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 «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Д. В. Топольский
«___»_____ 2022 г.

РАЗРАБОТКА 2D-ИГРЫ С ПРОЦЕДУРНОЙ ГЕНЕРАЦИЕЙ И ВСТРОЕННЫМ РЕДАКТОРОМ УРОВНЕЙ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ-09.03.01.2021.257 ПЗ ВКР

Руководитель работы,
к.т.н., доцент кафедры ЭВМ
_____ Е. С. Ярош
«___»_____ 2022 г.

Автор работы,
студент группы КЭ-406
_____ А. В. Цветков
«___»_____ 2022 г.

Нормоконтролёр,
к.п.н., доцент кафедры ЭВМ
_____ М. А. Алтухова
«___»_____ 2022 г.

Челябинск 2022

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 «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Д. В. Топольский
«___»_____ 2022 г.

ЗАДАНИЕ
на выпускную квалификационную работу бакалавра
студенту группы КЭ-406
Цветкову Андрею Владимировичу,
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. **Тема работы:** «Разработка 2D-игры с процедурной генерацией и встроенным редактором уровней»

2. **Срок сдачи студентом законченной работы:** 1 июня 2022 г.

3. **Исходные данные к работе.**

Обеспечить основной функционал приложения:

1) реализацию игровой логики, позволяющей завершать уровни и получать статистику о прохождении;

2) создание уровня с помощью процедурной генерации;

3) создание уровня в редакторе уровней;

4) создание уровня с помощью процедурной генерации внутри редактора уровней для последующей доработки сгенерированного автоматически уровня вручную;

5) сохранение уровней, созданных любым из способов, в файл.

Предусмотреть 2 типа пользователей, каждый из которых сможет проходить уровни, а также генерировать их с помощью процедурной генерации. При этом доступ к редактору уровней должен быть только у

одного из них.

4. **Перечень подлежащих разработке вопросов:**

– анализ аналогов разрабатываемого продукта;

– формирование технического задания;

– выбор среды и средств реализации;

– проектирование продукта;

– реализация продукта;

– тестирование продукта.

5. **Дата выдачи задания:** 1 декабря 2021 г.

Руководитель работы _____/*Е. С. Ярош*/

Студент _____/*А. В. Цветков*/

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы	10.03.2022	
Разработка модели, проектирование	21.03.2022	
Реализация системы	04.04.2022	
Тестирование, отладка, эксперименты	25.04.2022	
Компоновка текста работы и сдача на нормоконтроль	16.05.2022	
Подготовка презентации и доклада	24.05.2022	

Руководитель работы _____/*Е. С. Ярош*/

Студент _____/*А. В. Цветков*/

АННОТАЦИЯ

А. В. Цветков. Разработка 2D-игры с процедурной генерацией и встроенным редактором уровней. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШЭКН; 2022, 52 с., 26 ил., библиогр. список – 26 наим.

В рамках выпускной квалификационной работы разрабатывается 2D игра с возможностью как процедурной генерации уровней, так и их ручной сборки во встроенном редакторе.

Работа состоит из 5 глав. В первой производится анализ предметной области, во второй определяются требования, в третьей – выполняется проектирование, в четвертой показана реализация, а в пятой – результаты тестирования.

Пояснительная записка к выпускной квалификационной работе оформлена в текстовом редакторе LibreOffice Writer версии 7.3.2.2.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Мировая индустрия компьютерных игр растёт и развивается уже достаточно продолжительное время. Не является исключением и российский сегмент. Так, по данным портала Tadviser [1], объём российского рынка видеоигр в 2021 году вырос до 158 мрлд. рублей, что на 7,7% больше по сравнению с предыдущим, 2020 годом.

Игры, в которых грамотно проработаны ключевые моменты, способствующие удержанию игроков, приносят неплохую прибыль. Таких моментов много, но один из самых важных из них – это увлекательность игрового процесса. Как правило, действие игр разворачивается на уровнях (картах, мирах). Поэтому очень важно, чтобы они были грамотно спроектированы, ведь именно от их проработанности зависит, какие ощущения от игры будет испытывать игрок во время прохождения конкретной локации.

Уровни могут создаваться несколькими способами. Чаще всего, отдельный специалист в команде разработчиков – дизайнер уровней, собирает их вручную. Для удобства работы таких специалистов зачастую разрабатывается редактор уровней – программное обеспечение, позволяющее проектировать уровни. Ещё один способ создания уровней – их автоматическая генерация по заданному алгоритму, то есть процедурная генерация. В данном случае уровень может создаваться полностью автоматически, но чаще всего предполагается ввод некоторых начальных параметров для генерации, таких как размер уровня, количество определенных структур, и т. д.

Наличие редактора уровней заметно упрощает и ускоряет процесс создания новых игровых локаций. Возможность процедурной генерации новых уровней делает каждый из них уникальным, а это значит, что при их прохождении игрок будет получать новый игровой опыт. Поэтому разработка игры с редактором уровней, а также с возможностью их процедурной генерации, является достаточно актуальной задачей.

Целью данной выпускной квалификационной работы является разработка 2D игры с процедурной генерацией и встроенным редактором уровней.

В соответствии с обозначенной выше целью работы, были сформулированы следующие задачи:

– определить существующие на данный момент решения с похожим функционалом, рассмотреть их достоинства и недостатки;

– на основании проведённого анализа осуществить постановку задачи и сформировать техническое задание;

– провести сравнение доступных методов и средств реализации, по его результатам определить те, которые будут использоваться для разработки;

– разработать проект будущей игры;

– с использованием выбранных средств реализации по созданному ранее проекту реализовать игру, отвечающую требованиям, перечисленным в техническом задании;

– провести тестирование реализованной игры для проверки её работоспособности.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Обзор аналогов

В настоящее время у многих игр имеется собственный редактор уровней. Иногда он бывает встроенным в саму игру, но чаще всего он является отдельной программой. В этом случае отличаются способы его распространения. Такой редактор уровней может:

– находиться в закрытом доступе и использоваться только компанией, разрабатывающей игру. В ряде таких случаев разработчиками для каких-то целей (например, проведения конкурса на создание лучшего уровня среди игроков) может быть опубликована одна из версий редактора;

– поставляться вместе с игрой;

– быть доступным для скачивания на официальных ресурсах компании-разработчика.

В ряде случаев, когда игра не обладает редактором уровней либо он находится в закрытом доступе, игроки-энтузиасты создают неофициальный редактор уровней.

Процедурная генерация – тоже распространенное явление в играх. Чаще всего она позволяет создавать новые уровни прямо из интерфейса игры. Иногда она является частью редактора карт.

Рассмотрим несколько игр, которые обладают редактором уровней, а некоторые и процедурной генерацией.

TrackMania – это серия аркадных автосимуляторов для платформ Windows, Nintendo DS, и Wii. Есть возможность одиночной и многопользовательской игры, причем в обоих случаях на выбор предлагается несколько различных режимов. Изначально имеется 65 испытательных трасс для прохождения. Присутствует внутриигровой редактор для создания своих собственных трасс (рисунок 1), видеостудия для реализации собственных фильмов и покрасочная для изменения транспорта.

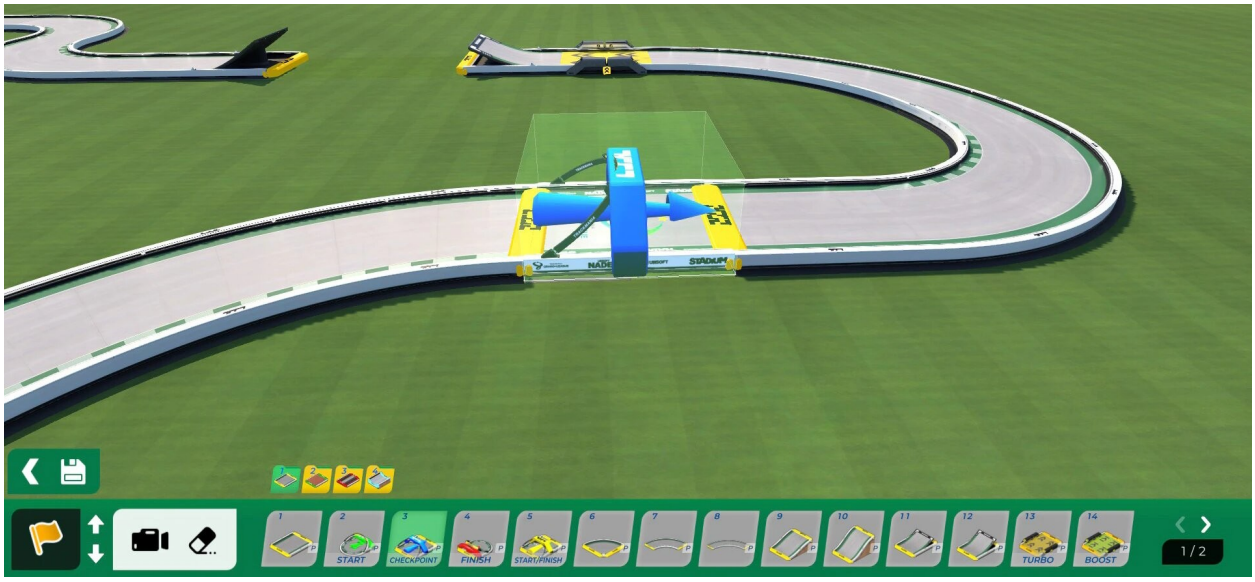

Рисунок 1 – Редактор трасс в игре TrackMania [2]

Одной из отличительных особенностей игры является использование открытой спецификации и открытых форматов [3]. Это позволяет модифицировать игру, добавлять дополнения, писать скрипты. Например:

– создавать новые модели и раскраски машин (формат 3DS и DDS);

– создавать визуальные модификации;

– использовать внутриигровой редактор камер;

– использовать форматирование текста (например, в нике).

Heroes of Might and Magic – фэнтезийная серия компьютерных игр в жанре пошаговой стратегии с элементами RPG, на данный момент включающая в себя семь основных частей [4]. В играх серии Heroes of Might and Magic игроку предстоит сражаться с другими людьми или компьютерными противниками (NPC) за обладание городами, источниками ресурсов, сокровищами и артефактами. В процессе игры он управляет героями – игровыми персонажами, которые исследуют глобальную карту и возглавляют армии существ во время боя, а также способны применять магию.

С игрой поставляется редактор уровней – отдельное ПО для создания своих собственных карт (рисунок 2). В нем, помимо прочего, есть возможность создать уровень с помощью процедурной генерации, после чего менять его вручную.

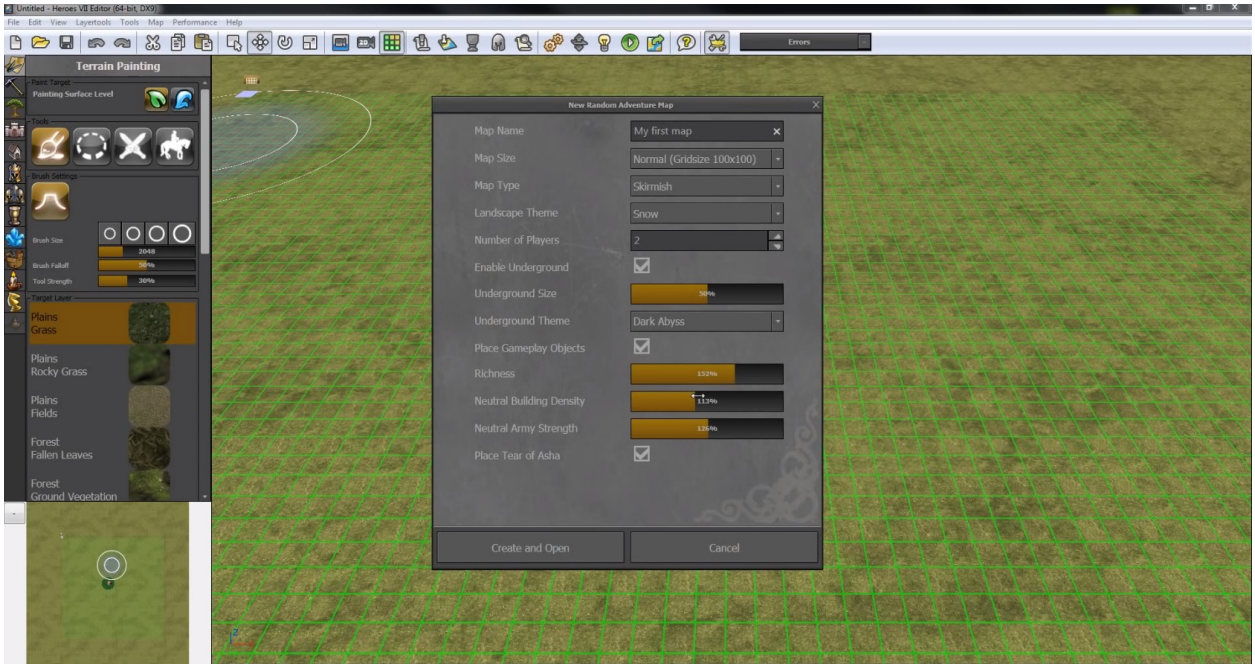

Рисунок 2 – Редактор уровней в игре Might & Magic Heroes VII [5]

Создаваемые уровни сохраняются в файл со специальным расширением (например, для пятой версии игры это .h5m). Возможности протестировать уровень в редакторе нет, только в самой игре.

