Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей
Кафедра Информатики
Дисциплина: Программирование

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

Визуализация физических экспериментов

(Физическое моделирование)

Выполнил:
студент группы 953503
Новиков А. А.

Научный руководитель:
ассистент кафедры информатики
Удовин И.А.

Минск 2020

Содержание

Введение

Что такое физика?

Отвечая на вопрос «что такое физика» можно сказать, что это наука о неживой природе, то есть физика изучает окружающий нас физический мир. В Древнем мире это было действительно так, так как в те времена знаний о природе было мало и отдельные науки не выделялись. Однако по мере накопления знаний, от физики отделились такие науки как химия, география, геология, астрономия. Не говоря уже о том, что биология, математика выделились как науки еще раньше.

В настоящее время физику относят к естественным наукам, под ней понимают науку, которая изучает общие законы природы, структуру и движение материи. Физика изучает природу как единое целое, при этом предметом ее исследования являются как мельчайшие частицы, так и вся Вселенная.

В наше время в ответ на вопрос «что такое физика» можно включить и все окружающие нас технические приборы. Все они появились в результате достижений в различных разделах физики (механике, электромагнетизме, оптике, термодинамике и др.).

В Древнем мире физические явления просто наблюдали, подмечали происходящие в природе общие закономерности и записывали их. Объясняли все это наличием божественных сил.

Где-то начиная с эпохи Возрождения (примерно с XV века) в физике в качестве метода исследования явлений природы стал использоваться эксперимент, который в наше время является основным методом исследования.

Наблюдаемые в природе закономерность стали называть физическими законами. Часто они описывали количественные соотношения в природе, что требовало их записи в виде математических формул. Таким образом, можно сказать, что математика стала языком физики.

Еще один важный метод познания — это теоретическое описание природных явлений. На основе таких описаний создаются физические теории, включающие общие законы природы, которые объясняют различные явления. Первую физическую теорию создал И. Ньютон. В настоящее время ее называют «классической механикой».

На сегодняшний день в физике используется еще один метод исследования — компьютерное моделирование.

Физические явления делят на механические, тепловые, электромагнитные, световые, квантовые. Их изучением занимаются различные разделы физики. Так что в каком-то смысле, отвечая на вопрос «что такое физика», можно сказать, что это комплексная наука, включающая в свой состав такие науки (разделы физики) как механика, термодинамика, молекулярная физика, электромагнетизм, оптика, атомная физика, ядерная физика.[1]

В данной курсовой рассматривается такой раздел физики, как механика, для реализации физического моделирования.

Меха́ника (греч. μηχανική — искусство построения машин) — раздел физики, наука, изучающая движение материальных тел и взаимодействие между ними; при этом движением в механике называют изменение во времени взаимного положения тел или их частей в пространстве[2].

В курсовой работе ставятся следующие задачи:

1. Реализовать реалистичную физическую модель мяча, отскакивающего от стен и от пола, с возможностью пнуть его.

2. Реализовать симуляцию гравитации в космосе, сохранив реальные пропорции массы, расстояния и скорости., добавить возможность выбора режима симуляции:

1) Земля и Луна
2) Вывод спутника на геостационарную орбиту
3) Вывод спутника на высокую эллиптическую орбиту
4) Демонстрация. Создание нескольких планет с 1 или более спутником из исходного облака космической пыли.

Для справки:

Геостациона́рная орби́та (ГСО) — круговая орбита, расположенная над экватором Земли (0° широты), находясь на которой, искусственный спутник обращается вокруг планеты с угловой скоростью, равной угловой скорости вращения Земли вокруг оси. В горизонтальной системе координат направление на спутник не изменяется ни по азимуту, ни по высоте над горизонтом — спутник «висит» в небе неподвижно.[3]

Высокая эллиптическая орбита (ВЭО) — это тип эллиптической орбиты, у которой высота в апогее во много раз превышает высоту в перигее.[4]

