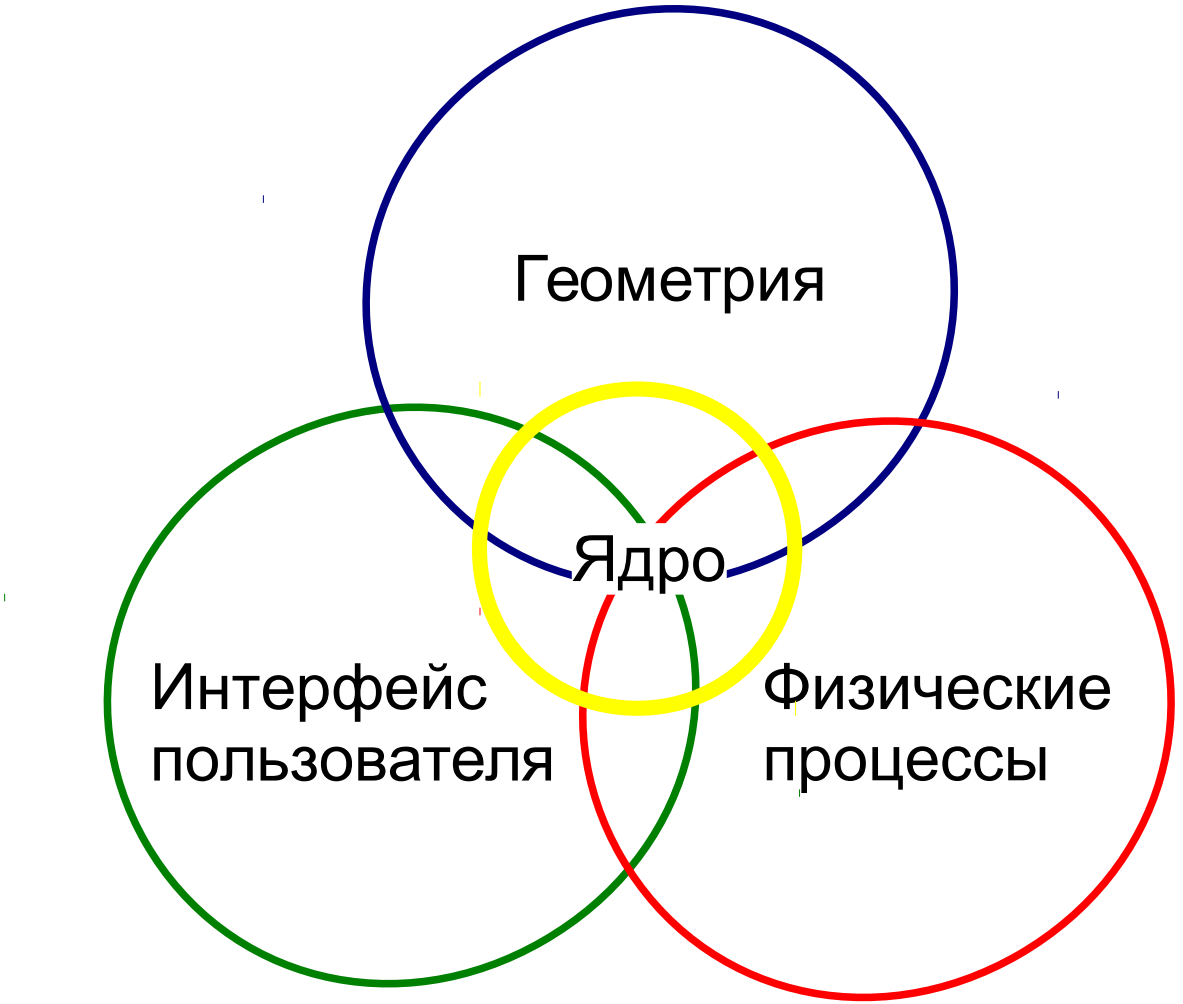
Пакет программ GEANT4

Geant4 – это набор инструментов

- GEANT4 представляет собой набор программ для моделирования прохождения частиц через вещество
- Включает в себя инструменты для гибкого описания геометрии
- Содержит множество физических моделей взаимодействия частиц с веществом
  - **Электромагнитные процессы**
  - **Адронные процессы**
  - **Фотон-адронные и лептон-адронные процессы**
  - **Процессы с участием оптических фотонов**
  - **Моделирование распадов**
  - **Параметризация ливней**
  - **Методики использования статистических весов**


