СОДЕРЖАНИЕ

Введение 4

1 Технический проект 6

1.1 Классификация игровых приложений 6

1.2 Выбор целевой операционной системы 8

1.3 Постановка задачи 10

1.4 Выбор средств реализации 11

1.5 Требования к программному обеспечению и техническим средствам 15

1.6 Проектирование и моделирование игрового приложения 16

1.6.1 Сценарий и стратегия игры 16

1.6.2 Функции персонажей 16

1.6.3 Графика 17

1.6.4 Основные функции игрового приложения 18

2 Рабочий проект 20

2.1 Стандартные объекты игрового приложения 20

2.2 Пользовательские объекты игрового приложения 21

2.3 Установка и настройка игрового приложения 23

2.4 Работа с игровым приложением 24

Заключение 27

Список использованных источников 29

Приложение А – Листинг скриптов 30

ВВЕДЕНИЕ

Темой курсового проекта является «Разработка и создание игрового приложения на Unity».

С момента начала информационно-технической революции мир стремительно движется в будущее, создавая все более совершенные компьютерные системы, чтобы облегчить жизнь человека, а также занять его досуг. Индустрия компьютерных игр вошла в число самых прибыльных и догнала по показателям роста прибыли киноиндустрию и индустрию спорта. Компьютерные развлечения – это мощная экономическая сфера, приносящая огромные доходы. Поэтому неслучайно, что особая роль в жизни современного человека отводится компьютерным играм, первые из которых существовали на самой заре компьютерной техники.

Вопреки ошибочному мнению, игры – это не просто программный код. Можно сказать, что игры – это современный вид искусства, который может нести не только развлекательный характер, но и заставлять думать, переживать, поднимать серьезные вопросы.

Целью данного курсового проекта является создание 2D игры для ОС Windows.

Написание данного курсового проекта целесообразно, так как в настоящее время приложения, особенно игровые, основанные на ОС Windows, являются востребованными на рынке программного обеспечения.

Объектом исследования будут игры для ОС Windows.

Предметом данного дипломного проекта будут особенности реализации игрового приложения для платформы Windows, а также средства реализации программного продукта.

Для достижения поставленной цели в проекте будут рассмотрены и решены следующие вопросы и задачи:

* выбран жанр для игрового приложения;
* спроектирована структура и функциональность игры;
* спроектированы история, персонажи и уровни;
* выбраны средства реализации компьютерных игр для ОС Windows;
* определены минимальные требования к техническим и программным средствам;
* описаны входные и выходные данные;
* разработан программный код приложения;
* создана справочная система;
* создана инсталляция.

Прежде чем начать создавать новую игру, необходимо определиться с игровым движком, то есть с программным компонентом, позволяющим создавать и запускать видеоигры. Он предоставляет разработчикам инструменты для создания большинства компонентов игры, а потом позволяет им собрать их воедино. Движок игры затрагивает все компоненты игры, начиная от рендеринга, физики, звукового оформления, скриптинга, создания искусственного интеллекта и заканчивая сетевыми аспектами. Если что-то создать с его помощью невозможно, то это создается в специализированной программе и потом импортируется в игру.

Теоретическими основами и методами решения поставленных задач будут методологии проектирования функциональности приложения, особенности реализации программного кода с помощью конкретного языка программирования, принципы и алгоритмы применения инструментальных средств на различных этапах разработки программного продукта.

Ресурсы игры будут браться из открытых источников.

Данный программный продукт будет предназначен для широкого круга пользователей и предполагает коммерческое использование только с помощью встроенной рекламы.

1 ТЕХНИЧЕСКИЙ ПРОЕКТ

1.1 Классификация игровых приложений

Жанры игр формировались бесструктурно и интуитивно на протяжении длительного времени. Разработчики игр проводили смелые эксперименты, создавая новые игровые механики. Неудавшиеся эксперименты забывались, а удачные игры становились примером для других разработчиков. Разработчики копировали популярную игровую механику, добавляя немного идей от себя, таким образом, вокруг наиболее популярных игр образовывались целые классы схожих между собой игр, именно они и стали называться игровыми жанрами.

В отличие от классических жанров фильмов и книг, жанры игр подразумевают под собой совсем другие особенности. Они определяются не по сюжету, а по игровым действиям, которые наиболее часто совершает игрок.

Разделение на жанры очень полезно на практике. Игры определенного жанра интересны уже сформировавшейся игровой аудитории. Разработчик объявляет, что выпускает игру в таком-то жанре, и игроки уже примерно представляют себе, что будет происходить в игре, даже без уточняющих комментариев разработчиков.

Основные игровые жанры рассмотрены ниже.

Аркады – категория компьютерных игр с простым игровым процессом, сводящимся к выполнению ограниченного набора действий, нацеленных на достижение определенного результата. Задачей игр этого жанра будет набор максимального количества очков и/или минимальное время прохождения игры.

Приключенческие игры (квесты) – сюжетные игры, в которых герой выполняет различные задания, для чего ему требуется общение с другими персонажами, поиск и использование различных предметов, решение задач и головоломок.

Ролевые игры (англ. RPG – Role Playing Games) – многоплановые игры с нелинейным сюжетом, где игрок, управляя персонажем, имеет возможность улучшать его способности. Для этого требуется побеждать врагов, выполнять задания, продавать и покупать артефакты, и тому подобное. На характеристики персонажа могут влиять, например, своевременный прием пищи и сон. Притягательность таких игр во многом определяется адекватностью искусственного интеллекта, управляющего поведением прочих персонажей.

Развитие широкополосного Интернета привело к появлению разновидности ролевых игр – массовых многопользовательских ролевых онлайн-игр (ММОРПГ). В этом виде игровых программ виртуальные персонажи, управляемые реальными игроками, способны взаимодействовать друг с другом.

