УО «Полоцкий государственный университет»

Кафедра технологий программирования

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

по дисциплине «Языки программирования»

Тема: «Разработка игровой программы «Звёздные Войны»»

Исполнитель: _____ Яблонский А.С.
(подпись)
студент 2 курса группы 16-ИТ-3
Руководитель: _____ Магеров В.В.

Оценка защиты курсового проекта:_____

Дата защиты: «____» _____ 2017г.

Члены комиссии: _____
(подпись)

Полоцк, 2017 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 4

1 АНАЛИЗ ИСХОДНЫХ ДАННЫХ И ПОСТАНОВКА ЗАДАЧИ 6

2 ПРОЕКТИРОВАНИЕ 9

3 РЕАЛИЗАЦИЯ ПРОГРАММЫ 14

4 МЕТОДИКА И РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ 23

ЗАКЛЮЧЕНИЕ 26

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 27

ПРИЛОЖЕНИЕ А 28

ПРИЛОЖЕНИЕ Б 30

ПРИЛОЖЕНИЕ В 33

ВВЕДЕНИЕ

Компьютерные игры – одно из наиболее массовых применений электронных вычислительных машин. Поэтому каждый день в этом мире разрабатывается десятки методов и модулей для новой игровой программы.

Самой первой компьютерной игрой стал симулятор ракеты, созданный в 1942 году Томасом Голдсмитом Младшим (англ. Thomas T. Goldsmith Jr.) и Истл Рей Менном (англ. Estle Ray Mann). 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.

Задание курсовой работы требует реализации игровой программы «Звёздные войны» стандартного типа. Игровая программа обязательно должна включать систему подсчета количества набранных баллов, а также реализовывать ведение таблицы рекордов на десять записей.

Тема курсового проекта «Звёздные войны» была выбрана связи с необходимостью рисования игровых объектов и текстур, связанных с данной тематикой, поиск и обработка звукового сопровождения и эффектов, а также проектированием игровой логики: движение противников, появления игровых предметов, ведение игровой статистики и др. Несмотря на то, что первая часть кинофраншизы «Звёздные войны» была опубликована в далёком-далёком 1977 году, данная игровая тематика остаётся популярной и по сей день.

Курсовое задание было необходимо реализовывать на языке C++ с помощью программного интерфейса для написания приложений, использующего двумерную и трёхмерную компьютерную графику – OpenGL.

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 совместимость), 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.

В качестве среды разработки был выбран Clion 2013.3 от JetBrains, язык программирования C++ стандарта ISO/IEC 14882:201.

Язык C++ является универсальным языком программирования с достаточно большими возможностями работы с ОС и аппаратной частью электронных вычислительных машин. В дополнение к его встроенным способностям, 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 ИСХОДНЫХ ДАННЫХ И ПОСТАНОВКА ЗАДАЧ

1.1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Цель курсовой работы является создание игровой программы «Звёздные Войны», с реализацией системы подсчета количества набранных очков, а также ведение таблицы рекордов на десять записей. Под стандартным типом понимается реализация игровой программы с использованием текстур и моделей, а также звукового сопровождения из оригинальной кинофраншизы «Star Wars». Кроме этого, стилистика игры предполагает воссоздание атмосферы игровых автоматов 80-90х годов, при помощи звуко-визуального оформления программы.

Отличительными элементами игры по вселенной «Звёздные Войны» от любой другой являются:

1. Наличие космических кораблей игрока. Например, Сокол Тысячелетия (Millennium Falcon) или X-Wing.
2. Наличие вражеских космических кораблей. Например, TIE Истребитель (TIE Fighter).
3. Наличие световых мечей, световых бластеров и лазеров.
4. Наличие звезды смерти и/или Дарта Вейдера.

1.2 ПОСТАНОВКА ЗАДАЧ

Тематика игровой программы звёздные войны предполагает наличие одного или нескольких ключевых элементов, определяющих вселенную игры. Поэтому было принято решение о включении в игровую программу следующих основных игровых элементов:

1. Космический корабль игрока – игровой объект, обладающий уникальной текстурой, которым находится под управлением игрока и способен перемещаться право, влево, а также производить выстрелы. Постоянно находится на игровом поле в момент игры.

2. Космические корабли противников – игровые объекты, обладающие возможностью самостоятельно принимать решения о передвижении и стрельбе по кораблю игрока. За их появление отвечает логика игры.

3. Дроп-объекты – игровые объекты, появляющиеся на игровом поле случайным образом или выпадающие из уничтоженного корабля противника. При их активации игроком, происходит действие на игровом поле, помогающее игроку различным образом, в зависимости от типа дроп-объекта.

4. Снаряды или лазеры – игровые объекты, появляющиеся на игровом поле после выстрела игроком или кораблями противников. При попадании в другой корабль, причиняют ему урон, определённый в параметрах конкретного снаряда.

5. Задний фон – динамически генерируемое изображение, содержащее определённое количество маленьких звёзд разного размера и случайное количество планет (или Звезду Смерти). Благодаря логики генерации заднего фона, осуществляется воссоздание эффекта полёта космических кораблей в космическом пространстве.

6. Панель информации – нижняя часть окна игровой программы, содержащая индикатор состояния корабля игрока, количество очков, и индикатор отображения прогресса игрока в таблице рекордов.

Важной частью игрового процесса является аудио-звуковое сопровождение.

Музыку в игровой программе можно разделить на два главных типа:

1. Музыка главного меню и меню паузы.
2. Музыка игрового процесса.

Звуковое сопровождение программы позволяет пользователю получать отклик от взаимодействия с ней, например, звук клика по клавиши. Можно выделить следующие типы звуков в игровой программе:

1. Звук наведения указателя мыши на клавишу в главном меню и меню паузы.
2. Звук клика указателем мыши по клавише в главном меню и меню паузы.
3. Звуки выстрела корабля игрока.
4. Звуки выстрелов кораблей противников.
5. Звук попадания по кораблю игрока или королям противника
6. Звук уничтожения кораблей противников.
7. Звуки активации дроп-объектов. Тип звука может быть указан в структуре конкретного объекта.

2 ПРОЕКТИРОВАНИЕ

В связи с большим количеством игровых объектов, необходимостью реализации логики врагов, наличием большой аудио-звуковой базы и большим количеством текстур, было принято решение о разделение кода на отдельные логические модули. Данный подход способен существенно упростить архитектуру проекта, и облегчить дальнейший процесс масштабирования при необходимости. Еще одним плюсом модульной системы, является взаимозаменяемость модулей с другими проектами.

В процессе проектирования были выделены следующие основные модули:

1. Player – модуль, отвечающий за обновления, обработку движений и событий, связанных с игроком и его кораблём.
2. Enemy – модуль, отвечающий за обновления, обработку движений и их логики и событий, связанных с вражескими кораблями.
3. Drop – модуль, отвечающий за обработку движений, событий, действий, связанных с дроп-объектами.
4. Background – модуль отвечает за рисование и перемещение объектов заднего фона: звёзды и планеты.
5. Bullet – модель отвечает за рисование, перемещение и обслуживание всех пуль(лазеров) на игровом экране.

Модуль Player основной модуль, отвечающий за обработку действий игрока. Главные методы данного модуля:

1. mdlPlayerInit() – метод инициализации основных полей модуля и структуры игрока.
2. mdlPlayerUpdate(…) – метод обновления модуля. Вызывается каждый раз, при обновлении экрана. Планируется использовать для перерасчёта позиции, количества патронов в орудии, состояния и тд.
3. mdlPlayerSetShotListener(callback(…)) – метод установки callback функции совершения выстрела игроком.

4. mdlPlayerSetHealthListener(callback(…)) – метод установки callback функции изменения количества жизней игрока.

5. mdlPlayerSetTakeDropListener(callback(…)) – метод установки callback функции активация дроп-объектов игроком.

6. mdlPlayerGoRight() – метод перемещения игрока вправо. Выполняется после нажатия пользователем клавиши движения вправо.

7. mdlPlayerGoLeft() – метод перемещения игрока влево. Выполняется после нажатия пользователем клавиши движения влево.

8. mdlPlayerShoot() – метод совершения выстрела игроком. При совершении выстрела вызывается callback метод переданный в mdlPlayerSetShotListener(…).

Модуль Enemy основной модуль, отвечающий за обработку, рисование и логику действий врагов. Главные методы данного модуля:

1. mdlEnemyInitAll() – метод инициализации основных полей модуля и структур врагов.

2. mdlEnemyUpdateAll(…) – метод обновления модуля. Вызывается каждый раз, при обновлении экрана. Планируется использовать для появления новых врагов на экране, перерасчёта их позиций, количества патронов в орудиях, состояний и тд.

