Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
Южно-Уральский государственный университет
(национальный исследовательский университет)
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ

.....
..... 2017 г.

Разработка мультимедийного игрового приложения «Push Him All»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ-230100.2017.382 ПЗ ВКР

Руководитель работы
доцент кафедры ЭВМ ВШЭКМ
..... / И.Н. Надточий
«____» _..... 2017 г.

Автор работы Иванов А.С.
студент группы КЭ–445
«_____» _..... 2017 г.

Нормоконтроллер,
ст. преп. каф ЭВМ ВШЭКМ
_...../В.В. Лурье
«_____» _..... 2017 г.

Челябинск 2017

АННОТАЦИЯ

Иванов А.С. Разработка расширяемого игрового приложения «Push Him All». – Челябинск: ЮУрГУ, **КЭ-445**, 73 с.,42 илл., библиогр. список –18 наим., 1 прил.

В рамках выпускной квалификационной работы разработчик знакомится с отраслью GameDev. GameDev (произносится «*геймдев*») — сокращение от Game Development (разработка игр).

Разработка игры — процесс создания компьютерной программы с целью развлечь ее пользователя (игрока). В этот процесс входит разработка дизайна игрового процесса (геймплея), программирование игровых библиотек или использование готовых решений, разработка визуального концепта и его составляющих, создание графики и моделирование трёхмерных или двухмерных объектов, создание музыкального и звукового сопровождения.

Цель: Разработка мультимедийного программного продукта (игры) с целью развития моторики, реакции и развлечения игрока, используя среду разработки Qt и язык программирования C++.

В результате выпускной квалификационной работы была создана игра «Push Him All» в жанре аркада, особенность которой заключается в «гибких» настройках: настройка моделей, звукового сопровождения, сложности и логики врагов.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

IT-специалист широкое понятие, объединяющее в себе представителей многих профессий, работающих в области информационных технологий. Это программисты, разработчики, администраторы сетей и баз данных, модераторы, специалисты по робототехнике, по информационной безопасности, web-дизайнеры и 3D-аниматоры. При этом, с проникновением информационных технологий во всё новые сферы деятельности, появляются новые профессии для IT-специалистов. По данным на сегодняшний день и мнению многих аналитиков [1] специалисты данной области являются востребованными и будут востребованы в ближайшем будущем. В частности, сегодня в мире появляются много перспективных игровых разработок и проектов.

GameDev (произносится «*геймдев*») ─ сокращение от Game Development (разработка игр). В процесс разработки игрового приложения входит: разработка дизайна игрового процесса (геймплея), программирование игрового движка или использование готовых решений, разработка визуального концепта и его составляющих, создание графики и моделирование объектов, создание музыкального и звукового сопровождения.

Игровая индустрия продолжает развиваться и с каждым годом ее прибыль на рынке увеличивается, как в мире, так и, в частности, в России. Таким образом, идея создания интересной и новой игры является перспективным направлением.

Цели и задачи

Целью работы является разработка игры «Push Him All» с гибкими настройками, в жанре «Аркада» на платформе персонального компьютера под управлением операционной среды Windows. Жанр «Аркада» был выбран с целью развития моторики и реакции пользователя.

Для достижения данной цели должны быть решены следующие задачи:

1) осуществить постановку игровой задачи;

2) выполнить обзор аналогов и программных средств разработки компьютерной игры;

3) спроектировать приложение;

4) реализовать и составить документацию (руководство пользователя) на приложение.

Структура и объем работы

Работа состоит из введения, четырех глав, заключения и библиографического списка. Объем работы составляет 70 страниц, объем библиографии – 18 источников.

Первая глава содержит анализ и обоснование выбора индустрии, платформы, операционной среды и других технических составляющих, а также обзор программных средств разработки компьютерных игр.

Во второй главе проводится обзор игровых аналогов. Проводится их анализ и составляется таблица особенностей каждого из программных продуктов. Выявляются их недостатки и формируется вывод.

В третьей главе представлено описание игровой задачи. Определение функциональных требований к разрабатываемой видеоигре. Представляются диаграммы вариантов использования, краткое описание реализации основных технических составляющих приложения, а также файловая структура приложения.

Четвертая глава посвящена демонстрации разработанного игрового приложения. Содержит в себе краткое руководство пользователя.

В заключении кратко приведено описание проделанной работы.

Приложение содержит в себе код программы главного меню (исходный, заголовочный и файл формы).

РАЗРАБОТКА МУЛЬТИМЕДИЙНОГО ИГРОВОГО ПРИЛОЖЕНИЯ

В результате выпускной квалификационной работы требуется создать игру в жанре «Аркада» и «ТаймКиллер». Игра должна быть сделана в 2D-стиле. При запуске программы игрок имеет определенное количество здоровья и нулевой счетчик очков. Игрок с помощью выстрелов (атаки) должен уничтожать появляющихся врагов и не допускать их передвижение к нижней границе экрана. Если враг дойдет до границы или заденет игрока, то здоровье игрока уменьшится. При уничтожение врага (попаданием в него пули) счетчик очков увеличивается. Когда здоровье игрока упадет до определённого уровня (нуля) – игра закончится. В дальнейшем пользователь может запустить игру снова.

Особенностью игры является то, что пользователь может менять текстуры игрока, врагов и пуль. Запуская игру, пользователь может перейти в меню настроек игры и явно указать текстуры (картинки) моделей врагов и игрока. Если же пользователь не зайдет в меню настроек или не укажет свои модели, то будут применены стандартные параметры (модели).

Основные положения игры:

- главная цель игры – набрать наибольшее кол-во очков;
- в начале игры игрок имеет показатель здоровья равный 3;
- если враг достигнет нижней границы игры показатель очков уменьшится на единицу;
- при достижения показателя здоровья в значение 0 – игра заканчивается;
- чтобы увеличить показатель очков на единицу игроку нужно уничтожить одну вражескую единицу;
- для уничтожение врага игрок должен использовать пулю;
- враг постоянно стремится достигнуть нижней границы игры.

1 ОБОСНОВАНИЕ ВЫБОРА ТЕХНОЛОГИИ И ОПЕРАЦИОННОЙ СРЕДЫ

1.1 Игровая индустрия

На сегодняшний день игровая индустрия [2] — это один из самых знаменитых сегментов цифрового контента в мире. Данная индустрия постоянно развивается и растет (рис 1.1.1). Так по итогам 2016 года – мировой игровой рынок составил около 100 миллиардов долларов.

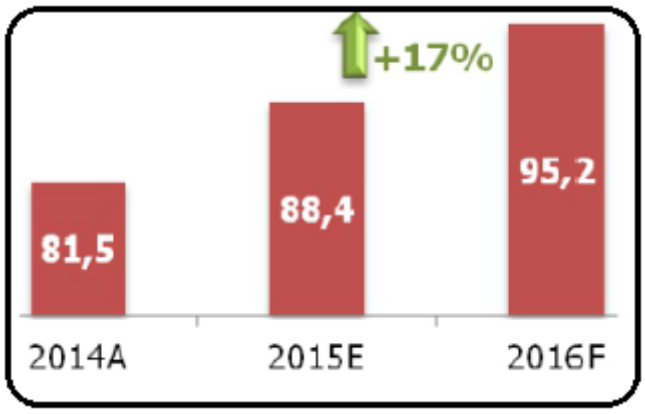


Рисунок 1.1.1 – Мировой рынок игр в 2014-2017 годах, млрд. долл.

По результатам 2015-2016 года, игровой рынок в России является значимым и развивающимся направлением. При этом, в России объем игрового рынка продолжает увеличиваться с 2010 года (рис 1.1.2) [3]. Рынок игр в России до сих пор является инвестиционно привлекательным.

Рисунок 1.1.2 – Динамика роста рынка игра в России с 2010 по 2016 год

Российский рынок только недавно стал развиваться в данном направление. Доля России в мировом рынке игр составила около 2.4% (рис 1.1.3), однако стабильный рост Российского игрового рынка привлекает западные игровые компании.

Рисунок 1.1.3 – Доля Российского игрового рынка на 2013 год

1.2 Выбор платформы

На сегодняшний день можно выделить три игровые платформы:

- персональные компьютеры;
- мобильные устройства;
- консоли.

Зарождение игр началось с появления компьютера, поэтому пользователи ПК являются основной игровой аудиторией. Так, по данным на 2015 год [4], больше половины игрового рынка принадлежит платформе персональных компьютеров (рис 1.2.1).

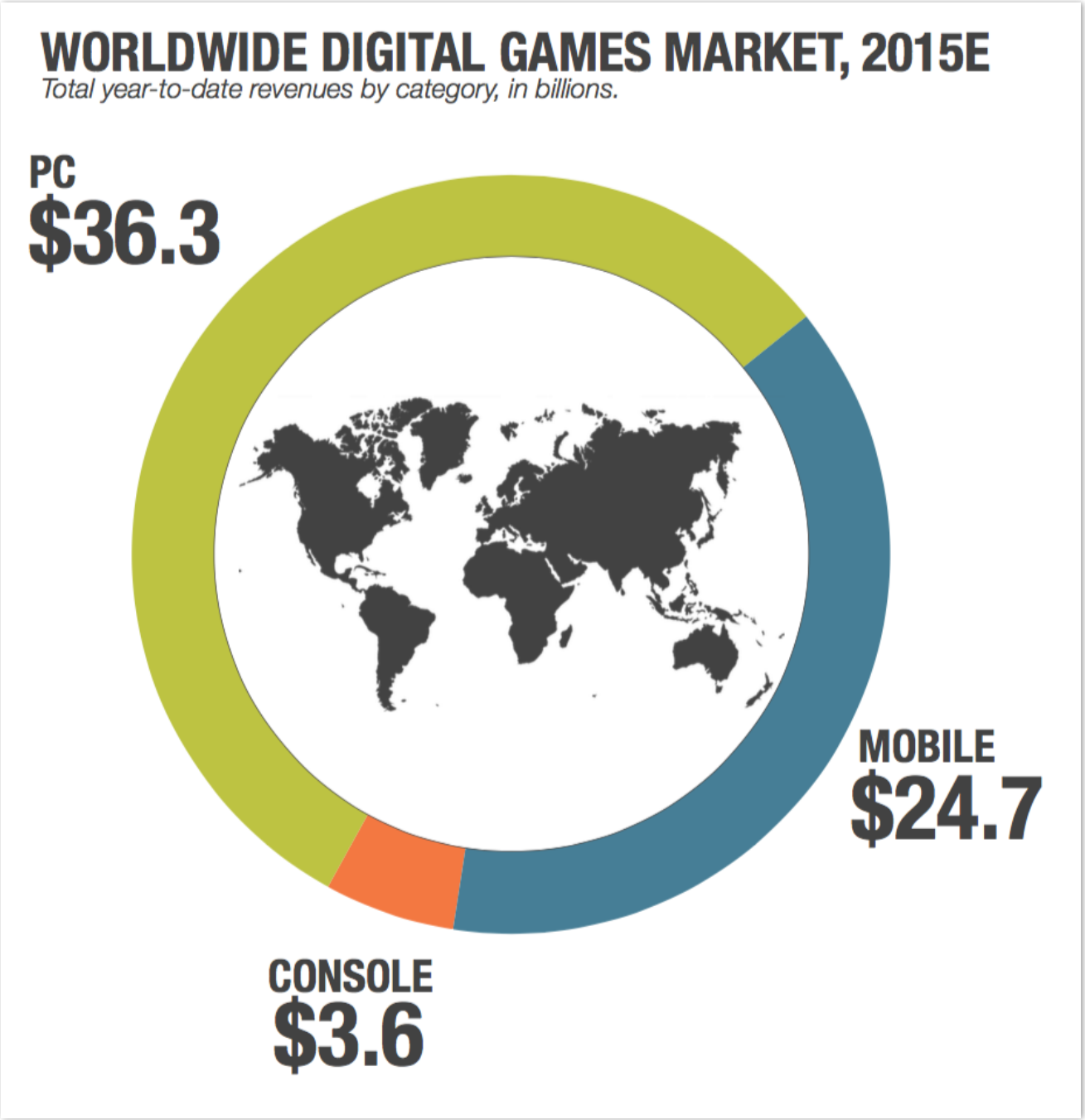


Рисунок 1.2.1 – Игровой рынок на разных платформах в 2015 году

При анализе рынка персональных компьютеров был произведен приблизительный подсчет величины операционных систем, используемых на них [5]. Результаты анализа представлены на рисунке 1.2.2.

Из рисунка 1.2.2 видно, что большинство пользователей персональных компьютеров предпочитают операционную систему семейства Windows.

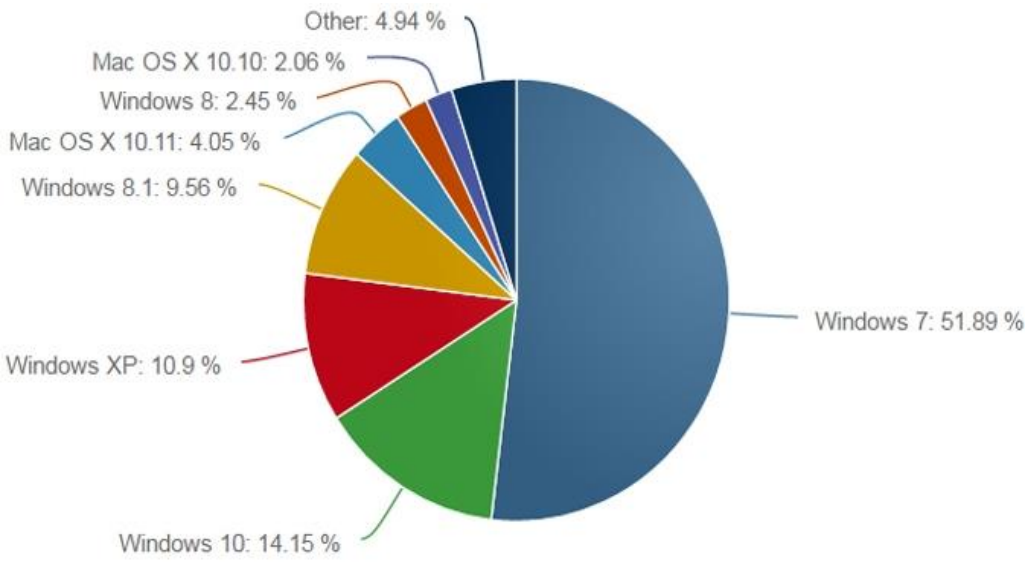


Рисунок 1.2.2 – Используемые операционные системы

1.3 Выбор жанра игры

Сегодня в мире преимущественно выделяют следующие жанры игр:

- азартные
- спортивные
- головоломки
- стратегии
- боевики
- аркады

По данным компании «Mail.Ru» многие пользователи предпочитают жанр «Аркада» [6]. Аркады являются достаточно популярными относительно других жанров по следующим причинам (рис 1.3):

1. Аркады не требуют мощных технических характеристик

Не все пользователи имеют возможность приобретать современные и мощные устройства с соответствующими характеристиками. Многие пользователи отмечают, что чем мощнее игра, тем больше она тормозит. Таким образом, игровой процесс является достаточно затруднительным и неприятным пользователю, что отбивает желание играть и, тем самым, уменьшает рейтинг приложение, что в свою очередь уменьшает заинтересованность новых пользователей (игроков)

устанавливать данное приложение. Однако аркады не требуют серьёзных технических характеристик, что позволяет большинству пользователей наслаждаться игровым процессом без задержек, зависаний и других проблем, связанных с производительностью.

2. Аркады не занимают много времени

Все пользователи игровых приложений (игроки) занимаются какой-либо деятельностью (работают, учатся и т.д.). Таким образом, у игроков не всегда есть свободное время, чтобы проводить его за играми. Аркады созданы для людей у которых нет возможности проводить за игрой достаточное количество времени. Обычно данный жанр позволяет игроку занять его время на какой-то период. Тем самым, большинство пользователей называют аркады ─ «ТаймКиллерами», так как они являются интересными и незаменимыми помощниками весело провести период времени.

3. Аркады интересные

Многие пользователи отмечают, что жанр аркада разносторонний. Он может включать в себя логические загадки, красивую графику или интересный игровой процесс (геймплей), т.е. не позволяют игроку заскучать.

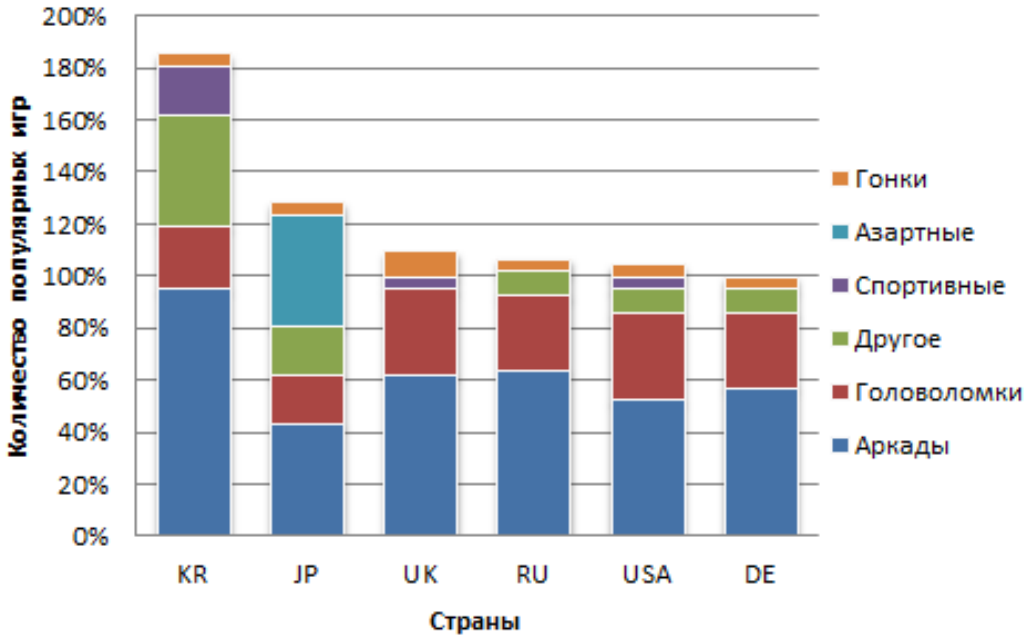


Рисунок 1.3 – Популярность игровых жанров

Таким образом, в качестве жанра игры «PushHimAll» было решено выбрать жанр «Аркада».