Симуляторы – игровые приложения, имитирующие управление каким-либо процессом или устройством. В основе хороших симуляторов лежат математические модели представляемых объектов и процессов. Чем полнее описана модель и ее окружение, тем реалистичней симулятор. Хорошие компьютерные симуляторы могут использоваться в качестве тренажеров.

Стратегические игры – задача этого жанра сводится к получению преимущества над противником, достигаемого путем выработки и реализации определенного плана. Объектами управления являются не отдельные персонажи, а целые корпорации, армии, государства и даже цивилизации. Различают пошаговые стратегические игры, где игроки ходят поочередно и стратегии реального времени (RTS – Real-time strategy), в которых все игроки действуют асинхронно.

Боевики – этот жанр, более известный под названием «экшен» (от англ. Action – действие) является одним из самых популярных видов компьютерных игр. Быстрота реакции и способность оперативно принимать решения – основные слагаемые успешного прохождения таких игр. В боевиках присутствует элемент насилия, возможно, этим и обусловлена популярность жанра.

Раннер (Runner и Endless Runner) – жанры игр, в центре внимания которых герой-бегун. Практически всегда персонаж бежит автоматически и от игрока требуется лишь корректировать его траекторию движения, вовремя совершать прыжки и уклоняться от препятствий. Это требует концентрации внимания и хорошей реакции.

Рогалик ( Roguelike) – особенностями классического roguelike являются генерируемые случайным образом уровни, [пошаговость](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%88%D0%B0%D0%B3%D0%BE%D0%B2%D0%B0%D1%8F_%D1%82%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D0%B0) и необратимость смерти персонажа – в случае его гибели игрок не может загрузить игру и должен начать её заново.

В курсовом проекте будет реализовываться игра в жанре Roguelike.

1.2 Выбор целевой операционной системы

ОС – комплекс программ, обеспечивающий управление аппаратными средствами компьютера, организующий работу с файлами и выполнение прикладных программ, осуществляющий ввод и вывод данных.

На сегодняшний день, ОС– это первый и основной набор программ, загружающийся в компьютер. Помимо вышеуказанных функций, ОС может осуществлять и другие, например, предоставление общего пользовательского интерфейса.

Сегодня наиболее известными ОС являются системы семейства Microsoft Windows, Linux, iOS и Android.

Microsoft Windows представляет целую серию ОС и рабочих сред, разработанных корпорацией Microsoft. Самая первая версия ОС Windows – MS-DOS, была представлена в 1985 году, имела графический пользовательский интерфейс. В ней была обеспеченна поддержка нескольких документов, поддержка компьютерной мыши, выпадающее меню, и всё это было возможным увидеть в цветах.

Следующие версии Windows постепенно заменяли многое построенное в MS-DOS, аппаратный и программный функционал. Все внесённые Microsoft изменения в MS-DOS с дальнейшей интеграцией поспособствовали получению развитой ОС.

В настоящее время Microsoft Windows является самой популярной ОС, не только из-за привычного удобства для использования и внушительного функционала, но и из-за высокого уровня интеграции с возможностями её ядра и другого программного обеспечения, в том числе MS Office. [1]

Linux – общее название UNIX-подобных ОС на основе одноимённого ядра и собранных для него библиотек и системных программ. К ОС GNU/Linux также часто относят программы, дополняющие эту ОС, и прикладные программы, делающие её полноценной многофункциональной ОС. В отличие от большинства других ОС, GNU/Linux поставляется в большом количестве так называемых дистрибутивов, в которых программы GNU соединяются с ядром Linux и другими программами. [6]

ОС iOS, в отличие от Windows Phone и Google Android, выпускается только для устройств, производимых фирмой Apple: смартфона iPhone, медиапроигрывателя iPodTouch и планшетного компьютера iPad. Данная ОС занимает сегодня лидирующие позиции на рынке мобильных платформ. iOS достаточно проста в использовании – в ней нетрудно разобраться даже самому непрофессиональному пользователю. Однако за внешней простотой скрывается довольно мощная система. На базе iOS работают смартфоны, медиаплееры и планшеты, поэтому можно с уверенностью сказать, что пользователь одного устройства с данной ОС не столкнется с проблемой сложности использования другого устройства, ввиду большой схожести функционала. [6]

Google Android – это не просто ОС для смартфона, а целая инфраструктура. Android является первой бесплатной ОС с открытым исходным кодом. На нем работают телефоны, планшеты, телевизоры, умные часы, и другие гаджеты, а скоро и автомобили будут управляться с помощью Android. ОС основана на ядре Linux и собственной реализации виртуальной машины Java от Google. Изначально разрабатывалась компанией Android Inc, которую в 2005 году купила Google. Впоследствии Google инициировала создание альянса Open Handset Alliance (OHA), который сейчас занимается поддержкой и дальнейшим развитием платформы. Android позволяет создавать Java-приложения, управляющие устройством через разработанные Google библиотеки. Android Native Development Kit позволяет портировать (но не отлаживать) библиотеки и компоненты приложений, написанные на Си и других языках. По данным 2014 года ОС Android была установлена на 86% смартфонов. [6]

Игра будет реализовываться для ОС Windows.

1.3 Постановка задачи

Необходимо разработать и создать 2D-игру в жанре Roguelike для ОС Windows.

Приложение должно иметь следующую структуру:

* главное меню;
* игровой уровень.

Приложение должно иметь дружественный интерфейс и обеспечивать:

* интерфейс главного меню (начать, настройки, выход);
* сохранение и отображение статистики;
* подсчет очков за прохождение;
* отображение справочного окна;
* музыкальное сопровождение.