3. mdlEnemySetCrossBorderListener(callback(…)) – метод установки callback функции пересечения границы каким-либо врагом. При вызове данной callback функции, игра заканчивается – Game Over.

4. mdlEnemySetEnemyDamageListener(callback(…)) – метод установки callback функции изменения количества жизней какого-либо врага.

5. mdlEnemySetShootListener(callback(…)) – метод установки callback функции совершения выстрела каким-либо врагом.

Модуль Drop основной модуль, отвечающий за обработку, рисование и действия дроп-объектов при их активации. Главные методы данного модуля:

1. mdlDropInit() – метод инициализации основных полей модуля.

2. mdlDropUpdate(…) – метод обновления модуля. Вызывается каждый раз, при обновлении экрана. Планируется использовать для перерасчёта позиций, состояний и тд.

3. mdlDropAddNew(drop) – метод добавляет новый дроп-объект в список для последующей обработке модулем.

4. mdlDropAction(drop) – метод активирует действие данного дроп-объекта.

Модуль Background основной модуль, отвечающий за рисование, обновление и смещение заднего фона. Главные методы данного модуля:

1. mdlBackgroundInitAll() – метод инициализации основных полей модуля и структур заднего фона, например звезд и планет.

2. mdlBackgroundUpdateAll(…) – метод обновления модуля. Вызывается каждый раз, при обновлении экрана. Планируется использовать для перерасчёта позиций, состояний и тд.

Модуль Bullet основной модуль, отвечающий за рисование, обновление и перемещение снарядов. Также обрабатывает попадания снарядов в игрока и/или врагов. Главные методы данного модуля:

3. mdlBulletInitAll() – метод инициализации основных полей модуля.

4. mdlBulletUpdateAll(…) – метод обновления модуля. Вызывается каждый раз, при обновлении экрана. Планируется использовать для перерасчёта позиций, состояний, регистрации попаданий и тд.

5. mdlBulletDrawBulletHit(bullet) – метод рисования попадания снаряда по цели. Рисует анимацию попадания пули по телу.

6. mdlBulletAddNew(bullet) – метод добавляет новый снаряд в список для последующей обработке модулем.

Схема подключения основных модулей изображена на рисунке далее (Рисунок 2.1).

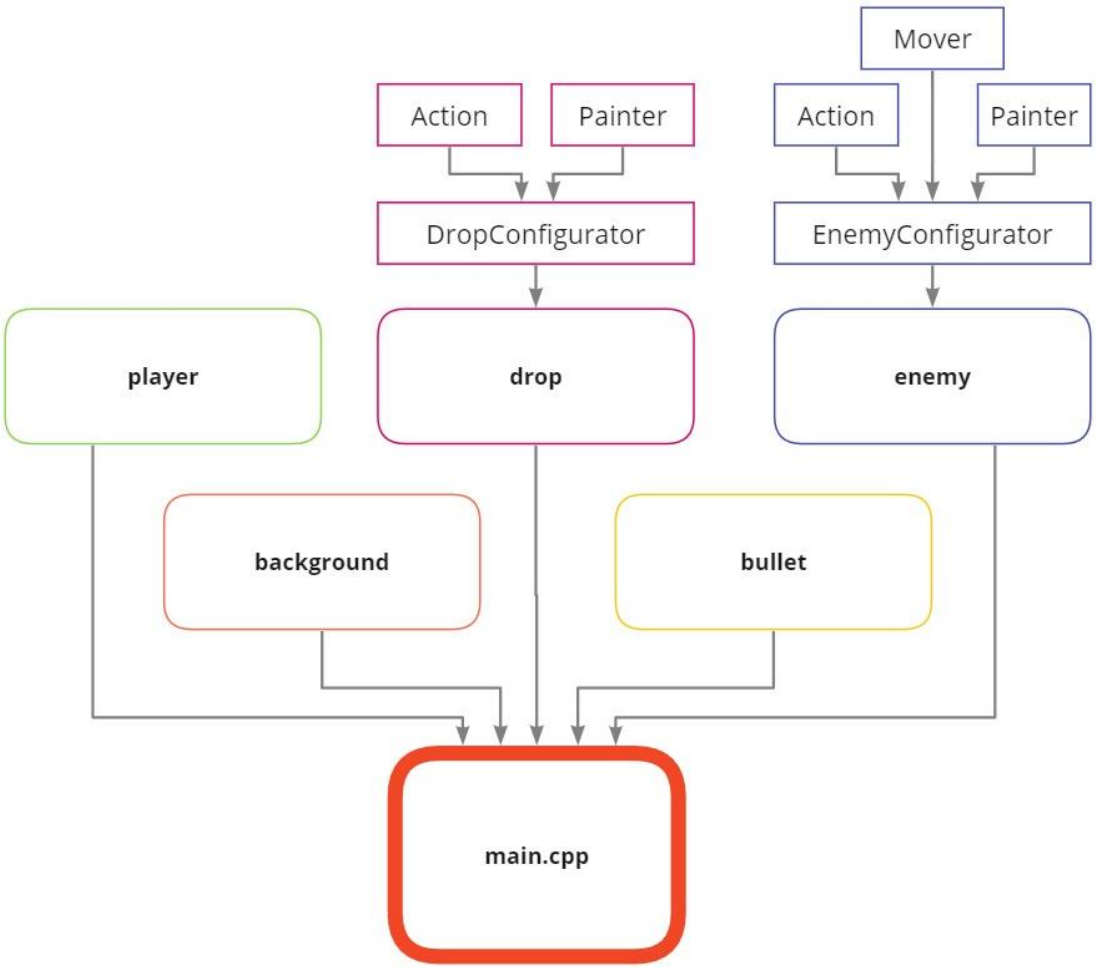

Рисунок 2.1 – Схема подключения основных модулей программы

Кроме основных игровых модулей возникла необходимость обслуживать дополнительные элементов игрового процесса, таких как интерфейс, ресурсы и другое. Для этого были созданы следующие модули:

1. IOProcessor – обслуживает потоки ввода вывода, обрабатывает события ввода OpenGL. Позволяет реализовать буфер нажатых клавиш для обработки нажатий сразу от нескольких источников, например, одновременное нажатие клавиши пробела и стрелок клавиатуры. Также модуль позволяет обрабатывать отдельные состояния нажатия: press (нажатие), hold (зажатие), release (отпускание).

2. Модули Sound, Music и Texture Manager – загружают необходимые ресурсы, и позволяют эффективно взаимодействовать с ними.

3. UI модуль – отвечает за рисование игрового интерфейса (нижний части окна игры). Упрощает разметку игрового интерфейса и рисование индикаторов игрового процесса.

4. Dialog модуль – является частью UI модуля. Реализует функции рисования главного меню, меню паузы, таблиц рекордов. Осуществляет обработку ввода текста имени нового рекорда.

Было принято решение разделить сложные модули на подмодули, например, модули Drop и Enemy, на: Painter, Aimer, Mover и подмодули конфигурации. Это облегчило добавление новых игровых объектов данных типов. Схема подключения дополнительных модулей изображена на рисунке (Рисунок 2.2).

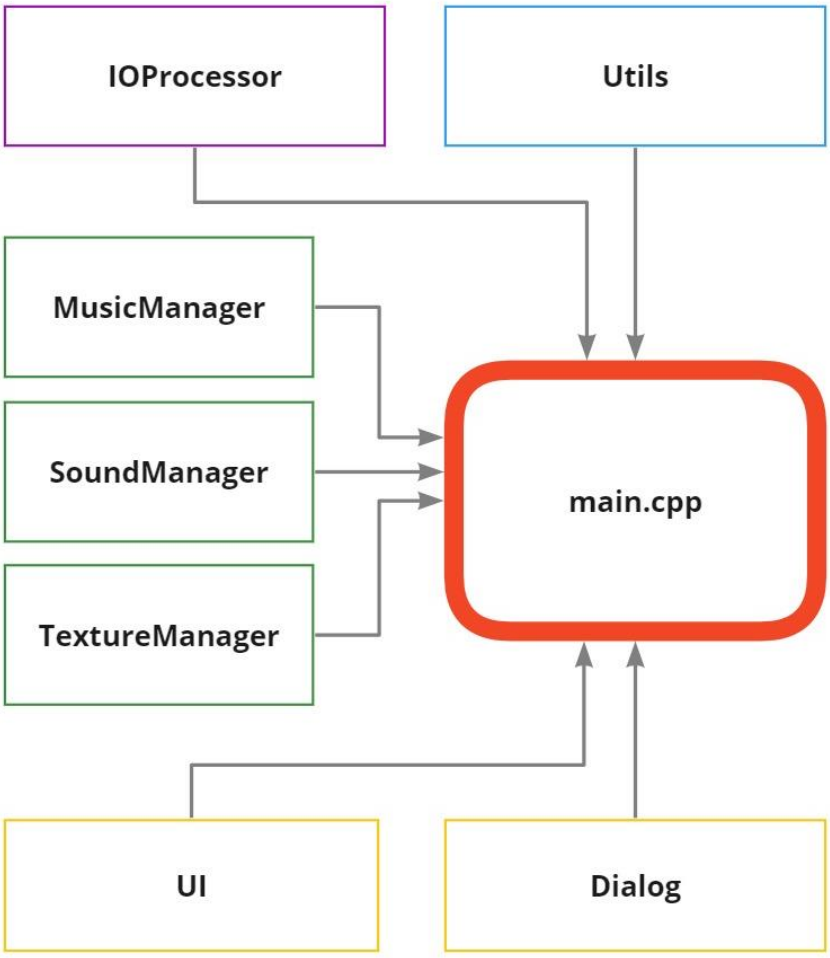

Рисунок 2.2 – Схема подключения дополнительных модулей программы

3 РЕАЛИЗАЦИЯ ПРОГРАММЫ

Как можно заметить на схемах подключения моделей (Рисунок 2.1), (Рисунок 2.2), связующим звеном является файл main.cpp. Именно в нем реализуется подключение и вызов функция инициализации первичными значениями у всех модулей. Также main.cpp подключает и устанавливает параметры библиотеки OpenGL.

