МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«**Вятский государственный университет**»

**(ФГБОУ ВПО «ВятГУ»)**

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Допущено к защите

Руководитель проекта

Преподаватель

\_\_\_\_\_\_\_\_\_\_( Коржавина А.С.)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_2019г.

РАЗРАБОТКА двухмерной ИГРЫ - пРИКЛЮЧЕНИЕ

Пояснительная записка курсового проекта по дисциплине

«Компьютерная графика»

ТПЖА.230200.447 ПЗ

Разработал студент группы ИВТ-22 \_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Заев Е.В./

Руководитель

преподаватель кафедры ЭВМ \_\_\_\_\_\_\_\_\_\_\_\_\_/Коржавина А.С./

Проект защищен с оценкой «\_\_\_\_\_\_\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оценка) (дата)

Члены комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

(подпись) (Ф.И.О)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

Киров 2019

**Оглавление**

[1. Использование компьютерной графики в двухмерных играх 6](#_Toc41391091)

[1.1 Графические изображения в двухмерных играх 7](#_Toc41391092)

[1.2 Движение объектов на экране 7](#_Toc41391093)

[1.3 Анимирование спрайтов 8](#_Toc41391094)

[1.4 Столкновение игровых объектов 8](#_Toc41391095)

[1.5 Использование слоёв 9](#_Toc41391096)

[1.6 Эффект сдвига слоёв 9](#_Toc41391097)

[1.7 Система частиц 9](#_Toc41391098)

[2. Анализ существующих сред разработки 11](#_Toc41391099)

[3 Обоснование актуальности темы 15](#_Toc41391100)

[4 Расширенное техническое задание 16](#_Toc41391101)

[4.1 Управление 16](#_Toc41391102)

[4.2 Правила игры 16](#_Toc41391103)

[4.3 Системные требования 17](#_Toc41391104)

[4.4 Программное обеспечение 17](#_Toc41391105)

[5 Схема взаимодействия функциональных блоков 18](#_Toc41391106)

[5.1 Пользовательский интерфейс 18](#_Toc41391107)

[5.2 Игровая логика 18](#_Toc41391108)

[5.3 Отображение игрового процесса 18](#_Toc41391109)

[6 Разработка диаграммы классов 19](#_Toc41391110)

[7 Разработка алгоритмов 20](#_Toc41391111)

[7.1 Главное меню 20](#_Toc41391112)

[7.2 Окно туториала 21](#_Toc41391113)

[7.3 Сцена уровня 22](#_Toc41391114)

[Заключение 23](#_Toc41391115)

[Приложение 24](#_Toc41391116)

[Исходный код 24](#_Toc41391117)

[Эффект сдвига слоёв 24](#_Toc41391118)

[Движение объектов в игре 24](#_Toc41391119)

[Персонаж под управлением игрока 25](#_Toc41391120)

[Персонаж под управлением компьютера 27](#_Toc41391121)

[Скрипт очков здоровья персонажа 27](#_Toc41391122)

[Скрипт окончания игры 28](#_Toc41391123)

[Стрельба 29](#_Toc41391124)

[Оружие и перезарядка 29](#_Toc41391125)

[Библиографический список 31](#_Toc41391126)

**Введение**

Во всём мире любят электронные развлечения и в частности видеоигры. Эта индустрия развивается с каждым годом всё активнее и догнала по прибыли кинематограф. В наше время для любого возраста есть свои игры, геймдизайнеры разрабатывают их, основываясь на целевую аудиторию. Создание игры зависит от навыков, умений и таланта людей многих профессий: художников, сценаристов, композиторов, но важнейшую роль в этом играют программисты.

Причиной разработки проекта стало активное развитие индустрии видеоигр. Видеоигровой рынок с каждым годом растёт. Например в 2019 году по данным сайта Newzoo.com(международная маркетинговая компания специализирующаяся на игровом рынке), занимающимся аналитикой в сфере игр и киберспорта, игровая индустрия заработала 152.1 миллиарда долларов, что на 13.4 миллиарда больше, чем в 2018 году[1]. Отсюда следует, что существует производственная необходимость выпуска приложения на активно развивающемся рынке.

