МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ имени М. В. ЛОМОНОСОВА

Научно-исследовательский институт ядерной физики имени Д.В. Скобельцына

В. В. Леонтьев, И. А. Орлов

Задачи раздела «Информационные методы в физике высоких энергий», часть 2

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ имени М. В. ЛОМОНОСОВА

НАУЧНО-ИССЛЕДОВАТЕЛЬСКИЙ ИНСТИТУТ
ЯДЕРНОЙ ФИЗИКИ имени Д. В. СКОБЕЛЬЦЫНА

В. В. Леонтьев, И. А. Орлов

Задачи раздела «Информационные методы в физике высоких энергий», часть 2.

Москва

2013

УДК 004
ББК 74.58с51
ISBN 978-5-

В. В. Леонтьев, И. А. Орлов

Задачи раздела «Информационные методы в физике высоких энергий», часть 2: Описание задач практикумов.

Практикум предназначен для студентов старших курсов физического факультета МГУ. Целью задач описываемого раздела является обучение навыкам работы в современной информационной среде, принятой в физике высоких энергий и адронной физике. Задачи были предложены и созданы на основании собственного опыта работы выпускниками кафедры ФЭЧ, молодыми научными сотрудниками, работающими в лабораториях ОИЯИ.

Оглавление

Предисловие к пособию

Данное пособие содержит текстовые описания задач одного из трех разделов специального физического практикума кафедры физики элементарных частиц, предлагаемого студентам для прохождения в филиале НИИЯФ МГУ, г. Дубна. Описания должны использоваться студентами для самоподготовки перед занятиями, а также как конспект лекционных материалов, читаемых преподавателем во время занятий. Неотъемлемой частью этого курса являются ресурсы, размещенные на сайте кафедры ФЭЧ по адресу [1]. В тексте пособия повсеместно содержатся ссылки на наглядные материалы и коды программ (скриптов), необходимых для работы. Без непосредственного обращения к этим *on-line* ресурсам изучение данного пособия неэффективно.

Пакет Geant4

Введение

Geant4 [2] – это программный пакет для моделирования прохождения частиц через вещество. Несмотря на то, что Geant4 является громоздкой и весьма сложной программой, в большинстве случае всё сводится к простейшей схеме: “есть частицы определённого типа, они попадают в детектирующую систему. Надо узнать, что получается на выходе”. В Geant4 можно контролировать все необходимые для моделирования соответствующих процессов параметры:

- геометрию системы,
- материалы, которые используются в эксперименте,
- фундаментальные физические частицы,

- генерацию первичных событий,
- прокладку трека частиц через различные материалы и электромагнитные поля,
- описание физики процессов, определяющих взаимодействие частиц,
- отклик чувствительных элементов детектирующей системы,
- генерацию и хранение данных о событии (треки, вершины, частицы)
- визуализацию детектора и траекторий частиц.

Geant4 написан на языке C++. По своей сути он является гигантским набором классов, написанным разными людьми в разное время. Кроме стандартного ядра программы, существуют дополнительные пакеты, которые расширяют область применимости Geant4 в более узких областях (описание физических процессов в этих областях не является основной задачей Geant4 и часто имеет более точные аналоги).

Структура программы Geant4.

Для работы Geant4 пользователю необходимо создать несколько собственных классов, унаследованных от базовых. Обязательными для использования являются классы: G4VUserDetectorConstruction (описание геометрии эксперимента), G4VUserPhysicsList (описание физических процессов и частиц, которые необходимо учитывать в данном эксперименте), G4VUserPrimaryGeneratorAction (создание генератора первичных частиц). Кроме того существуют дополнительные классы, которые позволяют контролировать и изменять поведение Geant4 на том или

ином этапе моделирования (G4UserRunAction и др.). Определение каждого класса помещается в соответствующий файл и в последствие сводится воедино в главном файле проекта. Все используемые классы должны быть зарегистрированы в G4RunManager (этот класс отвечает за пуск и остановку моделирования и является главным управляющим классом в программе).

G4VUserDetectorConstruction

Данный класс отвечает за геометрические характеристики моделирования. В нём необходимо:

- создать все необходимые для моделирования материалы, из которых будут состоять тела,
- объявить формы тел, необходимые для описания геометрии, в случае сложных форм их необходимо создать при помощи операций со стандартными формами (вырезать из куба шар и т.п.),
- объявить “чувствительные” области детектора и задать им в соответствие объект класса G4VSensitiveDetector (отвечает за моделирование пролёта частицы через чувствительную область и позволяет получать данные о частице, находящейся в ней)
- задать магнитные поля
- определить параметры визуализации для элементов детектора.

Стоит прояснить особенности геометрического описания в Geant4. Существует три различных типа т.н. “объёмов”:

1. G4VSolid (форма) – класс отвечающий за форму и размеры объёма,
2. G4LogicalVolume (логический объём) отвечает за материал, из которого состоит объём, магнитные поля внутри него, а также определяет является ли данный объём “чувствительным”,
3. G4VPhysicalVolume (физический объём) отвечает за положение объёма в пространстве.

Рис. 1: Иерархия физических объёмов Geant4.

Отдельного упоминания заслуживает иерархия объёмов. Первым делом должен быть создан “мир” – главный объём, соответствующий экспериментальной зоне. Когда создаётся любой другой физический объём, возникает необходимость указания для него “материнского” физического объёма. Координаты расположения дочернего физического объёма задаются относительно центра материнского. Дочерний объём не может выйти за пределы материнского. Программа обрезает выходящие за пределы материнского объёма части, но таких ситуаций лучше не допускать, так как это ошибка геометрии и никто не гарантирует, что в более новых версиях программы это будет работать тем же образом.

Кроме того, материал дочернего объёма заменяет материал материнского. Если два объёма являются дочерним к одному и тому же, и при этом перекрываются, то область перекрытия будет считаться принадлежащей первому созданному объёму (но опять же стоит заметить, что это ошибка геометрии). Любой физический объём, кроме “мира”, должен иметь материнский объём и может иметь дочерние объёмы. Иерархия физических объёмов наглядно представлена на Рис. 1.

G4VUserPhysicsList

По умолчанию в Geant4 не определены частицы и физические процессы, в связи с этим необходимо либо создать свой класс, либо использовать один из классов с изначально заданной физикой.

В случае создания своего класса необходимо:

- сконструировать все необходимые частицы
- сконструировать все необходимые процессы и связать их с частицами
- задать пороги для рождения частиц

Стоит заметить, что пороги для рождения частиц и их последующего моделирования задаются в пересчёте на расстояние. Для каждой частицы или объёма можно задать своё пороговое значение, как это описано ниже в подразделе SensitiveDetector. Тем самым можно получить необходимое соотношение точности расчёта и времени вычислений (чем выше порог, тем меньше время расчёта и меньше точность).

G4VUserPrimaryGeneratorAction

Данный класс отвечает за генерацию первичных частиц. От пользователя требуется задать тип генерируемой

частицы, её начальное положение и импульс. Для этого можно применить различные описанные ниже методы, также можно использовать внешний генератор событий физики высоких энергий.

После описания этих трёх классов программа готова к запуску. Однако стоит сделать замечание о том, как происходит моделирование трека частиц. Трек частицы выглядит как ломаная линия, расстояние между двумя точками (вершинами трека) – шаг моделирования. Его длина зависит от заданных в программе порогов и материала, в котором находиться частица. Текущие параметры частицы могут быть считаны только в вершинах трека, в них же происходит расчет процессов рождения новых частиц, рассеяния и т.д. Моделирование треков частиц происходит по очереди: от начала до конца рассчитывается трек первой частицы, при этом определяются вершины, в которых родились новые частицы, но моделирование их треков начинается только после того, как закончится моделирование первичной. Во время шага моделирования Geant4 определяет, через какие материалы пролетает частица, и в зависимости от этого рассчитывает, сколько энергии потеряла частица из-за взаимодействия с веществом. Теперь перейдём к разбору кода примера.

Пример программы Geant4.

В основной папке проекта, доступного на кафедральном сайте [1], располагаются файлы Prak.cc, cmakefile, vis.mac а также папки include и src. В папках include и src располагаются описания пользовательских классов DetectorConstruction, PhysicsList, PrimaryGeneratorAction, SensitiveDetector (расширения имен файлов данных классов: .hh и .src). Теперь рассмотрим подробнее содержание каждого из перечисленных файлов.