В главном файле игровой программы можно выделить следующие основные функции:

1. Функция onRedraw() – один из важнейших методов программы. Является параметром функции рисования OpenGL – glutDisplayFunc(). Данная функция вызывает методы обновления главных и вспомогательных моделей.

2. Функция createNewGame() – инициализирует необходимые поля модулей стартовыми значениями. Выполняется каждый раз, когда игрок начинает новую игру.

3. Функция initGame() вызывается один раз при запуске игры. Вызывает методы загрузки ресурсов у соответствующих модулей.

4. Функции onKeyPress(), onKeyHold(), onKeyRelease() – являются callback методами модуля IOProcessor. Выполняются каждый раз, когда пользователь нажимает, удерживает или отпускает клавишу мыши и/или клавиатуры соответственно. К методам этой группы также можно отнести callback метод onMouseMove() этого же модуля. Выполняется каждый раз, когда пользователь перемещает курсор мыши.

Листинг главных методов и функций файла main.cpp приведён далее (Листинг 2.1).

Листинг 2.1 – основные функции main.cpp и их описание

```
1:   void onKeyPress(int key, int x, int y, bool special); //
2:   callback метод модуля IOProcessor. Выполняется каждый раз,
3:   когда пользователь нажимает клавишу мыши или клавиатуры.
4:   void onKeyHold(int key, int x, int y, bool special); //
5:   callback метод модуля IOProcessor. Выполняется каждый раз,
```

```
6:   когда пользователь зажимает клавишу мыши или клавиатуры.
7:   void onKeyRelease(int key, int x, int y, bool special); //
8:   callback метод модуля IOProcessor. Выполняется каждый раз,
9:   когда пользователь отпускает клавишу мыши или клавиатуры.
10:  void onMouseMove(int x, int y); // callback метод модуля
11:  IOProcessor. Выполняется каждый раз, когда пользователь
12:  перемещает указатель мыши.
13:  void onUIItemSelect(GameState state, int select); // callback
14:  метод модуля UI. Выполняется каждый раз, когда пользователь
15:  кликает по кнопки главного меню, меню паузы, или окна
16:  таблицы рекордов.
17:  GameFieldStruct createNewGame(); // метод вызывается каждый
18:  раз, когда пользователь начинает новую игру. Инициализирует
19:  все необходимые поля модулей и структур начальными
20:  значениями.
21:  void initGame(); // вызывается единожды, сразу после запуска
22:  игры. Вызывает методы загрузки ресурсов у соответствующих
23:  модулей и инициализацию полей структур.
```

В процессе реализации программы была выявлена необходимость в создании структуры, хранящей состоянии игры в каждый момент времени. Для этих целей была разработана структура GameFieldStruct (Листинг 3.1).

Листинг 3.1 – главная структура игры

```
1:   struct GameFieldStruct {
2:   float windowX; // размеры окна по горизонтали
3:   float windowY; // размеры окна по вертикали
4:   GameState gameState = GAME_STATE_MAIN_MENU; // текущие
состояние игры: игра, меню, пауза и т.д.
5:   int difficult; // сложность
6:   int score; // счёт
7:   SW_Borders gameBorders; // границы игрового поля
8:   SW_Borders interfaceBorders; // границы строки интерфейса
9:   SW_Background background; // структура модуля фона
10:  SW_Bullet_Map bulletMap; // структура модуля снарядов
11:  SW_Enemy_Map enemyMap; // структура модуля врагов
```

```
12:  SW_Drop_Map dropMap; // структура модуля дроп-объектов
13:  SW_Player player; // структура модуля игрока
14:  SW_Record  recordList[PREF_RECORD_LIST_SIZE];  //  таблица
рекордов
15:  };
```

Каждый основной модуль имеет собственную структуру для обработки. Например, для модуля Background структура Background, для модуля Bullet - SW_Bullet_Map, для модуля Enemy - SW_Enemy_Map, для модуля Drop - SW_Drop_Map и для модуля Player структура SW_Player. Листинг данных структур приведён далее.

Листинг 3.2 – структура модуля Bullet

```
1:   struct SW_Bullet_Map {
2:   const static int maxNumber = PREF_BULLET_BUFFER_SIZE;
3:   SW_Bullet list[maxNumber];
4:   };
```

Листинг 3.3 – структура модуля Enemy

```
1:   struct SW_Enemy_Map {
2:   loat enemiesHealth = 0; // здоровье всех врагов на игровом
поле
3:   const static int maxNumber = PREF_ENEMY_BUFFER_SIZE;
4:   SW_Enemy list[maxNumber];
5:   };
```

Листинг 3.4 – структура модуля Drop

```
1:   struct SW_Drop_Map {
2:   const static int maxNumber = PREF_DROP_BUFFER_SIZE;
3:   SW_Drop list[maxNumber];
4:   };
```

Листинг 3.5 – структура модуля Player

```
1:   struct SW_Player {
2:   SW_Borders hitBox; // хит-бокс игрока
3:   SW_Gun gun; // текущее оружие игрока
4:   SW_State state = PLAYER_STATE_STAY_FORWARD; // состояние
5:   float health; // здоровье
6:   SW_Pos pos; // позиция
7:   SW_Speed speed; // скорость перемещения
8:   };
```

В процессе разработки возникла необходимость хранения константных значения для настроек приложения. Для данных целей был создан заголовочный файл PREFERECES.h. Файл предназначен для хранения и быстрого доступа к константам разного типа. Например, размеры окна игры, изначальное значение здоровья корабля игрока, сложность игры, и т.д.

В программе был нарисован интерфейс для отображения главного меню (Рисунок 3.1), меню паузы (Рисунок 3.2), таблицы рекордов с возможностью добавление записи (Рисунок 3.3), таблицы рекордов (Рисунок 3.4). За рисование неигрового интерфейса отвечает модуль Dialog, которые предоставляет функции для рисования окна, кнопок, подсвеченных кнопок и т.д. в одном стиле.