Besiege – компьютерная игра в жанре головоломки-конструктора [6]. Игра позволяет создавать диковинные средневековые осадные машины, чтобы противостоять замкам или армиям. Игроки выбирают из набора механических частей, которые можно соединить вместе, чтобы построить машину. Переходя от уровня к уровню, выполнять необходимые задания становится всё тяжелее и игроку приходится модернизировать свою машину под конкретную цель.

Доступна одиночная и многопользовательская игра. Изначально доступно 54 уровня, но можно создавать свои собственные в редакторе уровней, который является частью игры (рисунок 3). Протестировать созданный уровень можно в том числе и с другими игроками в многопользовательском режиме.

Рисунок 3 – Редактор уровней в игре Besiege [7]

Geometry Dash – компьютерная игра в жанре 2D-платформера [8]. Игровой процесс заключается в прохождении уровня со множеством препятствий под ритмичную музыку. Помимо встроенных уровней (всего игра содержит 21 такой уровень), игроки могут создавать свои и выкладывать их для прохождения другими игроками (рисунок 4).

Интересно, что по редактору уровней существует официальная документация, которую, как и сам редактор, можно найти в самой игре.

Рисунок 4 – Редактор уровней в игре Geometry Dash [9]

Portal 2 – компьютерная игра в жанре головоломки от первого лица [10]. Продолжение развивает игровой процесс первой игры, состоящий из головоломок, построенных на использовании специального устройства для создания порталов (разрывов в пространстве), позволяющих игроку мгновенно перемещаться из одного места в другое.

В отличии от всех перечисленных ранее игр, в Portal 2 присутствует сюжетная линия, как в одиночном, так и кооперативном (на двух игроков) режимах игры.

Редактор уровней (также известный как Puzzle Creator) позволяет создавать уровни для обоих режимов игры (рисунок 5). Имеет достаточно простой функционал и доступен внутри игры.

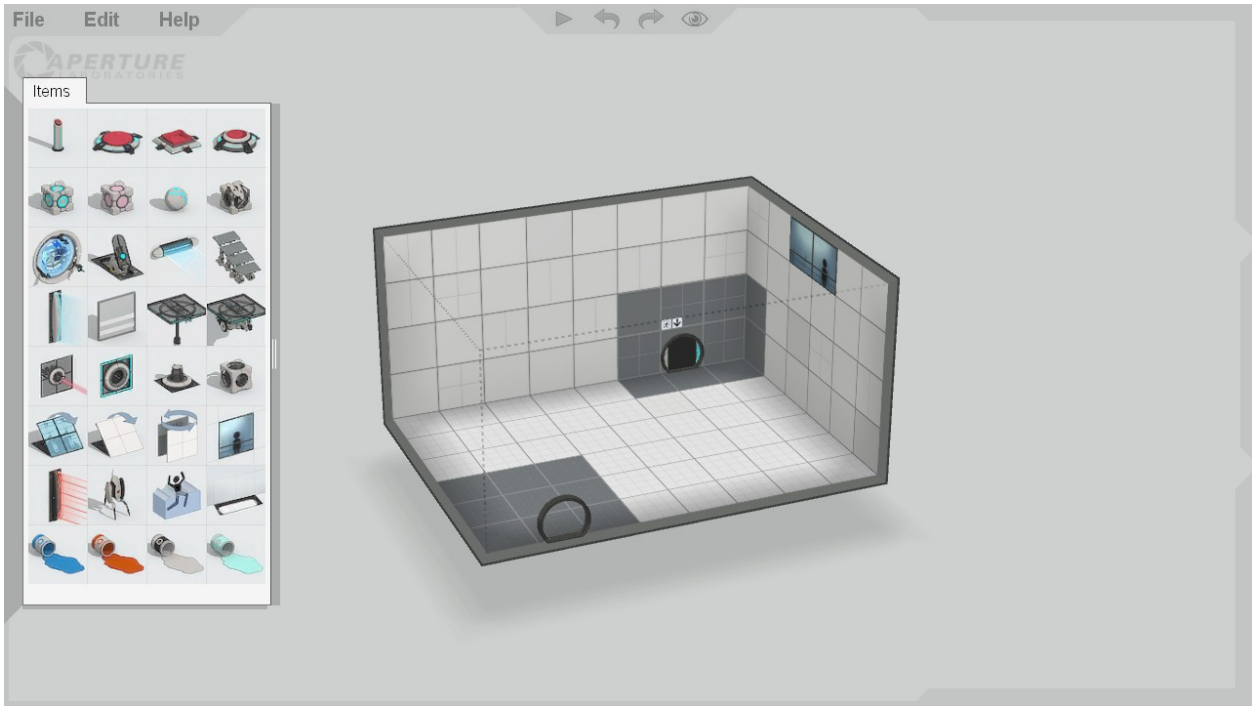

Рисунок 5 – Редактор уровней в игре Portal 2 [11]

Сравним перечисленные игры по интересующим нас критериям. Результат представим в сводной таблице (таблица 1).

Таблица 1 – Сравнительный анализ существующих аналогов

Название игры	**Наличие редактора уровней**	**Редактор уровней встроен в саму игру**	**Возможность процедурно генерировать уровни в редакторе**	**Возможность процедурно генерировать уровни вне редактора**
TrackMania	+	+	-	-
Heroes of Might and Magic	+	-	+	-
Besiege	+	+	-	-
Geometry Dash	+	+	-	-
Portal 2	+	+	-	-

Вывод. Проанализировав результаты сравнения, можно заметить, что в целом редактор уровней – вещь достаточно распространенная. Зачастую он также бывает встроен в саму игру. Но возможность процедурной генерации уровней в играх с редактором уровней встречается достаточно редко. Из этого можно сделать вывод, что разработка игры со всем перечисленным функционалом является актуальной задачей.

1.2 Анализ основных технологических решений

Выбор игрового движка. Прежде всего необходимо определиться, с использованием какого инструмента для создания игр будет идти разработка. Выбор достаточно обширный – различных игровых движков на сегодняшний день существует огромное количество. Рассмотрим несколько наиболее популярных из них [12] и определим наиболее подходящий [13, 14].

Unity – кроссплатформенная среда разработки компьютерных игр, разработанная американской компанией Unity Technologies [15]. Позволяет создавать игры и приложения, работающие более чем на 25 разных платформах. Для написания скриптов используется язык программирования C#. Также присутствует поддержка системы визуального программирования Bolt Visual Scripting, которая позволяет разрабатывать механику игрового процесса или логику взаимодействия с помощью визуальной графической системы вместо написания строк традиционного кода [16].

Достоинства:

– кроссплатформенность;

– есть много шаблонов и примеров работ;

– большая база знаний и сообщество;

– интуитивно понятный и удобный интерфейс.

Недостатки:

– закрытый исходный код;

– в некоторых версиях могут встречаться баги;

– не самая высокая скорость работы.

Unreal Engine – игровой движок, разрабатываемый и поддерживаемый компанией Epic Games [17]. В прошлом движок распространялся на условиях оплаты ежемесячной подписки, с 2015 года Unreal Engine бесплатен, но на определенных условиях. Использует язык C++, поддерживает систему визуального скриптинга Blueprints [18].

Достоинства:

– очень широкий набор инструментов;

– кроссплатформенность;

– достаточно стабильная работа.

Недостатки:

– высокий порог вхождения;

– малый выбор готовых примеров и шаблонов.

GDevelop. Мощный конструктор игр с открытым исходным кодом для создания кроссплатформенных 2D игр любой сложности без знания программирования – вся логика задается с помощью системы событий [19].

Достоинства:

– не требует знаний программирования;

– гибкий и понятный интерфейс;

– полностью бесплатный и с открытым исходным кодом.

Недостатки:

– маленькое сообщество;

– возможно наличие багов;

– позволяет создавать только 2D игры.

Stride – бесплатный кроссплатформенный игровой движок с открытым исходным кодом, первоначально разработанный компанией Silicon Studio [20]. Скрипты создаются на языке C#.

Достоинства:

– схожесть с движком Unity;

– полностью бесплатный и с открытым исходным кодом.

Недостатки:

– достаточно молодой движок, как следствие: маленькое сообщество и сырость некоторых компонентов;

– доступен под меньшее число платформ.

CryEngine – игровой движок, созданный немецкой частной компанией Crytek [21]. Позволяет создавать скрипты на языках C#, C++, а также с помощью Flow Graph.

Достоинства:

– мощные графические инструменты позволяют добиться фотореалистичной графики;

– имеет инструменты для создания продвинутого ИИ.

Недостатки:

– высокий порог вхождения;

– закрытое сообщество;

– нельзя создавать игры для мобильных устройств и консолей, только для ПК.

Проанализировав достоинства и недостатки перечисленных выше движков, делаем выбор в пользу Unity, поскольку именно он больше всего подходит под наши задачи и не имеет критических недостатков.

Выбор ПО для написания скриптов. Сравним два решения от компании Microsoft: Visual Studio и Visual Studio Code [22].

Visual Studio – это полноценная интегрированная среда разработки (IDE). В бесплатной версии имеет ограниченный, но всё ещё богатый функционал. Достаточно тяжеловесна. Доступна только на Windows и macOS.

Visual Studio Code – это редактор кода с функциями IDE. По сравнению с Visual Studio менее функциональный, но зато гораздо более быстрый и легковесный. Доступен на Windows, macOS и Linux.

В нашем случае правильным решением будет использовать Visual Studio Code для написания скриптов.

Для создания текстур и элементов интерфейса также следует выбрать ПО. На данный момент существует огромное количество различных

графических редакторов. Для названных задач подойдет Krita – это бесплатный растровый графический редактор с открытым кодом [23]. Он доступен на Windows, macOS, Linux, BSD и имеет весь необходимый нам функционал.

Для реализации эскизного проекта используем ПО Ramus. Это программный продукт в области управления знаниями предприятия [24]. Позволяет проводить описание, анализ и моделирование бизнес-процессов, а также строить систему классификации и кодирования.

1.3 Вывод

В результате обзора существующих аналогов разрабатываемого продукта выяснилось, что они обладают лишь частью требуемого функционала. Следовательно, разработка полнофункционального решения является актуальной задачей. Для этого в результате сравнительного анализа были выбраны следующие технологические решения: игровой движок Unity со скриптами на языке C#, редактор кода Visual Studio Code для написания скриптов, графический редактор Krita для создания текстур и элементов интерфейса, а также ПО Ramus для реализации эскизного проекта.

2 ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

2.1 Функциональные требования

Готовый продукт должен:

1) при запуске обеспечить появление окна авторизации;

2) обеспечить возможность зарегистрировать нового пользователя указанного типа в окне авторизации;

3) обеспечить возможность авторизоваться под существующим пользователем в окне авторизации;

4) обеспечить возможность автоматического входа в систему в случае выбора пункта «запомнить» в окне авторизации;

5) позволить пользователю выбрать уровень для прохождения из списка существующих уровней;

6) позволить пользователю создать новый уровень по алгоритму процедурной генерации, предложив ввести начальные параметры для генерации;

7) после завершения прохождения уровня пользователем вывести на экран время, за которое был пройден уровень, а также количество набранных очков, рассчитанное по формуле;

8) обеспечить доступ пользователю к его статистике прохождений конкретного уровня;

9) позволить администратору запускать редактор уровней;

10) обеспечить возможность в редакторе уровней создавать уровни, вручную расставляя все элементы будущей локации;

11) обеспечить возможность использовать процедурную генерацию в редакторе уровней для автоматического создания уровня, предложив ввести начальные параметры для генерации;

12) обеспечить возможность в редакторе уровней свободно взаимодействовать с элементами локации, которая была сгенерирована процедурно, позволяя администратору изменять ее точно так же, как и при

создании уровня вручную;

13) обеспечить возможность сохранять созданный в редакторе уровень в файл;

14) позволить администратору пополнять список существующих уровней, загружая новый уровень из файла.

