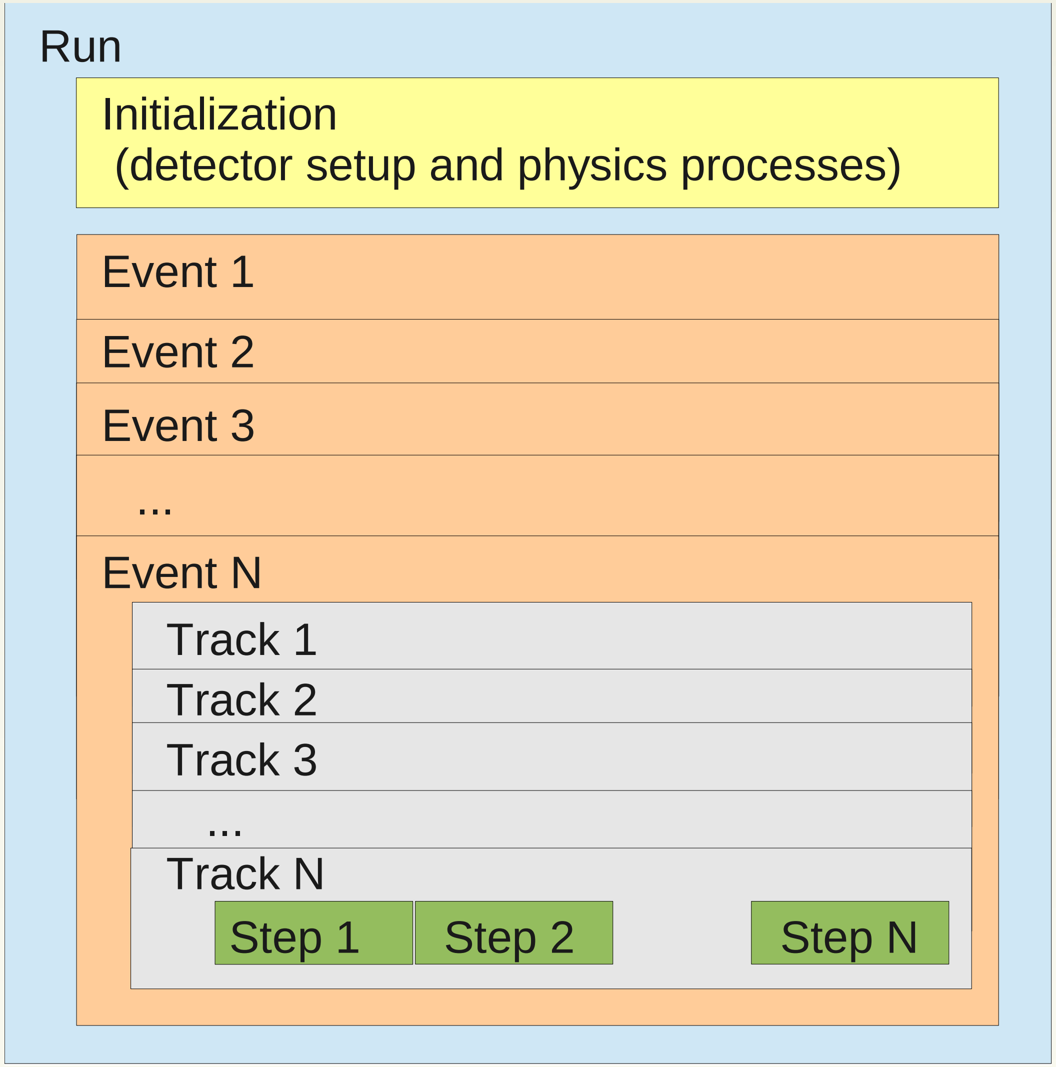
Geant4 Run

- User defines
  - Detector geometry, physics setup and primary particles in sets called (primary) events
- Geant4 kernel then loops over events
- In each event:
  - Loops over primaries
  - Each primary
    - Is tracked through the detector undergoing the registered physics processes
    - Which may create secondary particles (daughters)
- It tracks also its daughters
- Each track
  - Processed via steps

Email:
xuyin@nankai.edu.cn

Geant4 Run

- As an analogy of the real experiment, a run of Geant4 starts with "Beam On"
- Conceptually, *a run is a collection of events* which share the same detector and physics conditions.
- A run *consists of one event loop*
- G4RunManager class manages processing a run
- A run is represented by G4Run class or a user-defined class derived from G4Run.



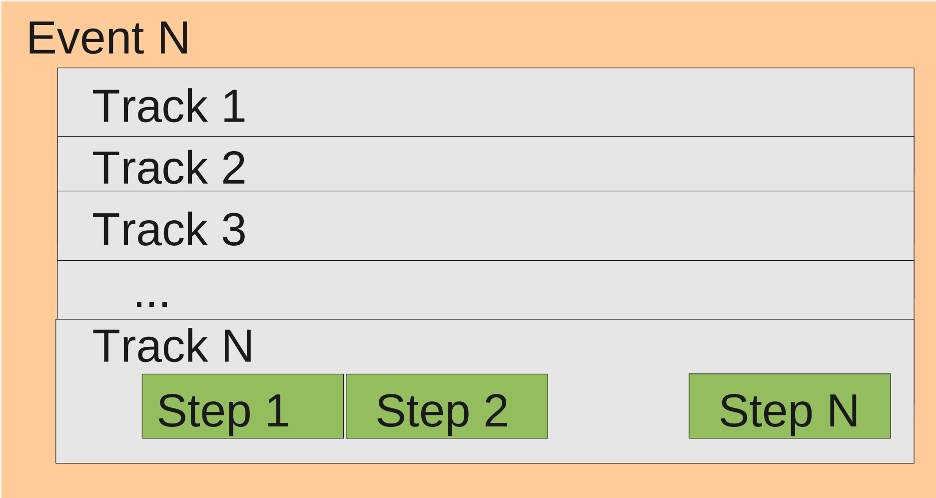
Email:
xuyin@nankai.edu.cn

Geant4 Event

- An event is the basic unit of simulation in Geant4.