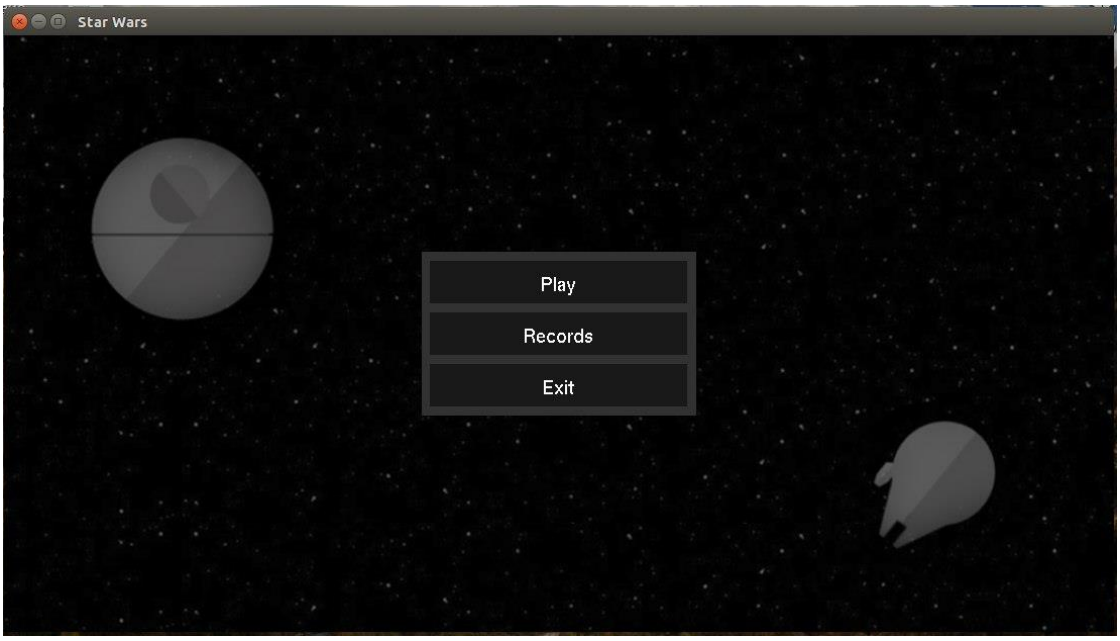

Рисунок 3.1 – Основное меню игры

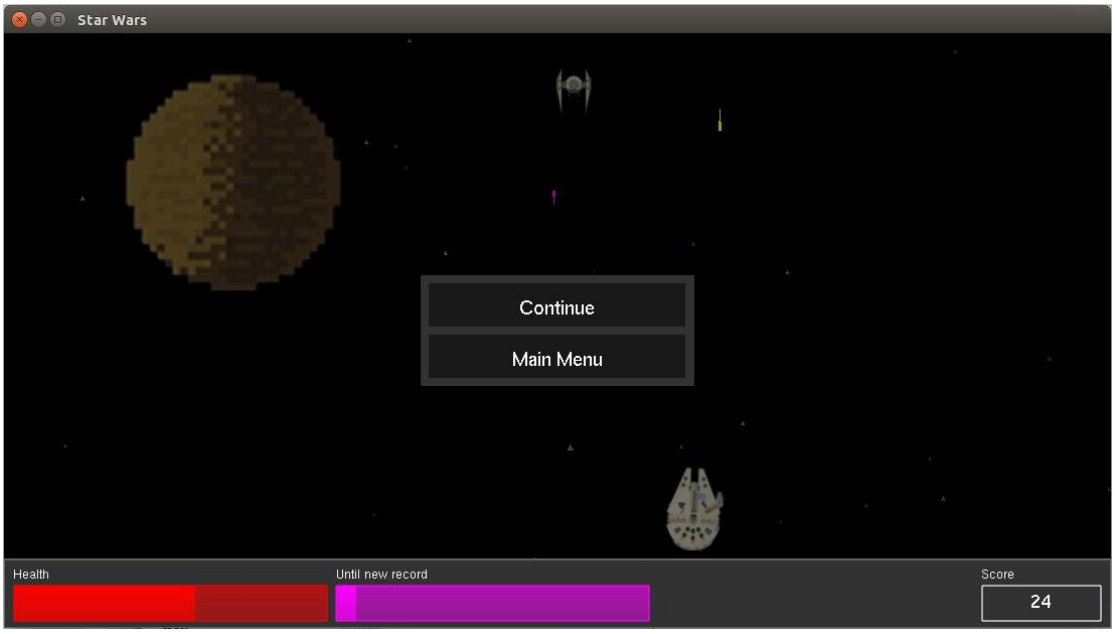

Рисунок 3.2 – Меню паузы игры

Благодаря модулю рисования диалогов - Dialog, есть возможность рисования окна, кнопок, подсвеченных кнопок и т.д. в одном стиле. Что положительно сказывается на общем восприятии игры игроком.

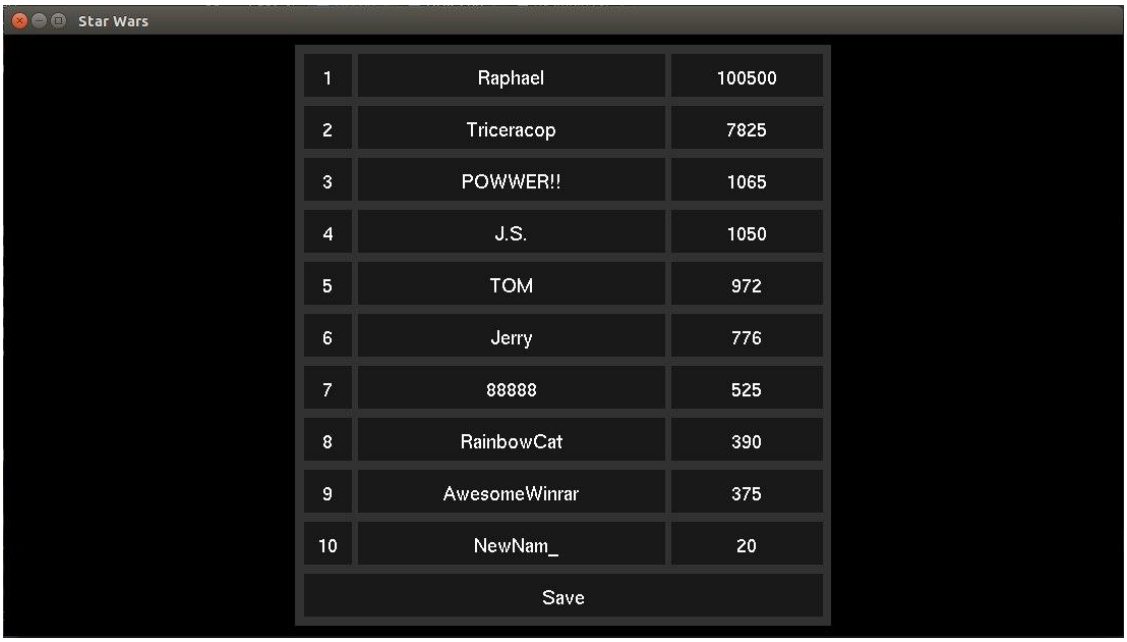

Рисунок 3.3 – Талица добавления нового рекорда

При добавлении нового рекорда в таблицу, указания позиции ввода осуществляется мигающим горизонтальным курсором. Максимальная доступная для ввода длинна имени – 16 символов.

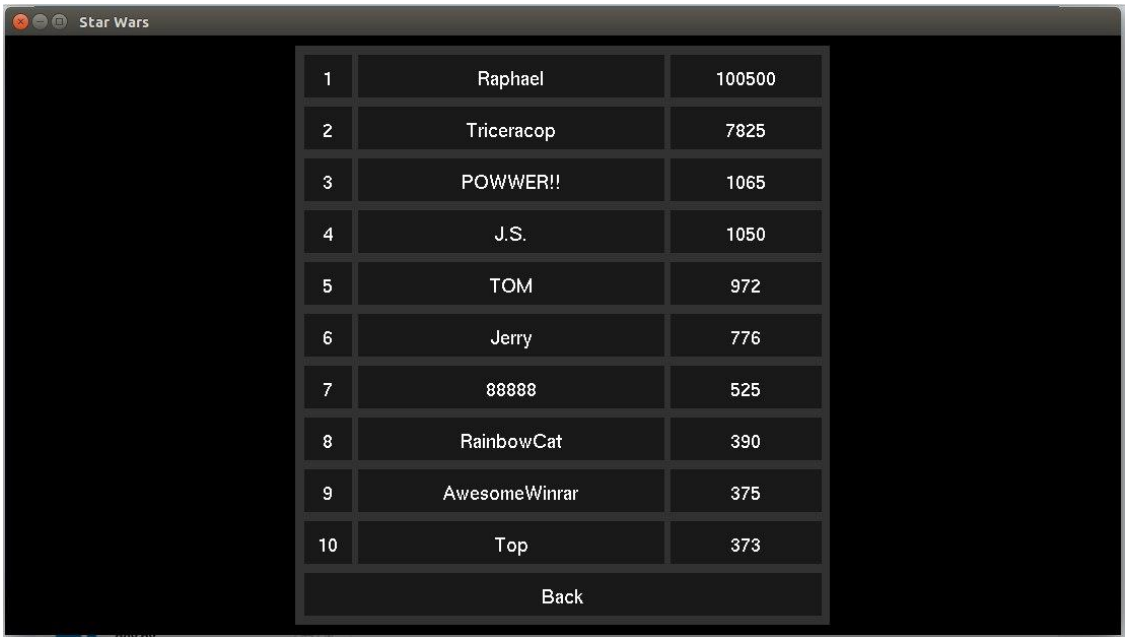

Рисунок 3.4 – Талица рекордов

Таблица отображение рекордов (Рисунок 3.3), и таблица добавления нового рекорда (Рисунок 3.4) отличаются текстом кнопки и возможностью осуществлять ввод текста. Такой подход к использованию возможностей модуля Dialog положительно сказывается на общем размере исходного файла игры.

Подобным образом рисуется окно окончания игры и окно окончания игры с новым рекордом (Рисунок 3.5). Отличия заключаются лишь в присутствии надписи «New Record!» и действии после нажатия клавиши «Enter». Данный подход также помогает уменьшить размер файлов игры.

