УО «Полоцкий государственный университет»

Кафедра технологий программирования

Пояснительная записка

К курсовому проекту

по дисциплине «Языки программирования»

Тема: «Арканоид»

Исполнитель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дёкин А.Ю.

студент 2 курса группы 18-ИТ-1

Руководитель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Пяткин Д.В.

Оценка защиты курсовой работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты: « \_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

Члены комиссии: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Новополоцк, 2019

**СОДЕРЖАНИЕ**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

3

ДАЮ.1810321.ПЗ

Разраб.

Дёкин А.Ю.

Провер.

Пяткин Д.В.

Реценз.

Н. Контр.

Утверд.

*Голу*

Разработка игровой программы «Арканоид»

для ГУО «Боровухской средней школы № 15» г. Новополоцка рЛит.

Листов

25

Учреждение образования «Полоцкий государственный университет» гр.18-ИТ-1

[ВВЕДЕНИЕ 3](#_Toc26389480)

[1 АНАЛИЗ ЗАДАНИЯ И ПОСТАНОВКА ЗАДАЧ 4](#_Toc26389481)

[2 ПРОЕКТИРОВАНИЕ ПРОГРАММЫ 6](#_Toc26389482)

[3 3 РЕАЛИЗАЦИЯ ПРОГРАММЫ 9](#_Toc26389483)

[4 Тестирование программы 20](#_Toc26389484)

[Заключение 24](#_Toc26389485)

[Список использованных источников 25](#_Toc26389486)

ПРИЛОЖЕНИЯ 27

# 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.

При запуске проекта будет запущена функция «main», в которой будет определен размер окна opengl c помощью функции glutInitWindowSize(640, 480). Установлена позиция окна на рабочем столе устройства с помощью функции glutInitWindowPosition(50, 50). Будет создано окно и установлен его заголовок с помощью функции glutCreateWindow(«Arcanoid»).

Для анимации будет использоваться функция glutTimerFunc. Она устанавливает функцию в вашей программе, которая будет отвечать за перерисовку окна. Это нужно для того, что когда системе потребуется перерисовать ваше окно (например, для анимации предметов), то она вызовет соответствующую функцию в вашей программе, которая ответственна за перерисовку содержимого окна.

Функция glutPassiveMotionFunc(MouseMove) получает координаты курсора мыши. Обработчик активен, когда положение курсора изменено, но клавиша мыши не нажата.

Функция glutMouseFunc(MouseMoveClick), которая будет отвечать за обработку событий, создаваемых щелчками клавиш мыши.

Функция glutKeyboardFunc(keyBoardFunc), используется, чтобы сообщить системе, что мы хотим обработать «нормальные» нажатия клавиш. Под «нормальными» нажатиями клавиш, мы имеем в виду буквы, цифры, все, что имеет ASCII код.

Функция glutSpecialFunc(specialKeysFunc), используется, чтобы сообщить системе, что мы хотим обработать нажатие «специальных» клавиш, в данном случае стрелочек. Стрелочками осуществляется управление перелистыванием страниц при выборе уровня и выборе платформы.

Функция glutMainLoop(). Эта функции никогда не возвращается, наша программа по существу входит в бесконечный цикл. В каждом приложении OpenGL функция main() должна заканчиваться вызовом glutMainLoop().

Данный проект разбит из главного модуля программы и модуля, реализующего нужный функционал для создания кнопок.

# 3 РЕАЛИЗАЦИЯ ПРОГРАММЫ

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

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

В модуле main описаны следующие основные 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. Описание данной функции находится в листинге 3.1;

**Листинг 3.1 –** Функция GLuint loadTexture(const char \*apFileName)

1. GLuint loadTexture(const char \*apFileName) {
2. GLuint texture;
3. texture = SOIL\_load\_OGL\_texture(apFileName,
4. SOIL\_LOAD\_RGBA,
5. SOIL\_CREATE\_NEW\_ID,
6. SOIL\_FLAG\_NTSC\_SAFE\_RGB | SOIL\_FLAG\_MULTIPLY\_ALPHA | SOIL\_FLAG\_INVERT\_Y);
7. return texture;
8. }

* 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. Описание данной функции находится в листинге 3.2;

**Листинг 3.2** – Функция void drawTexture(int aX, int aY, int aW, int aH, GLuint aTextID)

1. void drawTexture(int aX, int aY, int aW, int aH, GLuint aTextID) {
2. glColor3f(1.0, 1.0, 1.0);
3. glEnable(GL\_TEXTURE\_2D);
4. glBindTexture(GL\_TEXTURE\_2D, aTextID)