- G4EventManager class manages processing an event
- G4Event class represents an event.

At the end of its (successful) processing, it has:
- List of primary vertices and particles (as input)
- Hits and Trajectory collections (as output)

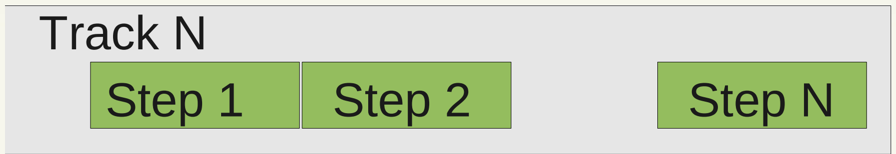


Email:
xuyin@nankai.edu.cn

Geant4 Track

- 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 its processing is finished
- No track object persists at the end of event.
  - For the record of tracks, use trajectory class objects.

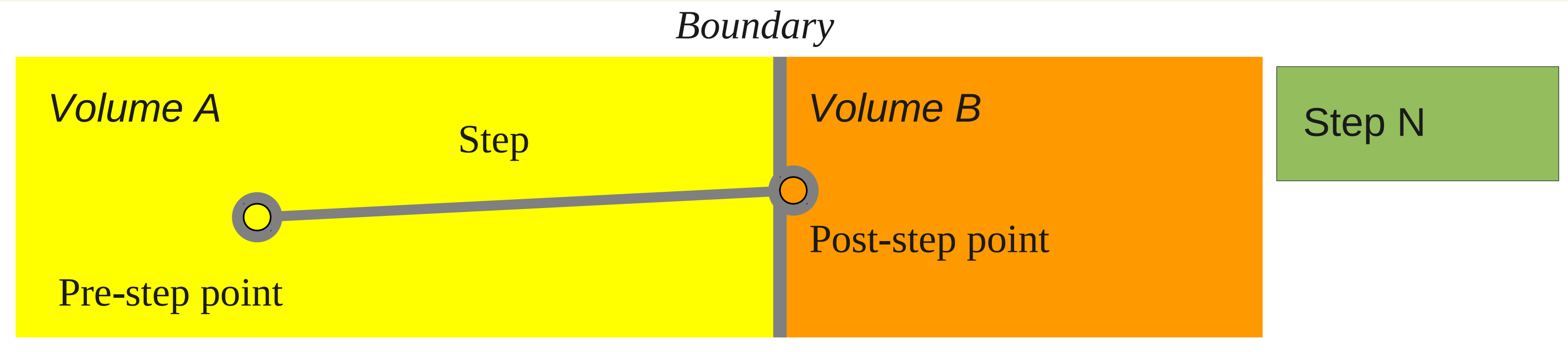
- G4TrackingManager class manages processing a track
- G4Track class represents a track.