Рисунок 1 – Геостационарная орбита

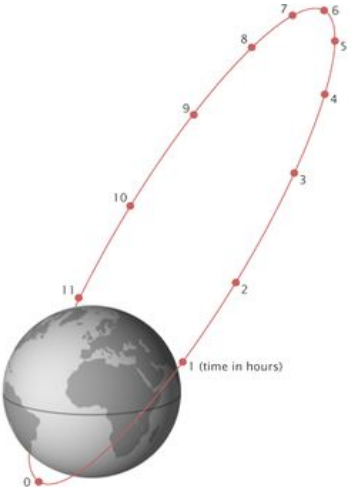

Рисунок 2 – ВЭО

1 Описание среды разработки

1.1 Среда разработки

Интегрированная среда разработки (англ. Integrated development environment – IDE) – комплекс программных средств, используемый программистами для разработки программного обеспечения (ПО).

Среда разработки включает в себя:

1) текстовый редактор,
2) транслятор (компилятор и/или интерпретатор),
3) средства автоматизации сборки,
4) отладчик.

Иногда содержит также средства для интеграции с системами управления версиями и разнообразные инструменты для упрощения конструирования графического интерфейса пользователя.

ИСР предназначенные для нескольких языков программирования: IntelliJ IDEA, NetBeans, Eclipse, Qt Creator, Geany, Embarcadero RAD Studio, Code::Blocks, Xcode, Microsoft Visual Studio. [5]

Выбор остановим на среде разработки **Microsoft Visual Studio**.

1.2 Среда разработки Microsoft Visual Studio

Microsoft Visual Studio – линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения

и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии **Windows Forms**, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии **IntelliSense** и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. **Visual Studio** позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, **Subversion** и **Visual SourceSafe**), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент **Team Explorer** для работы с **Team Foundation Server**).[6]

Для физического моделирования было решено использовать компилируемый, статически типизированный язык программирования общего назначения **C++** и объектно-ориентированный язык программирования **C#**.

1.3 Описание языков программирования C++, C#

C++ поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником — языком C, — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования.[7]

C# – современный объектно-ориентированный и типобезопасный язык программирования. **C#** относится к широко известному семейству языков **C**, и покажется хорошо знакомым любому, кто работал с **C**, **C++**, **Java** или **JavaScript**. [2]

C# является объектно-ориентированным языком, но поддерживает также и компонентно-ориентированное программирование. **C#** подходит для создания и применения программных компонентов.

Вот лишь несколько функций языка **C#**, обеспечивающих надежность и устойчивость приложений:

- Сборка мусора автоматически освобождает память, занятую недостижимыми неиспользуемыми объектами.
- Обработка исключений предоставляет структурированный и расширяемый подход к обнаружению ошибок и их восстановлению.
- Типобезопасная структура языка делает невозможным чтение из неинициализированных переменных, индексацию массивов за пределами их границ или выполнение непроверенных приведений типов.

- В **C#** существует единая система типов. Все типы **C#**, включая типы-примитивы, такие как **int** и **double**, наследуют от одного корневого типа **object**.
- **C#** поддерживает пользовательские ссылочные типы и типы значений, позволяя как динамически выделять память для объектов, так и хранить упрощенные структуры в стеке.
- Чтобы обеспечить совместимость программ и библиотек **C#** при дальнейшем развитии, при разработке **C#** много внимания было уделено управлению версиями.
- **C#** включает функции, поддерживающие приемы функционального программирования, такие как лямбда-выражения.

Кроме того, **C#** обеспечивает полную поддержку объектно-ориентированного программирования, включая инкапсуляцию, наследование и полиморфизм.[8]

1.4 Описание OpenGL и GLUT

Последним важным выбором на этапе планирования стало освоение **OpenGL и GLUT**.

OpenGL (Open Graphics Library) — спецификация, определяющая платформонезависимый (независимый от языка программирования) программный интерфейс для написания приложений, использующих двумерную и трёхмерную компьютерную графику.