Геометрия
Ядро
Интерфейс
пользователя
Физические
процессы

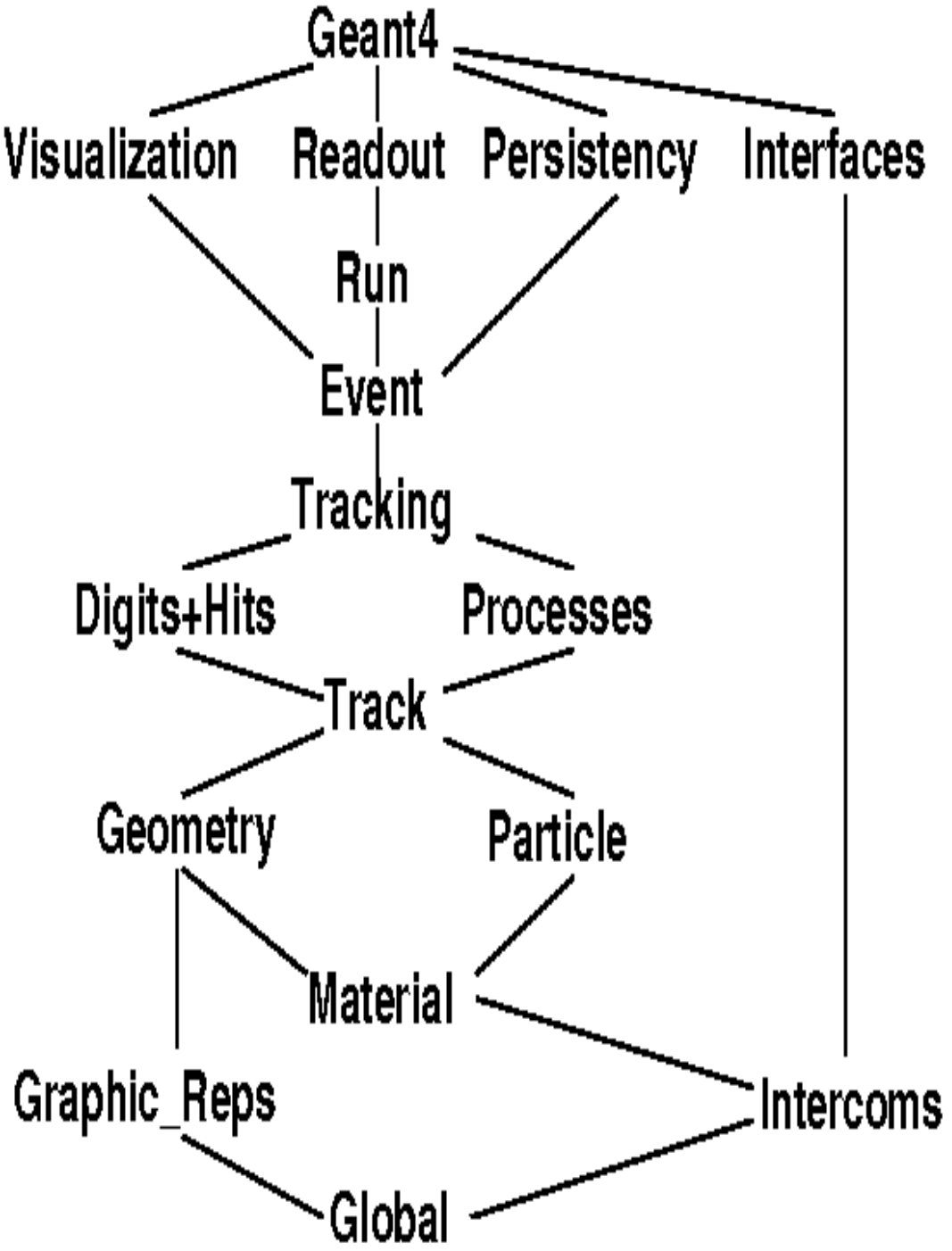
Ядро Geant4

Состоит из 17 категорий классов

Классы, описывающие основные понятия:
сеанс, событие, трек, шаг срабатывание, траектория

Базовый механизм

- описания геометрии
- описания физических процессов
- визуализации и интерфейсов пользователя



Системные требования

Основные платформы и компиляторы

- Linux + gcc
- Windows + VisualC++
- MacOSX + gcc

Внешние пакеты

- CLHEP (начиная с версии 4.9.5 встроен в Geant4)
- cmake
- Xerces-C (опционально, при использовании GDML)
- графические библиотеки (опционально)

Требует около 1.2 Гб дискового пространства (полная установка)

Требует минимум 128 Мб оперативной памяти для работы

Документация и исходный код

http://cern.ch/geant4

Основные публикации

S. Agostinelli et al.,
Geant4: a simulation toolkit, NIM A 506 (2003) 250-303

J. Allison et al.,
Geant4 developments and applications
IEEE Trans. Nucl. Sci. 53 No. 1 (2006) 270-278

Установка на Ubuntu

- Установка средств разработки
  - *sudo apt-get install cmake cmake-curses-gui g++*
- Загрузка исходного кода с http://cern.ch/geant4
  - *wget http://geant4.web.cern.ch/geant4/source/geant4.10.01.p02.tar.gz*
  - *tar xzvf geant4.10.01.p02.tar.gz*
- Установка внешних зависимостей
  - *sudo apt-get install libexpat-dev libqt4-dev libxmu-dev*
- Конфигурирование с помощью cmake