1.4 Выбор программной составляющей приложения

В настоящее время существует множество программного обеспечения, позволяющего разрабатывать собственные проекты различного характера, в том числе и компьютерных игровых приложений. Выделяют две основные группы из них:

- использование программного компонента (комплекса), позволяющего создавать игровое приложение (игровой движок);
- создание собственного «движка» с помощью языка программирования и специализированных библиотек.

Рассмотрим каждую из групп поподробнее:

1) игровые «движки»;

Unity 3D [7] ─ это мощный инструмент для создания приложений и игр под разные платформы. Имеется возможность создавать как 2D, так и 3D проекты. Имеется поддержка DirectX и OpenGL, поддерживает множество различных форматов данных. Unity 3D поддерживает такие языки программирования как: C#, JavaScript. Основное рабочее окно Unity 3D представлено на рисунке 1.4.1.

Достоинства:

- не самые большие затраты ресурсов (по сравнению с другими «движками»);
- кроссплатформенность;
- большое русскоязычное общество;
- регулярные обновления.

Недостатки:

- обладает малым набор инструментов;
- требует приличные технические комплектующие;

- платная коммерческая лицензия;
- неудобная работа при создании 2D – приложения.

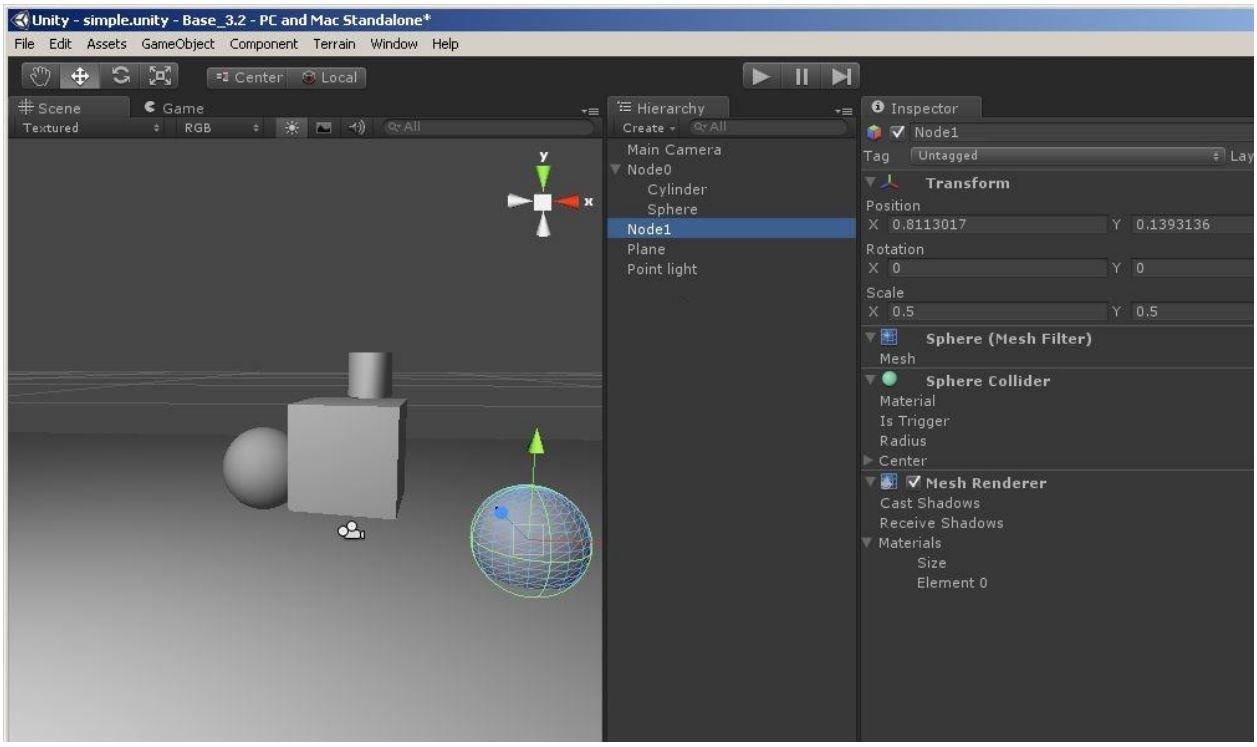


Рисунок 1.4.1 – Рабочее окно игрового движка Unity 3D

CryEngine [8] — самый мощный из современных игровых движков, обеспечивающий фотореалистичную графику с поддержкой DirectX 12 и шейдеров (теней). Третья версия «движка» создана в 2009 году, распространяется платно. Пятая версия «движка» вышла в конце 2016 года и имеет условно бесплатную лицензию. Основным направлением данного средства является создание игр жанра «Шутер». Рабочее окно CryEngine представлено на рисунке 1.4.2.

Достоинства:

- высокие показатели графики;
- кроссплатформенность.

Недостатки:

- обладает малым набор инструментов;
- направлен на создание определённого жанра игр;
- самый ресурсоемкий «движок» из всех представленных.

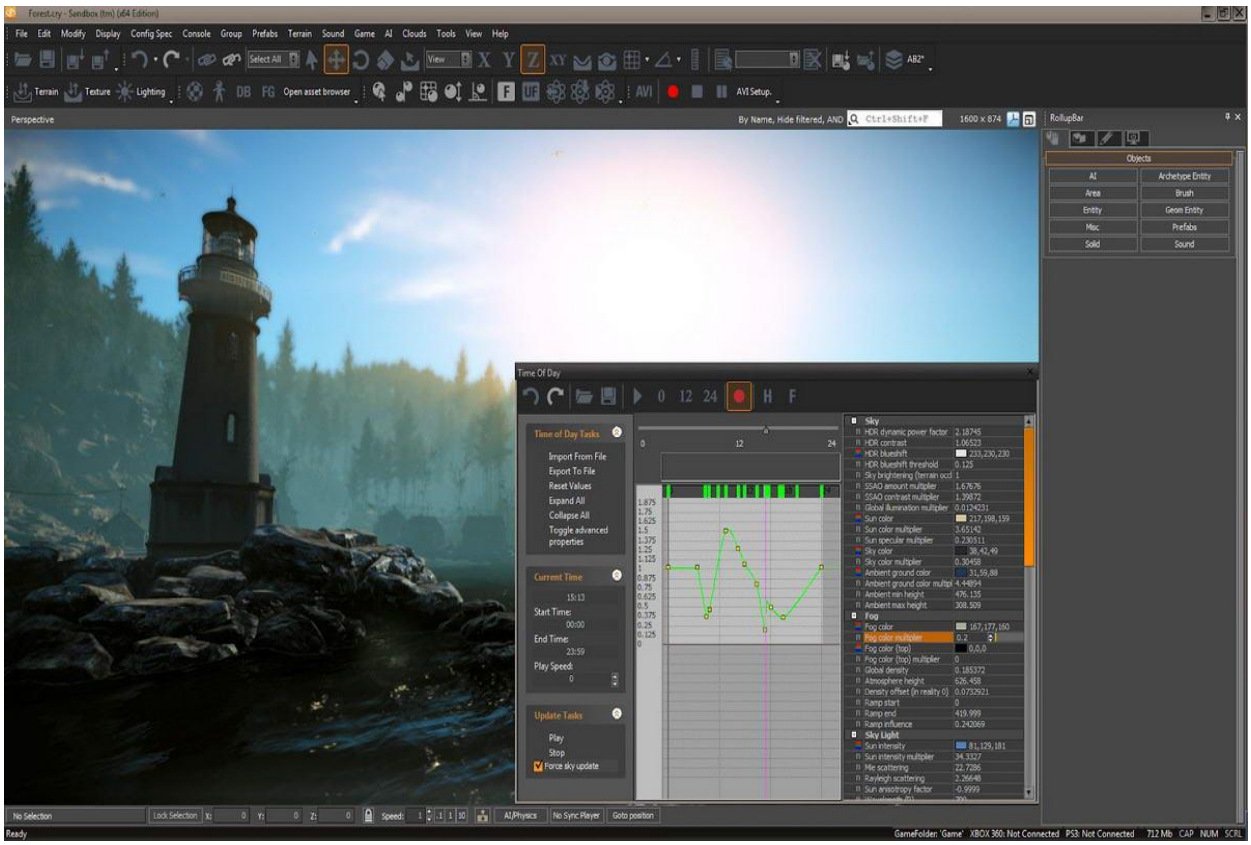


Рисунок 1.4.2 – Рабочее окно игрового движка CryEngine

Unreal Engine [9] ─ игровой «движок», разрабатываемый и поддерживаемый компанией Epic Games. Предоставляет большой набор инструментария для создания 2D и 3D проектов. Начиная с четвертой версии ─ распространяется бесплатно. Однако до тех пор, пока разработчик не выпустит свой первый коммерческий продукт на основе UE4. Является основным движком большинства современных игр. Рабочее окно игрового «движка» Unreal Enginge представлено на рисунке 1.4.3.

Достоинства:

- высокие показатели графики;
- кроссплатформенность;
- большие и регулярные обновления;
- разнообразный инструментарий.

Недостатки:

- высокие технические требования;
- сложность в освоение;
- малое количество русскоязычной документации;
- коммерческая лицензия.

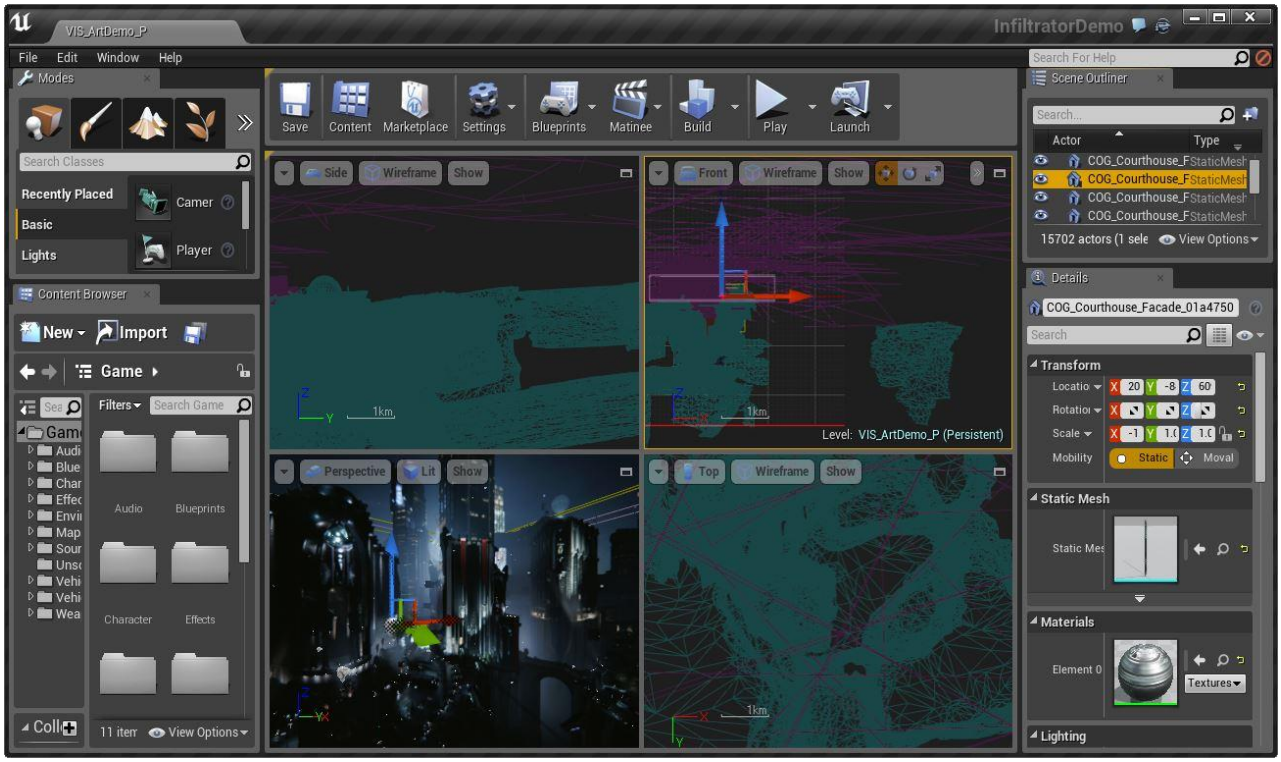


Рисунок 1.4.3 – Рабочее окно игрового движка Unreal Engine

Все рассмотренные игровые «движки» в основном предназначены для создания 3D приложений и требуют высокие показатели технических требований, поэтому использовать их нецелесообразно (у многих пользователей ПК отсутствуют современные и мощные комплектующие). Перейдем к рассмотрению другой группы программного обеспечения для создания игрового приложения.

2) библиотеки для языков программирования (API).

Для начала стоит определиться какой язык программирования использовать [10]. На сегодняшний день выделяют следующие языки программирования в качестве основы последующей разработки:

1. C#

C# позволяет стартовать разработку быстрее, а это позволяет быстрее получить прототип решения. Скорость разработки на начальных этапах проекта значительно выше по сравнению с другими языками.

C# спроектирован быть кроссплатформенным, однако его развитие не пошло в этом направлении, поэтому под Windows образовалась достаточно полная .net инфраструктура, а на других платформах равноценной инфраструктуры не появилось.

При разработке небольших проектов производительность C# не уступает другим языкам программирования, однако при увеличение исходного кода, алгоритмов и т.д. ─ скорость работы приложений значительно падает.

C# обладает приличным количеством библиотек со старта, что существенно облегчает разработку.

2. HTML и JavaScript

Являются незаменимыми при создание простых веб-приложений или игр, однако на этом достоинства данных средств заканчиваются.

3. C++

Является основным языком программирования при разработке игр, так как позволяет осуществить полный контроль над средствами и логикой. Является абсолютным рекордсменом по производительности, по кроссплатформенности, по совместимости и по кол-ву существующих библиотек среди прочих языков программирования. Однако многие программисты выделяют главный его недостаток ─ сложность освоения данного языка.

Сведем полученные данные в таблицу 1.3.

Сравнение языков программирования Таблица 1.3

Критерий	C#	JavaScript/HTML	C++
Производительность	★★★☆☆	★★☆☆☆	★★★★★ (максимальная)
Кол-во библиотек	★★★☆☆	★★☆☆☆	★★★★★
Удобство программирования	★★★★☆ (легкая)	★★★☆☆	★★★☆☆
Сложность освоения	★★★★☆ (легкая)	★★★★☆	★★★☆☆
Кроссплатформенность	★★★☆☆	★★★★★ (полная)	★★★★☆

Каждый язык программирования хорошо себя показывают в определённых задачах. Например: веб-разработка (JavaScript) не имеет привязанности к операционной системе; если в приоритете разработки стоит скорость (получение прототипа приложения), то следует использоваться C#. В качестве языка программирования для создания игр было решено использовать язык C++, так как он поддерживает огромное кол-во библиотек и обладает хорошей производительностью.