Актуальность написанной работы заключается в том, чтобы изучить возможности создания современных двумерных приключенческих игр, выполненных в пиксельной графике.

В ходе курсовой работы необходимо изучить и проанализировать множество игр в жанре приключений и выбрать наиболее интересные механики и ситуации. На этой основе придумать свою игру. Также необходимо понять, как наиболее правильно интегрировать эти механики в игру. Чтобы привлечь аудиторию, нужно создать красивый внешний вид игры, весь игровой процесс украсить динамической фоновой музыкой, добавить различные бонусы и этим всем добиться, чтобы игра понравилась как можно большему количеству людей. В последнее время в играх стала популярна стилизация под ретро, поэтому все детали игры будут созданы при помощи простой пиксельной графики и также использованы нарисованные мной спрайты. Необходимо описать основные преимущества и недостатки использования пиксельной графики.

Целью курсовой работы является создание рабочего прототипа игры для людей любого возраста, вне зависимости от наличия доступа к сети Интернет, а также с минимальной зависимостью от используемой платформы. Сейчас мною планируется выпуск только на персональных компьютерах.





# **Использование компьютерной графики в двухмерных играх**

Двухмерная компьютерная графика классифицируется по типу представления графической информации. Обычно компьютерную графику разделяют на векторную и растровую, хотя выделяют ещё и фрактальный тип представления изображений.

Векторная графика представляет изображение как набор геометрических примитивов. Обычно в качестве них выбираются точки, прямые, окружности, прямоугольники и кривые. Объектам присваиваются некоторые атрибуты, например, толщина линий или цвет. Рисунок хранится как набор координат, векторов и других чисел, характеризующих набор примитивов. При воспроизведении перекрывающихся объектов имеет значение их порядок.

Растровая графика всегда оперирует двумерным массивом точек. Каждой точке сопоставляется значение яркости, цвета и прозрачности. Растровый образ имеет некоторое число строк и столбцов. В растровом виде представимо любое изображение, однако этот способ хранения имеет свои недостатки: больший объём памяти, необходимый для работы с изображениями, потери информации при редактировании.

Фрактальная графика построена на объекте, отдельные элементы которого наследуют свойства родительских структур. Поскольку более детальное описание элементов меньшего масштаба происходит по простому алгоритму, описать такой объект можно всего лишь несколькими математическими уравнениями. Фракталы позволяют описывать целые классы изображений, для детального описания которых требуется относительно мало памяти. С другой стороны, фракталы слабо применимы к изображениям вне этих классов.

## **Графические изображения в двухмерных играх**

Для отображения графических объектов в двухмерных играх используются спрайты и тайлы. Спрайтом называется растровое изображение, которое можно отобразить на экране. Спрайт представляет из себя готовое изображение, какого либо игрового объекта: персонажа, детали интерьера, игрового предмета, снаряда и т.п.[2]. Тайлом называется небольшое растровое изображение одинаковых размеров из которых могут быть собраны большие изображения. Тайлы активно стали применятся из-за нехватки аппаратных ресурсов игровых систем и компьютеров для вывода на экран больших изображений, но производительности хватало, для того, чтобы вывести на экран большое количество небольших изображений из которых можно было составить крупные игровые объекты: задний фон уровней, особо крупные игровые объекты: транспортные средства, боссы, платформы. Так-же тайлы позволяли экономить память на носителях информации, т.к. в памяти приходилось держать одно небольшое изображение [3].

## **1.2 Движение объектов на экране**

Изначально двухмерные игры состояли из одного экрана, на котором двигались спрайты игровых объектов. Это достигалось за счёт того, что у каждого игрового объекта было свое положение на экране по высоте и ширине. При изменении этих параметров, игроку казалось, что объект движется. Например, Тетрис состоит из одного экрана, по которому передвигаются фигуры из квадратов. В определённый момент игрокам стало мало одного игрового экрана. Вышла игра Pitfall, где была реализована смена игровых экранов. Если игровой персонаж под управлением игрока подходил к краю экрана, то все игровые объекты на экране удалялись и затем отрисовывался другой игровой экран уже со своими объектами. Такие игры стали пользоваться популярностью, т.к. это позволяло делать большие и разнообразные игровые уровни. Следующая революция произошла в игре Super Mario Bros. где присутствовало плавное движение игрового экрана вслед за игровым персонажем. Для этого необходимо было отрисовывать тайлы, когда они появляются на экране и удалять их, как только пропадают из поля зрения игрока.

