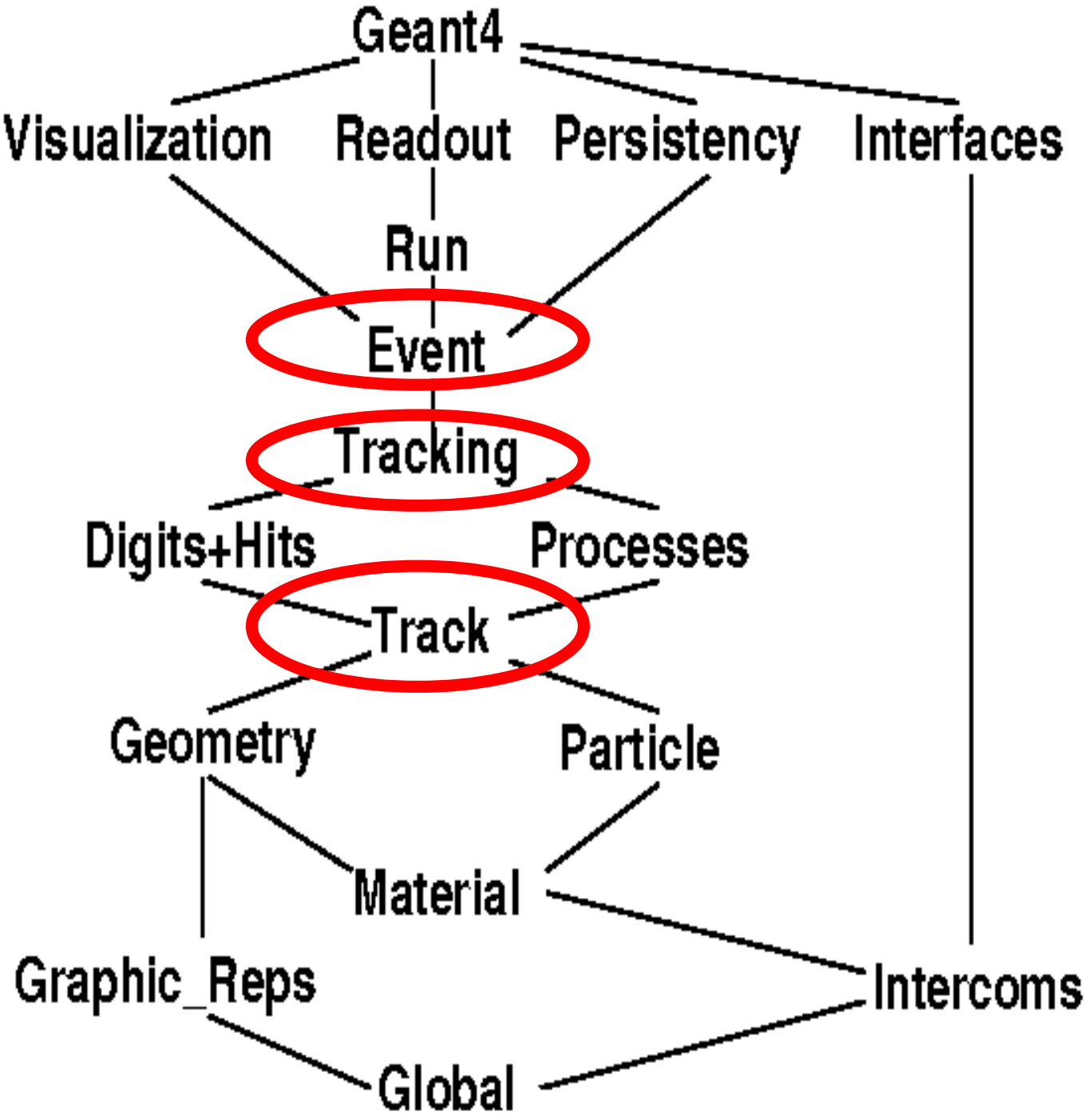
Трекинг


Geant4
Visualization
Readout
Persistency
Interfaces
Run
Event
Tracking
Digits+Hits
Processes
Track
Geometry
Particle
Material
Graphic_Reps
Intercoms
Global