Продолжение листинга 3.2

1. glBlendFunc(GL\_ONE, GL\_ONE\_MINUS\_SRC\_ALPHA);
2. glBegin(GL\_QUADS);
3. glTexCoord2i(1, 0); glVertex2i(aX + aW, aY + aH);
4. glTexCoord2i(0, 0); glVertex2i(aX, aY + aH);
5. glTexCoord2i(0, 1); glVertex2i(aX, aY);
6. glTexCoord2i(1, 1); glVertex2i(aX + aW, aY);
7. glEnd();
8. glDisable(GL\_TEXTURE\_2D);
9. }

* void sortRecords() – функция, которая сортирует вектор структур, содержащих рекорд, по полю набранных очков. Описпание данной функции находится в листинге 3.3;

**Листинг 3.4** – Функция void sortRecords()

1. void sortRecords() {
2. for (int i = 0; i < recordsRows; i++) {
3. for (int j = i; j < recordsRows; j++) {
4. if (records[i].score < records[j].score)
5. std::swap(records[i], records[j]);
6. }
7. }
8. }

* 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 renderBM(float x, float y, void \*font, char \*str) – функция, выводящая на экран строку, указатель на которую передается в качестве параметра. Также в функцию передается позиция выводимого текста и шрифт, которым будет напечатана строка;
* void printText(int xPos, int yPos, int info) – функция, выводящая число на экран, в этой функции происходят преобразования из числа в строку, а затем вызывается функция renderBM;
* void menu() – функция, отображающая главное меню игры, отслеживающая нажатие кнопок игрового меню и передающая информацию об отображаемой информации в функцию, отвечающую за отображение игрового процесса;



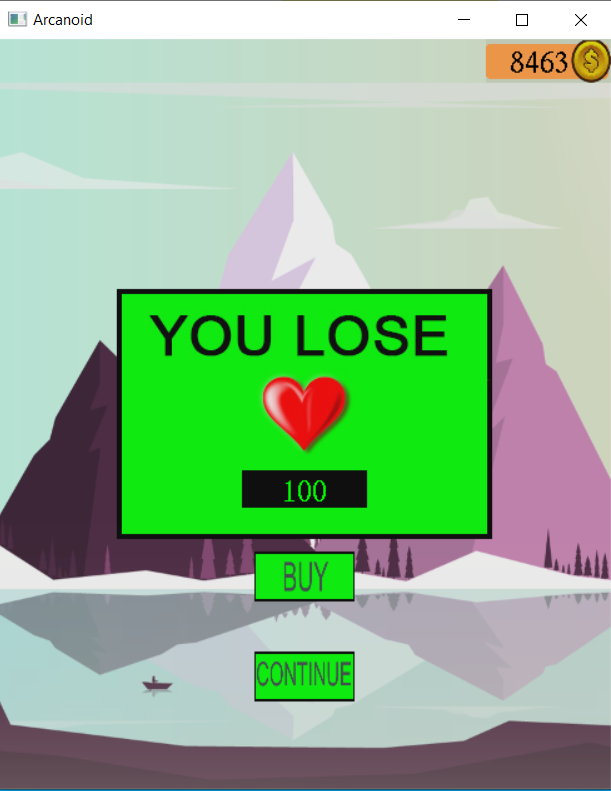
**Рисунок 3.1** – Главное меню игры

* 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;
* void showRecords() – функция, отображающая таблицу рекордов;



**Рисунок 3.2** – Таблица рекордов

* void display() – функция, отвечающая за отображение всех элементов игры, в зависимости от игрового процесса, который меняется при определенных действий;
* void game() – функция, в которой реализован основной игровой процесс, работающий до тех пор, пока количество жизней больше нуля. Если количество жизней стало равным нулю, в данной функции происходят процессы добавления игровой валюты, проверка на рекорд и отчистка игрового поля;
* bool ballLose(int x, int y, int index) – функция, проверяющая уровень мяча и возвращающая истину, если шарик находится ниже уровня платформы, иначе возвращает ложь. Функция принимает координаты шарика и индекс шарика, так как их может быть несколько;
* bool toOfferExtraLife() – функция, в которой предлагается возможность покупки дополнительной жизни в случае проигрыша. Возвращает истину, если игрок купил дополнительную жизнь. В этом случае игра продолжается. Если игрок не купил дополнительную жизнь, игра завершается;