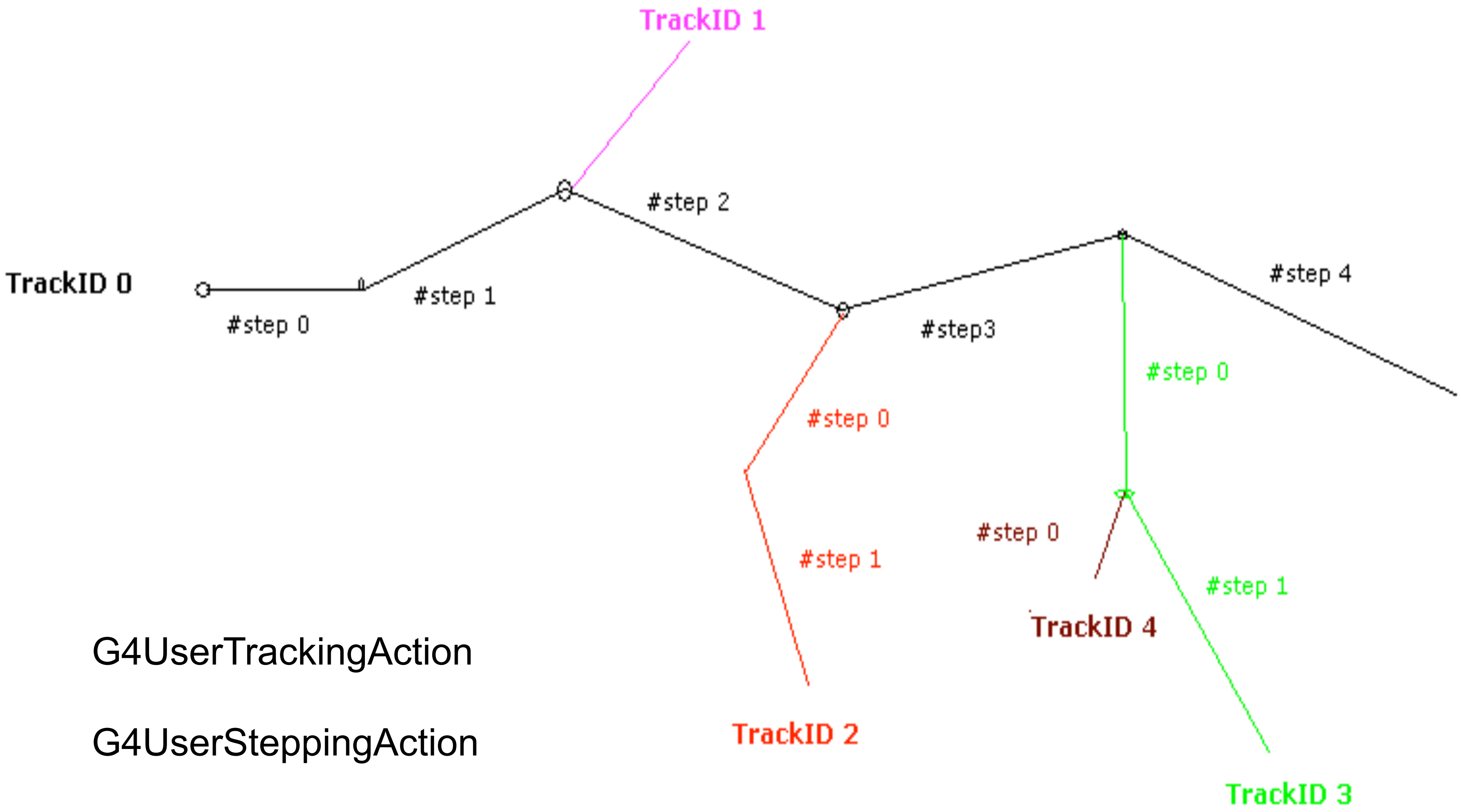
Email:
xuyin@nankai.edu.cn

Geant4 Step

- 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.



Email:
xuyin@nankai.edu.cn



G4UserTrackingAction

G4UserSteppingAction

Email:
xuyin@nankai.edu.cn

Geant4 Trajectory

- Track does not keep its trace. No track object persists at the end of event.
- G4Trajectory is the class which copies some of G4Track information. G4TrajectoryPoint is the class which copies some of G4Step information.
  - G4Trajectory has a vector of G4TrajectoryPoint.
  - At the end of event processing, G4Event has a collection of G4Trajectory objects.
  - Storing trajectories is optional, it has to be activated by user
- Given G4Trajectory and G4TrajectoryPoint objects persist till the end of an event, you should *be careful not to store too many trajectories*.
  - E.g. avoid for high energy EM shower tracks.
- G4Trajectory and G4TrajectoryPoint store only the minimum information.
  - You can create your own trajectory / trajectory point classes to store information you need. G4VTrajectory and G4VTrajectoryPoint are base classes.

Email:
xuyin@nankai.edu.cn

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 being tracked.
- G4DynamicParticle
  - "Dynamic" physical properties of a particle, such as momentum, energy, spin, etc.
  - Each G4Track object has its own and unique G4DynamicParticle object.
- 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

Email:
xuyin@nankai.edu.cn

User Application

- User action classes derived from
  G4UserRunAction, G4UserEventAction, G4UserStackingAction, G4UserTrackingAction, G4UserSteppingAction

```
G4UserRunAction
    G4Run* GenerateRun()
    void BeginOfRunAction(const G4Run*)
    void EndOfRunAction(const G4Run*)

G4UserEventAction
    void BeginOfEventAction(const G4Event*)
    void EndOfEventAction(const G4Event*)

G4UserSteppingAction
    void UserSteppingAction(const G4Step*)
```


Email:
xuyin@nankai.edu.cn

User Application

Geant4 is a toolkit. You have to build an application. You have to

- Define your geometrical setup (materials, volumes), physics to get involved (particles, physics processes/models), production thresholds, 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

- main() program
- User initialization classes derived from Geant4 base classes: G4VUserDetectorConstruction, G4VUserPhysicsList, G4VPrimaryGeneratorAction
- User action classes derived from G4UserRunAction, G4UserEventAction, G4UserStackingAction, G4UserTrackingAction, G4UserSteppingAction

Email: xuyin@nankai.edu.cn

Main program

- Geant4 does not provide the **main()**
  - Geant4 is a toolkit!
  - The main() is part of the user application

- In his/her main(), the user **must**
  - construct **G4RunManager** (or his/her own derived class)
  - notify the G4RunManager mandatory user classes derived from
    - ***G4VUserDetectorConstruction***
    - ***G4VUserPhysicsList***
    - ***G4VUserPrimaryGeneratorAction***

- The user **may** define in his/her main()
  - optional user action classes
  - VisManager, (G)UI session

Email:
xuyin@nankai.edu.cn

main()

```
{
    // Construct the default run manager
    G4RunManager* runManager = new G4RunManager;

    // Set mandatory user initialization classes
    MyDetectorConstruction* detector = new MyDetectorConstruction;
    runManager->SetUserInitialization(detector);
    runManager->SetUserInitialization(new MyPhysicsList);

    // Set mandatory user action classes
    runManager->SetUserAction(new MyPrimaryGeneratorAction);

    // Set optional user action classes
    MyEventAction* eventAction = new MyEventAction();
    runManager->SetUserAction(eventAction);
    MyRunAction* runAction = new MyRunAction();
    runManager->SetUserAction(runAction);
}
```


