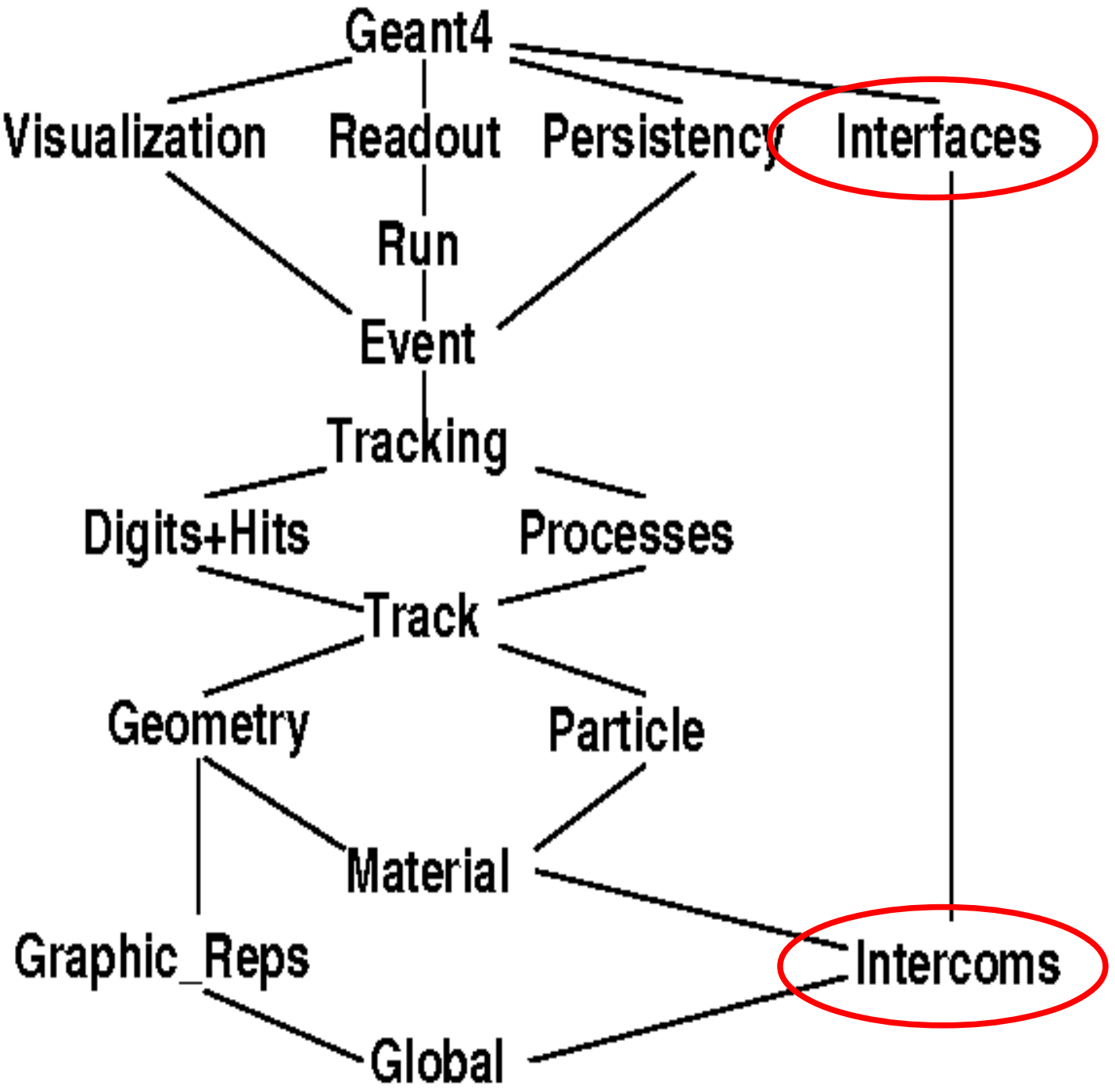
Интерфейс пользователя


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

Взаимодействие пользователя с программой моделирования

- Пакетный режим
- Пакетный режим, управляемый сценарием
- Интерактивный режим с командной строкой
- Интерактивный режим с графическим интерфейсом

Взаимодействие пользователя с программой моделирования

- **Пакетный режим**
- **Пакетный режим, управляемый сценарием**
- **Интерактивный режим с командной строкой**
- **Интерактивный режим с графическим интерфейсом**

```
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();

  // start a run
  int numberOfEvent = 1000;
  runManager->BeamOn(numberOfEvent);

  // job termination
  delete runManager;
  return 0;
}
```