Действия, определяемые пользователем

- Классы-наследники G4UserTrackingAction и G4UserSteppingAction
- G4UserTrackingAction: для каждого трека (до и после трекинга частицы) вызываются методы соответственно:
  - ***virtual void PreUserTrackingAction(const G4Track*)***
  - ***virtual void PostUserTrackingAction(const G4Track*)***
- G4UserSteppingAction: на каждом шаге вызывается метод
  - ***virtual void UserSteppingAction(const G4Step*)***

UserStackingAction

- Дает возможность управлять порядком трекинга частиц (аналог триггера)
- Обычный алгоритм:
  - частица тянется до остановки, исчезновения в реакции или границы мирового объема
  - все произведенные ей вторичные частицы помещаются в стек (есть два стека: UrgentStack и WaitingStack)
  - по умолчанию, все треки помещаются в UrgentStack
  - последняя рожденная частица трассируется первой
  - для третичных и т. д. применяется аналогичная схема
- UserStackingAction: наследник **G4UserStackingAction**

void PrepareNewEvent();
инициализация перед новым событием
G4ClassificationOfNewTrack ClassifyNewTrack(const G4Track*);
определение приоритета обработки данного трека
void NewStage();
действия после обработки UrgentStack

Пределы, устанавливаемые пользователем (UserLimits)

- Применяются в целях оптимизации
- Могут устанавливаться для отдельных частиц и отдельных логических объемов
- Позволяют устанавливать
  - **максимально допустимую длину шага**
  - **максимальную длину трека**
  - **максимальное время жизни**
  - **минимальную кинетическую энергию**
  - **минимальное расстояние пролета**

Примеры UserLimits (1)

Принудительное ограничение длины шага:

В **UserAppDetectorConstruction:**

```
G4double maxStep = 0.1*cm;
logicTracker->SetUserLimits(new G4UserLimits(maxStep));
```

В **UserAppPhysicsList**

```
pmanager->AddDiscreteProcess(new G4StepLimiter);
```

Примеры UserLimits (2)

Более общая конструкция:

В **UserAppDetectorConstruction**

```
G4double maxTime = 10*ms;
logicTracker->SetUserLimits(new
                    G4UserLimits(DBL_MAX,DBL_MAX,maxTime));
```

В **UserAppPhysicsList**

```
G4ProcessManager* pmanager =
                              G4Neutron::Neutron->GetProcessManager();
pmanager->AddProcess(new G4UserSpecialCuts(),-1,-1,1);
```

Как происходит один шаг в моделировании (1)

1. Считается скорость частицы в начале шага
2. Определяется длина свободного пробега независимо для каждого активного процесса. Выбирается наименьшая.
3. Рассчитывается расстояние до граничной поверхности ближайшего объема. Если это расстояние больше, чем наименьшее определенное для физических процессов, выбирается последнее, и дальнейшие геометрические расчеты не делаются

Как происходит один шаг в моделировании (2)

4. Производится расчет непрерывных взаимодействий согласно длине шага, и соответственно в конце изменяется значение кинетической энергии
5. Производится проверка, стоит ли продолжать трекинг данной частицы
6. Обновляются параметры частицы (энергия и положение)
7. Производится расчет дискретных процессов. Если требуется, создается список вторичных частиц.

Как происходит один шаг в моделировании (3)

8. Производится проверка, стоит ли продолжать трекинг данной частицы
9. Рассчитывается расстояние до граничной поверхности ближайшего объема.
10. Если шаг ограничен этой поверхностью, происходит переход в следующий объем
11. Производятся действия, определяемые пользователем (G4UserSteppingAction) и обрабатывается информация в детектирующих элементах
12. Добавляется точка в траектории движения

Повышение эффективности моделирования

Как повысить эффективность расчетов

$$\sigma \sim 1/\sqrt{N} \qquad T_{CPU} \sim N$$

Повышение точности в два раза требует вчетверо больших затрат процессорного времени