При создании игрового приложения необходимо будет:

* создать сценарий игры и разработать её логику;
* спроектировать персонажа и уровень;
* разработать функционал игры;
* разработать макет страницы игрового поля;
* разработать графический интерфейс пользователя;
* прорисовать главного героя и анимировать его действия;
* разработать систему хранения информации.

Входными и выходными данным для приложения будет количество набранных очков.

Способ решения поставленных задач зависит от выбора среды разработки и языка программирования.

Тестирование будет проводиться на персональном компьютере с 64-разрядной ОС Windows 10, процессором IntelCore i3, CPU 3.40 GHz, 4 Гб RAM, дисплеем VGA с разрешением не менее 1366х768 точек на дюйм, оснащенном клавиатурой и мышью.

1.4 Выбор средств реализации

Прежде чем начать создавать новую игру, необходимо определиться с игровым движком, то есть с программным компонентом, позволяющим создавать и запускать видеоигры. Он предоставляет разработчикам инструменты для создания большинства компонентов игры, а потом позволяет им собрать их воедино. Движок игры затрагивает все компоненты игры, начиная от рендеринга, физики, звукового оформления, скриптинга, создания искусственного интеллекта и заканчивая сетевыми аспектами. Если что-то создать с его помощью невозможно, то это создается в специализированной программе и потом импортируется в игру.

На данный момент в мире существует огромное количество игровых движков. Рассмотрим наиболее популярные из них – UnrealEngine и Unity.

Unity – это кроссплатформенный игровой движок для разработки двухмерных и трехмерных приложений и игр. Unity3d имеет очень простой Drag and Drop интерфейс, который человек осваивает за месяц. Весь движок только на английском языке. Русификации Unity5 нет.

Интерфейс Unity разбит на несколько окон:

* Hierarchy – здесь находятся названия всех объектов на сцене, которые можно группировать и легко переходить по ним;
* Scene – здесь можно рассмотреть определенную сцену под нужным ракурсом;
* Inspector – поможет с настройкой выделенного объекта;
* Project – в нем отображаются все материалы проекта;
* Toolbar– содержит все доступные инструменты.

Unity 5 поддерживает использование двух языков программирования: C# (наиболее используемый) и Javascipt. Разработчику необходимо знать один из языков в совершенстве, а другой на среднем уровне, так как некоторые моменты Unity 5 делает только на одном из двух языков, или это делается намного труднее, чем на другом языке программирования.

Unity 5 обладает огромным количеством преимуществ перед другими игровыми движками. Его сообщество на сегодняшний момент является самым большим в мире. На официальном сайте Unity есть специальный раздел, в котором можно найти статистику по игровым движкам. По этим данным Unity 5 используют более 50% разработчиков видеоигр. 20% принадлежат UnrealEngine, а остальные игровые движки – 30%.

Для разработки 2D игр Unity 5 подходит по всем параметрам. В нем очень просто собирать проекты. Причем можно создать один проект под множество платформ, что очень сильно облегчает процесс дальнейшего распространения игр. Все скрипты, используемые в Unity 4, можно будет автоматически исправить в Unity 5. [9]

Unreal Engine – игровой движок, разрабатываемый и поддерживаемый компанией EpicGames. Написанный на языке C++, движок позволяет создавать игры для большинства ОС и платформ. Для людей, которые боятся высокого порога вхождения в C++, существует система под названием Blueprints, которая позволяет не набирать код руками, а составлять его из специальных блоков, которые графически расположены на экране.

Интерфейс Unity считается наиболее удобным и простым в использовании. Благодаря этому новички смогут быстрее добиться результата. Unreal Engine же, напротив, ощущается как инструмент для профессионалов. И как все профессиональное, пользовательский интерфейс здесь не слишком интуитивный.

Для упрощения портирования движок использует модульную систему зависимых компонентов; поддерживает различные системы рендеринга (Direct3D, OpenGL, Pixomatic; в ранних версиях: Glide, S3, PowerVR), воспроизведения звука (EAX, OpenAL, DirectSound3D; ранее: A3D), средства голосового воспроизведения текста, распознавание речи, модули для работы с сетью и поддержки различных устройств ввода.

Для игры по сети поддерживаются технологии WindowsLive, XboxLive, GameSpy и прочие, включая до 64 игроков (клиентов) одновременно. Таким образом, движок адаптировали и для применения в играх жанра MMORPG.

В отличие от Unity3d, в UnrealEngine 4 есть еще один большой плюс для программистов – это открытый код.

Для данного курсового проекта был выбран движок Unity. Главные аргументы – это удобство и скорость разработки под различные платформы, а также основной язык программирования С#.

В Unity можно использовать языки программирования C# и JavaScript. Рассмотрим плюсы и минусы обоих.

Плюсы C#:

* поддержка Microsoft. В отличие от JavaScript, которому не пошел на пользу переход в собственность Oracle, C# хорошо развивается благодаря усилиям Microsoft;
* в последнее время много совершенствуется. Так как C# был создан позже, чем JavaScript и другие языки, то требовалось очень много доработать. Также это касается популяризации и бесплатности – было обещано открыть исходный код, а инструменты (VisualStudio, Xamarin) стали бесплатными для частных лиц и небольших компаний;
* существует много конструкций, созданных для облегчения написания и понимания кода, которые не играют роли при компиляции;
* синтаксис похож на C, С++ или JavaScript, что облегчает переход на С# для других программистов;
* наличие Xamarin. Благодаря покупке Xamarin на C# теперь можно писать под Android и iOS.

Минусы С#:

* ориентированность, в основном, только на .NET (Windows платформу);
* бесплатность только для небольших компании, учащихся и программистов-одиночек;
* сохранился оператор goto. [3]

У языка JavaScript есть много преимуществ перед другими языками программирования, что позволяет решать с его помощью практически любые задачи.

