# Введение

В данной курсовой работе необходимо реализовать игровую программу «Марио» стандартного типа. Также реализовать возможность звукового сопровождения, настройки игры, пять стандартных карт.

Интерес выбора данной темы в том, что помимо освоения работы с OpenGL, необходимо работать с графикой, звуком и проектированием игровой логики.

На базовом уровне, OpenGL — это просто спецификация, то есть документ, описывающий набор функций и их точное поведение. Производители оборудования на основе этой спецификации создают реализации — библиотеки функций, соответствующих набору функций спецификации. Реализация призвана эффективно использовать возможности оборудования. Если аппаратура не позволяет реализовать какую-либо возможность, она должна быть эмулирована программно. Производители аппаратуры проходят ряд специфических тестов (conformance tests — тесты на соответствие), прежде чем реализация будет классифицирована OpenGL-реализация. Так как разработчикам программного обеспечения достаточно научиться использовать функции, описанные в спецификации, их реализация остается разработчикам аппаратного обеспечения.

Цель работы: Создание специализированного программного обеспечения, для получения удовольствия и хорошего времяпровождения в игре жанра «Аркада».

Актуальность выбора темы обоснована тем, что последняя версия игры была выпущена очень давно и хотелось бы реализовать ее с помощью современных технологий.

Также тема курсовой работы достаточно актуальна в настоящее время. Это объясняется тем, что большинство людей играет в аркадные игры.

В качестве среды разработки выбрана Visual Studio 2012, язык программирования – C++.

Язык Си++ является универсальным языком программирования, в дополнение к которому разработан набор разнообразных библиотек. Поэтому, строго говоря, он позволяет решить практически любую задачу программирования. Тем не менее, в силу разных причин (не всегда технических) для каких-то типов задач данный язык употребляется чаще, а для каких-то – реже.

# 1 Анализ задания и постановка задач

* 1. **Описание предметной области**

Предметной областью разрабатываемой программы является игровой процесс «Марио» стандартного типа, который включает в себя:

* управление игровым персонажем по игровому полю;
* использование звукового сопровождения;
* настройки.

Звуковое сопровождение представляет следующие мелодии:

* главная тема;
* мелодия потери 1 жизни;
* мелодия прыжка;

Игрок проигрывает, если счетчик его жизней стал равен 0.

Графический интерфейс представляет собой игровую карту с действующими элементами, а также меню с настройками, началом новой игры и выходом из игры.

В данной игровой программе игрок будет управлять одним персонажем с помощью клавиатуры. Игровой процесс будет составлять продвижение игрока по виртуальному миру, перепрыгивая через движущихся ему навстречу врагов, и собирая бонусы, увеличивающие количество жизней.

Программа должна содержать математические расчеты, в основном связанные с игровой физикой.

# Постановка задач

Темой курсовой работы является разработка игровой программы «Марио» стандартного типа. Основной целью работы является оживить давно созданную игру.

Разрабатываемая программа должна иметь следующий функционал:

* пользователь может управлять игровым персонажем с помощью «стрелок»;
* уничтожать блоки;
* собирать бонусы из некоторых разрушенных блоков;
* уничтожать противников;
* включать\выключать звуковое сопровождение;
* выбирать скорость игры.

Программа включает в себя:

* генерацию персонажа, земли, врагов и бонусных блоков;
* создание панели меню, для получения информации о программе и выхода из программы;
* возможность передвижения персонажем по карте вперед, вверх, вниз и назад;