Включает более 300 функций для рисования сложных трёхмерных сцен из простых примитивов. Используется при создании компьютерных игр, САПР, виртуальной реальности, визуализации в научных исследованиях. На платформе Windows конкурирует с Direct3D.[9]

OpenGL Utility Toolkit (GLUT) — библиотека утилит для приложений под **OpenGL**, которая в основном отвечает за системный уровень операций ввода-вывода при работе с операционной системой. Из функций можно привести следующие: создание окна, управление окном, мониторинг за вводом с клавиатуры и событий мыши. Она также включает функции для рисования ряда геометрических примитивов: куб, сфера, чайник. **GLUT** даже включает возможность создания несложных всплывающих меню.

GLUT был создан Марком Килгардом (Mark Kilgard), во время его работы в Silicon Graphics Inc.

Использование библиотеки **GLUT** преследует две цели. Во-первых, это создание кроссплатформенного кода. Во-вторых, **GLUT** позволяет облегчить изучение **OpenGL**. Чтобы начать программировать под **OpenGL**, используя **GLUT**, требуется всего страница кода. Написание аналогичных вещей на API требует несколько страниц, написанных со знанием API управления окнами операционной системы.

Все функции **GLUT** начинаются с префикса glut (например, glutPostRedisplay отмечает текущее окно как требующее перерисовки).[10]

