Geant4 Kernel

Geant4 and its Application to HEP and Astrophysics
December 5, 2022

Sunanda Banerjee

Geant4

- Geant4 is a toolkit which helps to build an application program for simulating the performance of a detector exposed to radiation
  - Originally built for experiments in high energy and nuclear physics
  - Also finds its application in space physics and medical science

- The public production version has been available since the year 1999

- The package came with a number of examples to guide buildings of application programs

- It is written mostly in the C++ language using object-oriented technology

- Available as open source through CERN and works on a number of platforms
  - Latest version Geant4.11.1

- Three main reference papers:
  - Nuclear Instruments and Methods A506 (2003) 250
  - IEEE Transactions on Nuclear Science 53 (2006) 270
  - Nuclear Instruments and Methods A835 (2016) 186

Code Arrangement

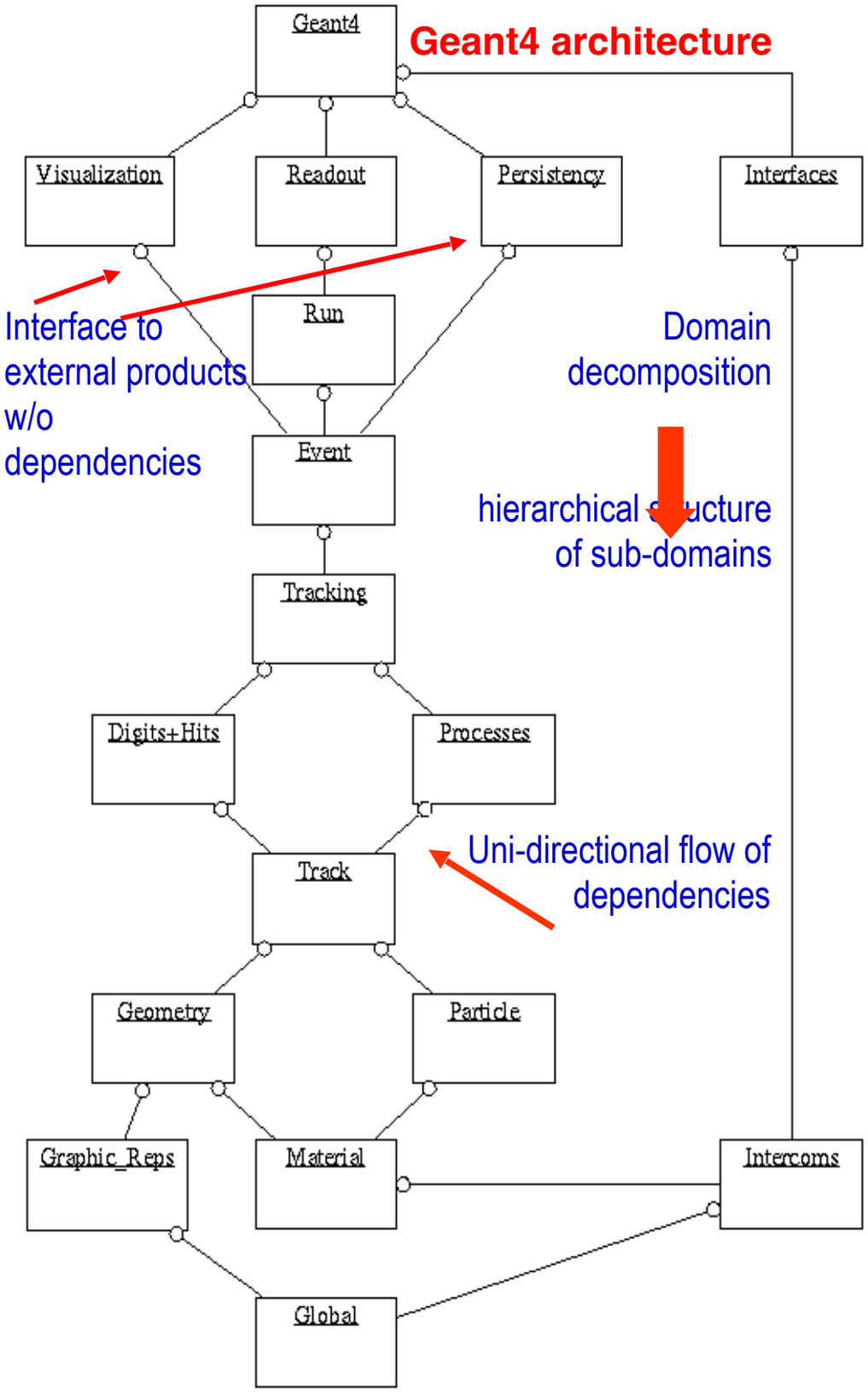
- All codes of Geant4 are grouped into 17 categories

- Relationships among classes from two categories:
  - Always one sided
  - No cyclic dependencies

- "Global" is at the lowest level
  - No dependency on any class from other categories of Geant4