# 2 Проектирование программы

**2.1 Выбор и обоснование средств разработки**

Для разработки используется язык программирования С. C (рус. Си) – компилируемый статически типизированный язык программирования общего назначения, разработанный в 1969—1973 годах сотрудником Bell Labs Деннисом Ритчи как развитие языка Би. Первоначально был разработан для реализации операционной системы UNIX, но, впоследствии, был перенесён на множество других платформ. Согласно дизайну языка Си, его конструкции близко сопоставляются типичным машинным инструкциям, благодаря чему он нашёл применение в проектах, для которых был свойственен язык ассемблера, в том числе как в операционных системах, так и в различном прикладном ПО для множества устройств — от суперкомпьютеров до встраиваемых систем. Язык программирования Си отличается минимализмом. Авторы языка хотели, чтобы программы на нём легко компилировались с помощью однопроходного компилятора, чтобы каждой элементарной составляющей программы после компиляции соответствовало весьма небольшое число машинных команд, а использование базовых элементов языка не задействовало библиотеку времени выполнения. Однопроходный компилятор компилирует программу, не возвращаясь назад, к уже обработанному тексту. Поэтому использованию функций и переменных должно предшествовать их объявление. Код на Си можно легко писать на низком уровне абстракции, почти как на ассемблере. Иногда Си называют «универсальным ассемблером» или «ассемблером высокого уровня», что отражает различие языков ассемблера для разных платформ и единство стандарта Си, код которого может быть скомпилирован без изменений практически на любой модели компьютера. Си часто называют языком среднего уровня или даже низкого уровня, учитывая то, как близко он работает к реальным устройствам. Однако, в строгой классификации, он является языком высокого уровня.

Компиляторы Си разрабатываются сравнительно легко благодаря простоте языка и малому размеру стандартной библиотеки. Поэтому данный язык доступен на самых различных платформах (возможно, круг этих платформ шире, чем у любого другого существующего языка). К тому же, несмотря на свою низкоуровневую природу, язык позволяет создавать переносимые программы и поддерживает в этом программиста. Программы, соответствующие стандарту языка, могут компилироваться на самых различных компьютерах.

Для работы с графикой будет использоваться OpenGL.

В качестве среды разработки была выбрана Microsoft Visual Studio – линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом. 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).

Когда мы запускаем программу у нас первым делом должно отрисоваться меню.

Для реализации рисования будут созданы следующие функции:

void DrawField();

void DrawMario();

void MoveMario();

void Keyboard(int key, int x, int y);

void KeyboardUp(int key, int x, int y);

void timer(int k);

void display();

void Collision();

void DrawEnemy();

void MoveEnemy();

void timergo(int k);

void redraw(int k);

void renderBitmapString(float x, float y, float z, void \*font, char \*string);

GLuint LoadTexture(char \*filename);

GLuint LoadTextureP(char \*filename);

void DrawMenu();

Карта будет выглядеть как массив строк:

**Листинг 2.1** – Предположительная реализация карты

1. string Map1[H] = {
2. " "
3. " "
4. " "
5. " "
6. " "
7. " "
8. " WWWWWWWWWWWWW "
9. " WWWWWWWWWWWWWW "
10. " WWWWWWWWWWWWWWW "
11. " WWWWWWWWWWWWWWWW "
12. " WWWWWWWWWWWWWWWWW "
13. " WWWWWWWWWWWWWWWWWW "
14. " WWWWWWWWWWWWWWWWWWW WWWWWWWWWWWWW "
15. " WWWWWWWWWWWWWWWWWWWW "
16. " WWWWWWWWWWWWWWWWWWWWW "
17. " WWWWWWWWWWWWWWWWWWWWWW WWQWWWQW "
18. " WWWWWWWWWWWWWWWWWWWWWWW "
19. " WWWWWWWWWWWWWWWWWWWWWWWW "
20. " WWWWWWWWWWWWWWWWWWWWWWWWW E W q"
21. "WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW"
22. };

**Листинг 2.1** – Предположительная реализация управления

1. if (key == GLUT\_KEY\_LEFT)
2. {
3. Mario.Left = true;
4. }
5. else if (key == GLUT\_KEY\_RIGHT)
6. {
7. Mario.Right = true;
8. }
9. else if (key == GLUT\_KEY\_UP)
10. 1 {

Продолжение листинга 2.1

1. Mario.Up = true;
2. }

Текстуры были скачаны из интеренета.



**Рисунок 2.1** – Марио



**Рисунок 2.2** – Марио



**Рисунок 2.3** – Марио



**Рисунок 2.4** – Меню



**Рисунок 2.5** – Конец игры



**Рисунок 2.6** – Враг

# 3 Реализация программы

В программе, как и во всех программах OpenGL была описана главная функция main, из которой и происходил вызов основных функций.