Prak.cc

Prak.cc - главный файл программы, он начинается с подключения заголовочных файлов. Для простых программ этот файл можно оставлять практически без изменений, только добавляя в случае необходимости дополнительные заголовочные файлы, описывающие нужные классы, и регистрируя их в G4RunManager.

```
#include<G4RunManager.hh>
#include<G4UImanager.hh>
#include<G4UIterminal.hh>
#include<G4VisExecutive.hh>
#include<G4Material.hh>
#include<G4UserRunAction.hh>
#include<G4Run.hh>
#include<iostream>
#include<string>
#include<CLHEP/Random/Random.h>
#include<unistd.h>
#include<time.h>
```

Здесь перечислены сначала системные файлы, входящие в состав Geant4 и компилятора, а за ними происходит подключение заголовочных файлов текущего проекта, они задаются не в угловых скобках, а в кавычках. Данные файлы должны быть размещены в папке include.

```
#include "DetectorConstruction.hh"
#include "PrimaryGeneratorAction.hh"
#include "EventAction.hh"
#include "SteppingAction.hh"
#include "SteppingVerbose.hh"
#include "PhysicsList.hh"
```

Далее следует определение функции main(). В языке C++ main() является основной функцией программы, с которой начинается её выполнение. В ней производится инициализация GEANT4 и совершаются все необходимые для выполнения программы действия, о которых упоминалось в предыдущем разделе. Для начала

открывается файл, в который будет выводиться собранная информация:

```
outfile.open("outfile.dat"); //при желании
вывод информации можно сделать при помощи пакета
ROOT,  что может быть удобно, если её необходимо
представить в виде гистограмм.
```

Далее происходит настройка генератора случайных чисел, так как по умолчанию он всегда возвращает одну и ту же их последовательность (что, в принципе, может быть удобно при отладке). Для генерирования более случайных последовательностей необходимо задать зерно (seed), которое в нашем случае задается как сумма текущего времени в секундах и программного идентификатора:

```
CLHEP::HEPRandom::setTheSeed(time(0)+getpid());
```

Далее создаём объект типа G4RunManager, который будет управлять запуском и установкой моделирования:

```
G4RunManager * runManager = new G4RunManager;
```

Создаётся объект DetectorConstruction (используется пользовательский класс DetectorConstruction.cc) и регистрируется в G4RunManager. Эти действия передают программе данные о геометрии эксперимента:

```
DetectorConstruction* detector_c = new
DetectorConstruction;
runManager->SetUserInitialization(detector_c);
```

Схожим образом создаётся и регистрируется в G4RunManager пакет, описывающий физические процессы, PhysicsList.

```
G4VUserPhysicsList *p = new PhysicsList;
runManager->SetUserInitialization(p);
```

Создаётся и инициализируется объект класса G4VisExecutive, который позволяет различными способами визуализировать моделирование, а также пользовательские классы действий регистрируются в G4RunManager.

```
G4VisManager* visManager = new G4VisExecutive;
visManager->Initialize();
runManager->SetUserAction(new
   PrimaryGeneratorAction);
```

В конце концов вызывается метод G4RunManager:Initialize() и процесс инициализации Geant4 завершается.

```
runManager->Initialize();
```

Печать информации о зарегистрированных материалах.

```
cout<<"=====================================";
cout<<endl;
cout<< *(G4Material::GetMaterialTable()) <<
endl;
cout<<"=====================================";
cout<<endl;
```

Наконец, через объект класса G4UImanager производится выполнение макрокоманд из файла vis.mac (на него указывает определенная выше переменная macros). Это удобно, потому что при изменении vis.mac не нужно перекомпилировать всю программу. В vis.mac находятся команды, непосредственно запускающие моделирование.

```
G4UImanager * UI = G4UImanager::GetUIpointer();
G4UIExecutive* ui = new G4UIExecutive(argc, argv);
UI->ExecuteMacroFile(macros);
```

После завершения выполнения команд vis.mac моделирование закончено. Освобождается память и программа закрывается.

```
outfile.close();
delete session;
delete visManager;
delete runManager;
return 0; }
```

Таким образом, как можно заметить, по большей части Prak.cc имеет стандартную структуру и может переноситься из программы в программу с минимальными изменениями.

DetectorConstruction

Объявление класса DetectorConstruction находится в файле DetectorConstruction.hh. В начале файла находится так называемый include guard, необходимый во всех заголовочных файлах.

```
#ifndef DetectorConstruction_h
#define DetectorConstruction_h 1
```

Для удобства определен класс World, который соответствует внешнему объему в геометрии GEANT4, внутри которого находится вся моделируемая установка. Этот объем можно было бы создать и просто через цепочку Solid→Logic→Physic, но можно и инкапсулировать ее в одном классе World.

```
class World {
   protected:
     G4VSolid *solid;
     G4LogicalVolume *logic;
     G4VPhysicalVolume *physic;
     G4Material *mater;
     double sizex, sizey, sizez;
   public:
     World(double size_x, double size_y, double
size_z, G4Material *mater_=NULL);
     operator G4LogicalVolume*() {return logic;}
     G4LogicalVolume *getLogic() {return logic;}
     //    void setLogic(G4LogicalVolume *volA);
     G4VSolid *getSolid() {return solid;}
     G4VPhysicalVolume *getPhysic() {return physic;}
   };
```

Объявление класса DetectorConstruction, наследуемого от G4VUserDetectorConstruction.

```
class DetectorConstruction : public
G4VUserDetectorConstruction{
```

Далее идет объявление конструктора и деструктора класса DetectorConstruction — функций, которые автоматически вызываются при создании и уничтожении

объекта.

```
public:
     DetectorConstruction();
     ~DetectorConstruction();
```

Функция Construct заключает в себе основную функциональность класса DetectorConstruction. Она создает геометрию и материалы.

```
  G4VPhysicalVolume* Construct();
```

Также объявляется защищенная переменная-указатель на World. Она будет проинициализированна позднее.

```
World *world;
```

Определение объявленного класса DetectorConstruction находится в файле **DetectorConstruction.cc.**

Здесь определяется макрос Mat(), просто для того чтобы сократить запись: теперь вместо того чтобы писать G4NistManager::Instance()->FindOrBuildMaterial(«G4_Si») достаточно написать Mat("G4_Si"). При этом будет произведена инициализация соответствующего материала в базе данных GEANT4 (если он в ней содержится, конечно). Список имеющихся в ней материалов можно посмотреть в файле $G4INSTALL/source/material/management/src/G4NISTMaterialBuilder.cc.

```
#define Mat(x) (G4NistManager::Instance()-
>FindOrBuildMaterial(x))
```

Определение конструктора класса World. Он принимает четыре параметра: ширину, высоту, глубину и материал, и создает в качестве материнского объема бокс с этими параметрами через цепочку вызовов Solid→Logic→Physic.

```
World::World(double size_x, double size_y, double
size_z,  G4Material *mater_):  mater(mater_),
sizex(size_x), sizey(size_y), sizez(size_z){
 solid = new G4Box("world", sizex/2, sizey/2,
sizez/2);
 logic = new G4LogicalVolume( solid, mater, "World",
```

```
0, 0, 0);
 physic = new G4PVPlacement(0, G4ThreeVector(),
logic, "World",
 0, false, 0);}
```

Конструктор и деструктор DetectorConstruction пустые, т. к. и сам класс очень простой.

```
DetectorConstruction::DetectorConstruction(){}
DetectorConstruction::~DetectorConstruction(){}
```

Основную работу в DetectorConstruction выполняет функция Construct()

```
G4VPhysicalVolume*
DetectorConstruction::Construct(){
```

Для начала создается материнский объем размером 30x30x30 см., «заполненный» вакуумом.

```
world = new World(30*cm, 30*cm, 30*cm,
Mat("G4_Galactic"));
```

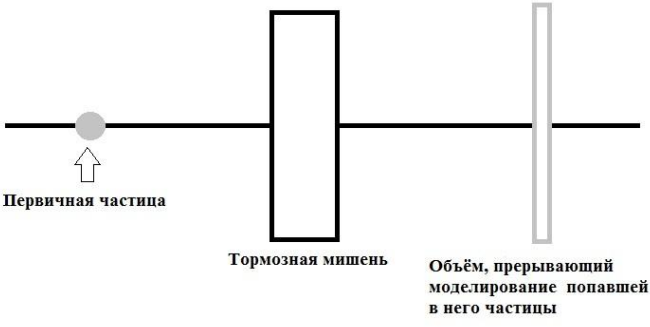


Рис. 2: Схема геометрического описания установки.