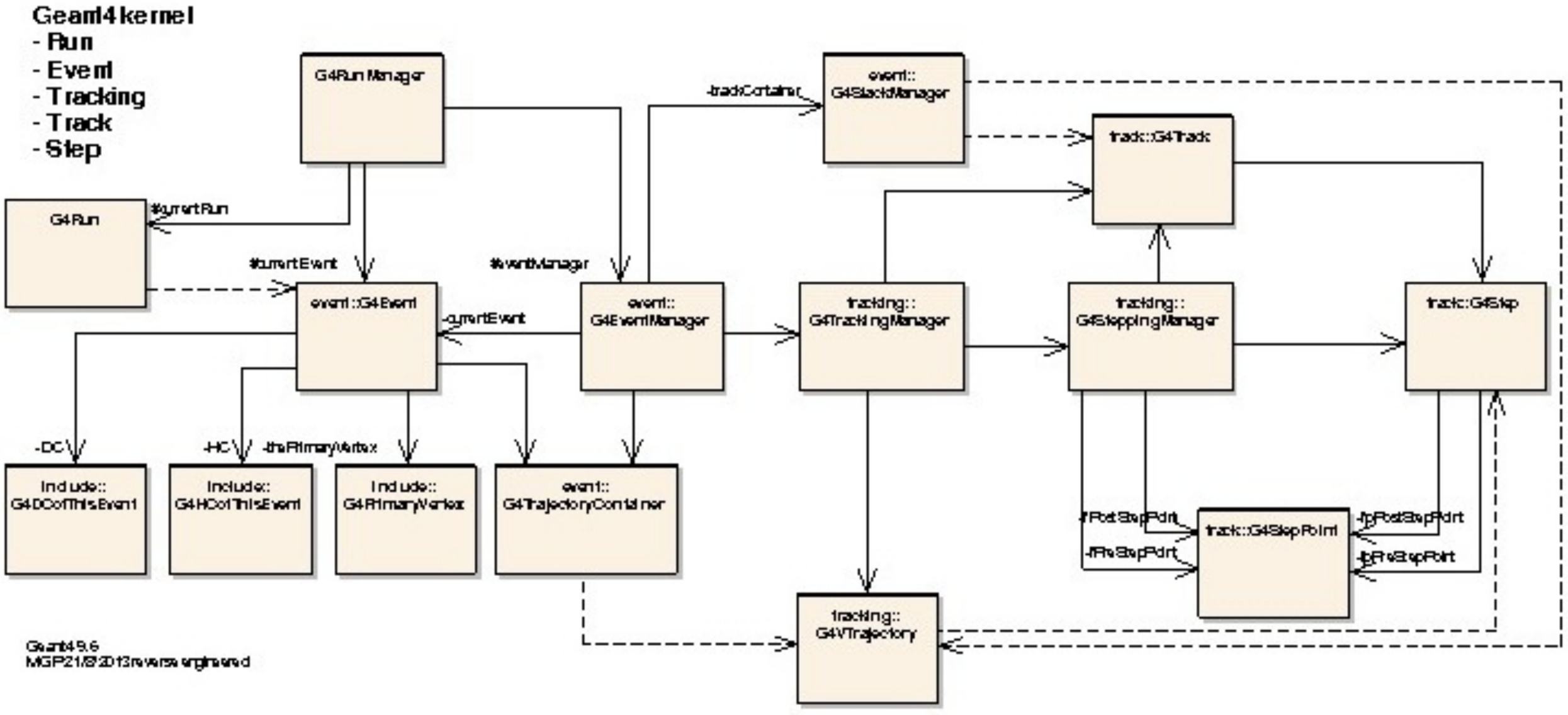
- "Geant4" is at the highest level
  - Classes in this category depend on all other categories

- External dependencies:
  - CLHEP
  - PTL
  - EXPAT
  - XERCES
  - … some Graphics packages …