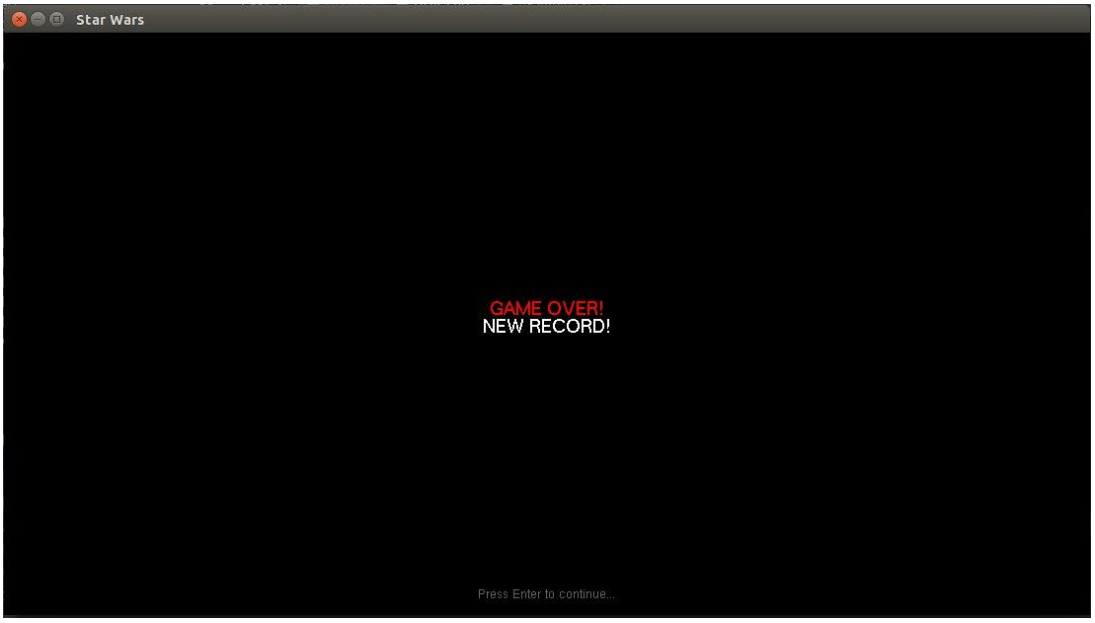

Рисунок 3.5 – Окно окончания игры с новым рекордом

Окно программы в процессе игры можно разделить на две зоны: игровая зона и зона игрового интерфейса.

Первая представляет игровой поле, на котором происходит всё действие: передвижение игрока, появление врагов и т.д. Оно расположено в верхней части экрана (Рисунок 3.6).

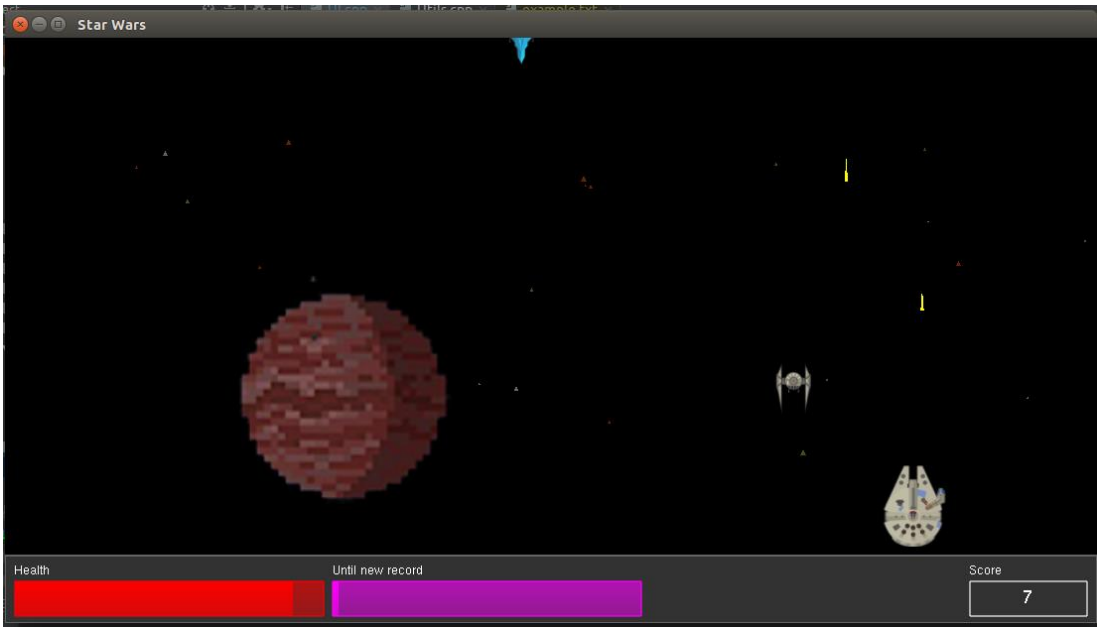

Рисунок 3.6 – Окно программы в процессе игры

Игровой интерфейс располагается в нижней части экрана. Его задача предоставлять игроку информацию о состоянии и действий в игре. В данном интерфейсе игрок найдет индикатор здоровья, индикатор до нового рекорда и свои очки за текущую игру. Все поля игрового интерфейса подписаны, что поможет игроку ориентироваться в нем.

Данная реализация игры «Звёздные Войны» полностью на английском языке, что позволяет беспроблемно взаимодействовать с программой практически каждому.

Благодаря применению текстур, игра становится динамичнее и реалистичнее, что позволяет игроку глубже погрузиться во вселенную Звёздных Войн.

Некоторые игровые элементы содержат несколько вариантов текстур, которые могут накладываться в зависимости от состояния объекта или быть выбраны случайным образом. Например, текстуры планет на заднем фоне (Рисунок 3.8). Также, с некоторыми текстурами похожего эффекта можно добиться простым масштабированием и/или поворотом. Например, такой способ используется для рисования текстуры попадания снаряда по кораблю.

Таким способом можно получить внутри игровое разнообразие с минимальными требованиями к памяти.

Рисунок 3.7 – Некоторые текстуры вражеских кораблей

Рисунок 3.8 – Некоторые текстуры планет

Хорошим подходом с использованием графических ресурсов является кооперация текстур со встроенными в OpenGL функциями трансформации примитивов. В данном проекте такое можно заметить при перемещении корабля игрока: корабль изменяет угол наклона в зависимости от направления движения (Рисунок 3.9).

Рисунок 3.9 – Корабль игрока в разных положениях

4 ТЕСТИРОВАНИЕ

Тестирование программного обеспечения – процесс исследования, испытания программного продукта. Целью тестирования является выявить ситуации, в которых поведение программы является неправильным, нежелательным или не соответствующим спецификации.

Тестированием программного обеспечения проводилось в определённом порядке с целью получения информации о качестве работы.

Тестирование обеспечивает:

1. Обнаружение ошибок.
2. Демонстрацию соответствия функций программы ее назначению.
3. Демонстрацию реализации требований к характеристикам.

Игровая программа разрабатывалась на ноутбуке со следующими характеристиками:

1. Процессор: Intel Core i5-6200U 2.80 ГГц.
2. Оперативная память: 8 Гбайт.
3. Экран: 1366 x 768 Пикселей.

Корректность выполнения определяется сравнением фактического и ожидаемого результатов. Если результаты совпали, то тестирование пройдено успешно. Если же результаты не равны, то произойдет сбой системы и пользователю выведется системное окно об ошибке.

На этапе компиляции, запуска программы ошибок не обнаружено.

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.

Входе выполнения тестирования были выявлены некоторые проблемы и неточности в работе программы:

1. Некорректное сохранение таблицы рекордов в файл – исправлено устранением ошибки в функции сохранения данных (Рисунок 4.1).

2. Отображение ошибочных данных индикатором нового рекорда после 10 места – исправлено оптимизацией алгоритма просчёта процента для индикатора.

3. Проблемы рисования планет (Рисунок 4.2) – исправлено корректировкой методов рисования.