**Рисунок 3.3** – Покупка дополнительной жизни

* void ballMotion() – функция, отвечающая за движение шарика или шариков, если их несколько. В данной функции происходит изменение траектории полета шарика при в результате соприкосновения его с какими-либо объектами;
* void initBlocks(char \*lvlPath) – функция, заполняющая вектор блоков, считывая данные об уровне из текстового файла. Функция принимает имя файла, содержащего информацию об уровне;
* void collision() – функция, определяющая столкновение шарика и блока, а также меняющая направление полета шарика, в зависимости от типа столкновения. При столкновении шарика с блоком, происходит уменьшение прочности блока или его удаление, если прочность стала равна нулю;
* void initFunctions() – функция, в которой вызываются функции инициализации координат всех кнопок, а также происходит подсчёт количества доступных уровней, текстур и проданных текстур для платформ;
* void selectLvl() – функция, в которой пользователь выбирает уровень для прохождения. В меню выбора уровня отображается одна страница, если количество уровней больше десяти, добавляется новая страница;



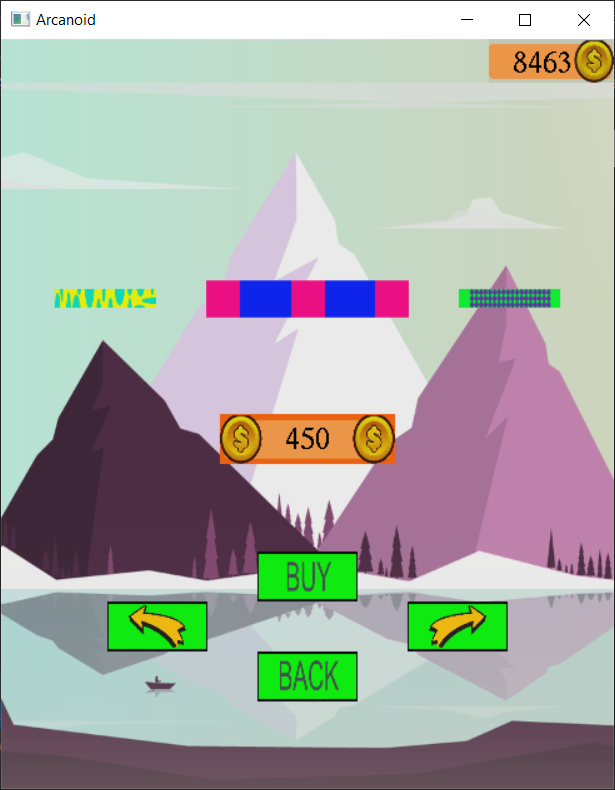
**Рисунок 3.4** –Меню выбора уровня

* void initBonuses() – функция, инициализирующая позиции бонусов и их тип. Позиция присваивается случайно и зависит от количества блоков. Количество бонусов также задается случайно, но это значение может всегда приблизительно равно десяти процентам от общего количества блоков;
* void showCoins() – функция, которая отображает количество монет, заработанных в игре;
* void addRecord() – функция, которая добавляет рекорд в таблицу рекордов. Функция всегда добавляет рекорд, если их количество меньше десяти, так как по условию задания нужно было реализовать именно такую таблицу рекордов. Если количество рекордов равно десяти, то происходи проверка с остальными рекордами. Если новый рекорд больше какого-либо другого, уже занесенного в таблицу, то удаляется последний рекорд и добавляется новый;
* void enterRecordName(unsigned char key) – функция, осуществляющая ввод имени рекорда. Функция получает символ, проверяет его с клавишей «Enter», если это не она, то символ добавляется в строку, если это клавиша «Enter», то ввод имени завершается;



**Рисунок 3.5** – Ввод имени рекорда

* void choosePlate() – функция, осуществляющая пункт меню «Shop», в котором происходит покупка или выбор текстуры для платформы. Для отображения следующей текстуры ракетки используется функция void drawPlateSkins(int curPlatePage, int maxPage), принимающая номер текущей страницы магазина и номер максимальной страницы. В этой функции происходит изменение номера текущей страницы в зависимости от нажатой кнопки или стрелочек на клавиатуре;



**Рисунок 3.6** – Меню выбора текстуры платформы

* void selectGameMode() – функция, отвечающая за выбор режима игры;



**Рисунок 3.**7 – Меню выбора режима игры

При реализации курсового проекта были созданы структуры Ball, Block, Bonus, Record, которые 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. };

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

**Листинг 3.3** — Структура Bonus

1. struct Bonus {
2. int x;
3. int y;
4. int bonusType;
5. GLuint texture;
6. bool move;
7. };

В структуре Bonus описаны поля координат его левого верхнего угла, тип бонуса, текстура бонуса и булевая переменная, отвечающая за движение бонуса.

