Biasing in Geant4

ENSAR2 Workshop
24 - 26 April 2019
CIEMAT Madrid, Spain

Marc Verderi
LLR, Ecole Poytechnique

Overview

I. Introduction

II. Existing Biasing Options before v10.0
   – Primary Particle Biasing
   – Options In Hadronic
   – Geometry based importance biasing
   – Weight Window Technique
   – Reverse Monte Carlo
   – User defined biasing

III. Generic biasing (since v10.0)

I. Introduction

Rare events

› There are simulation problems in which what we are interesting in is "*rare*"
  – rare because of the physics,
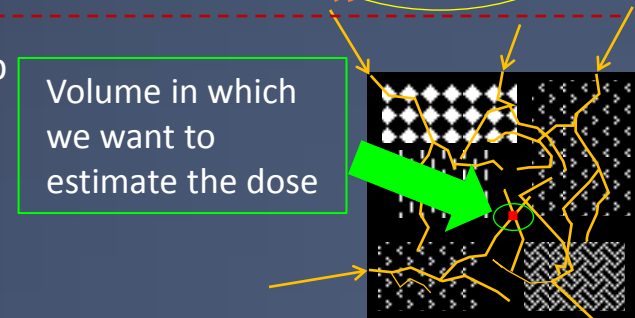 – or because of the setup,
  – Etc.

› Examples of such problems:
  – Estimating the efficiency of a shield
    › A large flux enters a shield…
    › … with a lot of interactions inside, which is CPU intensive
    › … and only a tiny fraction of particles exiting : the rare ones.
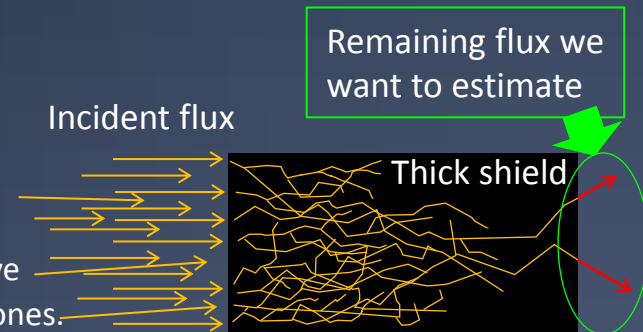
  – Obtaining the response of a very thin detector
    › For example to obtain the response to neutrons
    › Most of them do not interact in
    › But the signal is made by the rare ones which interact
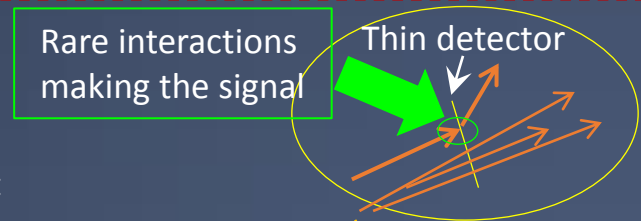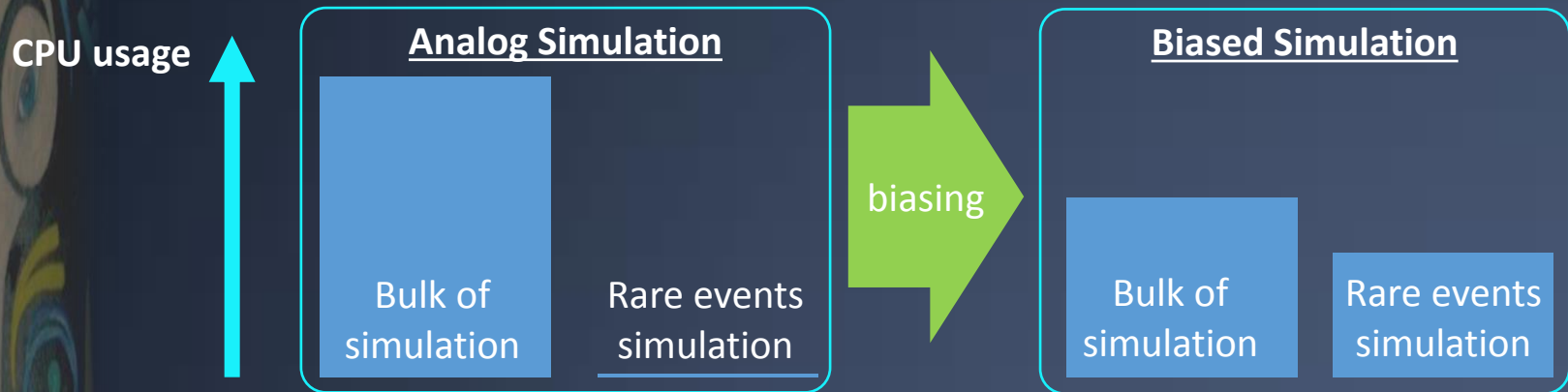
  – Estimating the dose in a very small part of a big setup
    › For example an electronic chip inside a satellite
    › Most of the time is spent simulating showers that do not contribute the dose
    › While we want to tally this dose

› An "analog" (usual) simulation is very inefficient in addressing these problems.

Remaining flux we want to estimate

Incident flux

Thick shield

Rare interactions making the signal

Thin detector

Volume in which we want to estimate the dose

What is "event biasing" ?

› "Event biasing" (or "biasing" or "variance reduction") is a set of techniques to simulate rare events efficiently
  - It is an accelerated simulation with respect to these events
    › And to these events only : ie not everywhere !
  - It focuses/transfers the CPU power usage on what we want to tally

CPU usage ↑

Analog Simulation		biasing →	Biased Simulation	
Bulk of simulation	Rare events simulation		Bulk of simulation	Rare events simulation

  - It can provide LARGE CPU improvements
    › Several orders of magnitude ! Depending on the problem.

› "Biasing" stands for "biased simulation":
  - "Biased simulation" because rare events are enhanced compared to the analog simulation.

Main principles

› Enhancement of rare events is tracked with statistical "weights":

– Weights are used to "de-bias" and go back to analog quantities:

› Eg : $E_{deposit} = \sum_{i=track} weight_i \times E^i_{deposit}$

Track i, weight i

› There is a large variety of techniques (sometimes with ≠ names for ≈ techniques) but two methods make the "backbone" of most of them:
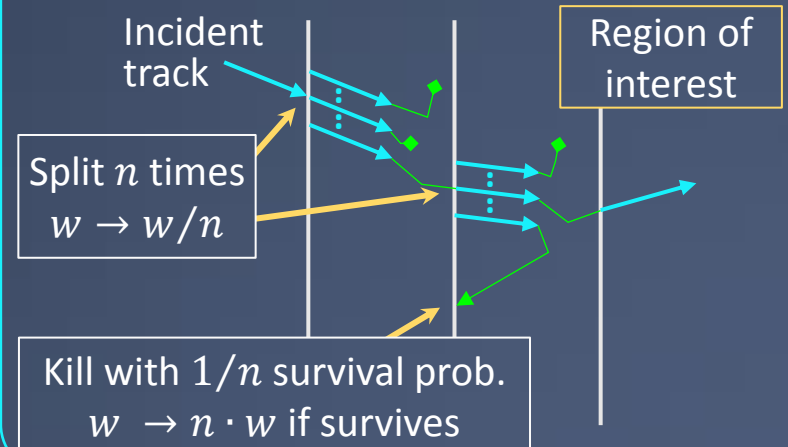
Importance Sampling

• Modify physics pdf's enhancing "important" parts of them

$analog\_pdf(x)$

Region of interest

x

$biased\_pdf(x)$

x

$w(x) = \dfrac{analog\_pdf(x)}{biased\_pdf(x)}$
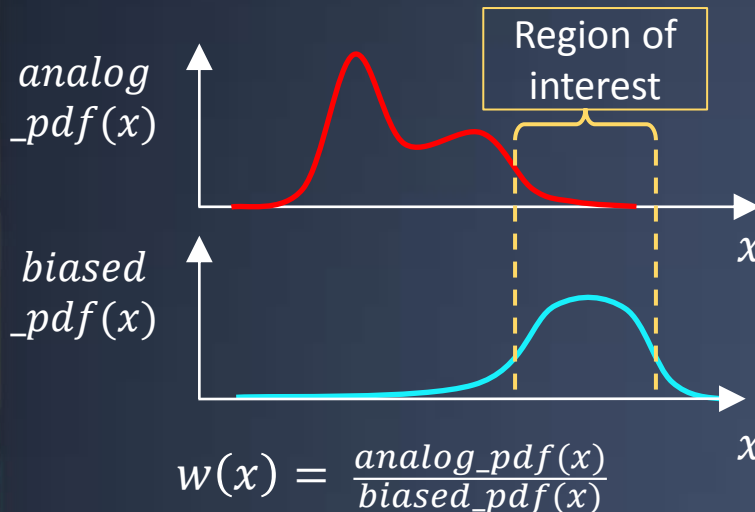
Splitting (& Killing)

• Keep pdf's unchanged but split (kill) when moving towards (receding from) region of interest

Incident track

Region of interest

Split n times
$w \to w/n$

Kill with $1/n$ survival prob.
$w \to n \cdot w$ if survives

Biasing versus other acceleration techniques ?

› Other acceleration techniques exist:
  - "fast simulation" : replacement of detailed simulation by approximate but faster model (eg: EM showers)
  - Deterministic computation (in some cases)

› Interest of biasing:
  - When using biasing, the physics modeling is the same than the detailed simulation one
    › Biasing is "simply" an other way to process the full detailed physics
  - So, results obtained with biasing are statistically the same than the ones obtained with a large/huge processing of standard simulation

› Difficulties with biasing:
  - Only "rare events" problems can be treated this way
  - Ensuring a proper convergence can be difficult
    › Issue of random appearance of very large weights
      - that destroy the convergence
    › This signs that parts of the problem are not sampled enough (illustration in backup slides)

II. Existing Biasing Options before v10.0

Primary Particle Biasing

› The General Particle Source (GPS) is a tool to generate primary particles
  - This generation can be done according to many features
  - Energy, angular and position distributions
  - Combine several sources with relative intensities
  - Make "beam-like" source
  - Etc…

› GPS allows to bias the particle source:
  - Position
  - Angular Distribution
  - Energy Distribution

› This is an "importance sampling" biasing

Options In Hadronic

› Cross-section Biasing:
  – Possibility to enhance a (low) cross-section
  – Available for photon inelastic, electron & positron nuclear processes

› Leading Particle Biasing:
  – It is a technique used for estimating penetration of particles in a shield
  – Instead of simulating the full shower, at each interaction, only keep:
    › The most energetic particle
    › + randomly one particle of each species
  – This is a « killing » - based biasing

› Radioactive decay
  – Enhance particular decay within a given time window
  – Sample all decay modes with equal probabilities
  – Split parent nuclide in a user defined number of copies, letting the decay go
  – Enhance emission in a particular direction
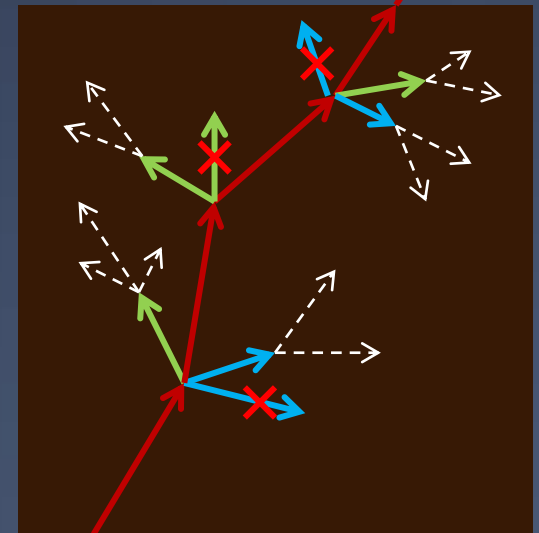
Geometry based importance biasing

