МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

РАБОТА ПРОВЕРЕНА

Рецензент
Доцент кафедры ПРИМА ЮУрГУ,
к.т.н.
_____Е.А. Геренштейн
“___”_____ 2020 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой,
д.ф.-м.н., профессор
_____ Л.Б. Соколинский
“___”_____ 2020 г.

Разработка компьютерной игры в жанре «Квест»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2020.308-429.ВКР

Научный руководитель,
к.ф.-м.н., доцент кафедры СП
_____ А.В. Геренштейн

Автор работы,
студент группы КЭ-402
_____ А.А. Непеин

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
“___”_____ 2020 г.

Челябинск-2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____Л.Б. Соколинский

09.02.2020

ЗАДАНИЕ
на выполнение выпускной квалификационной работы
студенту группы КЭ-402
Непеину Александру Асхатовичу,
обучающемуся по направлению
02.03.02 «Фундаментальная информатика и информационные технологии»

1. Тема работы (утверждена приказом ректора от 24.04.2020 № 627)
Разработка компьютерной игры в жанре «Квест».
2. Срок сдачи студентом законченной работы:07.06.2020 г.
3. Исходные данные к работе
3.1. Документация Unity. [Электронный ресурс] URL: https://docs.unity3d.com/Manual/index.html.
3.2. Adams, Ernest. Fundamentals of Game Design. — 2nd edition. — Berkeley CA: New Riders, 2010. — С. 547. — 700 p.
3.3. Бонд Джереми Гибсон. Unity и C#. Геймдев от идеи до реализации. 2-е изд. — СПб.: Питер, 2019. — 928 с.: ил. — (Серия «Для профессионалов»).
4. Перечень подлежащих разработке вопросов
4.1 . Выполнить обзор аналогов игрового приложения.
4.2.Выполнить обзор средств для реализации игрового приложения.
4.3.Спроектировать игровое приложение.
4.4.Реализовать игровое приложение.
4.5.Провести тестирование игрового приложения.
5. Дата выдачи задания: 09.02.2020 г.

Научный руководитель
к.ф.-м.н., доцент кафедры СП А.В. Геренштейн

Задание принял к исполнению А.А. Непеин

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 4
1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 6
1.1. Обзор аналогичных проектов 6
1.2. Обзор программных средств разработки 9
2. ПРОЕКТИРОВАНИЕ ИГРОВОГО ПРИЛОЖЕНИЯ 12
2.1. Техническое предложение 12
2.2. Функциональные требования 15
2.3. Нефункциональные требования 16
2.4. Диаграмма прецедентов 16
3. РЕАЛИЗАЦИЯ ИГРОВОГО ПРИЛОЖЕНИЯ 19
3.1. Визуальное и звуковое оформление приложения 19
3.2. Файловая структура приложения 19
3.3. Реализация сохранения игры 20
3.4. Интерфейс 21
3.5. Реализация персонажа 23
3.6. Реализация диалогов 26
3.7. Реализация уровней 28
4. ТЕСТИРОВАНИЕ 32
ЗАКЛЮЧЕНИЕ 34
ЛИТЕРАТУРА 35

ВВЕДЕНИЕ

Актуальность

На сегодняшний день разработка компьютерных игр является одной из самых прибыльных отраслей в сфере информационных технологий. Тяжело представить компьютер современного человека, на котором не было установлено хотя бы одной игры. В настоящие время компьютерные игры являются не просто развлечением, а чем-то более значимым. В играх человек может подчеркнуть познавательные и значимые для развития личности вещи, проходя испытания, которые были даны главному герою, и испытывая на себе груз переживаний, персонажа, с которым себя может ассоциировать игрок.

Сейчас рынок компьютерных игр может предоставить потребителю необъятный простор для выбора разных жанров, таким образом, любой человек способен найти себе игру, которая могла бы ему понравиться и оставить приятный след после прохождения.

Среди самых популярных жанров можно упомянуть *Квест* [2]. Ведь данный жанр предоставляет интересный сюжет, простое управление и интересные загадки, что позволяет даже самому далекому от компьютерных игр человеку погрузиться с головой в увлекательное развлечение и заинтересовать его в дальнейшем изучении сферы компьютерных игр.

Цель и задачи

Главной целью моей выпускной квалификационной работы является разработка компьютерной игры в жанре *Квест.*

Для достижения поставленной цели необходимо решить следующие задачи.

1. Выполнить обзор существующих аналогов игрового приложения.
2. Выполнить обзор программных средств для разработки игрового приложения.
3. Спроектировать игровое приложение.
4. Реализовать игровое приложение.

5. Провести тестирование игрового приложения.

Структура и объем работы

Работа состоит из введения, 4 разделов, заключения, списка использованной литературы.

Работа составляет 36 страниц, в списке литературы указано 23 источников.

В первой главе производится анализ предметной области, приведен обзор существующих аналогов и программных средств разработки.

Во второй главе раскрывается проектирование приложения, его компонентов, приведены поясняющие диаграммы.

Третья глава описывает разработку игрового приложения.

В четвертой главе приведены таблицы с результатами функционального тестирования.

В заключении выведены основные результаты проделанной работы.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

В данной главе рассматриваются игровые аналоги приложения в жанре *Квест,* обнаруживаются их недостатки и достоинства. Также производится обзор программных средств для разработки игрового приложения.

1.1. Обзор аналогичных проектов

Квест – жанр компьютерных игр, где в основе лежит исследование игрового мира и повествование сюжета, также главную роль в процессе прохождения занимают загадки, которые требуют от игрока внимания и умственных усилий. Для обзора аналогичных проектов было выделено несколько популярных *Квест* игр.

Grim Fandango

Grim Fandango – одна из самых популярных игр в жанре квест. Была разработана и издана компанией LucasArts Entertainment. В основе игры заложен увлекательнейший сюжет, трудные и интересные головоломки. Сюжет представляет из себя историю туристического агента в Департаменте мертвых, который продает душам умерших билеты на путешествие к вечному покою, однако на небесах происходит неразбериха и игроку нужно раскрыть заговор, который угрожает жизни главного героя. К достоинствам игры можно отнести интересный игровой мир, захватывающее дух повествование и огромное количество головоломок. К недостаткам можно отнести трудность головоломок, из-за чего можно часами искать ответ на то, чего же хочет игра для дальнейшего продвижения по сюжету. Снимок экрана представлен на рисунке 1.

Рис. 1. Снимок экрана Grim Fandango

Machinarium

Machinarium – компьютерная игра в жанре квест, разработанная независимой студией-разработчиком компьютерных игр Amanita Design [14]. Сюжет игры повествует о роботе, которого по вине бандитов выбросили на помойку, но он не собирается мириться с произошедшим и возвращается в город, чтобы спасти свою подругу от злоумышленников, по вине которых и оказался на свалке. К достоинствам можно отнести интересный сюжет, потрясающую атмосферу и множество интересных головоломок. К недостаткам следует отнести то, что решение головоломок не всегда может лежать на поверхности, из-за чего можно потратить много времени и нервов. Снимок экрана представлен на рисунке 2.