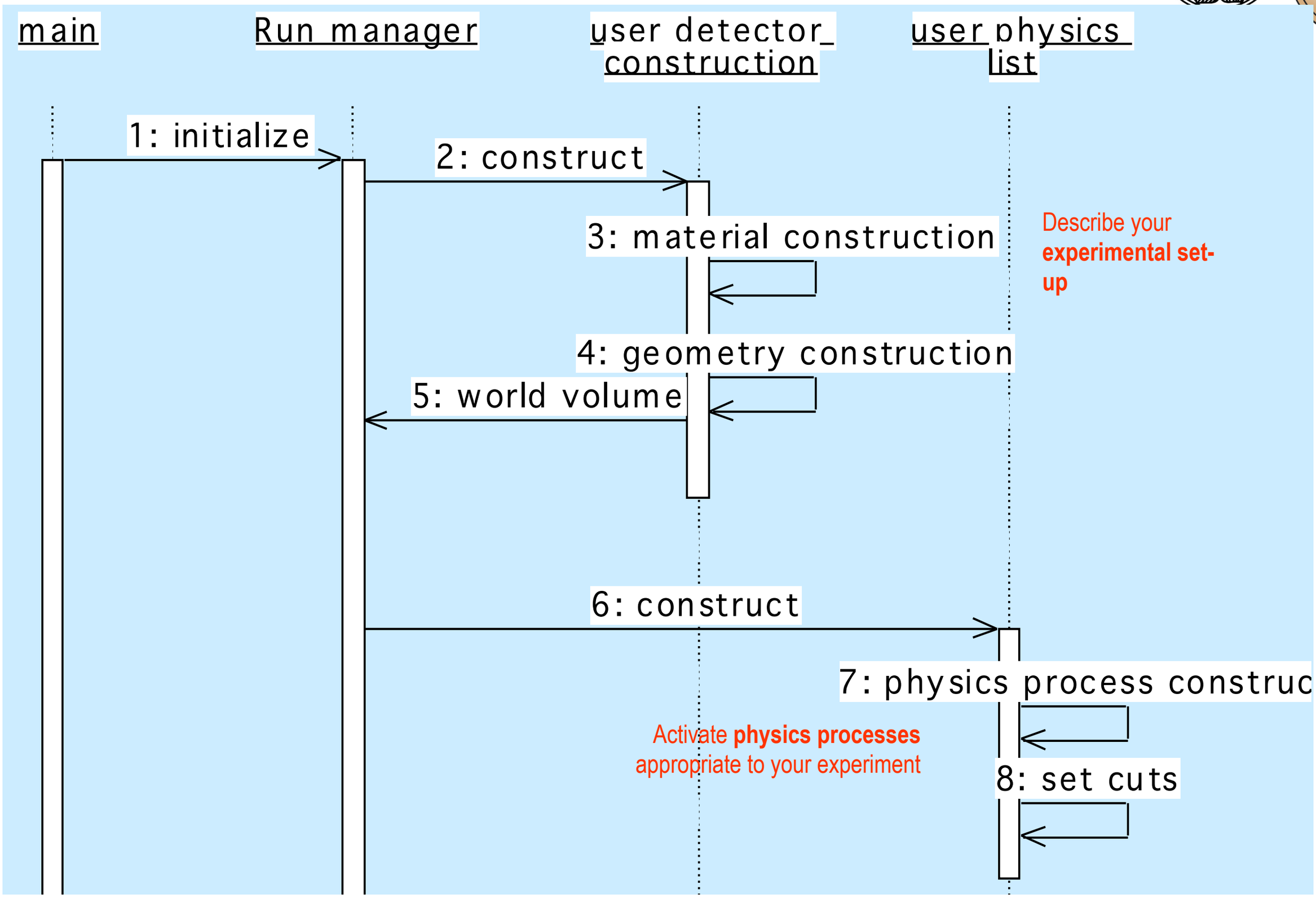


Some of the terminologies

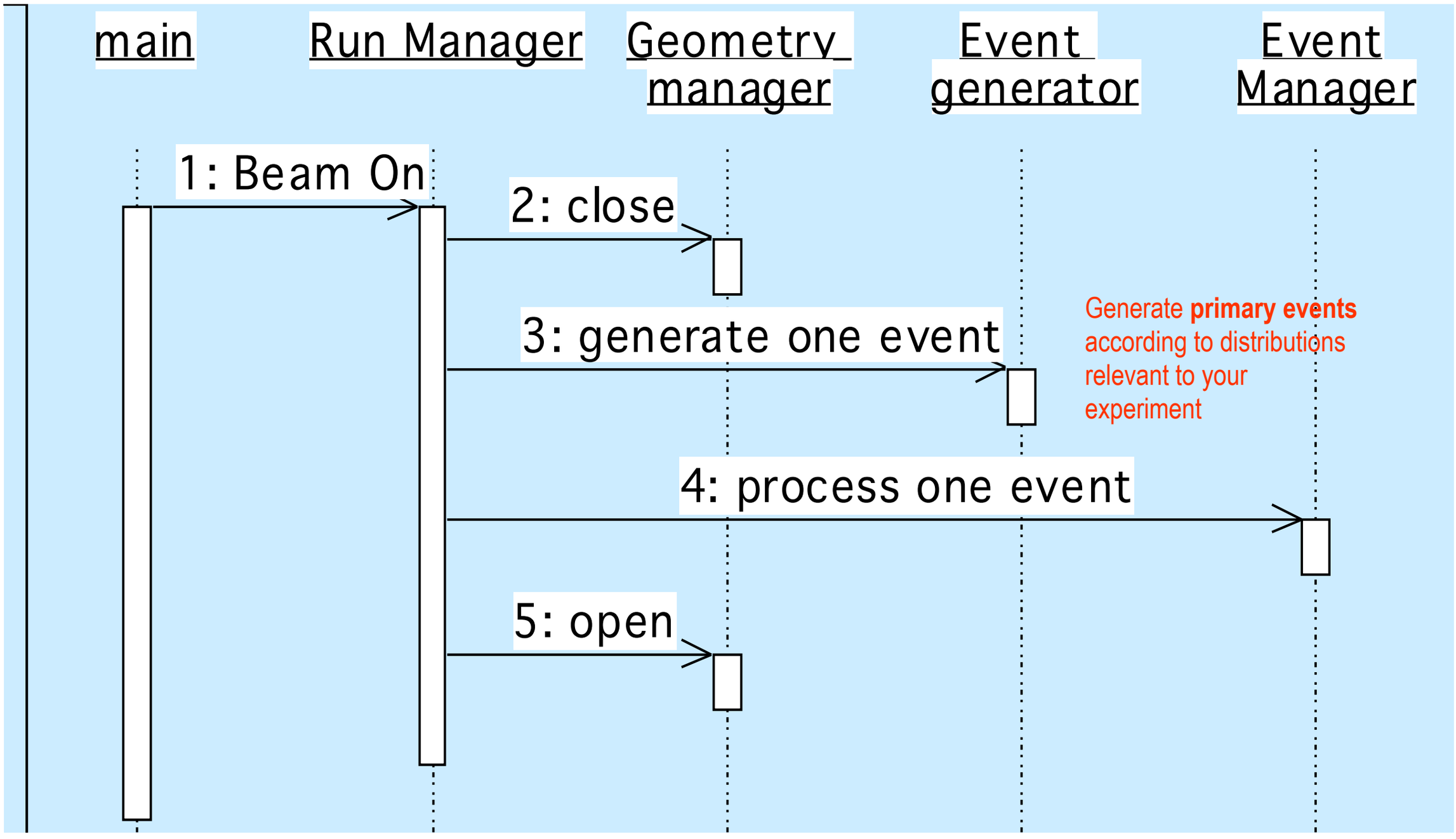
- Run, Events Track, Step, Stack, …
- Track vs. Trajectory; Step vs. Trajectory Point
- Particle, Process, Hits, …



