# ВВЕДЕНИЕ

Игровая индустрия начала развиваться еще в прошлом веке. С появлением первых игр и первых интерактивных электронных устройств, предназначенных специально для игр, она стала одной из самых крупных отраслей программирования, в которой в наше время задействовано большое количество людей. На данный момент существуют десятки тысяч уникальных игр. Поэтому найти игру, которая будет соответствовать вашим предпочтениям не так уж и сложно, главное знать конкретно, что искать. В настоящее время создано множество игровых площадок, где можно найти интересующую игру, что значительно облегчает поиск.

Первые примитивные компьютерные и видео игры были разработаны в 1950-х и 1960-х годах. Они работали на таких платформах, как осциллографы, университетские мейнфреймы и компьютеры EDSAC. Самой первой компьютерной игрой стал симулятор ракеты, созданный в 1942 году Томасом Голдсмитом Младшим и Истл Рей Менном Позже, в 1952 году, появилась программа «OXO», имитирующая игру «крестики-нолики», созданная А.С. Дугласом как часть его докторской диссертации в Кембриджском Университете. Игра работала на большом университетском компьютере, известном как EDSAC (Electronic Delay Storage Automatic Calculator). В настоящее время, разработка игры – это многомиллионный процесс, в котором задействована целая команда разработчиков, сложные современные технологии и даже маркетинговые ходы.

Целью же данного курсового проекта будет разработка игры «Арканоид».

Для решения поставленной задачи, разработка программы будет происходить в среде Microsoft Visual Studio 2015 с использованием языка программирования C++ и спецификации OpenGL .

C++ широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также игр[6].

OpenGL (Open Graphics Library) — спецификация, определяющая платформонезависимый (независимый от языка программирования) программный интерфейс для написания приложений, использующих двумерную и трёхмерную компьютерную графику[5].

# АНАЛИЗ ЗАДАНИЯ И ПОСТАНОВКА ЗАДАЧ

В данном курсовом проекте будет осуществлена реализация игры «Арканоид» с помощью спецификации OpenGL.

В ходе выполнения курсовой работы необходимо реализовать игру «Арканоид» стандартного типа. Arkanoid — видеоигра для игровых автоматов, разработанная компанией Taito в 1986 году. Игра основана на играх серии Breakout фирмы Atari. С момента появления создан уже целый класс подобных игр, получивших тоже название – «Арканоид». Игрок контролирует небольшую платформу, которую можно передвигать горизонтально от одной стенки до другой, совершая из нее выстрелы. Попадание снарядом по кирпичу приводит к разрушению кирпича. Существует несколько типов кирпичей, отличающиеся друг от друга количеством ударов, необходимых для разрушения. Вначале игры у игрока имеется 3 жизни. После того как все кирпичи на данном уровне уничтожены игрок переходит на следующий уровень. Если шарик падает ниже уровня платформы, отнимается одна жизнь, если количество жизней становится равным нулю, игрок проигрывает.

Суть игры состоит в том, чтобы достичь максимального уровня и заработать как можно больше очков и вследствие этого поставить рекорд. Чтобы таблица рекордов была более разнообразной, то есть состояла из разных рекордов и имела меньшее количество повторяющихся рекордов, в игру была введена система бонусов, которые случайным образом распределяются среди блоков.

Игра может быть интересна людям разной возрастной категории, так как эта игра была популярна во время появления компьютеров и игровых автоматов

При создании проекта «Арканоид» ставятся задачи:

* Создание интуитивно понятного интерфейса.
* Создание алгоритмов отрисовки и логики игры.
* Сохранение результатов игры.

Интуитивно понятный интерфейс должен включать следующие пункты:

* Удобное игровое меню.
* Удобное управление.
* Простой и приятный дизайн.

Для того, чтобы заинтересовать игрока, также потребуется ввести некоторые особенности игры:

* Система уровней.
* Система сложностей.
* Система прогресса прохождения.

Главные задачи курсового проекта:

* Реализовать корректно работающий интерфейс игры.
* Реализовать обработку нажатий клавиатуры и мыши.
* Реализовать сохранение результатов игры в файл.
* Реализовать загрузку результатов игры из файла.
* Реализовать таблицу рекордов на 10 записей.