Ниже перечислены основные преимущества JavaScript:

* язык прост для изучения. При разработке JavaScript было уделено большое внимание простоте языка, поэтому программы на нем, по сравнению с программами на других языках, проще писать, компилировать, отлаживать и изучать;
* JavaScript – это объектно-ориентированный язык. Это позволяет создавать модульные программы, исходный код которых может использоваться многократно;
* язык не зависит от платформы.

Одним из основных преимуществ языка JavaScript является возможность переноса программ из одной системы в другую. Поскольку программы на JavaScript не зависят от платформы, как на уровне исходного кода, так и на двоичном уровне, их можно запускать в различных системах, что особенно важно для программ, предназначенных для WorldWideWeb.

Выбор был остановлен на языке программирования C# и среде программирования Microsoft Visual Studio 2017.

Руководство пользователя, объясняющее смысл игры, что и как нужно делать игроку, будет отображаться в сцене главного меню, созданная с помощью текстовых и графических средств Unity.

Для создания инсталляции приложения будет использована программа Smart InstallMaker.

1.5 Требования к программному обеспечению и техническим средствам

Для корректной и комфортной разработки игрового приложения необходимо, чтобы компьютер удовлетворял следующим минимальным системным требованиям:

* ОС Windows 7/8/10;
* процессор 2 GHz и выше;
* 512 MB RAM;
* свободное пространство на диске: 5 ГБ и больше;
* дисплей SVGA с разрешением не менее 800х600 точек на дюйм;
* клавиатура и мышь.

Для разработки приложения на компьютере должно быть установлено следующее программное обеспечение:

* DirectX 11.0c;
* Visual Studio 2017;
* среда разработки Unity5.

Для обеспечения корректной работы игрового приложения, необходимо чтобы компьютер был оснащен:

* процессором Intel Core 2 Duo 2, 2 GHz;
* оперативной памятью не менее 128 МБ;
* дисплеем не менее 1024х768 точек;
* видеокартой с поддержкой PixelShader 3.0;
* местом на жестком диске 100 МБ;
* манипулятором «мышь»;
* клавиатурой.

1.6 Проектирование и моделирование игрового приложения

1.6.1 Сценарий и стратегия игры

Главный персонаж умирает в реальном мире и переносится в бездну, место, куда попадают души, имеющие шанс на возрождение. Главному герою и его противникам даётся несколько очков жизни, и чтобы возродиться нужно при помощи магических снарядов вытянуть очки жизней из всех противников.

Стратегия игры заключается в том, чтобы победить всех противников запуская в них снаряды и уворачиваться от вражеских снарядов.

1.6.2 Функции персонажей

В приложении будет два типа персонажей:

* герой, управлять которым будет пользователь;
* противники, управление которыми будет осуществляться компьютером.

У героя будут следующие состояния:

* бездействие;
* атака;
* передвижение.

У героя и противников возможны следующие функции:

* возродиться;
* нанести урон;
* получить урон;
* умереть;
* восстановить здоровье;
* сместиться вправо, влево, вверх или вниз.

1.6.3 Графика

Разработка и проработка графических объектов игрового приложения разделена на следующие направления:

* прорисовка героя и его действий (рисунок 1).
* прорисовка врагов (рисунок 2);
* прорисовка игровой локации (рисунок 3);
* прорисовка снарядов (рисунок 4).