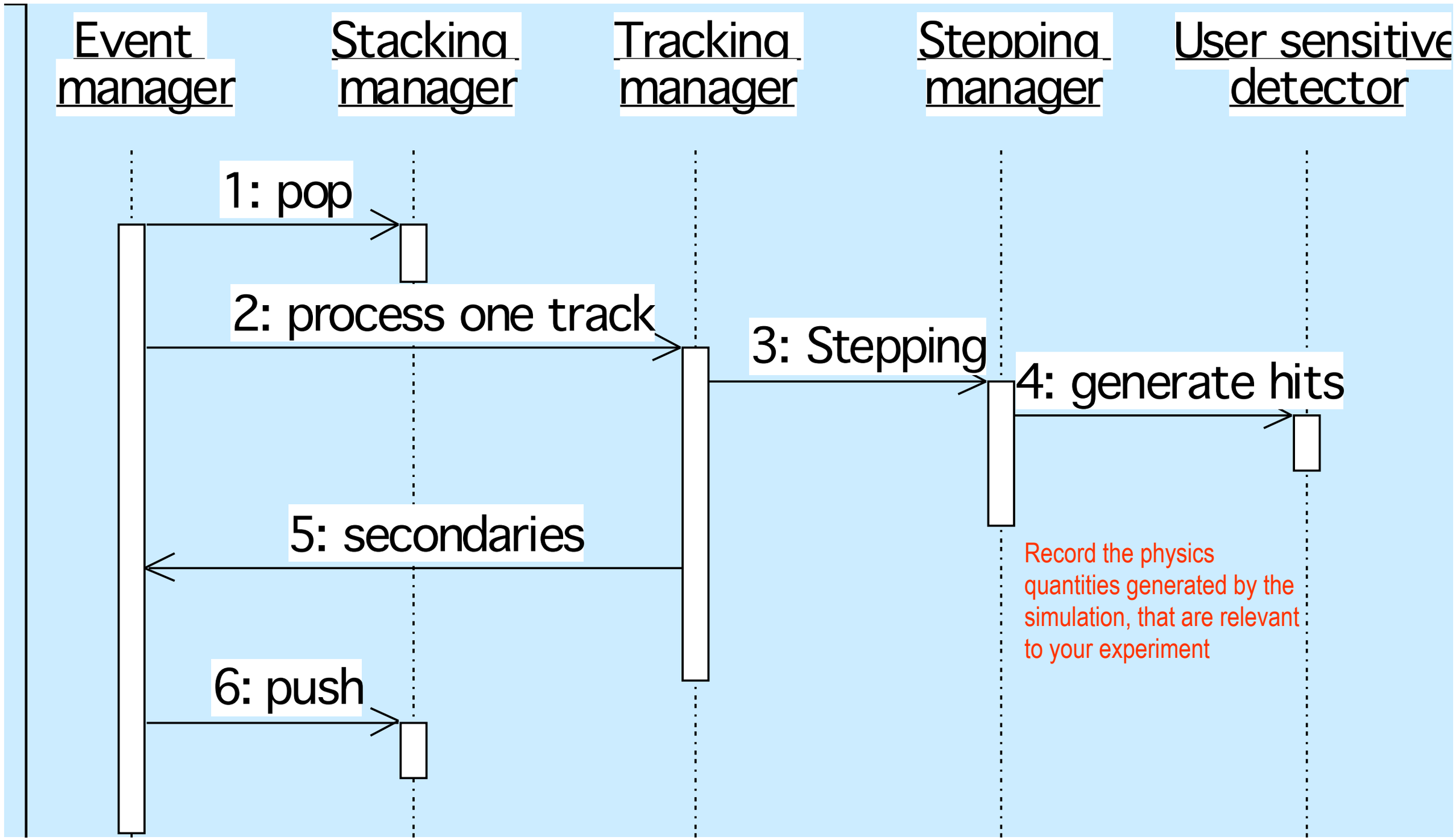
Initialization



Event Loop


main
Run Manager
Geometry_ manager
Event generator
Event Manager
1: Beam On
2: close
3: generate one event
Generate primary events according to distributions relevant to your experiment
4: process one event
5: open

Event Processing



Run in Geant4

- Conceptually, a run is a collection of events which share the same detector setup and physics conditions
  - A run consists of one loop over events

- Within a run, the user cannot change
  - the detector setup
  - settings of the physics processes

- As an analogy of the real experiment, a run of Geant4 starts with “Beam On”

- At the beginning of a run, the geometry is optimised for navigation and cross-section tables are calculated according to materials which appear in the geometry and with the defined cut-off values

- G4RunManager class manages the processing of a run, a run is represented by the G4Run class or a user-defined class derived from it

- G4UserRunAction is an optional user hook

Event in Geant4

- An event is the basic unit of simulation in Geant4
- At the beginning of processing, primary tracks are generated. These tracks are pushed into a stack
- A track is popped up from the stack one by one and is traced through the detector. Resulting secondary tracks, if any, are pushed into the stack
  - This “tracking” lasts as long as the stack has a track
- When the stack becomes empty, the processing of the event is over
- G4Event class represents an event. It has the 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 the processing of an event
- There is an optional user hook: G4UserEventAction

Track in Geant4

- A Track is a snapshot of a particle
  - It has physical quantities of the current instance only. It does not contain a record of previous quantities
  - A step is a “delta” information of a track. A track is not a collection of steps. Instead, a track is being updated by steps.

- The track object is deleted when
  - it goes out of the world volume,
  - it disappears (through decays, inelastic scattering, …),
  - it goes down to zero kinetic energy and no “AtRest” additional process is required for the particle, or
  - the user decides to kill it artificially.

- No Track object persists at the end of an event
  - For the record of tracks, use Trajectory class objects

- G4TrackingManager manages the processing of a track. A track is represented by the G4Track class