## **1.3 Анимирование спрайтов**

Несмотря на то, что игровые объекты на экране двигаются, они представляли из себя статичные изображения. Если спрайты не будут изменятся со временем, игра мало кого сможет привлечь. У фигурок при перемещении должны двигаться ноги, а у деревьев шевелится листва. Для того, чтобы реализовать это в двухмерной графике используют метод из мультипликации, при котором быстро сменяющиеся кадры кажутся зрителю движущимися. Для того, чтобы реализовать идущего персонажа, придётся нарисовать несколько спрайтов этого персонажа, немного отличающиеся друг от друга. Если эти спрайты будут быстро сменять друг друга, то у игрока сложится впечатления, что персонаж бежит.

## **1.4 Столкновение игровых объектов**

При создании игры, часто приходится отслеживать сталкиваются игровые объекты или нет. Например стоит ли игрок на платформе или нет. Если персонаж не стоит на платформе, то падает вниз. Для этого нужно отслеживать сталкиваются ли игровые объекты. Для этого создаётся невидимый четырёхугольник, который накладывается на игрового объект, и регистрирует столкновения с другими такими же четырёхугольниками.[4] Например, если игрок видит, как снаряд персонажа попадает в противника, он видит лишь движение спрайтов, а для компьютера это столкновение двух невидимых четырёхугольников, после которого происходит определённое действие.

## **1.5 Использование слоёв**

Двухмерные игры представляют из себя различные слои с располагающимися на них объектами. Слои используются для того, чтобы отделять изображения располагающиеся на переднем плане, от изображений на заднем плане. Например, если два спрайта пересекаются на экране, то поверх рисуется тот, который находится на ближайшем слое.

## **1.6 Эффект сдвига слоёв**

Эффект сдвига слоёв уже давно применяется в двухмерных играх, ещё его называют эффектом параллакса. Заключается он в том, что чем ближе объект расположен к наблюдателю, тем быстрее он движется. Например, если персонаж игрока бежит на переднем плане, то кусты и камни расположенные на нём движутся очень быстро в противоположную сторону. Деревья, расположенные на среднем плане, движутся в противоположную от персонажа игрока сторону чуть медленнее, а горы на заднем плане движутся ещё медленнее. Получившийся эффект параллакса визуально выглядит очень красиво и полюбился многим игрокам.

## **1.7 Система частиц**

Для изображения объектов не имеющих чётких геометрических границ (туман, облака, дым, врыв, огонь и дождь) в компьютерной графике используют систему частиц. Система частиц состоит из определённого (фиксированного или произвольного) количества частиц. Математически каждая частица представляется как материальная точка с дополнительными атрибутами, такими как внешний вид, скорость, ориентация в пространстве, угловая скорость, и т. п. В ходе работы программы каждая частица изменяет своё состояние по определённому, общему для всех частиц системы, закону. Например, частица может подвергаться воздействию гравитации, менять размер, цвет, скорость и так далее, и, после проведения всех расчётов, частица визуализируется [5].



1. **Анализ существующих сред разработки**

Среды разработки видеоигр делятся на свободно распространяемые и проприетарные. Также они могут быть мультиплатформенными или рассчитанными только для одной платформы. Мне нужно проанализировать программы для разработки игр и выбрать одну наиболее подходящую моим требованиям:

* свободно распространяемая лицензия;
* простота работы с платформой;
* возможность создания двухмерной игры;
* простота портирования игры на различные электронные устройства (смартфоны, игровые приставки, компьютеры).

Для реализации игры рассматривались следующие программы:

* Cry Engine;
* Corona;
* Unreal Engine;
* Unity 3D;
* Godot Engine;
* Cocos 2D-X.