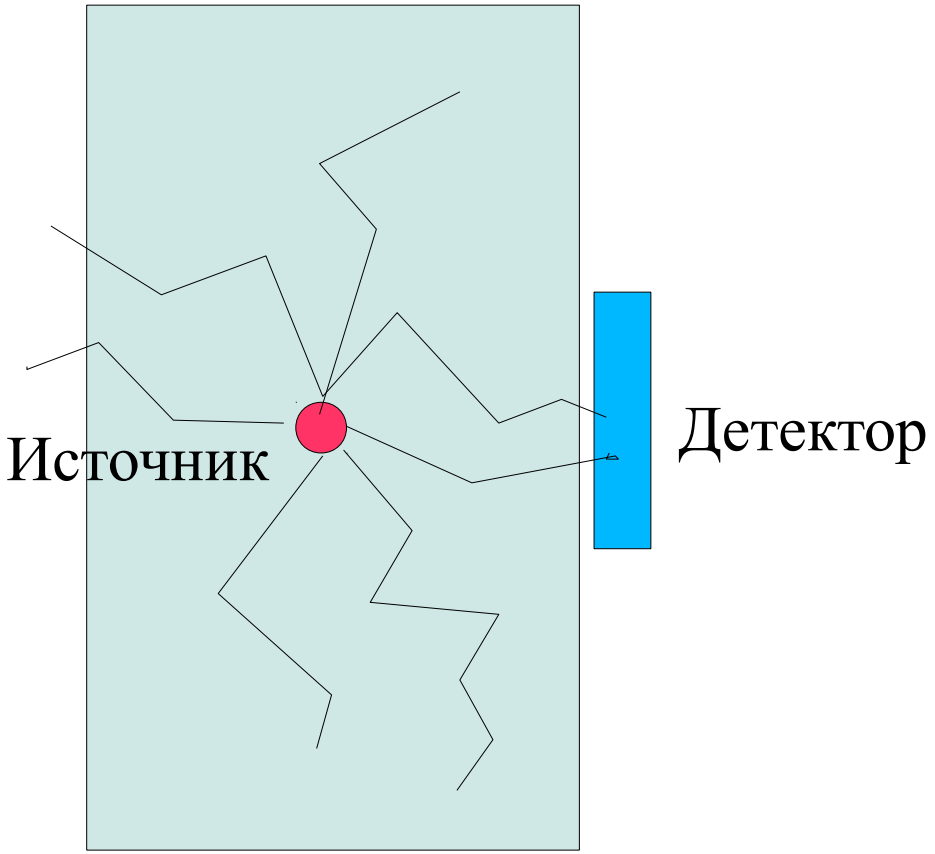
Как достичь большей точности при меньших затратах CPU:

- Подробное моделирование более интересных областей установки за счет менее подробного моделирования неинтересных (выборка по значимости или importance sampling)
- Изменение вероятности интересных физических процессов для их более частого моделирования
- У каждого трека появляется дополнительный вес

  G4double w = aTrack->GetWeight();

  aTrack->SetWeight(G4double w);

