Федеральное государственное образовательное бюджетное учреждение высшего образования

**«Финансовый университет при Правительстве Российской Федерации»**

**(Финансовый университет)**

Колледж информатики и программирования

Выпускная квалификационная работа

на тему: «Разработка 3D игры «Secret things» в жанре adventure с элементами survival horror»

Студент группы 4ПКС-219 Смирнов Егор Альбертович \_\_\_\_\_\_\_\_

*(№ учебной группы) (фамилия имя отчество полностью (подпись)*

Образовательная программа

09.02.03 Программирование в компьютерных системах

*(индекс и наименование специальности)*

Форма обучения очная

Руководитель

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И.В.Сибирев

*(подпись) (И.О. Фамилия)*

Консультант \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(при наличии) (подпись) (И.О. Фамилия)*

Председатель ПЦК \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т.Г.Аксёнова *(подпись) (И.О. Фамилия)*

Москва – 2023

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc132298473)

[ГЛАВА 1 ПРЕДПРОЕКТНОЕ ИССЛЕДОВАНИЕ 5](#_Toc132298474)

[1.1 Описание предметной области 5](#_Toc132298475)

[1.2 Сравнительный анализ программ-аналогов 5](#_Toc132298476)

[1.3 Постановка задачи 6](#_Toc132298477)

[1.4 Характеристика инструментальных средств разработки 7](#_Toc132298478)

[ГЛАВА 2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ПРОГРАММЫ 9](#_Toc132298479)

[2.1 Анализ требований и разработка спецификаций 9](#_Toc132298480)

[2.2 Проектирование программного обеспечения 10](#_Toc132298481)

[2.3 Разработка программного обеспечения 10](#_Toc132298482)

[2.4 Отладка и тестирование программного обеспечения 16](#_Toc132298483)

[2.5 Руководство по использованию программы 16](#_Toc132298484)

[ЗАКЛЮЧЕНИЕ 17](#_Toc132298485)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 18](#_Toc132298486)

# ВВЕДЕНИЕ

Тема научного исследования: «Разработка 3D игры «Secret things» в жанре adventure с элементами survival horror».

Объект исследования: 3D игра.

Предмет исследования: технологии разработки приложений на Unity.

Главной целью проекта является создание игры, которая позволит человеку научиться быстро действовать и принимать решения в разных ситуациях.

В наше время индустрия видеоигр твёрдо удерживает позицию в списке популярных видов развлечений. Это и не удивительно, ведь с помощью игр можно не только лишь интересно провести своё свободное время, но и также получать новые знания. Так, например, игры позволяют человеку развить его реакцию, мышление, а также научить действовать в различных непредвиденных ситуациях.

Жанр action представляет собой один из популярных жанров среди всех видеоигр. Такую популярность он обрёл из-за своих отличительных особенностей, которые вместе создают захватывающий и динамичный геймплей. Главной чертой такого жанра, как раз является динамичность – у игрока просто не будет времени заскучать, так как почти постоянно на экране будут происходить разные события, начиная от неожиданных препятствий и заканчивая различными противниками, стремящимися помешать пользователю продвинуться дальше.

Чаще всего разработчики выпускают продукт, который содержит в себе несколько разных жанров, такое решение помогает разнообразить игровой процесс и не дать человеку быстро заскучать. Одним из таких жанров является survival horror, характерными чертами которого является акцент на выживании персонажа в суровом игровом мире, а также постоянно нагнетающая атмосфера.

Для предпроектного исследования будут применены следующие методы: анализ предметной области, сравнение существующих решений и моделирование нового решения.

В качестве источниковой базы исследования используем полное руководство по языку программирования С# и учебное пособие по методам и алгоритмам обработки данных.

Для разработки будут использоваться язык программирования C# следующие инструментальные средства: интегрированная среда разработки программного обеспечения Visual Studio, кроссплатформенный движок для создания игр и приложений Unity, а также программное обеспечение для 3D моделирования MagicaVoxel и Blender 3D.

# ГЛАВА 1 ПРЕДПРОЕКТНОЕ ИССЛЕДОВАНИЕ

1.1 Описание предметной области

Темой выпускной квалификационной работы является разработка 3D игры «Secret things» в жанре action с элементами survival horror.