Cry Engine – среда разработки созданная компанией Crytek в 2002 году. На текущий момент последней считается пятая версия этой платформы разработки. Это программа свободно не распространяется и позволяет создавать только трёхмерные игры[6].

Среда для разработки игр Corona направлена на разработку 2D игр (Zip Zap, Designer City). Использует язык Lua. Преимущество данного продукта – кроссплатформенность, что позволяет портировать игру на Android и IOS. Но присутствуют и недостатки. Программа подходит только для создания двумерных игр, 3D составляющая отсутствует. В дальнейшем создание подобных игр на нём будет невозможно[7].

Godot Engine – открытая кроссплатформенная среда для разработки двухмерный и трёхмерных проектов. Игры создаются на языке C++ или используется встроенный динамически типизированный язык GDScript. Godot имеет собственный редактор скриптов со многими возможностями. Среда разработки абсолютно бесплатная, но и обладает недостатками[8]. Она не подходит для серьёзных проектов, так как в большом количестве функций уступает конкурентам (Unity 3D, Unreal Engine).

Unreal Engine среда разработки появилась в 1998 году. Была разработана компанией Epic Games для того, чтобы составить конкуренцию id Tech 2 от компании id Software, разработанный годом ранее. Благодаря успеху серии игр Unreal Tournament за авторством тех же Epic Games, Unreal Engine активно развивался и имеет на данный момент уже четыре версии и готовится к выходу пятая. Раньше он был платным, но 2 марта 2015 года Unreal Engine 4 стал бесплатным при условии, что, если доходы игры превысят 3000 долларов за квартал, 5% от этих доходов должна будет получить компания Epic Games. Данный движок хорошо показывает себя в создании трёхмерных игр и поддерживает большое количество технологий: HDR, PhysX, FaceFX, DirectX. Использует язык C++ и позволяет создавать игры под операционные системы Windows и Mac, iOS и Android, а также множество стационарных и портативных игровых приставок. Также этот движок нашёл применение в кинематографе при создании спецэффектов[9].

Unity – платформа для создания видеоигр вышла в свет в 2005 году, и с тех пор активно развивается. Несмотря на то что Unity позволяет работать с трёхмерными объектами, но большую долю игр, созданных на этой платформе, составляют двухмерные игры. Unity обладает удобными инструментами для создания 2D игр и возможностью создавать различные эффекты: эффекты частиц, освещения, анимированные спрайты. Недостатком этой среды разработки является то, что она платная. Есть конечно и бесплатная версия. Она ограничена в создании трёхмерных игр, но очень хорошо подходит для создания двумерных игр[10]. Unity большую долю своей популярности приобрела благодаря кроссплатформенности; игры, созданные на Unity, легко могут быть перенесены на множество операционных систем, смартфонов и игровых приставок.

Проект Cocos 2D-X появился в 2010 году. Изначально задумывался для разработки под мобильные устройства, поэтому обладает высокой кросплатформенностью (Windows, Linux, OS X, iOS, Android). Cocos2d-x позволяет писать на таких языках, как С++, Lua, Javascript. Работа в данной среде разработки основана на использовании спрайтов. Над ними можно совершать различные действия: перемещать, переворачивать, масштабировать и т.д.. Фреймворк позволяет использовать эффекты частиц и применение шейдерных фильтров[11].

Таблица 2 – Анализ среды разработки

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Открытый доступ | Простота портирования | 2D проекты | 3D проекты | Поддержка нескольких языков | Возможность создавать простые проекты |
| Cry Engine | + | + | - | + | - | - |
| Corona | + | + | + | - | - | + |
| Unreal Engine | +/- | + | + | + | - | +/- |
| Unity 3D | +/- | + | + | + | + | +/- |
| Godot Engine | + | + | + | +/- | +/- | + |
| Cocos 2D-X | + | + | + | - | + | + |

Проанализировав движки, было принято решение разрабатывать игру на Unity, т.к. она соответствует моим требованиям:

Открытый доступ, простота портирования на множество платформ и возможность работы с двухмерной графикой.