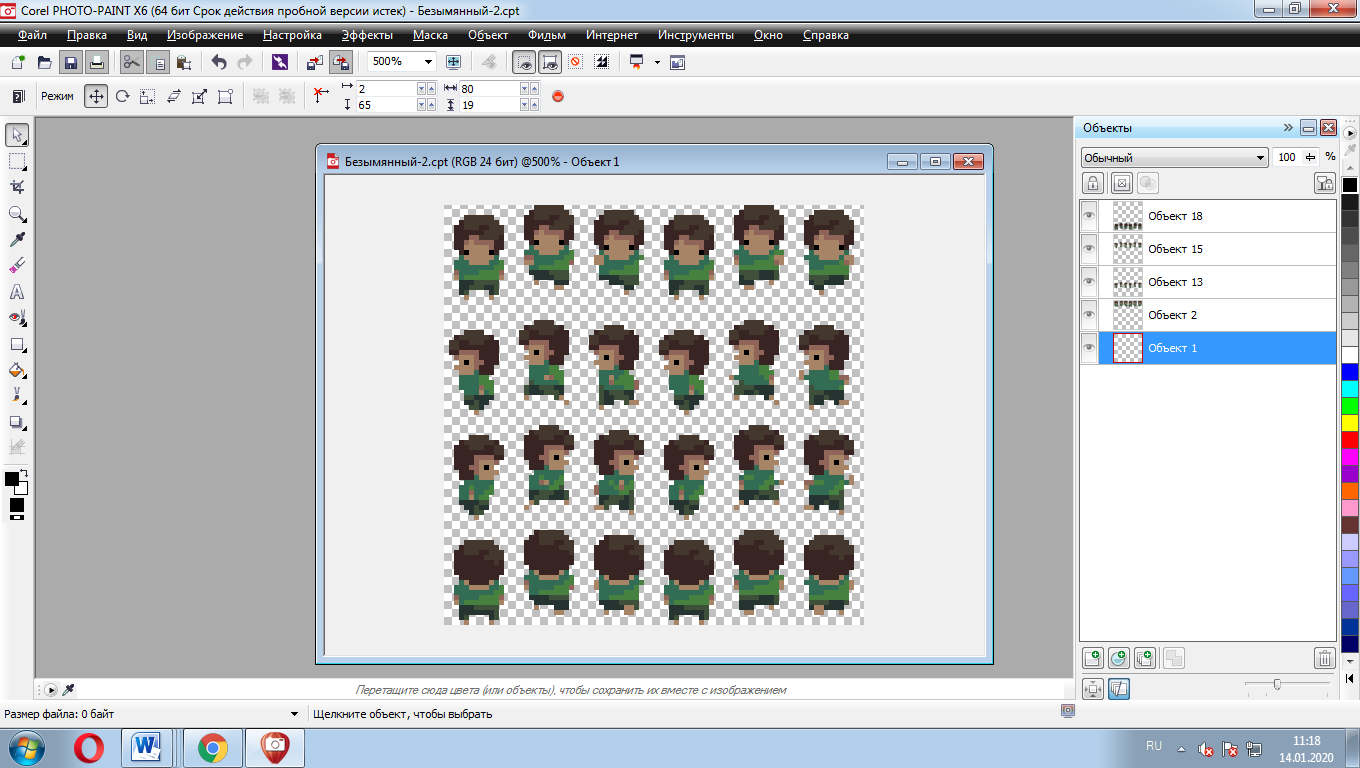


Рисунок 1 – Анимация персонажа

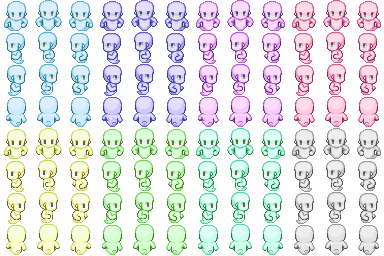


Рисунок 2 – Анимация врагов



Рисунок 3 – Игровая локация

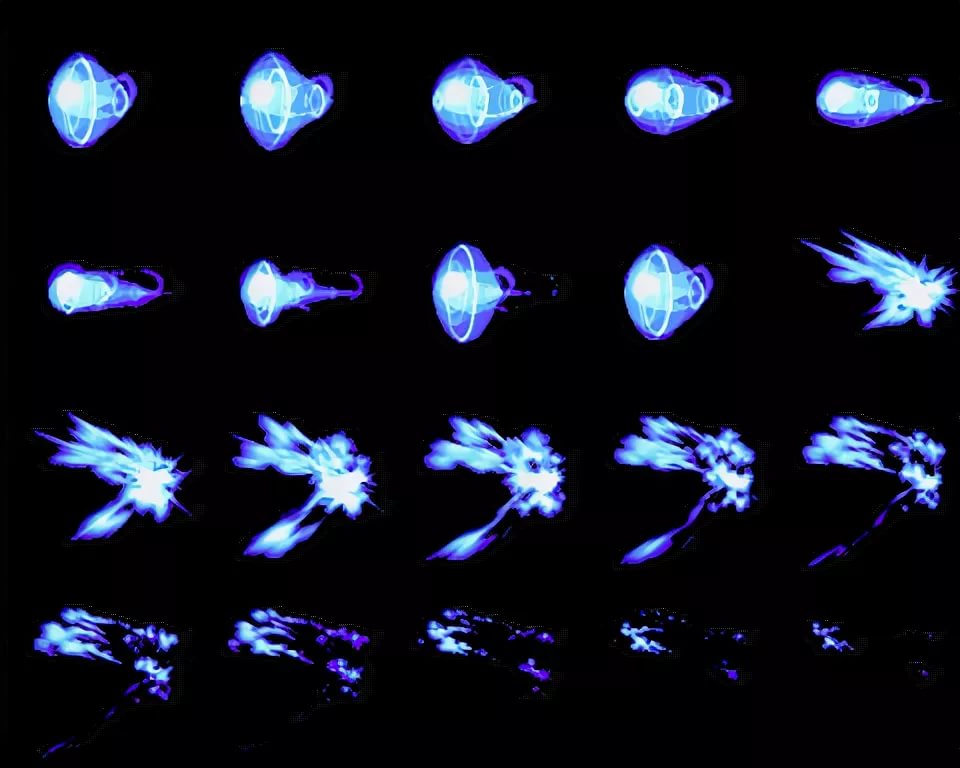


Рисунок 4 – Анимация снаряда

1.6.4 Основные функции игрового приложения

Игровое приложение будет иметь следующие функции:

* возможность начать новую игру;
* установка на паузу;
* рестарт игры;
* возможность изменить качество графики;
* возможность изменить уровень громкости звуков;
* подсчёт количества очков.

Новая игра, пауза, настройка графики и рестарт будут реализованы в виде кнопок. Количество очков будет считаться с помощью счетчика, и выводиться в конце игры. Результат будет сохраняться в файлах игры.

2 РАБОЧИЙ ПРОЕКТ

2.1 Стандартные объекты игрового приложения

На основе технического проекта было разработано игровое приложение «Lost in the abyss» для ОС Windows.

Для создания игры была использована среда разработки VisualStudio 2019, язык программирования С# и движок Unity5, а также ресурсы, взятые из открытых источников.

В программе используются следующие системные классы С#:

* System – содержит фундаментальные и базовые классы, определяющие часто используемые типы значений и ссылочных данных, события и обработчики событий, интерфейсы, атрибуты и исключения обработки;
* System.Collections – содержит функции для работы с массивами, списками и прочими коллекциями;
* System.Collections.Generic – системный класс, предоставляющий функционал для работы с динамическим типом. [4]

В движке Unityприменяется объектно-ориентированный подход, при котором используется множество классов, каждый объект является сочетанием множества скриптов, которые добавляют им функциональности и могут подключаться и отключаться по требованию.