Следующий шаг – создание геометрии эксперимента. Часто это наиболее долгий и трудоёмкий шаг, хотя и не самый сложный. В случае с большими установками, подробное описание их геометрии занимает много времени, но может являться повторяющейся механической работой. Геометрия установки, описанная в представленном ниже коде,

изображена на Рис. 2.

Создается тормозная мишень (по аналогии с которой будет необходимо создать регистрирующий сцинтиллятор). Как уже было сказано выше, каждый элемент геометрии GEANT4 задается тремя объемами. На этапе SolidVolume создаваемая в данном примере форма - это G4Box, то есть параллелепипед. Его параметры это половинные размеры.

```
G4Box *solidDet = new G4Box("solidDet", 20*cm,
20*cm, 10*mm);
```

На следующем этапе создается логический объем G4LogicalVolume, который содержит информацию о материале и магнитных свойствах среды. В данном случае никаких магнитных свойств нет.

```
G4LogicalVolume *logicDet = new
G4LogicalVolume(solidDet, Mat("G4_Si"), "logicDet");
```

По умолчанию, любой физический объём, созданный из данного логического, будет отображаться при визуализации с использованием белого цвета. Для того чтобы объём при отображении был окрашен в другой цвет или чтобы он не отображался в Geant4, существуют параметры отображения G4VisAttributes.

```
G4VisAttributes *att_1 = new G4VisAttributes(); //
создание новых параметров отображения
att_1->SetColour(0,0,1); // определение цвета
(R,G,B)
logicDet->SetVisAttributes(att_1); // приписывание
параметров определённому логическому объёму
```

Третий этап — это физический объем, в котором содержится информация о положении объекта относительно материнского, то есть World. Здесь оно задается вектором G4ThreeVector(0,0,20 *cm), то есть 20 см по оси Z.

```
G4PVPlacement *physiDet = new G4PVPlacement(0,
G4ThreeVector(0,0,20*cm), logicDet, "sens_det0",
world->getLogic(), false, 0);
```

Схожим образом создаётся объём, который будет прерывать процесс моделирования частицы при попадании в него. Однако, в данном случае создаётся цилиндр.

```
double innerRadiusOfTheTube = 0.0*mm;
double outerRadiusOfTheTube = 300.0*mm;
double  hightOfTheTube = 1.0*mm;
double startAngleOfTheTube = 0.*deg;
double spanningAngleOfTheTube = 360.0*deg;
G4Tubs *solidStop = new G4Tubs("solidStop",
innerRadiusOfTheTube,
outerRadiusOfTheTube,
hightOfTheTube,
startAngleOfTheTube,
spanningAngleOfTheTube);
G4LogicalVolume *logicStop = new
G4LogicalVolume(solidStop, Mat("G4_Galactic"),
"logicStop");
G4PVPlacement *physiStop = new G4PVPlacement(0,
G4ThreeVector(0,0,35*cm), logicStop,  "sens_det1",
world->getLogic(), false, 0);
```

Для того чтобы детектор реагировал на пролет частиц, он должен быть обозначен, как “чувствительная область”. Для этой цели в G4LogicalVolume есть специальное поле SensitiveDetector, которое содержит указатель на объект класса G4VSensitiveDetector. Этот объект автоматически вызывается каждый раз, когда очередной шаг моделирования частицы попадает внутрь данного объема. Программируя класс SensitiveDetector, можно получать выходные данные моделирования.

```
SensitiveDetector *detector = new
SensitiveDetector("SensDet");
```

Объект SensitiveDetector должен быть зарегистрирован в G4SDManager.

```
G4SDManager* SDman = G4SDManager::GetSDMpointer();
SDman->AddNewDetector(detector);
```

Здесь SensitiveDetector сопоставляется логическим

объёмам, которые были объявлены выше.

```
logicDet->SetSensitiveDetector(detector);
logicStop->SetSensitiveDetector(detector);
```

Стенки объекта World делаются прозрачными, чтобы не мешали на визуализации.

```
    world->getLogic()->SetVisAttributes
(G4VisAttributes::Invisible);
```

В конце концов, успешно выполнившаяся функция Construct должна возвратить указатель на физический объем объекта World.

```
  return world->getPhysic();
```

SensitiveDetector

Объявление класса SensitiveDetector производится в файле SensitiveDetector.hh. За исключением стандартного объявления, нас интересует строчка, описывающая метод класса, отвечающий за обработку частицы, пролетающей через “чувствительный” детектор. Стоит заметить, что при моделировании частица всегда имеет хотя бы 2 вершины трека внутри чувствительного объёма (сразу после того как частица влетает в объём и прямо перед вылетом из него). В то время как обычный объём, она может пролететь насквозь, не оставив там ни одной вершины, если пороги в единицах расстояния велики. Потому, если необходим просто счётчик числа частиц, то необходимо использовать класс SensitiveDetector, позволяющий считывать необходимую информацию. Так как в разных задачах может требоваться получать совершенно различные данные, класс SensitiveDetector, так же как и класс DetectorConstruction, каждый раз переписывается практически полностью. В нашем случае класс SensitiveDetector будет создан таким образом, чтобы считывать информацию о том, сколько энергии и где именно оставила частица при пролёте через

детектор. Описание этого метода находится в файле SensitiveDetector.cc

```
G4bool SensitiveDetector::ProcessHits(G4Step *step,
G4TouchableHistory *hist){
```

Сначала проверяется объём, в который попала частица.

```
G4VPhysicalVolume* volume = step->
GetPreStepPoint()->GetTouchableHandle()->
GetVolume();
```

Если эта частица попала в тормозную мишень, то программа на каждом шаге будет считывать координаты частицы и её энергопотери на данном шаге. После этого она записывает эти данные в файл. Здесь происходит проверка того, что частица находиться именно в тормозной мишени, при помощи проверки имени физического объёма.

```
if(volume->GetName()=="sens_det0"){
X=step->GetTrack()->GetPosition().getX();
Y=step->GetTrack()->GetPosition().getY();
Z=step->GetTrack()->GetPosition().getZ();
edep = step->GetTotalEnergyDeposit();
outfile  << X << "    " << Y << "    "<< Z << "
"<< edep << endl;}
```

Если же частица попадает во второй чувствительный объём, то её моделирование прекращается.

```
if(volume->GetName()=="sens_det1"){
step->GetTrack()->SetTrackStatus(fStopAndKill);}
return true;}
```

PhysicsList

В GEANT4 все используемые в моделировании физические процессы должны быть заранее подключены. Делается это в классе PhysicsList. В нем должны создаваться определения всех частиц и для каждой частицы должны выбираться процессы, которые с ней могут происходить. Но так как создание такого пакета вручную довольно сложное дело, где надо учесть массу деталей, обычно используются

готовые пакеты. В данном примере это стандартный PhysicsList, используемый в большинстве программ на GEANT4. Он включает в себя следующие процессы: для фотонов: 1) фотоэффект; 2) комптоновское рассеяние; 3) рождение пар. Для электронов, позитронов и мюонов: 1) тормозное излучение; 2) множественное рассеяние; 3) ионизация среды; 4) аннигиляция с античастицами.

Кроме того, для всех типов частиц подключается процесс Transportation, отвечающий за перемещение частиц в пространстве с учетом влияния магнитного поля.