Взаимодействие пользователя с программой моделирования

- **Пакетный режим**
- **Пакетный режим, управляемый сценарием**
- **Интерактивный режим с командной строкой**
- **Интерактивный режим с графическим интерфейсом**

```
int main(int argc,char** argv) {
   // Construct the default run manager
  G4RunManager * runManager = new G4RunManager;

  // set mandatory initialization classes
  runManager->SetUserInitialization(new MyDetectorConstruction);
  runManager->SetUserInitialization(new MyPhysicsList);

  // set mandatory user action class
  runManager->SetUserAction(new MyPrimaryGeneratorAction);

  // Initialize G4 kernel
  runManager->Initialize();

  //read a macro file of commands
  G4UImanager * UI = G4UImanager::getUIpointer();
  G4String command = "/control/execute ";
  G4String fileName = argv[1];
  UI->applyCommand(command+fileName);

  delete runManager;
  return 0;
}
```

Пример файла сценария

```
#
# Macro file for "myProgram.cc"
#
# set verbose level for this run
#
/run/verbose      2
/event/verbose    0
/tracking/verbose 1
#
# Set the initial kinematic and run 100 events
# electron 1 GeV to the direction (1.,0.,0.)
#
/gun/particle e-
/gun/energy 1 GeV
/run/beamOn 100
```

Взаимодействие пользователя с программой моделирования

- **Пакетный режим**
- **Пакетный режим, управляемый сценарием**
- **Интерактивный режим с командной строкой**
- **Интерактивный режим с графическим интерфейсом**

```
int main() {
  // Construct the default run manager
  G4RunManager * runManager = new G4RunManager;

  // set mandatory initialization classes
  runManager->SetUserInitialization(new MyDetectorConstruction);
  runManager->SetUserInitialization(new MyPhysicsList);

  // set user action classes
  runManager->SetUserAction(new MyPrimaryGeneratorAction);

  // Initialize G4 kernel
  runManager->Initialize();

  // Define UI terminal for interactive mode
  G4UIsession * session = new G4UIterminal;
  session->SessionStart();
  delete session;

  // job termination
  delete runManager;
  return 0;
}
```

```
jemtchou: ~/geant4/bin/Linux-g++$ ./prog01
*************************************************************
 Geant4 version Name: geant4-08-00-patch-01    (10-February-2006)
                      Copyright : Geant4 Collaboration
                      Reference : NIM A 506 (2003), 250-303
                            WWW : http://cern.ch/geant4
*************************************************************
Idle>
Idle> /tracking/verbose 1
Idle> /control/execute run01.mac
```

Доступные интерактивные сессии

G4UIterminal и G4UItcsh

командная строка (простой терминал и эмулятор tcsh)

G4UIXm, G4UIQt, G4UIQt, G4UIWin32

графический интерфейс с использованием модулей Motif, Qt, Athena или Win32

G4UIGAG и G4UIGainServer

Geant4 Adaptive GUI – графический интерфейс написаный на Java

- *http://erpc1.naruto-u.ac.jp/~geant4/*

Включение в программу моделирования

```
#include "G4UIxxx.hh"
...
G4UIsession* session = new G4UIxxx;
session->SessionStart();
delete session;
```

В случае G4UItcsh

```
G4UIsession* session = new G4UIterminal(new G4UItcsh);
```