В основе сюжета разрабатываемой игры лежит популярная в наше время история про «Закулисье» (ориг. «Backrooms»). Эта история рассказывает о различных случаях попадания людей в неизвестные пространства, напоминающие собой бесконечную сеть лабиринтов из офисных помещений. Чаще всего в этих помещения очень пусто и единственное, что можно встретить это некие существа, стремящиеся догнать человека.

Существует множество разных уровней этого самого «закулисья», у всех них есть одна общая черта – они похожи на лабиринты и относятся к лиминальным пространствам. Лиминальное пространство (англ. Liminal space), — интернет-эстетика, запечатлевающая места, которые кажутся переходными, жуткими и зачастую сюрреалистичными.

Главной целью игрока будет собрать доказательства, что всё это нереально и выбраться из этого пространства. Также нужно будет прятаться и избегать обитающих там существ.

1.2 Сравнительный анализ программ-аналогов

Проведя сравнительный анализ, были найдены следующие игры, сравнение которых показано в таблице 1.

Таблица 1. Сравнительный анализ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Тип графики | Наличие разных механик | Наличие различных NPC | Сюжет |
| The Complex: Found Footage | 3D | нет | нет | Сюжет отсутствует |
| Inside Backrooms | 3D | Присутствуют небольшое количество загадок, игрок может взаимодействовать с предметами | На каждой из локаций имеются свои враги | Сюжет рассказывается через записки и другие элементы повествования. Сам сюжет очень простой |

Проведя сравнительный анализ, можно сказать, что имеющиеся аналоги не обладают интересным и продуманным геймплеем, а также в них отсутствует какое-либо повествование за исключением некоторых моментов.

1.3 Постановка задачи

Для создания самого лабиринта необходимо разработать специальный программный модуль, который помог бы в создании игрового окружения и генерировал бы нужный лабиринт автоматически, не используя ручное моделирование.

Также важно создать интересные механики для большего взаимодействия с окружающим игровым миром.

Стоит отметить и о необходимости в создании умного искусственного интеллекта, которым можно было бы заполнить игровое пространство и сделать прохождение игры ещё интереснее.

Минимальные системные требования для запуска игры:

- ОС: Windows® 7/Vista/XP и выше;

- Процессор: Intel® Core™ 2 Duo E6600 или AMD Phenom™ X3 8750;

- Оперативная память: 2 GB ОЗУ;

- Видеокарта: Видеокарта с 256 МБ памяти или больше, совместимая с DirectX 9 с поддержкой Pixel Shader 3.0;

- DirectX: Версии 9.0c;

- Место на диске: 4 GB.

1.4 Характеристика инструментальных средств разработки

Для разработки проекта был выбран язык программирования C# (произносится как "си шарп") - современный объектно-ориентированный и типобезопасный язык программирования. C# позволяет разработчикам создавать разные типы безопасных и надежных приложений, выполняющихся в .NET. C# относится к широко известному семейству языков C, и покажется хорошо знакомым любому, кто работал с C, C++, Java или JavaScript. Здесь представлен обзор основных компонентов языка C# 8 и более ранних версий. C# предоставляет языковые конструкции для непосредственной поддержки такой концепции работы. Благодаря этому C# подходит для создания и применения программных компонентов. С момента создания язык C# обогатился функциями для поддержки новых рабочих нагрузок и современными рекомендациями по разработке ПО. В основном C# - объектно-ориентированный язык.

Для написания программного кода будет использоваться интегрированная среда разработки Visual Studio — это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Помимо стандартного редактора и отладчика, которые есть в большинстве сред IDE, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для улучшения процесса разработки.

В создании трёхмерного пространства поможет межплатформенная среда разработки компьютерных игр Unity. Unity позволяет создавать приложения, работающие на более чем 25 различных платформах, включающих персональные компьютеры, игровые консоли, мобильные устройства, интернет-приложения и другие. Основными преимуществами Unity являются наличие визуальной среды разработки, межплатформенной поддержки и модульной системы компонентов. К недостаткам относят появление сложностей при работе с многокомпонентными схемами и затруднения при подключении внешних библиотек.

С помощью программы MagicaVoxel сделаем воксельные модели и элементы игрового окружения для всей игры.

Программное обеспечение Blender 3d даст возможность создавать анимации для моделей.