Email:
xuyin@nankai.edu.cn

Describe Your Detector

- To describe your detector you have to derive your own concrete class from G4VUserDetectorConstruction abstract base class.
- Implement the virtual method Construct(), where you
  - Instantiate all necessary materials
  - Instantiate volumes of your detector geometry
  - 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

Email:
xuyin@nankai.edu.cn

Select Physics Processes

- Geant4 does not have any default particles or processes however it provides a rich set of the physics lists for various use-cases
  - You can just instantiate the most suitable one for your application
- If none of these lists suites your needs you can cook your own one
  - Derive your own concrete class from G4VUserPhysicsList abstract base class.
  - Define all necessary particles in ConstructParticle() virtual function
  - Define all necessary processes and assign them to proper particles in ConstructProcess() virtual function
- Define cut-off ranges applied to the world (and each region)
- Geant4 provides lots of utility classes/methods and examples.

Email: xuyin@nankai.edu.cn

Quick Overview of Physics Processes Provided by Geant4

- EM physics
  - “standard” processes valid from ~ 1 keV to ~PeV
  - “low energy” valid from 250 eV to ~PeV
  - optical photons
- Weak interaction physics
  - decay of subatomic particles
  - radioactive decay of nuclei
- Hadronic physics
  - pure strong interaction physics valid from 0 to ~TeV
  - electro- and gamma-nuclear valid from 10 MeV to ~TeV
- Parameterized or “fast simulation” physics

Email:
xuyin@nankai.edu.cn

Pre-packaged or Reference Physics Lists

- The pre-packaged physics list are a set of physics lists based on G4VModularPhysicsList and which respond to frequent use-cases.
  - HEP, medical, shielding, etc...
- Each pre-packaged (or reference) physics list includes different choices of EM and hadronic physics
- These can be found on the Geant4 web page at
  - geant4.cern.ch/support/proc_mod_catalog/physics_lists/physicsLists.shtml ; and subsequent "Reference Physics Lists" link

Email:
xuyin@nankai.edu.cn

Reference Physics Lists

A web page recommending physics lists according to the use case is under construction. The previous version of physics list web pages referring to 'are still available.

String model based physics lists

These Physics lists apply a **string model** for the modeling of interactions of high energy hadrons, i.e. for protons, neutrons, pions and kaons above ~(5-25) GeV depending on the exact physics list. Interactions at lower energies are handled by one of the intranuclear cascade models or the precompound model. Nuclear capture of negative particles and neutrons at rest is handled using either the Chiral Invariant Phase Space (CHIPS) model or the Bertini intranuclear cascade. Hadronic inelastic interactions use:

- a tabulation of the Barashenkov pion cross sections
- the Axen-Wellisch parameterization of the proton and neutron cross sections

The physics lists are:

- **QGSP and QGSP_EMV**
  QGSP is the basic physics list applying the quark gluon string model for high energy interactions of protons, neutrons, pions, and Kaons and nuclei. The high energy interaction creates an exited nucleus, which is passed to the precompound model modeling the nuclear de-excitation.

  QGSP_EMV is identical to QGSP, but parameters of electromagnetic processes tuned to yield better cpu performance with only slightly less precision.
- **QGSC and QGSC_EMV**
  As QGSP except applying CHIPS modeling for the nuclear de-excitation. In comparison to thin target experiments, this improves simulation of the nuclear de-excitation part of the interaction, resulting in slightly increased production of relatively low energy secondary protons (and neutrons).

  QGSC_EMV is identical to QGSC, but parameters of electromagnetic processes tuned to yield better cpu performance with only slightly less precision.
- **QGSP_EFLOW**
  This variant of QGSC uses a different algorithm setting up the excited nucleus created by the high energy interaction resulting in a good description of target fragmentation products; comparisons to thin target data well reproduce the proton production rate in the nuclear fragmentation region.
- **QGSP_BERT and QGSP_BERT_EMV**
  Like QGSP, but using Geant4 Bertini cascade for primary protons, neutrons, pions and Kaons below ~10GeV. In comparison to experimental data we find improved agreement to data compared to QGSP which uses the low energy parameterised (LEP) model for all particles at these energies. The Bertini model produces more secondary neutrons and protons than the LEP model, yielding a better agreement to experimental data.

  QGSP_BERT_EMV is like QGSP_BERT, but parameters of electromagnetic processes tuned to yield better cpu performance with only slightly less precision.

  Both QGSP_BERT and QGSP_BERT_EMV are less CPU performant as QGSP.
- **QGSP_BERT_HP**
  This list is similar to QGSP_BERT and in addition uses the data driven high precision neutron package (NeutronHP) to transport neutrons below 20 MeV down to thermal energies.
- **QGSP_BERT_TRV**
  This is a variant of QGSP_BERT where the Geant4 Bertini cascade is only used for particles below ~5.5 GeV.
- **QGSP_BIC and QGSP_BIC_HP**
  Like QGSP, but using Geant4 Binary cascade for primary protons and neutrons with energies below ~10GeV, thus replacing the use of the LEP model for protons and neutrons In comparison to the LEP model, Binary cascade better describes production of secondary particles produced in interactions of protons and neutrons with nuclei.