На языке C++ существуют два основных API для создания графики:

1) SFML [11]

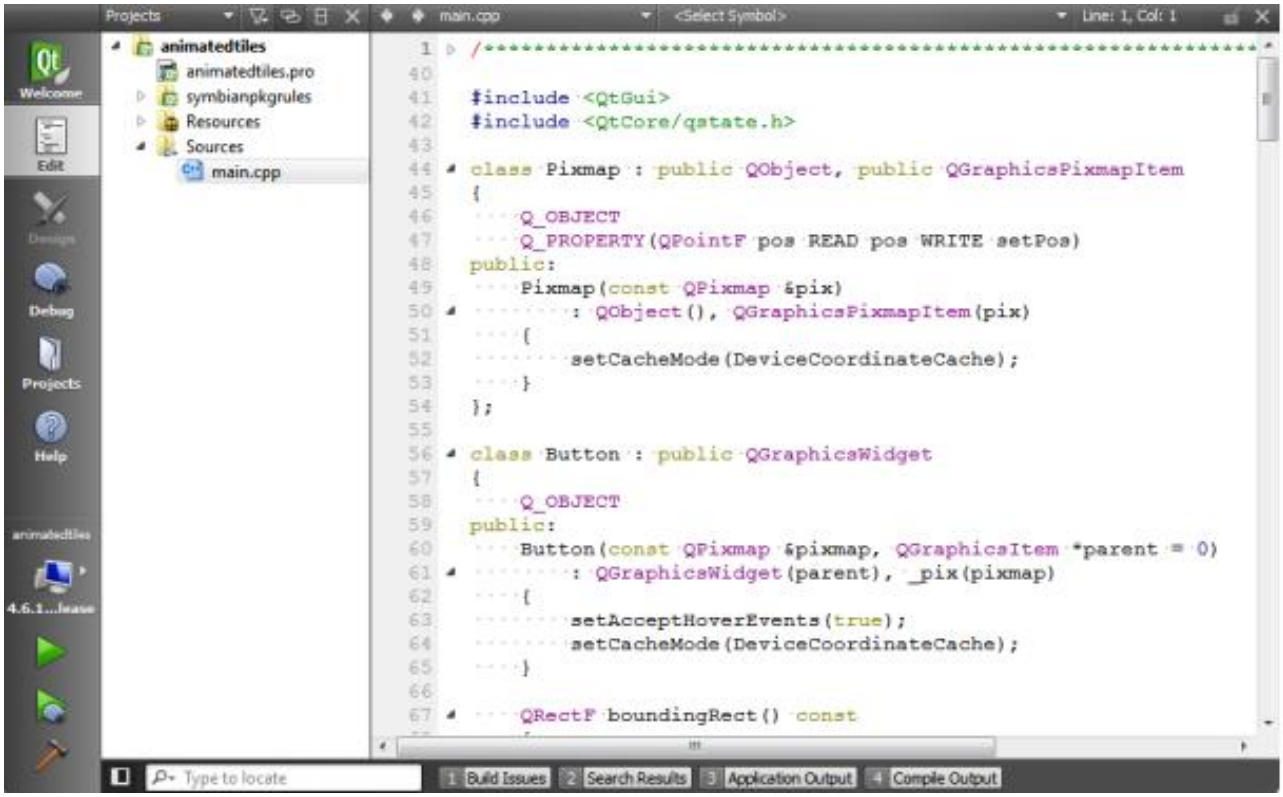
Простая и кроссплатформенная мультимедийная библиотека. Включается в себя модули для программирования игр и мультимедийных средств. Состоит из пяти модулей:

1. System — модуль управляет временем и потокам. Является базовым, то есть требуется для всех модулей
2. Window — управление окнами и вывод информации пользователю.
3. Graphics — производит вывод информации (изображений, геометрических фигур). Для использования нужно подключить модуль Window.
4. Audio — библиотека для работы со звуком.
5. Network — библиотека для работы с сетью.

2) Qt Graphics [12]

Более мощный инструмент (по сравнении с SFML). Входит в состав библиотеки Qt, которая представляет из себя инструментарий разработки ПО на языке программирования C++ и обладает кроссплатформенностью. Qt предоставляет программисту не только удобный набор библиотек классов, но и определённую модель разработки приложений, определённый каркас их структуры, существенно снижающий появление ошибок в приложение (утечки памяти, незакрытые файлы и т.д.). Библиотека Qt обладает собственной IDE (средой разработки) ─ Qt Creator, что существенно упрощает разработку приложений. Оболочка Qt Creator представлена на рисунке 2.4.4.

Существует способ объединить Qt и SFML, однако, в качестве средства разработки, был выбран Qt, так как он обладает более широким функционалом.




```
/****************************************************************************

#include <QtGui>
#include <QtCore/qstate.h>

class Pixmap : public QObject, public QGraphicsPixmapItem
{
    Q_OBJECT
    Q_PROPERTY(QPointF pos READ pos WRITE setPos)
public:
    Pixmap(const QPixmap &pix)
        : QObject(), QGraphicsPixmapItem(pix)
    {
        setCacheMode(DeviceCoordinateCache);
    }
};

class Button : public QGraphicsWidget
{
    Q_OBJECT
public:
    Button(const QPixmap &pixmap, QGraphicsItem *parent = 0)
        : QGraphicsWidget(parent), _pix(pixmap)
    {
        setAcceptHoverEvents(true);
        setCacheMode(DeviceCoordinateCache);
    }

    QRectF boundingRect() const
```

Рисунок 2.4.4 – IDE Qt Creator

1.5 Выводы по разделу

Таким образом, можно вывести основные требования к приложению «Push Him All»:

1. платформа создания ─ Персональный компьютер;
2. операционная среда ─ Windows 7 и выше;
3. жанр игры ─ аркада;
4. используемый язык программирования ─ C++;
5. используемая API ─ Qt;
6. IDE разработки ─ Qt Creator.

2 ОБЗОР АНАЛОГОВ

На сегодняшний день существует множество игровых приложений жанра аркада. Рассмотрим самые распространённые из них.

2.1 Space Invaders [13]

Самой популярной аркадой является Space Invaders (рис. 2.1.1), которая была одна из первых в этом жанре. Цель игры ─ уничтожить всех врагов на экране. Изначально игра имела простую графику и показатель здоровья, при достижение нулевого значение которого ─ игра завершалась, однако позже она усовершенствовалась (были введены новые цветовые схемы и показатель очков) (рис. 2.1.2).

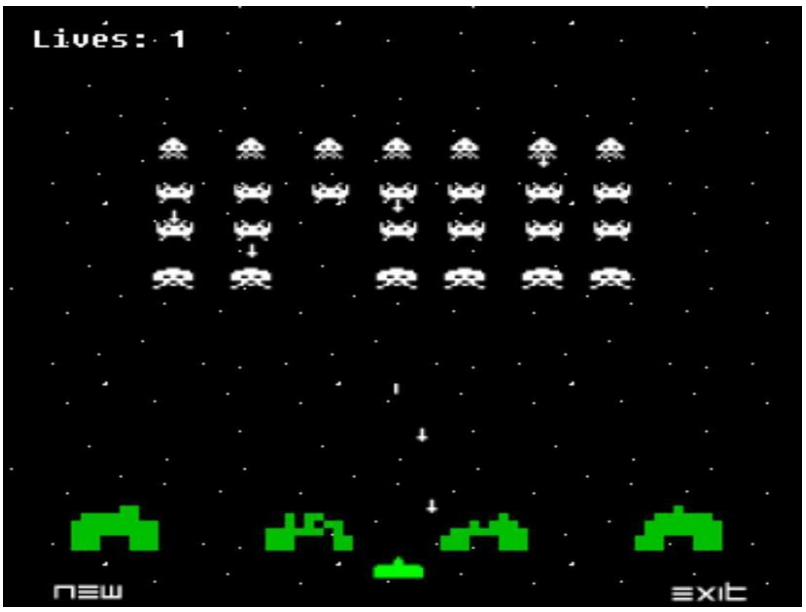


Рисунок 2.1.1 – Скриншот игры Space Invaders

Недостатки:

- пользователь не имеет возможности изменять игровые модели;
- пользователь не имеет возможность изменять логику работы игры (изменение сложности уровня, изменение траектории передвижения врагов);
- в первой версии игры отсутствовал показатель здоровья;
- игра имеет примитивные модели.

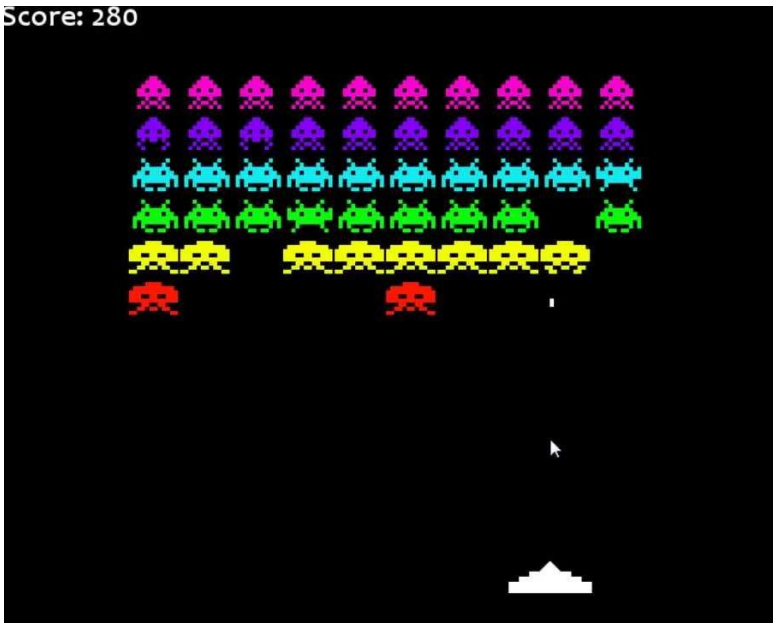


Рисунок 2.1.2 – Скриншот улучшенной Space Invaders

2.2 Radiant [14]

Аналог игры Space Invaders выпущенный в 2010 году. Цель игры – уничтожить всех врагов на экране. Главная особенность игры ─ улучшенная 2D графика. По мере увеличение времени игры увеличивается и сложность ─ появляются больше врагов, однако, у пользователя нет возможности изменять модели игры.

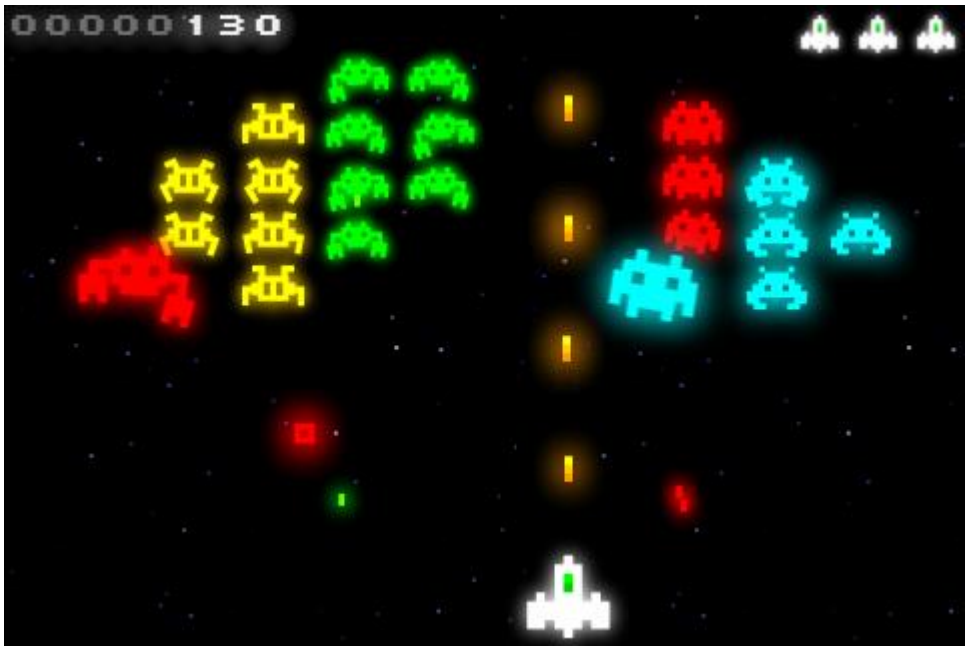


Рисунок 2.2 – Скриншот игры Radiant

Достоинства:

- улучшенная 2D-графика;
- присутствует нарастание сложности по мере прохождения.

Недостатки:

- в игре присутствуют только стандартные однотипные модели;
- сложность присутствует, однако, у пользователя нет возможности выбрать ее самому;
- отсутствует возможность выбрать звуковое сопровождение.

2.3 Luxor [15]

«Luxor» ─ серия аркадных компьютерных игр, разработанных компанией MumboJumbo и выполненные в 2D – стиле, базирующиеся на мифологии Древнего Египта. Задача игрока — истребить появляющихся на экране сферических объектов, до того, как они достигнут пирамиды.

В игре присутствует адаптивная сложность (сложность увеличивается с уровнем), однако, пользователь не имеет возможности изменять шары (игровые объекты) на определённый цвет или другие фигуры.

Рисунок 2.3 – Скриншот игры «Luxor»

Достоинства:

- присутствуют элементы головоломок (игра развивает логику);
- присутствует нарастание сложности по мере прохождения.

Недостатки:

- в игре есть только шарообразные модели. У пользователя нет возможности заменить их;
- сложность присутствует, однако, у пользователя нет возможности выбрать ее самому.

2.4 Beyond Space [16]

«Beyond Space» ─ космическая аркада для Android, выпущенная в 2015 году. Главная особенность игры ─ 3D-графика. Главная цель игры ─ уничтожить другие корабли. Присутствует возможность кастомизировать корабль (изменять цвет корабля). В игре присутствуют миссии разного уровня сложности ─ пользователь переходит к от легкого режима к сложному.

Рисунок 2.4 – Скриншот игры «Beyond Space»

Достоинства:

- красочная 3D-графика;
- присутствует частичное изменение модели корабля (кастомизация).

Недостатки:

- игра требует мощные технические характеристики;
- игра не оптимизирована (присутствуют проблемы с производительностью);
- сложность присутствует, но у пользователя отсутствует возможность выбрать определённый ее уровень для каждой миссии.

2.5 Чужой в космосе [17]

«Чужой в космосе» ─ это игра-аркада, выпущенная в 2013 году. Храбрый космодесантник должен провести ученых в безопасные бункеры, пока до них не добрались местные жуки переростки. Для выполнения этой не простой задачи игроку позволено пользоваться всевозможным оружием: бензопилой, ракетами, пулями.

Игра обладает более продуманной графикой (по сравнению с первыми тремя играми), а также имеет систему улучшения оружия (на модель игрока добавляются новые элементы, а также изменяется модель пули).

Достоинства:

- продуманная 2D-графика;
- присутствует частичное изменение модели корабля и оружия (кастомизация);
- присутствует нарастание сложности по мере прохождения.

Недостатки:

- сложность присутствует, однако, у пользователя нет возможности задать ее на определенном уровне;
- отсутствует возможность выбрать звуковое сопровождение в игре.

- в игре нельзя выбрать определённые модели врага.

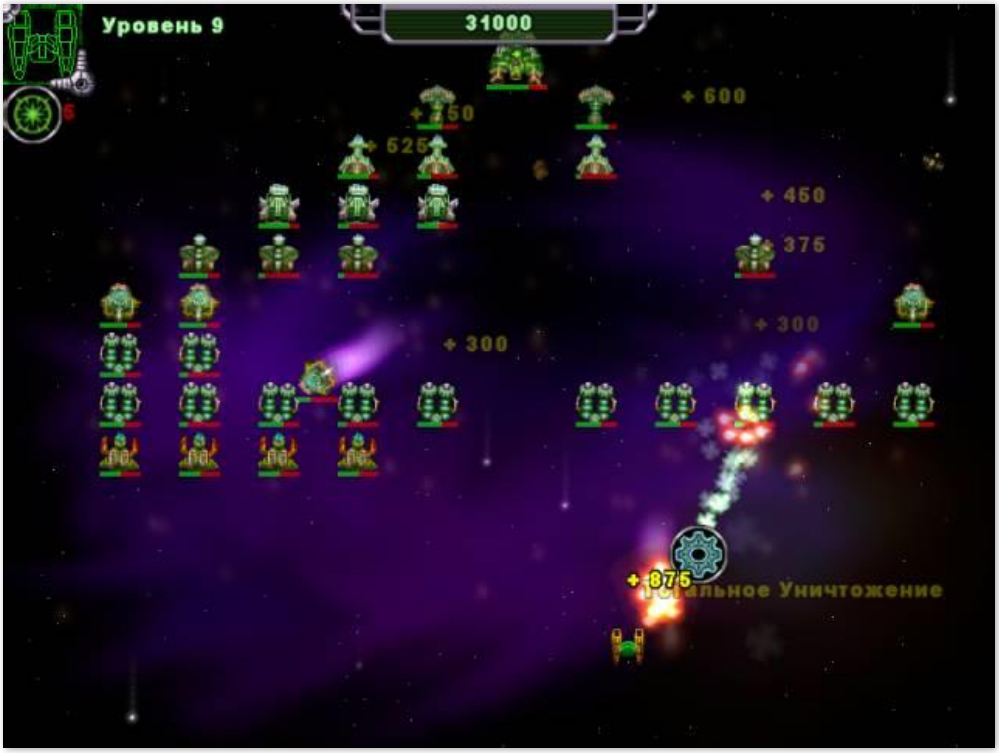


Рисунок 2.5 – Скриншот игры «Чужой в космосе»

2.6 Выводы по разделу

В результате обзора выше перечисленных игровых решений были выделены их функциональные составляющие и вынесены в таблицу 2.6. При анализе рассмотренных игр были выделены следующие особенности жанра:

- аркады включают в себя либо набор миссий, либо бесконечный режим игры;
- основная цель большинства аркад – набрать наибольшее количество очков;
- чтобы увеличить показатель очков нужно уничтожить модель врага с помощью функции выпуска пули (атаки);
- почти во всех аркадах есть показатель здоровья, который уменьшается либо по достижению модели врага какой-то определённой местности, либо при достижение самого игрока.

Сравнение особенностей игр жанра «Аркада» Таблица 2.6

Название игры	Графика	Сюжет	Возможность изменять модели	Возможность изменять поведение врагов
Space Invaders	2D	Бесконечный режим	-	-
Radiant	2D	Бесконечный режим	-	-/+ Присутствует увеличение сложности
Luxor	2D	Набор миссий	-	-/+ Присутствует увеличение сложности
Beyound Space	3D	Набор миссий	-/+ Добавление новых модулей	-/+ Присутствует увеличение сложности
Чужой в космосе	2D	Бесконечный режим	-/+ Добавление новых модулей	-

Из таблицы 2.6 видно, что большинство аркад не предоставляют пользователю возможность изменять поведение врагов или явно задавать уровень сложности, а также изменять модели в игре.