Для создания и оформления пояснительной записки используем Microsoft Word (часто - MS Word, WinWord или просто Word) - текстовый процессор, предназначенный для создания, просмотра и редактирования текстовых документов, с локальным применением простейших формтаблично-матричных алгоритмов.

Для описания инструментов для построения диаграмм и схем задействуем бесплатное кроссплатформенное программное обеспечение для рисования графиков Diagrams.net (ранее draw.io), а также онлайн сервис для построения диаграммы Ганта ganttpro.

# ГЛАВА 2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ПРОГРАММЫ

## 2.1 Анализ требований и разработка спецификаций

Далее будут предоставлены наработки по игре в виде диаграмм и части программного кода.

Изображение выглядит как текст, внутренний, снимок экрана

Автоматически созданное описаниеНа рисунке 1 изображена диаграмма переходов и состояний.

Рисунок 1. Диаграмма переходов и состояний

Данная диаграмма наглядно показывает, как программа переходит из одного состояния в другое. Сначала игроку нужно искать разные предметы для выживания. После происходит встреча и последующий бой с вражеским искусственным интеллектом. Далее пользователю необходимо найти ключевые предметы, которые нужны для открытия финальной сцены.

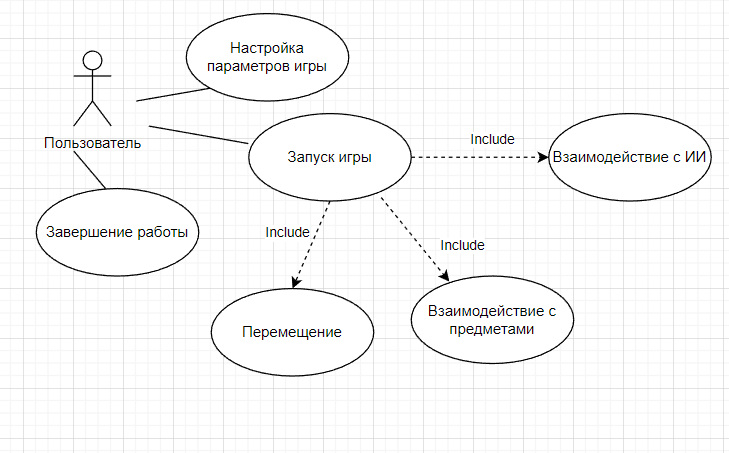
На рисунке 2 показана диаграмма вариантов использования.

Рисунок 2. Диаграмма вариантов использования

На этой диаграмме отображены все возможные действия игрока. Он может: настраивать параметры игры, запускать игру и выходить из неё, взаимодействовать с ИИ, свободно перемещаться по игровому миру, взаимодействовать с различными предметами.

## 2.2 Проектирование программного обеспечения

## 2.3 Разработка программного обеспечения

Разработка игры началась с создания алгоритма по генерации алгоритма.

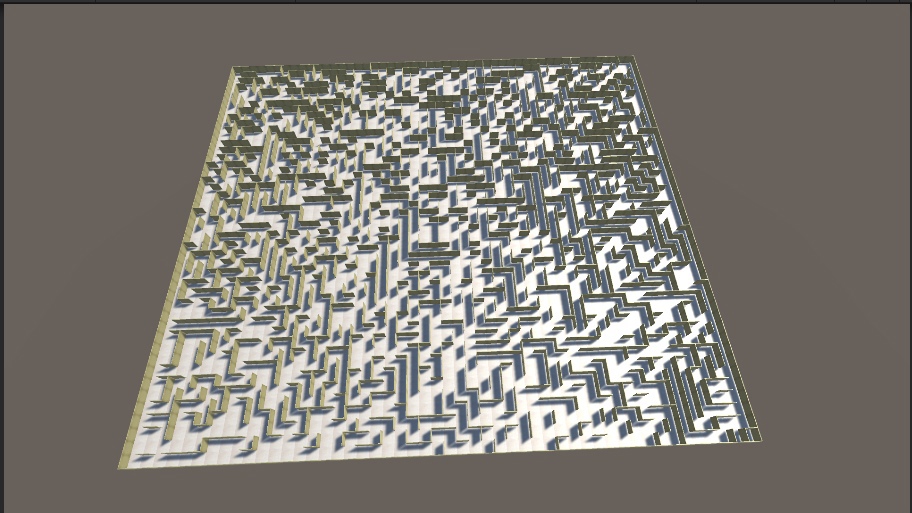
На рисунке 3 можно увидеть результат работы этого алгоритма. В дальнейшем базовая модель стен будет заменена на другую.