В программе используются следующие классы, предоставляемые движком Unity:

* UnityEngine.SceneManagement – класс, позволяющий управлять сценами, переключаться между ними, перемещать объекты между сценами и многое другое;
* UnityEngine – содержит основные функции для работы с движком;
* UnityEngine.UI – предоставляет типы и функции для работы с графическим интерфейсом игрового приложения. [2]

Самые часто используемые методы:

* void Start() – вызывается во фрейме, когда сценарий включен непосредственно перед тем, как любой из методов Update вызывается в первый раз;
* void Update() – вызывает каждый кадр, если MonoBehaviour включен;
* public Component GetComponent(Type type) – возвращает компонент Type компоненту type, если он прикреплен к игровому объекту, и NULL, если его нет;
* void OnCollisionEnter2D (Collision2D coll) –передается, когда входящий коллайдер контактирует с коллайдером данного объекта (только 2D физика). [2]

Среди встроенных модификаторов игровых объектов движка Unity можно выделить следующие основные компоненты:

* SpriteRenderer – компонент, отвечающий за отрисовку двумерных картинок (спрайтов) на экране;
* Animator – компонент, позволяющий создавать анимации и менять картинки в соответствии с состоянием объекта;
* Rigidbody – компонент, отвечающий за физические свойства объекта и влияние на него гравитации;
* Collider – создает границы столкновения объекта с другими объектами. [2]

2.2 Пользовательские объекты игрового приложения

Сцены в Unity содержат объекты игры. Они могут использоваться для создания главного меню, отдельных уровней и для других целей. Можно считать каждый файл сцены отдельным игровым уровнем. В каждой сцене можно разместить объекты окружения, заграждения, декорации, по кусочкам создавая дизайн и саму игру.

В игре есть две основные сцены:

* главноеменю;
* уровень.

Стартовый экран состоит из трех пунктов:

* Играть;
* Настройки;
* Выход.

В проекте имеется только один уровень. Он содержит игровой мир и врагов.

Сцены являются аналогами окон в Windows и содержат в себе так называемые игровые объекты (GameObject), к которым могут применяться различные модификаторы поведения и которые могут взаимодействовать друг с другом с помощью привязанных к ним модификаторов.

Для добавления особой функциональности игровым объектам были созданы следующие скрипты:

* GameManager –контроллер игры;
* PlayerController – контроллер героя (приложение А);
* PlayerStats – Статистика игрока;
* MainMenu–контроллер главного меню;
* Destroy– уничтожение препятствий для предотвращения их накопления;
* DontDestroy – предназначен для того, чтобы игровой объект не уничтожался при загрузке новой сцены;
* PlayerShooting – скрипт для запуска снарядов игрока;
* EnemyShooting – скрипт для запуска снарядов врагов (приложение А)
* PlayerProjectile – поведение снарядов игрока;
* EnemyProjectile – поведение снарядов врагов;
* CameraMove – перемещение камеры.

Контроллер героя (класс PlayerController) организует перемещение персонажа в игровом пространстве.

Переменная класса float speed хранит скорость персонажа.

Методы класса:

* Start() – получает информацию о твердом теле и анимации, запускает бег;
* FixedUpdate() – запускает Методы TakeInput() и Move();
* TakeInput() – считывает нажатие клавиш;
* Move () – отвечает за передвижение персонажа;
* SetAnimatorMovement() – отвечает за анимацию бега.

Запуск снаряда врагом (класс EnemyShooting) организует стрельбу врагов по игроку.

Методы класса:

* Start() – находит игрока на уровне и определяет частоту выстрелов;
* ShootPlayer() – определяет точное положение игрока относительно противника и производит выстрел по игроку.

2.3 Установка и настройка игрового приложения

Для установки программы необходимо запустить файл инсталляции setup.exe. В результате в выбранном каталоге будет создана папка Lost in the abyss, в которой будут размещены исполняемы файлы (Lost in the abyss.exe, UnityCrashHandler32.exe, UnityCrashHandler64.exe), библиотека (UnityPlayer.dll) и служебные каталоги (MonoBleedingEdge, Lost in the abyss\_Data).

Также на рабочем столе будет создан ярлык для запуска приложения. Ярлык для деинсталляции программы будет размещен в папке программы.

Для создания инсталляции была использована программа Smart Install Maker 5.04. В ходе инсталляции пользователю показываются следующие окна:

* выбор языка установки;
* выбор папки установки;
* создание дополнительных ярлыков – позволяет создать ярлык на рабочем столе или отказаться от создания ярлыка;
* ход выполнения процесса установки;
* завершение установки – позволяет выбрать файлы, которые нужно запустить после установки.

После запуска программы инсталляции нужно следовать указаниям в появляющихся окнах.

2.4 Работа с игровым приложением

Для запуска игрового приложения необходимо щёлкнуть на ярлык игры Lostintheabyss.exe. После запуска приложения открывается главное меню (рисунок 5).



Рисунок 5 – Главное меню приложения

При нажатии в главном меню на кнопку «Start» (Старт), начнётся новая игра (рисунок 6).

Для перемещения персонажа используются клавиши «W», «A», «S», «D».

Главный персонаж может стрелять по врагам, для этого необходимо нажать левую кнопку мыши.

При нажатии на клавишу «Escape» откроется меню паузы (рисунок 7).