- There is an optional user hook: G4UserTrackingAction

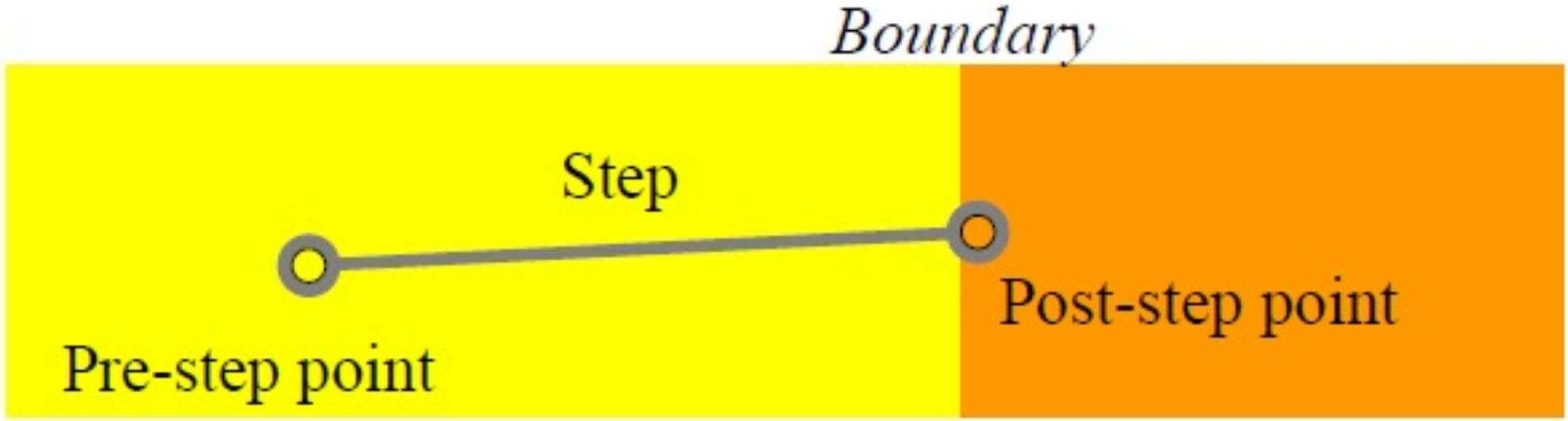
Step in Geant4

- A step has two points and also “delta” information of a particle (energy loss in the step, time-of-flight spent by the step, etc.)

- Each point knows the volume (and it's material) where it is in. In case a step is limited by a volume boundary, the endpoint will physically stand 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 reflection could be simulated

- G4SteppingManager class manages the processing of a step, and a step is represented by the G4Step class

- G4UserSteppingAction is the optional user hook



Trajectory and Trajectory Point

- Please remember, a track does not keep its trace and no track object persists at the end of an event

- G4Trajectory in the class which copies some of the information of a G4Track object. Likewise, G4TrajectoryPoint is the class which keeps some of the information of a G4Step object
  - A G4Trajectory object has a vector of G4TrajectoryPoint objects
  - At the end of event processing, the G4Event object has a collection of G4Trajectory objects provided
    - /tracking/storeTrajectory is set to 1

- Keep in mind the distinction:
  - G4Track vs G4Trajectory, G4Step vs G4TrajectoryPoint

- Given that the G4Trajectory and G4TrajectoriyPoint objects persist till the end of an event, one should be careful not to store too many trajectories
  - Avid for shower tracks from a high-energy particle

- G4Trajectory and G4TrajectoryPoint objects store only the minimum information
  - The user can create his/her own trajectory/trajectory-point classes to store the required information. These classes can be derived from the base classes G4VTrajectory and G4VTrajectoryPoint

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 unique G4DynamicParticle Object
    - This is a class representing an individual particle
  - G4ParticleDefinition:
    - “Static” properties of a particle, such as charge, mass, lifetime, decay channels, etc.
    - G4ProcessManager which describes the processes involving the particles
    - All G4DynamicParticle objects of the same kind of particles share the same G4ParticleDefinition

Tracking and Process

- Tracking in Geant4 is universal
  - It is independent of
    - the particle type
    - the physics processes involving the particle
  - It gives the chance to all processes
    - to contribute to the determination of the step length
    - to contribute any possible changes in physical quantities of the track
    - to generate secondary particles
    - to suggest changes in the state of the track
      - e.g. to suspend, postpone or kill it

Process in Geant4

- In Geant4, particle transportation is a process as well, by which a particle interacts with geometrical volume boundaries and fields of any kind
  - Because of this, the shower parametrization process can take over from ordinary transportation without modifying the transportation process

- Each particle has its own list of applicable processes. At each step, all processes involved are invoked to get proposed physical interaction lengths

- The process which requires the shortest interaction length (in space-time) limits the step

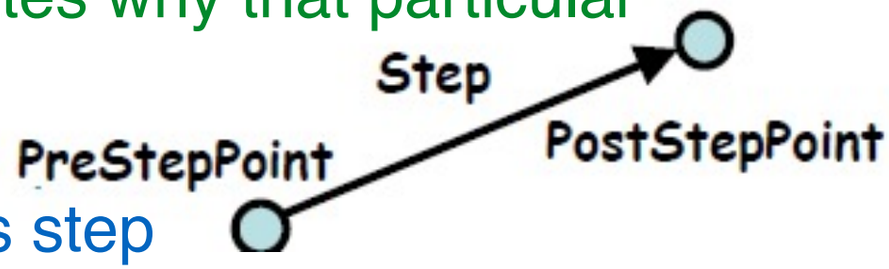
- Each process has one or combination of actions with the following nature:
  - At Rest
    - e.g. muons can decay at rest
  - Along Step (a.k.a. continuous process)
    - e.g. Cherenkov process
  - Post step (a.k.a. discrete process)
    - e.g. decay in flight

Track Status

- At the end of each step, the state of a track may change (according to the processes involved)
  - The user can also change the status in UserSteppingAction
  - Status, as mentioned below, are artificial, i.e. Geant4 kernel won't set them, but the user can
    - fAlive
      - continue the tracking
    - fStopButAlive
      - the track has come to zero kinetic energy, but still AtRest process to occur
    - fStopAbdKill
      - The track has lost its identity because it has decayed, interacted, or gone beyond the world boundary
      - Secondaries will be pushed to the stack
    - fKillAndSecondaries
      - kill the current track and also associated secondaries
    - fSuspend
      - suspend the processing of the current track and push it and its secondaries to the stack
    - fPostponeToNextEvent
      - Postpone processing of the current track to the next events
      - Secondaries are still being processed within the current event

Step Status

- The step status attached to the G4StepPoint indicates why that particular step was chosen
  - "PostStepPoint" gives the status of this step
  - "PreStepPoint" provides the status of the previous step



