User Actions

Jonathan R. Madsen

Department of Nuclear Engineering
Texas A&M University
College Station, TX, USA 77843

madsen_jr@tamu.edu

Outline

1. Overview
2. ActionInitialization
3. Run
4. Primary Generator
5. Event
6. Stacking
7. Tracking
8. Stepping

Overview

- Geant4 User Action classes
  - G4VUserActionInitialization [required]
  - G4VUserPrimaryGeneratorAction [required]
  - G4UserRunAction
  - G4UserEventAction
  - G4UserTrackingAction
  - G4UserSteppingAction
  - G4UserStackingAction
- User action classes provide polymorphic interfaces to insert customization to the Geant4 simulation
- The user action classes are used to setup and/or modify the simulation or collect information about the run

G4VUserActionInitialization

- Creates the user-actions for master and worker threads
- virtual void BuildForMaster() const
  - Create the user action objects for the master thread
  - Called once by master thread
  - Typically, you want at least an instance of G4UserRunAction
- virtual void Build() const [pure virtual]
  - Create the user action objects for the worker threads
  - Called once by each worker thread
- virtual G4VSteppingVerbose* InitializeSteppingVerbose() const
  - Create an instance of G4VSteppingVerbose

G4VUserActionInitialization (cont.)

- Protected member functions to add user action objects
  - void SetUserAction(G4VUserPrimaryGeneratorAction*)
  - void SetUserAction(G4UserRunAction*)
  - void SetUserAction(G4UserEventAction*)
  - void SetUserAction(G4UserStackingAction*)
  - void SetUserAction(G4UserTrackingAction*)
  - void SetUserAction(G4UserSteppingAction*)

G4UserRunAction

- virtual G4Run* GenerateRun()
  - This method is invoked at the beginning of BeamOn.
  - User hook to provide derived G4Run and create his/her own concrete class to store some information about the run
  - Ideal place to set variables which affect the physics table (such as production thresholds) for a particular run, because GenerateRun() is invoked before the calculation of the physics table.
- virtual void BeginOfRunAction(const G4Run*)
  - Invoked before entering the event loop
  - Typical use of this method would be to initialize and/or book histograms for a particular run
  - This method is invoked after the calculation of the physics tables

G4UserRunAction (cont.)

- virtual void EndOfRunAction(const G4Run*)
  - This method is invoked at the very end of the run processing
  - It is typically used for a simple analysis of the processed run
- virtual void SetMaster(G4bool val=true)
- G4bool IsMaster()
  - Commonly, a MT simulation will have a master-thread instance and a worker thread instance — provides ability to discern whether instance is for worker or master thread

G4VUserPrimaryGeneratorAction

- Provides user hook to set up the initial particles driving an event
- virtual void GeneratePrimaries(G4Event*) [pure virtual]
  - Invoked prior to G4UserEventAction::BeginOfEventAction()
  - Responsible for creating the primary particles in the event
  - Typically uses one or more G4ParticleGun objects to generate the primary vertex

G4UserEventAction

- virtual void BeginOfEventAction(const G4Event*)
  - This method is invoked before converting the primary particles to G4Track objects
  - A typical use of this method would be to initialize and/or book histograms for a particular event
- virtual void EndOfEventAction(const G4Event*)
  - This method is invoked at the very end of event processing
  - Typically used for a simple analysis of the processed event
  - If the user wants to keep the currently processing event until the end of the current run, the user can invoke G4EventManager::GetEventManager()->KeepTheCurrentEvent() so that it is kept in G4Run object.

G4UserStackingAction

- G4UserStackingAction is a user-hook to reorder the priority of the particle stack
- virtual G4ClassificationOfNewTrack ClassifyNewTrack(const G4Track*)
  - invoked by G4StackManager whenever a new G4Track object is "pushed" onto a stack by G4EventManager
  - Returns an enumerator whose value indicates to which stack the track should be sent. Value is determined by the user from four possible values
    - fUrgent — track is placed in urgent stack
    - fWaiting — track is placed in the waiting stack (when urgent is empty)
    - fPostpose — track is postponed to next event
    - fKill — track is deleted immediately and not stored

G4UserStackingAction (cont.)

- virtual void NewStage()
  - Invoked when the urgent stack is empty and the waiting stack contains at least one G4Track object
  - User may kill or re-assign to different stacks all the tracks in the waiting stack [G4StackManager::ReClassify()]
  - If no user action is taken, all tracks in the waiting stack are transferred to the urgent stack
  - The user may can decide to abort the current event here
- virtual void PrepareNewEvent()
  - Invoked at the beginning of each event
  - At this point no primary particles have been converted to tracks, so the urgent and waiting stacks are empty
  - However, there may be tracks in the postponed-to-next-event stack; for each of these the ClassifyNewTrack() method is called and the track is assigned to the appropriate stack

G4UserTrackingAction

- Provides user hooks to access a particle track at the beginning and end of the particle's lifetime
- virtual void BeginOfTrackingAction(const G4Track*)
  - Invoked at the beginning of a particle's lifetime (creation)
- virtual void EndOfTrackingAction(const G4Track*)
  - Invoked at the end of a particles lifetime
  - End of particle's lifetime can occur from
    - Zero kinetic energy
    - Track is explicitly killed (fStopAndKill, fKillTrackAndSecondaries)
    - Particle leaves the "world"

G4UserSteppingAction

- Provides user hook to a particle step
- virtual void UserSteppingAction(const G4Step*)
  - Invoked after a particle has undergone a "step"
  - A step can be defined by
    - Undergoing physical process (*e.g.* ionization, decay)
    - Transport step to boundary
    - ... etc.
  - Typically used for custom scoring that is not supported by primitive scorers
  - The most frequently called user hook
  - Special attention must be paid to thread-safety when custom scoring is done here