3 РАЗРАБОТКА ПРОГРАММНОЙ СИСТЕМЫ

Разработка игры состояла из четырех этапов:

3.1 Разработка концептуальной схемы

3.1.1 Формирование целей и задачи проекта

На этом этапе продумывается функционал и возможности. Формируется жанр и другие особенности игры. В качестве жанра игры было выбрано направление «Аркада», т.к. они достаточно распространены, не требует большого времени игры (геймплея) и позволяют игрокам скоротать время (ТаймКиллеры). Таким образом, на данном этапе для игры «PushHimAll» были сформированы следующие утверждения:

- игра будет представлять из себя Арконойд (жанр «Аркада»);
- в игре присутствует игрок, враги, оружие (пули);
- в игре учтена система очков и здоровья;
- по достижение нулевого показателя здоровья ─ игра закончится;
- цель игры ─ набрать наибольшее кол-во очков.

При анализе других аркад были выявлены следующие суждения:

- почти во всех аркадах присутствует адаптивное наращивание сложности по мере прохождения, однако, явное задание сложности отсутствует;
- в ранее рассмотренных проектах отсутствует возможность изменять логику поведения врагов (направление движения);
- в некоторых играх присутствует частичное изменение моделей игры, но, чаще всего, эти изменения незначительны.
- во всех рассмотренных экземплярах отсутствует возможность выбора звукового сопровождения.

Таким образом, в качестве особенностей проекта «Push Him All» будут выступать следующие функции:

1) в игре будет возможность изменять стандартные модели на пользовательские. Таким образом пользователь (игрок) сможет изменять:

- модель игрока;
- модель врага;
- модель пули;
- задний фон (BackGround).

2) в игре будет возможность изменять поведение врагов (направление движения).

3) в игре будет возможность изменять уровень сложности;

4) в игре будет присутствовать возможность изменить звуковое и музыкальное сопровождение.

3.1.2 Разработка UML- диаграмм

Для проектирования приложения был использован язык графического описания объектного моделирования UML [18], так как он позволяет продумать и описать архитектуру проекта без программного кода, при этом не требует особенных знаний для их понимания. В качестве основного средства описания проекта были выбраны и построены диаграммы использования, которые показывают функциональность будущей системы. Варианты использования показывают возможные взаимоотношения между системой и пользователем. Они отображают внешний интерфейс системы и указывают форму того, что система должна сделать. Действующее лицо (актер) – это элемент, который не является системой, а взаимодействует с ней, через особый инструмент – вариант В ходе проектирования был выделен следующий актер:

Игрок ─ пользователь приложения, обладающий полным доступом к функциям видеоигры. Может участвовать в игре, управлять настройками приложения.

Общая диаграмма использования представлена на рисунке 3.1.2.1

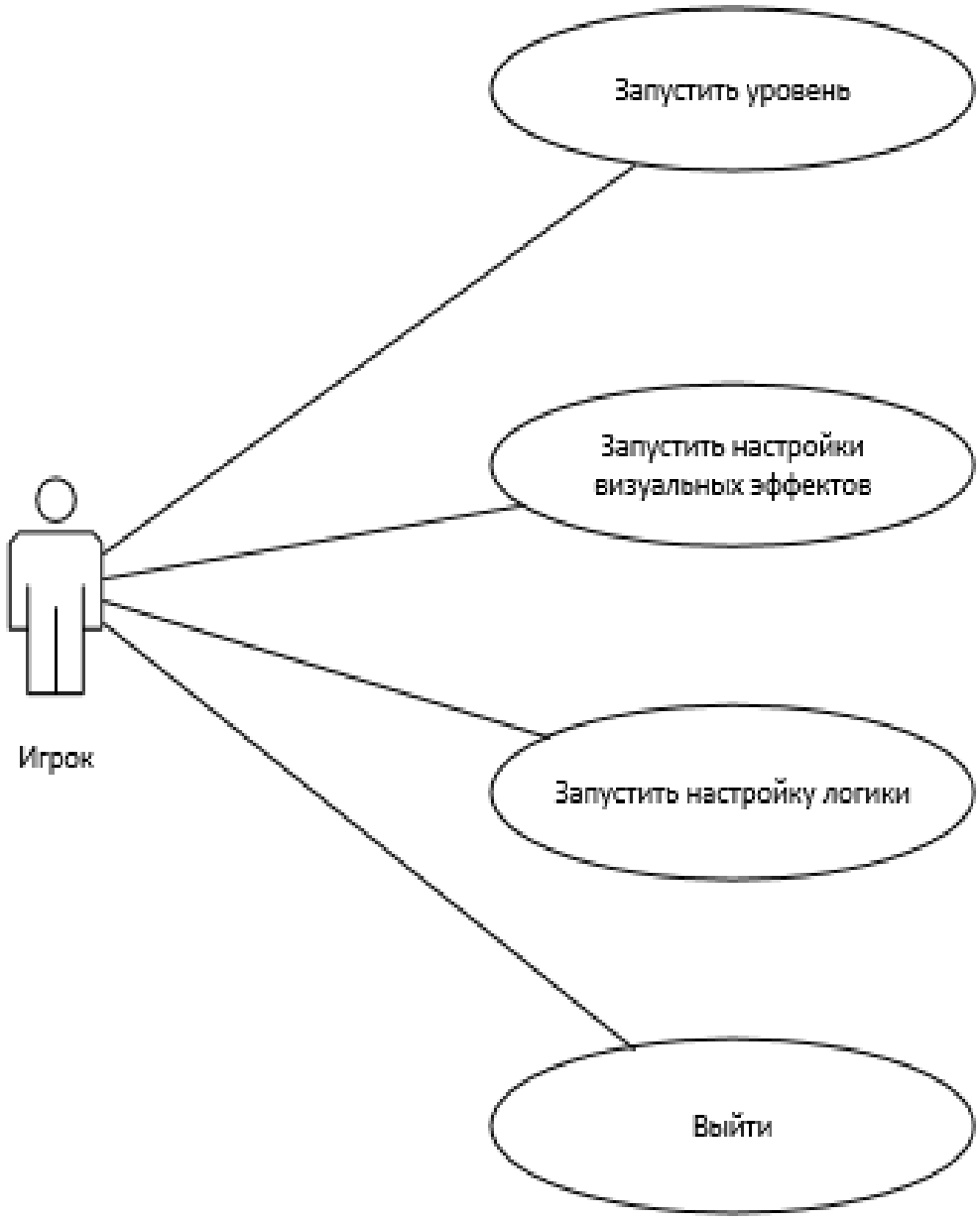


Рисунок 3.1.2.1 ─ Общая UML диаграмма использования

Распишем каждый вариант использования поподробнее. Для этого надо представить диаграммы отношений вариантов использования. При запуске уровня пользователем, он может проделать определенные действия, которые представлены на рисунке 3.1.2.2.

Рисунок 3.1.2.2 – Диаграмма отношения вариантов использования для действия «Запустить уровень»

В качестве другого варианта пользователь может использовать действие «Запустить настройки визуальных эффектов». В данном варианте использования у действующего лица (игрока) есть возможность настроить визуальные компоненты приожения. При настройки некоторых компонентов необязательно настраивать некоторые функции, которые помечены отношением «расширить». Диаграмма отношений варианта использования «Запустить настройки визуальных эффектов» представлена на рисунке 3.1.2.3.

Рисунок 3.1.2.3 - Диаграмма отношения вариантов использования для действия «Запустить настройки визуальных эффектов»

В варианте использования «Запустить настройку логики», пользователю предоставляется возможность менять сложность игры, а также настраивать траекторию передвижение врагов и действие «Woopsy».

Woopsy ─ специальная функция, вызывающая указанную игроком картинку по достижение определенного действия (по достижению определенного кол-во очков или через некоторое время, указанное пользователем). Диаграмма отношений варианта использования «Запустить настройку логики» представлена на рисунке 3.1.2.4.

Рисунок 3.1.2.4 – Диаграмма отношения вариантов использования для действия «Запустить настройку логики»

3.1.3 Разработка бизнес сущностей и логики

В результате проектирования приложения «Push Him All» были выделены бизнес сущности и логика, представленные в таблице 3.1.3.

Таблица 3.1.3

Бизнес сущности		Бизнес логика	
1	Игрок	1	Калькулятор очков
2	Пуля	2	Отслеживание положения пули
3	Показатель здоровье	3	Отслеживание положения врагов
4	Показатель очков	4	Отслеживание колизий (пересечения пуль и врагов)
5	Враги	5	Интерфейс для предоставления кол-ва очков
6	Текстуры	6	Интерфейс для предоставления кол-ва здоровья
7	Музыка	7	Интерфейс главного меню
		8	Интерфейс работы с выбором текстур
		9	Интерфейс настройки логики врагов

3.2 Разработка игры с использованием стандартных объектов

3.2.1 Создание Scene и View

Класс QGraphicsScene предоставляет поверхность для управления большим числом графических 2D элементов. QGraphicsScene предоставляет функциональность, которая позволяет определять положение этих элементов, а также позволяет определять видимость компонентов внутри произвольной области сцены. Чтобы добавить какой-либо элемент на сцену используется метод addItem.

Класс QGraphicsView представляет собой виджет для отображения содержимого QGraphicsScene. Он так же позволяет отобразить всю сцену целиком или только определенную ее часть. Чтобы указать сцену, которую будет отображать View, нужно использовать метод setScene. Чтобы отобразить View используется метод show().

Класс QGraphicsRectItem позволяет создать прямоугольный элемент, который будет использоваться в качестве модели. С помощью метода setRect (x, y, width, height) устанавливается местоположение и размеры прямоугольника:

- x – координата x прямоугольника;

- y – координата у прямоугольника;
- width – ширина прямоугольника;
- height – высота прямогульника.

Взаимодействие трех выше перечисленных классов можно увидеть на рисунке 3.2.1.1. Ниже представленный код создает сцену, окно вывода (View) и прямоугольник размерами 100 на 100. Результат работы кода можно увидеть на рисунке 3.2.1.2.

```
QGraphicsScene * scene = new QGraphicsScene();
QGraphicsRectItem * rect = new QGraphicsRectItem();
rect->setRect(0,0,100,100);
scene->addItem(rect);
QGraphicsView * view = new QGraphicsView(scene);
view->show();
```

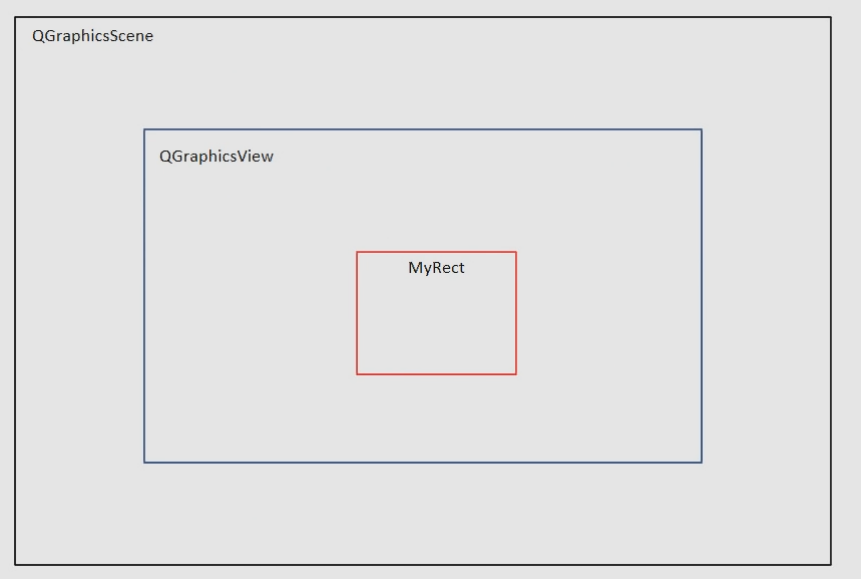


Рисунок 3.2.1.1 – Взаимодействия Scene, View, RectItem

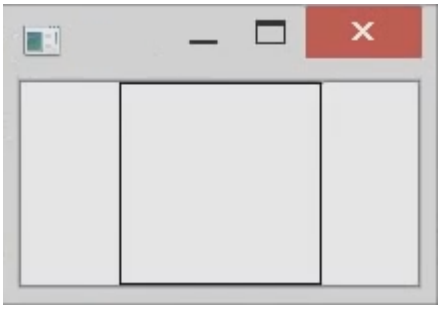


Рисунок 3.2.1.2 – Результат работы кода

3.2.2 Создание модели врагов и пули

Система координат графического представления продемонстрирована на рисунке 3.2.2. Графическое представление в библиотеках представляет собой декартовую систему координат. Таким образом, чтобы изменить положение элементов и их геометрию требуется воспользоваться двумя величинами: x-координаты и y-координаты.

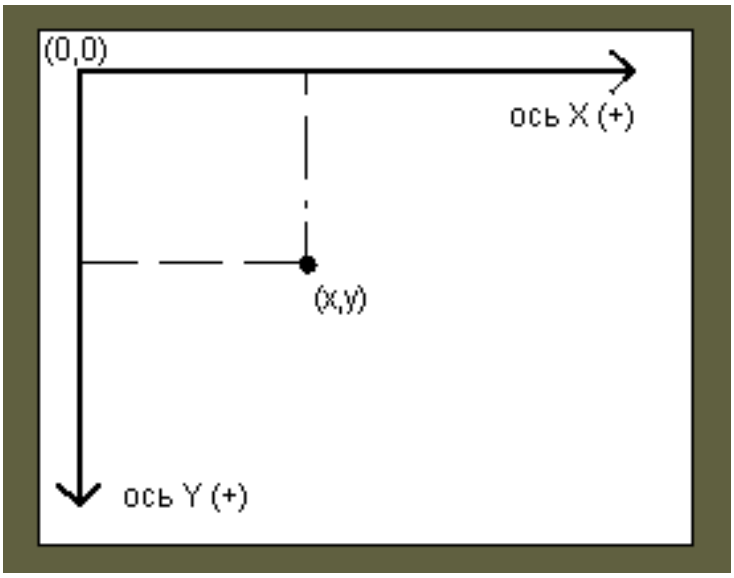


Рисунок 3.2.2 – Система координат графического представления

В качестве модели врага и пули будет выступать QGraphicsRectItem, который представляет собой прямоугольный элемент.

Враги должны появляться в верхней части экрана игры. Для определения позиции модели врага (появление слева, справа или по середине) используется функция rand(), которая генерирует местоположения модели врага перед ее появлением. Если модель врага будет передвигаться по косой траектории, следует учитывать местоположение ее появления (верхний левый/правый угол).

С помощью функцию setPos (double x, double y) – устанавливается позиция элемента, где:

- x - координата x устанавливаемого элемента;
- y – координата y устанавливаемого элемента.

3.2.3 Обработка коллизий

При взаимодействие (пересечение) модели врага и пули, обе модели должны быть удалены со сцены и памяти. Для этого используется специальный массив, который включает в себя модели представленные на сцене, а также специально созданную функция, которая проверяет пересечение моделей определенных фигур. При пересечение моделей фигуры – функция удаляет элементы из сцены и обращается к определенным (пересеченным) моделям массива и удаляет их. Таким образом, пересеченные модели теряют видимость (удаляются со сцены) и удаляются из памяти.

Рисунок 3.2.3 – Алгоритм обработки при пересечение модели врага и пули

В качестве примера на рисунке 3.2.3 представлен алгоритм логики работы при пересечение модели врага и пули.

При взаимодействие (пересечение) модели врага и игрока из сцены должна удаляться только модель врага, но при этом показатель здоровья игрока должен уменьшится.

3.2.4 Создание таймеров

Цель игры ─ уничтожить появляющихся врагов с помощью пуль. Для генерации врагов используется таймер ─ QTimer. Класс QTimer предоставляет повторяющиеся и однократные таймеры.

QTimer ─ это высокоуровневый программный интерфейс для таймеров. Для работы с ним следует:

- создать элемент QTimer;
- подключить его сигнал тайм-аута к соответствующему слоту (вызвать функцию по переполнению таймера);
- вызвать метод start ().

Для целей передвижения моделей следует создать таймер, при переполнение которого, положение пули и врага будет изменяться (враг будет передвигаться к нижней части экрана, а пуля к верхней). Ниже представлен пример создания таймера для передвижения врага.

1) Создание элемента QTimer с названием «timer»:

```
QTimer * timer = new QTimer();
```

2) Подключение таймера к слоту (функции) «move»:

```
connect(timer,SIGNAL(timeout()),this,SLOT(move()));
```

3) Включение таймера методом «start(msec)», где msec – количество миллисекунд до переполнения таймера:

```
timer->start(msec);
```

3.3 Загрузка текстур, музыки, моделей

На этой стадии все простые объекты QGraphicsRectItem (прямоугольники) заменяются на графические модели. В приложение «Push Him All» требуются следующие типы моделей:

- спрайты моделей;
- музыка и звуки;
- видео.

Файловая структура приложения представлена на рисунке 3.3.1.