В игре будут два режима: «тренировка» и «обычный». Их отличия в том, что в режиме «тренировка», игрок может выбрать любой существующий уровень. Данный режим сделан для того, чтобы игрок мог потренироваться проходить какой-либо уровень, или если он не хочет каждый раз начинать игру с первого уровня, или если ему интересно посмотреть существующие уровни. В «обычном» режиме, игрок проходит уровень за уровнем, начиная с первого, пока не потратит все свои жизни. Когда игрок потратит три жизни, данные на старте, он сможет купить дополнительную за игровую валюту. При следующих проигрышах цена за покупку дополнительной жизни увеличивается в два раза от предыдущей стоимости.

Для удовлетворяющего вида игры, требуется разработать удобный и простой интерфейс, в котором всё будет расположено интуитивно понятно, чтобы игрок мог легко ориентироваться.

Для корректной работы проекта потребуется использовать все необходимые знания языка программирования C++, а также особенности OpenGL.

# ПРОЕКТИРОВАНИЕ ПРОГРАММЫ

При запуске игры откроется меню, состоящее из четырёх пунктов:

* Play.
* Shop.
* Records.
* Exit.

В правом верхнем углу будет отображаться количество игровой валюты, за которую, в случае проигрыша, можно купить дополнительную жизнь или поменять текстуру платформы.

При нажатии левой кнопки мыши по пункту «Play», пользователю откроется меню выбора режима игры. При нажатии на «Обычный» режим будет загружен первый уровень. При нажатии на «Тренировочный» режим будет отображено меню выбора уровня. На одной странице такого меню отображается 10 уровней. Для того чтобы посмотреть следующие, нужно нажать кнопку, которая указывает вправо. Чтобы вернуться, нужно нажать кнопку, указывающую влево. Таким же способом осуществляется выбор текстуры для платформы в пункте меню «Shop». Также открывать следующую страницу можно с помощью стрелочек на клавиатуре.

Изначально шарик прикреплен к платформе и двигается вместе с ней. Платформа управляется при помощи мыши и края платформы не могут уходить за область окна. Чтобы запустить шарик, нужно нажать левую кнопку мыши. Если шарик упадет ниже уровня платформы, он снова прикрепится к центру платформы.

При нажатии левой кнопки мыши по пункту «Shop», пользователю откроется меню выбора внешнего вида платформы, но пользователь может выбрать только купленные. Купить их можно за игровую валюту, которая зарабатывается в процессе игры. В магазине осуществлено удобное управление с помощью стрелок на клавиатуре и кнопок, расположенных в нижней части окна. Если данная платформа была куплена, то появляется кнопка выбора данной платформы, иначе появляется кнопка покупки и цена, расположенная под платформой.

При нажатии левой кнопки мыши по пункту «Records», пользователю откроется таблица рекордов на десять записей.

При нажатии левой кнопки мыши по пункту «Exit», игра закроется, предварительно сохранив все результаты игры.

При нажатии левой кнопкой мыши по любой кнопке, кнопка уменьшится в размере и сместиться к центру. Это сделано для визуализации нажатия кнопки.

В каждом подменю будет возможность вернуться к главному меню с помощью клавиши «Esc».

При успешном прохождении уровня в режиме «Тренировка», будет отображено количество набранных очков, кнопка выхода в меню и кнопка, которая загрузки следующего уровня.

При успешном прохождении уровня в режиме «Обычный», будет загружен следующий уровень.

Если количество жизней стало равным нулю, в обоих режимах будет выведено сообщение о проигрыше, количество набранных очков и кнопка выхода в главное меню, но в режиме «Обычный» будет появляться кнопка перехода на следующий уровень. Если следующего уровня не существует, то будет загружен последний. Если пользователь проиграет в режиме «Обычный», будет производиться проверка на рекорд, если набранное количество очков больше, чем какой-либо из ранее установленных рекордов, или же таблица рекордов содержит менее десяти записей, пользователю будет предложен ввод имени для записи рекорда.