Готовый уровень должен представлять собой прямоугольную область с расставленными по ней игровыми элементами. При создании уровня любым из способов (вручную или с помощью процедурной генерации) необходимо обеспечить наличие на готовом уровне следующих элементов:

1) точка начального местоположения игрока (не более 1 единицы);

2) источники ресурсов, необходимых для завершения уровня (не менее 3 единиц для каждого типа ресурсов);

3) источники ресурсов, необходимых для восстановления шкалы здоровья игрока (не менее 2 единиц для каждого типа ресурсов);

4) источники ресурсов, необходимых для снятия с игрока негативных эффектов (не менее 2 единиц для каждого типа ресурсов);

5) точки начального местоположения агрессивных неигровых персонажей (не менее 2 единиц для каждого типа персонажей);

6) точка завершения уровня (не более 1 единицы).

Указанные требования к характеристикам готового уровня относятся к небольшим локациям и могут изменяться по мере увеличения размеров уровня.

2.2 Требования к надежности

Следующие требования должны обеспечить стабильное функционирование продукта:

1) реализовать проверку введенных при авторизации данных;

2) реализовать проверку добавляемых администратором уровней на существование соответствующего файла и корректность его данных.

2.3 Требования к аппаратному и программному обеспечению

Разрабатываемый продукт должен корректно запускаться и работать на операционных системах Windows/Linux. Минимальные версии операционных систем, а также требования к аппаратному обеспечению соответствуют системным требованиям игрового движка Unity и перечислены в таблице 2.

Таблица 2 – Системные требования

Минимальные требования	**Windows**	**Linux**
Версия ОС	Windows 7 (SP1+), Windows 10 и Windows 11, только 64-разрядные версии	Ubuntu 20.04, Ubuntu 18.04 и CentOS 7
Процессор	Архитектура X64 с поддержкой набора инструкций SSE2	Архитектура X64 с поддержкой набора инструкций SSE2
Графика	Графические процессоры с поддержкой DX10, DX11 и DX12	Графические процессоры Nvidia и AMD с поддержкой OpenGL 3.2+ или Vulkan
Дополнительно	Официальные драйверы	Среда рабочего стола Gnome, работающая поверх оконной системы X11, официального проприетарного графического драйвера Nvidia или графического драйвера AMD Mesa. Другая конфигурация и пользовательская среда, поставляемые с поддерживаемым дистрибутивом.

3 ПРОЕКТИРОВАНИЕ

3.1 Эскизный проект

Создаваемая структура реализует требования, приведенные в техническом задании. Компоненты структуры приведены на рисунках 6-10.

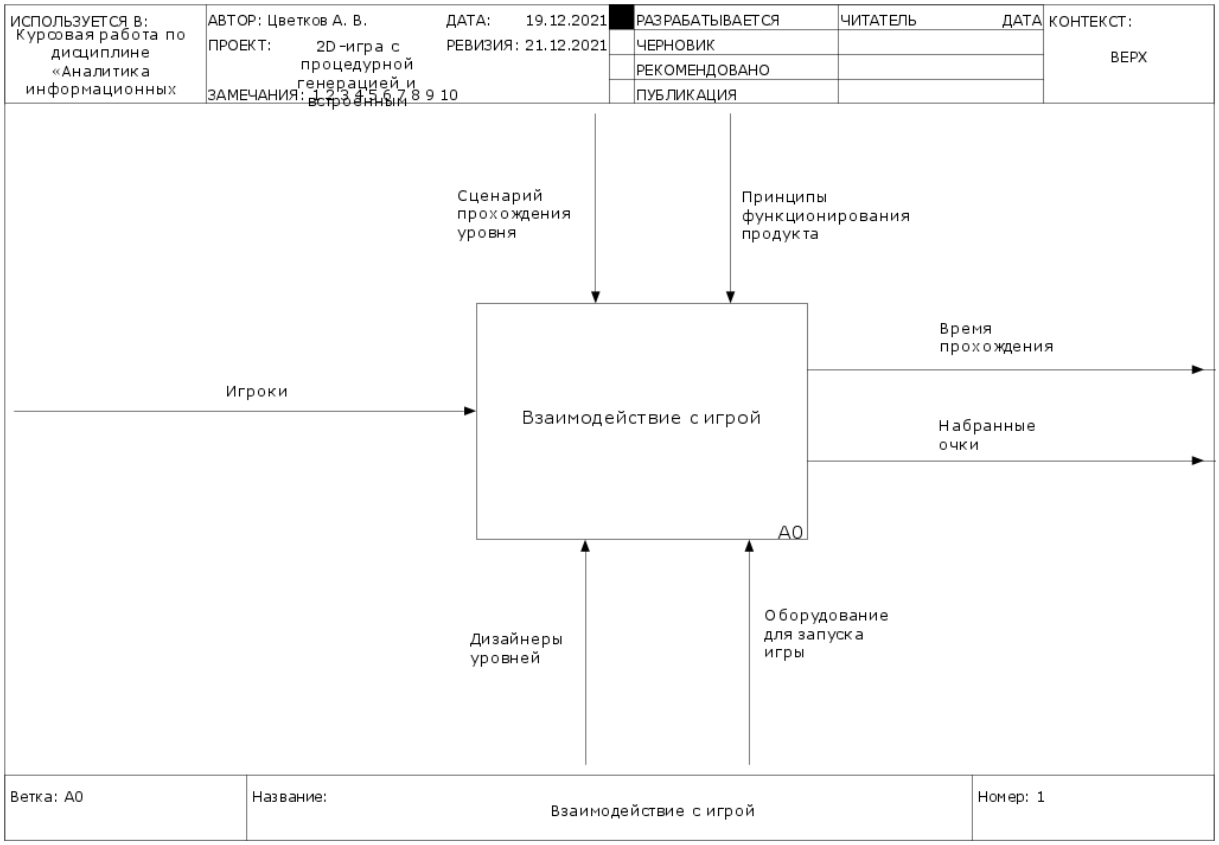

Рисунок 6 – Контекстная диаграмма

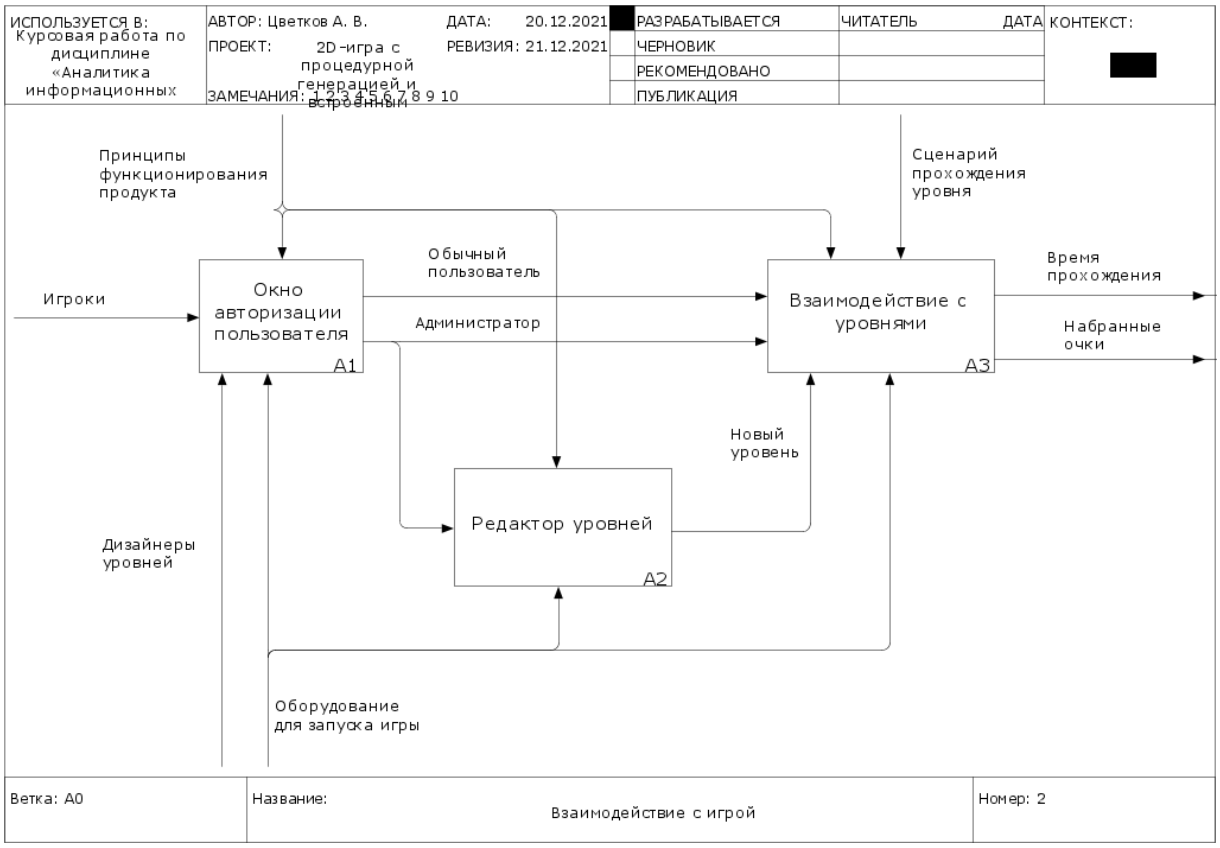

Рисунок 7 – Декомпозиция контекстной диаграммы

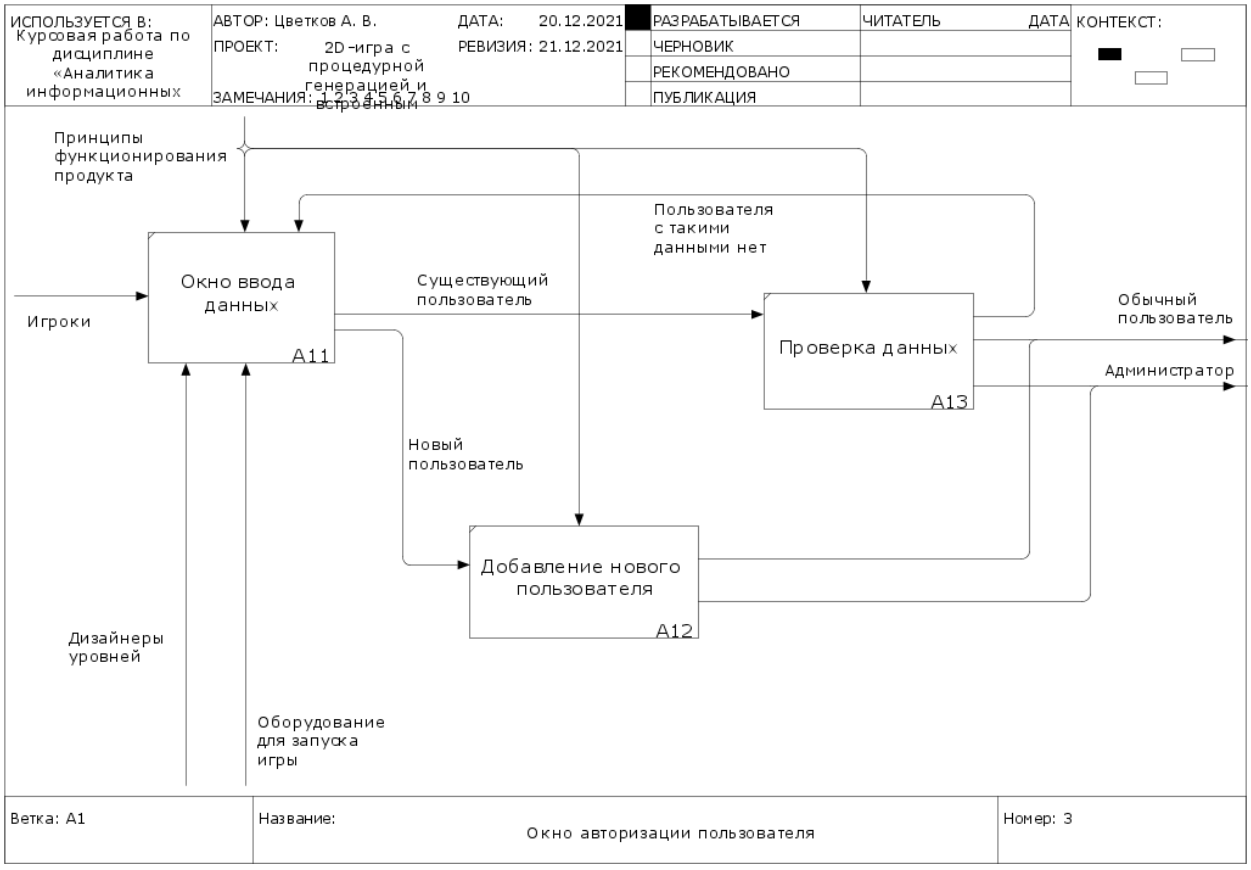

Рисунок 8 – Декомпозиция «Окно авторизации пользователя»