**Листинг 3.1** – Структура Марио

1. struct mario
2. {
3. float x\_pos, y\_pos, dx, dy, posup, posdown, posleft, posright, tileup, tiledown, tileleft, tileright, tileony, tileonx;
4. bool Up, Down, Left, Right, OnGround, Life;
5. int lives, coins;
6. } Mario;

**Листинг 3.2** – Структура Врага

1. struct enemy
2. {
3. float x\_pos, y\_pos;
4. bool Left, Life;
5. } Enemy1, Enemy2, Enemy3;

Теперь рассмотрим большинство функций по отдельности:

Функция void display(); служит для:

* Отрисовки меню
* Вызова функций отрисовки поля, Марио, врагов.
* Вызова функций движения Марио, врагов.
* Обработку количества монет
* Отображение текущих уровня, монет, жизней
* Отрисовка экрана поражения

**Листинг 3.3** – Функция void display();

1. void display()
2. {
3. glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);
4. if (Mario.lives > 7)
5. {
7. glBindTexture(GL\_TEXTURE\_2D, texturemenu);
8. glEnable(GL\_TEXTURE\_2D);
9. glBegin(GL\_POLYGON);

Продолжение листинга 3.3

1. glTexCoord2d(0, 0); glVertex2f(0, 0);
2. glTexCoord2d(0, 1); glVertex2f(0, 600);
3. glTexCoord2d(1, 1); glVertex2f(800, 600);
4. glTexCoord2d(1, 0); glVertex2f(800, 0);
5. glEnd();
6. glDisable(GL\_TEXTURE\_2D);
7. DrawMenu();
8. if (menu == 1)
9. {
10. Mario.lives = 3;
11. }
12. }
13. else if (Mario.lives != 0)
14. {
15. DrawField();
16. DrawMario();
17. MoveMario();
18. DrawEnemy();
19. MoveEnemy();
20. if (Mario.coins == 10)
21. {
22. Mario.lives++;
23. Mario.coins = 0;
24. }
25. if (level != 6)
26. {
27. char buffer[8];
28. strcpy(buffer,"lives");
29. char str[3];
30. sprintf(str, " %d", Mario.lives);
31. strcat(buffer, str);
32. glColor3f(1.0, 1.0, 1.0);
33. renderBitmapString(100, 550, 0, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buffer);
34. strcpy(buffer,"coins");
35. sprintf(str, " %d", Mario.coins);
36. strcat(buffer, str);
37. glColor3f(1.0, 1.0, 1.0);
38. renderBitmapString(200, 550, 0, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buffer);

Продолжение листинга 3.3

1. strcpy(buffer,"level");
2. sprintf(str, " %d", level);
3. strcat(buffer, str);
4. glColor3f(1.0, 1.0, 1.0);
5. renderBitmapString(300, 550, 0, GLUT\_BITMAP\_TIMES\_ROMAN\_24, buffer);
6. }
7. }
8. else
9. {
10. glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);
11. glBindTexture(GL\_TEXTURE\_2D, texturego);
12. glEnable(GL\_TEXTURE\_2D);
13. glBegin(GL\_POLYGON);
14. glTexCoord2d(0, 0); glVertex2f(0, 0);
15. glTexCoord2d(0, 1); glVertex2f(0, 600);
16. glTexCoord2d(1, 1); glVertex2f(800, 600);
17. glTexCoord2d(1, 0); glVertex2f(800, 0);
18. glEnd();
19. glDisable(GL\_TEXTURE\_2D);
20. glutTimerFunc(3000, timergo, 7);
21. }
22. glutSwapBuffers();
23. glFlush();
24. }

Функция void DrawField(); служит для отображения карты.