В данном проекте будет использована динамическая память, для того чтобы не пришлось переписывать код при необходимости добавления новых уровней и картинок для платформ. Для того чтобы это сделать, необходимо создать новый текстовый файл в папке проекта, в котором будут записаны цифры от нуля до четырех. В одной строке должно быть введено не более 9 значений, иначе они будут игнорироваться. Ноль означает отсутствие блока, единица означает блок первого уровня и так до 4. Имя файла должно начинаться на «lvl\_», а далее следует номер уровня. Такой же принцип используется и в добавлении остальных элементов. Для корректного отображения номера уровня в режиме «Тренировка» в папку с проектом необходимо добавить картинку с номером уровня, иначе будет отображен прямоугольник белого цвета, но при нажатии на этот прямоугольник уровень загрузится без проблем.

# РЕАЛИЗАЦИЯ ПРОГРАММЫ

В ходе работы была реализована программа, которая представляет собой игру «Арканоид».

Для реализации программы будут использоваться следующие функции:

* void loadTextures() – функция загрузки текстур. В данной функции происходит загрузка всех текстур с использованием библиотеки Soil. Вызов данной функции осуществляется после отображения загрузочного экрана. В этой функции вызывается функция GLuint loadTexture(const char \*path) для загрузки разных текстур, которая принимает имя файла. Также в данной функции вызывается функция GLuint \*loadTextureArr(int amount, char \*nameSubStr), написанная для загрузки однотипных текстур, то есть текстур с почти идентичным именем. Эта функция принимает количество текстур и часть строки, которая есть у других файлов. Функция возвращает массив текстур. Данная функция написана с целью упростить добавление новых текстур и дать возможность добавлять новые элементы в игру не переписывая код;
* void drawTexture(int aX, int aY, int aW, int aH, GLuint aTextID) – функция рисования текстуры, которая принимает координаты верхнего левого угла отрисовки, ширину текстуры, длину текстуры и саму текстуру;
* void sortRecords() – функция, которая сортирует вектор структур, содержащих рекорд, по полю набранных очков;
* void readRecords() – функция, считывающая рекорды из файла и записывающая их в вектор структур;
* void saveRecords() – функция, которая сохраняет рекорды из вектора структур в файл. Перед записью в файл вызывается функция void sortRecords();
* void showRecords() – функция, отображающая таблицу рекордов;
* int countAvailableLvls(char \*subStr, char \*extension) – функция, которая считает количество существующих уровней. Функция принимает часть имени файла, в данном случае ей будет передаваться «lvl\_», расширение файла. Далее в цикле будет собираться целое имя файла. К конечному пути присоединяется значение счетчика цикла и расширение файла, в котором хранится информация об уровне. Затем с помощью цикла происходят попытки открыть файл с таким именем. При удачной попытке происходит инкрементирование переменной, отвечающей за количество доступных уровней. Цикл заканчивается тогда, когда файл с некоторым именем не будет открыт. Функция возвращает количество созданных уровней;

При реализации были созданы структуры Ball, Block, Bonus, Record, которые были описаны в листингах 3.1, 3.2, 3.3 и 3.4 соответственно.

Листинг 3.1 — Структура Ball

1. struct Ball {
2. int radius = 7;
3. int x;
4. int y;
5. int xSpeed = 2.0;
6. int ySpeed = -1.0;
7. bool move;
8. };

В структуре Ball описаны поля радиуса шара, координаты его центра, скорости по осям Ox и Oy, и булевая переменная, отвечающая за движение шара.

Листинг 3.2 — Структура Block

1. struct Block {
2. int xb;
3. int yb;
4. int durability;
5. GLuint tex;
6. } item;

В структуре Block описаны поля координат его левого верхнего угла, прочность и текстура, соответствующая прочности блока.

Модуль main является главным модулем. В нём реализована главная функция, взаимодействия всех модулей.

Модуль settings отвечает за все настройки, глобальные переменные, подключения других модулей и заголовков.

Модуль play\_and\_draw содержит функции рисования сцен и логики игры.

Функция randSteps генерирует последовательность ходов, принимает количество кругов, изменяет массив ходов. Описание функции распложено в листинге 3.3.

Листинг 3.3 — Функция randSteps

1. void randSteps(int n)
2. {
3. nSteps = n \* STEPSPERCIRCLE;
4. if (difficulty == DIF\_INSANE)
5. nSteps += nBalls;
6. srand(time(NULL));
7. for (int i = 0; i < nSteps; i++)
8. {
9. steps[i]=((rand()+rand()+rand()+rand()+rand())/5)%n;
10. }
11. }

Функция drawScene прорисовывает сцену на экране, вызывает другие функции прорисовки, в зависимости от значения переменной номера текущей сцены.