Выборка по значимости объема



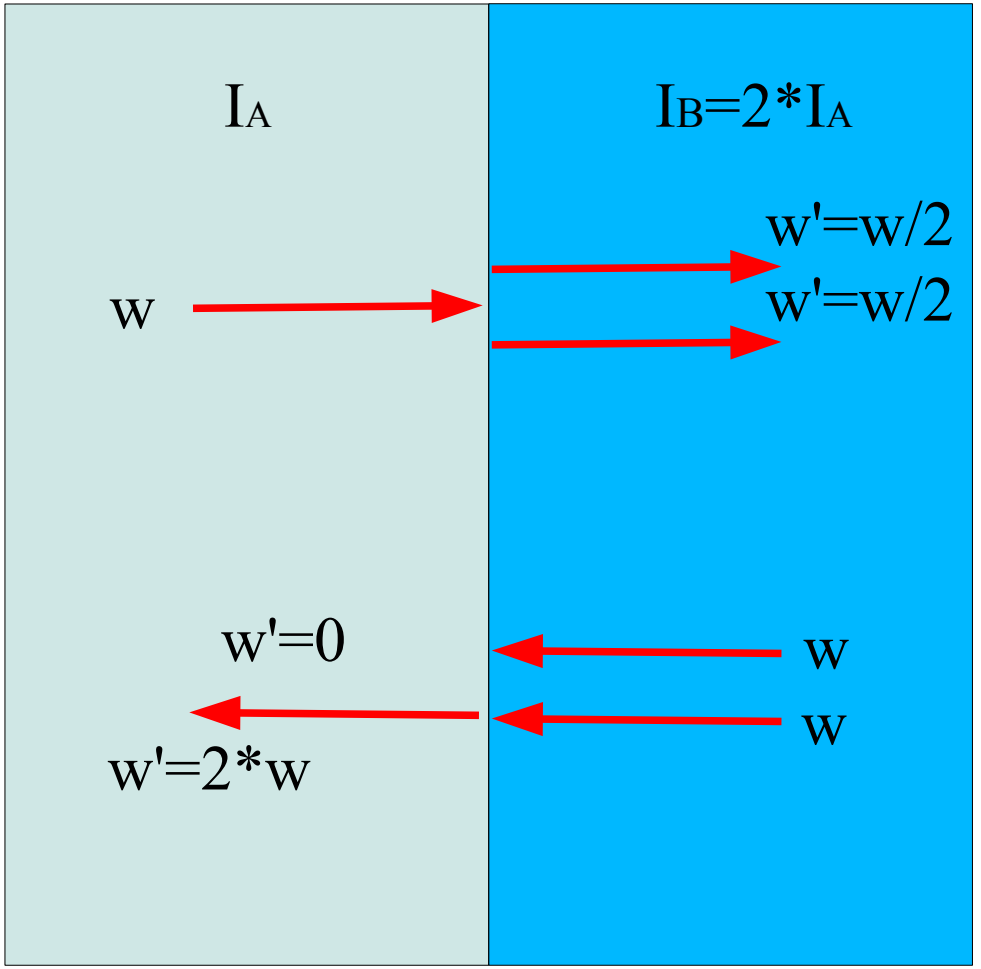
Выборка по значимости объема

- Детектор разбивается на области (cells) и каждой области в зависимости от ее важности для моделирования присваивается значимость
  - *Вариант 1: cell = реальный физический объем*
  - *Вариант 2: cell = объем в параллельной геометрии*

- Более значимые области моделируются более подробно = моделируется больше треков

- При пересечении границ объемов трек либо останавливается, либо дублируется, в зависимости от относительного изменения значимости

Выборка по значимости объема



- ***r=I_B/I_A***
- ***r=1*** обычный трекинг
- ***r<1*** трек останавливается с вероятностью 1-r (Russian roulette)
- ***r>1***
  - если r целое, создается r копий трека
  - если r не целое, то создается r+1 копий с вероятностью p=r-floor(r) или r копий с вероятностью 1-p
- Треку присваивается вес, сохраняющий число частиц

Реализация в программе

- DetectorConstruction.hh

```
class DetectorConstruction : public G4VUserDetectorConstruction {
public:
 ...
 G4GeometrySampler* sampler;
```

- DetectorConstruction.cc

```
sampler = new G4GeometrySampler(physWorld,"gamma");
G4VImportanceAlgorithm* ialg = new G4ImportanceAlgorithm();
G4IStore* istore = new G4IStore(*physWorld);
istore->AddImportanceGeometryCell(1,*physWorld);
istore->AddImportanceGeometryCell(1,*physSand);
istore->AddImportanceGeometryCell(2,*physWater);
istore->AddImportanceGeometryCell(4,*physIron);
sampler->PrepareImportanceSampling(istore, ialg);
```