  - *cd geant4.10.01.p02*
  - *mkdir build*
  - *cd build*
  - *ccmake ../* **OR** *cmake -D"GEANT4_INSTALL_DATA=ON" -D"GEANT4_USE_OPENGL_X11=ON" -D"GEANT4_USE_QT=ON" -D"CMAKE_INSTALL_PREFIX=/home/alexey/geant4.10.01.p02" ../*
- Сборка библиотек Geant4
  - *make install*
- Установка переменных окружения (не забывайте вызывать в каждой новой сессии)
  - *source ~/geant4.10.01.p02/bin/geant4.sh*

Если вам больше нравится MS Windows?

- Вариант 1: Использовать уже собранные библиотеки Geant4 с http://cern.ch/geant4 и компилятор MS VisualStudio VC++ 14.0
- Вариант 2:
  - Установите виртуальную машину с http://virtualbox.org
  - Установите ОС Ubuntu на виртуальную машину
  - Установите Geant4 по уже известному рецепту
  - Бонус: Вы можете легко использовать образ виртуальной машины для расчетовна облачных ресурсах (например, Amazon EC2)

Общие замечания

Основные типы переменных

- Для достижения переносимости кода в Geant4 переопределены основные типы переменных

  G4int, G4long, G4float, G4double,

  G4bool, G4complex, G4String

- Лучше использовать переопределенные типы

Ввод и вывод в коде Geant4

Можно использовать обычные printf() и cout

Однако более правильно использовать переопределенные потоки Geant4:

G4cout << *“test”* << *G4endl*;

G4cerr << *“error”* << *G4endl*;

При этом будет корректно обрабатываться ввод-вывод в случае, если интерфейс пользователя отличен от командной строки

