Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«Владимирский государственный университет имени

Александра Григорьевича и Николая Григорьевича Столетовых»

Кафедра информационных систем и программной инженерии

*КУРСОВОЙ ПРОЕКТ*

*по дисциплине: «Программирование компьютерной графики»*

Разработка графического приложения

на C# с использованием OpenGL

Предметная область:

«Космический корабль»

*Выполнил:* студент гр. ПРИ-120

Карпов М.В.

*Принял:* Жигалов И. Е.

Владимир, 2024

Аннотация

Курсовой проект выполнен по дисциплине «Программирование компьютерной графики». Темой курсового проекта является «Космический корабль». Предметом курсового проекта является разработка программы, формирующей трёхмерную графическую сцену с космическим кораблём использованием технологии OpenGL. Предметной областью является среда Visual C#.

The course project was carried out in the discipline "Computer graphics programming". The theme of the course project is "Spaceship". The subject of the course project is the development of a program that forms a three-dimensional graphic scene with a spaceship using OpenGL technology. The subject area is the Visual C# environment.

Содержание

[ВВЕДЕНИЕ 5](#_Toc160915698)

[1. Техническое задание 6](#_Toc160915699)

[1.1 Наименование приложения 6](#_Toc160915700)

[1.2 Глоссарий 6](#_Toc160915701)

[1.3 Описание 6](#_Toc160915702)

[1.4 Назначение и цель разработки 6](#_Toc160915703)

[1.5 Краткая характеристика и область применения 7](#_Toc160915704)

[1.6 Сценарий работы приложения 7](#_Toc160915705)

[1.7 Функциональные требования 7](#_Toc160915706)

[1.8 Нефункциональные требования 8](#_Toc160915707)

[1.8.1 Требования к реализации 8](#_Toc160915708)

[1.8.2 Требования к производительности 8](#_Toc160915711)

[1.8.3 Требования к хранению данных 8](#_Toc160915714)

[1.8.4 Требования к безопасности 8](#_Toc160915715)

[1.8.5 Требования к интерфейсу 9](#_Toc160915716)

[1.8.6 Требования к надёжности 9](#_Toc160915717)

[1.9 Инструменты разработки 9](#_Toc160915718)

[1.10 Интерфейс приложения 9](#_Toc160915719)

[1.11 Требования к программной документации 10](#_Toc160915720)

[1.12 Стадии и этапы разработки 10](#_Toc160915721)

[1.13 Порядок контроля и приёмки 10](#_Toc160915722)

[2. Цели и задачи проекта 11](#_Toc160915723)

[2.1 Цели и задачи 11](#_Toc160915724)

[3. Описание предметной области 12](#_Toc160915725)

[3.1 Моделируемая сцена 12](#_Toc160915726)

[3.2 Моделируемые объекты 13](#_Toc160915727)

[3.3 Моделируемые события 14](#_Toc160915728)

[4. Проектирование 15](#_Toc160915729)

[4.1 Описание архитектуры 15](#_Toc160915730)

[4.2 Описание технологий 15](#_Toc160915731)

[5. Моделирование объектов 16](#_Toc160915732)

[5.1 Моделирование фона 16](#_Toc160915733)

[5.2 Моделирование космического корабля 17](#_Toc160915734)

[5.3 Моделирование Луна 20](#_Toc160915735)

[5.4 Моделирование метеоритов 21](#_Toc160915736)

[5.5 Моделирование Би-Би 22](#_Toc160915737)

[6. Моделирование событий 25](#_Toc160915738)

[6.1 Логика работы приложения. Управление камерой 25](#_Toc160915739)

[6.2 Циклическая анимация космического корабля 25](#_Toc160915740)

[6.3 Игра с метеоритами 27](#_Toc160915741)

[6.4 Перемещение космического корабля и Би-Би 28](#_Toc160915742)

[ЗАКЛЮЧЕНИЕ 30](#_Toc160915743)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 31](#_Toc160915744)

[ПРИЛОЖЕНИЕ 32](#_Toc160915745)

ВВЕДЕНИЕ

Курсовой проект выполнен по дисциплине «Программирование компьютерной графики». Темой курсового проекта является «Разработка графического приложения на C# с использованием OpenGL». Предметной областью является космический корабль.

C# — объектно-ориентированный язык программирования общего назначения. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework и .NET Core.

Microsoft .NET Framework – это предложенная компанией Microsoft программная технология, основной задачей которой является предоставление разработчику набора удобных средств для разработки как простых программ, так и web-приложений.

OpenGL (Open Graphics Library — открытая графическая библиотека, графическое API) — спецификация, определяющая независимый от языка программирования платформонезависимый программный интерфейс для написания приложений, использующих двухмерную и трёхмерную компьютерную графику.

Компьютерная графика — это область информационных технологий, в которой создают и обрабатывают графические изображения на компьютерах. С помощью компьютерной графики можно делать двухмерные и трехмерные картинки, анимацию, виртуальную реальность, обрабатывать изображения, разрабатывать компьютерные игры и многое другое.