**Листинг 3.4** — Структура Record

1. struct Record {
2. int priority;
3. char name[20];
4. int score;
5. };

В структуре Record описаны поля приоритета рекорда, то есть его номер в списке, имя рекорда и количество очков, набранное игроком с эти именем.

В результате выполнения курсового проекта был создан модуль Button для реализации кнопок. В нем была создана структура Button, которая описана в листинге 3.5.

**Листинг 3.5** — Структура Button

1. struct Button
2. {
3. private:
4. int cx;
5. int cy;
6. int width;
7. int height;
8. int curW;
9. int curH;
10. int curCX;
11. int curCY;
12. GLuint text;
13. public:
14. void Button::ShowBtn();
15. void Button::setButtonPosition(int cx, int cy, int
16. width, int height, GLuint text);
17. bool Button::isClicked(float xMousePos, float
18. yMousePos, int btnState);
19. };

В структуре описаны поля координат верхнего левого угла, ширина и высота кнопки, текстуры, а также дубликаты полей координат и размеров кнопки, для создания эффекта нажатия.

В структуре также описаны следующие функции:

* void Button::setButtonPosition(int cx, int cy, int width, int height, GLuint text) – функция, с помощью которой задаются положение и размеры кнопки. Описание функции находится в листинге 3.6;

**Листинг 3.6** — Функция setButtonPosition(int cx, int cy, int width, int height,

GLuint text)

1. void Button::setButtonPosition(int cx, int cy, int width,
2. int height, GLuint text) {
3. this->cx = cx;
4. this->cy = cy;
5. this->width = width;
6. this->height = height;
7. this->text = text;
8. curH = height;
9. curW = width;
10. curCX = cx;
11. curCY = cy;
12. }

* void Button::ShowBtn() – функция, которая отображает кнопку. Описание функции находится в листинге 3.7;

**Листинг 3.7** — Функция ShowBtn()

1. void Button::ShowBtn() {
2. glColor3f(1.0, 1.0, 1.0);
3. glEnable(GL\_TEXTURE\_2D);
4. glBindTexture(GL\_TEXTURE\_2D, text);
5. glBegin(GL\_QUADS);
6. glTexCoord2i(1, 0); glVertex2i(curCX + curW, curCY +
7. curH);
8. glTexCoord2i(0, 0); glVertex2i(curCX, curCY + curH);
9. glTexCoord2i(0, 1); glVertex2i(curCX, curCY);
10. glTexCoord2i(1, 1); glVertex2i(curCX + curW, curCY);
11. glEnd();
12. glDisable(GL\_TEXTURE\_2D);
13. }

* bool Button::isClicked(float xMousePos, float yMousePos, int btnState) – функция,
* которая проверяет нажатие на кнопку, а также уменьшает размеры кнопки при ее нажатии для видимого эффекта нажатия кнопки. Функция срабатывает при поднятии левой клавиши мыши. Если отпустить ее не в области кнопки, то ничего не произойдет. Если зажать одну кнопку, а отпустить ее на другой, сработает именно та, на которой произошел отпуск клавиши мыши. Описание функции находится в листинге 3.8;

**Листинг 3.8** — Функция isClicked(float xMousePos, float yMousePos, int btnState)

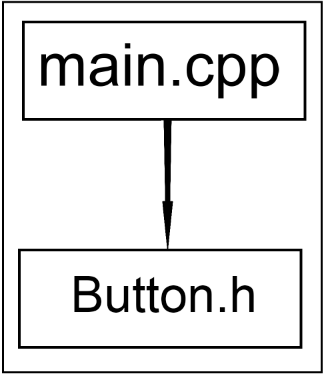
1. bool Button::isClicked(float xMousePos, float
2. yMousePos, int btnState) {
3. if (xMousePos >= cx && xMousePos <= cx + width &&

4:yMousePos >= cy && yMousePos <= cy + height) {

1. if (btnState == GLUT\_DOWN) {
2. curW -= width / 25;
3. curH -= height / 15;
4. curCX += width / 50;
5. curCY += height / 30;
6. }
7. if (btnState == GLUT\_UP) {
8. curW = width;
9. curH = height;
10. curCX = cx;
11. curCY = cy;

Продолжение листинга 3.9

1. return true;
2. }
3. }
4. else if (btnState == GLUT\_UP && curH != height &&
5. curW != width) {
6. curW = width;
7. curH = height;
8. curCX = cx;
9. }
10. return false;
11. }



**Рисунок 3.8** – Схема подключения модулей

# 4 Тестирование программы

Тестирование программы курсового проекта проводилось методом чёрного ящика. Сути данного метода заключается в том, что программное обеспечение рассматривается как «чёрный ящик», то есть анализируются только основные аспекты системы. Преимущества данного метода:

– эффективен для большого сегмента кода;

– более быстрое создание теста;

– простота восприятия.