**Листинг 3.4** – Часть функции void DrawField();

1. if (Mario.x\_pos >= WIN\_WIDTH / 2) //sdvig ekrana
2. {
3. offset\_x = Mario.x\_pos - WIN\_WIDTH / 2;
4. }
5. if ((Mario.x\_pos >= WIN\_WIDTH / 2) && (offset\_x >= 2400.0))
6. {
7. offset\_x = 2400.0;
8. }
9. for (int i = 0; i < H; i++)

Продолжение листинга 3.4

1. for (int j = 0; j < W; j++)
2. {
3. if (Map1[i][j] == 'W')
4. {
6. glBindTexture(GL\_TEXTURE\_2D, textureW);
7. glEnable(GL\_TEXTURE\_2D);
8. glBegin(GL\_POLYGON);
9. glTexCoord2f(0.3, 0.01); glVertex2f(j\*32.0 - offset\_x, abs(i - 19.0)\*32.0 - offset\_y); //Координаты квадрата
10. glTexCoord2f(0.3, 1.0); glVertex2f(j\*32.0 - offset\_x, abs(i - 19.0)\*32.0 + 32.0 - offset\_y);
11. glTexCoord2f(1.29, 1.0); glVertex2f(j\*32.0 + 32.0 - offset\_x, abs(i - 19.0)\*32.0 + 32.0 - offset\_y);
12. glTexCoord2f(1.29, 0.01); glVertex2f(j\*32. + 32.0 - offset\_x, abs(i - 19.0)\*32.0 - offset\_y);
13. glEnd();
14. glDisable(GL\_TEXTURE\_2D);
15. }
16. if (Map1[i][j] == 'w')
17. {
19. glBindTexture(GL\_TEXTURE\_2D, texturew);
20. glEnable(GL\_TEXTURE\_2D);
21. glBegin(GL\_POLYGON);
22. glTexCoord2f(0.3, 0.01); glVertex2f(j\*32.0 - offset\_x, abs(i - 19.0)\*32.0 - offset\_y); //Координаты квадрата
23. glTexCoord2f(0.3, 1.0); glVertex2f(j\*32.0 - offset\_x, abs(i - 19.0)\*32.0 + 32.0 - offset\_y);
24. glTexCoord2f(1.29, 1.0); glVertex2f(j\*32.0 + 32.0 - offset\_x, abs(i - 19.0)\*32.0 + 32.0 - offset\_y);
25. glTexCoord2f(1.29, 0.01); glVertex2f(j\*32. + 32.0 - offset\_x, abs(i - 19.0)\*32.0 - offset\_y);

Продолжение листинга 3.4

1. glEnd();
2. glDisable(GL\_TEXTURE\_2D);
3. }
4. if (Map1[i][j] == 'B')
5. {
7. glBindTexture(GL\_TEXTURE\_2D, textureB);
8. glEnable(GL\_TEXTURE\_2D);
9. glBegin(GL\_POLYGON);
10. glTexCoord2f(0.3, 0.01); glVertex2f(j\*32.0 - offset\_x, abs(i - 19.0)\*32.0 - offset\_y); //Координаты квадрата
11. glTexCoord2f(0.3, 1.0); glVertex2f(j\*32.0 - offset\_x, abs(i - 19.0)\*32.0 + 32.0 - offset\_y);
12. glTexCoord2f(1.29, 1.0); glVertex2f(j\*32.0 + 32.0 - offset\_x, abs(i - 19.0)\*32.0 + 32.0 - offset\_y);
13. glTexCoord2f(1.29, 0.01); glVertex2f(j\*32. + 32.0 - offset\_x, abs(i - 19.0)\*32.0 - offset\_y);
14. glEnd();
15. glDisable(GL\_TEXTURE\_2D);
16. }
17. if (Map1[i][j] == 'Q')
18. {
20. glBindTexture(GL\_TEXTURE\_2D, textureQ);
21. glEnable(GL\_TEXTURE\_2D);
22. glBegin(GL\_POLYGON);
23. glTexCoord2f(0.3, 0.01); glVertex2f(j\*32.0 - offset\_x, abs(i - 19.0)\*32.0 - offset\_y); //Координаты квадрата
24. glTexCoord2f(0.3, 1.0); glVertex2f(j\*32.0 - offset\_x, abs(i - 19.0)\*32.0 + 32.0 - offset\_y);
25. glTexCoord2f(1.29, 1.0); glVertex2f(j\*32.0 + 32.0 - offset\_x, abs(i - 19.0)\*32.0 + 32.0 - offset\_y);