1. Техническое задание
   1. Наименование приложения  
       Программа для интерактивного управления заданной графической сценой «Космический корабль»
   2. Глоссарий

*OpenGL* – спецификация, определяющая платформонезависимый программный интерфейс для написания приложений, использующих двумерную и трёхмерную компьютерную графику.

*Графическое приложение –* приложение, которое использует графический интерфейс пользователя, те. позволяет пользователю перемещаться по приложению и выполнять определённые действия с помощью визуальных индикаторов и графических значков.

*Примитив* – заранее определённый основной геометрический элемент, при помощи которых строятся более сложные модели.

*C#* - современный объектно-ориентированный и типобезопасный язык программирования. C# позволяет разработчикам создавать разные типы безопасных и надежных приложений, выполняющихся в . NET

* 1. Описание

В разработанном приложении будет осуществлена возможность интерактивно управлять заданной трёхмерной графической сценой «Космический корабль».

* 1. Назначение и цель разработки

Назначение разработки - управление анимированными объектами. Цель разработки состоит в освоении средств разработки приложений в среде Visual Studio с использованием графической библиотеки OpenGL.

* 1. Краткая характеристика и область применения

Для создания трёхмерной графической сцены будут использованы возможности C# и OpenGL по созданию графических приложений, формированию простых графических объектов из примитивов, применению геометрических преобразований, использованию цветов, текстур и освещения со специальными эффектами, пиксельных операций, анимации сцены и управлению сценой

* 1. Сценарий работы приложения

1. Запуск приложения
2. Основной функционал:
   1. Космический корабль:
      1. Циклическая анимация корабля
   2. Игра:
      1. Управляя кораблём, уклоняться от метеоритов
   3. Луна:
      1. Передвигать Би-Би по Луне
3. Просмотр информации о программе
4. Завершение работы приложения
   1. Функциональные требования

* Циклическая анимация космического корабля
* Воспроизведение аудио «Космическая Одиссея» в сцене №1 и №3
* Воспроизведение аудио «От винта!» в сцене №2
* Перемещение космического корабля нажатием на клавиши
* Перемещение Би-Би нажатием на кнопки
* Нажатием на кнопку начинать/перезапускать игру
* При попадании метеорита в корабль анимировать отклонение корабля назад
* При попадании метеорита в корабль отображать надпись о проигрыше и кнопку «Играть» повторно
* При прохождении игры добавить на сцену систему частиц и надпись о победе
* Переключение управления между объектами
  1. Нефункциональные требования
     1. Требования к реализации
* Использование возможностей C#
* Использование возможностей OpenGL
  + 1. Требования к производительности

• Приложение должно сохранять высокую отзывчивость в течение всего времени пользования приложением.

• Приложение не должно использовать большой объём памяти

* + 1. Требования к хранению данных

Сохранность информации должна быть обеспечена в следующих случаях:

* выход из строя аппаратных систем комплекса;
* хищение носителей информации, других систем комплекса;
* ошибки в программных средствах.
  + 1. Требования к безопасности
* Защита программы от попыток изменения и разрушения
* Защищённый доступ к исходному коду
  + 1. Требования к интерфейсу
* Шрифт ввода-вывода данных - кириллица;
* Интерфейс должен быть удобен и прост в понимании для пользователя.
  + 1. Требования к надёжности
* Для надежной работы программы необходимы высоконадежные аппаратные и программные системы.
  1. Инструменты разработки

*Microsoft Visual Studio* - это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Помимо стандартного редактора и отладчика, которые есть в большинстве сред IDE, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для улучшения процесса разработки.

*Open GL* - cпецификация, определяющая платформонезависимый 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.

* 1. Интерфейс приложения

В качестве среды визуализации модели для реализации поставленной задачи была выбрана библиотека OpenGL. Программы, использующие OpenGL, гарантируют одинаковый визуальный результат во многих операционных системах - на персональных компьютерах, на рабочих станциях и на суперкомпьютерах.

* 1. Требования к программной документации

Законченный курсовой проект включает:

- разработанную программу (исходные коды с проектом, исполняемый файл и все файлы, необходимые для его работы),

- пояснительную записку (включающую программную документацию),

- презентацию.

* 1. Стадии и этапы разработки

7 семестр, 1 рейтинг: разработка и утверждение технического задания на проектирование;

7 семестр, 2 рейтинг: выполнение и документирование элементов КП, относящихся к тематике 1 и 2 частей практикума по дисциплине;

7 семестр, 3 рейтинг: выполнение и документирование элементов КП, относящихся к тематике 1-3 частей практикума по дисциплине;

8 семестр, 1 рейтинг: выполнение и документирование элементов КП, относящихся к тематике 1-4 частей практикума по дисциплине;

8 семестр, 2 рейтинг: завершение программной части, сдача готовой пояснительной записки к КП с сопроводительной документацией и иллюстративным материалом, защита КП

* 1. Порядок контроля и приёмки

За несвоевременное предоставление материалов по этапам разработки КП баллы снижаются.

За непредоставление законченного курсового проекта в назначенный срок снимается 10 баллов.

Итоговая оценка за проект выставляется в соответствие с набранными баллами.