Основным недостатком данного подхода является то, что в действительности выполняется выборочное число сценариев, что ограничивает тестируемую область. Также, в данном методе отсутствует чёткая спецификация, что затрудняет разработку тестовых сценариев.

Тестирование программы и результаты представлены в таблице 4.1

**Таблица 4.1** – Тестирование программы

|  |  |  |  |
| --- | --- | --- | --- |
| **Действие** | **Ожидаемый результат** | **Полученный результат** | **Метка** |
| Запуск меню | Перед пользователем открывается окно программы, содержащее меню, пункты которого позволяют. | При запуске программы высвечивается окно с таблицей и главным меню, содержащим пункты, позволяющие пользователю осуществить ожидаемые действия. | Выполнено |
| Нажатие на «play» | Запускается функция selectGameMode() и все остальные функции находящиеся внутри. Мы увидим предполагаемый интерфейс. | Мы увидели предполагаемый интерфейс. | Выполнено |
| Выбор режима «Обычный» | Запуск игрового процесса с первого уровня. | Открылся первый уровень. | Выполнено |
| Выбор режима «Тренировка» | Открывается меню выбора уровня. | Открытое меню выбора уровней. | Выполнено |
| Нажатие на левую клавишу мыши в пределах окна | Ball.move присваивается значение true и шарик начинает движение. | Шарик начинает движение. | Выполнено |

Продолжение таблицы 4.1

|  |  |  |  |
| --- | --- | --- | --- |
| **Действие** | **Ожидаемый результат** | **Полученный результат** | **Метка** |
| Нажать на кнопку «Shop» | Откроется магазин, где можно купить или выбрать текстуру для платформы. | Отобразилось меню выбора и покупки текстуры для платформы. | Выполнено |
| Столкновение с левой стенкой | Шарик оттолкнется от стены и полетит дальше. | Шарик оттолкнулся и продолжил движение. | Выполнено |
| Столкновение с правой стенкой | Шарик оттолкнется от стены и полетит дальше. | Шарик оттолкнулся и продолжил движение | Выполнено |
| Столкновение с верхней стенкой | Шарик оттолкнется от стены и полетит дальше. | Шарик оттолкнулся и продолжил движение. | Выполнено |
| Пересечение нижней области игрового поля | Жизнь уменьшится на одну единицу и шарик переместится в начальную позицию на платформу. | Жизнь уменьшилась на одну единицу и шарик переместился в начальную позицию на платформу. | Выполнено |
| Столкновение с желтым кирпичиком | Шарик оттолкнется от кирпичика и полетит дальше, столкновение проведет к разрушению кирпичика и увеличению «score». | Шарик оттолкнулся от кирпичика и полетел дальше, столкновение провело к разрушению кирпичика и увеличению «score». | Выполнено |
| Столкновение с оранжевым кирпичиком | Шарик оттолкнется от кирпичика и полетит дальше, столкновение проведет к уменьшению жизни кирпичика на одну единицу и изменению цвета на желтый. | Шарик оттолкнулся от кирпичика и полетел дальше, к уменьшению жизни кирпичика на одну единицу, изменению цвета на желтый и увеличению «score». | Выполнено |
| Столкновение с красным кирпичиком | Шарик оттолкнется от кирпичика и полетит дальше, столкновение проведет к уменьшению жизни кирпичика на одну единицу и изменению цвета на оранжевый и увеличению «score». | Шарик оттолкнулся от кирпичика и полетел дальше столкновение провело к изменению цвета кирпичика на оранжевый и увеличению «score». | Выполнено |