  Both lists, QGSP_BIC and QGSP_BIC_HP, also use the binary light ion cascade to model inelastic interaction of ions up to few GeV/nucleon with matter.

  The list QGSP_BIC_HP is like QGSP_BIC with the addition to use the data driven high precision neutron package (NeutronHP) to transport neutrons below 20 MeV down to thermal energies.
- **QGSP_NEQ, QGSP_EMV_NQE, and QGSP_BERT_NQE**
  These lists correspond to the lists without the trailing _NQE, except that here the quasi-elastic channel for high energy inelastic reactions is ignored. This quasi-elastic channel was missing from string model based physics lists prior to release 8.3. To allow comparison to results obtained with older releases of Geant4, i.e. 8.2 and before, these lists are provided for a transisition period.
- **QGSP_INCLXX**
  This is an experimental physics list that uses the Liege Intranuclear Cascade model (INCL++) for proton-, neutron- and pion-induced reactions below ~3 GeV, instead of BERT or BIC. INCL++ is also used for reactions induced by light nuclei (up to A=18).
- **FTFP_BERT, FTFP, FTFP_EMV**
  In FTF physics lists, a different string model is used. The FTF model is based on the FRITIOF description of string excitation and fragmentation.

Email:
xuyin@nankai.edu.cn

Yours Physicslist

Hadronic Processes

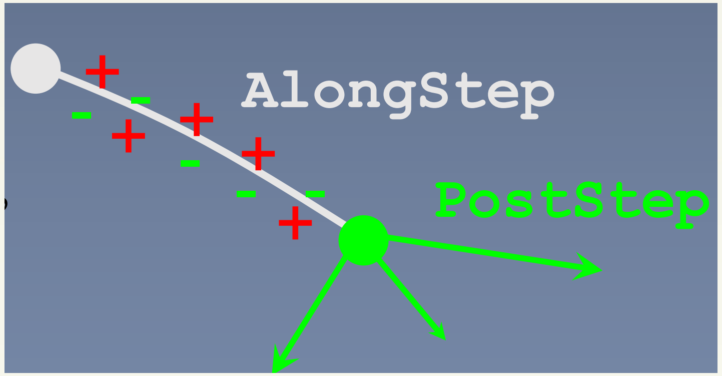
- Pure Hadronic Processes (0 - ~TeV)
  - elastic
  - Inelastic
  - Capture
  - fission
- Strong Radioactive Decay
  - at rest
  - In flight
- Photo-Nuclear (~10 MeV - ~TeV)
  - Gamma-nuclear reactions
- Lepton-Nuclear (~10 MeV - ~TeV)
  - e+, e- nuclear reactions
  - muon nuclear reactions

Email:
xuyin@nankai.edu.cn

kind of actions

Define three kinds of actions:

- AtRest actions:
  - Decay, e+ annihilation ...
- AlongStep actions:
  - To describe continuous(inter)actions, occuring along the path of the particle, like ionisation;
- PostStep actions:
  - For describing point-like (inter)actions, like decay in flight, hard radiation...



Email:
xuyin@nankai.edu.cn

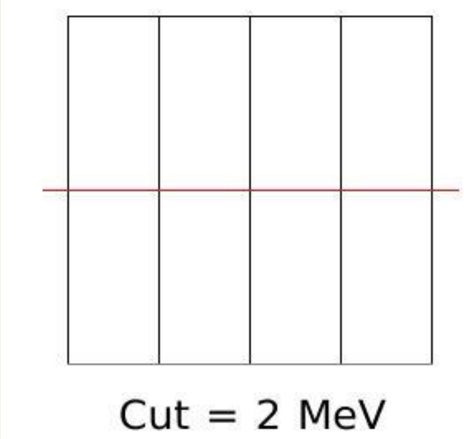
Production Thresholds （cuts）

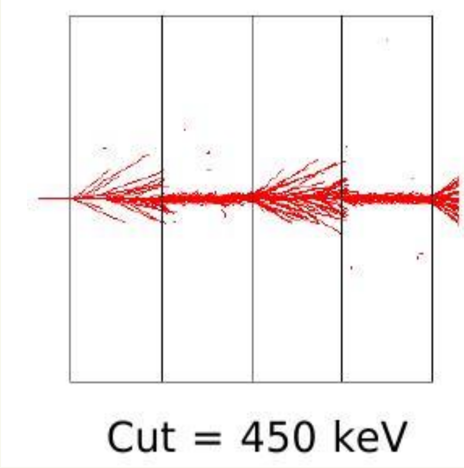
High Threshold

- No secondary production
- All energy lost by the primary particle goes into the local energy deposit
  - Continuous energy loss

Low Threshold

- Many secondaries produced
- Energy lost by primary shared between:
  - Local energy deposit
  - discrete secondary production