› A direct application of what was presented earlier

› Attach "importance" to cells in geometry
  – Not to be confused with "importance sampling" : ie change of probability density functions of interaction laws…
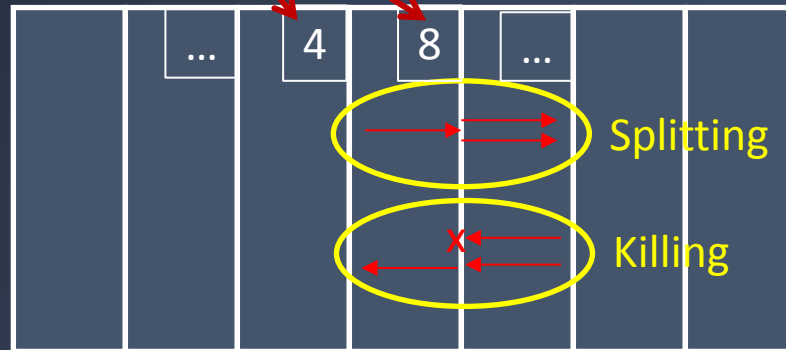

  – Applies splitting if the track moves forward
    › with splitting factor 8/4 = 2, if track goes from « 4 » to « 8 » (eg, here)
    › each copy having a weight = ½ of the incoming track
  – Applies killing if the track moves the other way
    › it is killed with a probability ½
    › If particle survives, its weight is multiplied by 2 (eg, here)

Example B01 - 10 MeV neutrons, thick concrete cylinder

Analogue Simulation
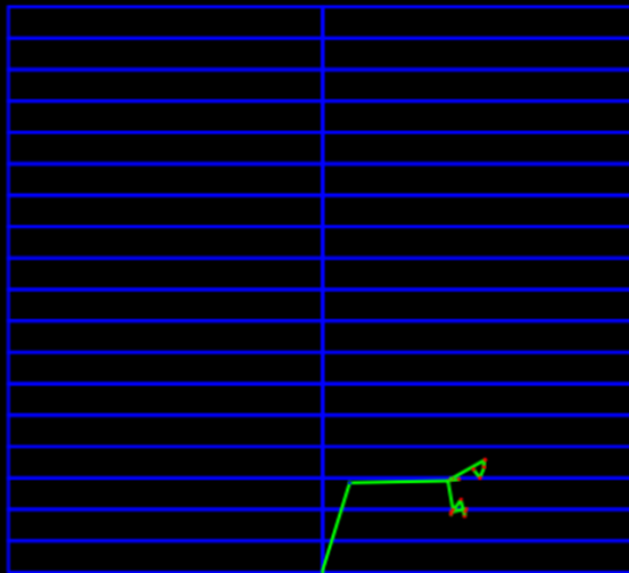
Importance Sampled

2^n

8
4
2
1
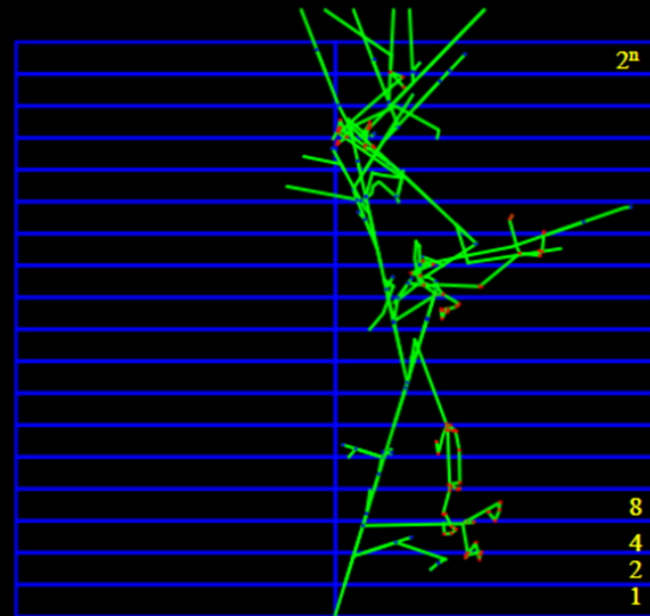
Neutron source

See: geant4/examples/extended/biasing/B01

Weight Window Technique

› This is a « splitting & killing » technique, to avoid having
  - Too high weight tracks at some point
    › As this makes the convergence of the estimated quantities difficult
    › Tracks with weight above some value are split
  - Too low weight tracks
    › As these are essentially a waste of time
    › Tracks below some value are "Russian roulette" - killed
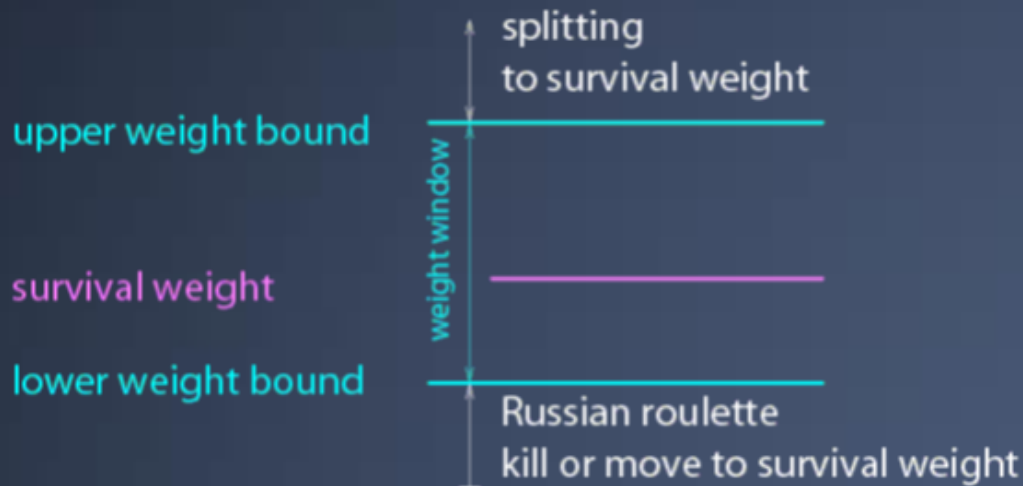
› As with importance, this is configured per cell
  - And can be configured per energy
› See: geant4/examples/extended/biasing/B01

Reverse Monte Carlo

› Simulation in which particles go back in time !

› Useful in case of small device to be simulated in a big environment
  – Requested by ESA, for dose study in electronic chips in satellites

› Simulation starts with the red track
  – So-called "adjoint" track
  – Going back in time
  – Increasing in energy
  – Up to reaching the extended source

Extended source (Sky)

Adjoint track

–
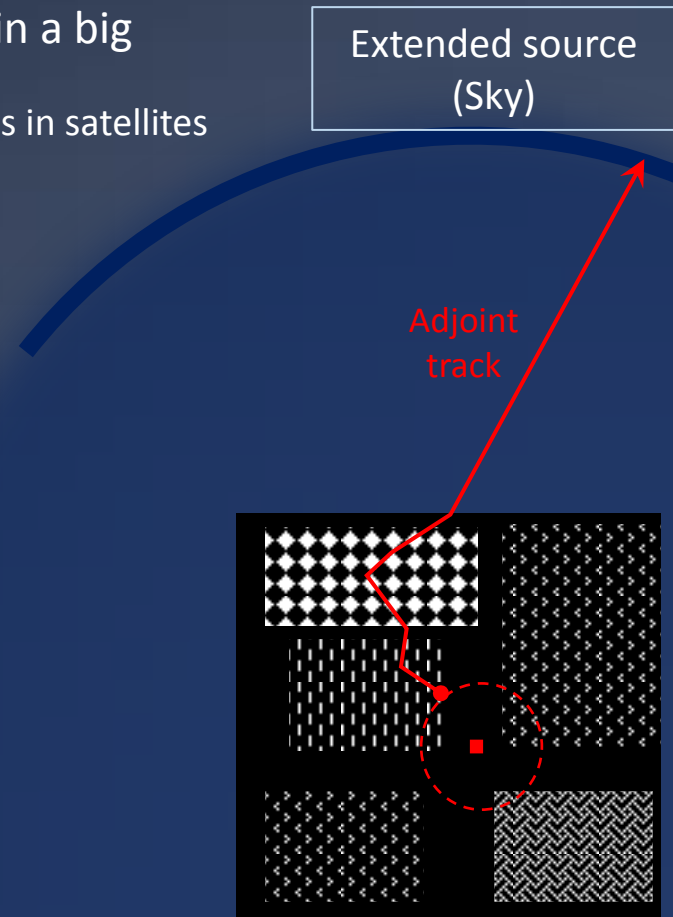
Reverse Monte Carlo

› Simulation in which particles go back in time !

› Useful in case of small device to be simulated in a big environment
  – Requested by ESA, for dose study in electronic chips in satellites

› Simulation starts with the red track
  – So-called "adjoint" track
  – Going back in time
  – Increasing in energy
  – Up to reaching the extended source

› If track energy < high energy limit of the sky source spectrum
  – Proceed with a usual "forward" simulation (green track)

› When many track simulated, adjust adjoint track weights so that energy spectrum of adjoint particle matches sky source spectrum
  – This provides the weight of the green tracks

› Dose in chip can then be computed, accounting for the forward track contribution, with the proper weight.

› See: geant4/examples/extended/biasing/ReverseMC01

Extended source (Sky)

Adjoint track

Forward (normal) track
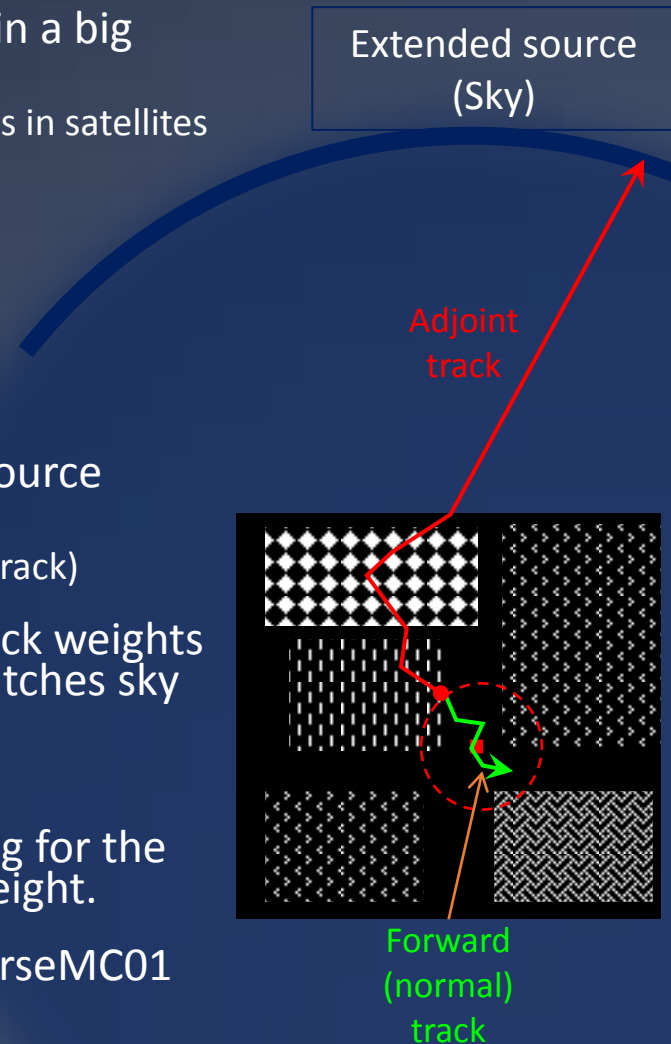
User defined biasing