Продолжение таблицы 4.1

|  |  |  |  |
| --- | --- | --- | --- |
| **Действие** | **Ожидаемый результат** | **Полученный результат** | **Метка** |
| Столкновение с бордовым кирпичиком | Шарик оттолкнется от кирпичика и полетит дальше, столкновение проведет к уменьшению жизни кирпичика на одну единицу и изменению цвета на красный и увеличению «score». | Шарик оттолкнулся от кирпичика и полетел дальше столкновение провело к изменению цвета кирпичика на красный и увеличению «score». | Выполнено |
| Столкновение с кирпичиком содержащий бонус «увеличение платформы» | Шарик оттолкнется от кирпичика и полетит дальше, столкновение проведет к разрушению кирпичика и увеличению «score» на десять единиц и выпадению бонусного шарика фиолетового цвета. | Шарик оттолкнулся от кирпичика и полетел дальше, столкновение провело к разрушению кирпичика и увеличению «score» на десять единиц и выпадение бонусного шарика фиолетового цвета. | Выполнено |
| Столкновение с кирпичиком содержащий бонус «дополнительная жизнь» | Шарик оттолкнется от кирпичика и полетит дальше, столкновение проведет к разрушению кирпичика, увеличению «score» и выпадению бонуса. | Шарик оттолкнулся от кирпичика и полетел дальше, столкновение провело к разрушению кирпичика и увеличению «score» и выпадение бонуса. | Выполнено |
| Столкновение с кирпичиком, в котором содержится бонус «дополнительный шарик» | Шарик оттолкнется от кирпичика и полетит дальше, столкновение проведет к разрушению кирпичика, увеличению «score» , выпадению бонуса. | Шарик оттолкнулся от кирпичика и полетел дальше, столкновение провело к разрушению кирпичика и увеличению «score» и выпадение бонуса. | Выполнено |

Окончание таблицы 4.1

|  |  |  |  |
| --- | --- | --- | --- |
| Действие | Ожидаемый результат | Полученный результат | Метка |
| Разрушить все кирпичики | Осуществится переходит на следующий уровень. | Загрузился следующий уровень. | Выполнено |
| Закончились все жизни в режиме игры «Обычный» | Появление окна, в котором предлагается купить дополнительную жизнь. | Появление окна, где можно купить дополнительную жизнь | Выполнено |
| Нажать на клавишу «Esc» | Переход в главное меню. | Переход в главное меню. | Выполнено |
| Нажать на клавишу «records»; | Откроются записи рекордов. | Открылись записи рекордов. | Выполнено |
| Нажать на клавишу «exit» | Осуществится выход из игры. | Осуществился выход из игры. | Выполнено |

Согласно результатам тестирования программы, проверяемые действия программы оправдали все ожидаемые результаты. Это означает, что основные аспекты программы работают правильно как визуально, так и в плане реализации. Тем не менее, проверка методом «Черного ящика» не гарантирует полного отсутствия изъянов в коде и достижения максимальной оптимизации работы программы.

# Заключение

В результате проделанной курсовой работы были закреплены и усвоены основные знания, полученные при изучении предмета «Языки программирования». Согласно полученному заданию данного курсового проекта, была разработана программа – «Арканоид». В процессе оформления данной пояснительной записки проводился подробный анализ исходного кода и алгоритмов программы, тестирование и отладка работы программы. Результаты проведенного тестирования отображают корректность функционирования программы.

В процессе выполнения данного курсового проекта получен теоретически и практический опыт разработки программного обеспечения с использованием определённых средств разработки и программирования.

# Список использованных источников

1. Arkanoid – Википедия [Электронный ресурс]. Режим доступа: [https://ru.wikipedia.org/wiki/Arkanoid.](https://ru.wikipedia.org/wiki/Arkanoid) – Дата доступа: 17.09.2019.
2. Павловская Т.А. С++ Программирование на языке высокого уровня. Учебник для вузов/Павловская Т.А – Москва, 2003 – 450с.
3. С.П Чеботарев УМК «Конструирование программ и языки программирования». Новополоцк: ПГУ, 2007 часть 1.
4. Шиманович Е.Л. С/С++ в примерах и задачах. – Минск: новое знание, 2004 – 528с.
5. Культин, Н. С/С++ в задачах и примерах / Н. Культин. – СПБ: БХВ – Петербург, 2007. – 288 с.
6. Учебные материалы [Электронный ресурс]. – Режим доступа: [https://works.doklad.ru/view/HK4w3IjbEzk.html](https://works.doklad.ru/view/HK4w3IjbEzk.html%20/). – Дата доступа: 27.09.2019.

**ПРИЛОЖЕНИЯ** **ПРИЛОЖЕНИЕ А**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

27

ДАЮ.1810321.ПЗ

Разраб.

Дёкин А.Ю.

Провер.

Пяткин Д.В.

Реценз.

Н. Контр.

Утверд.

*Голу*

Разработка игровой программы «Арканоид»

для ГУО «Боровухской средней школы № 15» г. Новополоцка рЛит.