Функция drawScoreboard прорисовывает таблицу счёта. Вызывается функцией drawScene.

Функция setCamera изменяет параметры камеры, и координатных осей сцены в зависимости от размера текущего окна рендера.

Функция playGame прорисовывает игровой процесс. Вызывается функцией drawScene. Описание функции распложено в листинге 3.4.

Листинг 3.4 — Функция playGame

1. void playGame()
2. {
3. for (int i = 0; i < nBalls; i++)
4. window.draw(circles[i]);
5. if (!isGame && !highlight) {
6. drawAlignedText("Press \"S\" to start", ALIGN\_CENTER, 0, 0, 0.03);
7. }
8. else if (isGame) {
9. drawAlignedText("", ALIGN\_CENTER, 0, 0, 0.08, nSteps - iStep);
10. }
11. drawAlignedText("Time: ", ALIGN\_LEFT\_TOP, -48, -48, 0.06, fTime);
12. drawAlignedText("Lifes: ", ALIGN\_LEFT\_BOTTOM, -48, 48, 0.06, lifes);
13. }

Функция gameOver прорисовывает сцену, содержащую сообщение о поражении. Вызывается функцией drawScene.

Функция gameWin прорисовывает сцену, содержащую сообщение о победе. Вызывается функцией drawScene.

Функция drawMenu прорисовывает сцену, содержащую главное меню. Вызывается функцией drawScene.

Функция menuPlay прорисовывает сцену, содержащую подменю Play. Вызывается функцией drawScene.

Функция menuTrain прорисовывает сцену, содержащую подменю Train. Вызывается функцией drawScene.

Функция menuTrain прорисовывает сцену, содержащую подменю rain. Вызывается функцией drawScene.

Функция menuScore прорисовывает сцену, содержащую подменю Scoreboard. Вызывается функцией drawScene.

Функция drawLine рисует линию, принимает позиции начала и конца линии.

Функция createCircle рисует круг, принимает позицию круга и его радиус, возвращает объект круга. Описание функции распложено в листинге 3.5.

Листинг 3.5 — Функция createCircle

1. CircleShape createCircle(float x, float y, float radius) {
2. CircleShape circle(radius, 0.06);
3. circle.setOutlineThickness(0.5);
4. circle.setPosition(x, y);
5. return circle;
6. }

Функция insertNick прорисовывает сцену, содержащую интерфейс ввода игрового имени пользователя. Вызывается функцией drawScene.

Функция startLevel задаёт параметры уровня, принимает номер уровня. Описание функции распложено в листинге 3.6.

Листинг 3.6 — Функция startLevel

1. void startLevel(int n)
2. {
3. level = n;
4. if (n >= 0) difficulty = DIF\_EASY;
5. if (n >= 5) difficulty = DIF\_HARD;
6. if (n >= 10)difficulty = DIF\_INSANE;
7. int nB = n + 2;
8. while (nB > 6)
9. nB -= 5;
10. nBalls = nB;
11. posPoints(nB);
12. randSteps(nB);
13. highlight = 0;
14. iStep = 0;
15. isGame = 0;
16. timer = 0;
17. nSteps = 1;
18. lifes = LIFES;
19. page = GAME;
20. fTime = 0;
21. isLevel = 1;
22. }

Функция checkScore проверяет и сохраняет результаты пройденного уровня, принимает количество кругов в уровне, сложность, и показатель времени.

Функция initEmptyScoreboard заполняет таблицу счёта нулевыми значениями.

Функция drawAlignedText рисует текст, принимает строку, режим выравнивания, позицию, размер, число, которое нужно нарисовать вместе с текстом, возвращает объект полученного текста.

Модуль points содержит функции взаимодействия с кругами.

Функция colorPoints задаёт цвета кругам. Описание функции распложено в листинге 3.7.

Листинг 3.7 — Функция colorPoints

1. void colorPoints()
2. {
3. points[0].r = 28;
4. points[0].g = 232;
5. points[0].b = 254;
6. points[1].r = 252;
7. points[1].g = 147;
8. points[1].b = 92;
9. points[2].r = 254;
10. points[2].g = 249;
11. points[2].b = 127;
12. points[3].r = 236;
13. points[3].g = 74;
14. points[3].b = 188;
15. points[4].r = 218;
16. points[4].g = 67;
17. points[4].b = 86;
18. points[5].r = 106;
19. points[5].g = 253;
20. points[5].b = 156;
21. }

Функция initPoints инициализирует круги с заданными цветами и позициями, принимает количество кругов.