Библиотека CLHEP
Class Library for High Energy Physics

- Содержит описание стандартных математических объектов, часто используемых в ФВЭ
  - 3-векторы и 4-векторы
  - действия с матрицами
  - геометрические объекты и преобразования
  - генераторы случайных чисел
  - система единиц и основные физические константы
- Широко используется в коде Geant4
- Подробное описание

http://cern.ch/clhep

- **G4ThreeVector**
  - трехкомпонентный (x,y,z) вектор и действия с ним
- **G4LorenzVector**
  - четырехкомпонентный (x,y,z,t) вектор
- **G4RotationMatrix**
  - матрица 3x3, определяющая вращение 3-вектора
- **G4LorenzRotation**
  - матрица 4x4, определяющая вращение 4-вектора
- Геометрические объекты и преобразования
  - **G4Plane3D, G4Transform3D, G4Normal3D, G4Point3D, G4Vector3D**

Система единиц Geant4

- Любое число, имеющее размерность, должно быть умножено на соответствующую единицу для перевода во внутреннюю систему единиц Geant4

  length = 10.0 * cm;

  kinetic_energy = 5.0 * GeV;
- Для получения величины в желаемых единицах следует делить число на единицу

  G4cout << eDep / MeV << " [MeV]" << G4endl;
- Все общеупотребительные единицы описаны в Geant4 (CLHEP). При необходимости можно определить свои единицы

Основные понятия Geant4

Сеанс (Run)

- Период набора статистики, в котором не меняются условия проведения эксперимента (параметры пучка, конфигурация и параметры детектора, материал мишени и т.п.)
- В Geant4 – самый крупный элемент моделирования, состоящий из последовательности событий. Во время сеанса описание геометрии и набор физических процессов остаются неизменными.
- Представлен классом G4Run
- Управление осуществляется объектом класса G4RunManager

Событие (Event)

- Единичное независимое измерение физического явления детектором.
- В Geant4 представлено классом G4Event
- G4Event содержит все входные и выходные характеристики (исходные частицы, срабатывания и т.д.) смоделированного события
- G4Event создается объектом класса G4RunManager и передается объекту класса G4EventManager, который осуществляет управление событием

Структура события

- Первичная вершина и первичная частица
- Траектории
- Коллекция срабатываний
- Коллекция оцифрованных сигналов

Кроме того, G4EventManager управляет объектами G4Track соответствующими данному событию, взаимодействуя с объектами классов G4TrackManager и G4StackManager

Трек (Track) и шаг (Step)

- Шаг (G4Step) описывает минимальное продвижение частицы через вещество с учетом различных физических процессов.

- Треки представлены классом G4Track, и содержат информацию о последнем шаге. Кроме того, объекты класса G4Track содержат указатели на объекты, описывающие частицу, текущий физический процесс и текущий геометрический объем.

- Объект G4Track описывает полное продвижение частицы в веществе к моменту обращения к данному объекту

Срабатывание (Hit)