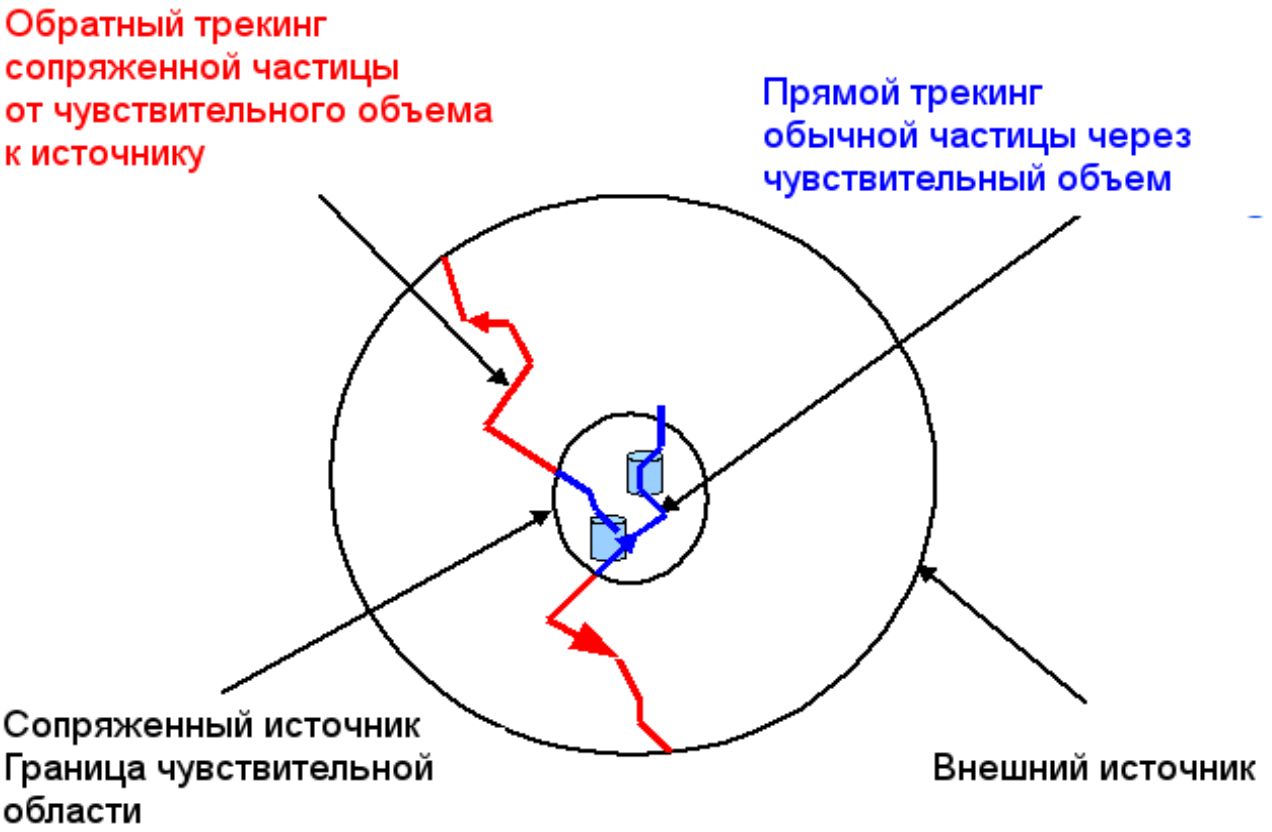
- main.cc

```
runManager->Initialize();
detectorConstruction->sampler->Configure();
```

Весовое окно

- Аналогичная техника может применяться не только для объемов, но и для «ячеек» в фазовом пространстве
- Применяется на границе объемов, при столкновении и при столкновении на границе
- Пользователь определяет
  - *нижнюю границу весового окна*
  - *отношение максимального и минимального весов*
  - *отношение веса выживания и минимального веса*
- Веса определяются для различных физических объемов и диапазонов энергии
- Наиболее эффективно в сочетании с выборкой по значимости (например, чтобы убрать треки со слишком маленьким или слишком большим весом)

Обратное моделирование
Adjoint/Reverse MC



- Реализовано только для некоторых ЭМ процессов (ионизация, многократное рассеяние, комптоновское рассеяние, тормозное излучение, фотоэффект)
- Пример: *examples/extended/biasing/ReverseMC01*

Смещенная выборка для физических процессов

- Вместо функции плотности вероятности p(x) используется функция p'(x). Каждому треку присваивается вес w=p(x)/p'(x)
- Таким образом можно изменять на более выгодные угловые распределения, энергетический спектр вторичных частиц, сечение взаимодействия и т.д.
- В Geant4 такая возможность реализована через «псевдопроцесс» G4WrapperProcess

G4WrapperProcess

```
class G4WrapperProcess : public G4VProcess {
G4VProcess* pRegProcess;
…
inline void G4WrapperProcess::RegisterProcess(G4VProcess* process)
{
      pRegProcess=process;
      …
}

…

inline G4VParticleChange*
G4WrapperProcess::PostStepDoIt(const G4Track& track, const G4Step& stepData)
{
      return pRegProcess->PostStepDoIt(track, stepData);
}
```