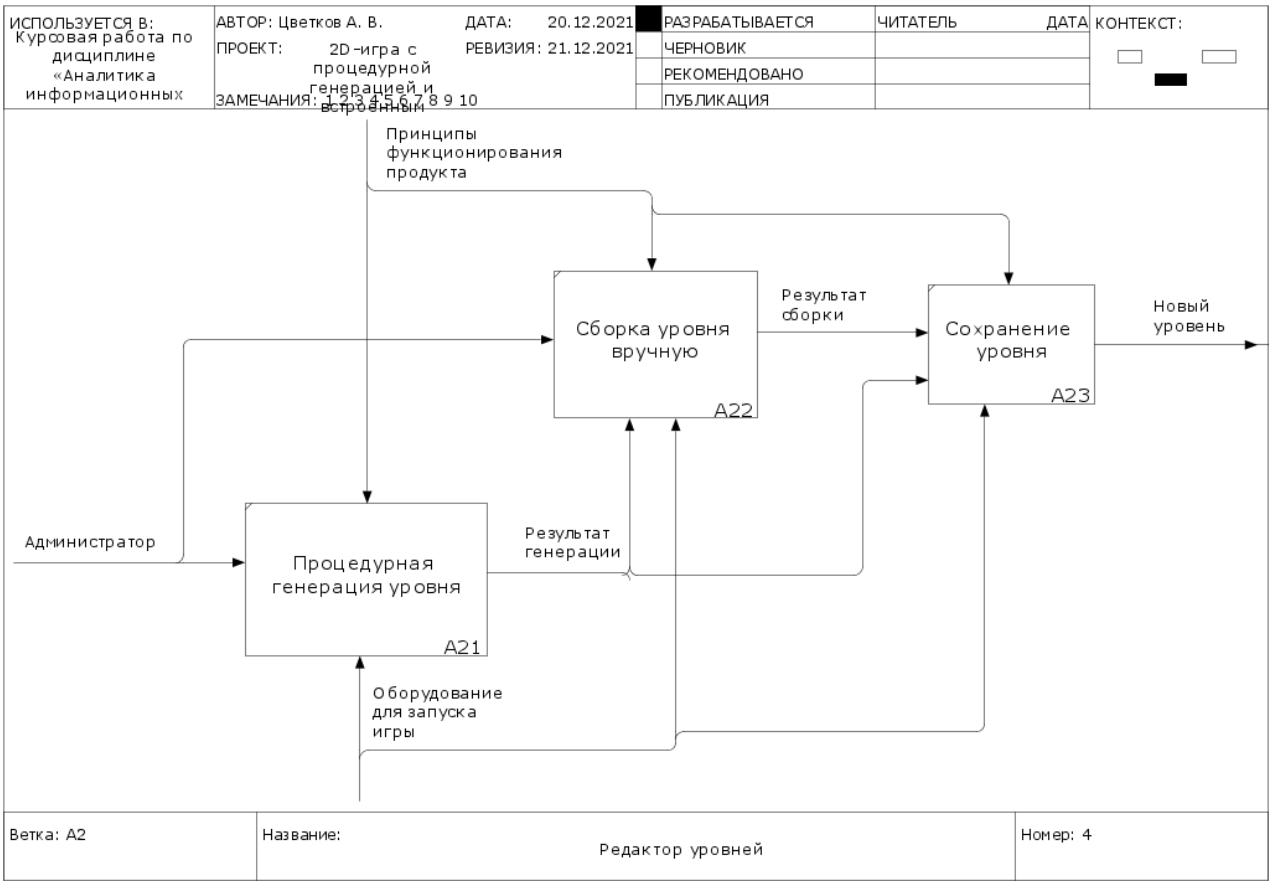

Рисунок 9 – Декомпозиция «Редактор уровней»

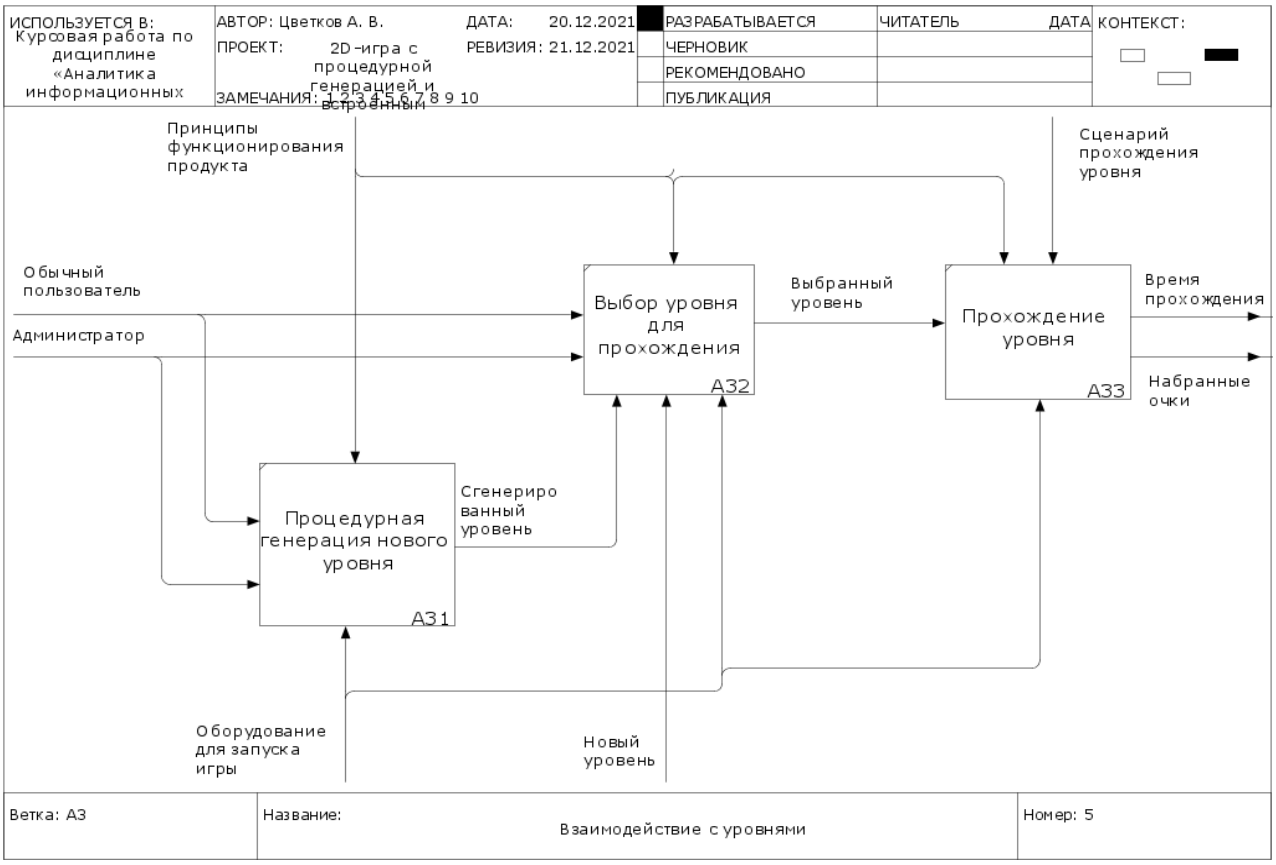

Рисунок 10 – Декомпозиция «Взаимодействие с уровнями»

Как видно из приведенных диаграмм, как игроки, так и дизайнеры уровней начинают свою работу с продуктом с окна авторизации. Введенные данные проверяются – если введенного пользователя не существует, войти не получится. Можно создать нового пользователя и продолжить работу.

Как обычные пользователи, так и администраторы, могут взаимодействовать с уровнями. У них есть выбор: сразу начать проходить уровень, выбрав нужный из списка, либо предварительно сгенерировать новый уровень, после чего тот появится в списке и станет доступным для выбора.

Администраторы могут запускать редактор уровней, в котором могут либо сразу начать собирать уровень вручную, либо предварительно сгенерировать уровень процедурно. В этом случае есть возможность как сразу сохранить уровень, так и начать его корректировку вручную и выполнить сохранение после нее.

3.2 Архитектура предлагаемого решения

3.2.1 Игровая логика

Игрок появляется на уровне в заданном месте. Уровень, как уже было сказано ранее, представляет собой прямоугольную область, ограниченную стенами, за пределы которых ни один игровой объект попасть не может.

Перед игроком стоит задача: набрать заданное количество грибов и ягод, после чего найти и встать на специальный игровой объект – если у игрока будет необходимое количество ресурсов, уровень завершится. При завершении уровня игрок увидит затраченное на прохождение время, а также количество очков, рассчитанное по специальной формуле, которая учитывает в том числе состояние показателей игрока на момент завершения уровня.

Грибы и ягоды добываются из соответствующих источников. Для добычи данных ресурсов игрок должен подойти к источнику поближе и кликнуть по нему – начнется процесс добычи, длящийся некоторое время.

Если в этот момент игрок отойдет слишком далеко от источника, то добыча прервется. Каждый источник имеет ограниченный запас ресурсов, по окончании которого он пропадает с уровня.

Существуют также источники ядовитых грибов и ягод. На уровне они не отличимы от обычных, но добытые из них ресурсы попадают в отдельную ячейку инвентаря, и игрок может это отследить. Данные ресурсы не участвуют в подсчете общего числа грибов и ягод, но они занимают место в рюкзаке игрока, грузоподъемность которого ограничена и отображается в виде отдельной шкалы.

Игрок может при необходимости употребить часть добытых ягод в пищу – это может понадобиться для восстановления шкалы здоровья. При наличии в инвентаре ядовитых ягод, существует шанс, что игрок может их употребить. В таком случае на игрока наложится негативный статус-эффект «Отравление», который снимется через некоторое время, а во время действия будет постепенно снижать здоровье игрока.

На уровне также присутствуют агрессивные неигровые персонажи (NPC). Каждый такой персонаж постоянно находится в движении, логика которого следующая: NPC определяет случайную точку в пределах окружности заданного радиуса, с центром в точке своего первоначального появления, и движется к ней. После ее достижения процесс повторяется.

При приближении игрока к агрессивному NPC слишком близко, он прерывает процесс передвижения в пределах окружности и начинает преследовать игрока. В случае, если игрок отойдет на достаточное расстояние, NPC перестает преследовать его и возвращается к патрулированию своей окружности. Если же он коснется игрока, то нанесет ему урон.

При получении игроком урона от агрессивного NPC есть вероятность, что игрок получит негативный статус-эффект «Кровотечение». Логика работы этого эффекта точно такая же, как и у «Отравления», за исключением того, что он не проходит сам по себе. Для его снятия игрок должен наложить повязку. Для создания таких повязок игрок должен собирать разбросанные по

локации фрагменты материала, из трех единиц которого можно сделать одну повязку.

Агрессивных неигровых персонажей существует два вида: волк и медведь. Они отличаются рядом характеристик, таких как скорость, наносимый урон.

Если шкала здоровья игрока опустится до нуля, уровень завершится досрочно, без внесения данного прохождения в статистику прохождений.

Также существует два вида стен. Первая из них, сделанная в виде кустов, выполняет роль границы уровня, не давая игроку и агрессивным NPC выйти за пределы локации. Вторая сделана в виде бревна и выполняет роль небольшого препятствия для тех же объектов.

3.2.2 Редактор уровней

Редактор уровней предназначен для сборки новых локаций прямо внутри игры. При его запуске игрока встречает шаблон уровня, на котором уже выставлены в единичных экземплярах необходимые для прохождения уровня объекты. Игрок может создавать копии объектов, перемещать их (с помощью курсора мыши, либо задавая координаты вручную). Также можно сгенерировать уровень с помощью процедурной генерации, после чего взаимодействовать уже со сгенерированными объектами.

После завершения расстановки игровых объектов уровень можно сохранить в файл.