1. Цели и задачи проекта
2. Цели и задачи

Цель проекта состоит в разработке настольного приложения в соответствии с техническим заданием по предметной области «Космический корабль», которое реализует трехмерное отображение сцены.

Для достижения поставленной цели необходимо решить следующие задачи:

* смоделировать сцену, ландшафт, объекты и события сцены;
* разработать алгоритм и дизайн приложения;
* разработать интерфейс пользователя;
* реализовать приложение;
* протестировать приложение;
* разработать программную документацию.

1. Описание предметной области
   1. Моделируемая сцена

Моделируемая сцена представляет из себя космическое пространство с некоторыми космическими объектами. В качестве прототипа данной сцены взята серия мультсериала «Смешарики» «Космическая Одиссея». В качестве фона в каждой из трёх сцен выступает текстура «Космос». Изначально перед пользователем предстаёт сцена №1 под названием «Корабль».

В сцене №1 представлен космический корабль. Здесь же применена циклическая анимация, включающая в себя вращение корабля по всем осям. На фоне играет финальная композиция серии мультсериала для задания атмосферы.

Сцена №2 под названием «Игра» копирует сюжет серии, где герои попадают под метеоритный дождь. При нажатии на кнопку «Играть» на корабль начнут лететь метеориты, а цель пользователя – используя клавиши клавиатуры, уклоняться от них. Всего в корабль летят 4 метеорита. Если хотя бы один из них попадёт в корабль – игра закончится проигрышем, если ни один не попадёт – победой и салютом из системы частиц. Если в корабль попадет метеорит, он отклоняется назад, а свет в окне гаснет. К сожалению, не удалось найти нужную композицию из этого фрагмента серии, поэтому в качестве саундтрека выступает не менее прекрасная песня «От винта!».

Сцена №3 под названием «Луна» представляет из себя, как ни сложно догадаться, Луну. Также на Луне находится робот «Би-Би», на поиски которого персонажи серии и полетели. Нажатием на кнопки Би-би можно передвигать по Луне. Одновременно с перемещением он также поворачивается в нужную сторону. На фоне играет всё та же финальная композиция серии.

* 1. Моделируемые объекты

Первым моделируемым объектом является космический корабль главных героев, представленный на рисунке 1. Здесь же представлены метеориты, которые летят в корабль:



Рисунок 1. Космический корабль

Третий моделируемый объект – Луна (рисунок 2)



Рисунок 2. Луна

Четвёртым же объектом является робот Би-Би, сын Пина. Он представлен на рисунке 3.

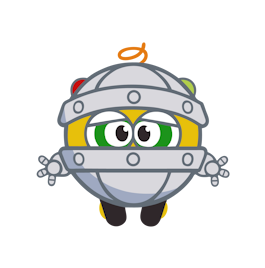


Рисунок 3. Би-Би

3.3 Моделируемые события

* При победе в игре запускать систему частиц
* При проигрыше в игре анимировать отклонение корабля и потушить свет в окне
* По нажатию клавиш перемещать космический корабль
* По нажатию клавиш перемещать Би-Би
* Циклическая анимация космического корабля

1. Проектирование
   1. Описание архитектуры

Консольное приложение для интерактивного управления заданной трёхмерной графической сценой с космическим кораблём. Приложение можно запустить на компьютере и передать управление пользователю.

* 1. Описание технологий

*OpenGL* – это программный интерфейс для графических устройств. Он включает в себя около 150 различных функций, с помощью которых программист может задавать свойства различных трехмерных и двумерных объектов и выполнять их визуализацию (рендеринг).

*Tao Framework* – это свободно-распространяемая библиотека с открытым исходным кодом, предназначенная для быстрой и удобной разработки кросс-платформенного мультимедийного программного обеспечения в среде .NET Framewrok и Mono.

*C#* - объектно-ориентированный язык программирования.

*.Net Framework* – программная платформа, выпущенная компанией Microsoft в 2002 году.

1. Моделирование объектов
   1. Моделирование фона

Фон представляет из себя текстуру космоса.

Код отрисовки представлен ниже:

//Фон (космос)

public void drawBackground(uint backgroundTexture)

{

Gl.glPushMatrix();

Gl.glRotated(-90, 0, 1, 0);

Gl.glEnable(Gl.GL\_TEXTURE\_2D);

Gl.glBindTexture(Gl.GL\_TEXTURE\_2D, backgroundTexture);

Gl.glPushMatrix();

Gl.glBegin(Gl.GL\_QUADS);

Gl.glVertex3d(-200, 5, 400);

Gl.glTexCoord2f(0, 0);

Gl.glVertex3d(-200, 5, -400);

Gl.glTexCoord2f(0, 1);

Gl.glVertex3d(200, 5, -400);

Gl.glTexCoord2f(1, 1);

Gl.glVertex3d(200, 5, 400);

Gl.glTexCoord2f(1, 0);

Gl.glEnd();

Gl.glPopMatrix();

Gl.glDisable(Gl.GL\_TEXTURE\_2D);

Gl.glPopMatrix();

}

Фон представлен на рисунке 4.



