Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Российский государственный профессионально-педагогический университет»

2D-ИГРА В ЖАНРЕ ROGUELIKE

Выпускная квалификационная работа
по направлению подготовки 09.03.02 Информационные системы
и технологии
профилю подготовки «Информационные технологии в медиаиндустрии»

Идентификационный номер ВКР: 153

Екатеринбург 2018

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Российский государственный профессионально-педагогический университет»
Институт инженерно-педагогического образования
Кафедра информационных систем и технологий

К ЗАЩИТЕ ДОПУСКАЮ
Заведующая кафедрой ИС
_____Н. С. Толстова
«_ _»_____ _____2018г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

2D-ИГРА В ЖАНРЕ ROGUELIKE

Исполнитель:
обучающийся группы ИТм-402 В. В. Демин

Руководитель:
ст. преподаватель И. А. Садчиков

Нормоконтролер: Н. В. Хохлова

Екатеринбург 2018

АННОТАЦИЯ

Выпускная квалификационная работа состоит из 2D игры в жанре roguelike и пояснительной записки на 67 страницах, содержащей 38 рисунков, 1 таблицу, 30 источников литературы, а также 4 приложения на 9 страницах.

Ключевые слова: ИГРА, КОМПЬЮТЕРНАЯ ИГРА, РАЗРАБОТКА КОМПЬЮТЕРНЫХ ИГР.

Демин В. В. 2D-игра в жанре roguelike: выпускная квалификационная работа / В. В. Демин ; Рос. гос. проф.-пед. ун-т, Ин-т инж.-пед. образования, Каф. информ. систем и технологий. — Екатеринбург, 2018. — 67 с.

В работе рассмотрена разработка компьютерной игры.

Целью работы является разработка компьютерной игры в жанре roguelike с помощью среды разработки компьютерных игр Unity.

Для достижения цели были проанализированы различные представители жанра roguelike и несколько сред разработки компьютерных игр, из которых была выбрана одна — Unity 2018. Функционал среды разработки также был изучен.

В ходе были созданы графические материалы, использующиеся в игре, и написаны скрипты, необходимые для работы игры.

С использованием данных материалов была разработана компьютерная игра в жанре roguelike на движке Unity.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность темы дипломной работы.

Во второй половине двадцатого века технический прогресс стремительно начал набирать свои обороты. Начали появляться все более мощные вычислительные устройства. С тех пор прошло много времени и компьютер для нас уже не кажется чем-то удивительным, он есть, практически, в каждом доме и воспринимается как совершенно обыденная вещь. Вместе с развитием компьютеров стали появляться и компьютерные игры.

В наши дни компьютерные игры является одним из наиболее распространенных видов развлечения. Начав свою историю с аркадных автоматов, в наши дни игры охватили практически все цифровые устройства: компьютеры, телефоны и планшеты и имеют широкую и разнообразную аудиторию.

Немалая часть компьютерных игр имеют первоисточники в виде настольных игр. Компьютерные игры имеют некоторые преимущества, по сравнению с настольными. Например, для их хранения не нужно много места, нежели для настольных. К тому же, части настольных игр (карточки или фишка, например) очень легко потерять.

В рамках дипломной работы планируется создать игру в жанре roguelike с помощью одной из сред разработки компьютерных игр.

Объект исследования: компьютерные игры.

Предмет исследования: средства и технологии разработки компьютерных игр.

Цель дипломной работы: разработать компьютерную игры в жанре roguelike с помощью одной из сред разработки компьютерных игр.

Задачи дипломной работы:

- проанализировать жанр roguelike и его представителей и выявить особенности жанра, на основании которых определить функциональные требования;
- проанализировать различные средства разработки компьютерных игр и изучить их функционал;
- разработать дизайн существ и окружения и отрисовать в графическом редакторе;
- разработать прототип игры;
- протестировать разработанный продукт на предмет ошибок.

1 АНАЛИТИЧЕСКАЯ ЧАСТЬ

1.1 Анализ и общая характеристика предметной области

Игра — свободная деятельность, являющаяся формой самовыражения субъекта и направленная на удовлетворение потребностей в развлечении, удовольствии, снятии напряжений, а также на развитие определенных навыков и умений.

Компьютерная игра — игра, являющаяся компьютерной программой и использующая мультимедийные возможности компьютера.

2D игра — компьютерная игра, визуальное пространство которой состоит из двумерных объектов.

Жанр — род произведений в пределах какого-либо искусства, отличающийся особыми, только ему свойственными сюжетными, стилистическими признаками.

Сеттинг — время и место, в которых происходят события какого-либо произведения (книги, фильма, игры).

Игровая механика — правила и ограничения, в рамках которых функционирует игра. Сюда можно отнести то, как и какие действия могут совершать игроки и как игроки могут взаимодействовать с игрой и друг с другом.