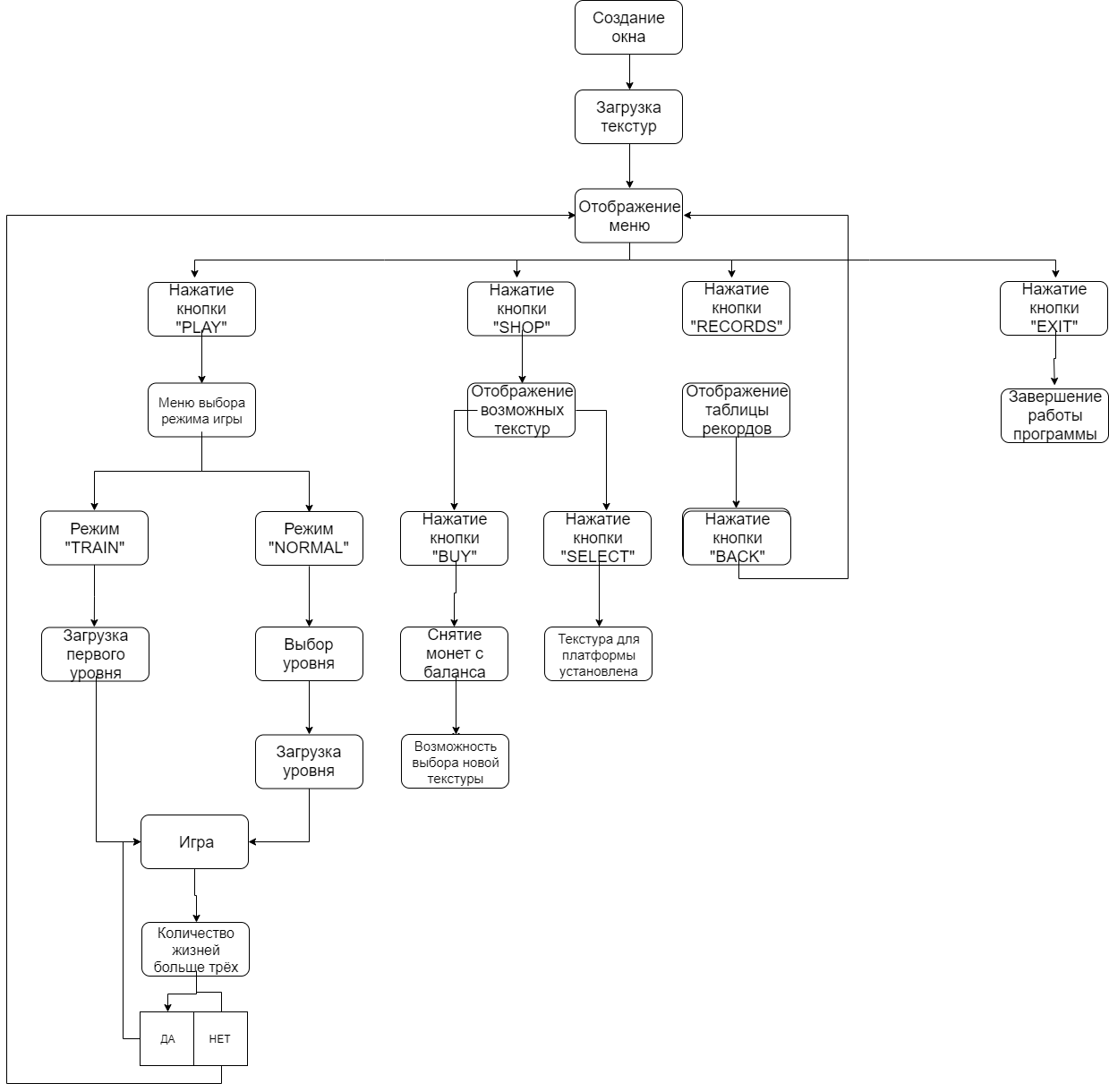
Листов

4

Учреждение образования «Полоцкий государственный университет» гр.18-ИТ-1

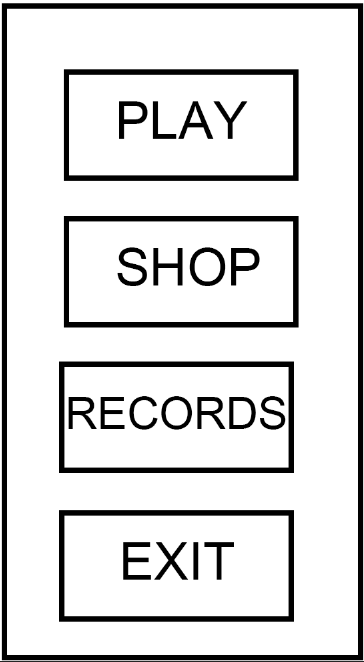
(обязательное)