# **Обоснование актуальности темы**

Разработка игр ставит высокие требования к навыкам работы с компьютерной графикой. Это касается как двухмерных, так и трёхмерных игр. Каждый год выходят новые программы и технологии для работы с компьютерной графикой. Современные игроки выставляют высокие требования к внешнему виду игр: как к дизайну, так и к технологиям. Даже разрабатывая простую игру, мы совершенствуем свои знания и умения в области компьютерной графики, а людей, которые профессионально занимаются разработкой игр, можно назвать экспертами в области компьютерной графики.

# **Расширенное техническое задание**

Требования к функциям программы:

* Понятный и простой интерес;
* Динамичный задний фон с эффектом сдвига слоёв;
* Анимированные спрайты персонажа и противников;
* Сбор статистики при прохождении этапов;
* Реализация возможности делать главного персонажа (под управлением игрока) сильнее и развивать его;
* Система бонусов;
* Боссы (сильные противники со специфичным поведением);
* Пользовательский интерфейс:

В сцене «Меню» присутствуют 2 кнопки: «Старт» и «Выход».

* Кнопка «Старт» перенаправляет пользователя в самый первый этап, где пользователю предоставляется возможность ознакомится с управлением;
* Кнопка «Выход» закрывает приложение;

## **4.1 Управление**

Для управления в игре требуется клавиатура + мышь. Для перемещения персонажа предназначены клавиши: LeftArrow, RightArrow, UpArrow, DownArrow; Для стрельбы предназначена клавиша: Z; Для использования способностей предназначена клавиша: X;

## **4.2 Правила игры**

В начале игрок появляется на стартовой локации. Игрок движется вслед за игровым экраном подбирает различные предметы, которые его усилят. Побеждает враждебно настроенных противников, в конце уровня он попадает на этап с боссом.

## **4.3 Системные требования**

* Операционная система – Windows XP или выше, либо дистрибутив OC Linux;
* Процессор – любой процессор с тактовой частотой выше 1.7 GHz;
* Оперативная память – 512 Mb и выше;
* Любой интегрированный или дискретный графический процессор;
* Клавиатура, мышь, монитор;

## **4.4 Программное обеспечение**

Для запуска игры на устройствах не потребуется ничего кроме установочного файла. Сам же файл будет устанавливать необходимые файлы и библиотеки и файлы, которые будут требоваться для запуска продукта. Для Пк версии возможно будет требоваться DirectX 9, который поддерживается на Windows Xp, Vista, 7, 8,10. На более ранних версиях windows игра может не пойти если будет использоваться данное программное обеспечение.

## **Схема взаимодействия функциональных блоков**

Разработанное приложение было разбито на несколько функциональных блоков, которые показаны на рисунке 1.



Рисунок 1 – Схема взаимодействия функциональных блоков.

## **5.1 Пользовательский интерфейс**

Все модули этого блока отвечают за отображение каких-либо данных об игре

## **5.2 Игровая логика**

Данный блок содержит алгоритмы программного представления и обработки текущей игровой сессии: реакция на действие игрока, взаимодействие с игровыми объектами.

## **5.3 Отображение игрового процесса**

Здесь находится части программы ответственные за рисование объектов на сцене, персонажа и противников.

## **Разработка диаграммы классов**

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок 3 – Диаграмма классов

## **Разработка алгоритмов**

## **7.1 Главное меню**

Позволяет начать или завершить работу с программой.



Рисунок 4 – Схема алгоритма главного меню.

## **7.2 Окно туториала**

Позволяет пользователю узнать управление персонажем и взаимодействие с игровыми объектами.

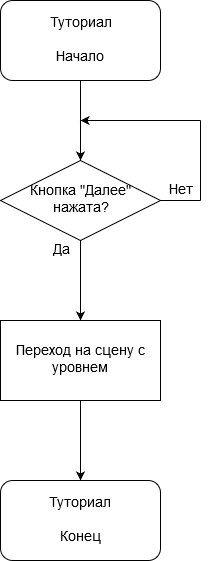


Рисунок 5 – Схема алгоритма туториала

## **7.3 Сцена уровня**

Позволяет перемещать персонажа, взаимодействовать с игровыми объектами и переходить на следующие уровни игры.