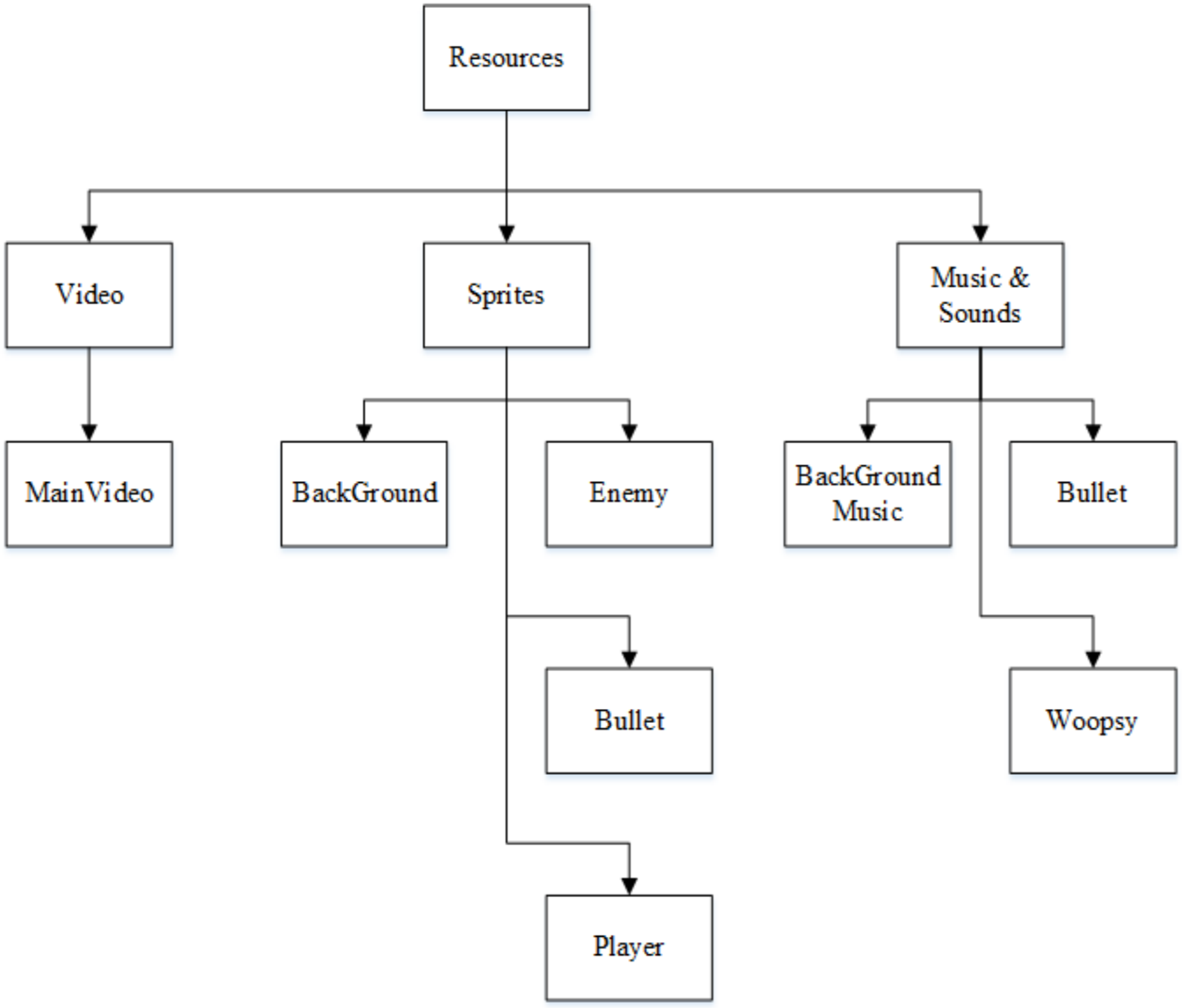


Рисунок 3.3.1 – Файловая структура приложения «Push Him All»

1) Sprites

- Enemy – стандартная модель врага. Представляет собой 2D спрайт космического корабля с корпусом красного цвета.
- Bullet – стандартная модель пули, выпускаемая игроком. В качестве спрайта пули была выбрана ракета.
- Player – стандартная модель игрока (пользователя). Представляет собой космический корабль с корпусом зеленого цвета.
- BackGround – задний фон игры. В качестве фона был выбран рисунок космоса.

Основные модели игры приведены на рисунке 3.3.2.

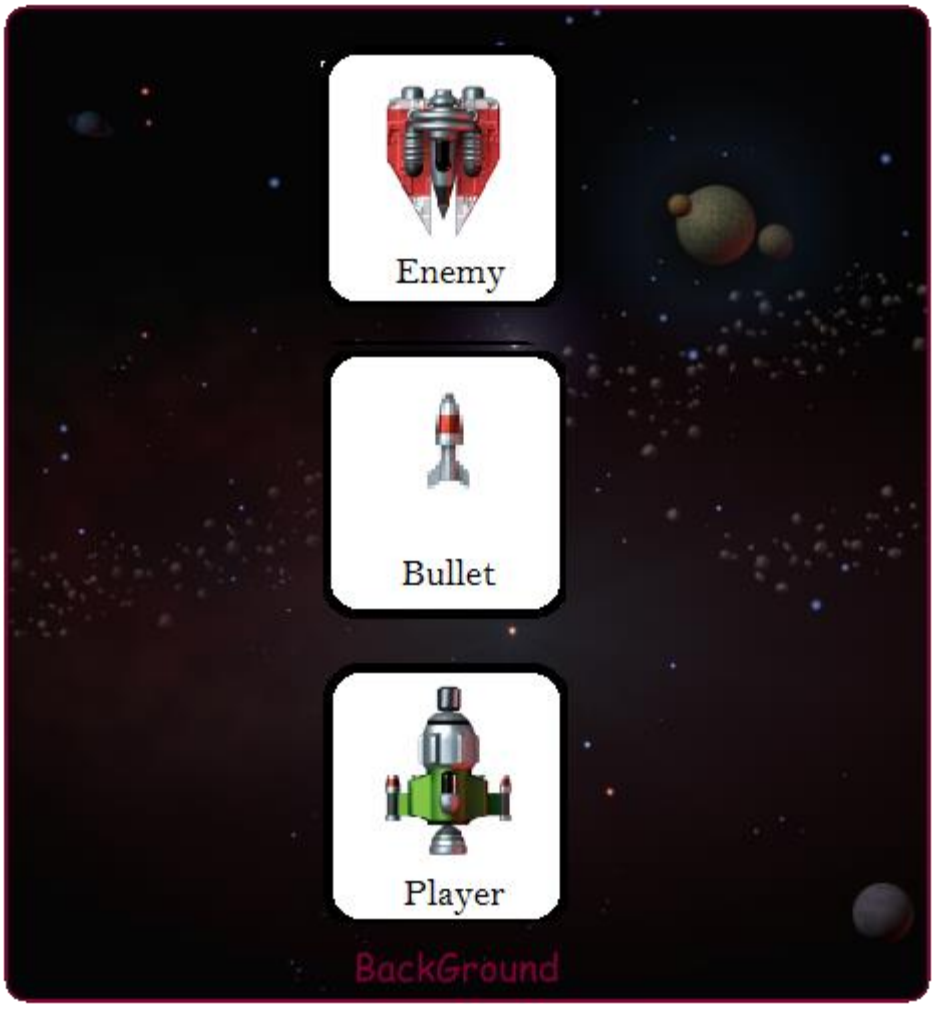


Рисунок 3.3.2 – Основные модели приложения «Push Him All»

2) Music & Sounds

- Bullet – звук который появляется в результате запуска пули игроком (в стандартной модели – ракета).
- BackGroundMusic – музыка которая проигрывается в качестве фона.
- Woopsy – звук издаваемый при включении функции «Woopsy» (см. пункт 4.1).

3) Video

- MainVideo – заставка в главном окне программы. Представляет собой анимацию корабля, летящего к красному карлику (см. рис. 4.1.1).

3.4 Разработка логической составляющей

3.4.1 Разработка уровней сложности

В игровом приложение «Push Him All» требуется разработать различные уровни сложности.

- Стандартная сложность

Сложность, при которой у пользователя не возникает трудностей в процессе игры. Данный вид сложности представляет из себя частный случай заданной сложности.

- Заданная сложность

Пользователь может задать уровень сложности показателем от 1 до 10, при этом 10 уровень сложности является самым трудным для прохождения, а 1 самым легким.

- Адаптивная сложность

Данный уровень сложности будет меняться в зависимости от успехов игрока в процессе игры. Таким образом сложность будет менять свое значение от самого легкого показателя до самого сложного и наоборот.

В качестве примера на рисунке 3.4.1 представлен алгоритм работы адаптивной сложности.

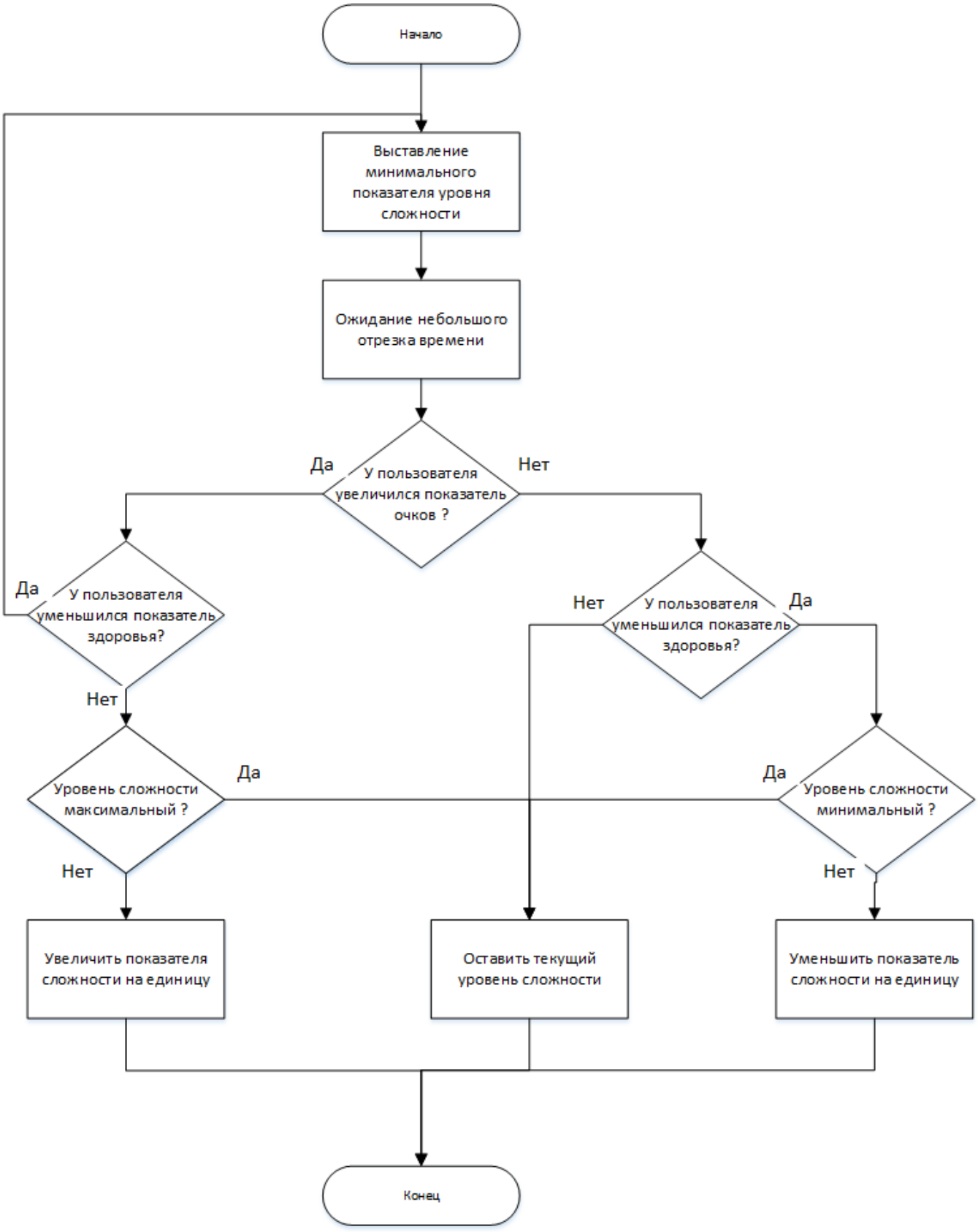


Рисунок 3.4.1 – Алгоритм работы адаптивной сложности

3.4.2 Разработка типов передвижения

Можно выделить три вида перемещения вражеских объектов (рис 3.4.2.1):

- передвижение по прямой траектории;
- передвижение по косой траектории;
- передвижение по гармоническому закону.

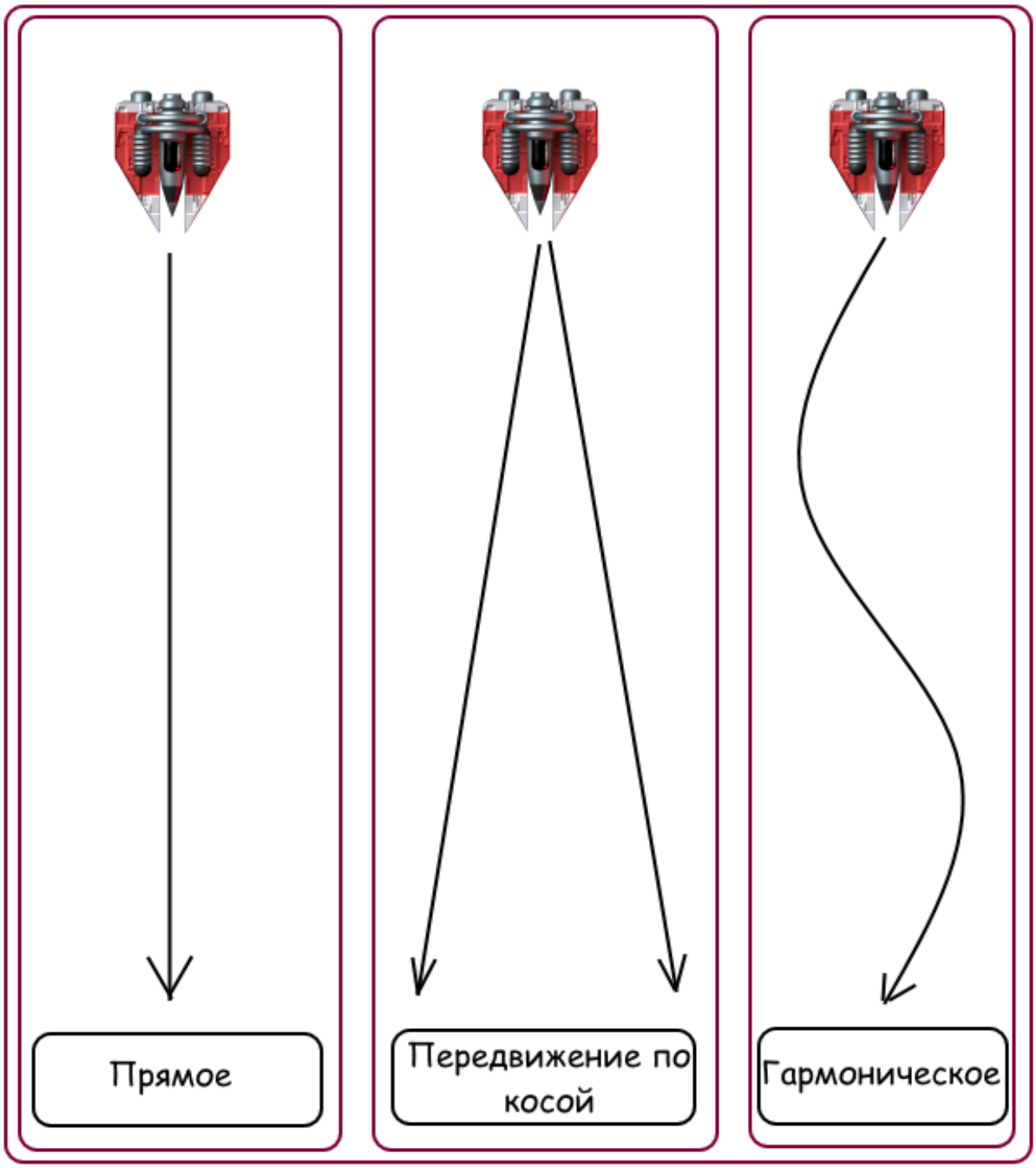


Рисунок 3.4.2.1 – Виды передвижение вражеских объектов

1) Передвижение объектов по прямой

Для передвижение объекта по прямой траектории следует смещать ее по координате Y из-за особенностей системы координат графического представления (рис. 3.2.2). Таким образом, чтобы переместить вражескую модель из верхней позиции экрана в нижнюю, воспользуемся следующей функцией:

```
SetPos (x(),y() + offset);
```

Где offset – величина смещения относительно координаты Y

2) Передвижение объектов по косой траектории

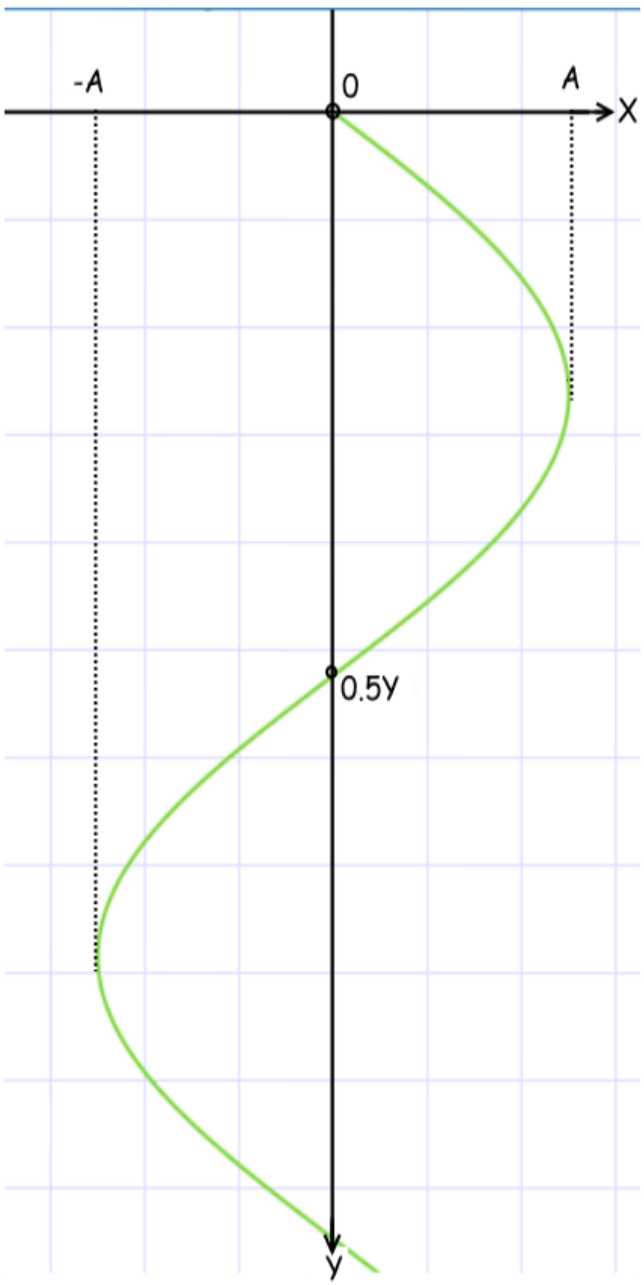
Для передвижение объекта по косой траектории следует смещать ее по координате Y и X. При этом значение смещение Y должно превышать значение смещения X в несколько раз. Так же следует учитывать местоположение появления вражеского объекта. Если объект появился в левой верхней части экрана, то его конечный пункт назначения нижняя правая граница и наоборот. Таким образом, функция передвижения примет следующий вид:

```
SetPos (x() ± offsetX, y() + offsetY);
```

Где offsetX (Y) – смещение по координате X (Y).