Функция hoverPoints обрабатывает наведение курсора мыши на круги.

Функция posPoints задаёт позиции кругам, принимает количество кругов.

Функция highlightPoints подсвечивает круги, необходима для отображения последовательности игроку. Описание функции распложено в листинге 3.8.

Листинг 3.8 — Функция highlightPoints

1. void highlightPoints()
2. {
3. if ((highlightDelay/(difficulty\*2))<=timer || timer<0)
4. {
5. mouse.x = 0;
6. mouse.y = 0;
7. }
8. else
9. {
10. mouse.x = points[steps[iStep]].x;
11. mouse.y = points[steps[iStep]].y;
12. }
13. }

Функция pressPoint обрабатывает нажатия кнопкой мыши по кругам.

Модуль save\_and\_load содержит функции сохранение и загрузки результатов игры.

Функция saveInFile сохраняет результаты игры в файл. Описание функции распложено в листинге 3.9.

Листинг 3.9 — Функция saveInFile

1. void saveInFile()
2. {
3. std::ofstream out(FILE);
4. if (out.is\_open())
5. {
6. out << progress << '\n';
7. for (int i = 0; i < 5; i++)
8. {
9. for (int j = 0; j < 3; j++)
10. {
11. for (int k = 0; k < 5; k++) {
12. out << scoreboard[i][j][k].a << " "
13. << scoreboard[i][j][k].b << " "
14. << scoreboard[i][j][k].c << " "
15. << scoreboard[i][j][k].time << '\n';
16. }
17. }
18. }
19. }
20. out.close();
21. }

Функция loadFromFile загружает сохранение игры из файла. Описание функции распложено в листинге 3.10.

Листинг 3.10 — Функция loadFromFile

1. void loadFromFile()
2. {
3. std::ifstream in(FILE);
4. if (in.is\_open())
5. {
6. in >> progress;
7. for (int i = 0; i < 5; i++)
8. {
9. for (int j = 0; j < 3; j++)
10. {
11. for (int k = 0; k < 5; k++) {
12. in >> scoreboard[i][j][k].a;
13. in >> scoreboard[i][j][k].b;
14. in >> scoreboard[i][j][k].c;
15. in >> scoreboard[i][j][k].time;
16. scoreboard[i][j][k].str[0] = scoreboard[i][j][k].a;
17. scoreboard[i][j][k].str[1] = scoreboard[i][j][k].b;
18. scoreboard[i][j][k].str[2] = scoreboard[i][j][k].c;
19. scoreboard[i][j][k].str[3] = '\n';
20. }
21. }
22. }
23. }
24. else
25. initEmptyScoreboard();
26. in.close();
27. }

Блок-схема модулей находится в Приложении А.

# ТЕСТИРОВАНИЕ ПРОГРАММЫ

При запуске игры открывается главное меню (рисунок 4.1) и начинает проигрываться музыка.



Рисунок 4.1 – Главное меню

При нажатии кнопки Play открывается меню выбора уровней (рисунок 4.2).



Рисунок 4.2 – Меню выбора уровней

При нажатии кнопки Train открывается такое же меню, но все уровни доступны.

При нажатии кнопки Scoreboard открывается таблица счёта (рисунок 4.3).