Продолжение листинга 3.4

1. glTexCoord2f(1.29, 0.01); glVertex2f(j\*32. + 32.0 - offset\_x, abs(i - 19.0)\*32.0 - offset\_y);
2. glEnd();
3. glDisable(GL\_TEXTURE\_2D);
4. }
5. if (Map1[i][j] == 'C')
6. {
8. glBindTexture(GL\_TEXTURE\_2D, textureC);
9. glEnable(GL\_TEXTURE\_2D);
10. glBegin(GL\_POLYGON);
11. glTexCoord2f(0.3, 0.01); glVertex2f(j\*32.0 - offset\_x, abs(i - 19.0)\*32.0 - offset\_y); //Координаты квадрата
12. glTexCoord2f(0.3, 1.0); glVertex2f(j\*32.0 - offset\_x, abs(i - 19.0)\*32.0 + 32.0 - offset\_y);
13. glTexCoord2f(1.29, 1.0); glVertex2f(j\*32.0 + 32.0 - offset\_x, abs(i - 19.0)\*32.0 + 32.0 - offset\_y);
14. glTexCoord2f(1.29, 0.01); glVertex2f(j\*32. + 32.0 - offset\_x, abs(i - 19.0)\*32.0 - offset\_y);
15. glEnd();
16. glDisable(GL\_TEXTURE\_2D);
17. }
18. if (Map1[i][j] == 'q')
19. {
21. glBindTexture(GL\_TEXTURE\_2D, textureq);
22. glEnable(GL\_TEXTURE\_2D);
23. glBegin(GL\_POLYGON);
24. glTexCoord2f(0.3, 0.01); glVertex2f(j\*32.0 - offset\_x, abs(i - 19.0)\*32.0 - offset\_y); //Координаты квадрата
25. glTexCoord2f(0.3, 1.0); glVertex2f(j\*32.0 - offset\_x, abs(i - 19.0)\*32.0 + 32.0 - offset\_y);

Продолжение листинга 3.4

1. glTexCoord2f(1.29, 1.0); glVertex2f(j\*32.0 + 32.0 - offset\_x, abs(i - 19.0)\*32.0 + 32.0 - offset\_y);
2. glTexCoord2f(1.29, 0.01); glVertex2f(j\*32. + 32.0 - offset\_x, abs(i - 19.0)\*32.0 - offset\_y);
3. glEnd();
4. glDisable(GL\_TEXTURE\_2D);
5. }
6. }

Далее будут приведены строки кода, которые не были размещены в функциях потому что они используются локально и не требуют постоянного вызова.

**Листинг 3.5 – Инициализация некоторых полей структуры Mario**

1. Mario.lives = 8;
2. Mario.coins = 0;
3. Mario.Life = true;
4. Mario.OnGround = true;
5. Mario.x\_pos = 96.0;
6. Mario.y\_pos = 32.0;
7. Mario.dx = 2.0;
8. Mario.dy = 0.0;

**Листинг 3.6 – Инициализация некоторых полей структур Enemy**

1. Enemy1.x\_pos = 1152;
2. Enemy1.y\_pos = 32;
3. Enemy1.Left = true;
4. Enemy1.Life = true;
5. Enemy2.x\_pos = 1760;
6. Enemy2.y\_pos = 32;
7. Enemy2.Left = true;
8. Enemy2.Life = true;
9. Enemy3.x\_pos = 2336;
10. Enemy3.y\_pos = 32;
11. Enemy3.Left = true;
12. Enemy3.Life = true;

# 

# 4 Методика тестирования

Тестирование программного обеспечения - процесс исследования, испытания программного продукта, имеющий две различные цели:

* продемонстрировать разработчикам и заказчикам, что программа соответствует требованиям;
* выявить ситуации, в которых поведение программы является неправильным, нежелательным или не соответствующим спецификации.