3) Передвижение объектов по гармоническому колебанию

Чтобы объект передвигался по гармоническому закону воспользуемся тригонометрической функцией – синусом (рис 3.4.2.2). Из рисунка видно, что для того чтобы увеличить смещение относительно координаты X – следует изменить значение параметра A (амплитуды колебания). Для увеличения смещения относительно координаты Y – требуется изменить параметр частоты w (циклическая частота). Параметром начальной фазы (f0) пренебрежём (установим значение в 0).



$X = A * \sin(wy + f0)$

A - амплитуда колебания (смещение относительно X)

w - циклическая частота

f0 - начальная фаза

Рисунок 3.4.2.2 – Тригонометрическая функция синуса

Тогда функция передвижения примет следующий вид:

```
SetPos (x() ± A*sin(y()*w), y() + offsetY);
```

Где A – смещение относительно координаты X; w – параметр циклической частоты; offsetY – смещение по координате Y.

3.5 Выводы по разделу

В результате процесса разработки были проделаны следующие задачи:

- выделены основные положения игры (тип игры, жанр и т.д.);
- выработаны основные цели и задачи игры;
- продуманы основные возможности и особенности игры;
- составлены UML – диаграммы функциональной составляющей программной системы;
- изучены технологии работы с графическими компонентами;
- для работы со звуковыми компонентами системы были изучены соответствующие библиотеки;
- продуманы логические компоненты системы;
- разработаны различные уровни сложности игрового приложения;
- разработаны различные типы передвижения, а также рассчитаны формулы для каждого типа перемещения вражеских единиц;
- были найдены и использованы бесплатные модели для программной системы;
- разработаны интерфейсы для взаимодействия с пользователем (загрузка текстур, выбор режима сложности/траектории);

На рисунке 3.5 представлена краткая диаграмма классов продукта «Push Him All», на которой можно увидеть следующее: есть основные классы – врага («Enemy»), игрока («Player»), пули («Enemy»), показателя здоровья («Health») и очков («Score»). С помощью интерфейса взаимодействия с пользователем («GetFile») происходит выборка текстур некоторых компонентов, а также, по желанию пользователя, звуковой составляющей. После чего все полученные данные (если данные не получены – используются стандартные компоненты) используются в классе «Game», который производит инициализацию всех компонентов игры, запускает ее, а также следит за внешними сигналами от

пользователя (нажатие клавиш). Работа (использование) продукта «Push Him All» представлена в главе 4.

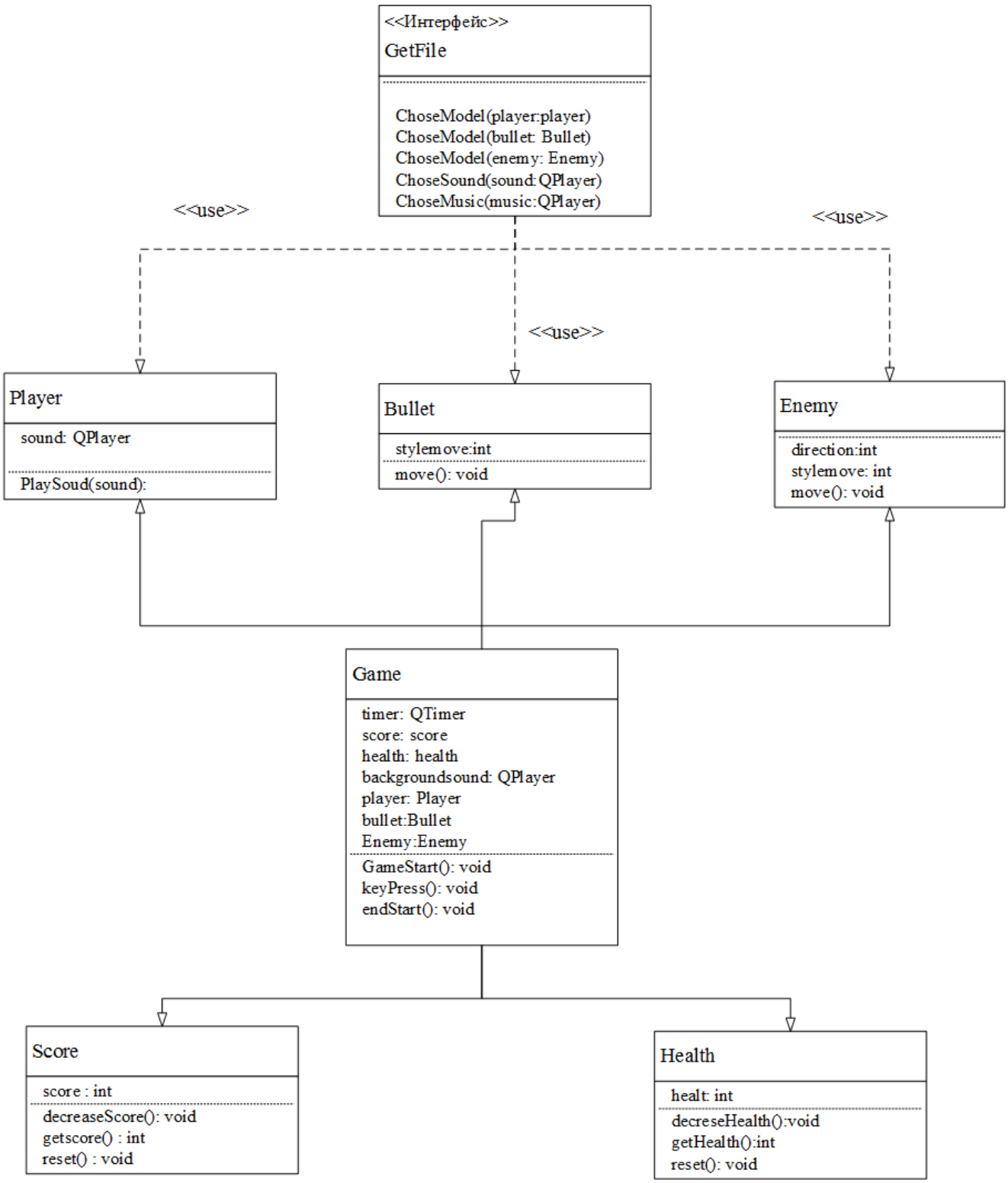


Рисунок 3.5 – Диаграмма классов программного игрового продукта «Push Him All»

4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

4.1 Общее описание

«Push Him All» ─ это игра в которой игроку требуется выживать, уничтожать как можно больше врагов и заработать как можно больше очков!

При запуске программы «Push Him All» пользователь видит основное меню программы, представленное на рисунке 4.1.1. Взаимодействие с элементами меню осуществляется с помощью компьютерной мыши – наведите курсор на интересующий вас элемент и нажмите на левую кнопку мыши.

Основные пункты меню:

- играть;
- настройка текстур;
- настройка логики;
- выход.

При нажатие пользователем на кнопку «Играть» появляется окно, в котором осуществляется основной процесс игры (рис. 4.1.2).

Рисунок 4.1.1 – Главное меню

Рисунок 4.1.2 – Основное окно игры

Основные элементы управления:

- клавиша «Влево» ─ передвижение модели игрока влево;
- клавиша «Вправо» ─ передвижение модели игрока вправо;
- клавиша «Пробел» ─ выпуск пули моделью игрока.

В верхней левой части экраны представлены три показателя.

- “Score” ─ счет игрока (показатель очков);
- “BestScore” ─ лучший показатель очков;
- “Health” ─ показатель здоровья (изначально данный показатель равен трем).

По окончанию игры, т.е. по достижение нулевого показателя здоровья (Health = 0) игроку выводится окно счета (рис 4.1.3), в котором игрок может выбрать следуюшие действия.

- «Да!» ─ начать игру снова, при этом все настройки игры сохраняются;
- «Настройки» ─ перейти в окно настроек графических компонентов;
- «Логика» ─ перейти в окно настроек логики;
- «Выход» ─ закрыть игру.

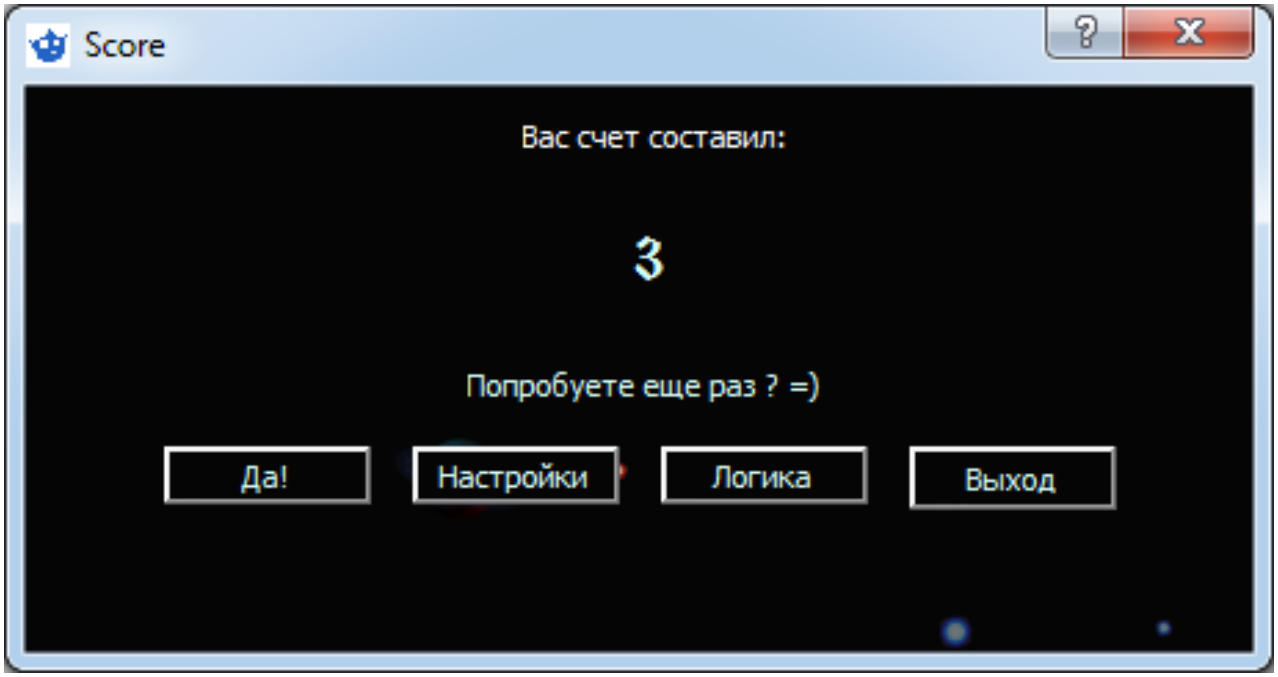


Рисунок 4.1.3 – Окно счета

При выборе пункта «Настройки» пользователю выводится окно, представленное на рисунке 4.1.4. В данном окне игрок может настроить:

- текстуру модели игрока;
- текстуру модели пули;
- текстуру модели врага;
- фон игры;
- звук пули (а так же его громкость);
- музыку игры (а так же ее громкость);

Так же у игрока есть возможность сохранить выбранные настройки с помощью кнопки «Сохранить» и в будущем загрузить их с помощью кнопки «Загрузить».

По окончанию всех настроек игрок должен нажать кнопку «Применить», расположенную в нижнем правом углу, чтобы его изменения вступили в силу.

При выборе пункта «Логика» пользователю выводится окно, представленное на рисунке 4.1.5. В данном окне игрок может настроить:

1) функцию «Woopsy»

Данная функция выводит картинку в нижнем правом углу и воспроизводит звуковой сигнал, которые можно указать через кнопки «Текстура–Выбрать» и «Звук–выбрать». Функция «Woopsy» срабатывает по определённому условию, которое может задать игрок:

- если выбрать пункт «Кол-во очков», то функция активируется по достижение кол-ва очков, указанных рядом (справа) от данной настройки.
- если выбрать пункт «Кол-во секунд», то функция активируется через указанное кол-во секунд, после запуска игры.

Изначально (в стандартных настройках) данная функция отключена.

2) сложность

Игрок может выбрать один из трех уровней сложности:

- обыкновенная сложность ─ оптимальный уровень сложности, никак не изменяется во время игры.
- адаптивная сложность ─ в начале игры уровень сложности равен стандартному, однако в зависимости от игры пользователя, уровень сложности может увеличиваться или уменьшаться.
- заданная сложность ─ пользователь может выбрать уровень сложности от 1 до 10, при этом уровень сложности не будет изменяться на протяжение всей игры.

3) изменение траектории передвижения врагов (пункт 4.3)