3.2.3 Процедурная генерация

Процедурная генерация представляет собой алгоритм, который принимает некоторые входные данные (размер создаваемой локации, а также сложность, которая влияет на количество ядовитых ресурсов и агрессивных NPC), после чего с учетом введенных данных создает уровень.

Сначала создаются границы локации. Далее внутри них расставляются игровые объекты, координаты которых генерируются случайным образом, при этом проверяется расстояние между объектами, чтобы исключить пересечение или слишком близкое местонахождение некоторых из них.

3.2.4 Пользователи

Для того, чтобы попасть в главное меню игры и получить возможность проходить и создавать уровни, необходимо авторизоваться.

Окно авторизации появляется при первом запуске. В нем можно войти под существующим пользователем (которые при первом запуске игры отсутствуют), либо создать нового пользователя, указав, будет ли он принадлежать к группе администраторов или нет. При авторизации можно выбрать пункт «запомнить». В этом случае при последующих запусках игры вход в систему под данным пользователем будет происходить автоматически.

В главном меню отображается логин текущего пользователя, а рядом с ним имеется кнопка, по нажатию которой можно выйти из системы, открыв окно авторизации.

Пользователи, которые при регистрации не были обозначены как администраторы, не могут запускать редактор уровней, а также пополнять список существующих уровней, загружая его из файла. Но основная функциональность игры им доступна – они могут проходить уровни, а также создавать новые с помощью процедурной генерации, не заходя в редактор уровней.

4 РЕАЛИЗАЦИЯ

4.1 Используемые скрипты, классы и методы

Перечислим скрипты, используемые в проекте, а также их назначение, в таблице 3.

Таблица 3 – Используемые скрипты

Название скрипта	**Назначение**	**Содержит классы**
AuthorizationManagement	Организует работу системы авторизации	AuthorizationManagement User UsersList
CameraController	Следование камеры за игроком	CameraController
EnemyMovement	Описывает ИИ неигровых персонажей	EnemyMovement
Health	Нужен для взаимодействия со здоровьем игрока	Health
LevelBuilder	Предназначен для сборки уровня	LevelBuilder
LevelEditorController	Организует работу редактора уровней	LevelEditorController LevelInfo LevelInfoList LevelList
MouseClick	Обрабатывает нажатия курсором на ресурсы	MouseClick
MousePosition	Нужен для перемещения объектов в редакторе уровней	MousePosition
PlayerMovement	Отвечает за передвижение игрока	PlayerMovement
ProcGeneration	Алгоритм процедурной генерации	ProcGeneration
Resourses	Самый основной скрипт, отвечает за взаимодействие с ресурсами	Resourses
ScreenButtons	Обрабатывает нажатия на некоторые кнопки	ScreenButtons
ShowText	Нужен для отображения текста при наведении курсора на объект	ShowText

Рассмотрим содержимое классов более подробно. Перечислим в таблице 4 методы, которые содержит каждый класс, и укажем их назначение.

Таблица 4 – Назначение классов и методов

Название класса	**Назначение класса**	**Название метода**	**Назначение метода**
AuthorizationManagement	Организует работу системы авторизации	void AuthFormInit()	Инициализирует форму авторизации
		void Awake()	Вызывает AuthFormInit(), считывает пользователей из файла и проверяет, нажимал ли кто «запомнить»
		void AddUser()	Добавляет нового пользователя
		void UpdateFile()	Обновляет файл с данными пользователей
		void LogIn()	Вход в аккаунт
		void LogOut()	Выход из аккаунта
CameraController	Привязывает камеру к игроку	void Awake()	Находит игрока среди других объектов
		void Update()	Логика движения камеры
EnemyMovement	Описывает ИИ неигровых персонажей	void Awake()	Находит игрока среди других объектов
		void Update()	Проверяет, достигнута ли целевая точка, обрабатывает задержку урона
		void GenRandomPoint()	Генерирует случайную точку в окружности
		void AngerCheck()	Проверяет расстояние до игрока
		void OnCollisionEnter2D(Collision2D other)	Действия при столкновении с объектом

Продолжение таблицы 4

		void OnCollisionStay2D(Collision2D other)	Действия во время столкновения с объектом
		void OnCollisionExit2D(Collision2D other)	Действия после столкновения с объектом
		void DealDamage()	Нанесение урона
Health	Нужен для взаимодействия со здоровьем игрока	void Awake()	Инициализация шкалы здоровья и UI-элементов
		void Update()	Выполняет проверки
		void DisableBleeding()	Отключает эффект кровотечения
		void TakeDamage(float _damage)	Получение урона
		void SetStatusEffectGUI(string _status)	Включает отображение статус-эффектов
LevelBuilder	Предназначен для сборки уровня	void Awake()	Считывает информацию об уровне из файла, и на ее основе собирает уровень
LevelEditorController	Организует работу редактора уровней	void Awake()	Находит камеру среди других объектов
		void Update()	Контролирует целевой объект и перемещает камеру
		void ChangeCoordinates()	Задает координаты
		void CloneObject()	Клонирует объект
		void NullifyTargetObject()	Сбрасывает выделение объекта
		void DeleteObject()	Удаляет объект
		void RotateObject()	Поворачивает объект на 90 градусов
		void ProcGen()	Выполняет процедурную генерацию уровня сразу в редакторе
		void SaveLevel()	Сохраняет уровень в файл

Продолжение таблицы 4

LevelInfo	Содержит координаты объекта на уровне, его поворот и тег	-	-
LevelInfoList	Содержит массив экземпляра классов LevelInfo	-	-
LevelList	Содержит массив строк, в которых записаны названия уровней	-	-
MouseClick	Обрабатывает нажатия курсором на ресурсы	void Awake()	Находит игрока среди других объектов
MousePosition	Позволяет перемещать объекты в редакторе уровней	async void OnPointerClick(PointerEventData eventData)	Действия при клике на объект
		void Awake()	Находит камеру среди других объектов
		void OnMouseDrag()	Действия при перетягивании объекта мышью
		void OnPointerDown(PointerEventData eventData)	Действия при нажатии кнопки мыши на объекте
PlayerMovement	Отвечает за передвижение игрока	void Update()	Передвигает игрока при нажатии соответствующих клавиш
ProcGeneration	Алгоритм процедурной генерации	void Generation()	Сам алгоритм генерации
		Vector3 GenRandomPoint(string type)	Генерирует случайную точку с проверкой на пересечение с другими объектами
Resourses	Отвечает за взаимодействие с ресурсами	void Awake()	Отключает не нужные в начале игры UI-элементы
		void Update()	Выполняет проверку дистанции до ресурса, отсчитывает время и связанные с ним действия

Продолжение таблицы 4

		void OnTriggerStay2D(Collider2D other)	При нахождении рядом игрока рядом с ресурсами, включает флаг, разрешающий их добычу
		void OnTriggerExit2D(Collider2D other)	При отдалении от ресурсов отключает соответствующий флаг
		async Task StartExtraction()	Запускает процесс добычи ресурсов
		async Task AddResources(string _tag, string _name, int _count)	Добавляет ресурсы
		RemoveResources(string _tag, int _count)	Удаляет ресурсы
		async void CreateDressing()	Создает повязку
		void ApplyDressing()	Применяет повязку
		void Eating()	Задает логику поедания ягод
		float CalculateScore()	Рассчитывает результат игры в очках
ScreenButtons	Обрабатывает нажатия на некоторые кнопки	void LoadLevelList()	Загружает список уровней
		void AddLevel()	Добавляет уровень в список
		void LaunchLevelEditor()	Запускает редактор уровней
		void OpenMenu()	Запускает главное меню
		void StartLevel(string _name)	Запускает уровень
		void RestartLevel()	Перезагружает уровень
		void QuitGame()	Выход из игры
ShowText	Нужен для отображения текста при наведении курсора на объект	void Awake()	Отключает текст в начале игры

Окончание таблицы 4

		void OnMouseOver()	Включает текст при наведении мышью
		void OnMouseExit()	Отключает текст, когда мышь не наведена на объект
User	Содержит поля с информацией о пользователе	-	-
UserList	Содержит массив экземпляров класса User	-	-

Принцип работы некоторых методов стоит раскрыть подробнее.

За поведение камеры во время игры отвечает класс CameraController. В методе void Awake() камера определяет, какой из всех игровых объектов является игроком, по тегу «Player», который есть только у игрока. В методе void Update(), который вызывается каждый кадр, камера в качестве своей целевой точки устанавливает местоположение игрока и движется к ней с определенной скоростью. Точно по такому же принципу работает поиск и преследование игрока агрессивными NPC в классе EnemyMovement. Поиск камеры в методе void Awake() класса LevelEditorController устроен аналогично, только камеру он ищет по тегу «MainCamera». А ее перемещение в редакторе уровней осуществляется удерживанием средней кнопки мыши, что описано в методе void Update().

В классе Health метод void Update() контролирует шкалу здоровья: выводит ее на экран, а также завершает прохождение уровня, если она достигла нуля. Здесь же организована работа со статус-эффектами, так как они напрямую связаны со шкалой здоровья.

Один из самых важных методов – это void SaveLevel() в классе LevelEditorController. Как понятно из названия, он отвечает за сохранение уровня в файл. Сохраняет он не весь уровень, а только ключевую информацию об объектах на нем, необходимую для его дальнейшего построения: координаты, поворот и тег.

Сохранение происходит в несколько этапов. Сначала в отдельные массивы помещаются все объекты с нужными тегами. Далее происходит заполнение указанной выше информацией об этих объектах экземпляра класса LevelInfoList. Теперь все готово к сохранению, но сначала выполняется проверка, какие уровни уже существуют, чтобы задать название вида «Level_{номер уровня}.json». После проверки, собранный массив с данными объектов сериализуется в файл формата JSON с заданным названием. Данный метод представлен в виде схемы на рисунке 11, А.

Второй важный метод – void Awake() в классе LevelBuilder, который вызывается при запуске уровня. Здесь также несколько этапов. Сначала из файла «TargetLevel.txt» считывается информация о том, какой уровень необходимо построить (в этот файл ее записывает метод void StartLevel(string _name) класса ScreenButtons). Далее данные из соответствующего файла десериализуются в массив – экземпляр класса LevelInfoList. Наконец, заполненный массив перебирается, и во время его перебора на уровне создаются объекты по указанным координатам и указанным поворотом, в зависимости от тега. Схема метода представлена на рисунке 11, В.

Рисунок 11 – Методы сохранения и сборки уровня

4.2 Файловая структура приложения

Структура файлов и директорий созданного приложения представлена на рисунке 12.

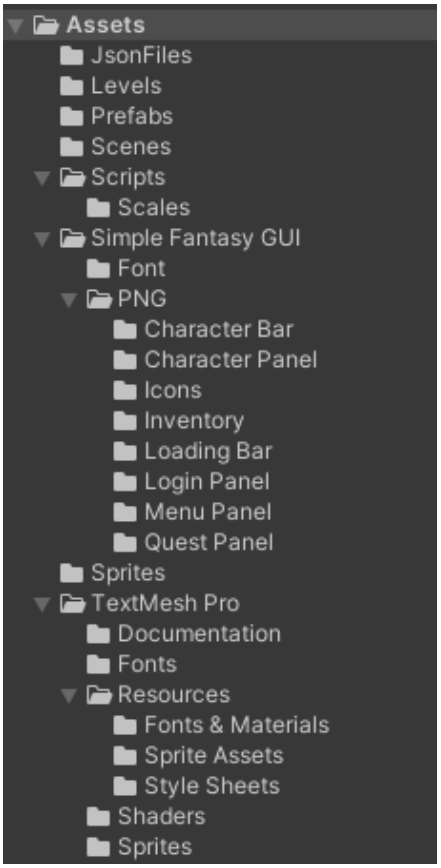

Рисунок 12 – Файловая структура

Назначение основных директорий:

– JsonFiles хранит ряд файлов в формате JSON, такие как данные пользователей, список названий уровней;

– Levels содержит файлы уровней;

– в Prefabs хранятся префабы игровых объектов, которые используются при сборке уровней;

– Scripts хранит перечисленные в таблице 4 скрипты, два из них (Resourses и Health) находятся в папке Scales;

– Simple Fantasy GUI содержит в себе одноименный набор ассетов, загруженный с сайта Unity Asset Store [25];

– Sprites содержит изображения, используемые игрой;

– TextMesh Pro хранит файлы, необходимые для работы одноименного компонента Unity Editor.

4.3 Алгоритм процедурной генерации

Как уже было сказано ранее в разделе 3.2.3, генерация уровня происходит в несколько этапов. Это можно представить в виде схемы, изображенной на рисунке 13.