2 Разработка моделей

2.1 Результат

Первая физическая модель

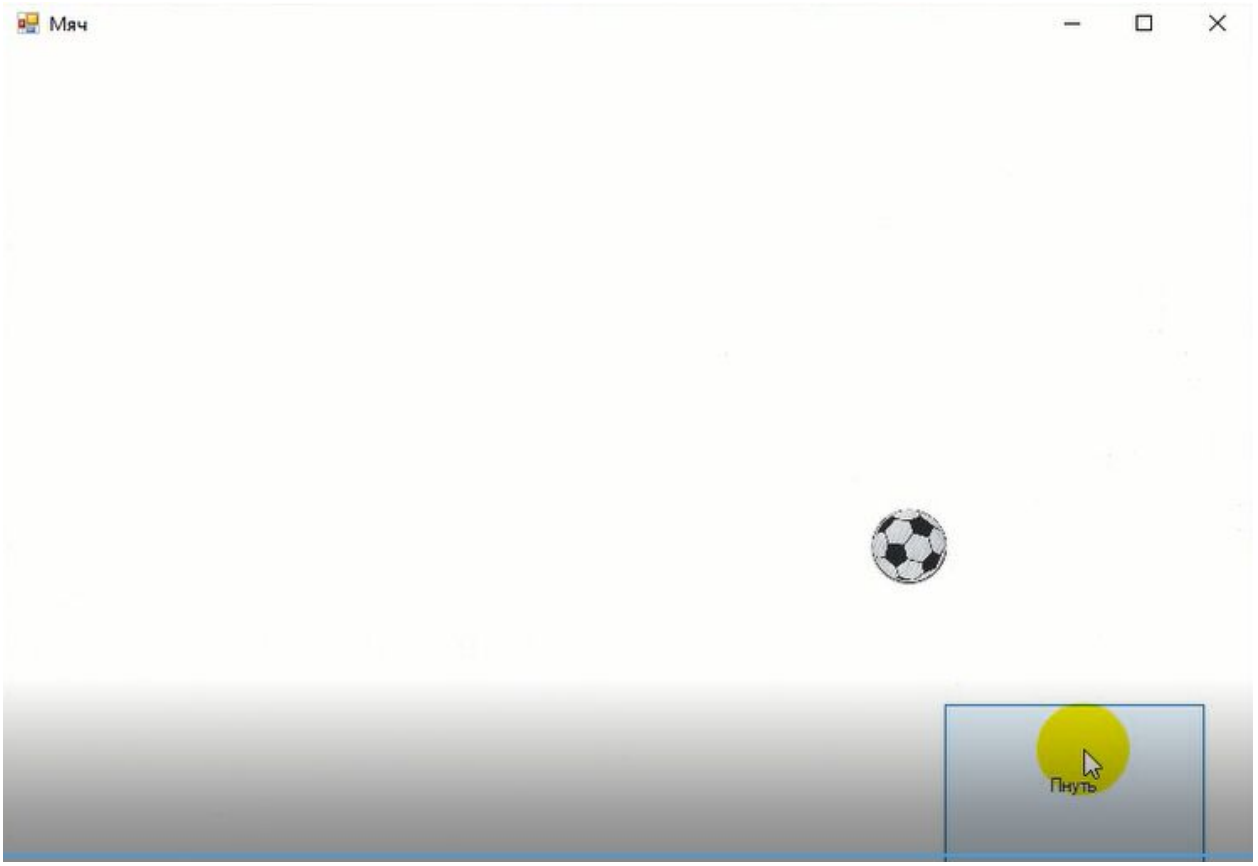

Рисунок 2.1.1 Физическая модель - прыгающий мяч

Мяч отскакивает от пола, ударяется об левую и правую стены, кнопка “Пнуть” направляет мяч к левой стене.