› G4WrapperProcess can be used to implement user defined event biasing

› G4WrapperProcess, which is a process itself, wraps an existing process

› All function calls forwarded to wrapped process

› Needed steps:
  1. Create derived class inheriting from G4WrapperProcess
  2. Override only the methods to be modified, e.g., PostStepDoIt()
  3. Register this class in place of the original
  4. Finally, register the original (wrapped) process with user class

Example with brem. splitting

› Bremstralhung splitting is a technique used for example in medical applications:
  – Radio-therapy with photon beam generated by bremstralhung
  – Interest is in generated photons, not in the electron → enhance photon production
› The bremstralhung process is repeated N times, with the same initial electron
  – All bremstralhung photons are given a weight 1/N
 – The electron continues with one of the state among the N ones generated

```cpp
G4VParticleChange* myWrappedProc::PostStepDoIt(const G4Track& track, const G4Step& step)
{

    G4double weight = track.GetWeight()/fNSplit;

    std::vector<G4Track*> secondaries(fNSplit);

    // Secondary store
    // Loop over wrapped PSDI method to generate multiple secondaries
    for (G4int i=0; i<fNSplit; i++)
        {
            particleChange = pRegProcess->PostStepDoIt(track, step);
            assert (0 != particleChange);
            particleChange->SetVerboseLevel(0);
            // Save the secondaries generated on this cycle
            for (G4int j=0; j<particleChange->GetNumberOfSecondaries(); j++)
                {
                    secondaries.push_back (new
                                           G4Track(*(particleChange->GetSecondary(j))));
                }
        }
        // -- and then pass these secondaries to the particle change…
}
```

Brem. Process in this case

Example with a brem. splitting

Uniform Bremsstrahlung Splitting

No splitting

Splitting factor = 100

Scoring Geometry
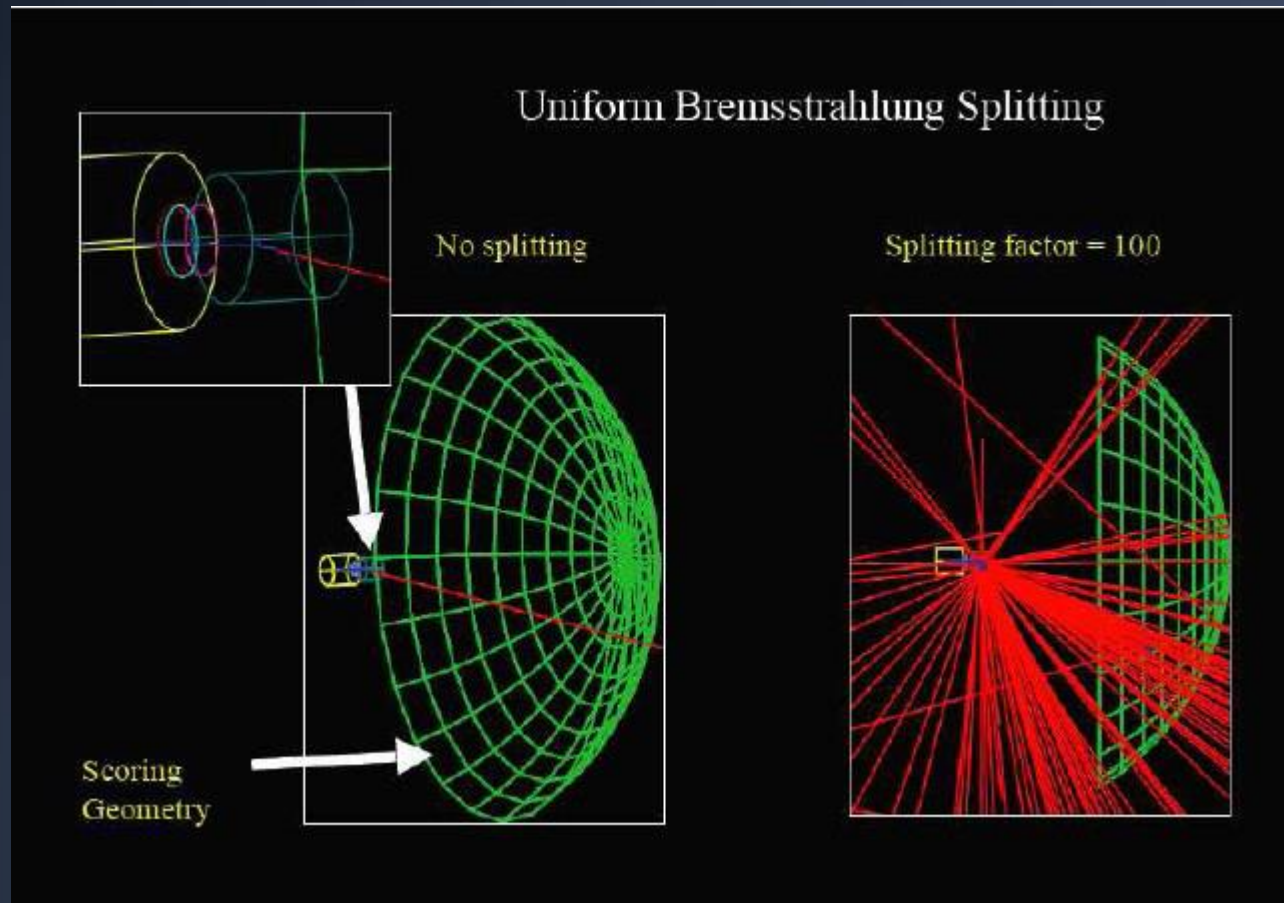
› Note: Since 9.6, a built-in brem. splitting option is now proposed.

III. Generic Biasing (since v10.0)

Why and what generic scheme ?

› Many options existed
  – But lived next to each other
    › Each introduced its own infrastructure
    › Making uneasy the combination of them
  – No general scheme to modify the physics process behavior
    › Interaction probability
    › Final state generation
    › While it is a the start of many biasing techniques
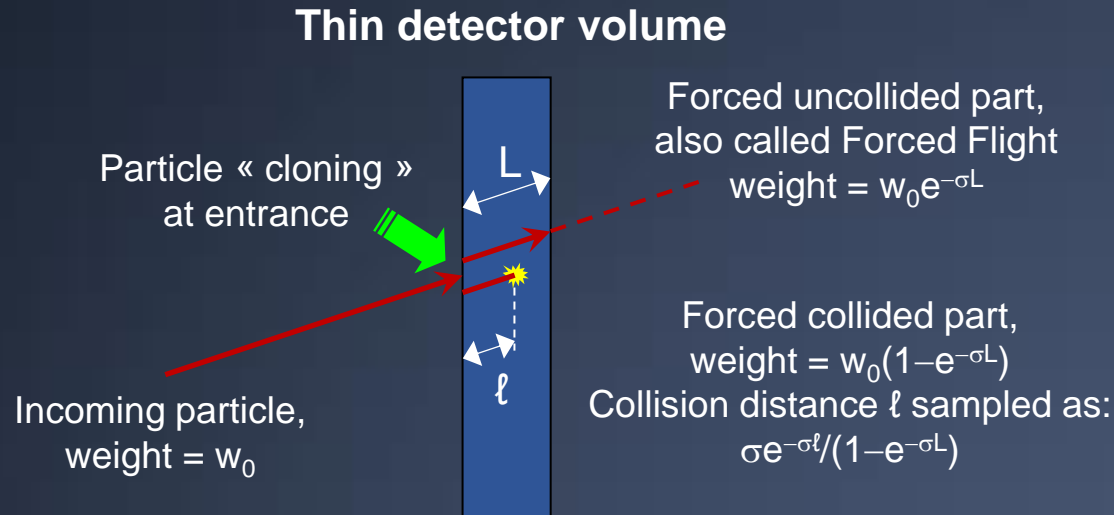
› Revise by adopting a classical "object oriented" (OO) approach:
  – An abstract layer, general enough
    › With concrete implementations deriving from it
    › Making, as anywhere in Geant4, this abstract layer open to users
  – Request this layer to handle physics process biasing + splitting and killing

› Try to make:
  – A "modular" scheme, with re-usable components
  – The user/developer having a maximum of information to make decisions

› Hoping this will favor users' creativity

Modularity

› Example of "force collision" scheme à la MCNP

Thin detector volume

Particle « cloning » at entrance

Incoming particle, weight = w_0

L

ℓ

Forced uncollided part, also called Forced Flight
weight = $w_0 e^{-\sigma L}$

Forced collided part,
weight = $w_0(1-e^{-\sigma L})$
Collision distance ℓ sampled as:
$\sigma e^{-\sigma \ell}/(1-e^{-\sigma L})$

› Approach considered:
  – Instead of providing the above scheme in one go
  – Consider "atomic" "biasing operations"
    › splitting, forced free flight, forced interaction, etc.
  – Compose these with a "biasing operator"
    › that selects, step after step, the biasing operations to build the needed sequence
    › which is interfaced with the tracking to have these operations applied

Design Ingredients

› G4VBiasingOperation

- An abstract class
- Can act on a physics process:
  - › By modifying its interaction law
  - › By modifying its final state generation
- Can act by itself
  - › To split or kill particles
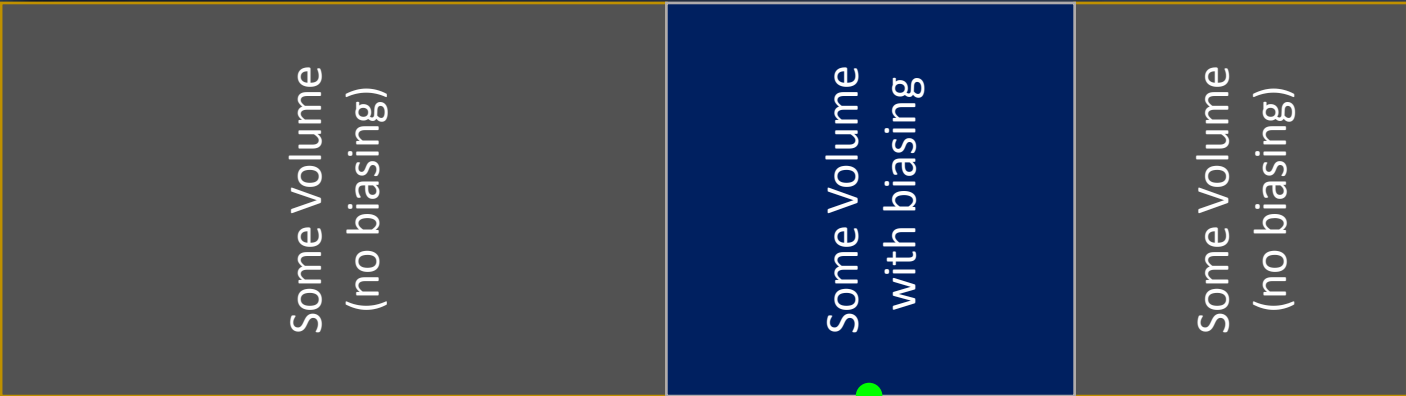
› G4VBiasingOperator