    - fWorldBoundary
      - step reached the world boundary
    - fGeomBoundary
      - step is limited by a volume boundary except for the world
    - fAtRestDoItProc, fAlongStepDoItProc, fPostStepDoItProc
      - step is limited by AtRest, AlonStep or PostStep process
    - fUserDefineLimit
      - step is limited by the user step limit
    - fExclusiveForcedProc
      - step is limited by an exclusively forced process (e.g. shower parametrisation)
    - fUndefined
      - step not defined
- If the first step in a volume is to be identified, pick fGeomBoundary status in the PreStepPoint
- If a step going out of a volume is to be identified, pick fGeomBounday status in the PostStepPoint

Extraction of Useful Information

- Given geometry, physics and primary track information, Geant4 does proper physics simulation “silently”
  - The user has to add a bit of code to extract useful information

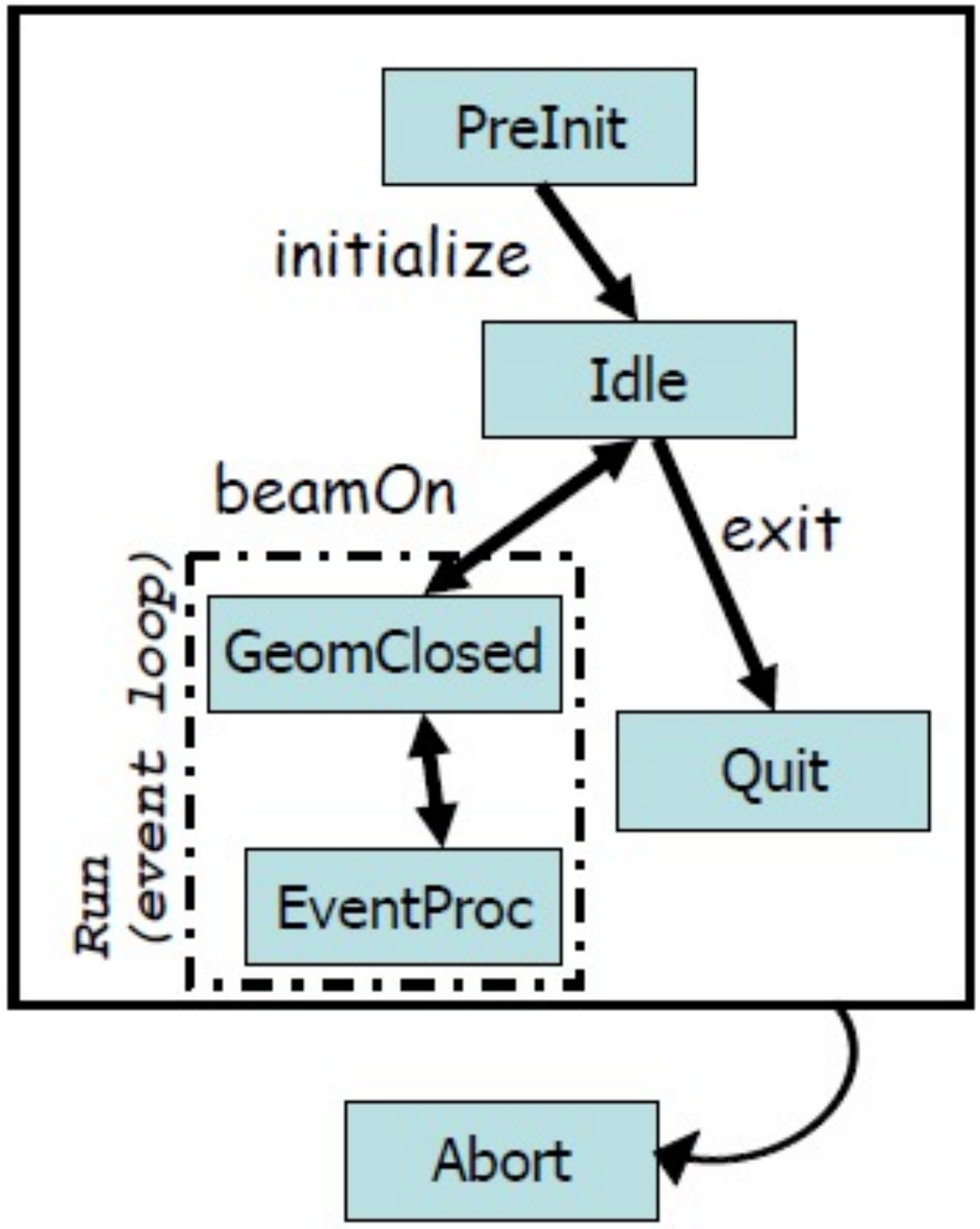
- There are two ways for extraction:

  - Use the user hooks provided by Geant4
    - These are: G4UserTrackingAction, G4UserSteppingAction, ….
      - The user has access to almost all information
      - It is straight-forward but do-it-yourself

  - Use Geant4 scoring functionality
    - Assign G4VSensitiveDetector to a volume
    - Hits collection is automatically stored in the G4Event object, and automatically accumulated if the user-defined Run object is used
    - Use user hooks to get event/run summary
      - The relevant action classes are G4UserEventAction, G4UserRunAction

Geant4 as a State Machine

- Geant4 has six application states:
  - G4State_PreInit
    - At this state, material, geometry, particle and physics process need to be defined and initialized
  - G4State_Idle
    - Geant4 is ready to start a run
  - G4State_GeomClosed
    - Geometry is optimised and ready to process an event
  - G4State_EventProc
    - An event is being processed
  - G4State_Quit
    - (Normal) termination
  - G4State_Abort
    - A fatal exception occurred and the program is aborting



Track Stacks in Geant4

- By default, Geant4 has three track stacks:
  - "Urgent", "Waiting" and "PostponeToNextEvent"
  - Each stack operates in a simple "last-in-first-out" mode
  - User can increase arbitrarily the number of stacks