Рис. 2. Снимок экрана Machinarium

Fahrenheit

Fahrenheit – это компьютерная игра, которая была разработана компанией Quantic Dream [16] и издана Atari [15]. Главным дизайнером игры и сценаристом является Дэвид Кейдж. В центре сюжета обычный банковский менеджер, который неожиданно лишившись собственного контроля убивает ни в чем неповинного человека. Целью всей игры будет разгадать причины преступления, которое сделал главный герой, погрузившись в расследование вместе с полицейским, расследующим убийство, совершенное главным героем. К достоинствам игры следует отнести наличие интерактивных роликов [4], что позволяет игроку сильнее влиться в сюжет, свободу выбора, которая сильно увеличивает реиграбельность [5]. К недостаткам следует отнести незавершенный сюжет. Снимок экрана представлен на рисунке 3.

Рис. 3. Снимок экрана Fahrenheit

1.2. Обзор программных средств разработки

В данный момент для разработки игрового приложения существует множество решений со своими положительными и отрицательными свойствами. Для реализации игры необходима платформа, которая будет удобна в использовании, иметь подробную документацию, возможность для разработки 3D игр, а также свободный доступ.

Для этого проведем анализ следующих популярных движков для разработки игр:

Unity [17] – межплатформенная среда разработки видеоигр. Unity позволяет создавать приложения, работающие на более чем 25 различных платформах, таких как персональные компьютеры, веб-приложения и другие. Любое приложение на Unity состоит из сцен, которые в свою очередь содержат игровые объекты, обладающие набором поведенческих скриптов и других компонентов [22].

Достоинства:

1) большой объем качественной документации;
2) кроссплатформенность;
3) лёгкость освоения;

4) простота в использовании;

5) большой выбор инструментов разработки;

6) большое сообщество пользователей.

Недостатки:

1) условно-бесплатная;

Unreal Engine [16] – кроссплатформенный игровой движок для разработки видеоигр, который позволяет создавать игры для всех популярных платформ, таких как Windows, macOS, Andriod, iOS и многие другие.

Достоинства:

1) большой выбор инструментов разработки;

2) хорошая документация;

3) кроссплатформенность.

Недостатки:

1) трудность освоения.

Pygame [1] – библиотека модулей языка Python, созданная для разработки 2D игр. Данная библиотека является кроссплатформенной, поддерживается многими популярными операционными системами, такими как Windows, macOS, Linux, Android и многими другими.

Достоинства:

1) бесплатная лицензия;

2) кроссплатформенность;

3) огромный объем документации;

4) большое сообщество разработчиков.

Недостатки:

1) поддерживает только один язык программирования;

2) узкий выбор инструментов разработки.

Вывод

В процессе анализа предметной области были рассмотрены аналоги проекта, которые позволили сформировать основные принципы игрового процесса и жанра в целом.

В результате обзора программных средств разработки, для создания игрового приложения была выбрана платформа Unity из-за своих достоинств.

2. ПРОЕКТИРОВАНИЕ ИГРОВОГО ПРИЛОЖЕНИЯ

2.1. Техническое предложение

Общая информация

Police Officer's Day – игра в жанре квест, рассказывающая историю о том, как офицеру предстоит расследовать загадочную пропажу мужа девушки, которая обратилась в полицейский участок. Для разгадки пропажи игроку потребуется искать улики на уровнях и взаимодействовать с жителями города.

Основная концепция

Игра является игровым приложением для операционной системы Windows. Основной задачей игрока является прохождение сюжетной кампании, после выполнения которой игра завершается.

Сюжетная кампания состоит из нескольких уровней, на которых спрятаны улики, при нахождении которых игрок может использовать их для продвижения по сюжету, переходя на новые уровни.

Игрок может взаимодействовать с игровым миром, а именно:

1) общаться с неигровыми персонажами;
2) получать предметы (например, найти улику);
3) использовать предметы (например, предоставить улику).

Интерфейс

Игровое приложение содержит четыре разных интерфейса: главное меню, меню инвентаря, интерфейс диалога и игровая сцена. В понятие интерфейса входит не только изображение на экране, но и взаимодействие пользователя с приложением [20].

Основа интерфейса главного меню, с которого начинается игра, представлена на рисунке 4.

Рис. 4. Макет главного меню

При нажатии кнопки «Новая игра» начинается игровой процесс. При нажатии кнопки «Загрузить игру» происходит загрузка игрового процесса с точки сохранения, которую создал игрок. При нажатии на «X» происходит выход из игры.

Меню инвентаря может быть вызвано во время игрового процесса нажатием на кнопку «Инвентарь». При нажатии на игровое поле или на кнопку «Инвентарь», меню закрывается, и игрок может продолжить игру. Макет меню инвентаря представлен на рисунке 5.

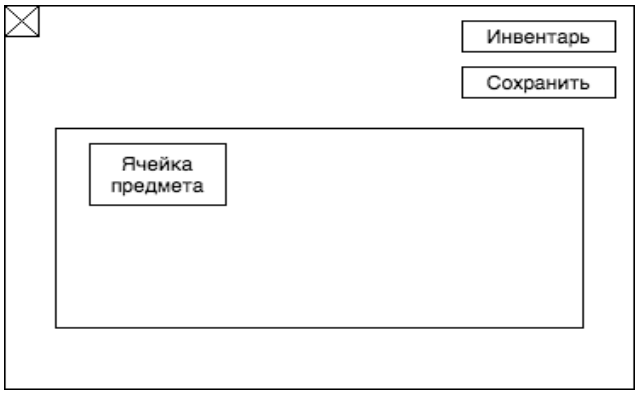

Рис. 5. Макет меню инвентаря

При проектировании интерфейса игрового поля, его элементы были расположены таким образом, чтобы не мешать пользователю во время игры и в то же время быть всегда под рукой. Макет игрового поля представлен на рисунке 6.

В правой верхней части интерфейса находятся кнопки «Инвентарь» и «Сохранить».

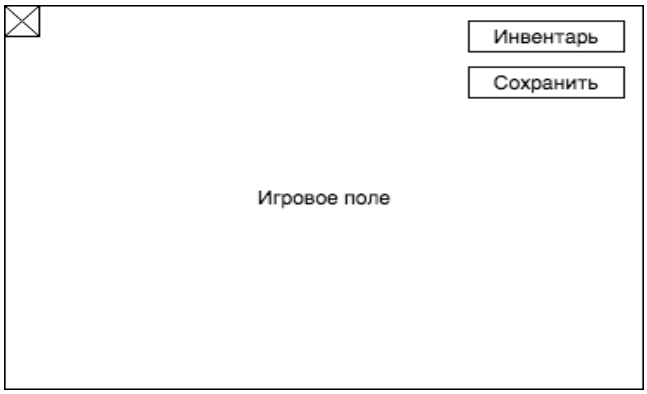

Рис. 6. Макет игрового поля

При взаимодействии с некоторыми неигровыми персонажами у игрока есть возможность провести диалог для того, чтобы узнать сюжетные детали или продвинутся дальше по сюжету. Чтобы выйти из диалогового окна можно нажать на игровое поле или окно закроется автоматически после завершения диалога. Макет диалогового окна показан на рисунке 7.