**Блок-схема алгоритма**

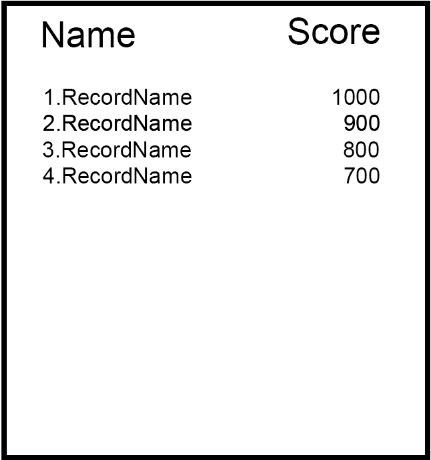
****

**Рисунок А.1** – Алгоритм работы основных функций**ПРИЛОЖЕНИЕ Б**

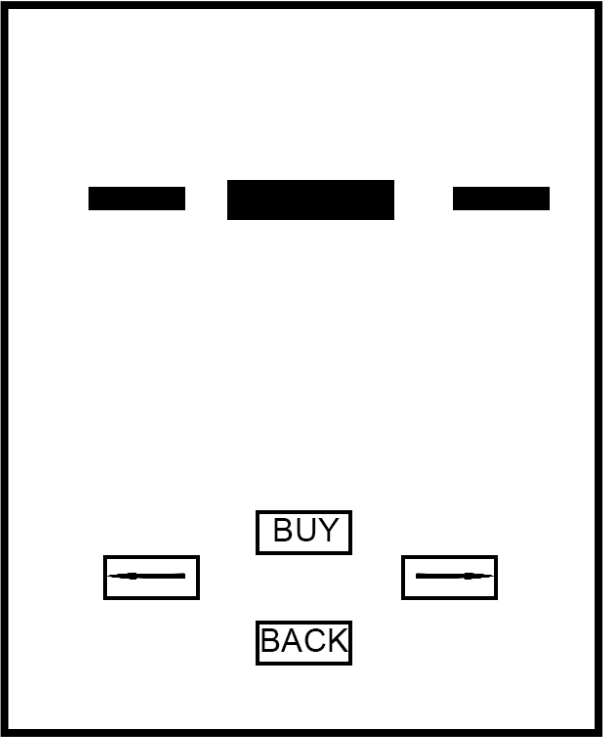
**Прототип интерфейса программы**



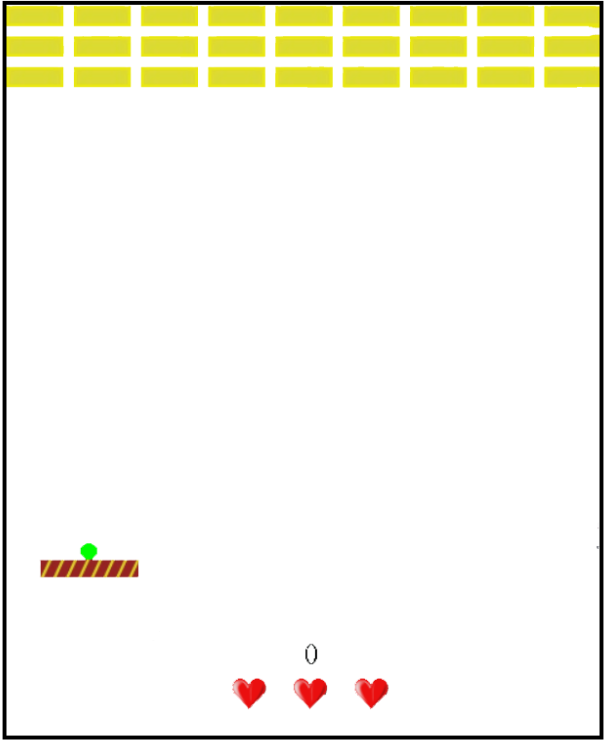
**Рисунок Б.1** – Прототип главного меню



**Рисунок Б.2** – Прототип списка рекордов



**Рисунок Б.3** – Прототип магазина



**Рисунок Б.4 – Прототип игрового процесса**

**ПРИЛОЖЕНИЕ В**

(на диске)

**Листинг программы**