- An abstract class
- Selects G4VBiasingOperation's
  - › At the beginning of the step
  - › At the final state generation level

› G4BiasingProcessInterface

- A concrete class
- Makes the interface between the tracking and G4VBiasingOperator's
- It can:
  - › Wrap a physics process : to let G4VBiasingOperation's modifying it
  - › Wrap no process : to let G4VBiasingOperation's to do splitting and killing

G4BiasingProcessInterface

G4GammaConversion

Some Volume (no biasing)

Some Volume with biasing

Some Volume (no biasing)

G4BiasingProcessInterface

(no process)

G4BiasingProcessInterface

G4PhotoelectricEffect

G4BiasingProcessInterface

G4GammaConversion

G4BiasingProcessInterface

G4ComptonScattering
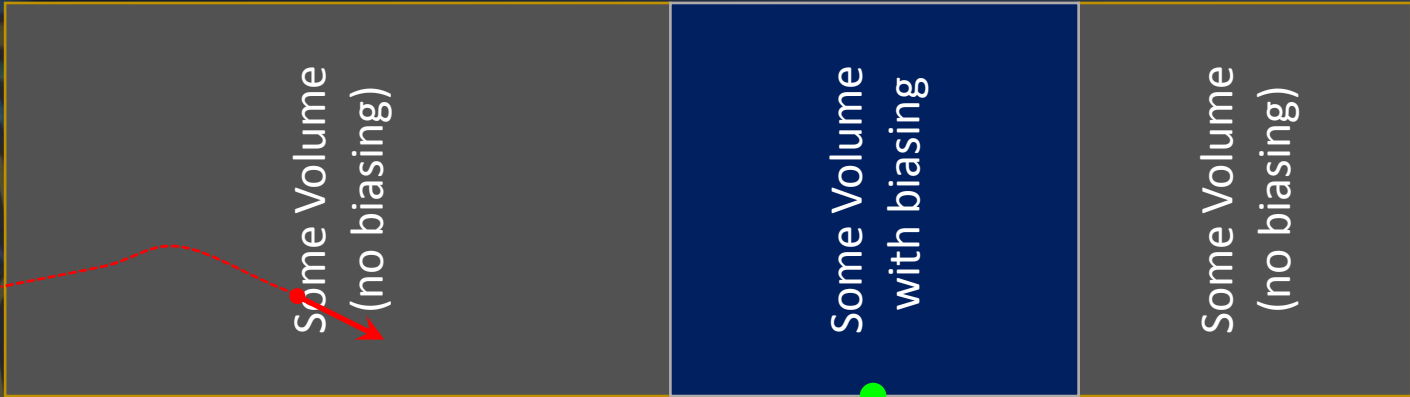
MyBiasingOperator

G4FlatForce-InteractionOperation

G4ExponentialForce-InteractionOperation

G4ForceFreeFlight-InteractionOperation

G4SplittingOperation

G4WeightWindow-Operation

G4MCNPForce-CollisionOperator

Some Volume (no biasing)

Some Volume with biasing

Some Volume (no biasing)

G4BiasingProcessInterface

(no process)

G4BiasingProcessInterface

G4PhotoelectricEffect

G4BiasingProcessInterface

G4GammaConversion

G4BiasingProcessInterface

G4ComptonScattering

MyBiasingOperator

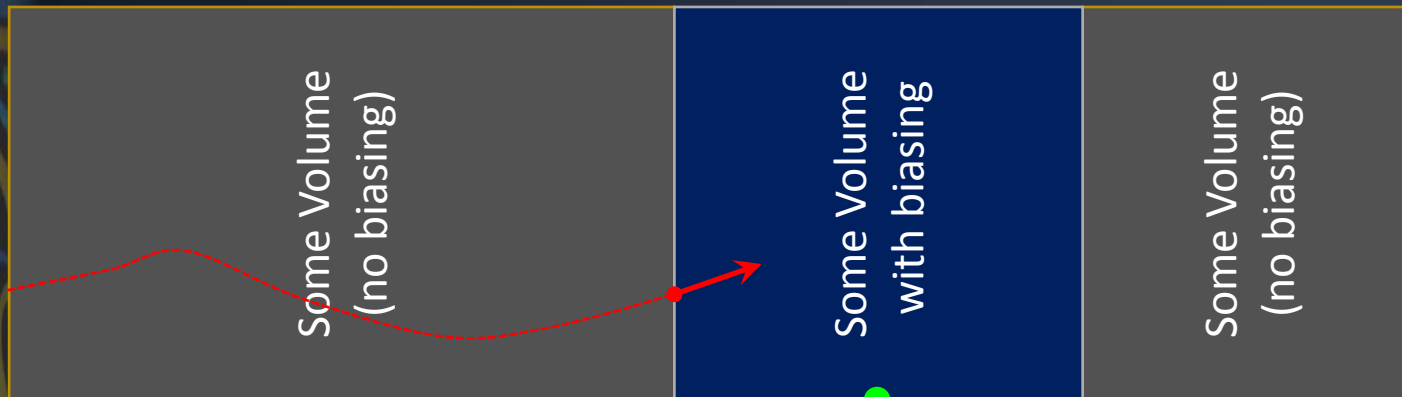G4FlatForce-InteractionOperation

G4ExponentialForce-InteractionOperation

G4ForceFreeFlight-InteractionOperation

G4SplittingOperation

G4WeightWindow-Operation

G4MCNPForce-CollisionOperator

Some Volume (no biasing)

Some Volume with biasing

Some Volume (no biasing)

G4BiasingProcessInterface

(no process)

G4BiasingProcessInterface

G4PhotoelectricEffect

G4BiasingProcessInterface

G4GammaConversion

G4BiasingProcessInterface

G4ComptonScattering

MyBiasingOperator

G4FlatForce-InteractionOperation

G4ExponentialForce-InteractionOperation

G4ForceFreeFlight-InteractionOperation

G4SplittingOperation

G4WeightWindow-Operation

G4MCNPForce-CollisionOperator

Some Volume (no biasing)

Some Volume with biasing

Some Volume (no biasing)

G4BiasingProcessInterface

(no process)

G4BiasingProcessInterface

G4PhotoelectricEffect

G4BiasingProcessInterface

G4GammaConversion

G4BiasingProcessInterface

G4ComptonScattering
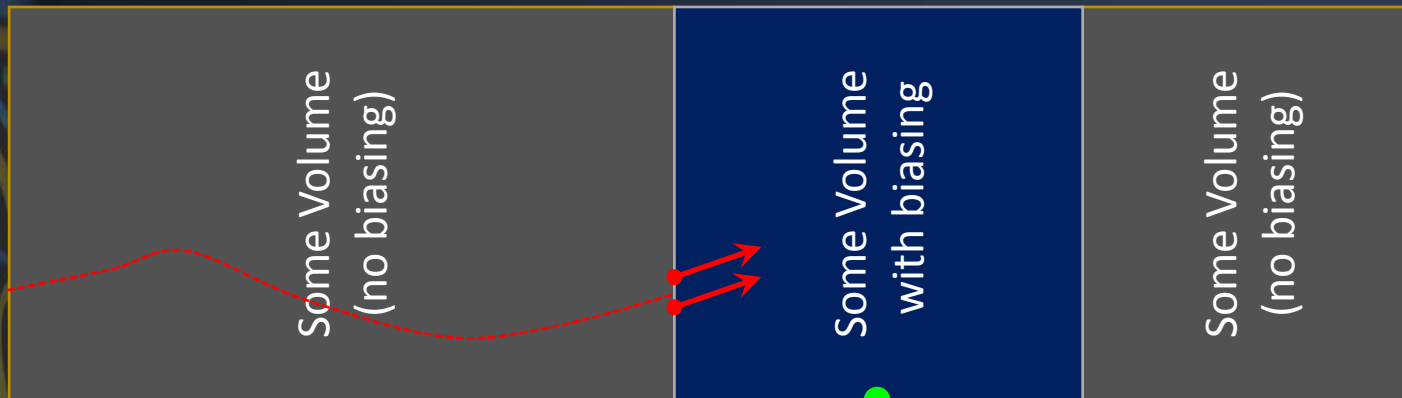
MyBiasingOperator

G4FlatForce-InteractionOperation

G4ExponentialForce-InteractionOperation

G4ForceFreeFlight-InteractionOperation

G4SplittingOperation

G4WeightWindow-Operation

G4MCNPForce-CollisionOperator

Some Volume (no biasing)

Some Volume with biasing

Some Volume (no biasing)

G4BiasingProcessInterface

(no process)

G4BiasingProcessInterface

G4PhotoelectricEffect

G4BiasingProcessInterface

G4GammaConversion

G4BiasingProcessInterface

G4ComptonScattering
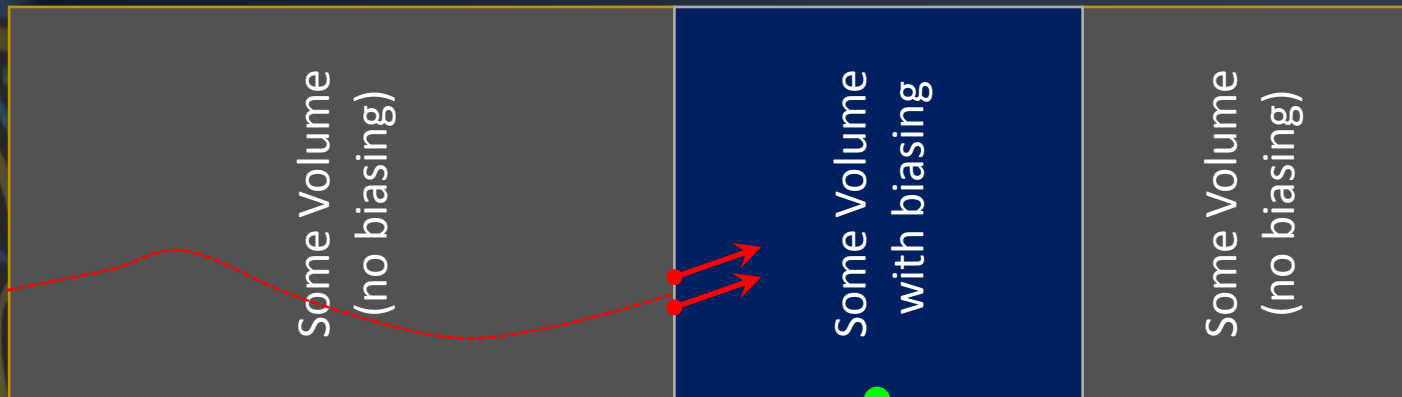
MyBiasingOperator

G4FlatForce-InteractionOperation

G4ExponentialForce-InteractionOperation

G4ForceFreeFlight-InteractionOperation

G4SplittingOperation

G4WeightWindow-Operation

G4MCNPForce-CollisionOperator

Some Volume (no biasing)

Some Volume with biasing

Some Volume (no biasing)

G4BiasingProcessInterface

(no process)