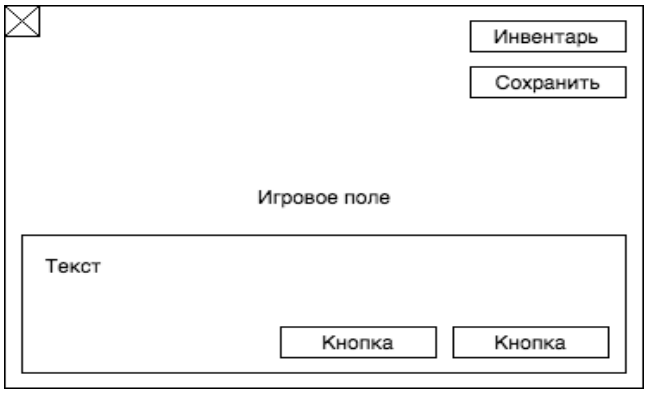

Рис. 7. Макет диалогового окна

Игровой процесс

Пользователь начинает игровой процесс на начальном уровне после сцены пролога, где ему повествуется завязка игры, в виде сцена пролога.

У пользователя в начале игры не имеется никаких предметов.

Игрок может передвигаться при помощи наведения курсора и нажатия кнопки мыши на экран, иными словами реализован метод управления Point and Click.

При взаимодействии с неигровыми персонажами, игрок может либо получить, либо отдать сюжетный предмет, а также есть возможность узнать подсказку для решения головоломки при помощи диалога.

Сюжет и концепт мира

Действие игры происходит в тихом городке, в наше время. Сюжет повествует об обычном офицере полиции, который получает с виду простое задание – найти пропавшего мужа женщины, но чем дальше он продвигается в расследовании, тем сильнее накаляется обстановка. Во время поисков главный герой сталкивается с интересными загадками, встречает как дружелюбных персонажей, которые стремятся помочь офицеру, так и недоброжелателей, которые будут мешать поискам.

2.2. Функциональные требования

Функциональные требования [6] дает понять, что должно выполнять разрабатываемое программное обеспечение, иными словами определяются задачи, которые должны быть выполнены.

1) Окно главного меню должно отображаться после включения игрового приложения корректно.

2) Выход из игры должен работать корректно.

3) Игровое приложение должно предоставлять пользователю возможность сохранять игровой процесс.

4) Игровое приложение должно предоставлять пользователю возможность загружать сохранённый игровой процесс.

5) Переход из главного меню в игровой процесс должен работать корректно.

6) Переход с одного уровня на другой должен работать корректно.

7) Игровое приложение должно предоставлять игроку возможность управления персонажем.

8) Игровое приложение должно предоставлять пользователю возможность вести диалог с неигровыми персонажами.

9) Игровое приложение должно предоставлять пользователю возможность брать или отдавать предметы неигровым персонажам.

10) Игровое приложение должно предоставлять пользователю возможность просмотра инвентаря во время процесса прохождения игры.

2.3. Нефункциональные требования

Нефункциональные требования описывают свойства и ограничения, накладываемые на систему. Для реализации приложения были выявлены следующие требования.

1. Приложение Police Officer's Day должно быть написано на языке C# [19] под платформу Windows с помощью платформы Unity.

2. Приложение должно сохранять и загружать нужные пользователю данные.

2.4. Диаграмма прецедентов

Исходя из проведенного анализа требований, при помощи языка графического описания объектного ориентирования UML [23], было построено две диаграммы прецедентов игрового приложения.

В реализуемом игровом приложении предусмотрено актер – *пользователь* который взаимодействуют с объектами игрового приложения.

Рассмотрим диаграмму вариантов использования в главном меню пользователем на рисунке 8.

Диаграмма вариантов использования игрового процесса пользователем представлена на рисунке 9.

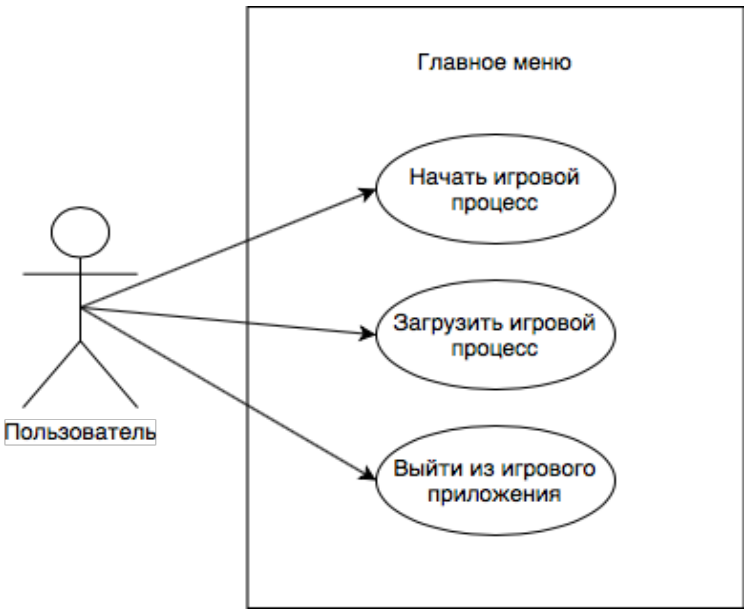

Рис. 8. Диаграмма вариантов использования в главном меню пользователем

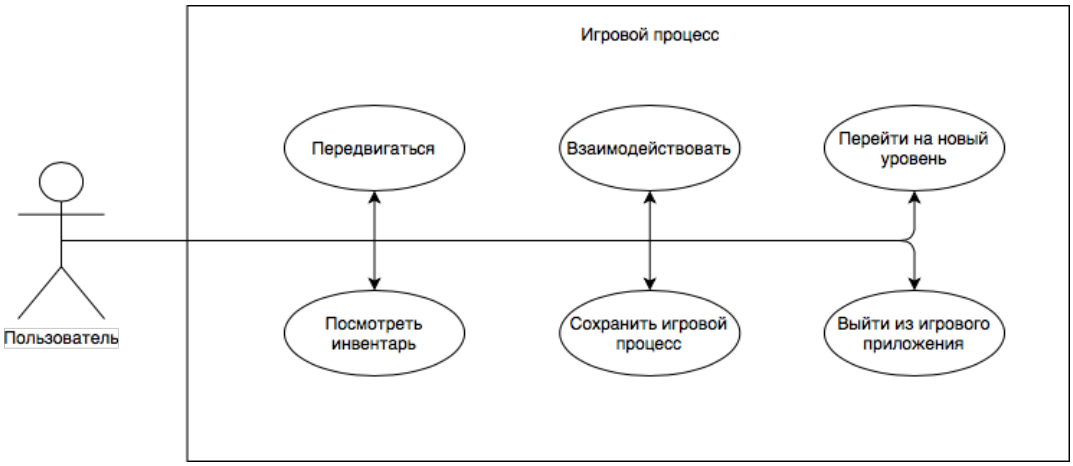

Рис. 9. Диаграмма вариантов использования игрового процесса пользователем

Краткое описание вариантов использования системы *пользователем.*

1) Начать игровой процесс – начинает игровой процесс.