Рисунок 3. Результат работы алгоритма.

Этот модуль представляет собой основную часть алгоритма по генерации игрового окружения.

using System.Collections;

using System.Collections.Generic;

using System.Runtime.InteropServices.WindowsRuntime;

using UnityEngine;

using UnityEngine.SceneManagement;

public class BackroomGeneration : MonoBehaviour

{

[Range(5, 500)]

public static int BackroomsWidth =5, BackroomsHeight =5;

public int startX, startY;

BackroomsCell[,] backrooms;

Vector2Int CurrentCell;

public BackroomsCell[,] GetBackrooms()

{

backrooms = new BackroomsCell[BackroomsWidth, BackroomsHeight];

for (int x = 0; x < BackroomsWidth; x++)

{

for (int y = 0; y < BackroomsHeight; y++)

{

backrooms[x, y] = new BackroomsCell(x, y);

}

}

CarvePath (startX, startY);

return backrooms;

}

// Создания листа для хранения направлений

List<Direction> directions = new List<Direction>

{ Direction.Up,Direction.Down, Direction.Left, Direction.Right};

List<Direction> GetRandomDirections()

{

List<Direction> dir = new List<Direction>(directions);

List<Direction> rndDir = new List<Direction>();

// Исключаем случаи использования одних и тех же направлений создания стен

while (dir.Count > 0)

{

int rnd = Random.Range(0, dir.Count);

rndDir.Add(dir[rnd]);

dir.RemoveAt(rnd);

}

return rndDir;

}

// Не используем ячейки в которых уже были

bool IsCellValid (int x, int y)

{

if (x < 0 || y < 0 || x > BackroomsWidth - 1 || y > BackroomsHeight - 1 || backrooms[x, y].visited) return false;

else return true;

}

Vector2Int CheckNeighbours ()

{

List<Direction> rndDir = GetRandomDirections();

for (int i = 0; i < rndDir.Count; i++)

{

Vector2Int neighbour = CurrentCell;

switch (rndDir[i])

{

case Direction.Up:

neighbour.y++;

break;

case Direction.Down:

neighbour.y--;

break;

case Direction.Right:

neighbour.x++;

break;

case Direction.Left:

neighbour.x--;

break;

}

if (IsCellValid(neighbour.x, neighbour.y)) return neighbour;

}

return CurrentCell;

}

void BreakWalls (Vector2Int PrimaryCell, Vector2Int SecondaryCell)

{

if (PrimaryCell.x > SecondaryCell.x)

{

backrooms[PrimaryCell.x, PrimaryCell.y].LeftWall = false;

}

else if (PrimaryCell.x < SecondaryCell.x)

{

backrooms[SecondaryCell.x, SecondaryCell.y].LeftWall = false;

}

else if (PrimaryCell.y < SecondaryCell.y)

{

backrooms[PrimaryCell.x, PrimaryCell.y].TopWall = false;

}

else if (PrimaryCell.y > SecondaryCell.y)

{

backrooms[SecondaryCell.x, SecondaryCell.y].TopWall = false;

}

}

void CarvePath (int x, int y)

{

if (x < 0 || y < 0 || x > BackroomsWidth - 1 || y > BackroomsHeight - 1)

{

x = y = 0;

Debug.LogWarning("Координаты начальной позиции вышли за пределы");

}

CurrentCell = new Vector2Int(x, y);

List<Vector2Int> path = new List<Vector2Int>();

bool DeadEnd = false;

while (!DeadEnd)

{

Vector2Int NextCell = CheckNeighbours ();

if (NextCell == CurrentCell)

{

for (int i = path.Count - 1; i >= 0; i--)

{

CurrentCell = path[i];

path.RemoveAt(i);

NextCell = CheckNeighbours();

if (NextCell != CurrentCell) break;

}

if (NextCell == CurrentCell)

DeadEnd = true;

}

else

{

BreakWalls(CurrentCell, NextCell);

backrooms[CurrentCell.x, CurrentCell.y].visited = true;

CurrentCell = NextCell;

path.Add(CurrentCell);