Рисунок 6 – Игровой уровень

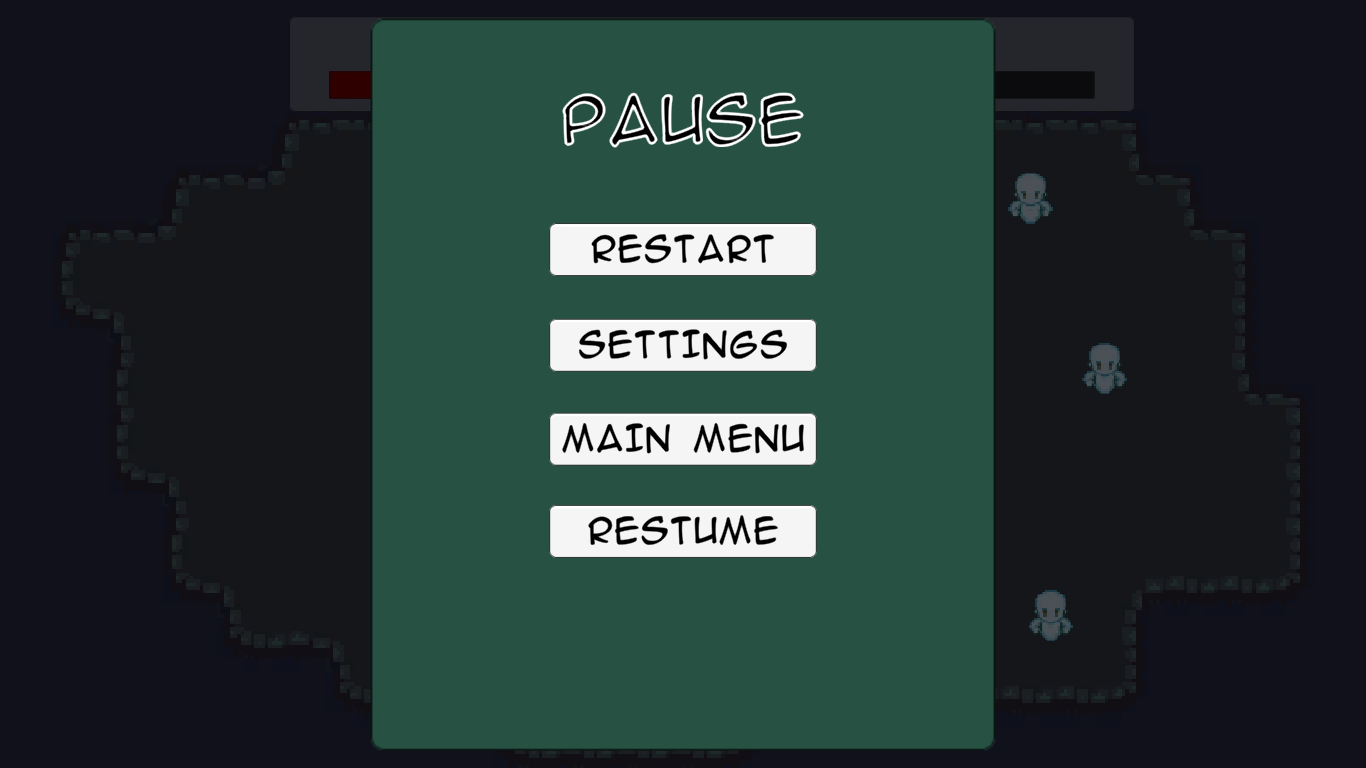


Рисунок 7 – Меню паузы

При этом игра останавливается и предлагается закрыть меню паузы (Restume), выйти в главное меню (Main menu), начать уровень заново (Restart), либо зайти в настройки (Settings).

В меню настройки (рисунок 8) можно изменить разрешение (Resolution), включить полноэкранный режим (FullScreen), настроить громкость музыки (Music) и звуков (SFX) и посмотреть управление (Control).

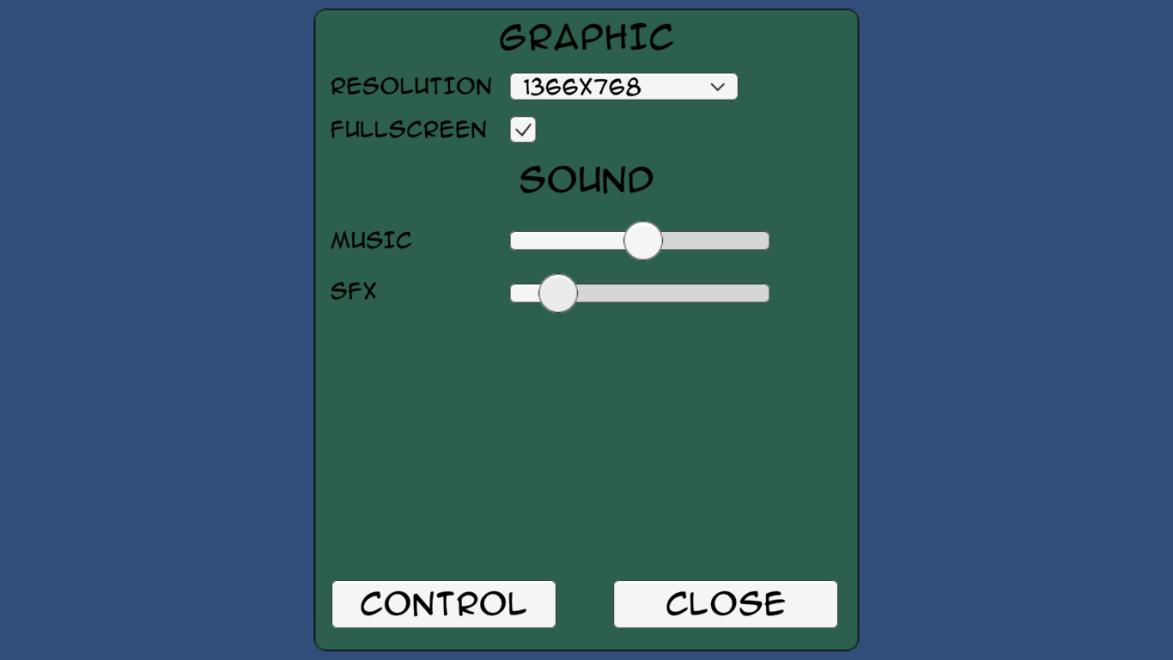


Рисунок 8 – Меню настройки

Чтобы закончить уровень необходимо победить всех врагов или умереть.