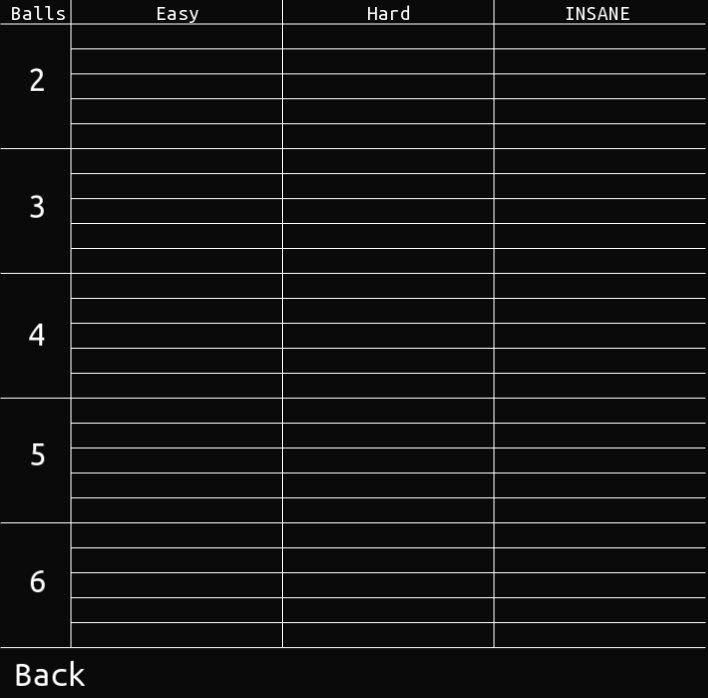


Рисунок 4.3 – Таблица счёта

При проигрыше игроку выводиться сообщение (рисунок 4.4).



Рисунок 4.4 – Поражение

Во время игры пользователю показываются количество жизней, количество оставшихся жизней, количество оставшихся ходов, время, затраченное на прохождение, при наведении курсора мыши на круг, цвет круга проявляется, ширина обводки уменьшается, а радиус фигуры увеличивается (рисунок 4.5).

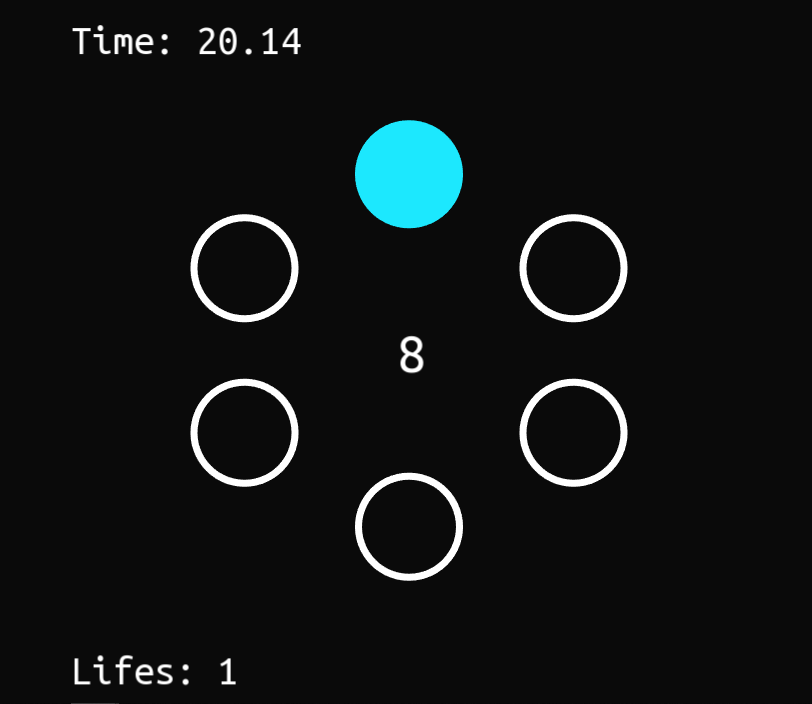


Рисунок 4.5 – Игровой процесс

При удачном прохождении уровня игроку выводиться сообщение с результатом, в виде затраченного времени на уровень (рисунок 4.6).



Рисунок 4.6 – Победа

При необходимости ввести игровое имя игрока, для сохранения результата прохождения уровня открывается меню ввода (рисунок 4.7).

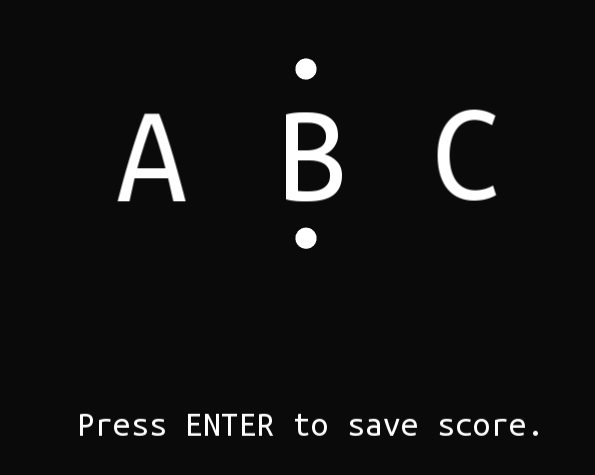


Рисунок 4.7 – Ввод игрового имени

Результат с данными показателями виден в таблице счёта (рисунок 4.8)