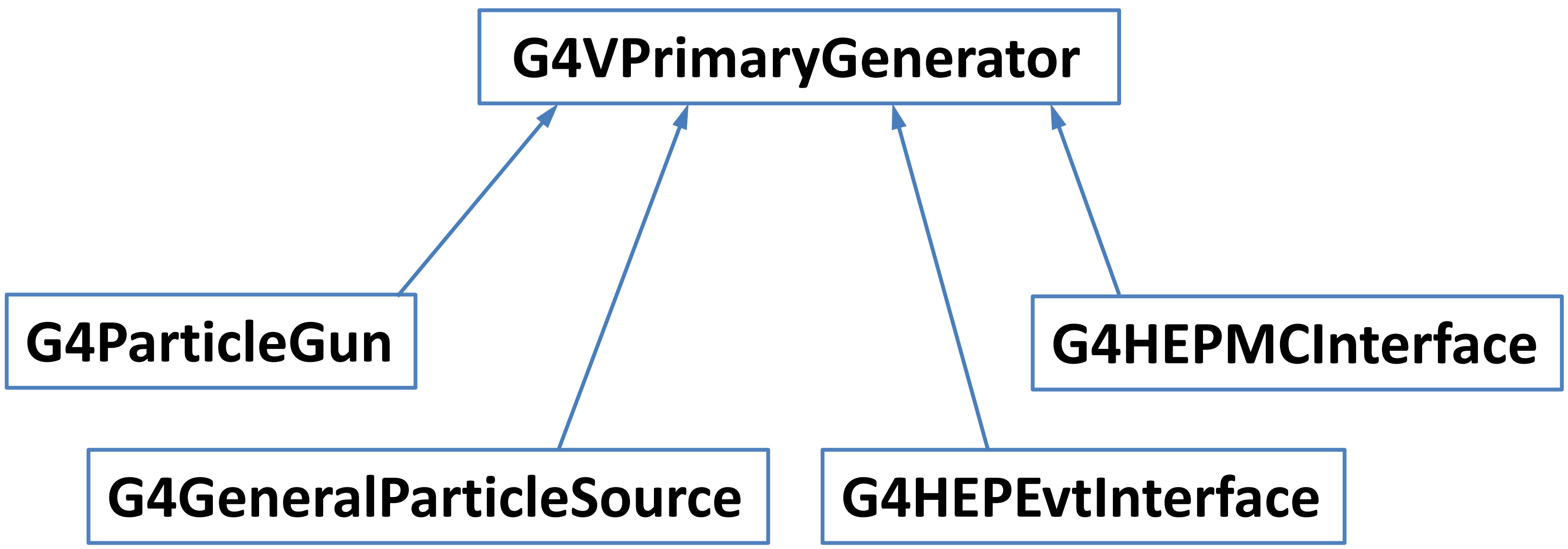




Email:
xuyin@nankai.edu.cn

Primary Particles

Geant4 provides some concrete implementations for G4VPrimaryGenerator:



Email:
xuyin@nankai.edu.cn

G4ParticleGun

- The simplest G4VPrimaryGenerator implementation:
  - Shoot one or several particle(s) at a time,
  - All of same fixed type, energy, momentum direction, position etc.
- Particle gun configured with methods:

  SetNumberOfParticles(G4int)
  SetParticleDefinition(G4ParticleDefinition*)
  SetParticleMomentum(G4ParticleMomentum)
  SetParticleMomentumDirection(G4ThreeVector)
  SetParticleEnergy(G4double)
  SetParticlePosition(G4ThreeVector)
  SetParticlePolarization(G4ThreeVector)

Email:
xuyin@nankai.edu.cn

```
void EDPrimaryGeneratorAction::GeneratePrimaries(G4Event* event)
{
  //this function is called at the begining of ecah event
  //

  // Define particle properties
  G4String particleName = "proton";
  //G4String particleName = "geantino";
  G4ThreeVector position(0, 0, -9.*m);
  G4ThreeVector momentum(0, 0, 1.*GeV);
  G4double time = 0;

  // Get particle definition from G4ParticleTable
  G4ParticleTable* particleTable = G4ParticleTable::GetParticleTable();
  G4ParticleDefinition* particleDefinition
    = particleTable->FindParticle(particleName);
  if ( ! particleDefinition ) {
    G4cerr << "Error: " << particleName << " not found in G4ParticleTable" << G4endl;
    exit(1);
  }

  // Create primary particle
  G4PrimaryParticle* primaryParticle = new G4PrimaryParticle(particleDefinition);
  primaryParticle->SetMomentum(momentum.x(), momentum.y(), momentum.z());
  primaryParticle->SetMass(particleDefinition->GetPDGMass());
  primaryParticle->SetCharge( particleDefinition->GetPDGCharge());

  // Create vertex
  G4PrimaryVertex* vertex = new G4PrimaryVertex(position, time);
  vertex->SetPrimary(primaryParticle);
  event->AddPrimaryVertex(vertex);
}
```