2) Загрузить игровой процесс – загружает сохраненный пользователем игровой процесс.

3) Выйти из игрового приложения – выходит из игры.

4) Передвигаться – контролирует перемещение персонажа.

5) Взаимодействовать – взаимодействует с игровыми объектами.

6) *Перейти на новый уровень* – происходит смена игровой сцены.

7) *Посмотреть инвентарь* – отображает инвентарь, игрока.

8) Сохранить игровой процесс – сохраняет игровой процесс.

3. РЕАЛИЗАЦИЯ ИГРОВОГО ПРИЛОЖЕНИЯ

3.1. Визуальное и звуковое оформление приложения

Для визуального оформления игрового приложения были найдены бесплатные модели персонажей в Uniti Asset Store [18].

Для спрайтов были найдены бесплатные изображения в интернете.

Некоторые бесплатные модели были дополнены деталями при помощи Paint 3D [3].

Для звукового оформления игрового приложения были найдены бесплатные звуки в сети Интернет.

3.2. Файловая структура приложения

Проект разработанного игрового приложения состоит из списка каталогов, содержащих материалы: 3D-модели, их анимации, аудиофайлы, готовые компоненты, которыми являются игровые объекты с заданными свойствами для повторного использования, игровые сцены, игровые скрипты, спрайты. Файловая структура приложения представлена на рисунке 10.

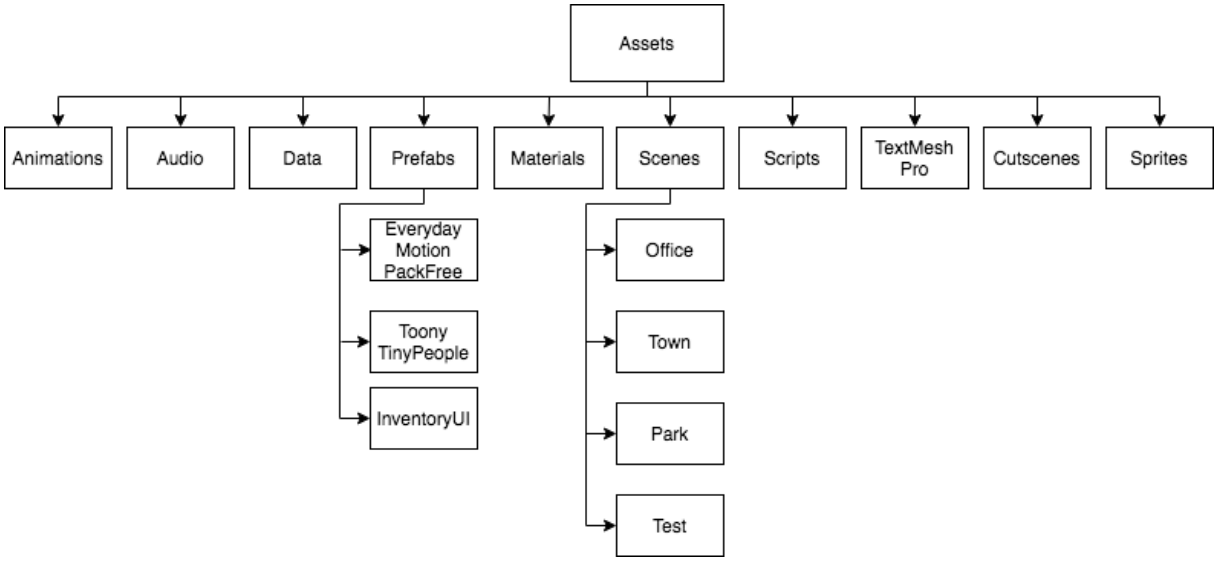

Рис. 10. Файловая структура приложения

3.3. Реализация сохранения игры

Для работы сохранения и загрузки данных в игровом приложении было создан класс EntData, который инициализирует координаты положения игрока на игровой сцене. При помощи методов SetPos и SetRot, которые представлены на рисунке 11.

```
public void SetPos (Vector3 position)
{
    x = position.x;
    y = position.y;
    z = position.z;
}

public void SetRot (Quaternion rotation)
{
    qw = rotation.w;
    qx = rotation.x;
    qy = rotation.y;
    qz = rotation.z;
}
```

Рис. 11. Фрагмент кода класса EntData

Далее был создан скрипт SaveState, изображенный на рисунке 12, и, который инициализирует объект класса EntData и добавляет в него значения координат игрока, после чего при помощи метода SaveEnts, представленный на рисунке 13, мы добавляем ключ, являющийся идентификационным номером объекта, а также значение, являющиеся данными о его местоположении, в словарь, находящийся в скрипте DataManager и хранящий сохраненные данные.

```
public void SaveState()
{
    entData = new EntData();
    entData.SetPos(transform.position);
    entData.SetRot(transform.rotation);
    for (int i = 0; i < transform.childCount; i++)
    {
        entData.childActiveStatus.Add(i,
            transform.GetChild(i).gameObject.activeInHierarchy);
    }
    DataManager.Instance.SaveEnts(instanceID, entData);
}
```

Рис. 12. Реализация метода SaveSate

```
public void SaveEnts(string id, EntData data)
{
    if (saveDatas[saveDataId].entsData.ContainsKey(id))
    {
        saveDatas[saveDataId].entsData[id] = data;
        return;
    }
    else
    {
        saveDatas[saveDataId].entsData.Add(id, data);
    }
}
```

Рис. 13. Реализация метода SaveEnts

3.4. Интерфейс

При запуске игрового приложения, игроку открывается окно главного меню, которое изображено на рисунке 14.

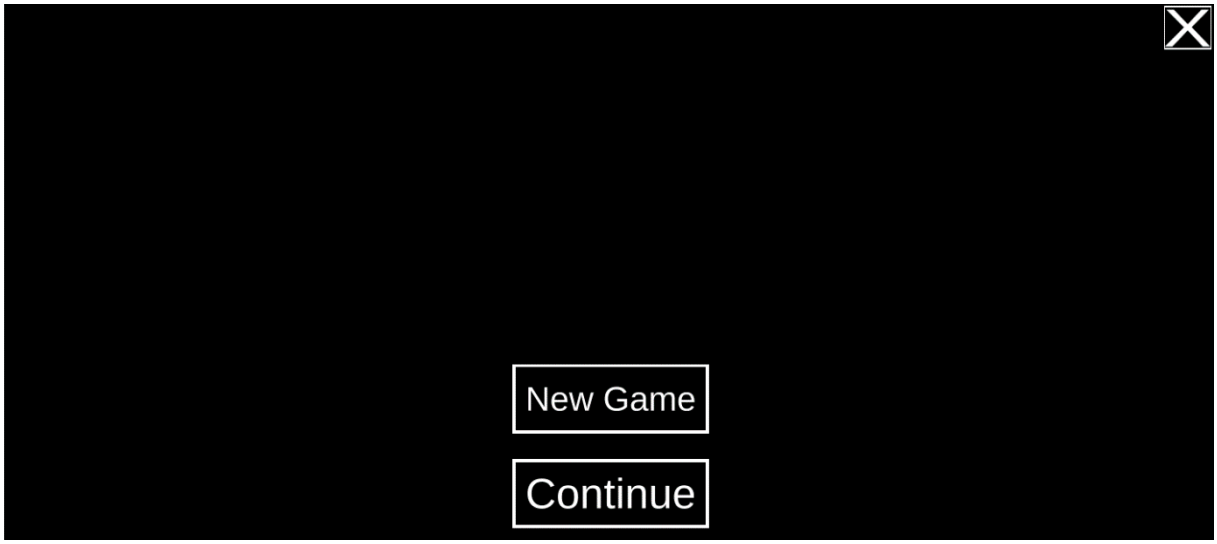

Рис. 14. Интерфейс главного меню

Основные объекты главного меню:

1) кнопка запуска нового игрового процесса;
2) кнопка загрузки игрового процесса;
3) кнопка выхода из игры.

Нажав на крест в верхнем угла экрана, пользователь выйдет из приложения.

Нажав на кнопку «Load Game», пользователь загрузит сохраненный им игровой процесс.

Нажав на кнопку «New Game», пользователь начнет игру сначала.

После запуска игрового процесса, игроку сцена пролога, где сообщается о его главном задании – найти пропавшего мужа потерпевшей женщины. Пролог представлен на рисунке 15. Интерфейс приложения сделан так, чтобы его элементы не отвлекали игрока во время игрового процесса, а также были четко видны на экране и оставались интуитивно понятными.

Рис. 15. Сцена пролога игрового приложения

Основные объекты данного интерфейса:

1) посмотреть в инвентарь;
2) сохранить игровой процесс;
3) кнопка выхода из игрового приложения;
4) диалоговое окно;
5) игровая сцена.

При нажатии на кнопку «Save» игра сохраняет прогресс игрового процесса.

При переходе на новую сцену возникает экран загрузки рисунке 16.

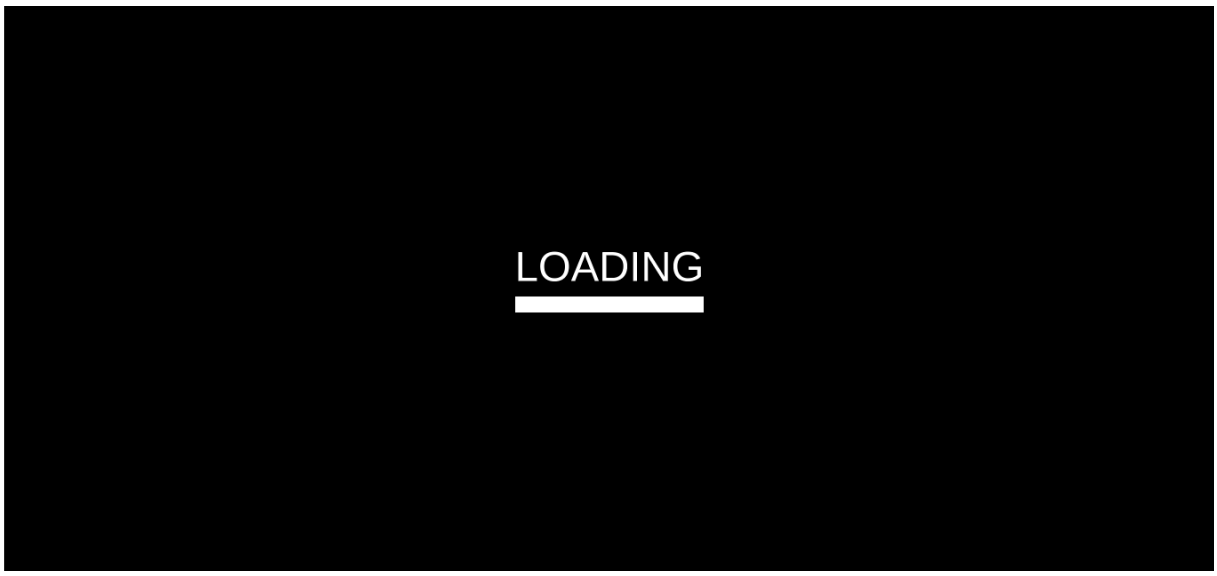

Рис. 16. Экран загрузки

При нажатии кнопки «Inventory» во время процесса игры, игроку открывается инвентарь, в котором он может наблюдать найденные им предметы. Интерфейс инвентаря меню представлен на рисунке 17.

Рис. 17. Интерфейс инвентаря

3.5. Реализация персонажа

Игровой персонаж является объектом приложения, а также он находится под контролем пользователя. При помощи компонента Regitbody на игрового персонажа действуют законы физики. Для перемещения по игро-

вой сцене и взаимодействия с объектами был реализован метод OnClick в скрипте Player Script, который изображен на рисунке 18. Для выполнения передвижения или взаимодействия требуется навести курсором на предмет или в любую точку, которая находится на сцене и нажать на левую кнопку. Компонент Capsule Collider и Nav Mesh Agent [9] позволяют игроку корректно передвигаться по игровой сцене, например, сталкиваться с препятствиями, стенами. Объект игрового персонажа включает в себя компоненты, которые изображены на рисунке 19.

```
void OnClick()
{
    Debug.Log("Left Clicked!");
    RaycastHit hit;
    Ray camToScreen = mainCamera.ScreenPointToRay(Input.mousePosition);
    if (Physics.Raycast(camToScreen, out hit, Mathf.Infinity,
        interactLayer))
    {
        if (hit.collider != null)
        {
            Interactable interactable =
                hit.collider.GetComponent<Interactable>();
            if (interactable != null)
            {
                if (!interactable.LookOnly)
                    MovePlayer(interactable.InteractPosition());
                interactable.Interact(this);
             }
             else
             {
                MovePlayer(hit.point);
             }
        }
    }
}
```

Рис. 18. Реализация метода OnClick

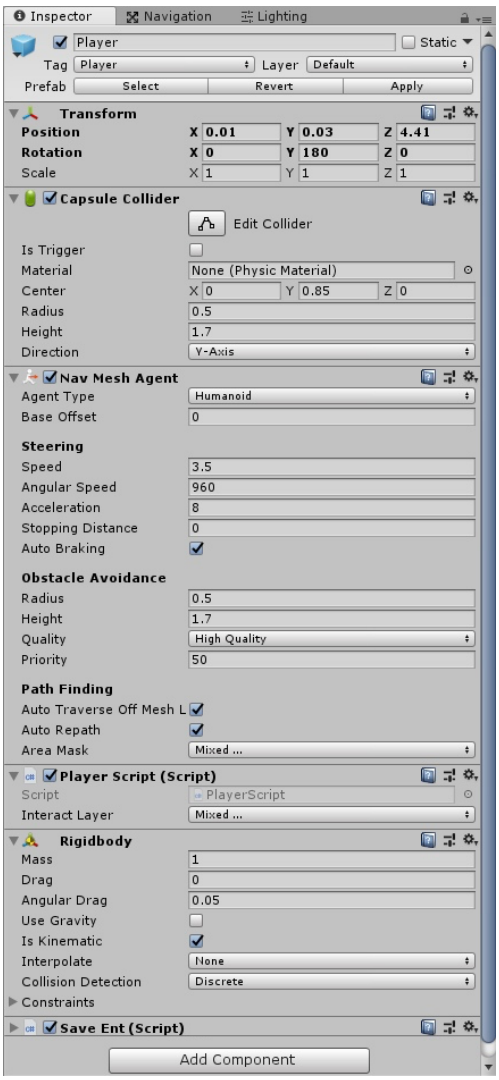

Рис. 19. Реализация сцены персонажа

1. Transform – компонент, который определяет положение, вращение и масштаб объекта на сцене [7].

2. Capsule Collider – это коллайдер, который состоит из двух полусфер, соединённых между собой цилиндром [8]. Используется для определения столкновений с объектами на игровой сцене (препятствия, стены).