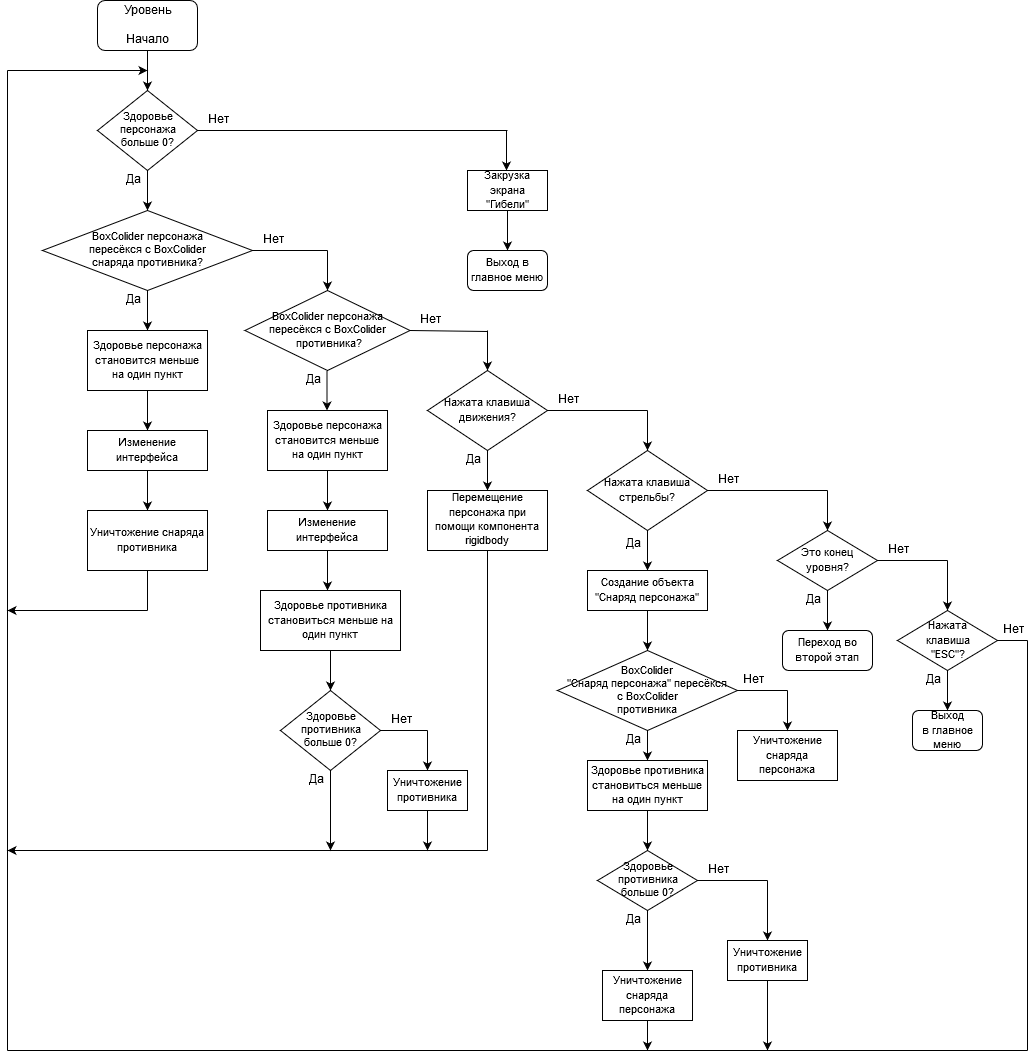


Рисунок 6 – Схема алгоритма уровня

# **Заключение**

Продукт получился основой для дальнейшего развития в данном направлении. При разработке игры был использован Unity, благодаря этому её можно будет реализовать не только на персональном компьютере, но и на портативных устройствах. Данная игра использует пиксельную графику и стилизована под игры четвёртого поколения игровых систем (16-ти битные игровые приставки). В игре использованы звуки и музыка из свободных источников, присутствует система очков, что позволяет соревноваться с друзьями. Это всё делает игровой процесс более увлекательным. Игра довольна проста в управлении, и в ней отсутствуют сцены жестокости, что позволяет её использовать любым игрокам, в том числе и детям.