Код класса PhysicsList находится в файлах PhysicsList.hh и PhysicsList.cc. Он совершенно стандартный, единственный параметр, который в нем может потребоваться изменить — это так называемый порог (кат). Он задается в PhysicsList.cc:

```
   PhysicsList::PhysicsList():
G4VUserPhysicsList(){
    defaultCutValue = .1*cm;
    SetVerboseLevel(1);}
```

Как видно, это величина с размерностью длины. Смысл ее таков: для каждого типа частиц — электронов, фотонов и т. д.,— и для каждого материала вычисляется энергия, при которой эта величина совпадает со средним пробегом в данной среде. Затем, в ходе моделирования, если на некотором шаге должна родиться вторичная частица, но ее энергия меньше, чем пороговая энергия, то эта частица не рождается, а считается поглощенной в среде в данной точке. При этом ее энергия добавляется к TotalEnergyDeposit. Например, если в тормозной мишени из вольфрама фотон с энергией 10 МэВ рождает пару электрон-позитрон, а величина ката равна 1 мм, то эти частицы не будут рождаться, т. к. энергия каждой 5 МэВ и их средний пробег в вольфраме меньше 1 мм. Понятно, что, во-первых, кат

должен быть меньше или равен характерной толщине моделируемого объекта, а во-вторых, что значение порога влияет, прежде всего, на низкоэнергетическую часть спектра.

PrimaryGeneratorAction

Этот класс, который часто приходиться изменять, в зависимости от поставленной задачи, так как именно он определяет то, какие первичные частицы будут моделироваться.

Конструктор класса. Здесь удобно создать G4ParticleGun и настроить его на определенный тип частиц. В данном случае это электроны с энергией 50 МэВ, а точка старта -10 см по оси Z. Кроме того, задается направление импульса вдоль Z, а сам импульс вычисляется автоматически.

```
PrimaryGeneratorAction::PrimaryGeneratorAction(){
   G4int n_particle = 1;
   particleGun = new G4ParticleGun(n_particle);
   G4ParticleTable* particleTable =
       G4ParticleTable::GetParticleTable();
   G4ParticleDefinition* particle = particleTable-
>FindParticle("e-");
   particleGun->SetParticleDefinition(particle);
   particleGun->
SetParticleMomentumDirection(G4ThreeVector(0., 0.,
1.));
   particleGun->SetParticlePosition(G4ThreeVector(0,
0, -10*cm));
   particleGun->SetParticleEnergy(50*MeV);}
```

В деструкторе G4ParticleGun удаляется.

```
PrimaryGeneratorAction::~PrimaryGeneratorAction(){
delete particleGun;}
```

GeneratePrimaries — это основной метод класса PrimaryGeneratorAction. Она вызывается в начале каждого события, чтобы сгенерировать первичные частицы, с которых начинается моделирование.

Параметры частиц были раз и навсегда заданы в конструкторе, а здесь производится запуск ParticleGun.

```
void
PrimaryGeneratorAction::GeneratePrimaries( G4Event*
anEvent){
particleGun->GeneratePrimaryVertex(anEvent);}
```

В случае, когда необходимо менять параметры частиц прямо во время работы программы, настройку параметров частиц лучше перенести в метод GeneratePrimaries. Допустим, если существует необходимость сгенерировать 50 фотонов и 50 протонов, то внутри метода можно сделать проверку. Класс G4Event имеет метод eventID, который возвращает номер события (счёт начинается с нуля), потому условием, при котором фотон сменится на электрон, можно поставить `eventID > 49`.

Компиляция и запуск

Компиляция программы управляется при помощи служебного файла типа cmakefile. Его можно разделить на две части, первая используется для определения местоположения пакета ROOT [3]. Хотя данной программой пакет ROOT напрямую не используется, он является чрезвычайно удобным при работе с Geant4. Поэтому большой кусок текста был добавлен, чтобы облегчить работу в том случае, если ROOT потребуется подключить. Для полной сборки программы необходимо создать папку (пусть название будет Prak-build) и, перейдя в неё, выполнить команды:

```
“cmake -Dgeant4_DIR=/your_path/geant4/geant4.9.6-
install/lib64/Geant4-9.6.0 /your_path/g4work/Prak”
“make”
```

Эти команды соберут программу и создадут исполняемый файл Prak, который запускается стандартным образом (./Prak).

Файл vis.mac

Как уже упоминалось, работа по запуску моделирования перенесена в файл vis.mac, который обрабатывается в ExampleG4.cc. Этот файл содержит команды для макропроцессора GEANT4. Удобство такого подхода заключается в том, что можно, во-первых, обойтись без перекомпиляции всего проекта в случае незначительных изменений (это открывает дорогу к написанию всевозможных скриптов), а во-вторых, язык макропроцессора допускает расширение и определение новых команд, исходя из требований конкретных задач. Каждая строка vis.mac является командой, и может иметь параметры. Имена команд выглядят как имена файлов и содержат «путь», указывающий на их функциональную принадлежность. Комментарии начинаются с #.

Выбор драйвера визуализации. OGLIX -- это OpenGL, рисование треков в окне. Если закомментировать эту строчку, визуализация OpenGL будет отключена и программа будет считать существенно быстрее.

```
/vis/open OGLIX
/vis/drawVolume
#Настройка положения камеры OpenGL.
/vis/viewer/set/viewpointThetaPhi 270 0 deg
#/vis/viewer/set/viewpointVector -1 -1 0
#Что следует отображать на визуализации.
/vis/scene/add/trajectories
/vis/scene/add/hits
```

Рисование объемов как закрашенных фигур (surface). По умолчанию они отображаются контурными линиями, то есть wireframe.

```
/vis/viewer/set/style surface
```

Чтобы программа визуализации сохраняла все треки, необходима строчка.

```
/tracking/storeTrajectory 1
```

Если есть необходимость, то в файле vis.mac можно выставить порог (Cut), который будет использоваться при моделировании, если во время моделирования потребуется заменить значение, взятое по умолчанию.

```
run/setCut 0.001 mm
```

Запуск моделирования. Запускается 100 первичных частиц.

```
/run/beamOn 100
```

Задание

Алгоритм действий:

1. Настройка переменных OS shell для того, чтобы можно было работать с geant4 (редактирование .cshrc на lx, перезапуск сессии lx).

2. Создание структуры рабочих папок в своей директории: директория Prak, содержащая папки src (для .cc файлов) и include (для .hh файлов).

3. Создание в папках файлов программы, код в [1].

4. Компиляция. В данной версии программы для компиляции используется сборщик cmake.

5. Проверка работоспособности программы и работы визуализации. Нужно запустить файл vis.mac и получить отображения объёма и треков, первичных и вторичных.

6. Доработка файла DetectorConstruction.cc до новой геометрии (варианты приведены на сайте).

7. Вывод данных о частицах в файл и считывание данных из файла при помощи ROOT-[1].

8. Построение гистограмм, изображающих энерговыделение частиц в сцинтилляторах.

9. Полученные 3 гистограммы следует экспортировать в файл графического формата и послать в качестве ответа, используя веб-форму этого задания[1].

Дополнительные материалы по Geant4.

Общие замечания.

Как говорилось выше, если объём слишком тонкий, а пороги моделирования большие, частица может пройти объём, не будучи зарегистрированной в нём (шаг моделирования не сделает остановку внутри него). При этом энергия потерянная частицей в этом объёме будет учтена, но никакого взаимодействия, приводящего к рождению частиц, смоделировано не будет, что может быть критично в случае объёма – мишени.

Рассмотрим две возможные реализации одной задачи. Изучается облучение крови пучком электронов определённой энергии, необходимо определить, какое количество лейкоцитов погибнет. В первом случае, можно создать смесь, повторяющую по составу кровь. Однородно заполнить этой смесью объём и после этого облучить его электронами. После этого можно посмотреть картину энерговыделения в веществе, посчитать объём области, в которой выделившийся энергии хватит на то, чтобы убить лейкоциты. Зная плотность лейкоцитов в крови, можно подсчитать примерное количество погибших лейкоцитов.

Второй случай. Создаётся модель лейкоцита. В кубе из крови равномерно распределяется несколько миллионов

лейкоцитов, каждый из которых является чувствительным объёмом. После этого измеряется энерговыделение в каждом из них, после чего определяется количество отмерших лейкоцитов. Второй способ требует в разы больше ресурсов и машинного времени, так как шаг моделирования ничтожно мал, а количество используемых объектов чрезмерно большое. При этом очевидно, что погибшие лейкоциты по большей своей части будут располагаться в областях с большим энерговыделением, то есть мы опять пришли к рассмотрению картины энерговыделения в веществе, только гораздо более ресурсоёмким способом. Поэтому, более реалистичная модель не всегда является оптимальной.

Работа с геометрией и материалами.

В Geant4 существует несколько типов материалов: изотопы, элементы, молекулы, сплавы и смеси. Для них можно указать такие характеристики как: температура, давление, состояние, плотность.