Вторая физическая модель

Рисунок 2.1.2 Физическая модель - Земля и Луна

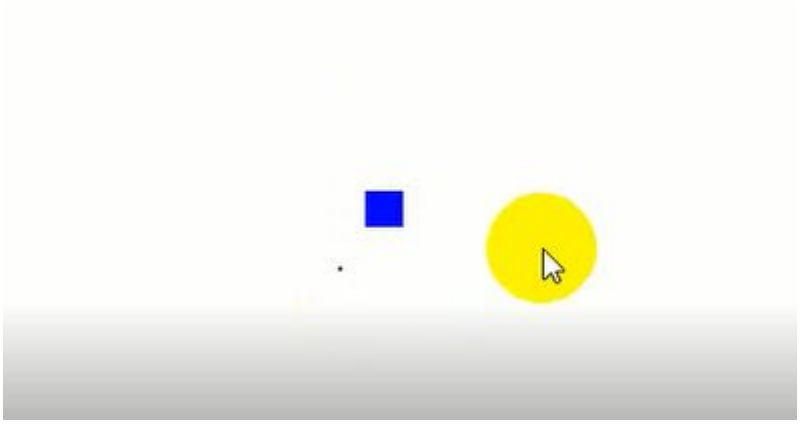

Рисунок 2.1.3 Вывод спутника на геостационарную орбиту

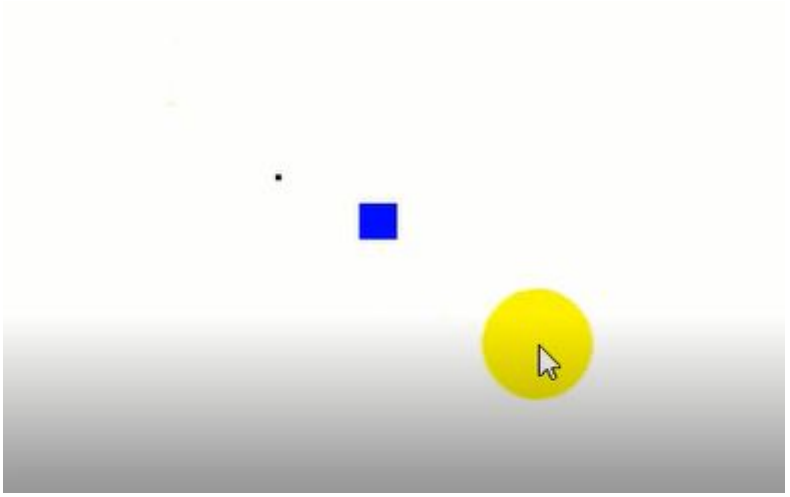

Рисунок 2.1.4 Вывод спутника на высокую эллиптическую орбиту

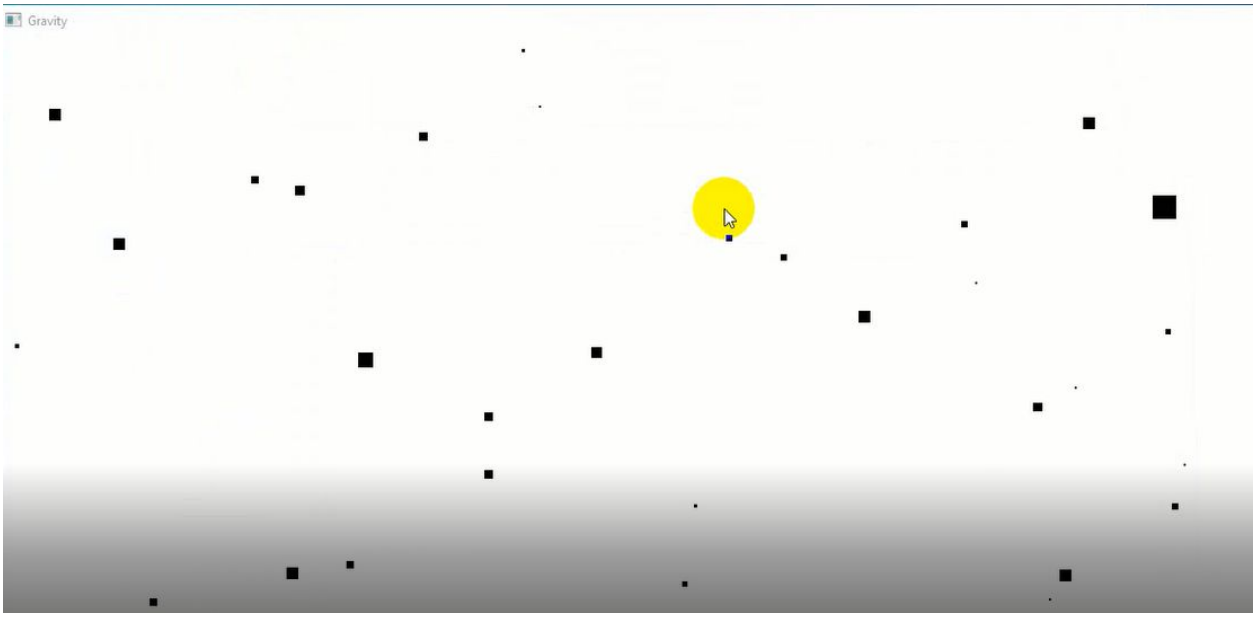

Рисунок 2.1.4 Создание нескольких планет с 1 или более спутником

Описание работы физических моделей

Первая модель

Разрабатывалась в Windows Forms на C#.

Классы:

```
public class Physics
 {
       public PointF position;
       float gravity;
       float a;
       float impulse;

       public float dx;

       public Physics();
       public void ApplyPhysics();
       public void CalculatePhysics();
       public void AddForce(float forceValue);
       //Пинаем с огромной силой
       public void AddForceQuickly(float forceValue);

}
//сущность
 public class Entity
  {
       public Physics myPhysics;
       public Image sprite;
       public Size mySize;
       int angle = 4; /*в этой версии не используется*/
       public Entity(Size size);
      public void DrawSprite(Graphics g);
 }
```

Весь подсчёт физики заключается в этом методе:

```
 public void CalculatePhysics()
       {
          //Если правый край, то dx - в другую сторону и затухает по модулю
          if (position.X > 750 && dx > 0)
          {
             dx /= 2;
             dx *= -1;
          }
          //Если левый край, то dx - в другую сторону и затухает по модулю
```

```
    if (position.X < 0 && dx<0)
    {
        dx /= 2;
        dx *= -1;
    }
    position.X += dx;

    if (position.Y < 500 || a<0)
    {
        if (a != 0.4f)
        {
            a += 0.05f;
        }
        position.Y += gravity;

        gravity += a;
    }
    else
    {
        //Если коснулся пола, то уменьшаем импульс, пока сам не остановится
        if (position.Y >= 500)
        {
            if (Math.Abs(impulse) > 0.1f)
            {
                impulse /= 2;
                AddForce(a/2);
            }
        }
    }
}
```