Рисунок 4. Фон

* 1. Моделирование космического корабля

Над космическим кораблём была проделана очень большая работа. Основа состоит из сферы (передняя часть) и цилиндра, повёрнутого на 90 градусов по оси Y. Турбины состоят из тех же частей, лишь смещённых и отмасштабированных, рисуются они с помощью цикла. Также здесь используются конусы, сильно уменьшенные по оси X. Для окна используются две сферы.

Код отрисовки космического корабля представлен ниже:

//Космический корабль

public void drawSpaceship(double delta, Scene scene, bool isSpaceshipIntact, double deltaX)

{

Gl.glPushMatrix();

switch (scene)

{

case Scene.First:

Gl.glTranslated(0 - deltaX, -40, 0+delta);

Gl.glRotated(delta, 0, 1, 0);

Gl.glRotated(delta, 1, 0, 0);

Gl.glRotated(delta, 0, 0, 1);

break;

case Scene.SecondGameNotStarted:

Gl.glTranslated(0 - deltaX, -40, 0 + delta);

break;

case Scene.SecondGameStarted:

Gl.glTranslated(0, -40, 0 + delta);

break;

}

Gl.glPushMatrix();

setColor(0.95f, 0.7f, 0.12f);

Gl.glScalef(2, 1, 1);

Glut.glutSolidSphere(30, 12, 12);

Gl.glColor3f(0.6f,0.35f,0.02f);

Gl.glLineWidth(3f);

Glut.glutWireSphere(30 , 15, 15);

Gl.glPopMatrix();

Gl.glPushMatrix();

Gl.glTranslated(-50, 0, 0);

setColor(0.95f, 0.7f, 0.12f);

Gl.glRotatef(90, 0, 1, 0);

Glut.glutSolidCylinder(30, 50, 12, 12);

Gl.glColor3f(0.6f, 0.35f, 0.02f);

Gl.glLineWidth(3f);

Glut.glutWireCylinder(30, 50, 15, 15);

Gl.glPopMatrix();

Gl.glPushMatrix();

Gl.glTranslated(-60, 0, -5);

Gl.glScalef(0.5f, 0.3f, 0.3f);

for (int i =0; i<3; i++)

{

Gl.glRotatef(72\*i, 1, 0, 0);

Gl.glPushMatrix();

Gl.glTranslated(0, 0, 60);

setColor(0.4f, 0.09f, 0.6f);

Glut.glutSolidSphere(30, 12, 12);

Gl.glColor3f(0.2f, 0f, 0.4f);

Gl.glLineWidth(3f);

Glut.glutWireSphere(30, 15, 15);

Gl.glPopMatrix();

Gl.glPushMatrix();

Gl.glTranslated(-30, 0, 60);

setColor(0.4f, 0.09f, 0.6f);

Gl.glRotatef(90, 0, 1, 0);

Glut.glutSolidCylinder(30, 30, 12, 12);

Gl.glColor3f(0.2f, 0f, 0.4f);

Gl.glLineWidth(3f);

Glut.glutWireCylinder(30, 30, 15, 15);

Gl.glPopMatrix();

if (scene == Scene.SecondGameStarted)

{

Gl.glPushMatrix();

Gl.glTranslated(-30, 0, 60);

Gl.glScalef(2, 1, 1);

setColor(0.9f, 0.09f, 0.03f);

Gl.glRotatef(90, 0, 1, 0);

Glut.glutSolidSphere(30, 12, 12);

Gl.glColor3f(1,1,1);

Gl.glLineWidth(5f);

Glut.glutWireSphere(30, 15, 15);

Gl.glPopMatrix();

}

}

Gl.glPopMatrix();

for (int i = 0; i<2; i++)

{

Gl.glPushMatrix();

if (i == 0) Gl.glTranslated(-50, -30, 0); else Gl.glTranslated(-50, 30, 0);

setColor(0.95f, 0.05f, 0.02f);

Gl.glRotatef(90, 0, 1, 0);

Gl.glScalef(0.15f, 0.6f, 1f);

Glut.glutSolidCone(30, 50, 12, 12);

Gl.glColor3f(1f, 0.1f, 0.2f);

Gl.glLineWidth(3f);

Glut.glutWireCone(30, 50, 15, 15);

Gl.glPopMatrix();

}

for (int i = 0; i < 2; i++)

{

Gl.glPushMatrix();

if (i == 0)

{

Gl.glTranslated(-50, 0, 30);

setColor(0.2f, 0.15f, 0.9f);

}

else

{

Gl.glTranslated(-50, 0, -30);

setColor(0.95f, 0.05f, 0.02f);

}

Gl.glRotatef(90, 0, 1, 0);

Gl.glScalef(0.6f, 0.15f, 1f);

Glut.glutSolidCone(30, 50, 12, 12);

Gl.glLineWidth(3f);

Glut.glutWireCone(30, 50, 15, 15);

Gl.glPopMatrix();

}

//Окно

for (int i = 0; i < 2; i++)

{

Gl.glPushMatrix();

if (i == 0)