Примеры создания материалов:

Материал из одного элемента.

```
G4double density = 1.390*g/cm3; //плотность
материала
G4double a = 39.95*g/mole; //молярная масса
//материал задаётся как (“название”, заряд,
молярная масса, плотность)
G4Material* lAr =  new
G4Material("liquidArgon",z=18.,a,density);
```

Молекула.

```
//определяем 2 элемента
a = 1.01*g/mole;
G4Element* elH  =  new
G4Element("Hydrogen",symbol="H",z=1.,a);
```

```
a = 16.00*g/mole;
G4Element* elO  =  new
G4Element("Oxygen",symbol="O",z=8.,a);
density = 1.000*g/cm3;
//создаём материал как (“название”,  плотность,
число компонент)
G4Material* H2O = new
G4Material("Water",density,ncomp=2);
//добавляем в молекулу 2 атома первого элемента и 1
атом второго
H2O->AddElement(elH, natoms=2);
H2O->AddElement(elO, natoms=1);
```

Смесь/сплав.

```
//Смесь может состоять как из чистых элементов, так
и из созданных заранее материалов
G4Element* elC  = …;   // define “carbon” element
G4Material* SiO2 = …;  // define “quartz” material
G4Material* H2O = …;   // define “water” material
density = 0.200*g/cm3;
//финальный материал задаётся через процентное
содержание компонент
G4Material* Aerog =  new
G4Material("Aerogel",density,ncomponents=3);
Aerog->AddMaterial(SiO2,fractionmass=62.5*perCent);
Aerog->AddMaterial(H2O ,fractionmass=37.4*perCent);
Aerog->AddElement (elC ,fractionmass= 0.1*perCent);
```

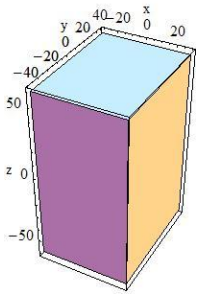
Особый случай газа.

```
//Иногда бывает необходимо определить температуру и
давление газа, эти характеристики влияют на dE/dx
G4double density = 27.*mg/cm3;
G4double temperature = 325.*kelvin;
G4double pressure = 50.*atmosphere;
G4Material* CO2 =
new G4Material(“CarbonicGas", density,
ncomponents=2
kStateGas, temperature, pressure);
CO2->AddElement(C,natoms = 1);
CO2->AddElement(O,natoms = 2);
```

Стоит заметить, что абсолютного вакуума в Geant4 не

существует, вместо него используют газ чрезвычайно малой плотности.

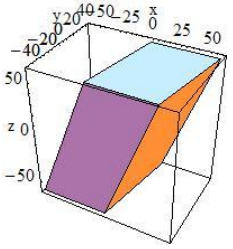
Примеры основных геометрических форм:



Прямоугольный параллелепипед

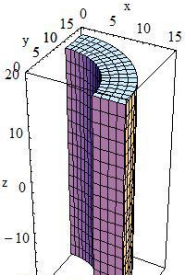
```
G4Box(const G4String& pName,
G4double  pX,
G4double  pY,
G4double  pZ)
// pX, pY, Pz – половины длины/ширины/высоты
```

Параллелепипед



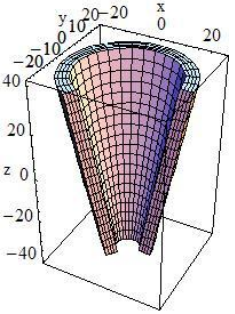
```
G4Para(const G4String& pName,
G4double  dx,
G4double  dy,
G4double  dz,
G4double  alpha,
G4double  theta,
G4double  phi)
//Создание схоже с предыдущим
случаем с добавлением трёх углов наклона
```

Трубка



```
G4Tubs(const G4String& pName,
G4double  pRMin,
G4double  pRMax,
G4double  pDz,
G4double  pSPhi,
G4double  pDPhi)
//pRMin, pRMax – внутренний и внешний
радиусы трубки
//pDz – половина длины трубки
//pSPhi, PDPhi – угол начала и конца
сегмента в радианах
//цилиндр при pRMin = 0
```

Конус



```
G4Cons(const G4String& pName,
```

```
G4double  pRmin1,
G4double  pRmax1,
G4double  pRmin2,
G4double  pRmax2,
G4double  pDz,
G4double  pSPhi,
G4double  pDPhi)
//pRmin1,pRmax1,pRmin2,pRmax2 – внутренний и
внешний радиус на концах конуса
//pDz – половина длины конуса
//pSPhi, pDPhi - угол начала и конца сегмента в
радианах
//конус при pRmin1=pRmin2=pRmax2 =0
```

Более сложные формы описаны на сайте Geant4. Однако, в случае если всех стандартных форм оказывается недостаточно, в Geant4 предусмотрены логические операции с формами. G4UnionSolid – объединение форм, G4SubstractionSolid – удаление части формы, G4IntersectionSolid – остаётся только та часть, в которой две формы пересекаются.

Пример:

```
//создаются прямоугольный параллелепипед  и цилиндр
  G4Box box("Box", 20, 30, 40);
  G4Tubs cylinder("Cylinder", 0, 50, 50, 0, 2*M_PI);
```

Создание из двух стандартных форм новой, стоит заметить, что в случае с вырезанием одной формы из другой необходимо учитывать последовательность форм, из первой вычитается вторая:

```
G4UnionSolid union("Box+Cylinder", &box, &cylinder);
G4IntersectionSolid intersect("Box Intersect
Cylinder", &box, &cylinder);
G4SubtractionSolid subtract("Box-Cylinder", &box,
&cylinder);
//есть опциональная возможность трансформировать
вторую форму
G4ThreeVector  translation(0, 0, 50); //вектор
сдвига
```

```
G4RotationMatrix invRot; //матрица поворота
invRot.rotateY(M_PI/4.*rad); //задаём поворот
вокруг оси Y
G4Transform3D transform(InvRot,translation);
```

создаём объект класса трансформация, который состоит из матрицы поворота и вектора сдвига, объединяем первую форму с трансформированной второй:

```
G4UnionSolid  Union( "Box1UnionCyl1Moved2",&box1,
&Cylinder1, transform);
```

Средства коммуникаций

Введение

Научному сотруднику для эффективной работы постоянно приходится иметь дело со средствами коммуникаций. В настоящее время все телекоммуникационные технологии обеспечиваются посредством сети Интернет. Перечислим ряд общеупотребительных интернет-сервисов, представляющих наибольший интерес для использования в нашей области.

Сервис удаленного терминала

Сервисы для работы в режиме удаленного терминала предоставляют пользователю возможность подключиться к удаленному компьютеру и работать на нем, запускать процедуры операционной системы и установленные программы. При этом локальный компьютер применяется лишь как консоль, предоставляя пользователю монитор, клавиатуру, манипуляторы и т.п. Первым популярным сервисом удаленного терминала стал сервис telnet, построенный на базе одноименного протокола (набора соглашений и правил обмена информацией). В настоящее время использовать данный сервис настоятельно не рекомендуется, вследствие его принципиальной уязвимости перед злоумышленниками, имеющими возможность прочитать информацию во время ее передачи по узлам сети Интернет. Даже единичное использование данного сервиса может привести к тому, что пароль и логин пользователя будут прочитаны постоянно работающими сканирующими программами, после чего можно взломать всю систему безопасности данного пользователя.

Вместо telnet рекомендуется пользоваться сервисом

протокола Secure Shell (ssh), когда все данные обмена, включая пароль пользователя, передаются в зашифрованном виде. Во время обмена используется принцип двойных ключей, когда при установке соединения компьютеры сначала обмениваются файлами, с помощью которых пакет информации можно зашифровать, но нельзя расшифровать (т.н. открытые ключи). Зашифрованный открытым ключом пакет информации пересылается партнеру по обмену, который при его получении расшифровывает его при помощи второго (закрытого) ключа, никогда не передаваемого в Интернет. Бесплатно предоставляемый разработчиком[4] алгоритм шифровки исключает возможность несанкционированной дешифровки информации на современных мощностях за разумное время для кого бы то ни было (возможно, за исключением разработчика). Существует также разработка программы с открытым кодом OpenSSH.