}

}

}

}

public enum Direction

{

Up,

Down,

Left,

Right

}

public class BackroomsCell

{

public bool visited;

public int x, y;

public bool TopWall;

public bool LeftWall;

public Vector2Int position

{

get

{

return new Vector2Int(x, y);

}

}

public BackroomsCell(int x, int y)

{

this.x = x;

this.y = y;

visited = false;

TopWall = LeftWall = true;

}

}

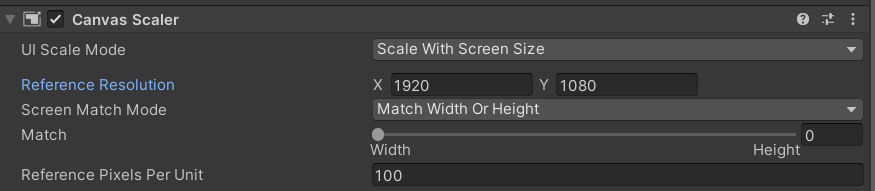


Рисунок 4. Настройки при создании меню

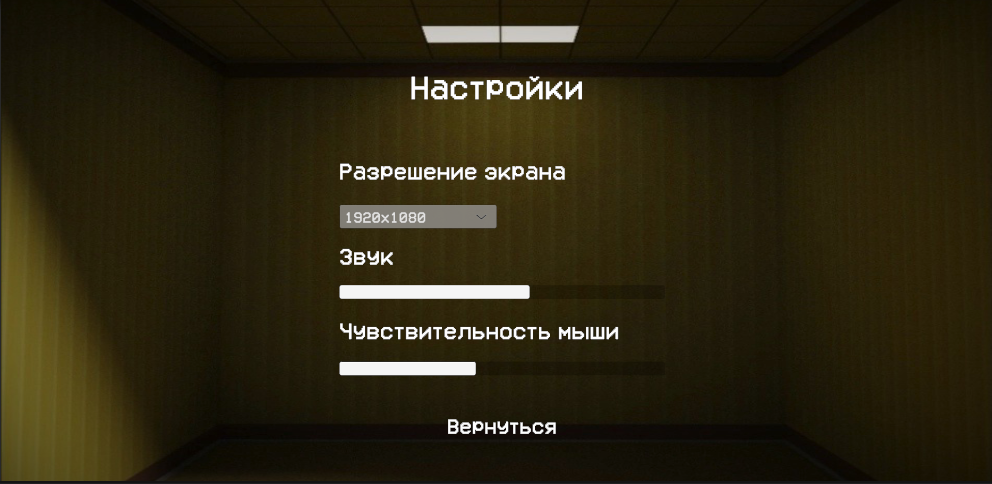
Чтобы меню и его объекты были адаптивными и подстраивались под различные экраны, в параметре UI Scale Mode ставим значение Scale With Screen Size. А в параметре Reference Resolution записываем значения для наиболее популярного разрешения Full HD (1920×1080).

Изображение выглядит как текст, стена, в помещении, потолок

Автоматически созданное описание

Рисунок 5. Главное меню

В качестве основного шрифта был выбран VCR OSD Mono. Данный шрифт имеет хорошую читабельность, а также за счёт своего необычного внешнего вида позволит игроку уже из меню начать погружаться в атмосферу игры (так как действия игры происходят в 1996 году, а главный герой снимает всё происходящую на камеру, интерфейс которой имел подобный шрифт). На фоне представлена локация из игры.

Рисунок 6. Меню с настройками

В данном меню с помощью ползунков можно настроить чувствительность мыши и громкость звука в игре. Также есть возможность изменить разрешение экрана, учтены три варианта: 1920х1080, 1600х900, 1280х1024. Снизу расположена кнопка для возвращения обратно в главное меню.