{

Gl.glTranslated(10, -30, 10);

Gl.glRotated(-18, 1, 0, 0);

}

else

{

Gl.glTranslated(10, 25, 10);

Gl.glRotated(-165, 1, 0, 0);

}

Gl.glPushMatrix();

setColor(0.95f, 0.1f, 0.12f);

Gl.glScalef(1, 0.1f, 1);

Glut.glutSolidSphere(13, 12, 12);

Gl.glColor3f(0.6f, 0.35f, 0.02f);

Gl.glLineWidth(3f);

Glut.glutWireSphere(13, 15, 15);

Gl.glPopMatrix();

Gl.glPushMatrix();

Gl.glTranslated(0, -3, 0);

if (isSpaceshipIntact)

setColor(0.8f, 1f, 0.4f);

else setColor(0.1f, 0.1f, 0.1f);

Gl.glScalef(1, 0.1f, 1);

Glut.glutSolidSphere(10, 12, 12);

Gl.glColor3f(0.6f, 0.35f, 0.02f);

Gl.glLineWidth(3f);

Glut.glutWireSphere(10, 15, 15);

Gl.glPopMatrix();

Gl.glPopMatrix();

}

Gl.glPopMatrix();

}

Модель космического корабля представлена на рисунке 5.

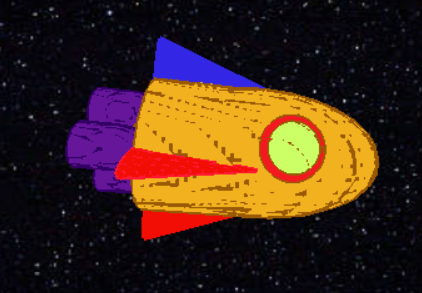


Рисунок 5. Космический корабль

* 1. Моделирование Луна

Луна – это большая сфера. Для создания кратеров использовались сферы меньшего радиуса, уменьшенные по оси Z.

Код отрисовки Луны:

//Луна

public void drawMoon()

{

Gl.glPushMatrix();

Gl.glTranslated(150, -40, 0);

Gl.glPushMatrix();

setColor(0.8f, 0.8f, 0.8f);

Glut.glutSolidSphere(50, 32, 32);

Gl.glPopMatrix();

Gl.glPushMatrix();

Gl.glTranslated(0,-40,32);

Gl.glRotated(-1, 1, 0, 0);

setColor(0.5f, 0.5f, 0.5f);

Gl.glScalef(1, 1f, 0.1f);

Glut.glutSolidSphere(1, 32, 12);

Gl.glPopMatrix();

Gl.glPushMatrix();

Gl.glTranslated(-7, -41, 28);

Gl.glRotated(50, 1, 0, 0);

setColor(0.5f, 0.5f, 0.5f);

Gl.glScalef(1, 1f, 0.1f);

Glut.glutSolidSphere(1.3, 32, 12);

Gl.glPopMatrix();

Gl.glPushMatrix();

Gl.glTranslated(-15, -45, 20);

Gl.glRotated(50, 1, 0, 0);

setColor(0.5f, 0.5f, 0.5f);

Gl.glScalef(1, 1f, 0.1f);

Glut.glutSolidSphere(1.7, 32, 12);

Gl.glPopMatrix();

Gl.glPushMatrix();

Gl.glTranslated(10, -40, 32);

Gl.glRotated(40, 1, 0, 0);

setColor(0.5f, 0.5f, 0.5f);

Gl.glScalef(1, 1f, 0.1f);

Glut.glutSolidSphere(1.2, 32, 12);

Gl.glPopMatrix();

Gl.glPushMatrix();

Gl.glTranslated(10, -45, 20);

Gl.glRotated(60, 1, 0, 0);

setColor(0.5f, 0.5f, 0.5f);

Gl.glScalef(1, 1f, 0.1f);

Glut.glutSolidSphere(1.5, 32, 12);

Gl.glPopMatrix();

Gl.glPopMatrix();

}

Модель Луны представлена на рисунке 6.

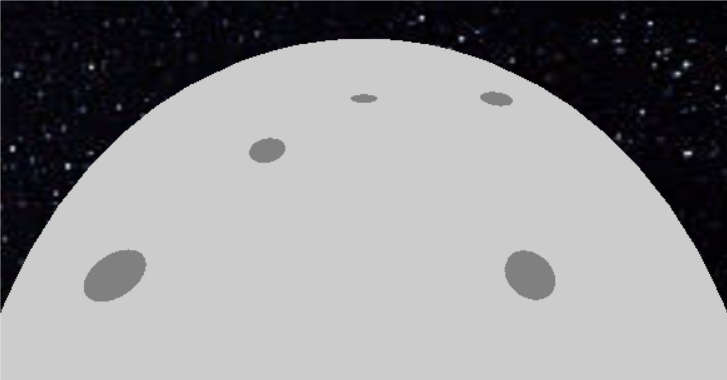


Рисунок 6. Модель Земли

* 1. Моделирование метеоритов

Метеориты представляют из себя обычные сферы серого цвета.

public void drawMeteorit(double deltaX, double translateZ)

{

Gl.glPushMatrix();

Gl.glTranslated(300 - deltaX, -40, translateZ);

setColor(0.2f,0.2f,0.2f);

Gl.glScalef(1, 1f, 1);

Glut.glutSolidSphere(10, 12, 12);

Gl.glColor3f(0,0,0);

Gl.glLineWidth(3f);

Glut.glutWireSphere(10, 15, 15);

Gl.glPopMatrix();

}

Модель метеорита представлена на рисунке 7.