На компьютерах с Unix-подобными ОС для использования Secure Shell сервиса применяется консольная утилита ssh, синтаксис которой обсуждался в первой части данного курса[7]. На машинах под управлением ОС Windows на данный момент популярно использование связки программ Putty и Xming, описание и практика работы с которыми приводится ниже.

Сервис обмена файлами

Данные сервисы специализируются на приеме и передаче информации, организованной в виде файлов. Первым популярным сервисом обмена файлами между компьютерами стал сервис ftp (File Transfer Protocol). Передача информации при помощи сервиса ftp организована схожим с telnet образом. Несмотря на это, данный сервис продолжает использоваться, делать это можно в тех случаях,

когда передается заведомо открытая информация, например, файлы для размещения на WWW-сайте. Для работы с сервисом ftp существуют много программ, в частности, скачивать выложенные в открыты доступ файлы можно при помощи веб-браузеров, при этом в адресной строке браузера будет высвечиваться не адрес WWW-сервиса (http://example.ru), а описываемого сервиса (ftp://example.ru). При работе на компьютере под управлением Unix-подобной ОС можно запустить консольную утилиту, например, находясь в сети ОИЯИ:

```
ftp faxe.jinr.ru
```

Если же требуется соблюдать требования безопасности, то рекомендуется использовать сервис передачи данных scp. Данный сервис организован по тем же правилам, что и ssh, синтаксис использования консольной утилиты scp на компьютерах с Unix-подобными ОС можно проиллюстрировать следующим примером:

```
scp user1@example1.ru:~/pic/01.jpg fig2.jpg
```

В этом примере с удаленной машины example1.ru берется файл 01.jpg, который лежит в папке /home/user1/pic и передается в текущую папку на локальной машине, но уже под именем fig2.jpg. При вводе этой команды локальная машина свяжется с удаленной, проведет процесс авторизации, при необходимости запросит у пользователя пароль для логина user1 на удаленной машине, произведет процесс обмена открытыми ключами и начнет процесс передачи. Если же переименование не требуется, то в качестве второго аргумента команды можно просто указать символ точки:

```
scp user1@example1.ru:~/pictures/scheme01.jpg  .
```

Аналогично организуется и передача фала на удаленную машину, или с одной удаленной машины на другую, в том числе под разными логинами.

Сервисы коммуникаций «человек-человек»

В отличие от перечисленных выше сервисов, которые обеспечивают взаимодействие человека с удаленным компьютером («человек-компьютер»), здесь речь идет передаче информации от человека к другому человеку (аудитории) при использовании компьютерных сетей.

Электронная почта, e-mail. В данный момент обмен электронными письмами не требует какого-либо специального описания, в большинстве случаев сейчас для использования e-mail сервиса применяются WWW-браузеры. Описание специальных программ данного сервиса (почтовых клиентов) можно найти на кафедральном сайте[1]. Следует все же отметить, что этот сервис способен работать независимо от WWW-сервиса, для его обеспечения на серверах функционирует специальное серверное ПО (POP3, SMTP).

Сервисы передачи коротких текстовых сообщений, ICQ, Jabber, talk и т.п. Упомянутые сервисы пережили свой короткий расцвет, когда всемирная сеть Интернет уже появилась, но обладала низкой пропускной способностью. На данный момент принцип передачи коротких текстовых сообщений в режиме реального времени используется в виде дополнения в программах других сервисов (skype chat, e-mail chat и т.п.).

Сервисы передачи звука и изображения. Такие сервисы, как skype, программы видеоконференций типа VRVS и т.п. широко используются для проведения рабочих совещаний в коллаборациях, которые обычно разбросаны по разным городам и странам. Компания Google также пытается завоевать себе место в этом бизнесе с сервисом Google Video. В данных сервисах изначально аналоговая информация, звук (голос) и видео, сначала кодируется в

цифровой формат, передается, а потом для получателя снова декодируется. На сегодняшний момент даже традиционные телефонные компании перешли на такой алгоритм. А использование для передачи пакетов оцифрованной информации возможностей сети Интернет (т.н. технология VoiceIP, VoIP) делает сервис беспрецедентно дешевым (можно обойтись только оплатой интернет-подключения), гибким и полезным. В частности, очень удобна возможность передавать собеседнику в режиме реального времени отображение со своего монитора («рабочего стола»).

WWW-сервис. На данный момент исключительно востребованным оказался использованный в данном сервисе принцип коммуникации, когда информация выкладывается в интернет-доступ в соответствующие места (WWW-сайты), а затем пользователи сами выбирают то, что им нужно, а также дополняют выложенную информацию. Начинающему научному сотруднику зачастую ставится задача разработки информационного обеспечения мероприятий, проектов или рабочих групп посредством WWW-сервиса, поэтому в данной работе будут затрагиваться некоторые практические моменты создания и использования содержимого WWW-сайтов.

Практика: доступ к серверам с Unix-подобными ОС через ПК с ОС Windows.

Если и локальная для пользователя, и удаленная машина работают под ОС Windows, то для работы в режиме удаленного терминала можно воспользоваться разработанным Microsoft сервисом HyperTerminal. Для связи двух машин под управлением Unix-подобных ОС, как правило, достаточно запуска утилиты ssh. Часто, однако, встречается ситуация, когда сотруднику в качестве персонального компьютера удобнее пользоваться машиной с

Windows и при этом использовать ее как консоль для удаленной работы на машинах коллективного пользования с Unix-подобными ОС.

Для работы в компьютерном классе филиала НИИЯФ МГУ сначала нужно загрузить рабочий ПК с ОС Windows, для этого при включении ПК нужно в период соответствующего приглашения нажать клавишу F8, а затем выбрать вариант загрузки «Local OS / Windows».

Xming, установка и настройка X-сервера для ОС Windows

Для запуска удаленных (remote) графических приложений (в том числе пакета ROOT в интерактивном режиме) нужно, чтобы на локальной машине, на которой пользователь непосредственно работает, была запущена специальная программа обработки приложений стандарта X11 под названием X-сервер. Программа Xming — это свободно распространяемый X-сервер для Windows, ее можно загрузить на странице разработчиков[5]. На ПК в компьютерном классе можно загрузить дистрибутив программы с кафедрального сайта, установить ее, например, на «Рабочий стол», папка «D:/Local Settings/…/Desktop». Для полноценной установки программ требуются права администратора, однако данная программа способна работать и без записей в реестр программ. Соответственно, после появления папки Xming нужно в нее зайти, запустить исполняемый файл Xlaunch.exe, а во время начала работы выбрать опции «Multiple windows», «no client». Если программа правильно запущена, то ее значок будет отображен в трее панели задач Windows, и можно переходить к следующему пункту подготовки.

Putty, ssh клиент для ОС Windows

Putty — это бесплатно распространяемая программа-клиент протокола ssh для Windows, загрузить ее можно на странице разработчиков[6]. Для начала работы с ней на машинах компьютерного класса следует ее найти в списке установленных программ.

На кафедральном сайте[1] описываются и иллюстрируются полезные настройки данной программы. Для выполнения работы наиболее важными их них является настройка кодировки шрифтов (выбор кодировки KOI8-R позволяет правильно отображать кириллические символы) и разрешение для программы Putty транслировать изображение графических приложений на локальную машину, для это нужно поднять флаг «Connection/SSH/Tunnels/Enable X11 forwarding».

Для выполнения дальнейшей работы следует проделать все рекомендуемые настройки для доступа на сервер lx.msu.dubna.ru. После проведения настроек полезно перейти в меню «Session», ввести в поле «Saved sessions» новое имя, например «username@lx X11» и нажать кнопку «Save». После этого настройки данной сессии можно будет загружать для использования при следующем старте программы. Для старта сессии обмена с сервером lx нужно нажать кнопку «Open» или дважды щелкнуть курсором по имени сохраненной сессии. После этого в открывшемся окне XTerminal можно работать точно также. Как это делалось в задачах первой части данного курса. В частности, можно запустить пакет анализа данных ROOT:

```
lx> root
```

Рекомендуется запустить созданные ранее скрипты ROOT для проверки открытия окон TBrowser, TCanvas, TTreeViewer.