3. Nav Mesh Agent – это компонент, который позволяет передвигаться по сцене благодаря поверхностям, которые созданы структурой данных навигационной системы Nav Mesh [9].

4. Player Script – это скрипт, который реализует перемещение и взаимодействие с различными объектами на игровой сцене.

5. Rigidbody – это компонент, который управляет объектом через имитацию физики [10].

6. SaveEnt – это скрипт, который сохраняет или загружает координаты игрового объекта.

3.6. Реализация диалогов

Для повествования сюжета в игровом приложении была создана система диалогов, которая представляет собой прозрачное окно с текстом и две кнопки, которые предоставляют варианты ответных действий или реплик. Пример диалога представлен на рисунке 20.

Рис. 20. Пример диалога

Для создания диалогов был написан скрипт Dialog System, который отображает диалог персонажей во время игрового процесса, фрагмент скрипта, где реализован метод ShowMultipleMessages, который выводит текст в окне диалога, представлен на рисунке 21.

```
IEnumerator ShowMultipleMessages(bool useDialog,
    List<Actions> chainActions = null)
{
    messageText.text = currentMessages[msgId];
    while(msgId < currentMessages.Count)
    {
        if (Input.GetKeyDown(KeyCode.Space) ||
            (Input.GetMouseButtonDown(0) && Extensions.IsMouseOverUI()))
        {
            msgId++;
            if (msgId < currentMessages.Count)
                messageText.text = currentMessages[msgId];
            if (!useDialog && msgId == currentMessages.Count)
            {
                if (chainActions != null)
                    Extensions.RunActions(chainActions.ToArray());
            }
         }
         if (useDialog && msgId == currentMessages.Count - 1)
            yesButton.transform.parent.gameObject.SetActive(true);
         yield return null;
        }
    if (!useDialog)
        panel.SetActive(false);
}
```

Рис. 21. Фрагмент скрипта редактора Dialog System

Для изменения, добавления реплик или ответных действий был создан пользовательский редактор для Unity [11], где внутри его можно добавлять реплики персонажей, а также ответные действия на них. Редактор изображен на рисунке 22. Фрагмент скрипта редактора PhraseControllerEditor, где реализуется его интрфейс, представлен на рисунке 23.

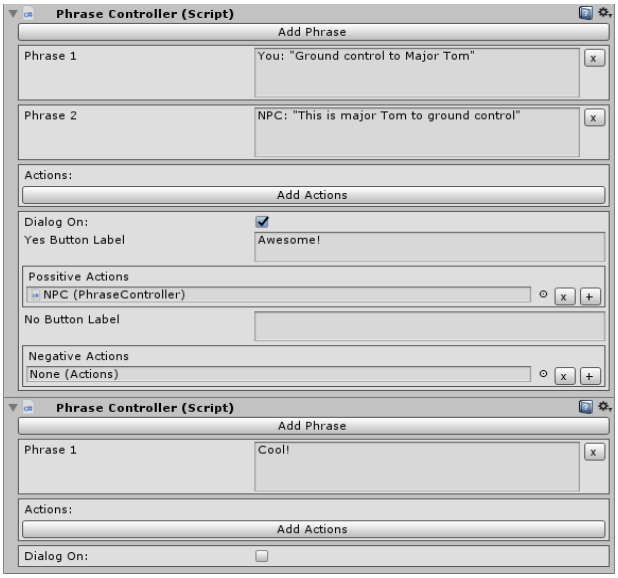

Рис. 22. Редактор диалогов

```
public override void OnInspectorGUI()
{
    serializedObject.Update();
    if (GUILayout.Button("Add Phrase"))
    {
        s_messages.InsertArrayElementAtIndex(s_messages.arraySize);
    }
    for (int i = 0; i < s_messages.arraySize; i++)
    {
        DrawMessagesEntry(s_messages.GetArrayElementAtIndex(i),
            "Phrase " + (i+1), i);
    }
    EditorExtensions.DrawActionsArray(s_chainActions, "Actions: ");
    GUILayout.BeginVertical("box");
    EditorGUILayout.PropertyField(s_enableDialog,
        new GUIContent("Dialog On:"));
    if (s_enableDialog.boolValue)
    {
        EditorGUILayout.PropertyField(s_yesText,
            new GUIContent("Yes Button Label"), GUILayout.Height(30f));
        EditorExtensions.DrawActionsArray(s_yesActions,
            "Possitive Actions");
        EditorGUILayout.PropertyField(s_noText,
            new GUIContent("No Button Label"), GUILayout.Height(30f));
        EditorExtensions.DrawActionsArray(s_noActions, "Negative Actions");
     }
     GUILayout.EndVertical();
     serializedObject.ApplyModifiedProperties();
}
```

Рис. 23. Реализация интерфейса редактора диалогов

3.7. Реализация уровней

Для создания сюжета игры была создана диаграмма активности сюжета, которая отображает нелинейную последовательность заданий, выполнение которых влияет на финальную концовку. Диаграмма изображена на рисунке 24.

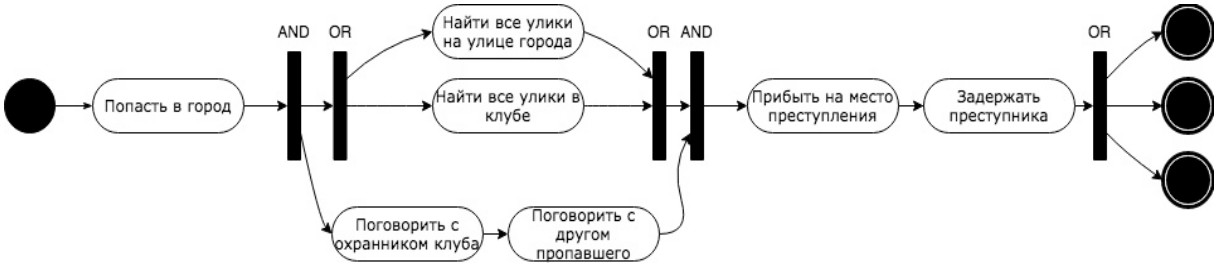

Рис. 24. Диаграмма активности сюжета

Для перемещения персонажа по сценам уровней была использована система навигации Unity [9], которая создаёт разметку по всей сцене и не

позволяет игровым объектам проходить сквозь препятствия. Пример реализации системы навигации на уровне изображен на рисунке 25.