Рисунок 7. Модель метеорита

* 1. Моделирование Би-Би

Основная часть Би-Би – сфера жёлтого цвета. Сверху и снизу него, его корпус – это уменьшенные по оси Z сферы серого цвета, нарисованы с помощью цикла. Колёса также рисуются с помощью цикла – две сферы чёрного цвета, уменьшенные по оси X. Глаза – это сферы, уменьшенные по оси Y. Руки и пальцы, лампочки на голове также нарисованы с помощью циклов.

//Рисуем Би-Би

public void drawBibi(RotateBibi rotate, double deltaX, double deltaY)

{

Gl.glPushMatrix();

Gl.glTranslated(150 + deltaX, -85 + deltaY, 23);

switch (rotate)

{

case RotateBibi.Back:

Gl.glRotated(180, 0, 0, 1);

break;

case RotateBibi.Left:

Gl.glRotated(-90, 0, 0, 1);

break;

case RotateBibi.Right:

Gl.glRotated(90, 0, 0, 1);

break;

}

//Тело

Gl.glPushMatrix();

setColor(0.7f, 0.7f, 0.05f);

Glut.glutSolidSphere(1.6, 32, 32);

Gl.glPopMatrix();

for (int i = 0; i <2; i++)

{

Gl.glPushMatrix();

if (i==0) Gl.glTranslated(0, 0, 0.85);

else Gl.glTranslated(0, 0, -0.8);

Gl.glScalef(1, 1f, 0.45f);

setColor(0.4f, 0.4f, 0.4f);

Glut.glutSolidSphere(1.65, 32, 32);

Gl.glColor3f(0.3f, 0.3f, 0.3f);

Gl.glLineWidth(3f);

Glut.glutWireSphere(1.65, 10, 10);

Gl.glPopMatrix();

}

//Колёса

for (int i = 0; i < 2; i++)

{

Gl.glPushMatrix();

if (i == 0) Gl.glTranslated(-1, 0, -1.5);

else Gl.glTranslated(1, 0, -1.5);

Gl.glScalef(0.6f, 1f, 1f);

setColor(0,0,0);

Glut.glutSolidSphere(0.5, 32, 32);

Gl.glPopMatrix();

}

//Глаза

for (int i = 0; i < 2; i++)

{

Gl.glPushMatrix();

if (i == 0) Gl.glTranslated(-0.5, -1.6, 0);

else Gl.glTranslated(0.5, -1.6, 0);

Gl.glScalef(1f, 0.1f, 1f);

setColor(1,1,1);

Glut.glutSolidSphere(0.4, 32, 32);

Gl.glTranslated(0, -0.4, 0);

setColor(0, 1, 0);

Glut.glutSolidSphere(0.25, 32, 32);

Gl.glTranslated(0, -0.25, 0);

setColor(0,0,0);

Glut.glutSolidSphere(0.2, 32, 32);

Gl.glPopMatrix();

}

//Руки

for (int i = 0; i < 2; i++)

{

Gl.glPushMatrix();

if (i == 0)

{

Gl.glTranslated(-1.8, 0, -0.7);

Gl.glRotated(40, 0, 1, 0);

}

else

{

Gl.glTranslated(1.8, 0, -0.7);

Gl.glRotated(-40, 0, 1, 0);

}

Gl.glPushMatrix();

Gl.glScalef(0.4f, 0.4f, 0.7f);

setColor(0.4f, 0.4f, 0.4f);

Glut.glutSolidSphere(0.5, 32, 32);

Gl.glColor3f(0.3f, 0.3f, 0.3f);

Gl.glLineWidth(3f);

Glut.glutWireSphere(0.5, 10, 10);

Gl.glPopMatrix();

Gl.glPushMatrix();

Gl.glTranslated(0, 0, -0.35);

setColor(0.4f, 0.4f, 0.4f);

Glut.glutSolidSphere(0.2, 32, 32);

Gl.glColor3f(0.3f, 0.3f, 0.3f);

Gl.glLineWidth(3f);

Glut.glutWireSphere(0.2, 10, 10);

Gl.glPopMatrix();

for (int j = 0; j < 3; j++)

{

Gl.glPushMatrix();

Gl.glTranslated(0, 0, -0.35);

Gl.glRotated(50 \* j, 0, 1, 0);

Gl.glScalef(0.3f, 0.4f, 0.9f);

setColor(0.4f, 0.4f, 0.4f);

Glut.glutSolidSphere(0.4, 32, 32);

Gl.glColor3f(0.3f, 0.3f, 0.3f);

Gl.glLineWidth(3f);

Glut.glutWireSphere(0.4, 10, 10);

Gl.glPopMatrix();

}

Gl.glPopMatrix();

}

//Лампочки

for (int i = 0; i < 2; i++)

{

Gl.glPushMatrix();

if (i == 0)

{

Gl.glTranslated(-1, 0, 1.4);

setColor(1, 0,0);

}

else

{

Gl.glTranslated(1, 0, 1.4);

setColor(0, 1, 0.6f);

}

Glut.glutSolidSphere(0.3, 32, 32);

Gl.glPopMatrix();

}

Gl.glPopMatrix();

}

Модель Би-Би представлена на рисунке 8.