Точно так же можно запускать и другие графические приложения, однако, в серверном кластере НИИЯФ МГУ это лучше делать на сервере sullen. Часто требуется запускать, например, программу emacs для редактирования скриптов ROOT, а также веб-браузер: `sullen> firefox &`

Практика: размещение информации в WWW.

Устройство WWW-сервиса

Работа WWW-сервиса организуется следующим образом. Существуют круглосуточно работающие компьютеры с климат-контролем и защищенным электропитанием. На этих машинах запущены специальные программы, т.н. «веб-серверы» (например, Apache или nginx), которые по запросам клиентов отсылают им файлы с данных серверов. На клиентских компьютерах для работы с WWW-сервисом также запускаются специальные программы, т.н. «веб-браузеры» (Opera, Google Chrome и т.д.), которые представляют полученную информацию пользователю, а также посылают на сервер некоторую информацию со стороны клиента (например, запросы). Для организации обмена существует набор соглашений HTTP (Hypertext Transfer Protocol), базовым языком разметки информации на веб-страницах является HTML (Hypertext Markup Language).

WWW-сайты можно разделить на две категории: статические и динамические сайты. Статические сайты содержат файлы, написанные на языке HTML, файлы для отображения мультимедиа-информации (картинки, звук, видео), а также файлы, не отображающиеся средствами веб-браузера, но предназначающиеся для скачивания. Содержимое таких сайтов меняется только в том случае, если администратор сайта его вручную поменяет, во всех

остальных случаях запрашиваемое клиентом содержимое сайта скачивается ему в неизменном (статическом) виде. Практически вся интерактивность такого сайта заключается в возможности перехода пользователя по встроенным в HTML файлы гиперссылкам.

Динамические сайты в дополнение к уже перечисленным ресурсам содержат еще и специальные программы (веб-скрипты или веб-сценарии), которые позволяют автоматизировать обновление содержимого страницы, обеспечивают гораздо более широкие возможности взаимодействия сайта и пользователя. За динамичность сайтов приходится платить возросшей сложностью и ресурсоемкостью сайта, меньшей наглядностью его создания.

Создание WWW-страниц, редактор NVu

Коды для сайта. Для того, чтобы создать веб-сайт, есть несколько путей. Можно воспользоваться готовым on-line сервисом типа livejournal.com или vk.com, это простое и быстрое решение. Однако, такие страницы обладают весьма ограниченным интерфейсом и, что еще более важно, полностью зависят от владельца данного сервиса. Поэтому обратимся к случаю, когда сервер для размещения сайта (т.н. «хост») предоставляет нам коммерческая или корпоративная служба, а нам предстоит самостоятельно разместить на нем коды создаваемого сайта.

Если поставлена задача создания на сайте достаточно сложных элементов, например, системы регистрации, форума или базы данных сообщества, то разумно будет не писать с чистого листа HTML файлы и веб-сценарии, а скачать и установить готовый набор таких ресурсов (т.н. «движок»). Существует много таких движков, как платных, так и бесплатных, нужно лишь выбрать тот, который

наиболее полно отвечает сформулированным требованиям. Так, кафедральный сайт функционирует на движке Moodle. Как правило, после того, как движок сайта установлен, сайт можно заполнить содержимым, не разбираясь в программных кодах, а просто пользуясь интерфейсами, доступными через веб-браузер.

На сегодняшний момент, даже создавая самый простой сайт (т.н. «страницу присутствия»), разумно пользоваться готовым движком типа WordPress. Однако все эти движки требуют поддержки, помимо HTML, дополнительных языков программирования (например, JavaScript, PHP, Pearl и т.д.) и системы управления базами данных (MySQL или PostgreSQL). Поэтому иногда бывает проще самому написать простой сайт на чистом HTML языке, чем добиваться выполнения перечисленных требований на сайте размещения. Кроме того, некоторое знание HTML и других языков полезно для приспособления движка под свои нужды.

Веб-редактор. В принципе, для создания и модификации HTML кода достаточно любого текстового редактора (Notepad или emacs), однако, гораздо рациональнее применять для этого специализированную программу. Очень популярным веб-редактором является Adobe Dreamweaver, который не только радикально уменьшает количество рутинной работы, но и, что не менее важно, заставляет создателя сайта выдерживать хороший стиль сайта. Однако, Dreamweaver это профессиональный программный продукт, который слишком дорого обходится для эпизодической работы. Поэтому для данной практики имеет смысл обойтись бесплатным Open Source визуальным веб-редактором NVu, хоть он гораздо менее функционален. Основой для создания NVu послужила часть кода Mozilla (Mozilla Composer) однако, в нем есть ряд дополнительных

возможностей. Это Менеджер сайтов, позволяющий редактировать страницы на удаленном сервере, встроенный Редактор CSS с предварительным просмотром стилей, подсветка синтаксиса кода, проверка орфографии и т.д.

Для запуска NVu достаточно скачать с кафедрального сайта[1] или со страницы разработчика его дистрибутив, а затем распаковать его в любое место на локальном компьютере. Далее нужно зайти в распакованную папку Nvu и запустить исполняемый файл nvu.exe.

В появившемся интерфейсе программы (Рис. 3) можно наблюдать командную строку сверху, затем панель инструментов (данные вещи схожи в любых популярных редакторах), ниже слева панель упомянутого Менеджера сайта, а самая большая область отображает содержимое HTML файла.

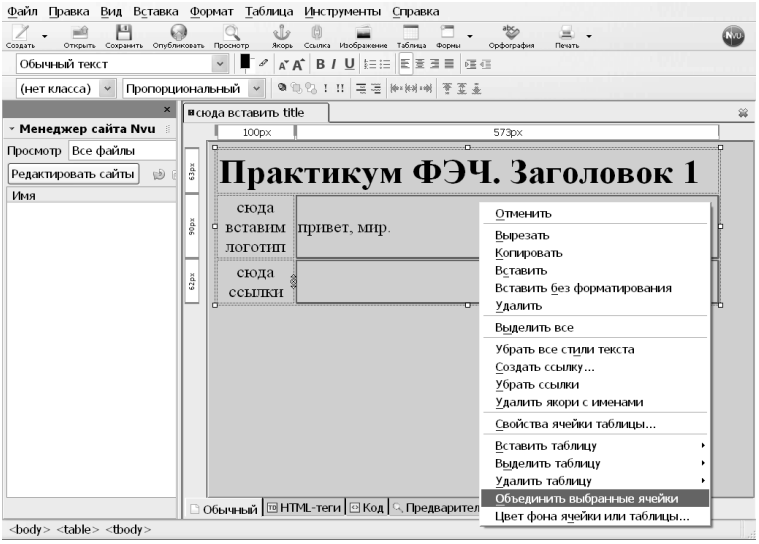


Рис. 3: Интерфейс Nvu, объединение ячеек таблицы.

Нажав на закладку «Код» снизу этой области, можно

увидеть HTML стандартного бланка WWW-страницы. Но для начала у той области выбран стиль представления «Обычный», когда набираемый текст и другие вводимые на страницу элементы отображаются так, как они должны выглядеть в окне браузера. Рекомендуется сразу же внести изменения в формат страницы (в командной строке опция «Формат/Заголовок и свойства страницы»), а именно: поменять кодировку с «Windows-1251» на более универсальную «Юникод(UTF-8)», а также ввести какой-нибудь заголовок (title) страницы, например, свою фамилию.

Макет страницы. Теперь можно приступать к редактированию отображаемого содержимого страницы, которое помещается внутри парного тега <body>. Лучше всего не собирать все содержимое в одну трудно контролируемую кучу, а заранее подготовить верстку (макет) страницы, т.е. визуально разделить ее на несколько частей, в каждую из которых будет помещаться своя информация. Причем, с точки зрения хорошего стиля данный макет доложен быть «эластичным», т.е. стремиться заполнить собой всю ширину открытого окна браузера, и при этом не вылезать за эту ширину, чтобы пользователю не приходилось применять горизонтальное перематывание. На это нужно обращать внимание, поскольку сайты делаются в первую очередь для пользователей, а экраны и открытые окна у них могут быть очень разные. Для создания эластичного макета существуют разные методы, в рамках настоящей задачи предлагается создать т.н. «макетную таблицу». Для этого нужно в командной строке выбрать опцию «Вставка/Таблица/Точная», указать желаемое количество строк и столбцов, выбрать ширину таблицы во все окно браузера (например, 95%), а границы ячеек можно сделать невидимыми («рамка 0 px»). Далее избыточное количество ячеек можно объединять, как это показано на

Рис. 3, настраивать конкретный вид ячеек, а все отображаемое содержимое страницы вставлять уже внутрь получившейся макетной таблицы. Рекомендуется, практикуясь создавать таблицу, изучать соответствующие изменения тегов таблицы (<table>, <tr>, <td>) и их дополнительных настроек (атрибутов).