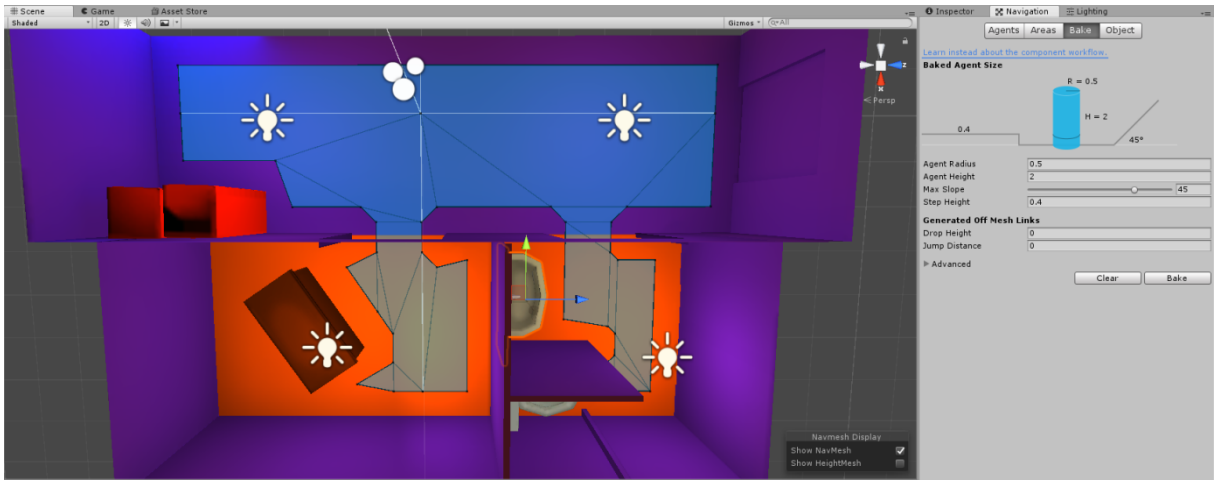

Рис. 25. Система навигации Unity

Для реализации взаимодействия игрока с неигровыми персонажами помимо диалогов были созданы предметы для более детального раскрытия сюжета. В скрипте ItemDatabase был создан обобщённый список ItemNames, который заполняется пользователем при помощи пользовательского редактора Item Data Editor, который изображен на рисунке 26, взаимодействующий с пользовательским asset меню [12] ItemData, представленное на рисунке 27, где хранятся данные созданных предметов. Реализация интерфейса меню представлена на рисунке 28. Игровой персонаж может использовать предметы при помощи скрипта Action Item, который изображен на рисунке 29.

```
public override void OnInspectorGUI()
{
    serializedObject.Update();
    if (GUILayout.Button("Add Item"))
    {
        Item newItem = new Item(s_items.arraySize, "", "", null, false);
        source.AddItem(newItem);
    }
    for (int i = 0; i < s_items.arraySize; i++)
    {
        DrawItemEntry(s_items.GetArrayElementAtIndex(i));
    }
    if (GUI.changed)
    ReCalculateID();
    serializedObject.ApplyModifiedProperties();}
```

Рис. 26. Фрагмента скрипта редактора Item Database Editor

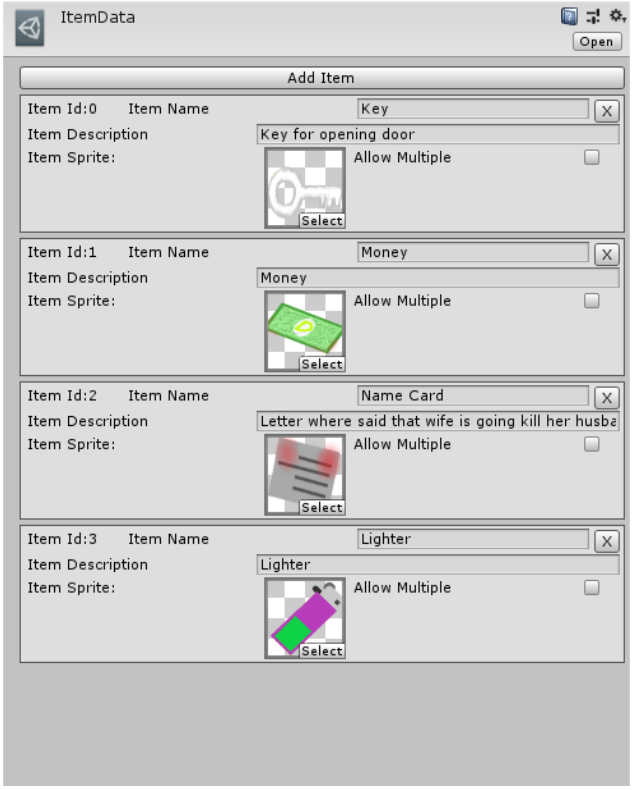

Рис. 27. Пользовательское asset меню ItemData

```
void DrawItemEnt(SerializedProperty item)
{
    GUILayout.BeginVertical("box");
    GUILayout.BeginHorizontal();
    EditorGUILayout.LabelField("Item Id:" +
        item.FindPropertyRelative("itemId").intValue,
            GUILayout.Width(75f));
    EditorGUILayout.PropertyField(item.FindPropertyRelative("itemName"));
    if (GUILayout.Button("X", GUILayout.Width(20f)))
    {
        s_itemsName.DeleteArrayElementAtIndex(
            item.FindPropertyRelative("itemId").intValue);
        s_items.DeleteArrayElementAtIndex(
            item.FindPropertyRelative("itemId").intValue);
        ReCalculateID();
        return;
     }
     GUILayout.EndHorizontal();
     EditorGUILayout.PropertyField(item.FindPropertyRelative(
         "itemDescription"));
     GUILayout.BeginHorizontal();
     item.FindPropertyRelative("itemSprite").objectReferenceValue =
        EditorGUILayout.ObjectField("Item Sprite: ",
            item.FindPropertyRelative("itemSprite").objectReferenceValue,
                typeof(Sprite), false);
     EditorGUILayout.PropertyField(item.FindPropertyRelative(
         "allowMultiple"));
     GUILayout.EndHorizontal();
     GUILayout.EndVertical();
}
```