Рисунок 13 – Схема процедурной генерации

Алгоритму передаются два параметра. В первую очередь это размер уровня, выраженный в единицах, равных ширине одного фрагмента стены, из которых собирается граница уровня. Ширина одного такого фрагмента составляет 2,56 (в Unity нет как таковых единиц измерения, но иногда для их обозначения используется термин «юнит»). Если в качестве размера алгоритму передать число 20, каждая сторона уровня будет состоять из 20 фрагментов, а размер уровня составит 51,2x51,2 юнитов по осям X и Y соответственно. Также алгоритм принимает сложность уровня, которая является целым числом в диапазоне от 1 до 5.

Оба этих параметра участвуют в расчетах количества игровых объектов, которое происходит после того, как будут созданы границы уровня.

Наконец, происходит расстановка объектов внутри созданных границ. Для каждого объекта вызывается метод, который генерирует случайную точку на карте, выполняя проверку, нет ли поблизости нее уже расставленных объектов.

Код процедурной генерации приведен в приложении А.

4.4 Интерфейсы

Для реализации интерфейсов и графики были использованы готовые изображения, распространяемые бесплатно. Часть из них была взята с сайта Unity Asset Store [25], еще часть с Flaticon [26]. Также некоторые изображения были созданы самостоятельно в ранее упомянутом редакторе Krita [23].

На рисунке 14 продемонстрировано окно авторизации.

Рисунок 14 – Авторизация при первом входе в игру

После авторизации открывается главное меню, которое показано на рисунке 15. Здесь можно открыть список уровней, запустить редактор уровней (рисунок 16), а также закрыть игру.

На рисунке 17 представлен интерфейс пользователя во время прохождения уровня. В нижней части экрана находится инвентарь со всеми

предметами, слева – шкала здоровья и поля для обозначения статус-эффектов над ней. Справа находится шкала загруженности рюкзака, над ней – кнопки для применения повязки, ее создания, а также для употребления 1 порции ягод в пищу. Сверху – прогресс-бар процесса добычи ресурсов.

Список соответствия внешнего вида элементов игры и их назначения приведен на рисунке 18.

Рисунок 15 – Главное меню

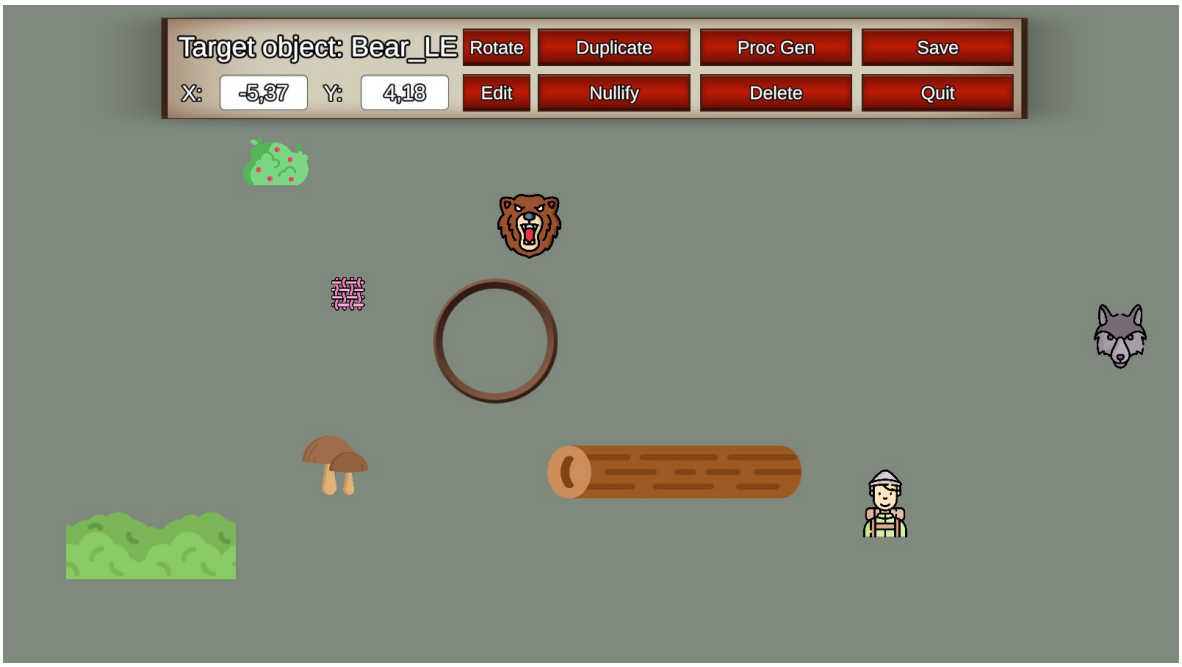

Рисунок 16 – Интерфейс редактора уровней

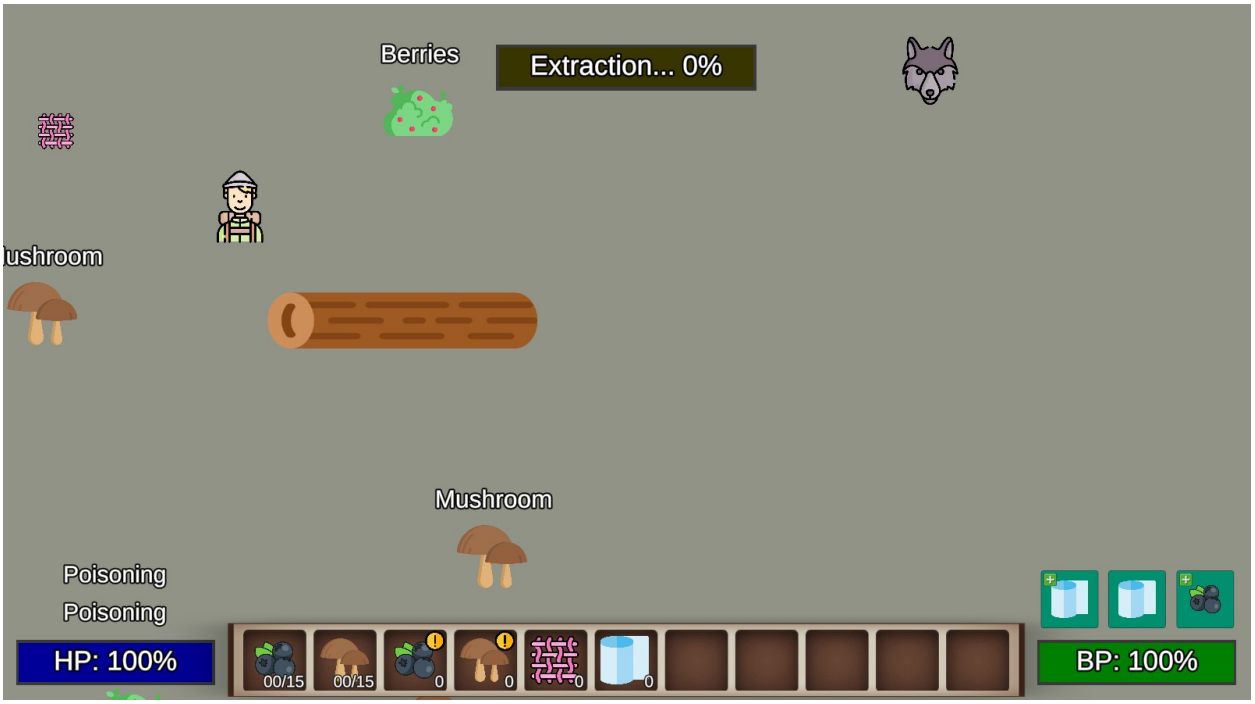

Рисунок 17 – Внутриигровой интерфейс пользователя

Рисунок 18 – Обозначения элементов игры

5 ТЕСТИРОВАНИЕ

Протестируем возможность создания нового пользователя. Для этого в окне авторизации введем логин и пароль, снимем галочку администратора. Результат представлен на рисунке 19.

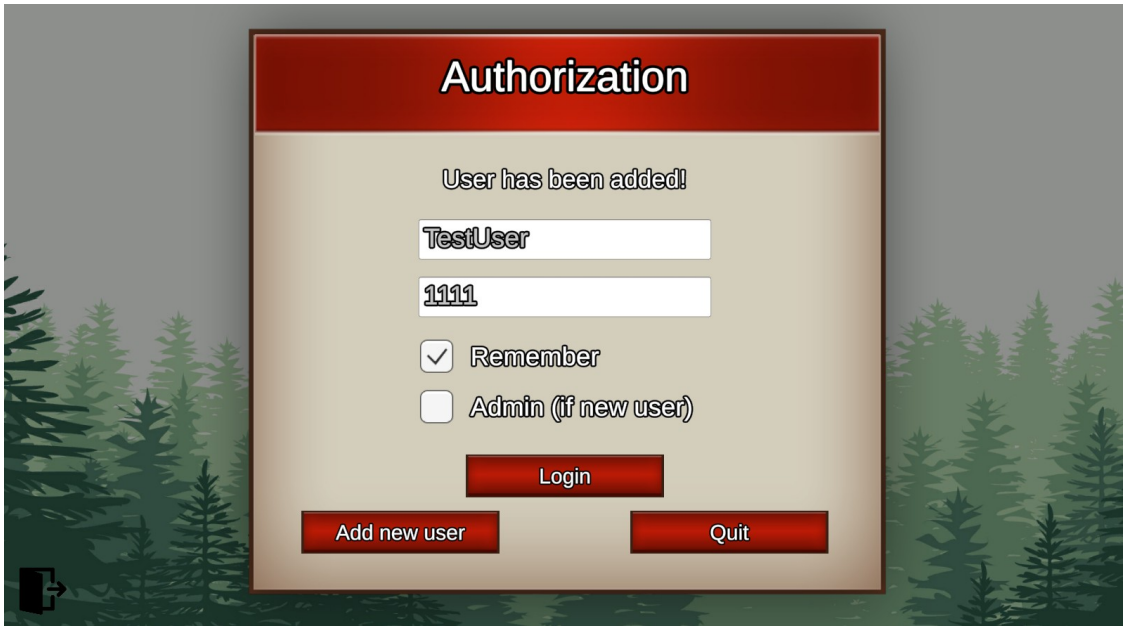

Рисунок 19 – Добавление нового пользователя

Как видим, пользователь был успешно добавлен. Теперь попробуем авторизоваться под ним, введя неправильный пароль, и проверим, обнаружит ли система ошибку. Результат эксперимента показан на рисунке 20.

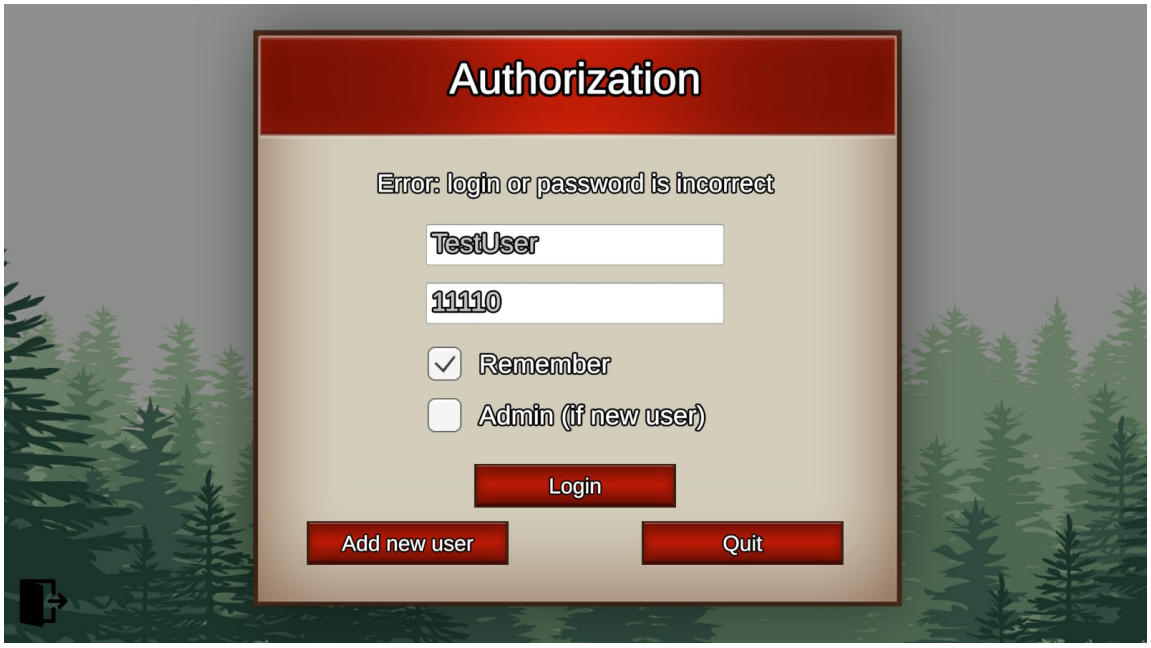

Рисунок 20 – Проверка некорректного ввода

Проверка была успешно пройдена, некорректный ввод распознан.