ЗАКЛЮЧЕНИЕ

В результате выполнения данного курсового проекта была разработана игровая программа «Lost in the Abyss» в жанре Roguelike для ОС Windows, а также разработано документальное сопровождение программы, включающее в себя технический и рабочий проекты.

Данная программа включает в себя главное меню, игровую зону и руководство пользователя. Она позволяет хорошо провести время и потренировать реакцию. Удобный интерфейс программы позволяет легко ориентироваться в приложении, не требуя от пользователя каких-либо специальных навыков.

Приложение обеспечивает следующие возможности:

* просмотр управления;
* перезапуск игры;
* установку игры на паузу;
* настройка громкости звуков и музыки, разрешения экрана;
* оконный режим работы.

Основными достоинствами программы является:

* наличие простого и удобного интерфейса;
* данный программный продукт будет предназначен для широкого круга пользователей;
* имеет инсталляцию;
* наличие музыкального сопровождения.

В ходе выполнения курсового проекта были достигнуты поставленные цели и решены следующие задачи:

* изучены средства разработки компьютерных игр на Windows;
* выбран жанр для игрового приложения;
* спроектирована структура и функциональность игры;
* спроектированы история, персонажи и уровни;
* выбраны средства реализации;
* определены минимальные требования к техническим и программным средствам;
* описаны входные и выходные данные;
* разработан программный код приложения;
* создана инсталляция.

К недостаткам данного игрового приложения можно отнести однообразный геймплей и малое количество препятствий, в будущем их планируется исправить.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Антонов Л.С., Кадомцев С.Б. Api функции. М.: Просвещение, 2015 г. 336с.
2. Биллиг В. Основы программирования на С#. СПб.: БХВ-Петербург, 2015.
3. Жарков В. А. «Компьютерная графика, мультимедиа и игры на Visual C# 2005». М.: «Жарков пресс», 2015.
4. МэйнС. Основы Windows Communication Foundation для .NET Framework 3.5. СПб.: БХВ-Петербург, 2015.
5. Петров А.В. Учебник юного программиста. М.: Просвещение, 2015г. 290 с.
6. РихтерД. CLR via C#. Программирование на платформе Microsoft.NET Framework 4.5 на языке C#. СПб.: Питер, 2017.
7. Сибраро П., Клайс К., Коссолино Ф., Грабнер Йохан. WCF4: Windows Communication Foundation и .NET 4. СПб.: БХВ-Петербург, 2015.
8. Торн А. Основы анимации в Unity. М.: ДМК, 2016. 176с.
9. Троелсен Э. Язык программирования С# 5.0 и платформа .NET 4.5.СПб.: БХВ-Петербург, 2017.
10. Хокинг Д. Unity в действии. Мультиплатформенная разработка на C#. СПб.: Питер, 2016. 336 с.
11. Библиотека классов .NET Framework. URL: <https://msdn.microsoft.com/ru-ru/library/mt472912(v=vs.110).aspx> (дата обращения: 11.10.2019).
12. Официальная документация Unity. URL: https://docs.unity3d.com/ru (дата обращения: 10.12.2019).

ПРИЛОЖЕНИЕ А

Листинг скриптов

EnemyShooting.cs

public class EnemyShooting : MonoBehaviour

{

public GameObject projectile;

GameObject player;

public float minDamage;

public float maxDamage;

public float projectileForce;

public float minCool;

public float maxCool;

public float smesh;

float cooldown;

private void Start()

{

player = GameObject.FindGameObjectWithTag("Player");

cooldown = Random.Range(minCool, maxCool);

StartCoroutine(ShootPlayer());

}

IEnumerator ShootPlayer()

{

yield return new WaitForSeconds(cooldown);

if (player != null)

{

GameObject spell = Instantiate(projectile, transform.position, Quaternion.identity);

spell.GetComponent<EnemyProjectile>().enemyName = gameObject.name;

Vector2 myPos = transform.position;

Vector2 playerPos = player.transform.position;

playerPos.y -= smesh;

Vector2 direction = (playerPos - myPos).normalized;

spell.GetComponent<Rigidbody2D>().velocity = direction \* projectileForce;

var angle = Vector2.Angle(Vector2.right, playerPos - myPos);

spell.transform.eulerAngles = new Vector3(0f, 0f, transform.position.y < playerPos.y ? angle : -angle);

spell.GetComponent<EnemyProjectile>().damage = Random.Range(minDamage, maxDamage);

StartCoroutine(ShootPlayer());

}

}

}

PlayerController.cs

public class PlayerController : MonoBehaviour

{

public float speed;

private Vector2 direction;

private Animator animator;

private void Start()

{

animator = GetComponent<Animator>();

}

private void FixedUpdate()

{

TakeInput();

Move();

}

private void Move()

{

transform.Translate(direction.normalized \* speed);

if (direction.x != 0 || direction.y != 0)

{

SetAnimatorMovement(direction);

}

else

{

animator.SetLayerWeight(1, 0);

}

}

private void TakeInput()

{

direction = Vector2.zero;

if(Input.GetKey(KeyCode.W)) direction += Vector2.up;

if (Input.GetKey(KeyCode.A)) direction += Vector2.left;

if (Input.GetKey(KeyCode.S)) direction += Vector2.down;

if (Input.GetKey(KeyCode.D)) direction += Vector2.right;

}

private void SetAnimatorMovement(Vector2 direction)

{

animator.SetLayerWeight(1, 1);

animator.SetFloat("xDir", direction.x);

animator.SetFloat("yDir", direction.y);

}

}