аналогично для остальных методов DoIt

Пример реализации

```
class MyWrapperProcess : public G4WrapperProcess {
…
G4VParticleChange* PostStepDoIt(const G4Track& track, const G4Step& step)
 {
     // Your own implementation
  }
}


void MyPhysicsList::ConstructProcess()
{
     …
     G4LowEnergyBremsstrahlung* bremProcess =
                                                  new G4LowEnergyBremsstrahlung();
     MyWrapperProcess* wrapper = new MyWrapperProcess();
     wrapper->RegisterProcess(bremProcess);
     processManager->AddProcess(wrapper, -1, -1, 3);
}
```

Изменение сечения адронных процессов

```
void MyPhysicsList::ConstructProcess()
 {
   ...
   G4ElectroNuclearReaction * theElectroReaction =
     new G4ElectroNuclearReaction;

   G4ElectronNuclearProcess theElectronNuclearProcess;
   theElectronNuclearProcess.RegisterMe(theElectroReaction);
   theElectronNuclearProcess.BiasCrossSectionByFactor(100);

   pManager->AddDiscreteProcess(&theElectronNuclearProcess);
   ...
 }
```

Метод работает для любого наследника G4HadronicProcess

Радиоактивный распад

- В классе G4RadioactiveDecay реализованы следующие способы смещенной выборки:
  - Изменение вероятности распада (sampling rate biasing)
  - Дублирование распадающихся нуклидов (nuclear splitting)
  - Изменение вероятностей мод распада (branching ratio biasing)

- Пример использования:
  examples/extended/radioactivedecay/exrdm

Организация массовых вычислений

Как смоделировать 10^{10} событий?

- Простое моделирование: 1 событие ~ 1 мс
- 10^{10} событий: 115 дней работы ЦПУ
- Решение — параллельные вычиления!
- Метод Монте-Карло идеально распараллеливается, каждое событие может моделироваться независимо
- Требуется:
  - организовать запуск N копий программы
  - передать каждой копии начальное значение генератора случайных чисел
  - объединить результат

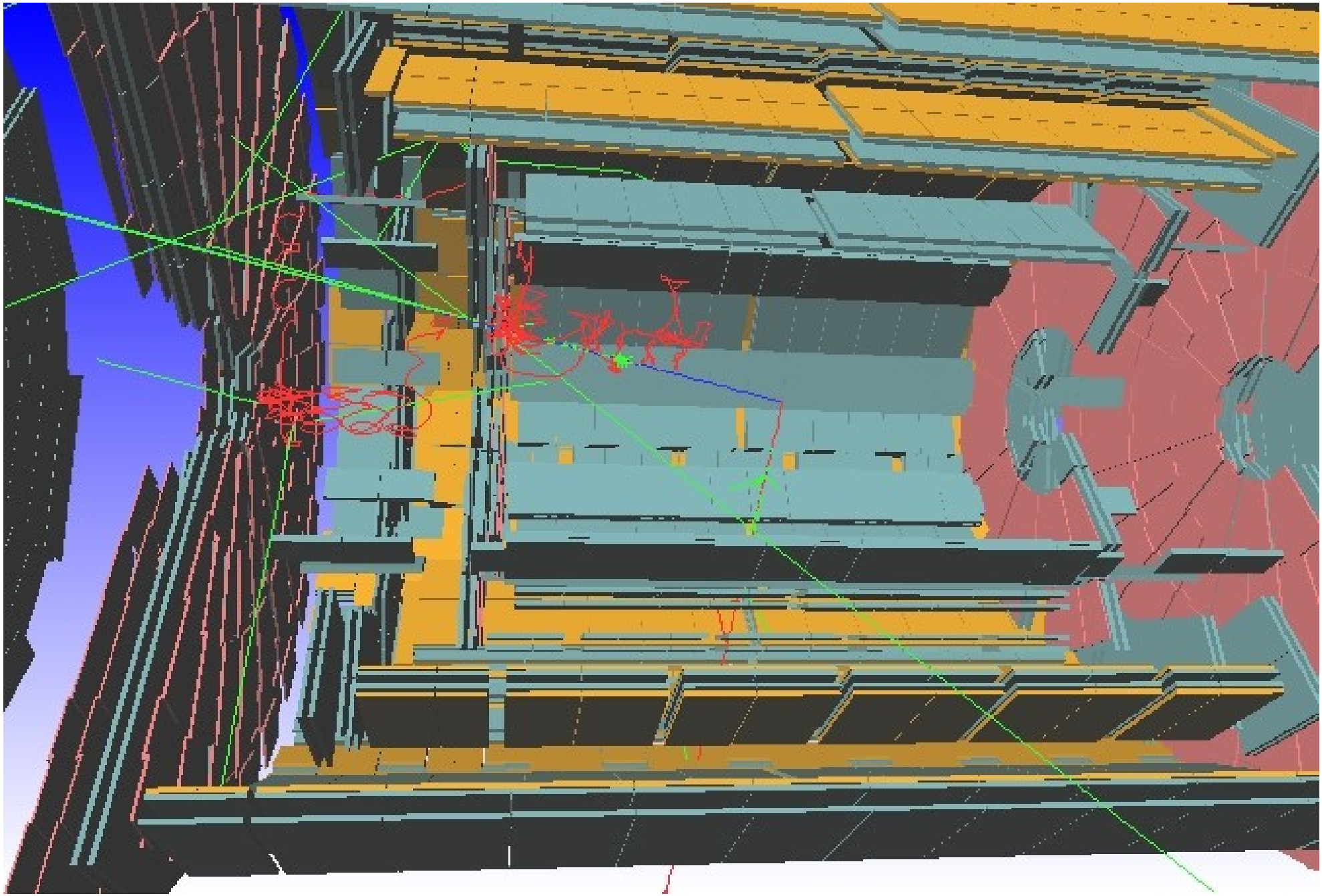
Как передать начальное значение генератора случайных чисел

```
int main()
{...
int runNumber = GetRunNumber(); // номер копии программы

runManager->Initialize();

long rand[2];
rand[0] = long(runNumber*1000000 + 123456);
rand[1] = 123456789;

char seeds[80];
sprintf(seeds,"/random/setSeeds %ld %ld",rand[0],rand[1]);
G4cout << seeds << G4endl;
UImanager->ApplyCommand(seeds);
...}
```