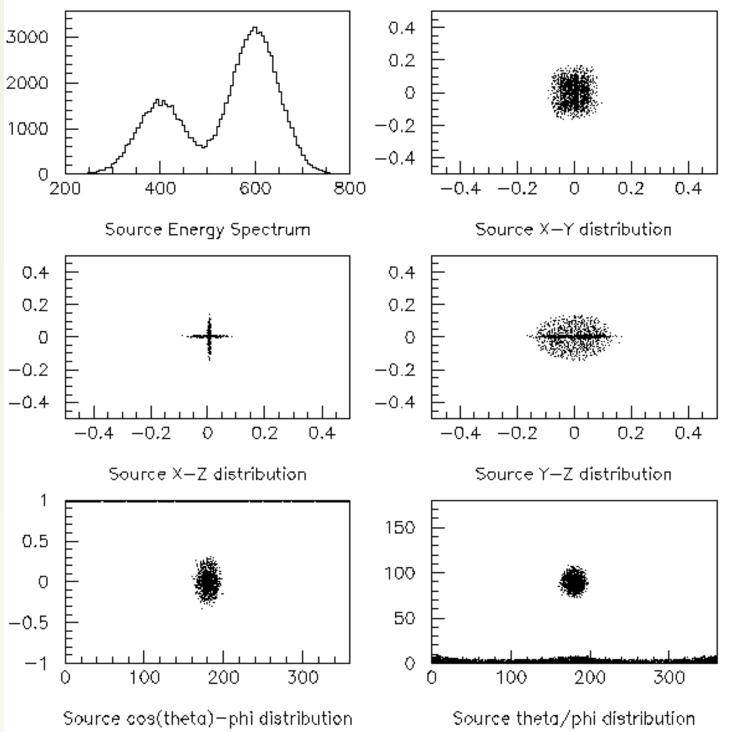

Email:
xuyin@nankai.edu.cn

G4GeneralParticleSource (GPS)

Macro file commands:
/gps/particle proton
/gps/pos/type Point
/gps/pos/centre 1. 2. 1. cm
/gps/ang/type iso
/gps/energy 2. MeV

http://reat.space.qinetiq.com/gps

Resulting distributions



Email:
xuyin@nankai.edu.cn

Interfaces to HEPEvt and HepMC

- Interface implementations of G4VPrimaryGenerator to standard formats in HEP:
  - useful for experiment-specific primary generator implementation
- G4HEPEvtInterface:
  - Suitable to /HEPEVT/ common block, which many of HEP physics generators are compliant to
  - – ASCIIfileinput
- G4HepMCInterface:
  - An interface to HepMC class, which a few new HEP physics generators are compliant to Eg :Pythia
  - ASCII file input or direct linking to a generator through HepMC

Email: xuyin@nankai.edu.cn

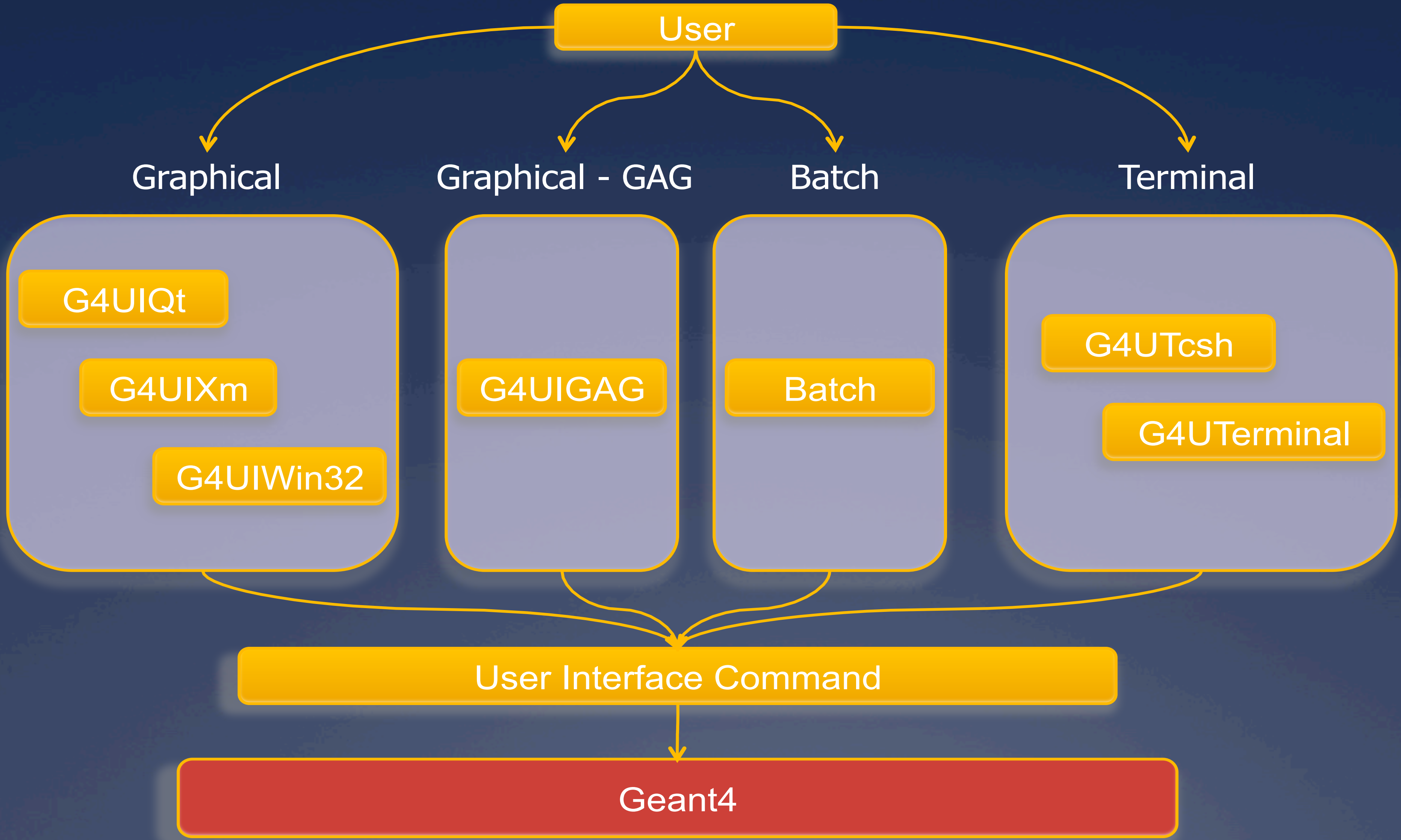
Optional User Action Classes

- All user action classes, methods of which are invoked during event processing, must be constructed in the user's main() and must be set to G4RunManager.
- The action classes methods are then called by Geant4 kernel in an appropriate phase of event processing