Проверим также работу процедурной генерации и редактора уровней. Запустим редактор уровней, нажмем кнопку для создания уровня автоматически, введем параметры и посмотрим на результат, который представлен на рисунках 21-23. Скриншоты для наглядности сделаны в представлении сцены.

После генерации объекты успешно перемещаются вручную, а по нажатию кнопки «Save» уровень сохраняется в файл.

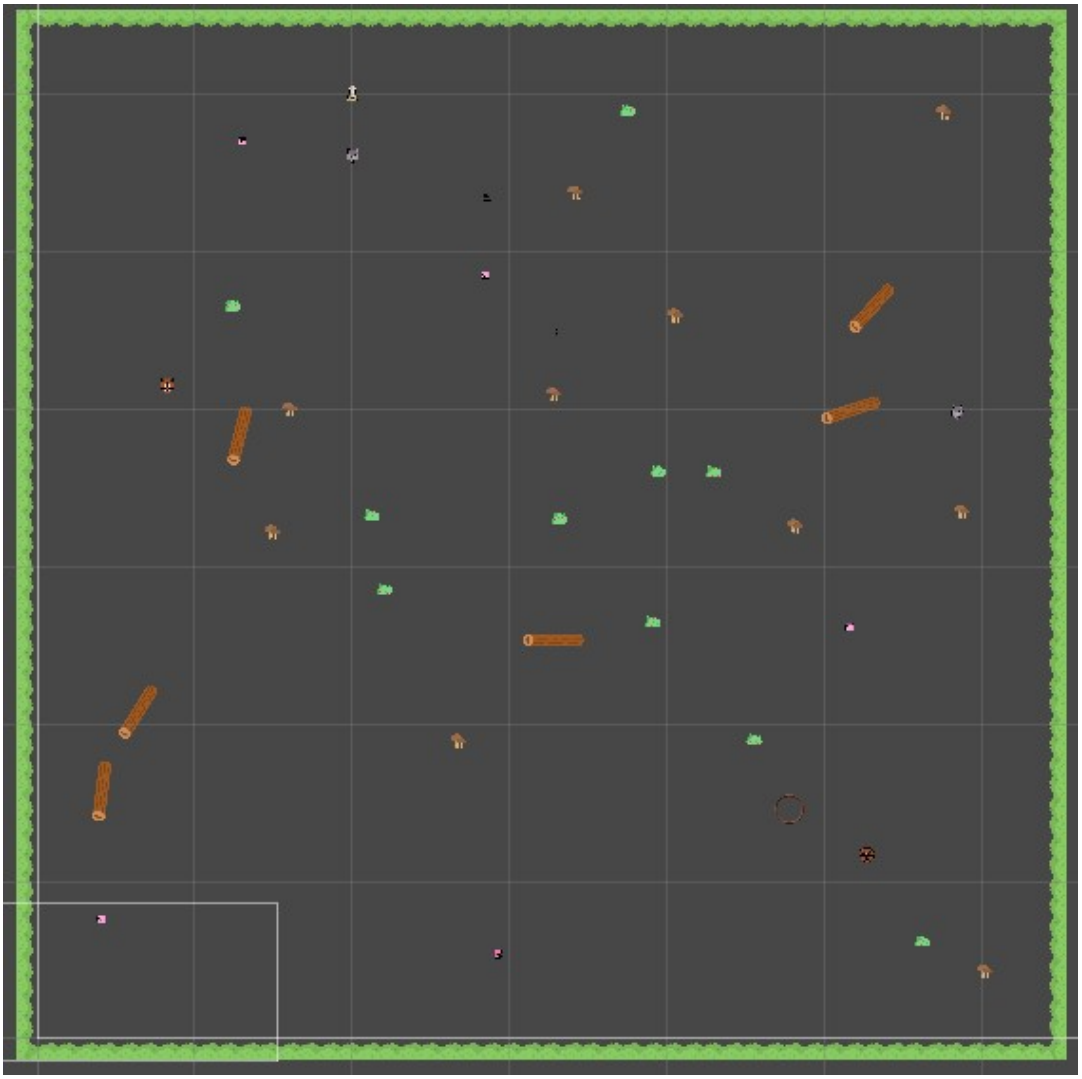

Рисунок 21 – Локация размером 25, сложность 3

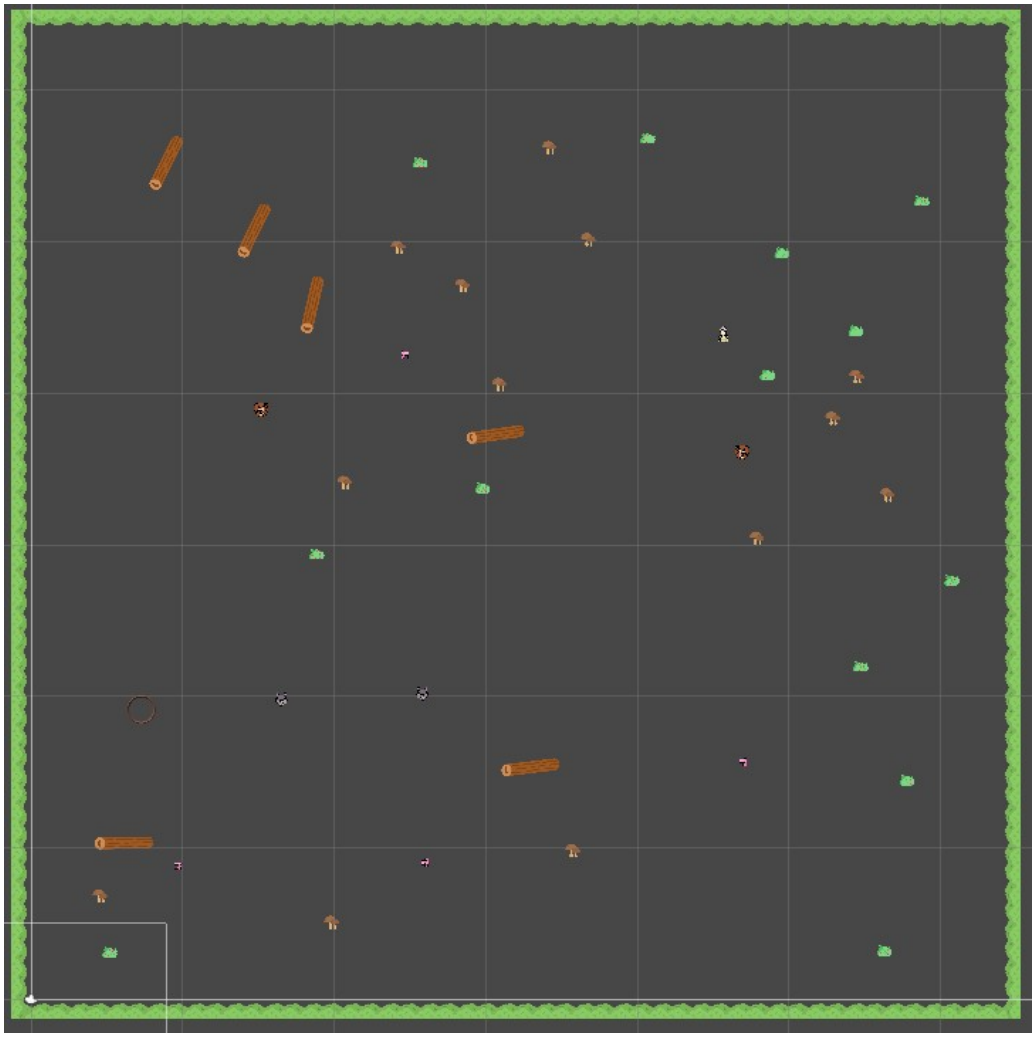

Рисунок 22 – Локация размером 25, сложность 5

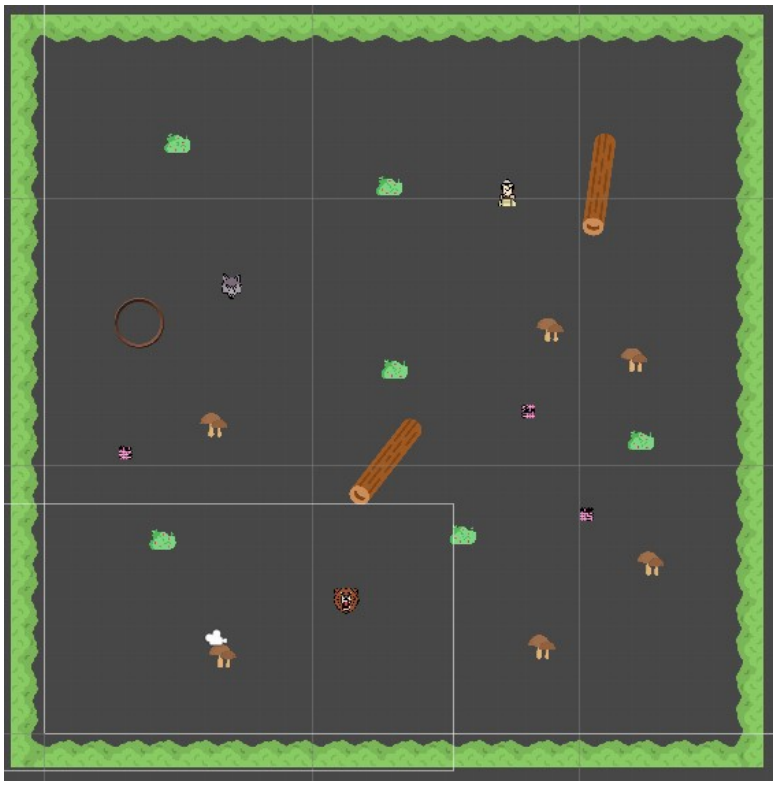

Рисунок 23 – Локация размером 10, сложность 3

Также были проверены на корректность работы игровые механики. Все работает нормально: ресурсы добываются (рисунок 24), агрессивные NPC наносят урон с некоторым шансом на появление кровотечения, которое успешно устраняет примененная перевязка, созданная из трех соответствующих частей. Так, на рисунке 25 на игрока наложено сразу два статус-эффекта – кровотечение после столкновения с медведем и отравления от съеденных ядовитых ягод. У игрока есть возможность восстановить здоровье ягодами и наложить уже имеющуюся повязку для остановки кровотечения.

Уровень успешно завершается при наборе нужного числа ресурсов и касания игроком финиша, статистика подсчитывается и отображается (рисунок 26).

Рисунок 24 – Процесс добычи ресурсов

Рисунок 25 – Демонстрация работы механик игры

Рисунок 26 – Статистика в конце уровня

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы была спроектирована и разработана компьютерная игра с процедурной генерацией и встроенным редактором уровней. Для того, чтобы данная цель была достигнута, были выполнены следующие задачи:

– выделены существующие на данный момент решения с похожим функционалом, рассмотрены их достоинства и недостатки;

– на основании проведённого анализа осуществлена постановка задачи и сформировано техническое задание;

– проведено сравнение доступных методов и средств реализации, выбранные по его результатам средства использовались для разработки;

– разработан эскизный проект игры, продумана игровая логика и механики;

– реализована сама игра;

– проведено тестирование реализованной игры для проверки её работоспособности, которое не выявило неправильной или нестабильной работы получившегося продукта.

Также возможно дальнейшее развитие игры. Среди возможных вариантов развития:

– улучшение графики игры и интерфейсов;

– добавление анимаций и эффектов для объектов;

– добавление новых игровых механик;

– создание звукового сопровождения;

– доработка системы авторизации для большей безопасности;

– перевод интерфейса на другие языки.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Компьютерные и видеоигры (российский рынок). – Текст. Изображение : электронные // TAdviser – портал выбора технологий и поставщиков : [сайт]. – 21 мая 2022 – URL: https://www.tadviser.ru/index.php/Статья:Компьютерные_и_видеоигры_(российский_рынок) (дата обращения: 30.05.2022).