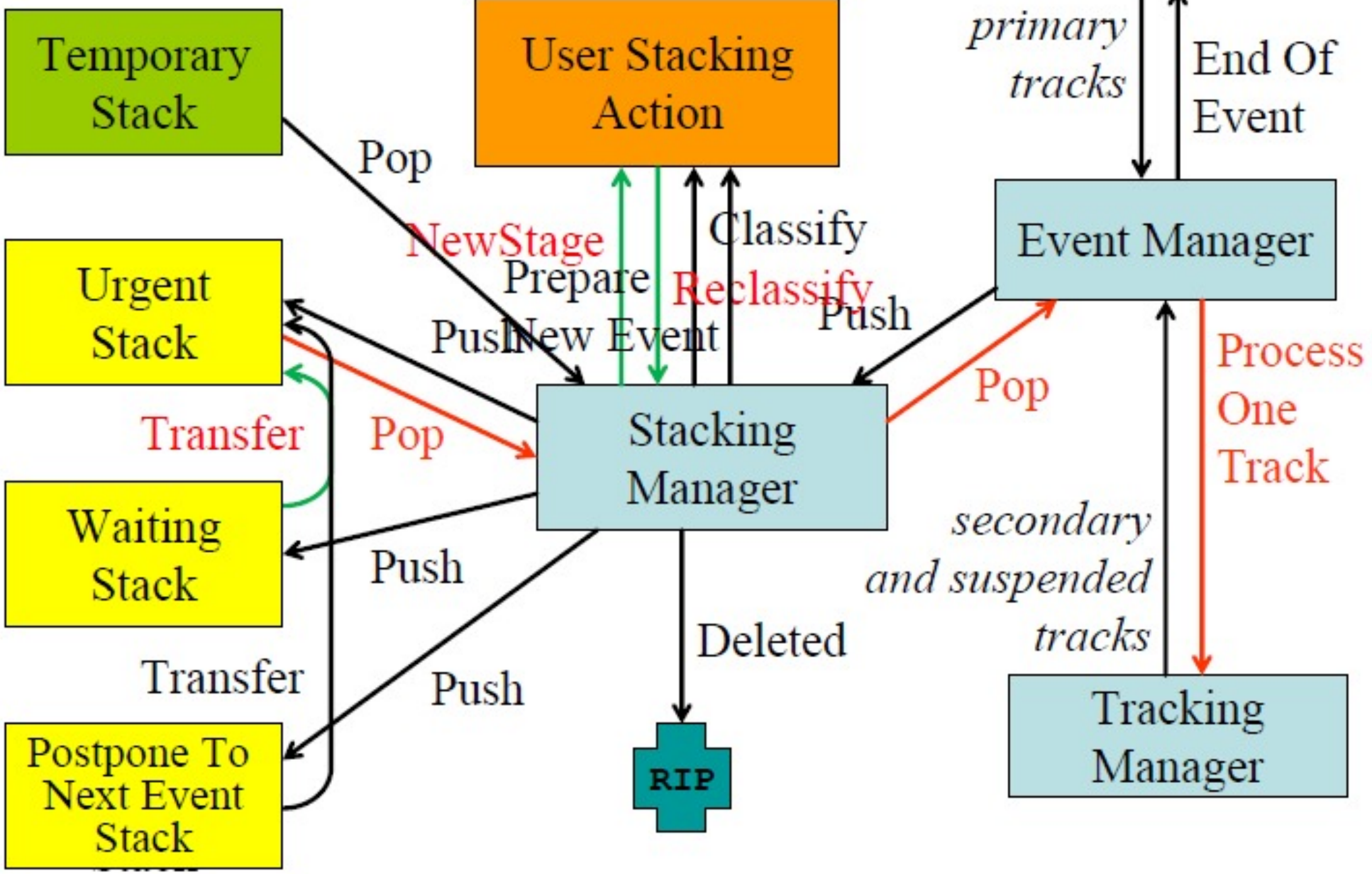
- ClassiftNewTrack() method of G4UserStackingAction class decides which stack each newly created secondary particle to be stored (or be killed)
  - By default, all tracks go to the "Urgent" stack

- A G4Track is popped up only from the "Urgent" stack

- Once the "Urgent" stack is empty, all tracks in the "Waiting" stack are transferred to the "Urgent" stack
  - And NewStage() method of th4 G4UserStackingAction is invoked

- Utilising more the one stack, the user can control the priorities of processing tracks without paying the overhead of "scanning the highest priority track"
  - Proper selection/abortion of tracks/events with well-designed stack management provides significant efficiency increase of the entire simulation

Stacking Mechanism



Tips of Stacking Manipulation

- Classify all secondaries as fWaiting until Reclassify() method is invoked
  - One can simulate all primaries before any secondary

- Classify secondary tracks below a certain energy as fWaiting until the Reclassify() method is invoked
  - One can roughly simulate the event before being bothered by low energy electromagnetic showers

- Suspend a track on its fly. Then this track and all of the already generated secondaries are pushed to the stack
  - Given the stack is “last-in-first-out”, secondaries are popped out prior to the original suspended track
  - This is quite effective for simulating Cerenkov radiation

- Suspend all tracks that are leaving a region, and classify these suspended tracks as fWaiting until Reclassify() method is invoked
  - One can simulate all tracks in this region prior to other regions
  - Note that some backsplash tracks may come back into this region later

Primary Generator

- Each Geant4 Event starts with the generation of one or multiple primary particles