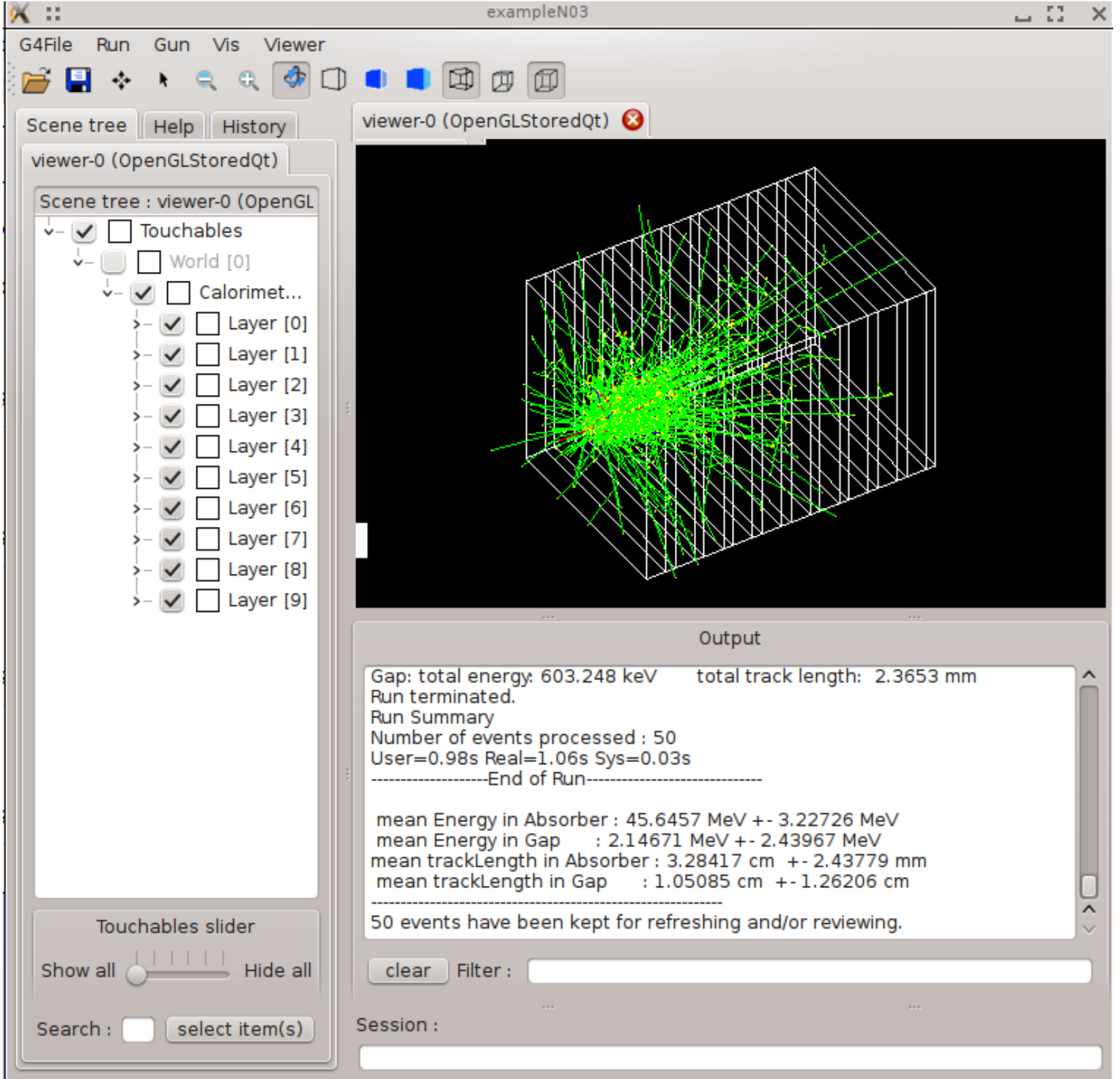
Задание графического интерфейса – команды /gui

```
/gui/addMenu   run Run

/gui/addButton run Init /run/initialize

/gui/addButton run  "Set gun" "/control/execute gun.g4m"

/gui/addButton run  "Run one event"    "/run/beamOn 1"
```


exampleN03
G4File
Run
Gun
Vis
Viewer
Scene tree
Help
History
viewer-0 (OpenGLStoredQt)
Scene tree : viewer-0 (OpenGL
Touchables
World [0]
Calorimet...
Layer [0]
Layer [1]
Layer [2]
Layer [3]
Layer [4]
Layer [5]
Layer [6]
Layer [7]
Layer [8]
Layer [9]
Touchables slider
Show all
Hide all
Search :
select item(s)
viewer-0 (OpenGLStoredQt)
Output
Gap: total energy: 603.248 keV total track length: 2.3653 mm
Run terminated.
Run Summary
Number of events processed : 50
User=0.98s Real=1.06s Sys=0.03s
----------End of Run----------
mean Energy in Absorber : 45.6457 MeV +- 3.22726 MeV
mean Energy in Gap : 2.14671 MeV +- 2.43967 MeV
mean trackLength in Absorber : 3.28417 cm +- 2.43779 mm
mean trackLength in Gap : 1.05085 cm +- 1.26206 cm
50 events have been kept for refreshing and/or reviewing.
clear
Filter :
Session :

Управление программой моделирования

- Geant4 имеет встроенный набор команд, управляющих моделированием
  - уровень диагностики
  - изменение геометрических параметров модели
  - изменение списка учитываемых процессов
  - ...
- Этот набор может быть расширен пользователем
- Команды могут использоваться в интерактивном режиме, считываться из файла сценария или быть запрограммированы в коде

Категории команд

/control	управление интерфейсом
/units	система единиц
/geometry	навигация и проверка перекрытий объемов
/tracking	управление объектами TrackingManager и SteppingManager
/event	управление объектом EventManager
/cuts	управление порогами рождения частиц
/run	управление сеансом
/random	управление генератором случайных чисел
/material	просмотр списка и свойств материалов
/particle	изменение списков и свойств частиц
/process	изменение списка активных процессов
/vis	визуализация
/gun	управление генератором первичной вершины
/hits	управление детектирующими объемами
/score	работа со счетчиками

/control

1) **execute** * Execute a macro file.
2) **loop** * Execute a macro file more than once.
3) **foreach** * Execute a macro file more than once.
4) **suppressAbortion** * Suppress the program abortion caused by G4Exception.
5) **verbose** * Applied command will also be shown on screen.
6) **saveHistory** * Store command history to a file.
7) **stopSavingHistory** * Stop saving history file.
8) **alias** * Set an alias.
9) **unalias** * Remove an alias.
10) **listAlias** * List aliases.
11) **shell** * Execute a (Unix) SHELL command.
12) **manual** * Display all of sub-directories and commands.
13) **createHTML** * Generate HTML files for all of sub-directories and commands.
14) **maximumStoredHistory** * Set maximum number of stored UI commands.