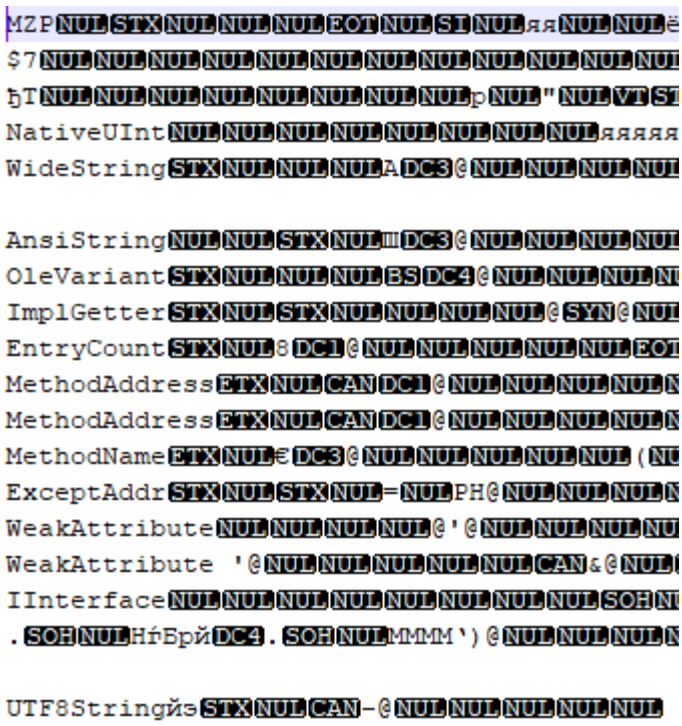

Рисунок 4.1 – Проблема сохранения рекордов в файл

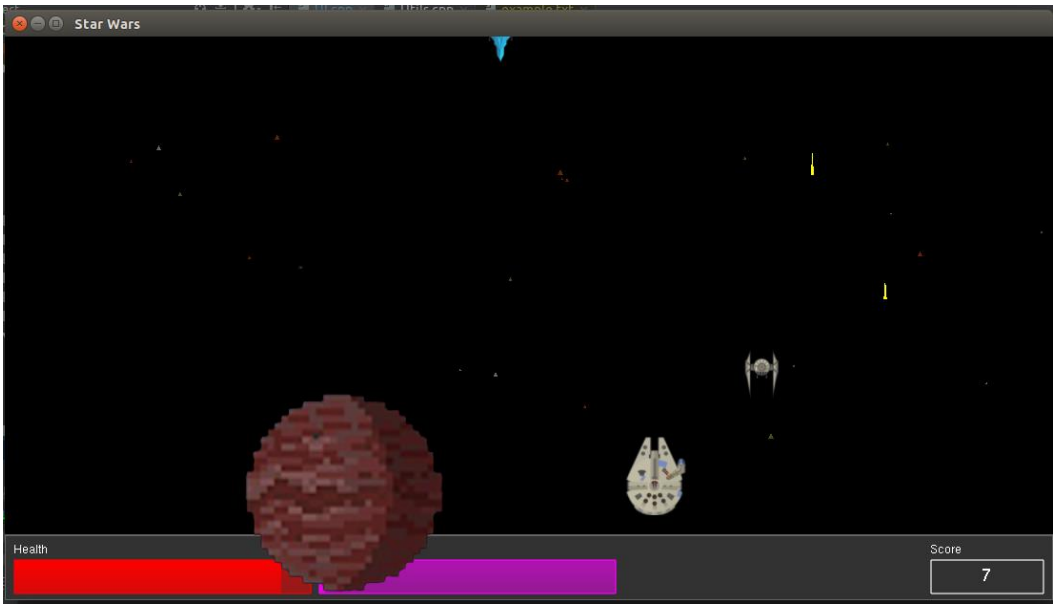

Рисунок 4.2 – Ошибка рисования

Все выявленные в ходе выполнения тестирования обнаруженные ошибки были своевременно исправлены. При повторном тестировании они не были обнаружены.

По окончанию тестирования было установлено, что программный продукт работает стабильно – ошибок и неисправностей не возникает.

ЗАКЛЮЧЕНИЕ

В данной курсовой работе была поставлена цель, используя полученные знания языка программирования C++ и библиотеки рисования 2D и 3D графики OpenGL, реализовать работоспособное игру «Звёздные войны» стандартного типа, с обязательной реализацией системы подсчёта количества набранных баллов, таблицей рекордов на десять записей.

В ходе выполнения были освоены навыки работы с функциями OpenGL и улучшены навыки работы с языком C++.

Целевой аудиторией игровой программы «Звёздные войны» предполагаются любители игрового жанра «Аркада», который требует напряжения внимания и быстрой реакции на происходящие в игре события.

Программа может быть модернизирована и дополнена. Благодаря гибкой модульной системе возможно увеличение количества типов противников, улучшение анимации, игровой графики в целом и др.

В программе присутствует интуитивно понятный интерфейс, лёгкое управление, аудио-звуковые эффекты и приятые глазу текстуры, что сказывается на общем впечатлении от реализованной игры.

В итоге программа реализована в соответствии всем заявленным требованиям, протестирована надлежащим образом, и может использоваться по установленному назначению. Поставленную задачу считаю выполненной в полной мере.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Конспект лекций по дисциплине «Языки программирования».
2. Учебное пособие Dave Shreiner - OpenGL Programming Guide. Versions 3.0 and 3.1 (7th Ed., 2009).
3. Учебное пособие Ричард Райт, Бенджамин Липчак. OpenGL Суперкнига.
4. Официальный сайт «OpenGL» – Режим доступа: https://www.opengl.org/.Дата доступа: 10.11.2017.
5. Основы программирования на языках Си и C++. – Режим доступа: http://cppstudio.com/cat/274/. Дата доступа: 10.11.2017.
6. Интернет-портал «Porti.ru – информационный портал». – Режим доступа: http://porti.ru/articles/502/. Дата доступа 20.11.2017.
7. Интернет-портал «Хабрахабр – Режим доступа»: https://habrahabr.ru/post/311198/. Дата доступа 25.11.2017.