Заключительные замечания

Особенности моделирования в больших экспериментах

- Сложные детекторы
  - большое (1000-1000000) число объемов
  - сложная геометрия объемов
- Обширная физическая программа
  - множество моделей физических процессов
  - различные цели моделирования = различные наборы существенных физических процессов, различные модели для разных G4Regions
- Сложное программное обеспечение
  - работа в составе программной оболочки (framework)
  - внешний, единый для программ моделирования и реконструкции, банк геометрических параметров детектора и калибровочных констант
  - единый подход к сохранению данных



Преимущества и недостатки Geant4

Преимущества

- все преимущества C++: гибкость, расширяемость, модульная структура, ...
- гибкий и мощный аппарат описания геометрии
- универсальное описание практически всех известных взаимодействий элементарных частиц в веществе
- вполне достоверное моделирование физических процессов при высоких энергиях
- минимум лицензионных ограничений, открытый код

Преимущества и недостатки Geant4

Недостатки

- Сложность выбора или составления адекватного набора физических процессов
- Недостаточно верифицированные модели физических взаимодействий при средних и низких энергиях ($\leq$100 МэВ для ЭМ и $\leq$10 ГэВ для сильных взаимодействий). В связи с этим, точные расчеты в этой области требуют дополнительно предварительной проверки адекватности используемых моделей
- Недостаточная встроенная оптимизация для ряда прикладных задач
- Отсутствие встроенных средств сохранения результатов моделирования
- Ограниченные возможности GUI и определенные трудности с визуализацией

Ожидаемые ближайшие перспективы развития Geant4

- Уточнение и верификация моделей физических процессов, особенно в области низких энергий и сверхвысоких энергий
  - *новые экспериментальные данные*
  - *новые модели*
  - *уточнение теоретических расчетов*
- Упрощение описания сложной геометрии
  - *использование универсального формата описания геометрии (GDML) (подробно http://cern.ch/gdml)*
  - *описание новых форм*
- Оптимизация для решения прикладных задач
  - *Переменная плотность вещества*
  - *Твердотельные эффекты*

Перспективы применения Geant4

- Основной инструмент моделирования в физике элементарных частиц в ближайшие 10 лет
- Растущее применение в ядерной физике
- Растущее применение в медицинской физике
- Потенциально широкое применение в прикладных исследованиях (исследование радиационной эффектов в веществе, расчет защиты космических аппаратов и т.д.)