G4UserRunAction
G4UserEventAction
G4UserStackingAction
G4TrackingAction
G4UserSteppingAction

Email:
xuyin@nankai.edu.cn

Geant4 UI overview



Email:
xuyin@nankai.edu.cn

Geant4 UI command

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

/run/verbose 1

/vis/viewer/flush

/control/	/random/
/units/	/particle/
/geometry/	/process/
/tracking	/hits/
/event/	/gun
/run/	/vis/

- Macro file can be executed using the command :

"Idle>/control/execute file_name"

Email:
xuyin@nankai.edu.cn

Batch mode/interactive mode

```
int main(int argc, char** argv)
 {
...
if (argc != 1)
{ // batch mode
G4String command = "/control/execute ";
G4String fileName = argv[1];
UImanager->ApplyCommand(command+fileName);
}
 else
{ // interactive mode : define UI session
G4UIExecutive* ui = new G4UIExecutive(argc, argv);
ui->SessionStart();
delete ui;
}
```

Call your executable

- Interactive mode
  $> my_application
- Batch mode
  $> my_application run1.mac

Email:
xuyin@nankai.edu.cn

Geant4 Visualization

- Some visualization drivers work directly from Geant4
  - OpenGL
  - OpenInventor
  - RayTracer
  - ASCIITree

- For other visualization drivers, you first have Geant4 produce a file, and then you have that file rendered by another application (which may have GUI control)
  - HepRepFile
  - DAWNFILE
  - VRML2FILE
  - gMocrenFile

Email:
xuyin@nankai.edu.cn

Geant4 Visualization

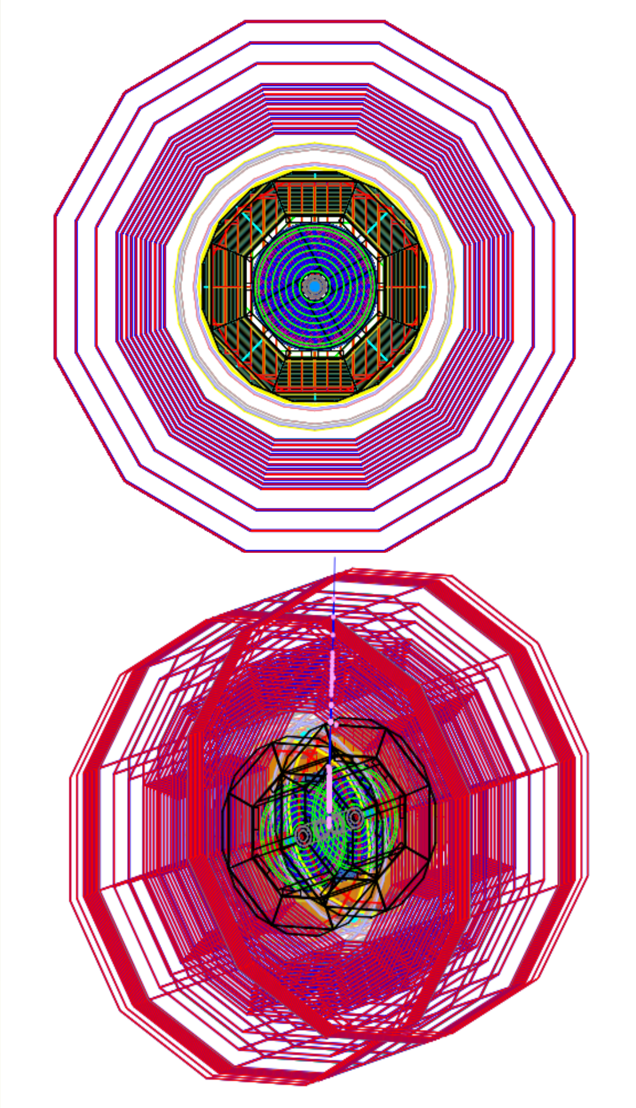
Driver	Variant	Hight quality print	Interactive	browse geometry hierarchies	Direct access to G4 kernel	Make movies	Web
OpenGL	X						
	Xm						
	Qt						
	Win32						
OpenInventor	Xt						
	Win32						
DAWN							
VRML							
HepRep							
gMocren							
RayTracer							
ACSII File							

Email:
xuyin@nankai.edu.cn

Geant4 Visualization

- To Open a Driver
  /vis/open <driver name>
  for example
  - /vis/open OGL
  - /vis/open DAWNFILE
  - /vis/open HepRepFile
  - /vis/open VRML2FILE

- To draw the entire detector geometry:
  - /vis/drawVolume



Email:
xuyin@nankai.edu.cn

Thanks

南開大學
Nankai University

徐 音
2014-8-12