Преимущества данной игры: яркая пиксельная графика, увлекательный игровой процесс и, вдобавок, приложение - мультиплатформенное.

Но продукт имеет и недостатки - это малое количество уровней с разнообразными противниками и боссами.

Для доработки проекта после его выхода можно будет добавить таблицу рекордов. Так игроки смогут соревноваться друг с другом по сети по всему миру.

# **Приложение**

## **Исходный код**

## **Эффект сдвига слоёв**

using UnityEngine;

public class ScrollingScript : MonoBehaviour

{

public Vector2 speed = new Vector2(2, 2);

public Vector2 direction = new Vector2(-1, 0);

public bool isLinkedToCamera = false;

void Update()

{

Vector3 movement = new Vector3(speed.x \* direction.x, speed.y \* direction.y, 0);

movement \*= Time.deltaTime;

transform.Translate(movement);

if (isLinkedToCamera)

{

Camera.main.transform.Translate(movement);

}}}

## **Движение объектов в игре**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class MoveScript : MonoBehaviour

{

public Vector2 speed = new Vector2(10, 10);

public Vector2 direction = new Vector2(-1, 0);

private Vector2 movement;

void Update()

{

movement = new Vector2(

speed.x \* direction.x,

speed.y \* direction.y); }



void FixedUpdate()

{

GetComponent<Rigidbody2D>().velocity = movement;

}}

## **Персонаж под управлением игрока**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class PlayerScript : MonoBehaviour

{

public int Score;

public Vector2 speed = new Vector2(50,50);

private Vector2 movement;

void Update()

{

float inputX = Input.GetAxis("Horizontal");

float inputY = Input.GetAxis("Vertical");

movement = new Vector2(

speed.x \* inputX, speed.y \* inputY);

bool shoot = Input.GetButtonDown("Fire1");

shoot |= Input.GetButtonDown("Fire2");

if (shoot)

{

WeaponScript weapon = GetComponent<WeaponScript>();

if (weapon != null)

{

weapon.Attack(false);

SoundEffectsHelper.Instance.MakePlayerShotSound();

}}

var dist = (transform.position - Camera.main.transform.position).z;

 var leftBorder = Camera.main.ViewportToWorldPoint(new Vector3(0, 0, dist)).x;

var rightBorder = Camera.main.ViewportToWorldPoint(new Vector3(1, 0, dist)).x;

var topBorder = Camera.main.ViewportToWorldPoint(new Vector3(0, 0, dist)).y;

var bottomBorder = Camera.main.ViewportToWorldPoint(new Vector3(0, 1, dist)).y;

transform.position = new Vector3(Mathf.Clamp(transform.position.x, leftBorder, rightBorder), Mathf.Clamp(transform.position.y, topBorder, bottomBorder),transform.position.z);

}

void Start()

{}

void Awake()

{}

void FixedUpdate()

{

GetComponent<Rigidbody2D>().velocity = movement;

}

void Destroy()

{}

void OnDestroy()

{

transform.parent.gameObject.AddComponent<GameOverScript>();

}

void OnCollisionEnter2D(Collision2D collision)

{

bool damagePlayer = false;

EnemyScript enemy = collision.gameObject.GetComponent<EnemyScript>();

if (enemy != null)

{

HealthScript enemyHealth = enemy.GetComponent<HealthScript>();

if (enemyHealth != null) enemyHealth.Damage(enemyHealth.hp);

damagePlayer = true;

 {

HealthScript playerHealth = this.GetComponent<HealthScript>();

if (playerHealth != null) playerHealth.Damage(1);

}}}}

## **Персонаж под управлением компьютера**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class EnemyScript : MonoBehaviour

{

private WeaponScript[]weapons;

void Awake()

{

weapons = GetComponentsInChildren<WeaponScript>();

}

void Update()

{

foreach (WeaponScript weapon in weapons)

{

if (weapon != null && weapon.CanAttack)