Рис. 28. Фрагмента скрипта редактора ItemDataEditor

```
public override void Act()
{
    if (giveItem)
    {
        int itemOwnd = DataManager.Instance.Inventory.CheckAmount(CurItem);
        if (itemOwned > 0)
        {
            if (CurItem.AllowMultple && amount <= itemOwned)
            {
               DataManager.Instance.Inventory.ModifyItemAmount(CurItem,
                   amount);
               Extensions.RunActions(yesActions);
            }
            else if (!CurItem.AllowMultple && itemOwned == 1)
            {
               DataManager.Instance.Inventory.ModifyItemAmount(CurItem,
                    -itemOwned);
               Extensions.RunActions(yesActions);
            }
            else
            {
               Extensions.RunActions(noActions);
            }
            }
        else
        {
            Extensions.RunActions(noActions);
        }
     }
    else
    {
        if (CurItem.AllowMultple)
        {
            DataManager.Instance.Inventory.ModifyItemAmount(CurItem,
                amount);
            Extensions.RunActions(yesActions);
        }
        else if (!CurItem.AllowMultple)
        {
            if (DataManager.Instance.Inventory.CheckAmount(CurItem)
                == 1)
            {
               Extensions.RunActions(noActions);
            }
            else
            {
               DataManager.Instance.Inventory.ModifyItemAmount(CurItem, 1);
               Extensions.RunActions(yesActions);
            }
        }
    }
}
```

Рис. 29. Фрагмента скрипта ItemActions

4. ТЕСТИРОВАНИЕ

Тестирование [21] – проверка соответствия между реальными и ожидаемыми поведениями программы, осуществляемыми на конечном наборе тестов, выбранных определенным образом. В таблице 1 приведен протокол функционального тестирования всех аспектов приложения.

Табл. 1. Функциональное тестирование.

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
1	Отображение окна игрового приложения	Запустить игровое приложение	На экране отобразилось главное меню игры	Да
2	Выход из игрового приложения	В игровом приложении нажать кнопку «X»	Происходит выход из игрового приложения	Да
3	Начать игровой процесс	В главном меню нажать кнопку «New Game»	Начинается игровой процесс	Да
4	Загрузить игровой процесс	Нажать кнопку «Load Game» в главном меню	Происходит загрузка сохранённого игрового процесса, который совершил пользователь	Да
5	Перемещение игрового персонажа	Навести курсором мыши на точку, находящуюся на игровой сцене, и нажать на левую кнопку мыши	Игровой персонаж перемещается в точку на игровой сцене, которая указана пользователем	Да
6	Взаимодействие с игровыми объектами	Навести курсором мыши на игровой объект, находящийся на игровой сцене, и нажать на левую кнопку мыши	Игровой персонаж передвигается к игровому объекту, на который указал пользователь, и открывается диалоговое окно, отображающие действие или диалог.	Да
7	Получение игрового предмета	Начать взаимодействие с игровым объектом	Игрок получает игровой предмет, который добавляется в инвентарь	Да

Окончание табл.1

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
8	Использование игрового предмета	Начать взаимодействие с игровым объектом	Происходит проверка наличия нужного предмета, после чего отображается диалоговое окно с описание действия	Да
9	Сохранение процесса	Нажать на кнопку «Save» во время игрового процесса	Происходит сохранение игрового процесса	Да
10	Просмотр инвентаря	Нажать на кнопку «Инвентарь» во время игрового процесса	Происходит отображение инвентаря и предметов, которые получил игрок во время игры	Да

ЗАКЛЮЧЕНИЕ

Основные результаты

В результате данной работы была разработана компьютерная игра для операционной системы Windows в жанре квест на платформе Unity, что было главной целью выпускной квалификационной работы.

Для реализации поставленной цели были выполнены следующие задачи:

1) выполнен обзор аналогов игрового приложения;
2) выполнен обзор средств для реализации игрового приложения;
3) спроектировано игровое приложение;
4) реализовано игровое приложение;
5) проведено тестирование игрового приложения.

ЛИТЕРАТУРА

1. About – Pygame wiki: [Электронный ресур] URL: https://www.pygame.org/wiki/about (дата обращения: 15.02.2020).

2. Adams E. Fundamentals of Game Design. – 2nd edition. – Berkeley, CA: New Riders, 2010. – 700 p.

3. Get Paint 3D – Microsoft Store [Электронный ресурс] URL: https://www.microsoft.com/en-us/p/paint-3d/9nblggh5fv99 (дата обращения: 16.02.2020).

4. Saunders K., Novak J. Game Development Essentials: Game Interface Design. – USA: Cengage Learning, 2012. – 320 p.

5. Timothy Frattesi, Douglas Griesbach, Jonathan Leith, Timothy Shaffer. Replayability of Video Games. — 2011. — 244 c.

6. Дин Л., Дон У. Принципы работы с требованиями к программному обеспечению. Унифицированный подход. – М.: Издательский дом "Вильямс", 2002. – 448 с.

7. Документация Unity. Transform. [Электронный ресурс] URL: https://docs.unity3d.com/ru/current/Manual/class-Transform.html (дата обращения: 15.02.2020).

8. Документация Unity. CapsuleCollider. [Электронный ресурс] URL: https://docs.unity3d.com/ru/current/Manual/class-CapsuleCollider.html (дата обращения: 15.02.2020).

9. Документация Unity. Navigation System in Unity [Электронный ресурс] URL: https://docs.unity3d.com/ru/current/Manual/nav-NavigationSystem.html (дата обращения: 15.02.2020).

10. Документация Unity. Regitbody. [Электронный ресурс] URL: https://docs.unity3d.com/ru/current/ScriptReference/Rigidbody.html (дата обращения: 15.02.2020).

11. Документация Unity. Custom Editors. [Электронный ресурс] URL: https://docs.unity3d.com/Manual/editor-CustomEditors.html (дата обращения: 15.02.2020).

12. Документация Unity. CreateAssetMenuAttribute [Электронный ресурс] URL: https://docs.unity3d.com/ScriptReference/CreateAssetMenuAttribute.html (дата обращения: 15.02.2020).

13. Официальный сайт Amanita Design [Электронный ресурс] URL: https://amanita-design.net/ (дата обращения: 15.02.2020).

14. Официальный сайт Atari [Электронный ресурс] URL: https://www.atari.com/ (дата обращения: 15.02.2020).

15. Официальный сайт Quantic Dream [Электронный ресурс] URL: https://www.quanticdream.com/ (дата обращения: 15.02.2020).

16. Официальный сайт Unreal Engine. [Электронный ресурс] URL: https://www.unrealengine.com (дата обращения 10.02.2020).

17. Официальный сайт Unity: [Электронный ресурс] URL: https://unity.com/ru (дата обращения: 15.02.2020).

18. Официальный сайт UnityAssetStore: [Электронный ресурс] URL: https://assetstore.unity.com/ (дата обращения: 15.02.2020).

19. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е изд. — СПб.: Питер, 2013. — 896 с.

20. Соловьев С.В., Цой Р.И., Гринкруг Л.С. Технология разработки прикладного программного обеспечения. – М.: Изд-во Российской академии естествознания, 2011. – 208 с.

21. Тамре Л. Введение в тестирование программного обеспечения. – М.: Вильямс, 2003. – 368 с.

22. Торн А. Искусство создания сценариев в Unity. – М.: ДМК-Пресс, 2016. – 360 с.

23. Хританков А.С. Проектирование на UML [Текст] / А.С. Хританков, В.А. Полежаев, А.И. Андрианов // Директ-Медиа: сб. статей. – Москва, 2018. – 242 с.