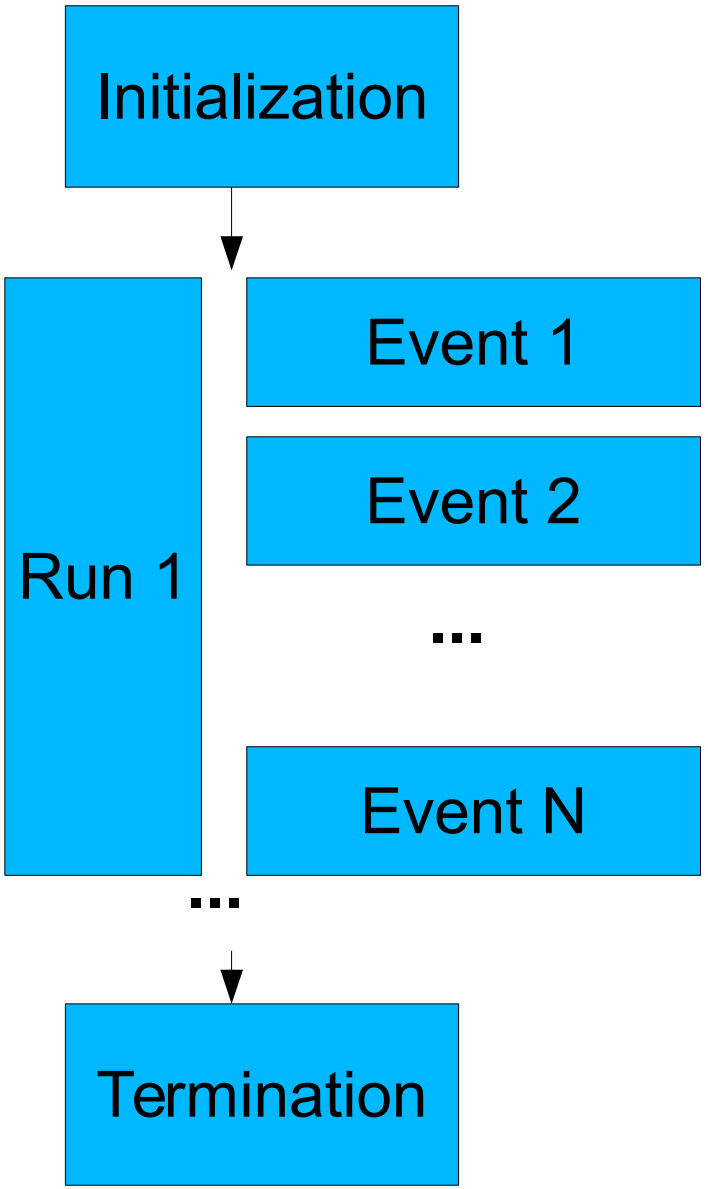
- Описывает единичное взаимодействие частицы с веществом в детектирующем объеме
- Содержит информацию о координате и времени взаимодействия, энергии и импульсе частицы в этой точке, энерговыделении, геометрическую информацию (объем, в котором произошло взаимодействие и т.п.)
- Служит исходной информацией для моделирования оцифрованного сигнала
- Является “истинной” Монте-Карло информацией (Monte-Carlo truth)

Оцифрованный сигнал (Digi)

- Моделируется на основе срабатываний, и содержит информацию в виде “канал-сигнал”
- Один оцифрованный сигнал может быть результатом нескольких срабатываний (например, несколько треков прошли через одну ячейку калориметра)
- Является полным аналогом измеряемых в реальном эксперименте величин
- Будучи необязательным для моделирования, служит в основном для отладки алгоритмов реконструкции

Создание простой программы моделирования

Цикл моделирования



Необходимо описать:

- Распределение вещества в детекторе и поля
- Генератор первичной вершины
- Список физических процессов, учитываемых в моделировании

Кроме того, по желанию:

- Чувствительные элементы и способы моделирования отклика
- Способы визуализации
- Графический интерфейс
- Пользовательские расширения

Простая программа моделирования

```
#include "G4RunManager.hh"
#include "G4UImanager.hh"
#include "ExN01DetectorConstruction.hh"
#include "ExN01PhysicsList.hh"
#include "ExN01PrimaryGeneratorAction.hh"
int main()
{
  // construct the default run manager
  G4RunManager* runManager = new G4RunManager;
  // set mandatory initialization classes
  runManager->SetUserInitialization(new ExN01DetectorConstruction);
  runManager->SetUserInitialization(new ExN01PhysicsList);
  // set mandatory user action class
  runManager->SetUserAction(new ExN01PrimaryGeneratorAction);
  // initialize G4 kernel
  runManager->initialize();
  // get the pointer to the UI manager and set verbosities
  G4UImanager* UI = G4UImanager::GetUIpointer();
  UI->ApplyCommand("/run/verbose 1");
  UI->ApplyCommand("/event/verbose 1");
  UI->ApplyCommand("/tracking/verbose 1");
  // start a run
  int numberOfEvent = 3;
  runManager->BeamOn(numberOfEvent);
  // job termination
  delete runManager;
  return 0;
}
```