Существующие на сегодня методы тестирования программного обеспечения не позволяют однозначно и полностью выявить все дефекты и установить корректность функционирования анализируемой программы, поэтому все существующие методы тестирования действуют в рамках формального процесса проверки исследуемого или разрабатываемого программного обеспечения.

Такой процесс формальной проверки, или верификации, может доказать, что дефекты отсутствуют с точки зрения используемого метода. (То есть нет никакой возможности точно установить или гарантировать отсутствие дефектов в программном продукте с учётом человеческого фактора, присутствующего на всех этапах жизненного цикла программного обеспечения.)

Существует множество подходов к решению задачи тестирования и верификации программного обеспечения, но эффективное тестирование сложных программных продуктов - это процесс в высшей степени творческий, не сводящийся к следованию строгим и чётким процедурам или созданию таковых.

Работоспособность основных компонентов приложения проверялась при помощи тестирования вида чёрный ящик. При тестировании чёрного ящика, тестировщик имеет доступ к программе только через те же интерфейсы, что и заказчик или пользователь, либо через внешние интерфейсы, позволяющие другому компьютеру либо другому процессу подключиться к системе для тестирования.

Тестирование было проведено методом черного ящика. Результаты тестирования представлены в таблице 4.1.

**Таблица 4.1**- Тестирование программы

|  |  |  |
| --- | --- | --- |
| **Действия** | **Ожидаемый результат** | **Итоговый результат** |
| Запуск программы | Отображение главного меню | Отобразилось успешно |
| Нажатие на кнопку “New Game” | Начало новой игры | Новая игра началась |
| При конце игры | Отрисовка картинки-сообщения о конце игры и перехода на главное меню спустя 2000 миллисекунд | Успешно |
| Закрытие программы | Выход из программы | Успешно |
| Уничтожение всех разрушаемых блоков. | Все разрушаемые блоки разрушены | Успешно |
| Уничтожение всех противников. | Все противники уничтожены | Успешно |
| Утрата всех очков жизней. | Все очки жизни потрачены | Успешно |
|  |  | Продолжение таблицы 5.1 |
| Зайти игровым персонажем на противника. | Счетчик жизней декрементировался. | Успешно |
| Подобрать бонус. | Увеличен счетчик | Успешно |
| Подобрать бонус когда счетчик = 9. | Счетчик обнулился, Добавилась жизнь | Успешно |

# 

# Заключение

В процессе проектирования и реализации курсовой работы была создана игровая программа «Марио» со всем стандартным функционалом игр подобного типа. Целью разработки программного продукта ставилась реализация двумерной игровой программы жанра «Аркада».

Были освоены навыки работы с функциями OpenGL.

Приложение разрабатывалось в среде Microsoft Visual Studio 2012.

Разработанный программный продукт может считаться игровой программой и удовлетворять требованиям пользователя.

Целевой аудиторией игровой программы предполагаются любители игрового жанра «Аркада»*,* которыйтребует напряжения внимания и быстрой реакции на происходящие в игре события.

Программа может быть модернизирована и дополнена. Возможны увеличение количества типов противников, улучшение анимации и игровой графики в целом и др.

В итоге программа реализована в соответствии всем требованиям, протестирована надлежащим образом, работает стабильно и может использоваться для решения поставленной задачи. Поставленную задачу считаю выполненной в полной мере.

В программе присутствует интуитивно понятный интерфейс и лёгкое управление, что сказывается на общем впечатлении от реализованной программы.

Целью данной курсовой работы, являлось углубление знаний и расширение навыков по разработке алгоритмов и их реализации на персональном компьютере, на мой взгляд, разработанная мной программа, вполне отвечает поставленным целям.

По итогам была оживлена игра конца 1990-х годов с использованием современных технологий и средств разработки.

# Список литературы

1. Интернет-портал «Википедия». – Режим доступа: https://ru.wikipedia.org/ . дата доступа 22.11.2017.

2. Конспект лекций по дисциплине «Языки программирования».

3. Интернет-портал «The spriters resource». – Режим доступа: https://www.spriters-resource.com/nes/supermariobros/. дата доступа 22.11.2017.

# Приложение

# Листинг программы

Листинг программы находится в папке «Курсовая работа» на диске.