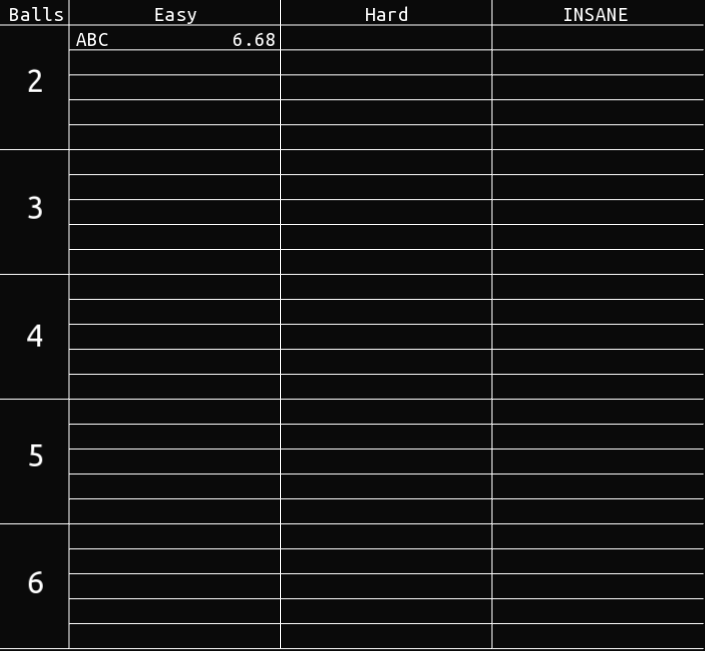


Рисунок 4.8 – Таблица счёта с результатом

# ЗАКЛЮЧЕНИЕ

В данном курсовом проекте была поставлена цель: реализовать игру «In Search of Diploma», с помощью спецификации OpenGL и языка программирования C++.

При анализе данных, были поставлены корректные задачи, что помогло в проектировании игрового приложения. Так же были рассмотрены все нюансы и особенности игры и определена полезность данной игровой программы.

В результате проектирования были предусмотрены необходимые функции, модули для работы приложения. А также разработаны прототипы графического интерфейса.

В итоге были реализованы такие функции, как отрисовка сцены, обработка нажатий кнопок мыши и клавиатуры, генерация случайной последовательности, изменение параметров камеры, сохранение результатов игры, загрузка результатов игры и другие необходимые функции.

Был реализован интуитивно понятный интерфейс, музыкальное сопровождение, воспроизведение звуковых эффектов.

Одной из технических особенностей игры является её адаптивность к любому дисплею, так как в игре отсутствуют какие-либо растровые изображения.

В ходе тестирования было установлено, что программа не имеет каких-либо слабых мест, также было установлено, что:

* Интерфейс программы интуитивно понятен и удобен.
* Меню работает корректно.
* Логика программы отлажена и работает корректно.
* Результат игры сохраняется и загружается.
* Дизайн выглядит минималистичным и вполне приемлемым.

Это позволяет с уверенностью сказать, что игра готова для использования любым пользователем.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Фридман, А. С/С++. Алгоритмы и приемы программирования / А. Фридман. – М.: Бином, 2007. – 560 с.
2. Керниган Б., Ритчи Д. Язык программирования С: 2-е издание. - М.: Издательский дом «Вильямс», 2007. - 304 с.
3. С.П. Чеботарёв УМК «Конструирование программ и языки программирования». Новополоцк: ПГУ, 2007 часть 1.
4. С.П. Чеботарёв УМК «Конструирование программ и языки программирования». Новополоцк: ПГУ, 2008.
5. Конспект лекций по дисциплине «Языки программирования».
6. Культин, Н. С/С++ в задачах и примерах / Н. Культин. – СПб. : БХВ-Петербург, 2007. – 288 c.
7. Эккель Б., Эллисон Ч. Философия С++. Практическое программирование. – СПб.: Питер, 2004. – 608 с.
8. Лаффе Р. Объектно-ориентированное программирование в С++. Классика Computer Science. – СПб.: Питер, 2006 – 928 c.
9. Шиманович Е. Л. C/C++ в примерах и задачах. - Минск: Новое знание, 2004, - 528с.
10. Гилберт Стивен, Макартни Билл. Самоучитель Visual C++ 6 в примерах. - К.: ООО "ТИД ДС", 2003. - 496с.