2. Построение трассы в редакторе карт Trackmania (простой редактор). – Текст. Изображение : электронные // Ubisoft : [официальный сайт]. – URL: https://www.ubisoft.com/ru-ru/help/trackmania/gameplay/article/building-a-track-in-the-trackmania-map-editor-simple-editor/000068295 (дата обращения: 30.05.2022).

3. TrackMania. – Текст. Изображение : электронные // Википедия – свободная энциклопедия : [сайт]. – 22 апреля 2022. – URL: https://ru.wikipedia.org/wiki/TrackMania (дата обращения: 30.05.2022).

4. Heroes of Might and Magic (серия игр). – Текст. Изображение : электронные // Википедия – свободная энциклопедия : [сайт]. – 14 марта 2022. – URL: https://ru.wikipedia.org/wiki/Heroes_of_Might_and_Magic_(серия_игр) (дата обращения: 30.05.2022).

5. Might and Magic Heroes 7. Герои 7. Редактор и генератор карт. – Изображение (подвижное ; двухмерное) : электронное // YouTube : [сайт]. – 5 октября 2015. – URL: https://www.youtube.com/watch?v=Lfhxuh1WCOo (дата обращения: 30.05.2022).

6. Besiege. – Текст. Изображение : электронные // Википедия – свободная энциклопедия : [сайт]. – 22 апреля 2022. – URL: https://ru.wikipedia.org/wiki/Besiege (дата обращения: 30.05.2022).

7. Besiege добавили редактор уровней и мультиплеер. – Текст. Изображение : электронные // ViVA la Cloud : [сайт]. – 9 декабря 2017. – URL: https://vivalacloud.ru/2017/12/besiege-multiplayer/ (дата обращения: 30.05.2022).

8. Geometry Dash. – Текст. Изображение : электронные // Википедия – свободная энциклопедия : [сайт]. – 22 апреля 2022. – URL: https://ru.wikipedia.org/wiki/Geometry_Dash (дата обращения: 30.05.2022).

9. Level Editor. – Текст. Изображение : электронные // Fandom : [сайт]. – URL: https://geometry-dash.fandom.com/wiki/Level_Editor (дата обращения: 30.05.2022).

10. Portal 2. – Текст. Изображение : электронные // Википедия – свободная энциклопедия : [сайт]. – 16 января 2022. – URL: https://ru.wikipedia.org/wiki/Portal_2 (дата обращения: 30.05.2022).

11. Puzzle Creator. – Текст. Изображение : электронные // Portal Wiki : [сайт]. – 28 марта 2022. – URL: https://theportalwiki.com/wiki/Puzzle_Creator (дата обращения: 30.05.2022).

12. Самый популярный движок в Steam – Unity. Но на Unreal Engine делают самые дорогие игры. – Текст. Изображение : электронные // App2Top.ru: сайт для разработчиков, издателей и маркетологов игр : [сайт]. – 03 сентября 2021. – URL: https://app2top.ru/analytics/samyj-populyarnyj-dvizhok-v-steam-unity-no-na-unreal-engine-delayut-samye-dorogie-igry-189951.html (дата обращения: 30.05.2022).

13. Путеводитель по геймдеву. Не Unity едины. Большой обзор игровых движков для начинающих и профи. – Текст. Изображение : электронные // DTF – игры, кино, сериалы, разработка, сообщество : [сайт]. – 02 декабря 2021. – URL: https://dtf.ru/indie/966434-putevoditel-po-geymdevu-ne-unity-ediny-bolshoy-obzor-igrovyh-dvizhkov-dlya-nachinayushchih-i-profi (дата обращения: 30.05.2022).

14. 11 популярных игровых движков в 2021 году. – Текст. Изображение : электронные // BlackCaviarGames : [сайт]. – 23 марта 2021. – URL: https://blackcaviar.games/obzor_igrovyh_dvizhkov (дата обращения: 30.05.2022).

15. Unity (игровой движок). – Текст. Изображение : электронные // Википедия – свободная энциклопедия : [сайт]. – 26 мая 2022. – URL: https://ru.wikipedia.org/wiki/Unity_(игровой_движок) (дата обращения:

30.05.2022).

16. Unity Visual Scripting. – Текст. Изображение : электронные // Unity Real-Time Platform : [сайт]. – URL: https://unity.com/features/unity-visual-scripting (дата обращения: 30.05.2022).

17. Unreal Engine. – Текст. Изображение : электронные // Википедия – свободная энциклопедия : [сайт]. – 22 апреля 2022. – URL: https://ru.wikipedia.org/wiki/Unreal_Engine (дата обращения: 30.05.2022).

18. Туториал по Unreal Engine. Часть 2: Blueprints. – Текст. Изображение : электронные // Хабр : [сайт]. – 22 апреля 2022. – URL: https://habr.com/ru/post/344446/ (дата обращения: 30.05.2022).

19. GDevelop. – Текст. Изображение : электронные // Wikipedia, the free encyclopedia : [сайт]. – 27 мая 2022. – URL: https://en.wikipedia.org/wiki/GDevelop (дата обращения: 30.05.2022).

20. Stride (game engine). – Текст. Изображение : электронные // Wikipedia, the free encyclopedia : [сайт]. – 27 марта 2022. – URL: https://en.wikipedia.org/wiki/Stride_(game_engine) (дата обращения: 30.05.2022).

21. CryEngine. – Текст. Изображение : электронные // Википедия – свободная энциклопедия : [сайт]. – 6 апреля 2022. – URL: https://ru.wikipedia.org/wiki/CryEngine (дата обращения: 30.05.2022).

22. Visual Studio Code или Visual Studio? Что выбрать, в чем отличия. – Текст. Изображение : электронные // Программирование от простого к сложному ALEKSEEV74.RU : [сайт]. – 21 февраля 2020. – URL: https://alekseev74.ru/lessons/show/visual-studio/code-vs-ide (дата обращения: 30.05.2022).

23. Krita. – Текст. Изображение : электронные // Википедия – свободная энциклопедия : [сайт]. – 12 мая 2022. – URL: https://ru.wikipedia.org/wiki/Krita (дата обращения: 30.05.2022).

24. Ramus. – Текст. Изображение : электронные // Словари и энциклопедии на Академике : [сайт]. – URL: https://dic.academic.ru/dic.nsf/ruwiki/608820 (дата обращения: 30.05.2022).

25. SIMPLE FANTASY GUI. – Текст. Изображение : электронные // Unity Asset Store - The Best Assets for Game Making : [сайт]. – 28 сентября 2018. – URL: https://assetstore.unity.com/packages/2d/gui/simple-fantasy-gui-99451 (дата обращения: 30.05.2022).

26. Flaticon. – Текст. Изображение : электронные // Векторные иконки и стикеры – PNG, SVG, EPS, PSD и CSS : [сайт]. – URL: https://www.flaticon.com/ru/ (дата обращения: 30.05.2022).

ПРИЛОЖЕНИЕ А
Код процедурной генерации

```
public class ProcGeneration
{
    public int LevelSize = 25;
    public int LevelDifficulty = 5;
    public LevelInfoList LIList;
    public void Generation()
    {
        LIList = new LevelInfoList();
        if (LevelSize < 10) LevelSize = 10;
        if (LevelSize > 200) LevelSize = 200;
        Vector3 EdgeWallGenPos = new Vector3(0, 0, 0);
        Quaternion EdgeWallGenRotation = new Quaternion(0,0,0,1);
        Quaternion rotationZ90 = Quaternion.AngleAxis(90,Vector3.forward);
        Quaternion RandomRotation;
        int quantityOfResources;
        int quantityOfPoisonousResources;
        int quantityOfWolves = 0;
        int quantityOfBears = 0;
        int quantityOfDressingParts;
        for (int j = 0; j < 4; j++)
        {
            for (int i = 0; i <= LevelSize; i++)
            {
                LIList.LevelObjects.Add(new LevelInfo(EdgeWallGenPos, EdgeWallGenRotation, "EdgeWall"));
                if (i != LevelSize)
                {
                    switch (j)
                    {
                        case 0:
                            EdgeWallGenPos = new Vector3(EdgeWallGenPos.x + 2.56f, EdgeWallGenPos.y, EdgeWallGenPos.z);
                            break;
                        case 1:
                            EdgeWallGenPos = new Vector3(EdgeWallGenPos.x, EdgeWallGenPos.y + 2.56f, EdgeWallGenPos.z);
                            break;
                        case 2:
                            EdgeWallGenPos = new Vector3(EdgeWallGenPos.x - 2.56f, EdgeWallGenPos.y, EdgeWallGenPos.z);
                            break;
                        case 3:
                            EdgeWallGenPos = new Vector3(EdgeWallGenPos.x, EdgeWallGenPos.y - 2.56f, EdgeWallGenPos.z);
                            break;
                    }
                }
            }
            EdgeWallGenRotation *= rotationZ90;
        }
        for (int i = 0; i < (LevelSize/4); i++)
        {
            RandomRotation = Quaternion.AngleAxis(Random.Range(0, 90), Vector3.forward);
            LIList.LevelObjects.Add(new LevelInfo(GenRandomPoint(""),RandomRotation, "log"));
        }
        quantityOfResources = LevelSize / 5;
        if (LevelDifficulty < 1) LevelDifficulty = 1;
        if (LevelDifficulty > 5) LevelDifficulty = 5;
        if (quantityOfResources < 3) quantityOfResources = 3;
        quantityOfPoisonousResources = Convert.ToInt32(Math.Ceiling(quantityOfResources * LevelDifficulty * 0.3));
        quantityOfDressingParts = 3;
        switch (LevelDifficulty)
        {
```

```
            case 1:
                quantityOfDressingParts = Convert.ToInt32(LevelSize * 0.6);
                break;
            case 2:
                quantityOfDressingParts = Convert.ToInt32(LevelSize * 0.45);
                break;
            case 3:
                quantityOfDressingParts = Convert.ToInt32(LevelSize * 0.3);
                break;
            case 4:
                quantityOfDressingParts = Convert.ToInt32(LevelSize * 0.225);
                break;
            case 5:
                quantityOfDressingParts = Convert.ToInt32(LevelSize * 0.15);
                break;
        }
        if (quantityOfDressingParts < 3) quantityOfDressingParts = 3;
        System.Random rnd = new System.Random();
        quantityOfWolves = LevelSize/10 * rnd.Next(0, LevelDifficulty);
        quantityOfBears = (LevelSize / 10 * LevelDifficulty) - quantityOfWolves;

        for (int i = 0; i < quantityOfResources; i++)
        {
            LIList.LevelObjects.Add(new LevelInfo(GenRandomPoint("Resources"), Quaternion.identity, "Mushrooms"));
            LIList.LevelObjects.Add(new LevelInfo(GenRandomPoint("Resources"), Quaternion.identity, "Berries"));
        }
        for (int i = 0; i < quantityOfPoisonousResources; i++)
        {
            LIList.LevelObjects.Add(new LevelInfo(GenRandomPoint("Resources"), Quaternion.identity, "MushroomsPoisonous"));
            LIList.LevelObjects.Add(new LevelInfo(GenRandomPoint("Resources"), Quaternion.identity, "BerriesPoisonous"));
        }
        for (int i = 0; i < quantityOfDressingParts; i++)
            LIList.LevelObjects.Add(new LevelInfo(GenRandomPoint(""), Quaternion.identity, "dressingPart"));
        for (int i = 0; i < quantityOfWolves; i++)
            LIList.LevelObjects.Add(new LevelInfo(GenRandomPoint("NPC"), Quaternion.identity, "Wolf"));
        for (int i = 0; i < quantityOfWolves; i++)
            LIList.LevelObjects.Add(new LevelInfo(GenRandomPoint("NPC"), Quaternion.identity, "Bear"));
        LIList.LevelObjects.Add(new LevelInfo(GenRandomPoint(""), Quaternion.identity, "Player"));
        LIList.LevelObjects.Add(new LevelInfo(GenRandomPoint(""), Quaternion.identity, "Finish"));
    }
    public Vector3 GenRandomPoint(string type)
    {
        bool isOK = false;
        bool IsIntersection;
        Vector3 point = new Vector3();
        float minDistance;
        if (type == "NPC") minDistance = 5f;
        else minDistance = 3f;
        while (!isOK)
        {
            point = new Vector3(Random.Range(2f, (float)(LevelSize*2.56 - 2)), Random.Range(2f, (float)(LevelSize*2.56 - 2)), 0);
            IsIntersection = false;
            foreach (var c in LIList.LevelObjects)
                if (Vector2.Distance(c.pos, point) < minDistance)
                {
                    IsIntersection = true;
                    break;
                }
            if (!IsIntersection) isOK = true;
        }
        return point;
    }
}
```