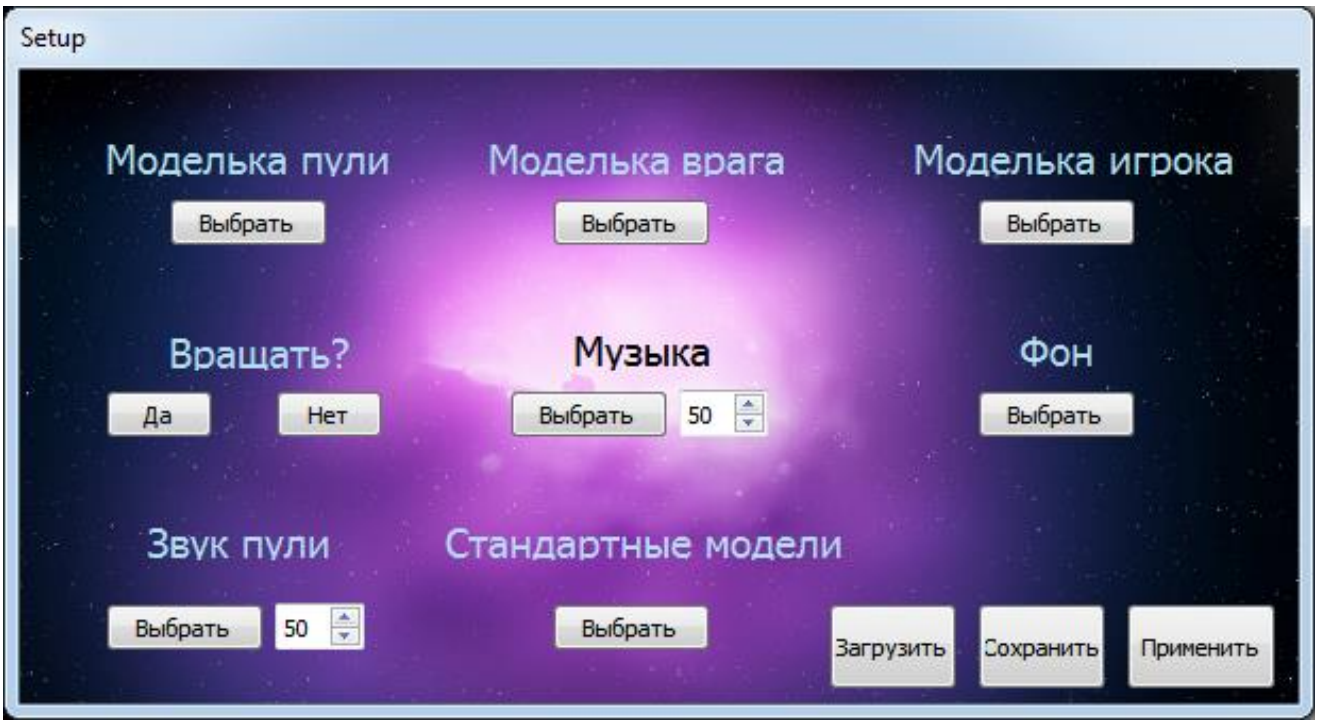


Рисунок 4.1.4 – Меню настроек

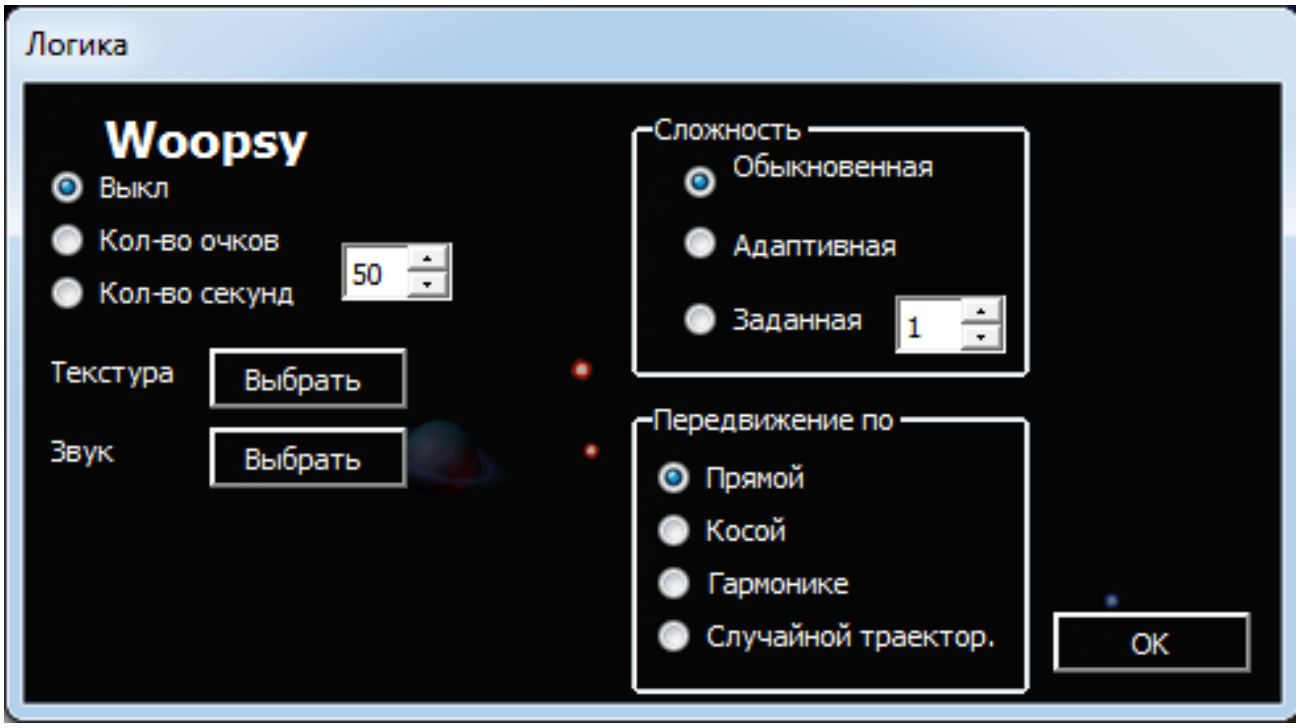


Рисунок 4.1.5 – Меню настроек логики

4.2 Настройка моделей

В качестве примера настройки приложения проделаем следующие действия:

- заменим модель игрока;
- заменим модель врага;
- заменим модель пули;
- заменим фон игры;
- сохраним настройки;

Для этого запустим приложение «Push Him All» и перейдем в меню «Настройка» (рис. 4.2.1).

Рисунок 4.2.1 – Переход в окно настроек

Для замены модели пули нужно нажать соответствующую кнопку «Выбрать» (рис. 4.2.2). После этого перед пользователем открывается новое окно, в котором он должен указать нужную ему модель (рис.4.2.3). В качестве модели пули укажем файл «R2d2».

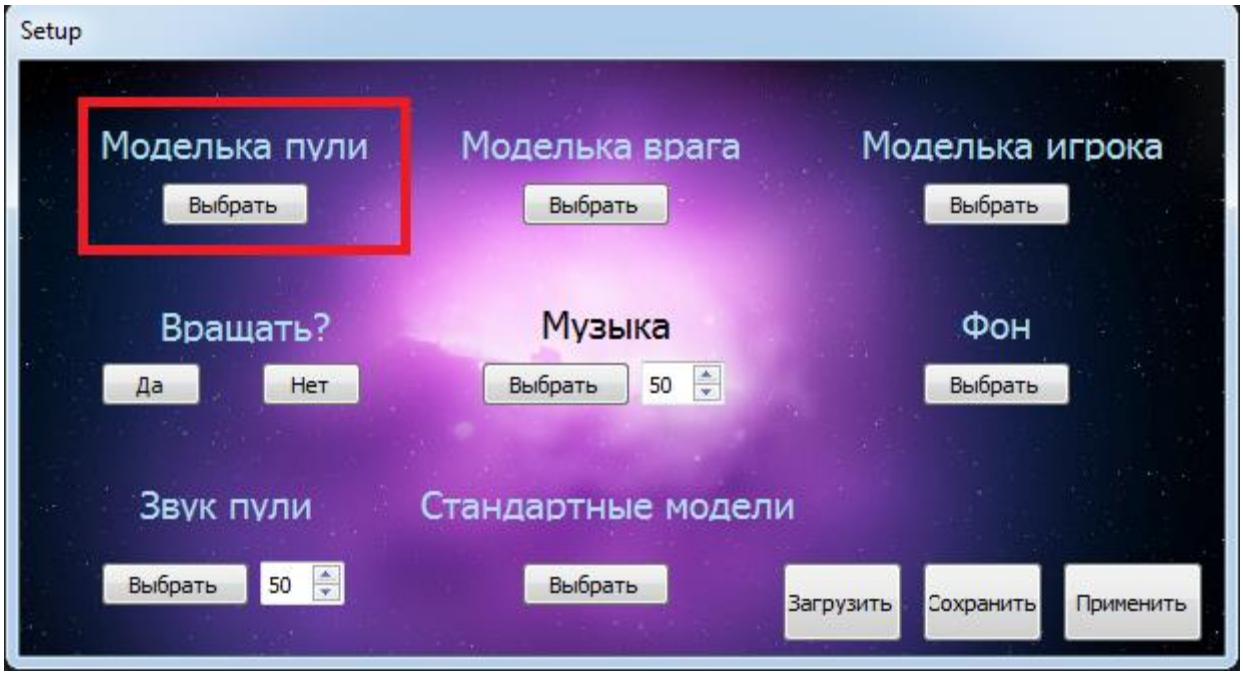


Рисунок 4.2.2 – Окно настроек

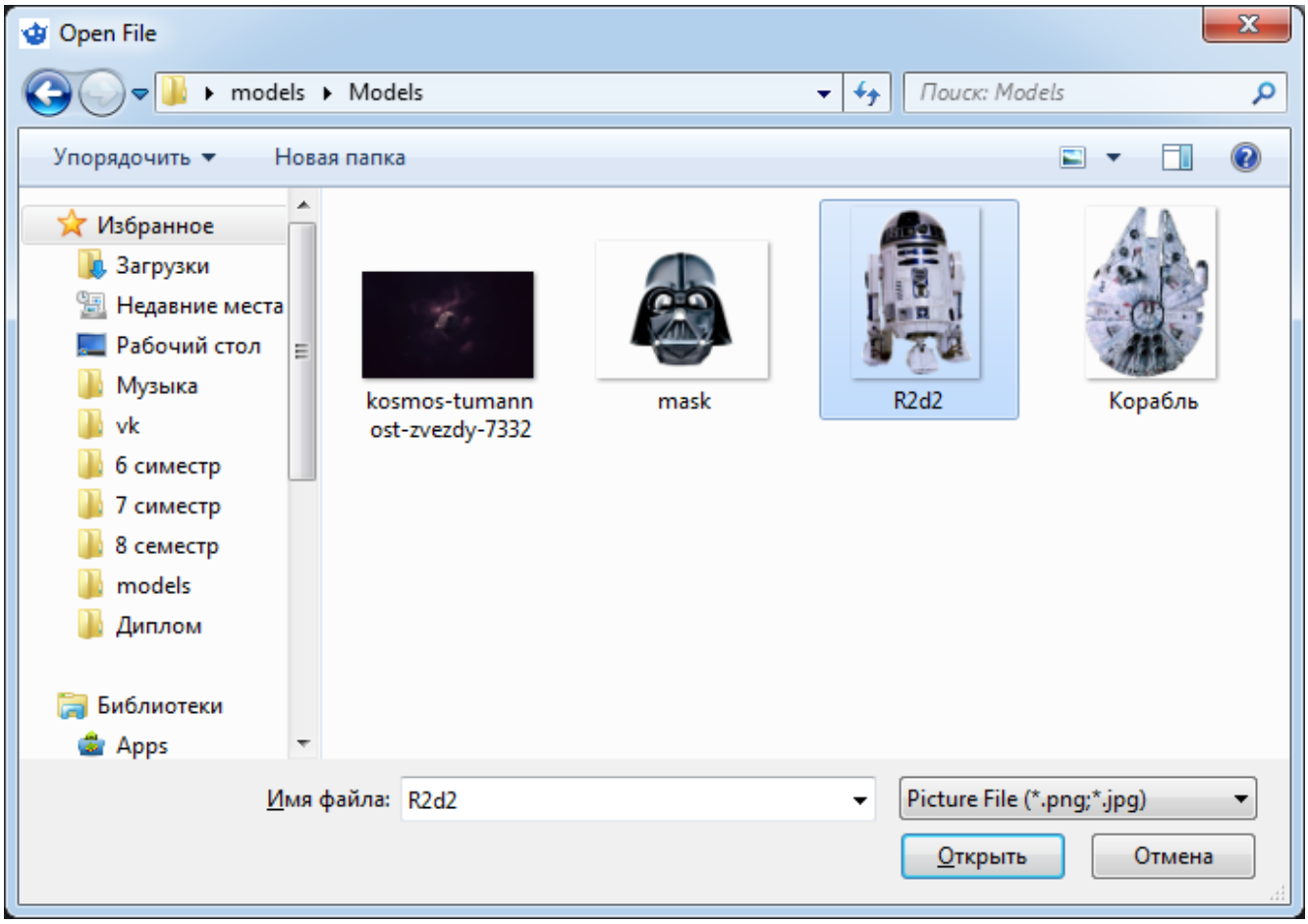


Рисунок 4.2.3 – Окно выбора модели

Проделаем подобные действия и для других моделей.

- В качестве модели врага укажем файл «mask».
- В качестве модели игрока укажем файл «Корабль».
- В качестве фона укажем файл «kosmos».

Сохраним настройки с помощью кнопки «Сохранить» (рис. 4.2.4).

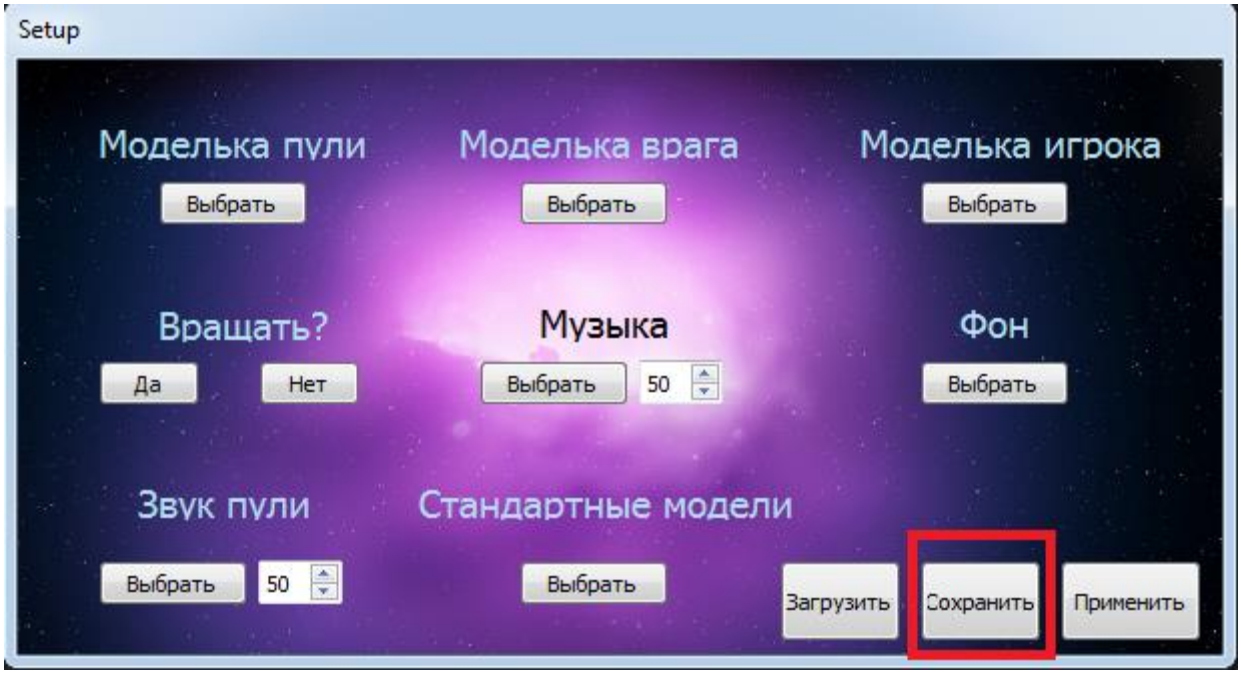


Рисунок 4.2.4 – Сохранение настроек

Для того, чтобы изменения вступили в силу, нужно нажать кнопку «Применить» (рис 4.2.5).

После этого игрок увидит главное окно программы. Теперь запустим игру с помощью кнопки «Играть» и посмотрим на результат (рис. 4.2.6).

Как видно из рисунка 4.2.6, все настройки успешно применились.

Чтобы загрузить ранее сохраненные настройки требуется перейти в меню «Настройки» и нажать на кнопку «Загрузить».