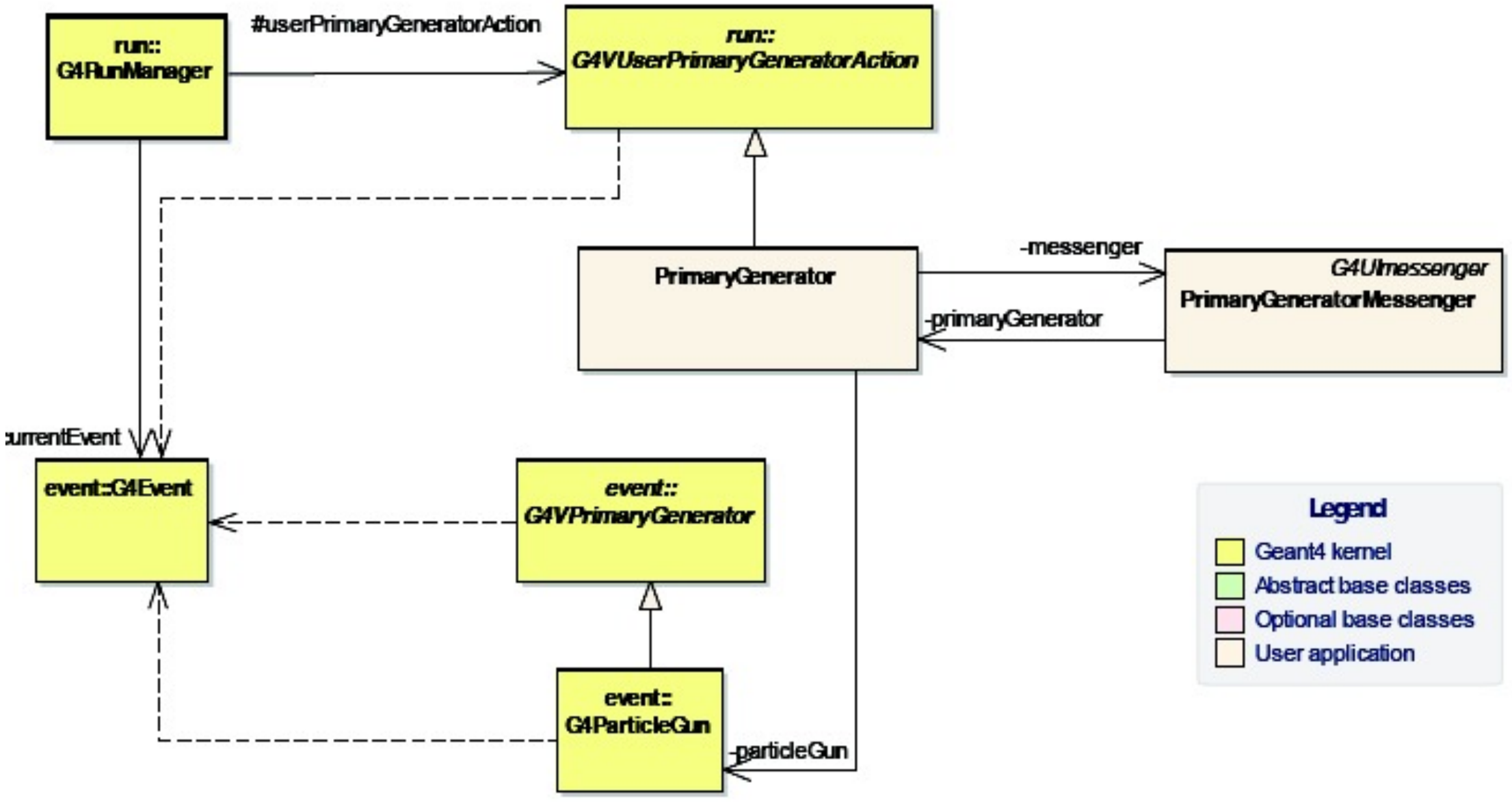
- The user has to define the properties of primary particles
  - Particle type, e.g. electron, gamma, ion, …
  - Initial kinematics, e.g. energy, momentum, origin, …
  - Additional properties, e.g. polarization, …

- These properties can be divided into:
  - G4PrimaryVertex: specifying start point in space and time
  - G4PrimaryParticle: specifying initial momentum, polarisation, PDG code, list of daughters for decay chains

- A primary generator is a class derived from G4VPrimaryGenerator and has an implementation of the method GeneratePrimaryVertex()
  - The primary vertex and the primary particle(s) are added in this method to a Geant4 Event
  - Several event generators are provided in the Geant4 toolkit
    - G4HEPEvtInterface, G4HEPMCInterface, G4GeneralParticleSoce, G4ParticleGun

PrimaryGeneratorAction

- This mandatory user action controls the generation of primary particles but does not generate the primaries itself. This task is delegated to G4PrimaryGenerator derived from G4VPrimaryGenerator



Attaching user information

- Abstract classes:
  - The user can use his/her own class derived from the period base class
  - G4Run, G4VHit, G4VDigit, G4VTrajectory, G4VTrajectoryPoint
- Concrete classes:
  - The user can attach a user information class object
    - G4Event — G4VUserEventInformation
    - G4Track — G4VUserTrackInformation
    - G4PrimaryVertex — G4VUserPrimaryVertexInformation
    - G4PrimaryParticle — G4VUserPrimaryParticleInformation
    - G4Region — G4VUserRegionInformation
  - User information class objects are deleted when the associated Geant4 class object is deleted

Bookkeeping Issues

- Connection from G4PrimaryParticle to G4Track
  G4int G4PrimaryParticle::GetTrackID()
  - Returns the track ID if this primary particle had been converted into G4Track, otherwise -1

- Connection from G4Track to G4PrimaryParticle
  G4PrimaryParticle* G4DynamicParticle::GetPrimaryParticle()
  - Returns the pointer of G4PrimaryParticle object if this track was defined as a primary or pre-assigned decay product, otherwise null

- G4VUserPrimaryVertexInformation, G4VUserPrimaryParticleInformation and G4VUserTrackInformation may be used for storing additional information
  - Information in G4VUserTrackInformation should be then copied to the user-defined trajectory class so that such information is kept until the end of the event

Additional Slides

Geant4 Application Software