Рисунок 8. Модель Би-Би

1. Моделирование событий
   1. Логика работы приложения. Управление камерой

Приложение начинает свою работу и перед пользователем предстаёт сцена с космическим кораблём.

Выбор позиции камеры:

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

if (comboBox1.SelectedIndex == 0)

{

angle = 3; angleX = -90; angleY = 0; angleZ = 0;

sizeX = 1; sizeY = 1; sizeZ = 1;

translateX = -40; translateY = 200; translateZ = -25;

initFirstPosition();

WMP.URL = @"kikoriki.mp3";

WMP.controls.play();

}

if (comboBox1.SelectedIndex == 1)

{

translateX = -60; translateY = 300; translateZ = -25;

angle = 3; angleX = -90; angleY = 0; angleZ = 0;

initSecondPosition();

WMP.URL = @"ot\_vinta.mp3";

WMP.controls.play();

}

if (comboBox1.SelectedIndex == 2)

{

translateX = -150; translateY = 100; translateZ = -25;

angle = 3; angleX = -90; angleY = 0; angleZ = 0;

initThirdPosition();

WMP.URL = @"kikoriki.mp3";

WMP.controls.play();

}

AnT.Focus();

}

Перемещение по конкретной координате определяется в выпадающем списке «Выбранная ось», чтобы крутиться по оси используем клавиши «R» и «E».

Для перемещения по сцене используются стандартные игровые клавиши «W», «A», «S», «D».

* 1. Циклическая анимация космического корабля

Циклическая анимация космического корабля представляет из себя его вращение по разным осям. Осуществляется оно с помощью переменной deltaRotateSystemSpaceship. Функция changeSpaceshipRotate() занимается перерасчётом отклонения корабля.

private void drawFirstScene()

{

spaceship.drawSpaceship(deltaRotateSystemSpaceship, Scene.First, isSpaceshipIntact, deltaXSpaceShip);

changeSpaceshipRotate();

}

private void changeSpaceshipRotate()

{

if (deltaRotateSystemSpaceship >= 10)

{

isSpaceshipUp = false;

deltaRotateSystemSpaceship = 9;

}

else if (deltaRotateSystemSpaceship <= -10)

{

isSpaceshipUp = true;

deltaRotateSystemSpaceship = -9;

}

else if (isSpaceshipUp) deltaRotateSystemSpaceship++;

else deltaRotateSystemSpaceship--;

}

На рисунках 9 – 10 представлены положения корабля.

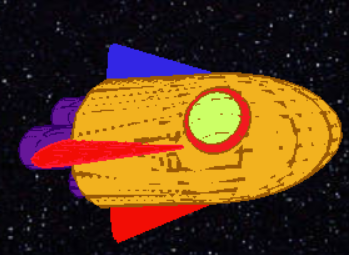


Рисунок 9. Положение 1

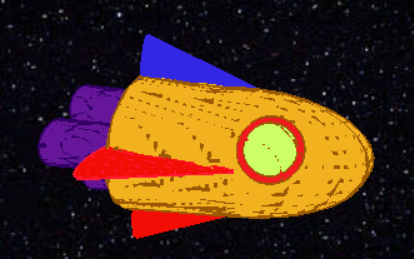


Рисунок 10. Положение 2

* 1. Игра с метеоритами

При нажатии на кнопку «Играть» можно начать игру. Нажатием на клавиши «I» и «K» можно перемещать космический корабль вверх и вниз соответственно. Всего по очереди запускаются 4 метеорита. Их положение по оси Z генерируется с помощью класса Random, а перемещение определяется с помощью переменной deltaXMeteorit. Если метеорит находится в зоне космического корабля по оси X и значения отклонения по оси Z метеорита и корабля менее 35, то значит метеорит попал в корабль. Игра заканчивается, снова появляется кнопка «Играть». Запускается анимация рисования отклонения корабля после столкновения, а окно закрашивается чёрным цветом.

Если же все 4 метеорита не попали в корабль, то игра считается успешно завершённой, запускается система частиц.