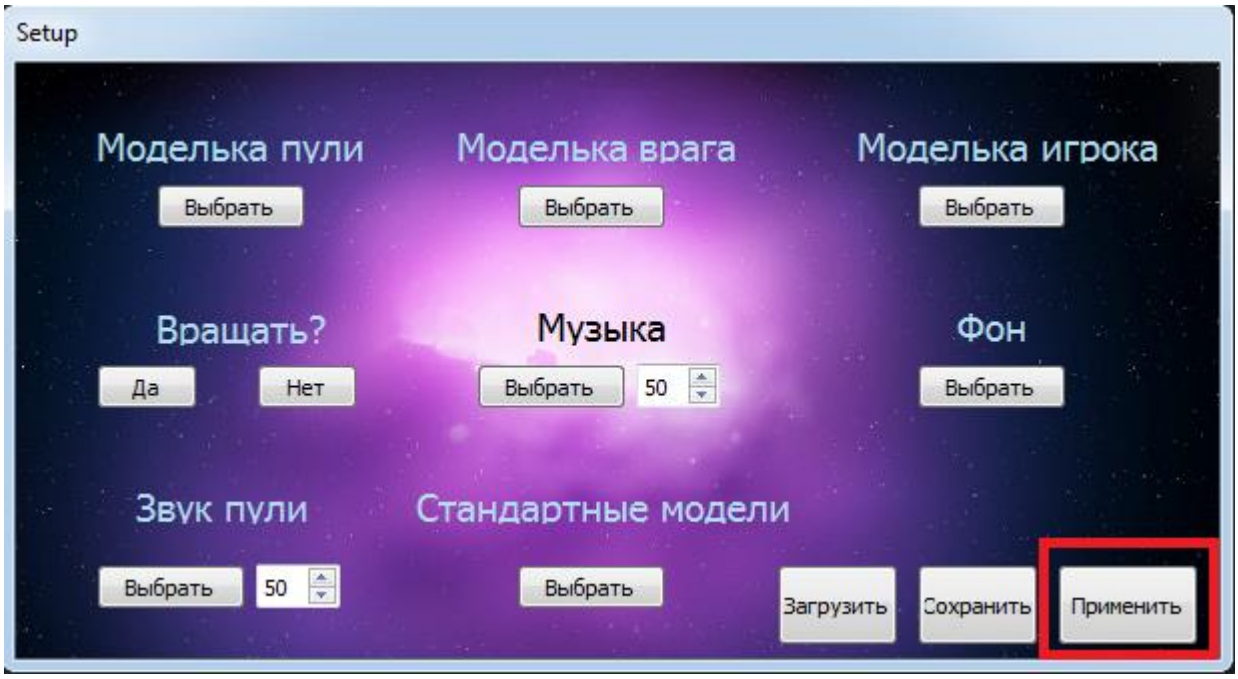


Рисунок 4.2.5 – Применение настроек

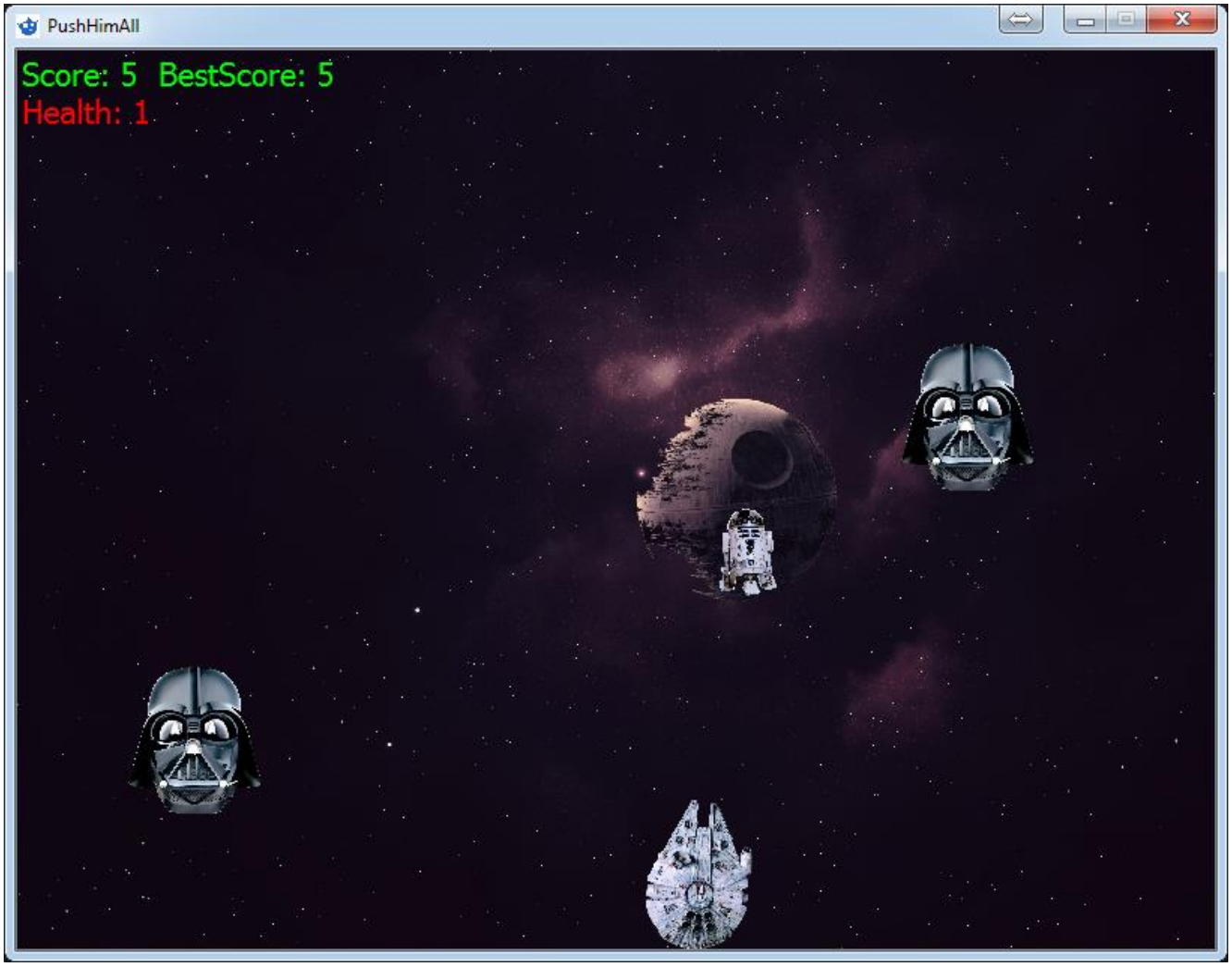


Рисунок 4.2.6 – Результат примененных настроек

Так же пользователь может выбрать опцию «Вращать». Данная функция позволяет вращать стандартную или заданную пользователем (игроком) модель пули. Чтобы модель пули вращалась нужно выбрать соответствующий пункт в меню настроек (рис. 4.2.7). «Да» – активирует функцию вращения. «Нет» - отключает данную функцию (модель пули не вращается).

Пользователю предоставляется возможно указать музыку в игре с помощью пункта «Музыка» и указать ее громкость, а также звук (и его громкость) пули с помощью пункта «Звук пули».

Для игры со стандартными моделями требуется нажать на кнопку «Выбрать» в пункте «Стандартные модели». После этого все настройки пользователя сбросятся, а игра примет стандартные значения.

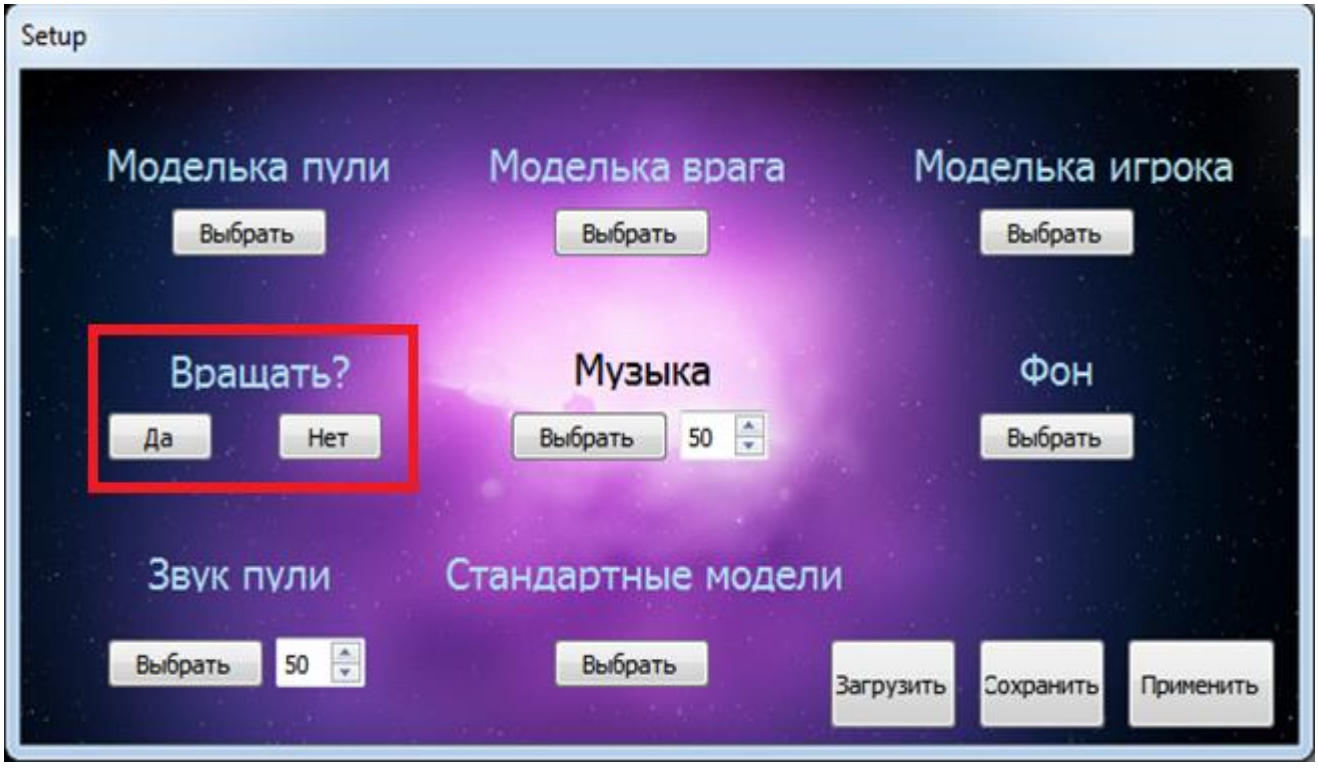


Рисунок 4.2.7 – Параметр вращения пули в меню настроек

4.3 Настройка логики

Игровое приложение «Push Him All» так же позволяет изменять поведение врагов, в частности:

1) изменять сложность игры;

2) изменять траекторию передвижения врагов;

Для того чтобы задать данные параметры нужно перейти с главного меню игрового приложения в меню «Логика» (рис 4.1.5) или по завершению игры, нажав кнопку «Логика» на окне счета (рис 4.1.3).

У пользователя (игрока) есть возможность задать траекторию передвижения вражеских единиц. Изначально все враги движутся по прямой траектории, однако, пользователь может изменить ее на следующие варианты:

- передвижение по прямой

Вражеская единица создается в верхней части игрового окна и начинает свое передвижение в нижнюю часть экрана по прямой траектории.

- передвижение по косой

В зависимости от положения создания вражеской единицы, она начнет передвигаться в нижний левый или правый угол игрового окна.

- передвижение по гармонике

В данном варианте вражеская единица будет менять свое положение в течение всего пути по гармоническому закону. В данном варианте попасть по цели становится значительно труднее, тем самым тренируются прогностические способности игрока.

- случайная траектория

Все вражеские единицы будут иметь собственный закон передвижения, которые перечислены выше (по прямой, косой, гармонике). Данный режим является самым трудным, так как требуется прогнозировать траекторию каждой вражеской единицы единовременно.

ЗАКЛЮЧЕНИЕ

Доля пользователей ПК и интернет аудитории растет ежедневно. В основном пользователи используют ПК в качестве развлекательной платформы. Таким образом, игровая аудитория продолжает расти, тем самым мотивируя разработчиков создавать новые игровые проекты.

В рамках выпускной квалификационной работы была изучена и проработана технология разработки игр путем создания игрового приложения «Push Him All» в жанре «Аркада» с возможностью изменять модели игры, а также логику поведения врагов. Разработка велась на языке программирования C++ с использованием библиотек Qt и IDE Qt Creator.

Для достижения данной цели были решены следующие задачи:

- осуществлена постановка игровой задачи;
- произведен анализ аналогов и программных средств разработки компьютерной игры;
- произведен поиск и подбор бесплатных моделей;
- повторены и углублены знания в языке программирования C++;
- изучена работа с некоторыми библиотеками Qt;
- спроектировано приложение;
- составлено руководство пользователя.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Профессии будущего. [Электронный ресурс]. – Режим доступа: http://www.proforientator.ru/publications/articles/detail.php?ID=5454 – (дата обращения 12.04.2017).

2. Анализ рынка игр в России и мире 2014-2016 годов [Электронный ресурс]. – Режим доступа: http://json.tv/ict_video_watch/analiz-rynka-igr-v-rossii-i-mire-2014-2016-gg-tekuschaya-situatsiya-prognozy-igroki-proekty-i-tendentsii. – (дата обращения 20.03.2017).

3. Рынок игр в России и мире с 2010 по 2016 год [Электронный ресурс]. – Режим доступа: http://json.tv/ict_telecom_analytics_view/rynok-igr-v-rossii-i-mire-2010-2016-gg-20141121113425. – (дата обращение 25.02.2017).

4. Worldwide Digital Games Market [Электронный ресурс]. – Режим доступа: https://twitter.com/_SuperData/status/675337478514024448. – (дата обращение 21.03.2017).

5. Top Operating System Software [Электронный ресурс]. – Режим доступа: http://www.jegsworks.com/lessons/ComputerBasics/lesson8/lesson8-3.html . – (дата обращение 25.03.2017).

6. Unity [Электронный ресурс]. – Режим доступа: https://unity3d.com/ru . – (дата обращение 15.03.2017).

7. Популярные игры [Электронный ресурс]. – Режим доступа: https://habrahabr.ru/company/mailru/blog/181974/. – (дата обращения 12.04.2017).

8. CRYENGINE - The complete solution for next generation game development [Электронный ресурс]. – Режим доступа: https://www.cryengine.com. – (дата обращение 20.03.2017).

9. What is Unreal Engine 4 ? [Электронный ресурс]. – Режим доступа: https://www.unrealengine.com/ . – (дата обращение 18.03.2017).

10. Game programming [Электронный ресурс]. – Режим доступа: goo.gl/rxJ4xzc. – (дата обращение 25.03.2017).

11. Qt Graphics Framework [Электронный ресурс]. – Режим доступа: http://doc.qt.io/qt-5/graphicsview.html . – (дата обращение 25.03.2017).

12. What is SFML ? [Электронный ресурс]. – Режим доступа: https://www.sfml-dev.org/grl-whatis. – (дата обращение 25.03.2017).

13. Space Invaders. [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/SpaceInvaders. – (дата обращения 13.04.2017).

14. Radiant. [Электронный ресурс]. – Режим доступа: http://www.hexag.net/radiant/. – (дата обращения 13.04.2017).

15. Luxor. [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Luxor_Game/. – (дата обращения 13.04.2017).

16. Beyond Space. [Электронный ресурс]. – Режим доступа: http://store.steampowered.com/app/297111/. – (дата обращения 13.04.2017).

17. Чужой в космосе. [Электронный ресурс]. – Режим доступа: http://onlineguru.ru/15104/view.html/. – (дата обращения 13.04.2017).

18. Диаграмма вариантов использования как концептуальное представление бизнес-системы в процессе ее разработки. [Электронный ресурс]. – Режим доступа: http://www.intuit.ru/studies/courses/32/32/lecture/1004 - (дата обращения 10.02.2017).

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД

В данном приложение представлен код главного меню программы:

- MainWindow.h – заголовочный файл главного меню.
- MainWindow.c – исходный код главного меню.
- MainWindow.ui – файл формы главного меню (настройка визуальных компонентов окна(формы)).

«MainWindow.h» - заголовочный файл главного меню

```
#ifndef MANWINDOWS_H
#define MANWINDOWS_H
#include <QManWindows>

namespace Uii
{
class ManWindows;
}

class ManWindows : public QManWindows{
    Q_OBJECT

public:
    ~ManWindows();

private slotss:
    void on_ExitButton_clicked();  //кнопка «Выход»
    void restartVideo();   //Зацикливание анимированного меню
    void on_GameButton_clicked(); // Кнопка «Играть» - запуск игр
    void   on_SettingButton_clicked();  //кнопка «Настройки» - открытие настроек текстур и музыки.
    void  on_BtnForLogic_clicked(); //кнопка «Логика» - открытие настроек логики

};
```

«MainWindow.c» - исходный код главного меню

```
//Подключение библиотек
#include "MainWindow.h"
#include "ui_MainWindow.h"
#include "Game.h"
#include "Settings.h"
#include <QMovie>
#include <QLabel>
#include <QGraphicsVideoItem>
#include <QGraphicsView>
#include <QGraphicsScene>
#include <QMediaPlayer>
#include <QPushButton>
#include <QFileDialog>
#include <QMessageBox>
#include <QUrl>
#include <QFile>

//Объявление переменных
extern Game * game;
QMediaPlayer *player=new QMediaPlayer();
int clickplay = 0;


//Инициализация меню
//Включение анимированного меню

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
```

```
        uii(new Uii::ManWindows)


        uii->setupUii(this);
        this->setFixedSize(this->size().width(),this->size().height());
           setAttribute(Qt::WA_DeleteOnClose);
           QGraphicsVideoItem *item = new QGraphicsVideoItem;
           item->setSize(ui->playerScreen->size());
           player->setVideoOutput(item);
           QGraphicsScene    *scene    =    new    QGraphicsScene(0,0,ui-
>playerScreen->size().width(),ui->playerScreen->size().height());
           ui->playerScreen->setScene(scene);
           ui->playerScreen->scene()->addItem(item);
           player->setMedia(QUrl::fromLocalFile("E:\\Diplom.mp4"));
           player->setVolume(100);
           player->play();

    connect(player,SIGNAL(stateChanged(QMediaPlayer::State)),this,SLOT(re
    startVideo()));
    }
    MainWindow::~MainWindow()
    {
        delete ui;
    }
    //Кнопка «Выход» - закрытие приложения
    void MainWindow::on_pushButton_2_clicked()
    {
        exit(0);
    }

    // Анимированное меню
    void MainWindow::on_pushButton_3_clicked()
    {
```

```
            clickplay++;
            player->stop();
            QString filename;
           player->setMedia(QUrl::fromLocalFile(filename));
            player->play();
            clickplay--;
        }
        void MainWindow::restartVideo()
        {
            if(!clickplay)
            player->play();
        }
        //Кнопка «Играть» - запуск игры
        void MainWindow::on_pushButton_clicked()
        {
            clickplay++;
            player->stop();
            delete player;
            game = new Game();
            game->show();
            this->close();
            clickplay--;
        }
        //Кнопка «Настройки» - открытие окна настройки текстур, музыки.
        void MainWindow::on_SettingButton_4_clicked()
        {
            Settings *setup = new Settings();
            setup->setAttribute(Qt::WA_DeleteOnClose);
            setup->setFixedSize(setup->size().width(),setup-
>size().height());
            setup->show();
```

```
        }
        //Кнопка «Логика» - открытие окна логики программы
        void MainWindow::on_LogicButton_clicked()
        {
            Settings *setup = new Settings();
            setup->setAttribute(Qt::WA_DeleteOnClose);
            setup->setFixedSize(setup->size().width(),setup-
>size().height());
            setup->show();
        }
```

«MainWindow.ui» - файл формы

```
Project created by QtCreator 2016-05-06T23:46:01
#Autor: Alexander Ivanov
#-------------------------------------------------

//Подключение библиотек QT
QT += core gui
QT += multimedia
QT += multimediawidgets
QT += widgets

QT_QPA_PLATFORM_PLUGIN_PATH=%QTDIR%\plugins\platforms\


greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

TARGET = untitled
TEMPLATE = app

//Подключение исходных файлов

/*
Enemy – код логики врага
Bullet – код логики пули
Game – код логики игры
Score – код обработки счета
Player – обработка нажатий пользователя
Health – обработка показателя здоровья
Mainwindows – форма главного меню
```

```
Settings - форма окна настроек
Ending – форма окна счета
 logic.cpp – логика передвжиения врагов
*/


SOURCES += main.cpp \
    Enemy.cpp \
    Bullet.cpp \
    Game.cpp \
    Score.cpp \
    Player.cpp \
    Health.cpp \
    ManWindows.cpp \
    Settings.cpp \
    Ending.cpp \
    logic.cpp

//Подключение заголовочных файлов файлов

HEADERS  += \
    Bullet.h \
    Enemy.h \
    Game.h \
    Score.h \
    Player.h \
    Health.h \
    ManWindows.h \
    Settings.h \
    Ending.h \
```

```
    logic.h

//Подключение в ресурсов

RESOURCES += \
    res.qrc
//Подключение файлов форм

FORMS += \
    MainWindow.ui \
    Settings.ui \
    Ending.ui \
    logic.ui

//Подключение иконки приложения
win32:RC_ICONS += Icon1.ico
```