/control/execute

Выполнение сценария:

/control/execute имя_файла_сценария

Например

/control/execute ~/run.mac

/control/shell

Выполнение команды оболочки Unix

/control/shell команда

Например

/control/shell ls -la

/units/list

Вывод списка используемых единиц

Пример:

Idle> /units/list

----- The Table of Units -----

category: Length
parsec (pc) = 3.08568e+19
kilometer (km) = 1e+06
meter (m) = 1000
centimeter (cm) = 10
millimeter (mm) = 1
micrometer (mum) = 0.001
nanometer (nm) = 1e-06
angstrom (Ang) = 1e-07
fermi (fm) = 1e-12

category: Surface
kilometer2 (km2) = 1e+12
meter2 (m2) = 1e+06
. . .

/geometry

Два подкаталога
/geometry/navigator
/geometry/navigator/reset - отладка трекинга
/geometry/navigator/verbose

/geometry/test - проверка геометрии на перекрытие объемов
/geometry/test/tolerance (по умолчанию 0.1 микрон)
/geometry/test/position
/geometry/test/direction
/geometry/test/run
/geometry/test/line_test /geometry/test/recursive_test
/geometry/test/grid_test
/geometry/test/cylinder_test

/tracking

/tracking/abort - прервать моделирование текущего трека

/tracking/resume - возобновить моделирование текущего трека

/tracking/storeTrajectory – сохранять траектории частиц

/tracking/verbose – уровень диагностики трекинга

/tracking/verbose

/tracking/verbose уровень_детализации

Уровни детализации диагностики

0 : Сообщения отсутствуют
1 : Минимальная информация о каждом шаге
2 : Уровень=1 + информация о вторичных частицах
3 : Дополнительно информация о состоянии в начале и конце каждого шага
4 : Дополнительно информация о состоянии в начале и конце каждого шага с детализацией по отдельным процессам
5 : Дополнительно информация об оценках длины шага согласно каждому отдельному процессу

/tracking/storeTrajectory

/tracking/storeTrajectory 0 - траектории частиц не сохраняются

/tracking/storeTrajectory 1 – траектории частиц сохраняются и могут быть визуализированы

В случае если частиц очень много (например, моделируется адронный ливень), может наступить переполнение оперативной памяти.

/event

/event/stack/status – вывести подробную информацию о стеках треков в событии

/event/stack/clear - сбросить стеки

/event/abort - прекратить моделирование текущего события

/event/verbose - диагностика

- /event/verbose 0 - нет сообщений
- /event/verbose 1 - вывод информации о стеках

/run

Подкаталог :

1) **/run/particle/** управление списком физических процессов

Команды :

2) **initialize** * Инициализация ядра Geant4
3) **beamOn** * Начало сеанса
4) **verbose** * Уровень диагностики объекта G4RunManager.
5) **dumpRegion** * информация об объектах G4Region
6) **dumpCouples** * вывод таблицы соответствия “материал-порог”
7) **optimizeGeometry** * флаг оптимизации геометрии
8) **breakAtBeginOfEvent** * пауза в начале каждого события
9) **breakAtEndOfEvent** * пауза в конце каждого события
10) **abort** * прервать текущий сеанс
11) **abortCurrentEvent** * прервать текущее событие
12) **geometryModified** * заново создать модель детектора
13) **physicsModified** * пересчитать таблицы физических процессов
14) **cutoffModified** * - не применяется

/run (продолжение)

15) **randomNumberStatusDirectory** * каталог для сохранения файла с начальным значением генератора случайных чисел
16) **storeRandomNumberStatus** * установить сохранение начального значения генератора в начале сеанса и в начале события
17) **restoreRandomNumberStatus** * считать начальное значение генератора из файла
18) **setCut** * установить значение порога по умолчанию
19) **setCutForRegion** * установить значение порога для объекта G4Region

/run/beamOn

Основная команда при моделировании

Примеры
/run/beamOn 1000 - Начать моделирование 1000 событий

/run/beamOn 1000 event.mac 100
Начать моделирование 1000 событий, и после каждого из первых 100 событий выполнить сценарий event.mac

/random

Действия с начальным значением генератора случайных чисел

/random/setDirectoryName [fileName]

/random/setSavingFlag [flag]

/random/saveThisRun

/random/saveThisEvent

/random/resetEngineFrom [fileName]

