Geometrical Event Biasing Facility

1. Geometrical Event Biasing
2. Space User experience
3. “Smart” Biasing Facility
4. Discussion

Alex Howard ETH, Zurich

Geometrical Event Biasing Facility

Geant4 Collaboration Meeting, Chartres

Geometric Biasing

The purpose of geometry based event biasing is to save computing time* by sampling less often the particle histories entering "less important" geometry regions, and more often in more "important" regions.

- Importance sampling technique
- Weight window technique

*** But what about development time?**
(including debugging/doing something crazy)

Comments:

- Importance based biasing is implemented without touching the physics list (Advantage)
- Requires a connection between parallel and mass geometries (i.e. you'd like to bias a region of interest which is related to how much material is present in the mass world)
  - Gets complicated with user RunManager (GRAS) and UI driven geometries (care with order and implementation)
- Previously most uses cases were concentrated on shielding studies with the biasing of neutrons
- Since 2007 geometrical biasing has been possible for any particle (electrons, γ's…)
- Common use case is electronic component shielding with complex geometry (also GDML)

Experience from Space/ELSHIELD:

- Request for simple UI to allow non-experts to use geometrical biasing
  - Especially engineers looking at shielding (EM and electronic components for example)
- How to make it smart (i.e. limited commands), reliable and accurate?

Space User Requirements

General purpose biasing applied at the UI level

MULASSIS (electronic shielding) and GRAS (general purpose space application) are examples

Engineering-style tools

- Self-checking statistics/variance reduction?
- Attach to mass geometry/region of interest
- Auto generate interfaces (Geometrical Biasing)
- UI selectable (change parameters, switch on/off)

As different tools/frameworks would be better to have a common solution within Geant4

GRAS Interface

The minimal (GRAS-based) interface the following interactive commands:

/gras/biasdet/geom_biasing → switching geometrical importance biasing off or on

/gras/biasdet/geometry → Choose importance sampling geometry: cylindrical, spherical or square

/gras/biasdet/particle → Choose particle for biasing: e+, e-, gamma, neutron, proton

/gras/biasdet/spacing → Choose importance sampling spacing: auto or custom

/gras/biasdet/length → Length of biasing geometry from start

/gras/biasdet/width → Radius of biasing geometry (cylindrical), side (square)

/gras/biasdet/add_interface → Add an importance sampling interface at a given point (linear distance from start of biasing)

/gras/biasdet/weight → Choose weight option: auto (~2), global (set value), custom (individual values)

/gras/biasdet/start → Set start of the biasing geometry

/gras/biasdet/update → Update geometry. Command MUST be applied before beamOn.

- Allows the user to apply an importance geometry on top of an existing application
- Number of interfaces and weights are automatically assigned within the user requested envelope.
- Only one particle can be biased, but the selection can be for electrons, photons or neutrons.
- The user is responsible for coinciding the bias geometry with the mass geometry and region of interest.

Biasing in Mulassis and SPENVIS

- "Simple" interface introduced into MULASSIS where commands are:
  - Biasing on/off
    - /gras/geom_biasing/switch true
      - Automatically sub-divides the mass geometry into 10 layers with bias weight of 2 per layer (splitting/russian roulette)
  - Particle e+,e-,gamma,neutron etc…
    - /gras/geom_biasing/particle e-
  - (if required) Granularity
    - /gras/geom_biasing/granularity fine
      - Doubles the number bias layers
- More "Advanced" interface is hidden
- Automatic biasing better placed within Geant4?

Smart Biasing Solution?

- Easy solution: Create parallel geometry attached to region of interest/mass geometry?
- Important biasing activated through simple UI commands
  - Geant4 takes care of matching the mass geoemetry (co-ordinate and envelope coincidence)
  - User has option to define directionality, granularity and weight settings
- Aside: Could we have a common interface/implementation with Reverse Monte Carlo?

Automatic Biasing

- Division of mass geometry into importance "slabs" follows a prescribed procedure:
  1. Decide on region of interest
  2. Direction from source to sensitive detector
  3. Particle of interest
  4. Divide up geometry "sensibly"
  5. Apply power law of splitting weights along direction of interest
- Choice of division depends on interaction length of source particle
- Could Geant4 do the job for the user?

Automatic Biasing Requirements

- Ability to "see" the mass geometry (or user defines start and end?)
- Calculate mean interaction length between source and target (SD)
- Create parallel geometry at the correct points
- Apply "sensible" weights
- Test result? Optimisation? Convergence testing?

Automatic Biasing - Advantages

- User doesn't have to worry about coincidence between mass geometry and parallel (biased) one
  - Removes co-ordinate transformations issues (e.g. GDML)
- User doesn't need to calculate what is sensible w.r.t. interaction length or energy and particle type
  - Removes over-biasing and creating billions of particles
- Biasing implementation is simple / hidden
  - Suitable for all users, not just advanced/developers
- Would increase usage of biasing and the inherent performance advantages
- Results more accurate/reliable
  - No garbage
  - Increases stability/removes crashes

Is it possible? / Which "helper" classes?

- How much would the user have to define?
  - E.g. GRAS vs. MULASSIS solutions
  - Start and end co-ordinates with direction?
- Can Geant4 be smart?
  - Take information from particle gun and sensitive detector?
  - Get cross-section table from physics list
- Can the biasing process be smart?
  - The instantiation, initialisation and implementation are crucial
    - Had some difficulties with GRAS (custom RunManager, detector constructor etc….)
    - Things must occur at the right time
- How to check the result?
  - Convergence testing? Sub-running/optimisation?
- Is it worth the effort? Users? Advanced knowledge?