# ПРИЛОЖЕНИЕ А

(обязательное)

**Модульная схема**

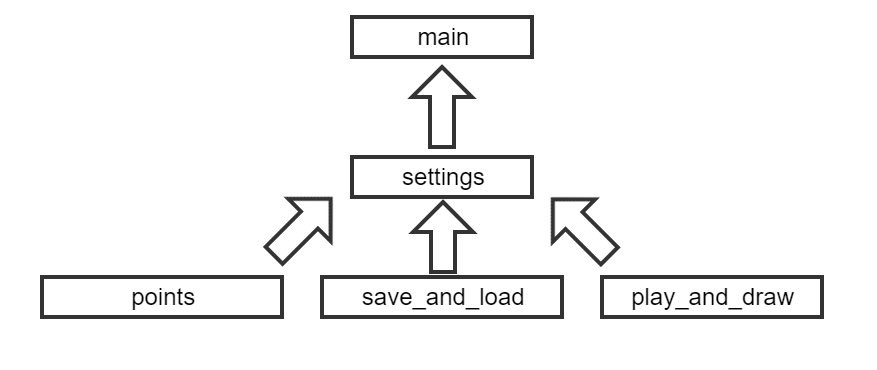


Рисунок А.1 – Блок-схема модулей

# ПРИЛОЖЕНИЕ Б

**Прототип графического интерфейса**

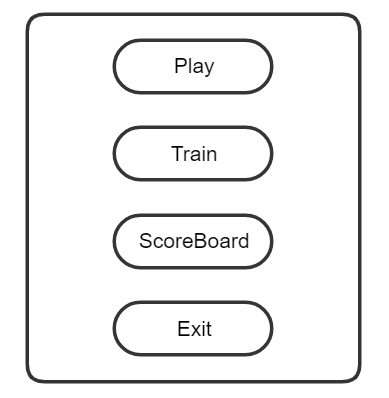


Рисунок Б.1 – Прототип интерфейса сцены главного меню

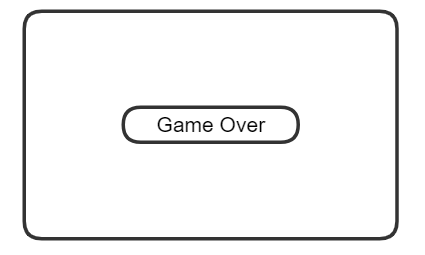


Рисунок Б.2 – Прототип интерфейса сцены поражения

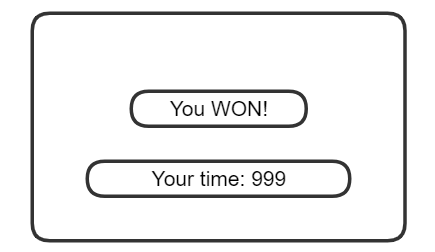


Рисунок Б.3 – Прототип интерфейса сцены победы

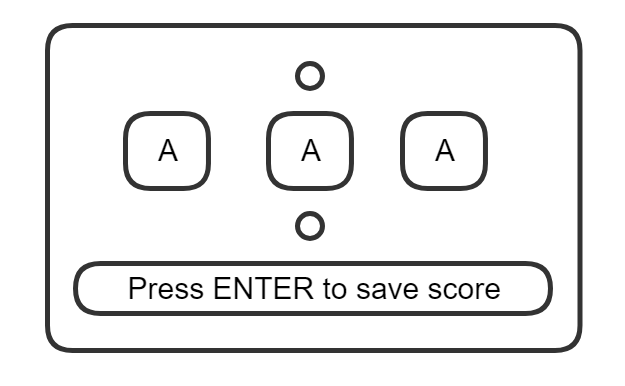


Рисунок Б.4 – Прототип интерфейса сцены ввода игрового имени

# ПРИЛОЖЕНИЕ В

(на диске)

**Листинг программы**