/material

/material/nist/printElement

аргумент – символ элемента либо all

/material/nist/printElementZ

аргумент - Z

/material/nist/listMaterials

аргумент- simple, compound, hep,all

/material/g4/printElement

/material/g4/printMaterial

/material/verbose

Пример

```
Idle> /material/nist/printElementZ   12
/material/nist/printElementZ 12
Nist Element: <Mg>  Z= 12  Aeff(amu)= 24.305  18 isotopes:
            N: 20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35
36  37
        mass(amu): 18647.5 19572.3 20492.5 21418.9 22341.9 23274.1 24202.6
25135.7 26066.8 27002.7 27936 28873.1 29807.1 30744.5 31679.3 32618.6
33554.7 34494.3
    abanbance: 0 0 0 0 0.7899 0.1 0.1101 0 0 0 0 0 0 0 0 0 0 0

Idle> /material/nist/printElement   Mg
/material/nist/printElement Mg
Nist Element: <Mg>  Z= 12  Aeff(amu)= 24.305  18 isotopes:
          N: 20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37
        mass(amu): 18647.5 19572.3 20492.5 21418.9 22341.9 23274.1 24202.6
25135.7 26066.8 27002.7 27936 28873.1 29807.1 30744.5 31679.3 32618.6
33554.7 34494.3
    abanbance: 0 0 0 0 0.7899 0.1 0.1101 0 0 0 0 0 0 0 0 0 0 0
```

Еще пример

```
Idle> /material/nist/listMaterials hep
/material/nist/listMaterials hep
=============================================================
###          HEP & Nuclear Materials                 ##
=============================================================
 Ncomp Name  ChFormula       density(g/cm^3)  I(eV)
=============================================================
1  G4_lH2  0.0708    21.8
1  G4_lN2  0.807     82
1  G4_lO2  1.141     95
1  G4_lAr  1.396     188
1  G4_lKr  2.418     352
1  G4_lXe  2.953     482
3  G4_PbWO4  8.28    0
        8     0.140637
        82    0.455366
        74    0.403998
1  G4_Galactic  1e-25    21.8
=============================================================
```

Самые главные команды

Idle> help
Command directory path : /
 Sub-directories :
 1) /control/ UI control commands.
 2) /units/ Available units.
.....
Type the number (0:end, -n:n level back) :
0
Exit from HELP.

Idle>exit

Как добавить свои команды (I)

Создается класс — Messenger:

```
class ExN03DetectorMessenger: public G4UImessenger

{
  public:
    ExN03DetectorMessenger(ExN03DetectorConstruction* );
   ~ExN03DetectorMessenger();
    void SetNewValue(G4UIcommand*, G4String);

  private:
    ExN03DetectorConstruction* ExN03Detector;
    G4UIdirectory*                  N03Dir;
    G4UIdirectory*                  detDir;
    G4UIcmdWithAString*             AbsMaterCmd;
    G4UIcmdWithADoubleAndUnit* AbsThickCmd;
    G4UIcmdWithAnInteger*           NbLayersCmd;
    G4UIcmdWithADoubleAndUnit* MagFieldCmd;
    G4UIcmdWithoutParameter*    UpdateCmd;                    ...
};
```

Как добавить свои команды (II)

Создаются объекты - «команды»:

```
ExN03DetectorMessenger::ExN03DetectorMessenger( ExN03DetectorConstr
uction* ExN03Det):ExN03Detector(ExN03Det)

{
  N03Dir = new G4UIdirectory("/N03/");
  N03Dir->SetGuidance("UI commands of this example");

  detDir = new G4UIdirectory("/N03/det/");
  detDir->SetGuidance("detector control");

  NbLayersCmd = new G4UIcmdWithAnInteger("/N03/det/setNbOfLayers",this);

  NbLayersCmd->SetGuidance("Set number of layers.");

  NbLayersCmd->SetParameterName("NbLayers",false);

  NbLayersCmd->SetRange("NbLayers>0 && NbLayers<500");

  NbLayersCmd->AvailableForStates(G4State_PreInit,G4State_Idle);
   ...
}
```

Как добавить свои команды (III)

Обработка команд:

```
void ExN03DetectorMessenger::SetNewValue(G4UIcommand*
command,G4String newValue)
{
 if( command == NbLayersCmd )
   { ExN03Detector->SetNbOfLayers(NbLayersCmd->GetNewIntValue(newValue));}


  if( command == UpdateCmd )
   { ExN03Detector->UpdateGeometry(); }

  if( command == MagFieldCmd )
   { ExN03Detector->SetMagField(MagFieldCmd->GetNewDoubleValue(newValue));}
...
}
```