G4BiasingProcessInterface

G4PhotoelectricEffect

G4BiasingProcessInterface

G4GammaConversion

G4BiasingProcessInterface

G4ComptonScattering
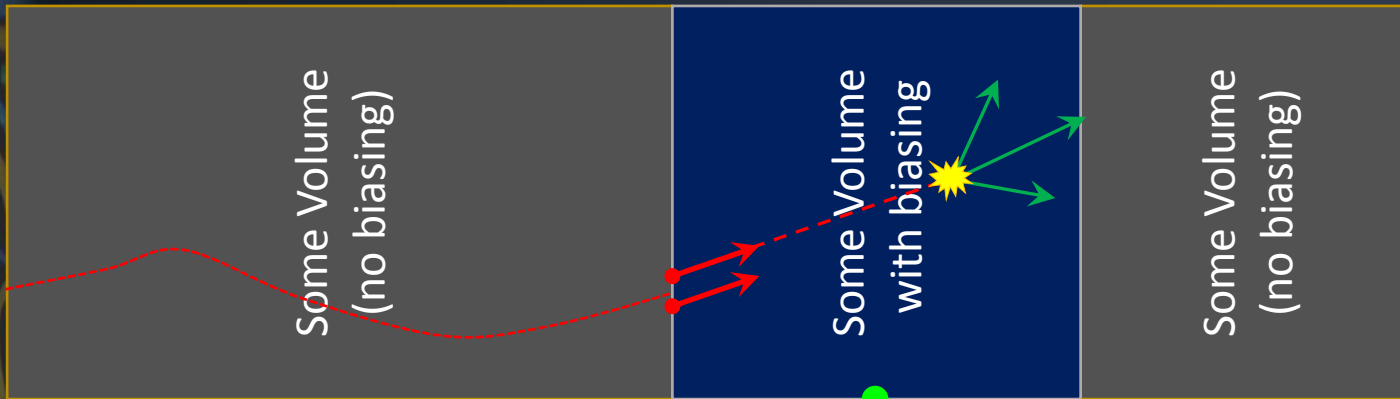
MyBiasingOperator

G4FlatForce-InteractionOperation

G4ExponentialForce-InteractionOperation

G4ForceFreeFlight-InteractionOperation

G4SplittingOperation

G4WeightWindow-Operation

G4MCNPForce-CollisionOperator

Some Volume (no biasing)

Some Volume with biasing

Some Volume (no biasing)

G4BiasingProcessInterface

(no process)

G4BiasingProcessInterface

G4PhotoelectricEffect

G4BiasingProcessInterface

G4GammaConversion

G4BiasingProcessInterface

G4ComptonScattering
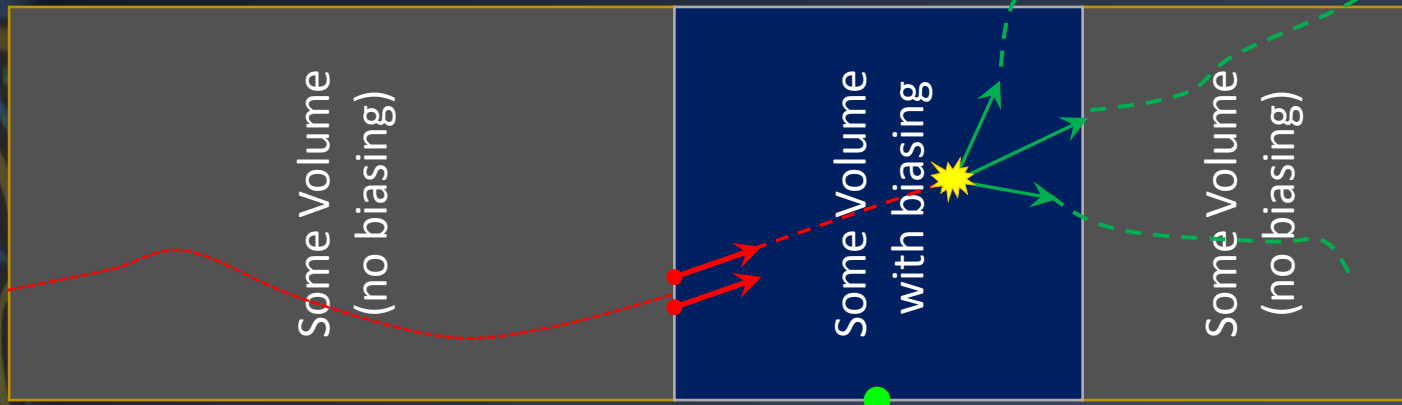
MyBiasingOperator

G4FlatForce-InteractionOperation

G4ExponentialForce-InteractionOperation

G4ForceFreeFlight-InteractionOperation

G4SplittingOperation

G4WeightWindow-Operation

G4MCNPForce-CollisionOperator

Some Volume (no biasing)

Some Volume with biasing

Some Volume (no biasing)

G4BiasingProcessInterface
(no process)

G4BiasingProcessInterface
G4PhotoelectricEffect

G4BiasingProcessInterface
G4GammaConversion

G4BiasingProcessInterface
G4ComptonScattering

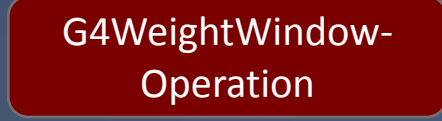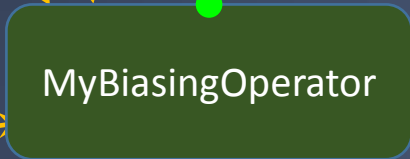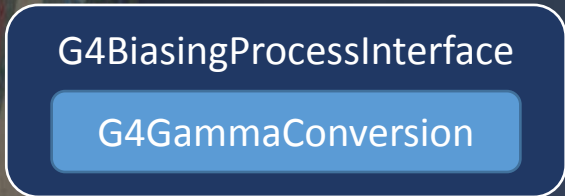MyBiasingOperator

G4FlatForce-InteractionOperation

G4ExponentialForce-InteractionOperation

G4ForceFreeFlight-InteractionOperation

G4SplittingOperation

G4WeightWindow-Operation

G4MCNPForce-CollisionOperator

Some Volume (no biasing)

Some Volume with biasing

Some Volume (no biasing)

G4BiasingProcessInterface

(no process)

G4BiasingProcessInterface

G4PhotoelectricEffect

G4BiasingProcessInterface

G4GammaConversion

G4BiasingProcessInterface

G4ComptonScattering
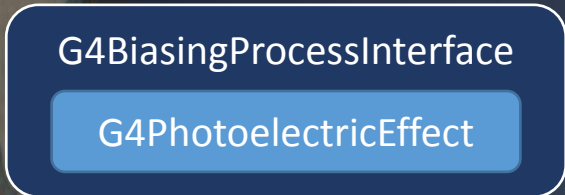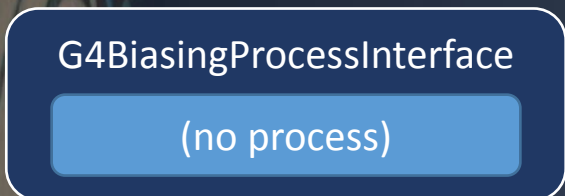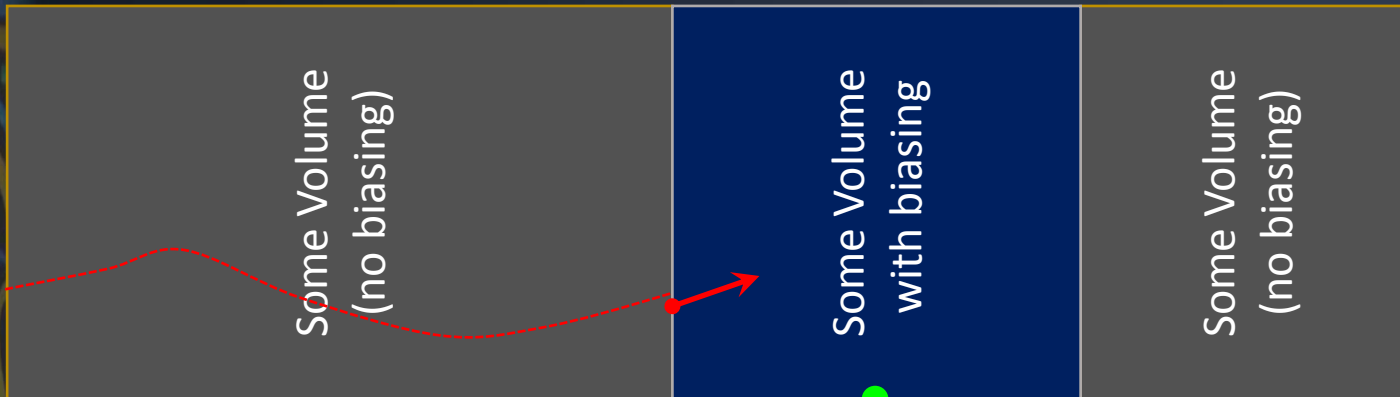
MyBiasingOperator

G4FlatForce-InteractionOperation

G4ExponentialForce-InteractionOperation

G4ForceFreeFlight-InteractionOperation

G4SplittingOperation

G4WeightWindow-Operation

G4MCNPForce-CollisionOperator

Some Volume (no biasing)

Some Volume with biasing

Some Volume (no biasing)

G4BiasingProcessInterface

(no process)

G4BiasingProcessInterface

G4PhotoelectricEffect

G4BiasingProcessInterface

G4GammaConversion

G4BiasingProcessInterface

G4ComptonScattering

MyBiasingOperator

G4FlatForce-InteractionOperation

G4ExponentialForce-InteractionOperation
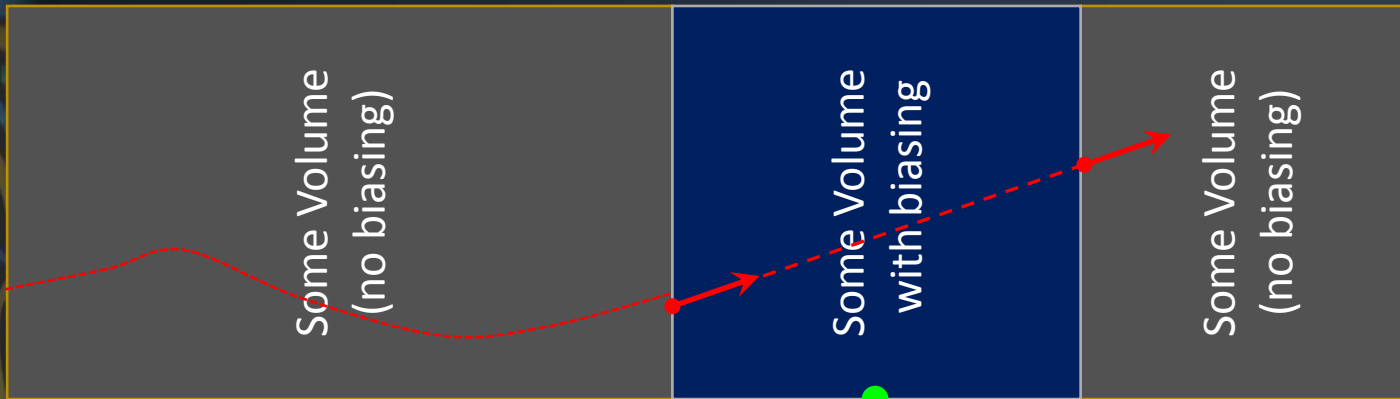
G4ForceFreeFlight-InteractionOperation

G4SplittingOperation

G4WeightWindow-Operation

G4MCNPForce-CollisionOperator

Existing functionalities & examples

› geant4/examples/extended/biasing/GB01 :
  – Individual process cross-section biasing
  – Implemented for neutral particles
  – Charged particle case requires development
    › Issue : variation of cross-section during step because of energy loss

› geant4/examples/extended/biasing/GB02 :
  – Force collision à la MCNP
  – Implemented for neutral particles (as MCNP)

› geant4/examples/extended/biasing/GB03 :
  – Geometry importance + further option
  – Scheme augmented compared to classical geometry importance
    › Allows kinds of "intermediate" non-integer splitting values

› geant4/examples/extended/biasing/GB04 :
  – Re-implementation of a classical Bremsstrahlung splitting

› geant4/examples/extended/biasing/GB05 :
  – Illustrates a "splitting by cross-section"
  – An invention (to my knowledge) where the splitting rate is directly adjusted from absorption cross-sections

› geant4/examples/extended/biasing/GB06 :
  – Parallel geometries with generic biasing.