{

weapon.Attack(true);

}}}}

## **Скрипт очков здоровья персонажа**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class HealthScript : MonoBehaviour

{

public int hp = 1;

public bool isEnemy = true;

public void Damage(int damageCount)

{

hp -= damageCount;

if (hp <= 0)

{

SpecialEffectScript.Instance.Explosion(transform.position);

Destroy(gameObject);

}}

void OnTriggerEnter2D(Collider2D otherCollider)

{

ShotScript shot = otherCollider.gameObject.GetComponent<ShotScript>();

if (shot != null)

{

if (shot.isEnemyShot != isEnemy)

{

Damage(shot.damage);

Destroy(shot.gameObject);

}}}}

## **Скрипт окончания игры**

using UnityEngine;

public class GameOverScript : MonoBehaviour

{

void OnGUI()

{

const int buttonWidth = 120;

const int buttonHeight = 60;

if (GUI.Button(

new Rect(Screen.width/2 - (buttonWidth/2),(1\*Screen.height/3)-(buttonHeight/2),buttonWidth,buttonHeight), "Начать сначала!"))

{

Application.LoadLevel("lvl1");

}

 if (GUI.Button( new Rect( Screen.width/2 - (buttonWidth/2),(2 \* Screen.height/3) - (buttonHeight/2),buttonWidth,buttonHeight),"Назад в меню"))

{

Application.LoadLevel("Menu");

}}}

## **Стрельба**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class ShotScript : MonoBehaviour

{

public int damage = 1;

public bool isEnemyShot = false;

void Start()

{

Destroy(gameObject, 3);

}}

## **Оружие и перезарядка**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class WeaponScript : MonoBehaviour

{

public Transform shotPrefab;

public float shootingRate = 0.25f;

private float shootCooldown;

void Start()

{

shootCooldown = 0f;

}



void Update()

{

if (shootCooldown > 0)

{

shootCooldown -= Time.deltaTime;

 }}

public void Attack(bool isEnemy)

{

if (CanAttack)

{

shootCooldown = shootingRate;

var shotTransform = Instantiate(shotPrefab) as Transform;

shotTransform.position = transform.position;

ShotScript shot = shotTransform.gameObject.GetComponent<ShotScript>();

if (shot != null)

{

shot.isEnemyShot = isEnemy;

}

MoveScript move = shotTransform.gameObject.GetComponent<MoveScript>();

if (move != null)

{

move.direction = this.transform.right;

}}}

public bool CanAttack

{

get

{

return shootCooldown <= 0f;

}}}

# **Библиографический список**

1. Официальный сайт NewZoo [Электронный ресурс] URL: <https://newzoo.com/solutions/standard/market-forecasts/global-games-market-report/>
2. Страница Википедии о спрайтах в компьютерной графике [Электронный ресурс] URL: https://en.wikipedia.org/wiki/Sprite\_(computer\_graphics)
3. Страница Википедии о тайловой графике в видеоиграх [Электронный ресурс] URL: https://en.wikipedia.org/wiki/Tile-based\_video\_game
4. Страница Википедии об обнаружении столкновений [Электронный ресурс] URL: https://en.wikipedia.org/wiki/Collision\_detection
5. Страница Википедии о Системе частиц [Электронный ресурс] URL: https://en.wikipedia.org/wiki/Particle\_system
6. Официальный сайт CryEngine [Электронный ресурс] URL: https://www.cryengine.com/
7. Официальный сайт Corona [Электронный ресурс] URL: https://ru.coronalabs.com/
8. Официальный сайт Godot Engine [Электронный ресурс] URL: https://godot-engine.ru/
9. Официальный сайт UnrealEngine [Электронный ресурс] URL: https://www.unrealengine.com/
10. Официальный сайт Юнити [Электронный ресурс] URL: https://unity.com/
11. Официальный сайт Cocos [Электронный ресурс] URL: https://www.cocos.com/en/
12. Хокинг Дж. Unity в действии. Мультиплатформенная разработка на C# - 2016 – 336 c.
13. Фленов М.E. Библия C# - 3-e изд., перераб. и доп. – 544 с.:ил.