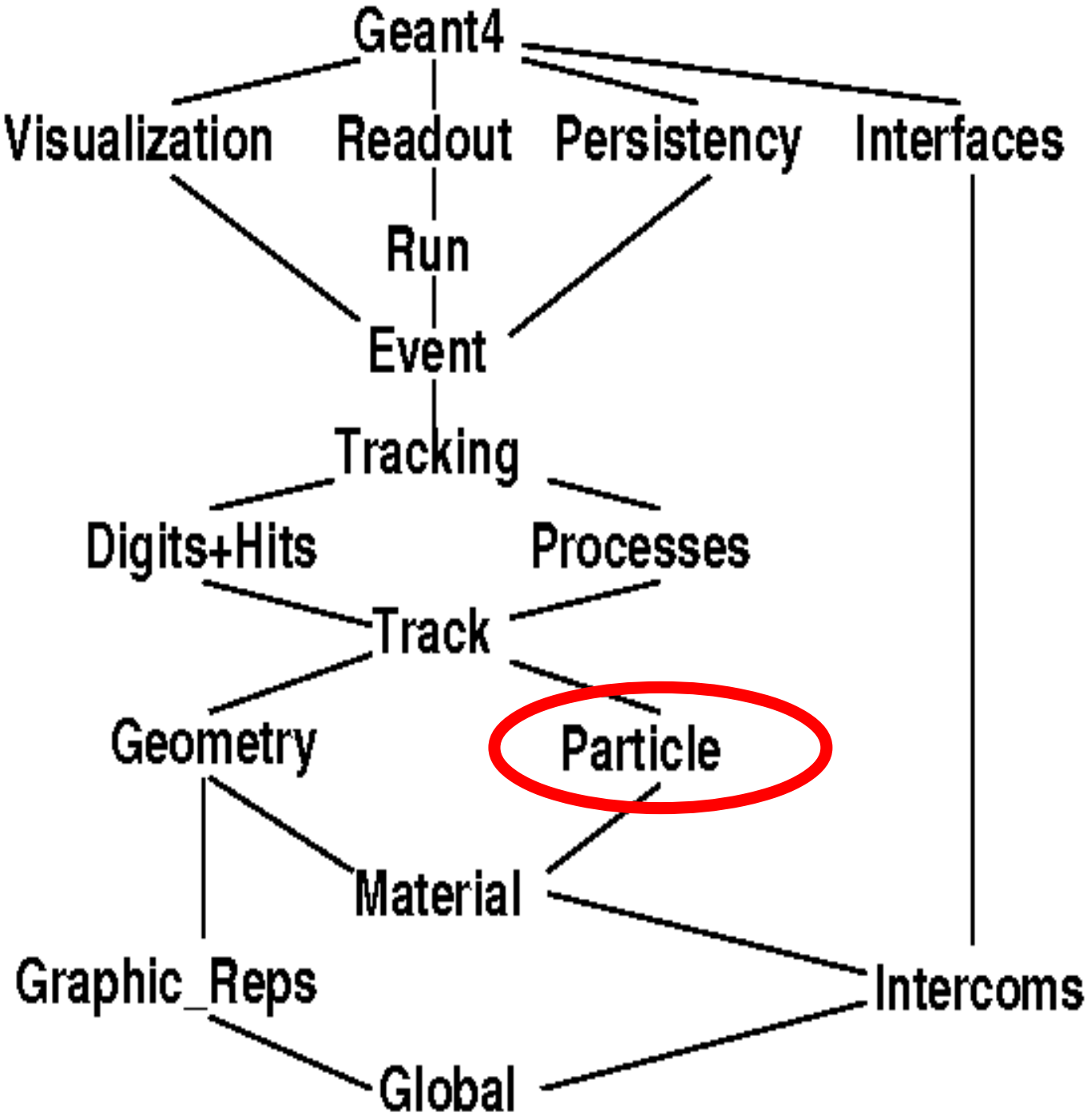
Как добавить свои команды (IV)

Idle> /N03/det/update
/N03/det/update

---> The calorimeter is 10 layers of: [10mm of Lead + 5mm of liquidArgon]

Obsolete world logical volume is removed from the default region.
World is registered to the default region.
...

Описание элементарных частиц


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

Описание элементарных частиц

Каждая частица — это отдельный класс C++ (кроме ионов)

Каждая частица описывается в три этапа:

- **G4ParticleDefinition** - описание постоянных свойств частицы (масса, заряд, название ...)
- **G4DynamicParticle** – описание свойств, изменяющихся при взаимодействии с веществом (энергия, импульс)
- **G4Track** – описание движения частицы в пространстве

Методы G4ParticleDefinition

G4String	GetParticleName()	название
G4double	GetPDGMass()	масса
G4double	GetPDGWidth()	ширина распада
G4double	GetPDGCharge()	заряд
G4double	GetPDGSpin()	спин
G4int	GetPDGiParity()	четность
G4int	GetPDGiConjugation()	зарядовое сопряжение
G4double	GetPDGIsospin()	изоспин
G4double	GetPDGIsospin3()	I_3
G4int	GetPDGiGParity()	G-четность
G4String	GetParticleType()	описание частицы
G4String	GetParticleSubType()	краткое описание частицы
G4int	GetLeptonNumber()	лептонный заряд
G4int	GetBaryonNumber()	барионный заряд
G4int	GetPDGEncoding()	код частицы согласно PDG
G4int	GetAntiPDGEncoding()	код соотв. античастицы