Понятие логической разметки на примере работы с текстом. В таблицу можно вставить текстовую информацию, и тут же сделать ее т.н. логическую разметку, последовательно выделяя нужные куски текста и выбирая в поле формата содержимого, например, «Заголовок 1 уровня», «Заголовок 2 уровня», «Абзац» и т.п. Программа внесет соответствующие теги в код, а далее каждый браузер выделит размеченные заголовки, списки, цитаты при помощи увеличенного размера шрифта, отступами и т.п. В принципе, синтаксис HTML позволяет самому вручную изменять размер шрифта, его цвет и делать прочую оформительскую разметку, однако, это плохой стиль оформления веб-страницы. Гораздо надежнее в данном конкретном месте файла сделать лишь логическую разметку, а потом стили отображения всех заголовков, гиперссылок и прочих элементов на сайте настроить, например, при помощи CSS технологий. Описание CSS можно найти на кафедральном сайте[1], настройку CSS в программе NVu можно сделать, выбрав опцию «Инструменты/Редактор CSS» в верхней командной строке. В данном случае под хорошим стилем понимается стабильное единообразное отображение всего содержимого сайта в разных браузерах.

Гиперссылки. При клике курсором на объект «гиперссылка» клиент переходит на другой сайт, на другой файл текущего сайта, либо в то место текущего HTML документа, куда разработчиком была вставлен т.н. «якорь» (named anchor). Для того, чтобы сделать гиперссылкой

какой-нибудь элемент страницы, например, фразу или картинку, нужно его выделить курсором и выбрать опцию «Вставка/Ссылка» в командной строке. Можно настроить ссылку так, чтобы она открывалась в новом окне браузера и т.д.

Изображения. В качестве изображения можно вставлять файлы формата JPG, GIF, PNG и т.д., выбрав опцию «Вставка/Изображение» в верхней строке. Рекомендуется ознакомиться с размещенной в [1] краткой характеристикой областей применения различных графических форматов, а также графических редакторов для подготовки изображений для веб-страниц, в частности, Open Source редактор GIMP. Следует подчеркнуть, что для хорошего стиля таблицы нужно при вставке каждого изображения присваивать ей т.н. «замещающий текст» (в коде это атрибут alt тега картинки <img>). Тогда созданный сайт будет хорошо индексироваться поисковыми роботами (Google, Yandex), а также в случае невозможности отображения картинки клиент хотя бы получит представление о том, что должно было быть отображено. Также у картинки можно настраивать ее положение относительно текста (атрибут align), отступ по вертикали и горизонтали (атрибуты vspace, hspace), отображаемый размер и т.д.

Мультимедиа информация. Видеофайлы (формата MP4, AVI и т.д.) и звуковые файлы (формата MP3, WAV и т.д.) модно вставить при помощи технологии т.н. «встраиваемых модулей» (Plug-In). Такие файлы веб-браузеры не способны самостоятельно обрабатывать, но они вызывают сторонние установленные на компьютере клиента программы (например, Windows Media Player). Ниже представлен код Plug-in модуля, который в рамке размером 500 на 32 точки представляет MP3 файл, взятый из папки «Music», размещенной в том же месте, где и основной

HTML файл:

```
<embed src="Music/18_Almaty.mp3" width="500"
height="32" align="middle"></embed>
```

Проблема здесь заключается в том, что неизвестно, установлена ли подходящая Plug-in программа на компьютере у клиента, сможет ли он воспринять эту мультимедиа информацию. В рамках современного решения данной проблемы большинство WWW-сервисов размещают и звук, и видео в наиболее универсальном Macromedia Flash формате (файлы SWF или FLV), а клиентов стимулируют постоянно обновлять соответствующую программу обработки Flash Player.

Веб-сценарии. Как уже упоминалось ранее, для того, чтобы выполнялись более или менее развитые программы динамических страниц на сервере (т.н. серверные веб-сценарии) требуется настройка на сервере модулей PHP или ASP, работа с СУБД типа MySQL, что не всегда возможно. Тем не менее, даже в простую веб-страницу можно вставить сценарии, которые прямо на компьютерах клиентов обрабатываются большинством современных веб-браузеров (т.н. клиентские веб-сценарии). Так, приведенный ниже сценарий при помещении его в HTML код сначала откроет еще одно окно браузера с фразой «Введите имя», текстовым полем, куда клиент может что-то напечатать, и кнопкой «ОК» для ввода напечатанного. Далее, применяется еще один метод класса window, который открывает дополнительное окно, в котором написано «Привет» и повторяется 3 раза напечатанный клиентом текст.

```
<script type="text/javascript">
      var a1=window.prompt("Введите имя");
      window.alert("Привет, "+a1+a1+a1);
</script>
```

Размещение кодов на сервере. После того, как коды

веб-сайта подготовлены на локальном компьютере, их можно переместить на сервер в указанное администратором сервера место. В случае размещения сайта в домене на sullen.msu.dubna.ru зарегистрированным пользователем user1 ему нужно создать в своей домашней директории папку со стандартным именем public_html. Затем нужно закачать туда все файлы сайта (например, при помощи Total Commander), а затем изменить права доступа к содержимому этой папки (добавив всем права чтения):

```
chmod -R 755 public_html
```

После этого сайт станет доступен по адресу http://users.msu.dubna.ru/~user1/ , при этом первым в окно браузера будет загружаться файл со стандартным именем index.html (при условии, что он в этой папке существует).

Задание

1. Используя статический HTML, подготовить свою домашнюю страницу. Как минимум, Ваша страница должна содержать:
   1.1. Ваше имя и фамилию;
   1.2. название Вашего учебного заведения, факультет, кафедру и курс (либо место работы и должность);
   1.3. координаты: e-mail, при наличии - номер офиса и телефон;
   1.4. Вашу фотографию.
2. Дополнительно Вы так же можете указать:
   2.1. область исследований, в которой вы работаете, или же просто интересуетесь;
   2.2. ссылки на Ваши работы/дипломы/доклады;
   2.3. ссылки на страницы Ваших коллег/друзей.
3. Страницу можно разместить на веб-сервере МГУ в Дубне (http://www.msu.dubna.ru), сервере Вашей лаборатории (например, http://lpp.jinr.ru) или института

(http://www.jinr.ru).

4. Перед размещением фотографию обработать при помощи графического редактора, или можно использовать онлайн-сервис типа Instagram или Phixr. Можно также разместить мультимедиа-ресурсы и веб-сценарии.
5. В качестве ответа нужно прислать HTTP ссылку на страницу, используя веб-форму задания на сайте[1].

Ссылки

1. *http://hep.msu.dubna.ru (4 курс, Практикум/ Раздел «Информационные методы»)*. Область на официальном сайте кафедры физики элементарных частиц (г. Дубна), где расположены необходимые ресурсы для данного практикума.
2. *http://geant4.web.cern.ch* – официальный сайт разработчиков *GEANT*.
3. *http://root.cern.ch* – официальный сайт разработчиков *ROOT*.
4. http://www.ssh.com/ – – официальный сайт разработчиков *Secure Shell*.
5. *http://www.straightrunning.com/XmingNotes/* – официальный сайт разработчиков *Xming*.
6. *http://www.chiark.greenend.org.uk/~sgtatham/putty/* – официальный сайт разработчиков *Putty.*
7. *В.В. Леонтьев, И.И. Белотелов, Задачи раздела «Информационные методы в физике высоких энергий» // Университетская книга Москва*, 2011.

Учебное издание

Владимир Викторович Леонтьев
Иван Андреевич Орлов

ЗАДАЧИ РАЗДЕЛА «ИНФОРМАЦИОННЫЕ МЕТОДЫ В ФИЗИКЕ ВЫСОКИХ ЭНЕРГИЙ», ЧАСТЬ 2

Работа поступила в ОНТИ 2013

Тираж 50 экз.