Summary

› Biasing techniques can provide very large acceleration factors in rare event problems

› Geant4 proposes biasing options
  - Primary source (GPS) , leading particle, cross-section (had), radioactive decay, splitting with importance in geometry, weight window, user biasing with G4WrapperProcess, and bremstralhung splitting

› New "generic" approach since v10.0
  - Allowing biasing of process interaction and process final state
  - Together with easy mix with killing, splitting, etc.
  - Introduction of base classes allows extension of functionalities et re-use of components

› Further expected functionalities:
  - DXTRAN style biasing, Implicit capture, biasing at rest
  - And also charged particle case for cross-section biasing, Woodcock tracking, etc.

› Main difficulty:
  - Lack of manpower… but some fresh blood coming now

Backup

Some Words of Caution with Convergence

Introduction

› Biasing techniques are powerful
  - If a technique is well suited to a problem, gains in simulation efficiency by several order of magnitudes are usual

› In contrast to an analog simulation where all "events" have the same weight (==1), biasing is not "democratic"
  - Because of the weight, some events are more important than others

› This weight disparity may cause convergence problems
  - That can be nasty !
  - What happens is that, from time to time, an event with big weight may come, and change drastically the –say- estimated current mean value
  - Always a fear that this may happen...

› Proper convergence is the issue the user of biasing has to care about
  - And this is the price to pay for using biasing

› Let's illustrate a "bad convergence case" with a simple problem
  - But this allows to spot the difficulties

Convergence

› A « toy » example of a bad sampling:

- – We want to estimate the mean value of the unknown « yellow » distribution.

- – As we don't know where it is, we try to sample it with the « blue » distribution

  › And we know how to compute the weight

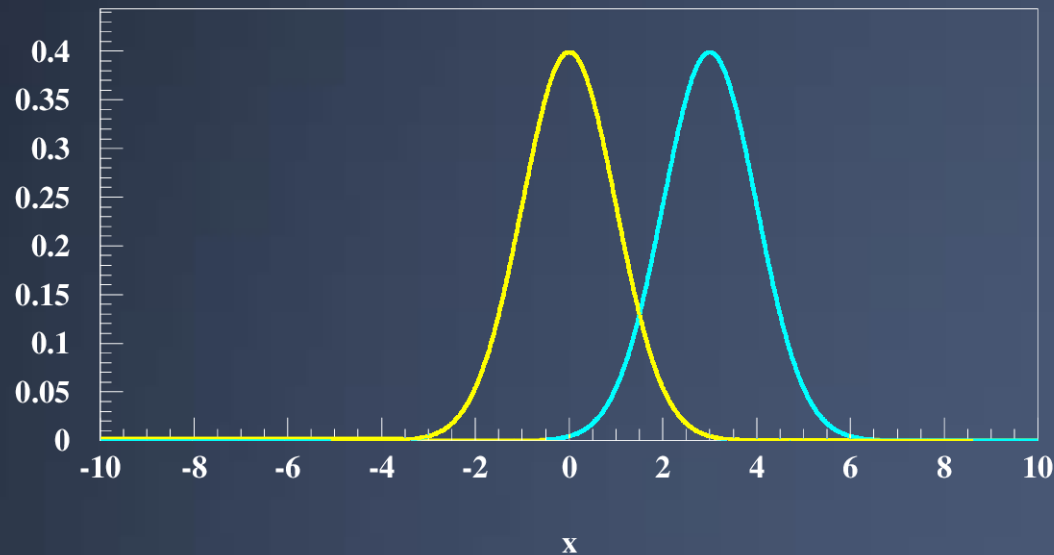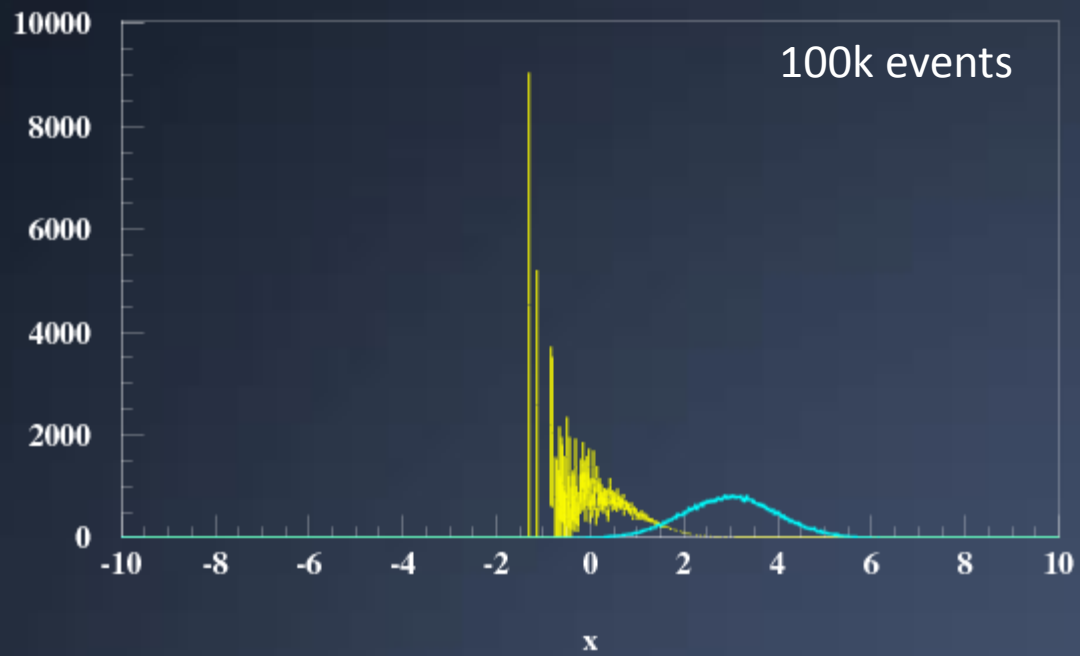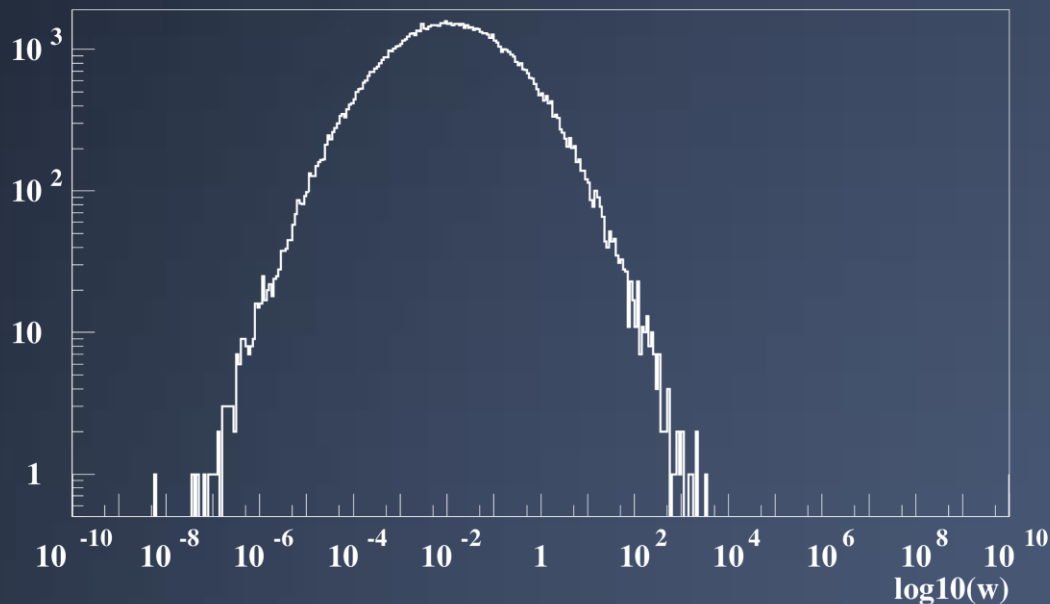# Convergence

› A « toy » example of a bad sampling:

  – We want to estimate the mean value of the unknown « yellow » distribution.

  – As we don't know where it is, we try to sample it with the « blue » distribution
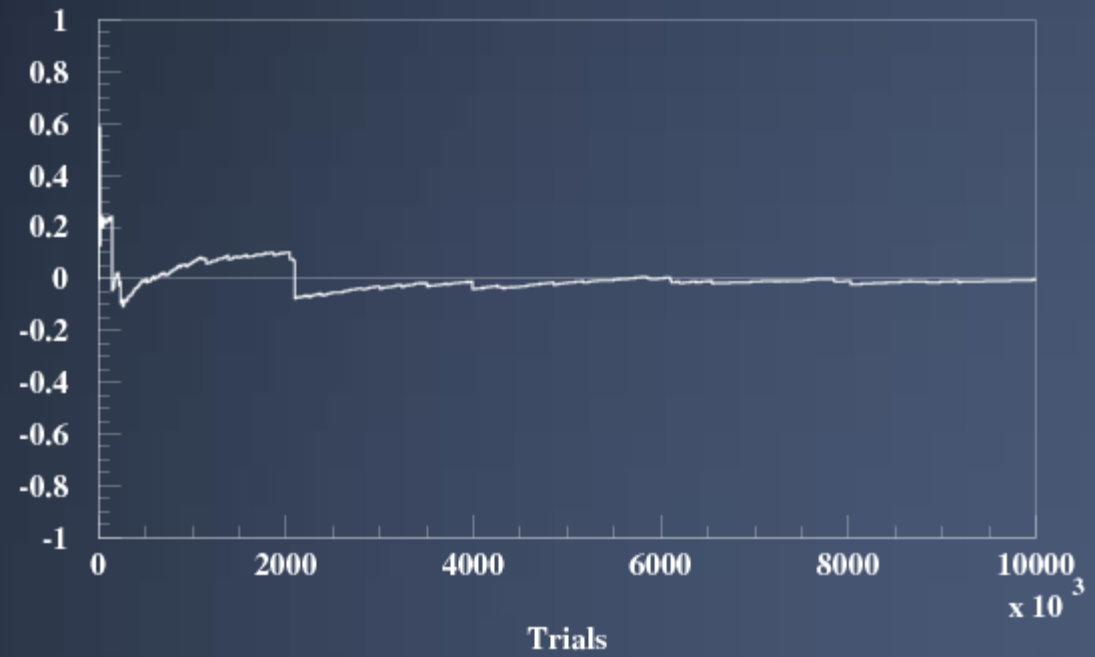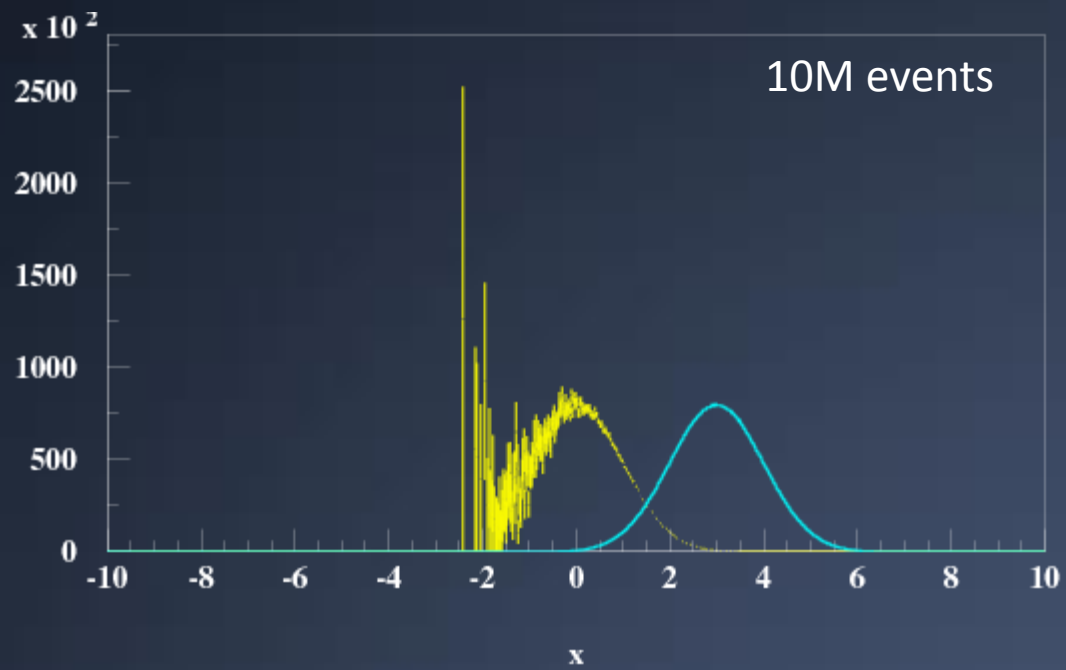
    › And we know how to compute the weight