PDG encoding

$$\pm\, n\, n_r\, n_L\, n_{q1}\, n_{q2}\, n_{q3}\, n_J$$

- частицы - '+', античастицы - '-'
- кварки и лептоны пронумерованы от 1 до 11
- n - >0 для частиц за пределами СМ (SUSY, technicolor etc)
- n_r - радиально возбужденные мезоны
- n_L - орбитальный угловой момент
- $n_{q1}\, n_{q2}\, n_{q3}$ - кварковый состав
- $n_J = 2J+1$ - спин

Подробнее по ссылке

http://pdg.lbl.gov/2016/reviews/rpp2016-rev-monte-carlo-numbering.pdf

Основные коды PDG

22 - гамма-квант
11 - электрон
-11 - позитрон
2212 - протон
2112 - нейтрон
+-100ZZZAAAI - ион
например
1000010020 - дейтрон
1000010030 - тритон
1000020040 - альфа
1000020030 - He3

Категории частиц

- **Частицы, участвующие в трекинге**
  - стабильные частицы (протон, электрон, фотон ...)
  - долгоживущие (>10^{-14} с) частицы (пион, мюон ...)
  - короткоживущие частицы, распад которых моделируется в Geant4 (π^0 ...)
  - K-мезоны
  - оптические фотоны
  - geantino
- **Ядра атомов**
  - легкие ядра (дейтрон, альфа-частица, ядро трития)
  - тяжелые ионы
- **Короткоживущие частицы**
  - кварки
  - глюоны
  - мезонные и барионные резонансы

в трекинге не участвуют
появляются только в некоторых моделях физических процессов

Что такое geantino?

Нейтральная частица, не имеющая физического аналог, не участвующая в физических взаимодействиях, и используемая для отладки транспортировки через объемы детектора.

Существует также заряженное geantino, которое кроме транспортирования через объемы может взаимодействовать с электромагнитным полем.

Генераторы первичной вершины

```
#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;
}
```

Класс UserPrimaryGeneratorAction

- Должен быть наследником **G4VUserPrimaryGeneratorAction**
- Должен содержать объект-наследник класса G4VPrimaryGenerator (например G4ParticleGun)
- Должен содержать описание метода *generatePrimaries*(), в котором происходит вызов метода G4VPrimaryGenerator::generatePrimaryVertex(), создающего первичную вершину
- В одном событии вершина может создаваться при участии нескольких объектов-наследников G4VPrimaryGenerator одновременно

```
#ifndef ExN01PrimaryGeneratorAction_h
#define ExN01PrimaryGeneratorAction_h 1
#include "G4VUserPrimaryGeneratorAction.hh"
class G4ParticleGun;
class G4Event;
class ExN01PrimaryGeneratorAction :
                              public G4VUserPrimaryGeneratorAction
{
  public:
    ExN01PrimaryGeneratorAction();
    ~ExN01PrimaryGeneratorAction();
  public:
    void generatePrimaries(G4Event* anEvent);
  private:
    G4ParticleGun* particleGun;
};
#endif
```

```
ExN01PrimaryGeneratorAction::ExN01PrimaryGeneratorAction()
{
  G4int n_particle = 1;
  particleGun = new G4ParticleGun(n_particle);
  particleGun->SetParticleDefinition(G4Geantino::GeantinoDefinition());
  particleGun->SetParticleEnergy(1.0*GeV);
  particleGun->SetParticlePosition(G4ThreeVector(-2.0*m,0.0*m,0.0*m));
}
ExN01PrimaryGeneratorAction::~ExN01PrimaryGeneratorAction()
{
    delete particleGun;
}
void ExN01PrimaryGeneratorAction::generatePrimaries(G4Event* anEvent)
{
    particleGun->SetParticleMomentumDirection(G4ThreeVector(1.0,0.0,0.0));
    particleGun->generatePrimaryVertex(anEvent);
}
```

G4ParticleGun

- Производит первичную вершину из одной или нескольких частиц с заданными импульсом и начальным положением в пространстве
- Может управляться интерактивно
- Методы:

void SetParticleDefinition(G4ParticleDefinition*)
void SetParticleMomentum(G4ParticleMomentum)
void SetParticleMomentumDirection(G4ThreeVector)
void SetParticleEnergy(G4double)
void SetParticleTime(G4double)
void SetParticlePosition(G4ThreeVector)
void SetParticlePolarization(G4ThreeVector)
void SetNumberOfParticles(G4int)

Интерактивное управление генератором – команды /gun/

- /gun/List показать список частиц
- /gun/particle задать тип частицы
- /gun/direction установить направление вылета
- /gun/energy **установить кинетическую энергию**
- /gun/position установить координаты вершины
- /gun/time установить начальное время
- /gun/polarization задать поляризацию
- /gun/number задать число первичных частиц
- /gun/ion задать свойства иона

Более общий случай: G4GeneralParticleSource

- Интерфейс аналогичен G4ParticleGun
- Встроенные генераторы распределений:
  - Спектр: линейниый, экспоненциальный, степенная функция, нормальное распределение, спектр черного тела, кусочная аппроксимация данных.
  - Угловые распределения: пучок, изотропное, cos(theta), определяемое пользователем
  - Пространственное распределение: на простых 2D или 3D поверхностях (диски, сферы, параллелепипеды)
  - В одном сеансе могут быть использованы несколько источников частиц