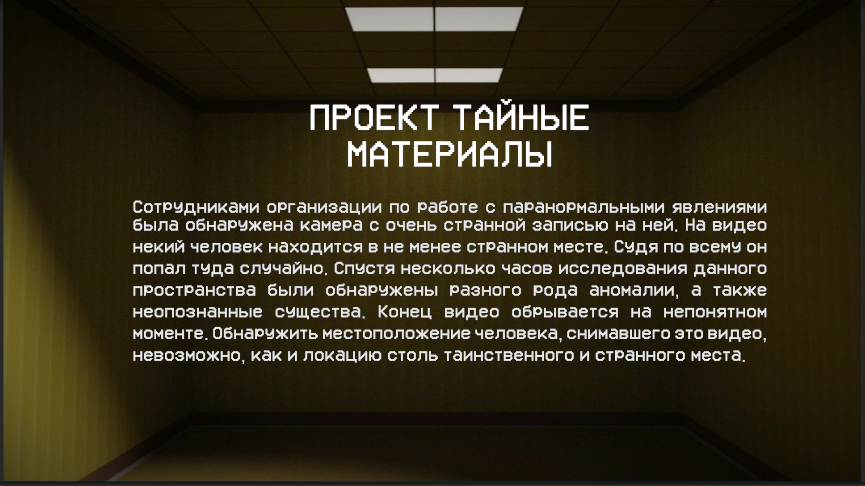


Рисунок 7. Раздел с повествованием

Содержит в себе краткое описание сюжета игры, чтобы игрок мог понять, что примерно его ожидает и, что от него может требоваться. При нажатии на кнопку Escape, игрок вернётся в главное меню.

## 2.4 Отладка и тестирование программного обеспечения

## 2.5 Руководство по использованию программы

2.5.1 Руководство системного администратора

2.5.2 Руководство программиста

2.5.3 Руководство оператора

# ЗАКЛЮЧЕНИЕ

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

*Законодательные и нормативные акты:*

1. ГОСТ Р 7.0.12-2011 Библиографическая запись. Сокращение слов и словосочетаний на русском языке. Общие требования и правила. – М.: Стандартинформ, 2012. – 61 с.
2. ГОСТ 7.1-2003 Библиографическая запись. Библиографическое описание. Общие требования и правила составления. – М.: Стандартинформ, 2010. – 92 с.
3. ГОСТ 7.32-2017 Отчет о научно-исследовательской работе. Структура и правила оформления. – М.: Стандартинформ, 2017. – 47 с.
4. ГОСТ 7.82-2001 Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления. – М.: ИПК Издательство стандартов, 2001. – 39 с.
5. ГОСТ Р 7.0.100-2018 Библиографическая запись. Библиографическое описание. Общие требования и правила составления. – М.: Стандартинформ, 2018. – 122 с.
6. ГОСТ Р 7.0.5-2008 Библиографическая ссылка. Общие требования и правила составления. – М.: Стандартинформ, 2008. – 32 с.
7. Единая система программной документации. – М.: Стандартинформ, 2005. – 128 с.

*Учебная и научная литература:*

1. Павловская, Т.А. C#. Программирование на языке высокого уровня: учебник для студентов вузов. – СПб: Питер, 2020. – 432 с.

*Интернет-документы*

1. Алгоритмы генерации игр. – [Электронный ресурс]. – URL: <https://ru.wikipedia.org/wiki/Алгоритм_Эллера#:~:text=Алгоритм%20Эллера%20—%20математический%20генератор,лабиринты%20с%20неограниченным%20количеством%20рядов> (дата обращения: 25.09.2022)
2. Алгоритмы генерации игр. – [Электронный ресурс]. – URL: <https://tproger.ru/articles/maze-generators/> (дата обращения: 25.09.2022)
3. Добро пожаловать в интегрированную среду разработки Visual Studio. – [Электронный ресурс]. – URL: <https://learn.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2019> (дата обращения: 25.09.2022)
4. Как создавать в играх бесконечные миры при помощи процедурной генерации. – [Электронный ресурс]. – URL: <https://habr.com/ru/company/piter/blog/593237/> (дата обращения: 25.09.2022)
5. Краткий обзор языка C#. – [Электронный ресурс]. – URL: <https://learn.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/> (дата обращения: 25.09.2022)
6. Обзор систем процедурной генерации игр. – [Электронный ресурс]. – URL: <https://cyberleninka.ru/article/n/obzor-sistem-protsedurnoy-generatsii-igr/viewer> (дата обращения: 25.09.2022)
7. Microsoft Word. – [Электронный ресурс]. – URL: <https://microsoft.fandom.com/ru/wiki/Microsoft_Word> (дата обращения: 25.09.2022)
8. Unity (игровой движок). – [Электронный ресурс]. – URL: <https://ru.wikipedia.org/wiki/Unity_(игровой_движок)> (дата обращения: 25.09.2022)