Методы добавления силы:

```
public void AddForce(float forceValue)

    {

        gravity=-gravity;

        gravity /= 2;

        a = -forceValue;

    }
```

```
//Пинаем с огромной силой
public void AddForceQuickly(float forceValue)
{
    gravity = forceValue*30;
    gravity *= -1;
    a = -forceValue;
    impulse = -15;
    dx += 5;
}
```

События Формы:

```
public void Init()
    {
      ball = new Entity(new Size(50, 50));

      timer1.Interval = 10;
      timer1.Tick += new EventHandler(update);
      timer1.Start();
    }

    private void update(object sender, EventArgs e)
    {
      ball.myPhysics.ApplyPhysics();
      Invalidate();
    }

    private void OnPaint(object sender, PaintEventArgs e)
    {
      Graphics g = e.Graphics;
      ball.DrawSprite(g);
    }

    private void button1_Click(object sender, EventArgs e)
    {
      ball.myPhysics.AddForceQuickly(0.2f);
    }
```

Вторая модель

Создана в консольном приложении на C++, с помощью библиотеки "glut.h"

Основные моменты:

Switch с различными вариантами исхода:

```
switch (num)
	{
		case 1:

				gravity.add(CelestialObject(Vector2(0, -100), Vector2(0, 0), 15,
				5973, Color(0, 0, 255)));

				gravity.add(CelestialObject(Vector2(0, 184), Vector2(0.6f, 0.), 4,
				735, Color(0, 0, 0)));
			break;
		case 2:

				gravity.add(CelestialObject(Vector2(60, 0), Vector2(0., 1.25f), 5, 1,
				Color(0, 0, 0)));

				gravity.add(CelestialObject(Vector2(0, 0), Vector2(0, 0), 40, 5973,
				Color(0, 0, 255)));
			break;
		case 3:

				gravity.add(CelestialObject(Vector2(60, 0), Vector2(0.275f, 1.27f),
				7, 1, Color(0, 0, 0)));

				gravity.add(CelestialObject(Vector2(0, 0), Vector2(0, 0), 40, 5973,
				Color(0, 0, 255)));
			break;
		case 4:
			srand(time(0));
			for (int i = 1; i != 70; i++)
			{
				int r = rand() % 20 + 5;
				int m = r * 160;
				Vector2 position(1500 - rand() % 3000, 750 - rand() % 1500);
				Vector2 direction(1 - rand() % 2, (1 - rand() % 2));
				Color color(0, 0, 0);
				gravity.add(CelestialObject(position, direction, r, m, color));
			}
			break;
		default:
			std::cout << "Wrong number";
			exit(-1);

	}
```

Самый главный метод всего решения -- это void Space::accelerationObjects(),

если объекты столкнулись, то мы увеличиваем массу большего на 90% массы

меньшего, рассчитываем новую скорость, в зависимости от соотношения масс столкнувшихся объектов, иначе рассчитываем ускорение, исходя из масс.

Заключение

Данная курсовая работа была выполнена в соответствии с поставленной задачей в среде **Microsoft Visual Studio 2019**. В процессе разработки приложения было проведено исследование компонентов программной среды **Microsoft Visual Studio 2019**, которые использовались при создании программы. Были изучены Windows Forms и библиотека “glut.h”.

Итогом курсовой работы стали готовые физические модели, которые носят не только развлекательный характер, а также развивают внимательность и помогают изучать физику наглядно. Они подойдут, как взрослому человеку, так и ребенку школьного возраста.

Как и любое приложение, данный проект можно улучшить. В качестве ближайших улучшений можно рассматривать:

1. Улучшение физической модели мяча;
2. Добавление новых режимов симуляции в симуляции гравитации в космосе.

Список используемых источников

1. Что такое физика
2. Механика
3. Геостационарная орбита — Википедия
4. Высокая эллиптическая орбита — Википедия
5. Интегрированная среда разработки — Википедия
6. Microsoft Visual Studio — Википедия
7. C++
8. Обзор языка C# — руководство по C
9. OpenGL — Википедия
10. GLUT — Википедия