Моделирование нескольких начальных частиц в вершине

```
 void PrimaryGeneratorAction::GeneratePrimaries(G4Event* evt)
{
  ...
  G4PrimaryVertex* vertex = new G4PrimaryVertex(position, ptime);

  G4PrimaryParticle* p =
      new G4PrimaryParticle(particle);
  p->SetKineticEnergy( energy );
  p->SetMass( mass );
  p->SetMomentumDirection( direction );
  p->SetCharge( 0. );
  p->SetPolarization(0., 0., 0.);
  vertex->SetPrimary( p );

  evt->AddPrimaryVertex( vertex );
}
```

Интерфейсы к внешним генераторам

- Кроме G4ParticleGun, в Geant4 существуют интерфейсы к данным в общепринятых форматах
  - **HEPEVT – de facto стандарт выходных данных для генераторов написанных на языке FORTRAN**
  - **HepMC – аналог HEPEVT для генераторов, написанных на C++**

HEPEVT

```
  PARAMETER (NMXHEP=4000)
  COMMON/HEPEVT/NEVHEP,NHEP,ISTHEP(NMXHEP),IDHEP(NMXHEP),
&JMOHEP(2,NMXHEP),JDAHEP(2,NMXHEP),PHEP(5,NMXHEP),VHEP(4,NMXHEP)
  DOUBLE PRECISION PHEP, VHEP
```

Поля записи HEPEVT

- **NMXHEP** – ***максимальное число частиц в событии***
- **NEVHEP** – ***номер события***
- **NHEP** – ***число частиц в событии***
- **ISTHEP(IHEP)** – ***статус частицы (1-конечное состояние, 2 – короткоживущая распавшаяся и т.д)***
- **IDHEP(IHEP)** – ***PDG код частицы***
- **JMOHEP(1,IHEP)** – ***индекс родительской частицы***
- **JDAHEP(1,IHEP)** – ***индекс первой дочерней частицы***
- **JDAHEP(2,IHEP)** – ***индекс последней дочерней частицы***
- **PHEP(1-5,IHEP)** – ***4-импульс и масса частицы***
- **VHEP(1-4,IHEP)** – ***координаты вершины и время рождения частицы***

Интерфейс Geant4 для HEPEVT

- Geant4 может читать данные в формате HEPEVT из ASCII файла, используя класс **G4HEPEvtInterface**
- Процедура моделирования с использованием генератора, написанного на языке FORTRAN и записи HEPEVT, выглядит так:
  - **программа-генератор событий на языке FORTRAN дополняется кодом, сохраняющим содержимое COMMON BLOCK HEPEVT в ASCII файл**
  - **Geant4 считывает данные из файла, используя G4HEPEvtInterface**

Пример кода для сохранения записи HEPEVT в ASCII файл

```
*************************************************************
      SUBROUTINE HEP2G4
*
* Convert /HEPEVT/ event structure to an ASCII file
* to be fed by G4HEPEvtInterface
*
*************************************************************
      PARAMETER (NMXHEP=2000)
      COMMON/HEPEVT/NEVHEP,NHEP,ISTHEP(NMXHEP),IDHEP(NMXHEP),
     >JMOHEP(2,NMXHEP),JDAHEP(2,NMXHEP),PHEP(5,NMXHEP),VHEP(4,NMXHEP)
      DOUBLE PRECISION PHEP,VHEP
*
      WRITE(6,*) NHEP
      DO IHEP=1,NHEP
       WRITE(6,10)
     >  ISTHEP(IHEP),IDHEP(IHEP),JDAHEP(1,IHEP),JDAHEP(2,IHEP),
     >  PHEP(1,IHEP),PHEP(2,IHEP),PHEP(3,IHEP),PHEP(5,IHEP)
10     FORMAT(4I10,4(1X,D15.8))
      ENDDO
*
      RETURN
      END
```

```
#include "G4HEPEvtInterface.hh"

ExN04PrimaryGeneratorAction::ExN04PrimaryGeneratorAction()
{
  HEPEvt = new G4HEPEvtInterface("pythia_event.data");
}

ExN04PrimaryGeneratorAction::~ExN04PrimaryGeneratorAction()
{
  delete HEPEvt;
}

void ExN04PrimaryGeneratorAction::
                                        GeneratePrimaries(G4Event* anEvent)
{
  HEPEvt->SetParticlePosition(G4ThreeVector(0.*cm,0.*cm,0.*cm));
  HEPEvt->GeneratePrimaryVertex(anEvent);
}
```

HepMC

- Набор классов, написанный на C++, и содержащий ту же информацию, что и запись HEPEVT (с некоторыми расширениями)

- Полное описание см. **http://cern.ch/hepmc**

- Пример: extended/eventgenerator/HepMC