100k events
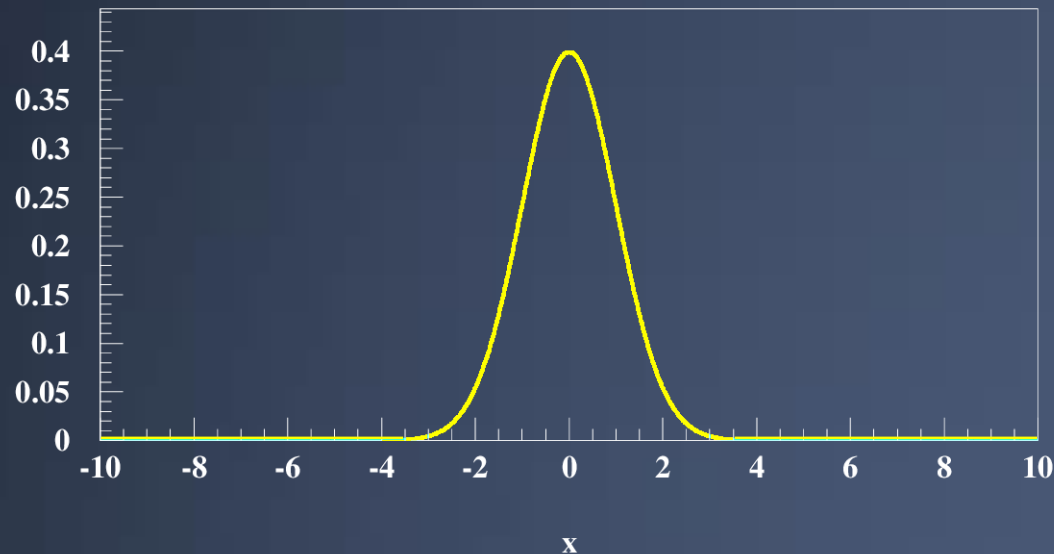
Weight distribution…

› Lots of tiny weights…

› Few huge ones…

10M events

Some qualitative observations

› Observations that can help spotting the inappropriate sampling:
- We have a wide variety of weights
  › Many tiny ones
    - Waste of time to determine the mean value
  › Few huge ones
    - Responsible for jumps
  › Such problem –if can not be improved- could at least be alleviated with a weight window technique
    - In a real problem (useless here in our toy problem)
- We have huge weights from time to time
  › These are not wrong !
  › Temptation would be to "dismiss" these events
  › But in our case, these events bring down the mean value to the correct value
  › These are not wrong, but *their presence is a sign of a bad sampling*
- We observe monotonic increase of the mean value
  › We are sampling only "one side" of the problem

› Convergence criteria have been established in other packages
- Like MCNP

› You are invited at staying critical when using such biasing techniques !

COMPARISON WITH FLUKA AND MCNPX

Few words of caution

› Compare the existing FLUKA and MCNPX functionalities with the existing and planned Geant4 ones:

- Ie : comparison is not exactly fair with FLUKA and MCNPX

› Need to say the above to be fair ;)

FLUKA / Geant4 biasing functionalities

| Biasing options in FLUKA from http://www.fluka.org/content/manuals/fluka2011.manual	Options in Geant4, Present, new, or future
Leading particle biasing for electrons and photons: region dependent, below user-defined energy threshold and for selected physical effects. → ?	Will be done, and will be shared with other "leading".
Russian Roulette and splitting at boundary crossing based on region relative importance.	Existing and re-done with generic scheme.
Region-dependent multiplicity tuning in high energy nuclear interactions.	Can be done and more general.
Region-dependent biased downscattering and non-analogue absorption of low-energy neutrons.	Can be done.
Biased decay length for increased daughter production.	Done.
Biased inelastic nuclear interaction length.	Done.
Biased interaction lengths for electron and photon electromagnetic interactions.	Done and validated for gamma.
Biased angular distribution of decay secondary particles.	Can de done.
Region-dependent weight window in three energy ranges (and energy group dependent for low energy neutrons).	Existing. Can be "re-provided" and more general.
Bias setting according to a user-defined logics.	(need more info in FLUKA, but is actual purpose of this dev.)
User-defined neutrino direction biasing.	Can be done, easily.
User-defined step by step importance biasing.	Can be done, easily.

Simple for neutral. More difficult for charged but understood.

MCNPX / Geant4 biasing functionalities

| Biasing options in MCNPX From LA-UR-03-1987, MCNP5 manual	Options in Geant4, Present, new, or future
Energy Cutoff & Time Cutoff	Existing (not considered as biasing)
Geometry Splitting with Russian Roulette	Existing and "re-provided" with generic scheme.
Energy Splitting/Roulette and Time Splitting/Roulette	Can be done easily with generic scheme.
Weight Cutoff	Existing (in some way). Easy in generic scheme.
Weight Window	Existing and will be "re-provided" with new design. Can be made more general.
Exponential Transform	Done (with generic scheme).
Implicit Capture (or "Implicit capture," "survival biasing," and "absorption by weight reduction")	Can be done. Planned for this year.
Forced Collisions	Done (with generic scheme).
Source Variable Biasing	Existing (GPS).
Point Detector Tally (?)	(not biasing ?)
DXTRAN	Planned, need more work. Doable.
Correlated Sampling	Not planned for now "à la MCNP". But doable with user's invest.

Examples of importance sampling techniques

› The "exponential transform" technique
  – It is a change of the total cross-section
  – Often made direction dependent
  – The usual/analog exponential law is replaced by an other law (still an exponential one here, but with different cross-section)

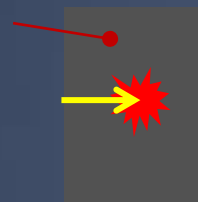Analog law → Biased law

If moving forward

If moving backward
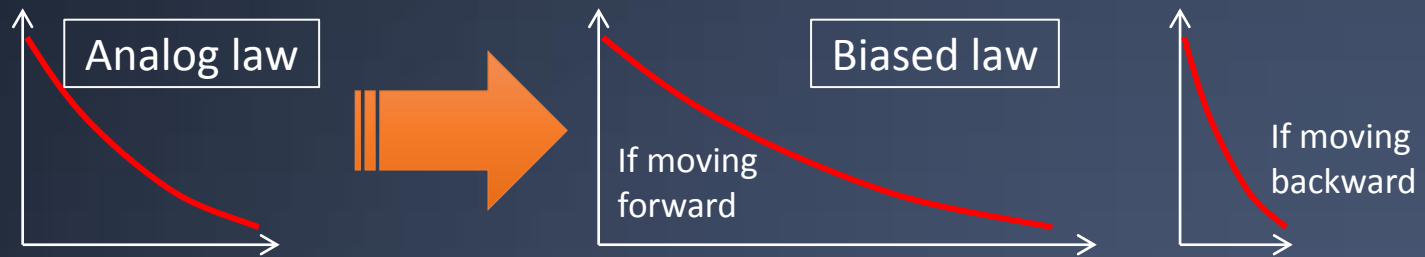
› Forcing the interaction in a thin volume:
  – The usual/analog exponential law is replaced by a "truncated" exponential law (not the only possible choice) that does not extend beyond the volume limit

Analog law

Biased law
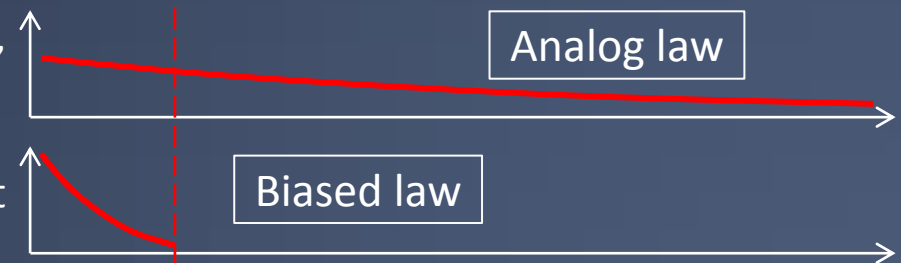
Volume in which we want the interaction to be forced

Geometry based importance biasing

› Geometry cells, meant to carry importance values, are created and associated to physical volumes in the detector construction:

```
G4IStore *istore = G4IStore::GetInstance();
// Loop on physical volumes:
istore->AddImportanceGeometryCell(importanceValue,  physicalVolumePointer [, nReplica]);
```

Replica number, if any

  – All volumes should be given an importance value.