Данный функционал реализован следующим образом:

private void drawSecondScene()

{

if (!isGameStarted)

{

if (!isSpaceshipIntact && isSpaceshipTranclatingAfterCrash)

{

if (deltaXSpaceShip >= 60)

{

isSpaceshipTranclatingAfterCrash = false;

}

else

{

deltaXSpaceShip += 5 \* power / 50;

power--;

}

}

spaceship.drawSpaceship(deltaZUserSpaceship, Scene.SecondGameNotStarted, isSpaceshipIntact, deltaXSpaceShip);

}

else

{

spaceship.drawSpaceship(deltaZUserSpaceship, Scene.SecondGameStarted, isSpaceshipIntact, deltaXSpaceShip);

spaceship.drawMeteorit(deltaXMeteorit, translateZMeteorit);

checkDeltaMeteorit();

}

}

private void checkDeltaMeteorit()

{

if (deltaXMeteorit > 500)

{

countOfMeteorits++;

if (countOfMeteorits == 4)

{

isGameWon = true;

isGameStarted = false;

initSecondPosition();

Gl.glPushMatrix();

explosion.SetNewPosition(40, -280, 15);

explosion.SetNewPower(50);

explosion.Boooom(global\_time);

Gl.glPopMatrix();

}

else

{

deltaXMeteorit = 0;

translateZMeteorit = random.Next(-100, 100);

}

}

else if (deltaXMeteorit > 230 && deltaXMeteorit < 420)

{

if ((deltaZUserSpaceship > translateZMeteorit && deltaZUserSpaceship - translateZMeteorit <= 35) ||

(deltaZUserSpaceship <= translateZMeteorit) && translateZMeteorit - deltaZUserSpaceship <= 35)

{

isGameLost = true;

isGameStarted = false;

isSpaceshipIntact = false;

isSpaceshipTranclatingAfterCrash = true;

initSecondPosition();

}

else

deltaXMeteorit += 10;

}

else

deltaXMeteorit+=10;

}

* 1. Перемещение космического корабля и Би-Би

Если индекс позиции камеры равен 1 и нажата клавиша I или K, космический корабль перемещается вверх или вниз соответственно.

Если индекс позиции камеры равен 2 и нажата клавиша F,T,G или H, то Би-Би перемещается влево, вверх вниз или вправо соответственно, плюс поворачивается в нужную сторону.

Данный функционал реализован с помощью функции AnT\_KeyDown:

private void AnT\_KeyDown(object sender, KeyEventArgs e)

{

…

switch (comboBox1.SelectedIndex)

{

case 1:

if (e.KeyCode == Keys.I) deltaZUserSpaceship++; else

if (e.KeyCode == Keys.K) deltaZUserSpaceship--;

break;

case 2:

if (e.KeyCode == Keys.F && deltaXBibi>-10)

{

deltaXBibi--;

rotateBibi = RotateBibi.Left;

}

else if (e.KeyCode == Keys.H && deltaXBibi < 10)

{

deltaXBibi++;

rotateBibi = RotateBibi.Right;

}

else if (e.KeyCode == Keys.G && deltaYBibi > -7)

{

deltaYBibi--;

rotateBibi = RotateBibi.Forward;

}

else if (e.KeyCode == Keys.T && deltaYBibi < 0)

{

deltaYBibi++;

rotateBibi = RotateBibi.Back;

}

break;

}

}

# ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта по дисциплине «Программирование компьютерной графики» была достигнута цель - освоены средства разработки приложений в среде Microsoft C# с использованием графической библиотеки OpenGL.

В соответствии с выбранной предметной областью «Космический корабль» было спроектировано и разработано графическое приложение на С# с использованием технологии OpenGL.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Жигалов И.Е. Компьютерная графика: Курс лекций/ Владим. гос. ун-т. Владимир, 2004. 124 с. ISBN 5-89368-459-1.
2. Жигалов И.Е., Новиков И.А. Программирование компьютерной графики. Учебное пособие./ Владим. гос. ун-т. Владимир, 2014. 96 с.
3. Жигалов И.Е., Новиков И.А. Программирование двухмерной компьютер-ной графики. Учебное пособие./ Владим. гос. ун-т. Владимир, 2015. 120 с.
4. Жигалов И. Е., Новиков И.А. Программирование трехмерной компьютер-ной графики : учеб. пособие / Владим. гос. ун-т им. А. Г. и Н. Г. Столетовых. ‒Владимир: Изд-во ВлГУ, 2016. ‒92 с.
5. Павловская ТА. С#. Программирование на языке высокого уровня. Учеб-ник для вузов.-СПб.: Питер, 2009.
6. Порев В.Н. Компьютерная графика.–СПб.: БХВ-Петербург, 2002.-432 с.
7. Тарасов И.А. Основы программирования в OрenGL. -М.: Телеком, 2000.-188 с.
8. Тихомиров Ю. Программирование трехмерной графики.-СПб. BHV, 1998.-320с.
9. ТроелсенЭ.С# и платформа .NET. Библиотека программиста. 2004.
10. Херн Д. Компьютерная графика и стандарт OpenGL -Москва: Вильямс, 2005 .-1158 c., -ISBN 5-8459-0772-1
11. Эйнджел Э. Интерактивная компьютерная графика. Вводный курс на базе OрenGL. – М.: ИД «Вильямс», 2001. – 592 с.

# ПРИЛОЖЕНИЕ

## Тексты программ с комментариями

Весь код приложения доступен по адресу: <место для твоей ссылки>