ПРИЛОЖЕНИЕ А

Блок-схемы алгоритмов

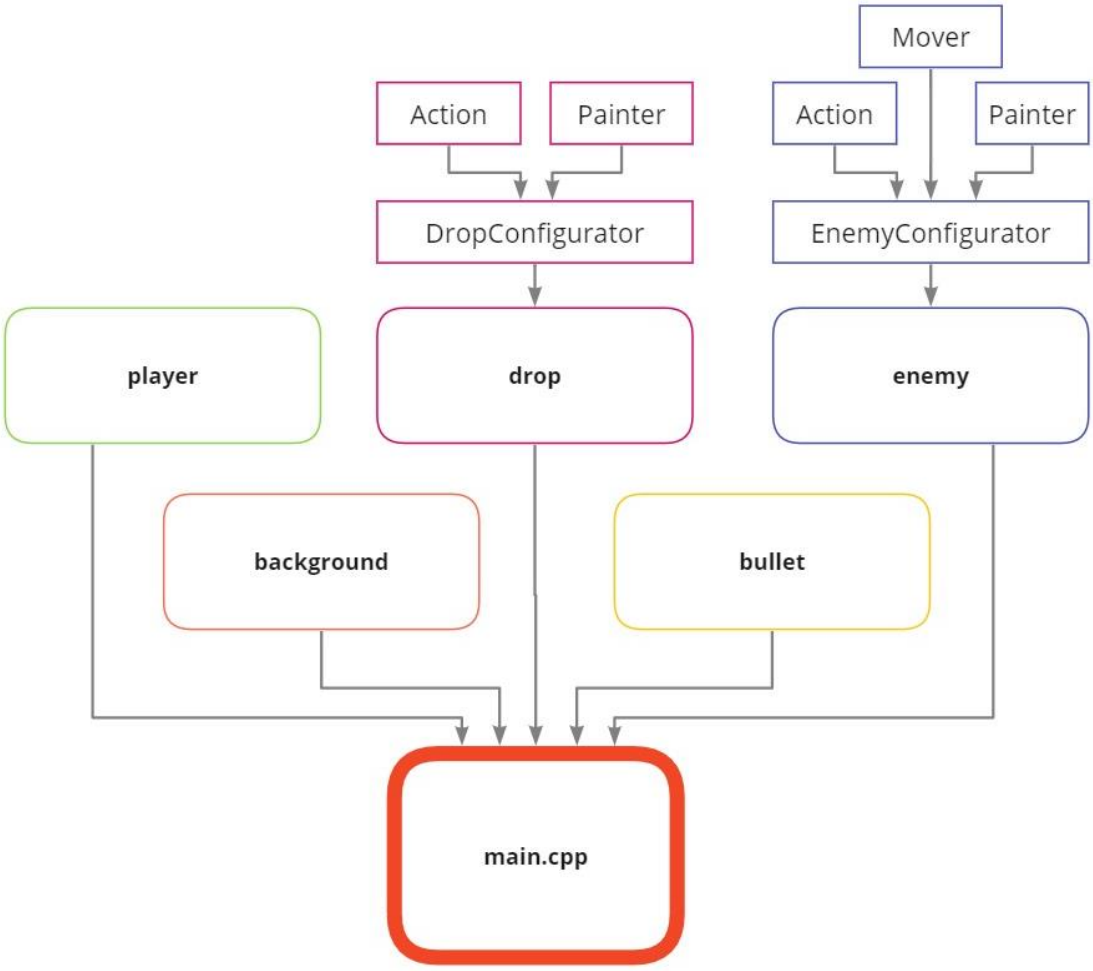

Рисунок А.1 – Схема подключения основных модулей программы

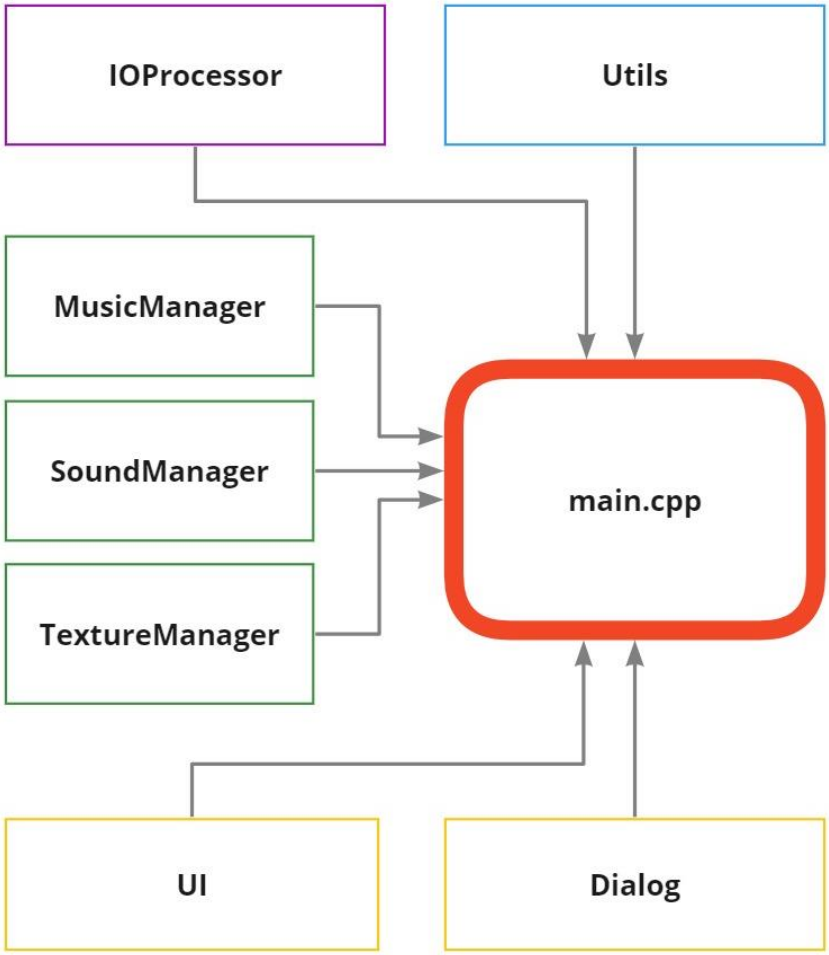

Рисунок А.2 – Схема подключения дополнительных модулей программы

ПРИЛОЖЕНИЕ Б

Интерфейс программы

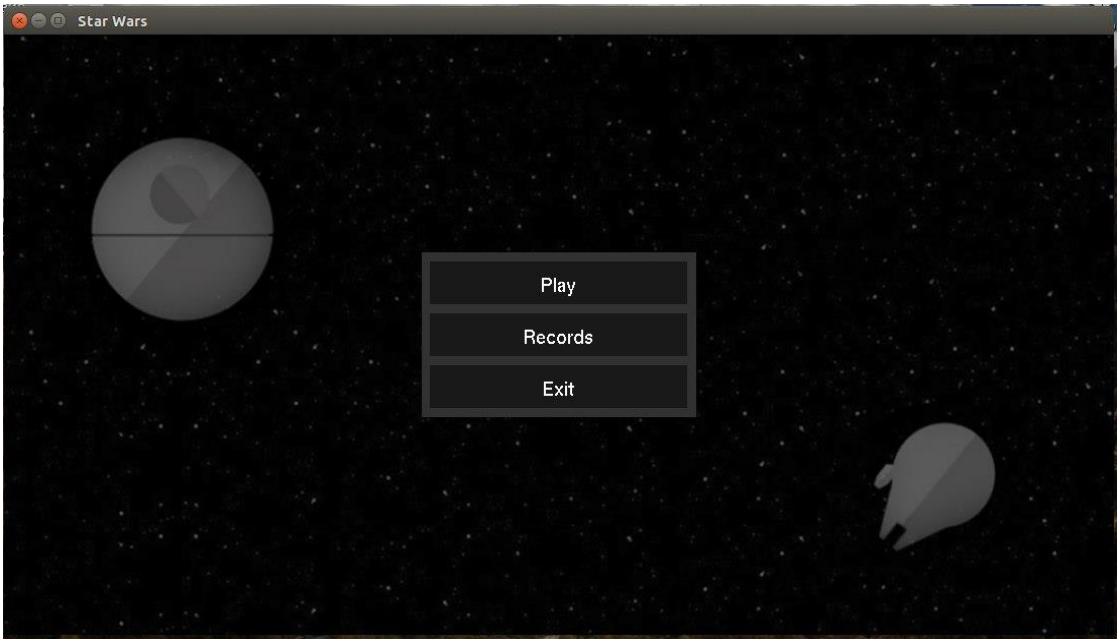

Рисунок Б.1 – Основное меню игры

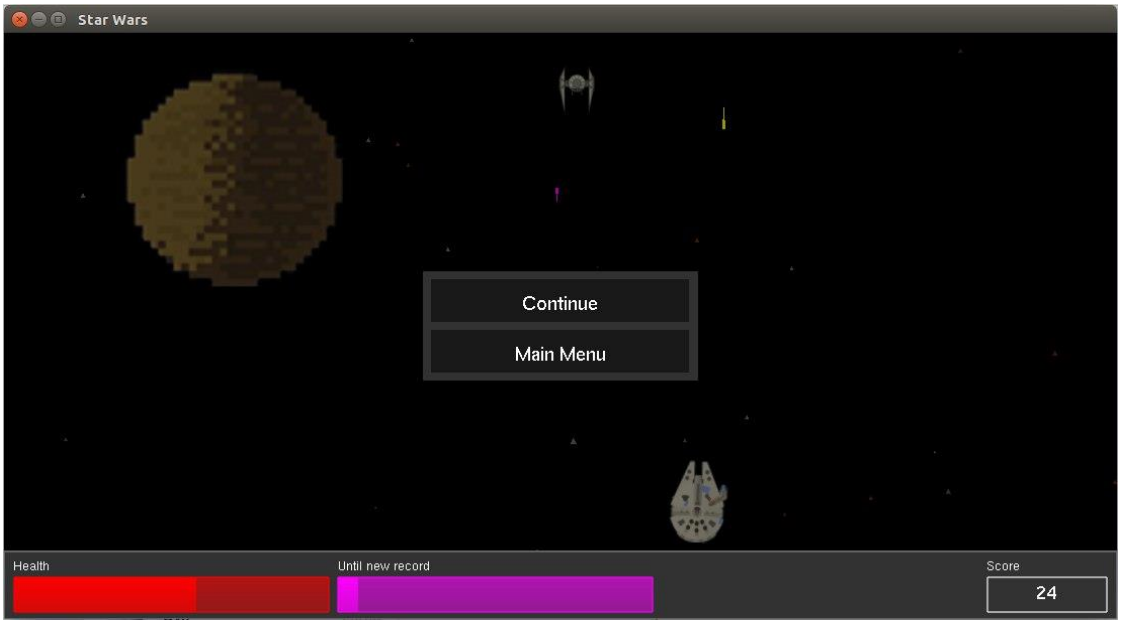

Рисунок Б.2 – Меню паузы игры

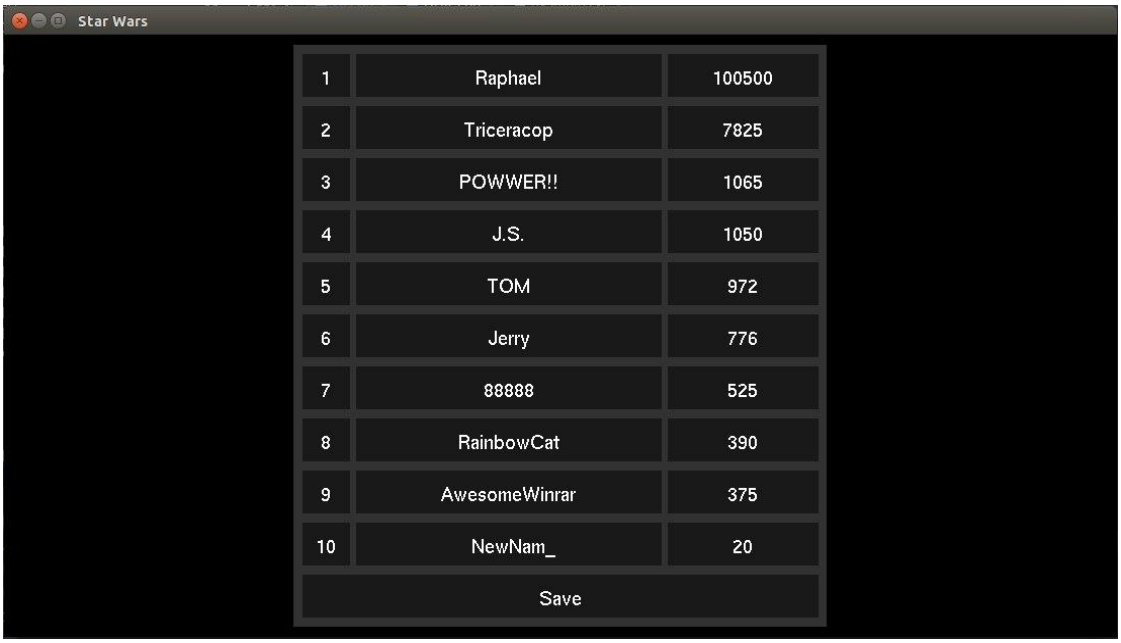