› Biasing may be applied to some particle type only: eg neutron. In the main program, this is specified as:

```
G4GeometrySampler geometrySample(worldVolume,"neutron");
```

  – (worldVolume specified as there is the ability to define cells in a parallel geometry: not discussed here)

› The actual splitting and killing are handled by dedicated processes. These are to be inserted in the physics list. If using a modular physics, it is done as:

```
G4VModularPhysicsList* physicsList = new FTFP_BERT;
physicsList->RegisterPhysics(new G4ImportanceBiasing(&geometrySampler));
```

› More details can be found in examples:

  – examples/extended/biasing/B01 : geometry biasing with importance cells or weight window
  – examples/extended/biasing/B02 : same, with cells in a parallel geometry
  – examples/extended/biasing/B03 : same, for the case of a non modular physics list

Introduction

› Many options already exist. But they
  – do not allow to change the probability distribution of processes
  – do not make "easy" to change the final state behavior of a process
  – do not make "easy" to mix biasing options

› Current development aims/hopes at improving this
  – to allow options like
    › Exponential transform: $p(\ell) = \sigma \cdot e^{-\sigma\ell} \rightarrow p'(\ell) = \sigma' \cdot e^{-\sigma'\ell}$
      – Change total cross-section
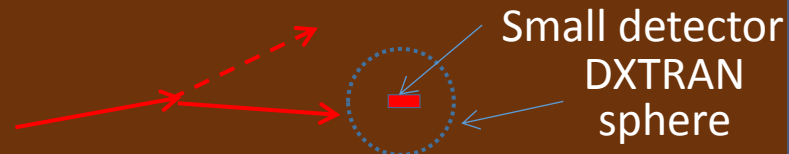   – Make change direction dependent
    › forced interaction:
      – Force interaction in thin volume

      Thin detector
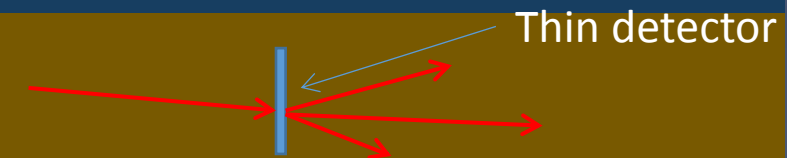
    › forced flight (towards detector)
      – So called DXTRAN
      – Force scattering towards detector

      Small detector
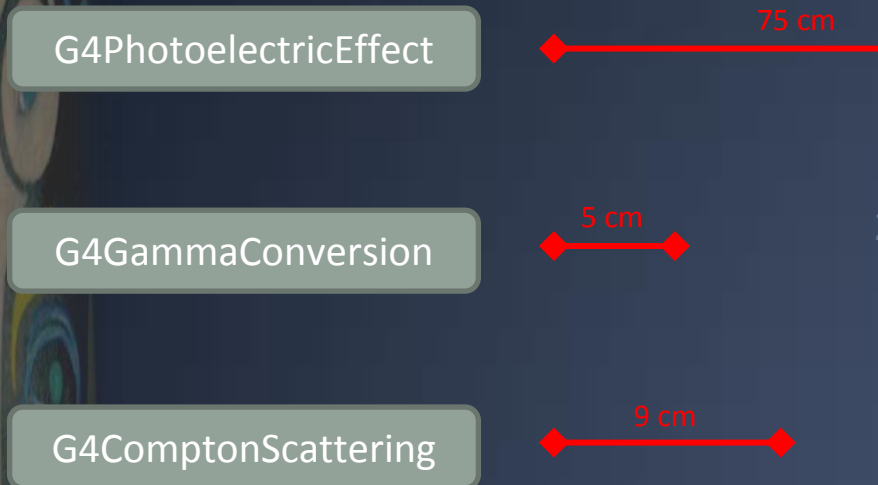   DXTRAN
      sphere

    › etc.
  – and to allow complex logic to be implemented by providing "building blocks" rather than "built-in" functionalities

Physics process behavior in Geant4 :
where to bias ?

› Illustrate with point-like processes:

1. GetPostStepPhysicalInteractionLength()
   - Returns the distance at which the process will make an interaction
   - Analog exponential law is at play
   - Above analog behavior superseded by biased behavior (if wished)

| G4PhotoelectricEffect | ◆————— 75 cm —————◆ |

2. PostStepDoIt()
   - Called if the process has responded the shortest of the interaction distances

| G4GammaConversion | ◆— 5 cm —◆ |

   - Generate final state, according to specific process analog physical law
   - Above analog behavior superseded by biased behavior (if wished)

| G4ComptonScattering | ◆—— 9 cm ——◆ |

› G4BiasingProcessInterface wraps the physics process. It is meant to intercept calls for distance of interaction and for final state production.

Physics process behavior in Geant4 :
where to bias ?

› Illustrate with point-like processes:

| G4PhotoelectricEffect |

75 cm

| G4GammaConversion |

5 cm

e⁺

e⁻

| G4ComptonScattering |

9 cm

1. GetPostStepPhysicalInteractionLength()
   - Returns the distance at which the process will make an interaction
   - Analog exponential law is at play
   - Above analog behavior superseded by biased behavior (if wished)

2. PostStepDoIt()
   - Called if the process has responded the shortest of the interaction distances
   - Generate final state, according to specific process analog physical law
   - Above analog behavior superseded by biased behavior (if wished)

› G4BiasingProcessInterface wraps the physics process. It is meant to intercept calls for distance of interaction and for final state production.
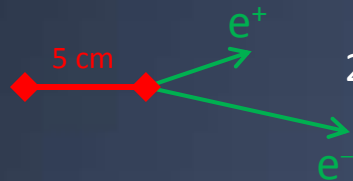
Physics process behavior in Geant4 : where to bias ?

› Illustrate with point-like processes:

G4BiasingProcessInterface

> G4PhotoelectricEffect

— 12 cm —

G4BiasingProcessInterface

> G4GammaConversion

— 10 cm —

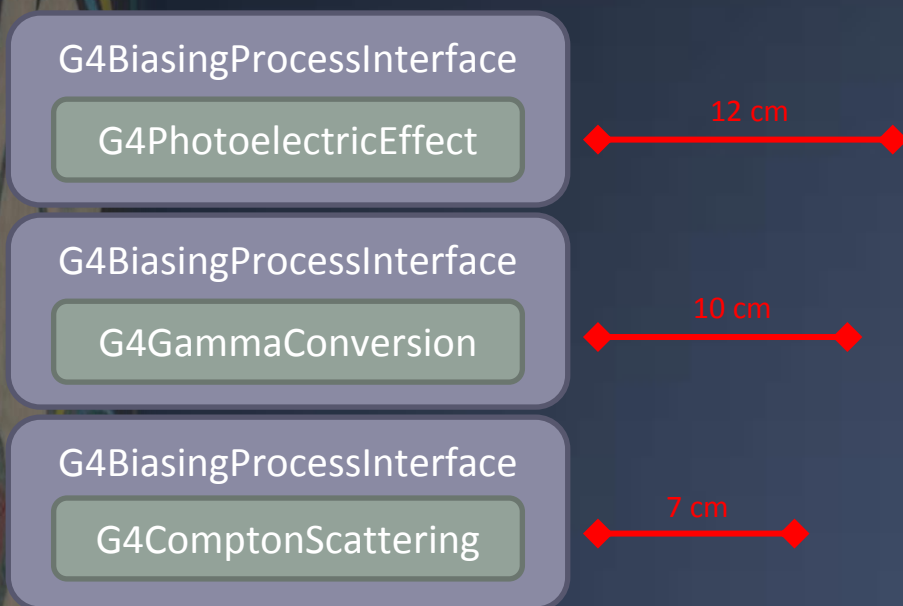**G4BiasingProcessInterface**

> G4ComptonScattering

— 7 cm —

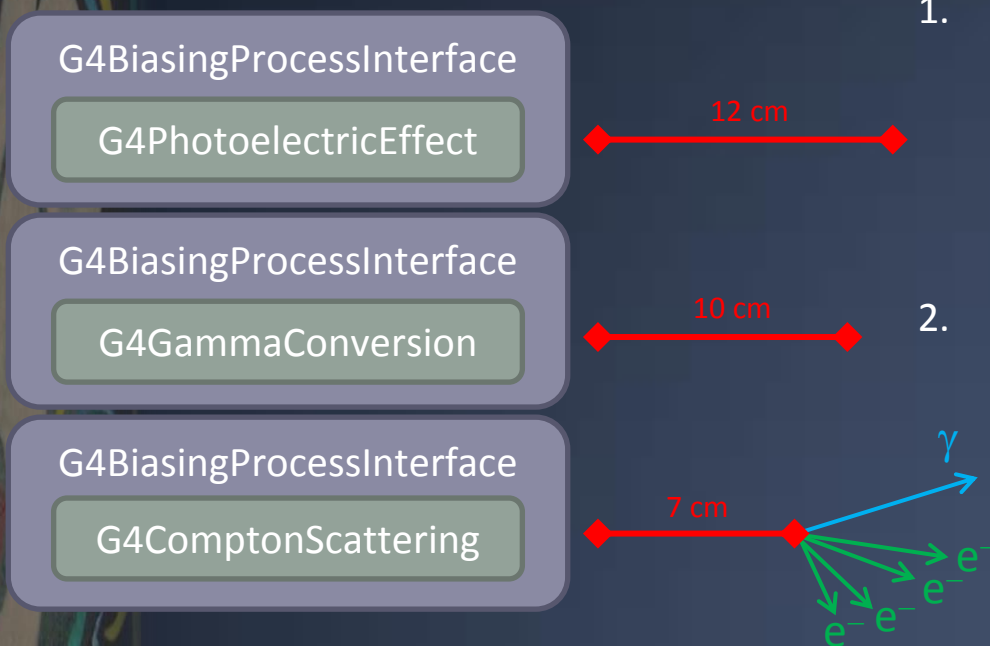1. GetPostStepPhysicalInteractionLength()
   - Returns the distance at which the process will make an interaction
   - ~~Analog exponential law is at play~~
   - Above analog behavior superseded by biased behavior (if wished)
2. PostStepDoIt()
   - Called if the process has responded the shortest of the interaction distances
   - Generate final state, according to specific process analog physical law
   - Above analog behavior superseded by biased behavior (if wished)

› G4BiasingProcessInterface wraps the physics process. It is meant to intercept calls for distance of interaction and for final state production.

Physics process behavior in Geant4 : where to bias ?

› Illustrate with point-like processes:

G4BiasingProcessInterface

> G4PhotoelectricEffect

⟵ 12 cm ⟶

G4BiasingProcessInterface

> G4GammaConversion

⟵ 10 cm ⟶

G4BiasingProcessInterface

> G4ComptonScattering

⟵ 7 cm ⟶ γ e^- e^- e^- e^-

1. GetPostStepPhysicalInteractionLength()
   - Returns the distance at which the process will make an interaction
   - ~~Analog exponential law is at play~~
   - Above analog behavior superseded by biased behavior (if wished)

2. PostStepDoIt()
   - Called if the process has responded the shortest of the interaction distances
   - ~~Generate final state, according to specific process analog physical law~~
   - Above analog behavior superseded by biased behavior (if wished)

› G4BiasingProcessInterface wraps the physics process. It is meant to intercept calls for distance of interaction and for final state production.

Existing functionalities & examples

› In main program:

```cpp
// -- Select a modular physics list
FTFP_BERT* physicsList = new FTFP_BERT;
// -- And augment it with biasing facilities:
G4GenericBiasingPhysics* biasingPhysics = new G4GenericBiasingPhysics();
biasingPhysics->Bias("gamma");
biasingPhysics->Bias("neutron");
physicsList->RegisterPhysics(biasingPhysics);
runManager->SetUserInitialization(physicsList);
```

› In ConstructSDandField() of detector construction:

```cpp
MyBiasingOperator* biasingOperator = new MyBiasingOperator();
biasingOperator->AttachTo( logicalVolumeToBias );
```

› Examples:
  – geant4/examples/extended/biasing/GB01 : individual process cross-section biasing
  – geant4/examples/extended/biasing/GB02 : force collision à la MCNP
  – geant4/examples/extended/biasing/GB03 : geometry importance + further option
  – geant4/examples/extended/biasing/GB04 : bremsstrahlung splitting
  – geant4/examples/extended/biasing/GB05 : illustrates a "splitting by cross-section"
  – geant4/examples/extended/biasing/GB06 : parallel geometries with generic biasing.

› Remember that G4VBiasingOperation and G4VBiasingOperator are base classes meant for extending functionalities (toolkit approach)