Рисунок Б.3 – Талица добавления нового рекорда

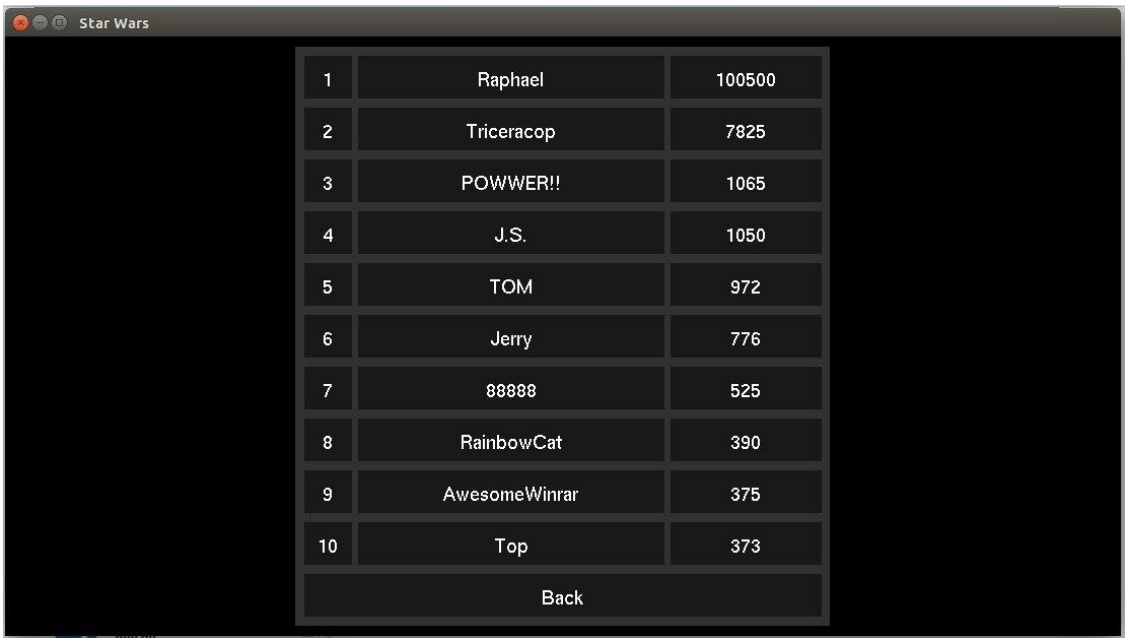

Рисунок Б.4 – Талица рекордов

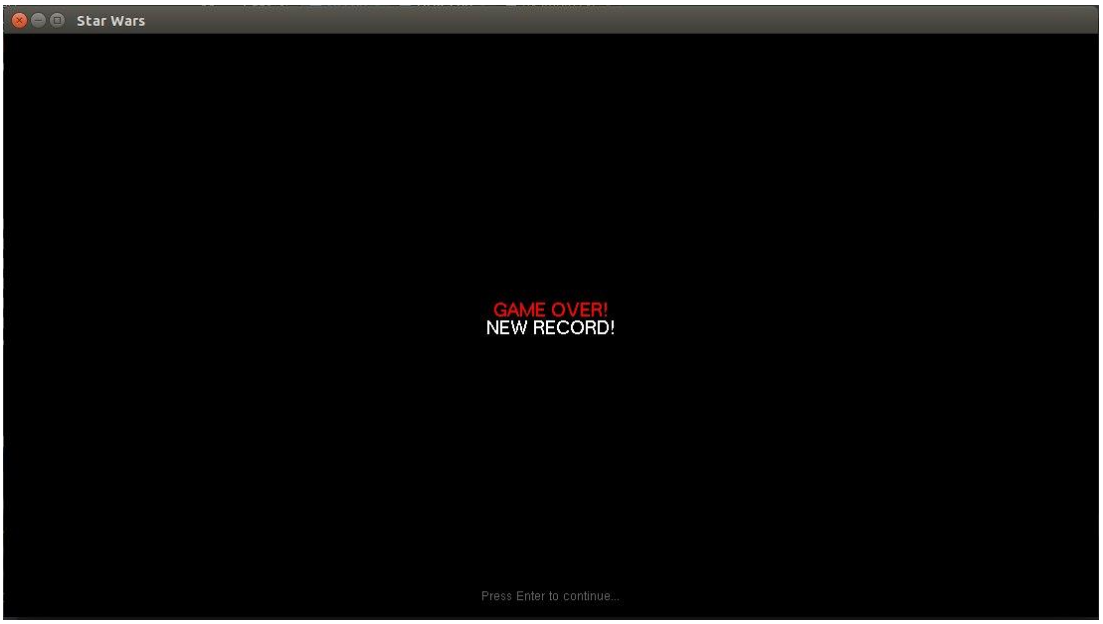

Рисунок Б.5 – Окно окончания игры с новым рекордом

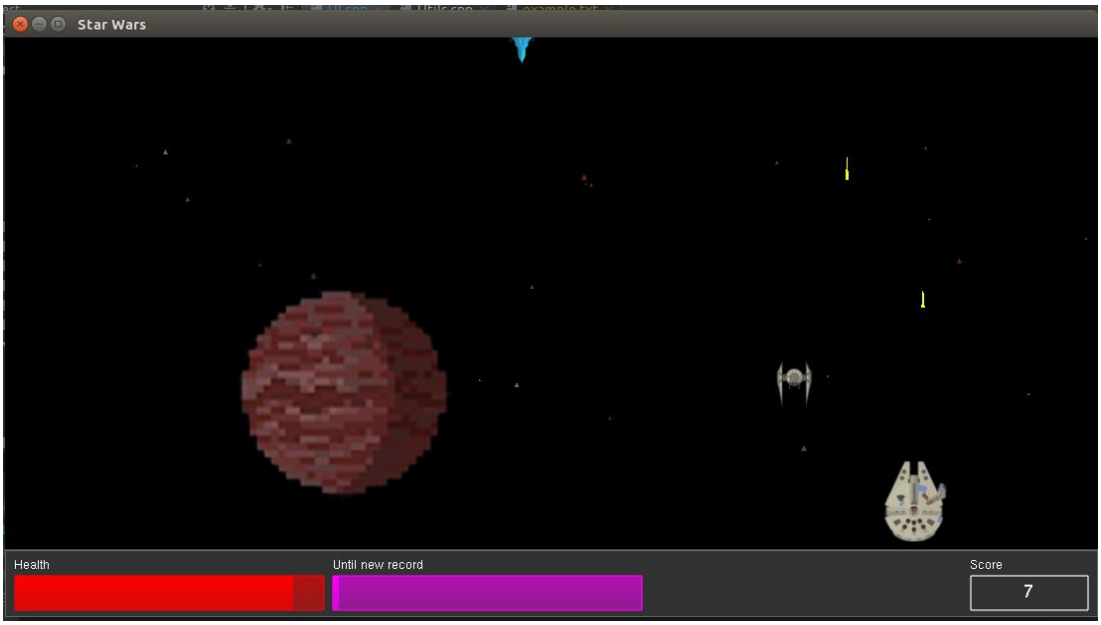

Рисунок Б.6 – Окно программы в процессе игры

ПРИЛОЖЕНИЕ В

Листинг программного кода

Листинг программного кода предоставлен на диске, приложенном к пояснительной записке по курсовому проекту.