Разработка компьютерной игры — процесс создания компьютерной игры. Данный процесс включает в себя следующее: выбор жанра и сеттинга, написание сюжета, выбор языка программирования и игрового движка, разработка текстур и моделей и прочее. Разработкой может заниматься как профессиональные студии разработчиков, так и группы инди-разработчиков, от этого, как правило, зависит бюджет игры, количество разработчиков, размер игры и количество контента в ней.

Инди-разработчик — один человек или группа людей, разрабатывающих компьютерные игры независимо от финансовой поддержки крупных иг-

ровых издательств. Данные игры чаще всего небольшие и простые по сравнению с играми профессиональных студий. Инди-игры чаще всего распространяются бесплатно или за небольшую цену через интернет или сервисы цифровой дистрибуции. Тем не менее, некоторые инди-игры довольно таки известны — игра Minecraft (рисунок 1) входит в книгу рекордов Гиннеса, как самая продаваемая инди-игра.

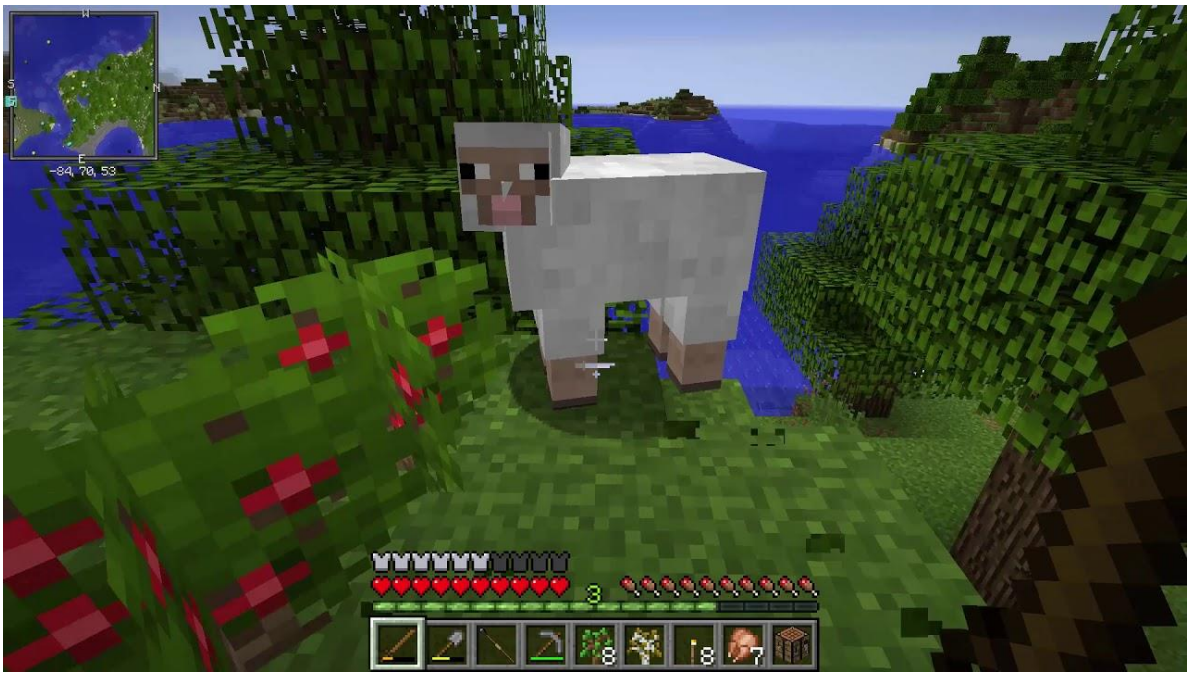

Рисунок 1 — Скриншот игры Minecraft

RPG — Role-Playing Game, один из жанров компьютерных игр. Основой игрового процесса является отыгрывание определенной роли. Игрок управляет определенным героем, имеющим свои характеристики, умения и навыки, которые могут меняться в течение игры. Данному жанру, как правило, присущи продуманный сюжет и мир, сюжет обычно подается через большое количество диалогов и заданий.

Тайловая графика — способ построения игровой графики, из отдельных изображений (тайлов), как правило прямоугольной формы (рисунок 2). Положительной чертой данного способа построения графики является то, что использовать набор тайлов для построения изображения намного проще, чем полностью рисовать изображение. Главным минусом тайловой графики является ее монотонность — легко заметить повторяющиеся элементы.

Проблема монотонности решается различными способами — например, создается несколько копий одного тайла с малозначительными изменениями.

При перемешивании данных тайлов в глаза не бросается их схожесть и создает иллюзию разнообразия. Кроме того, монотонность можно «обосновать» клеточной механикой игры — каждый тайл означает что-либо (например, тайл с кирпичами является непроходимой стеной, а тайл с потрескавшимися кирпичами — разрушаемую стену).

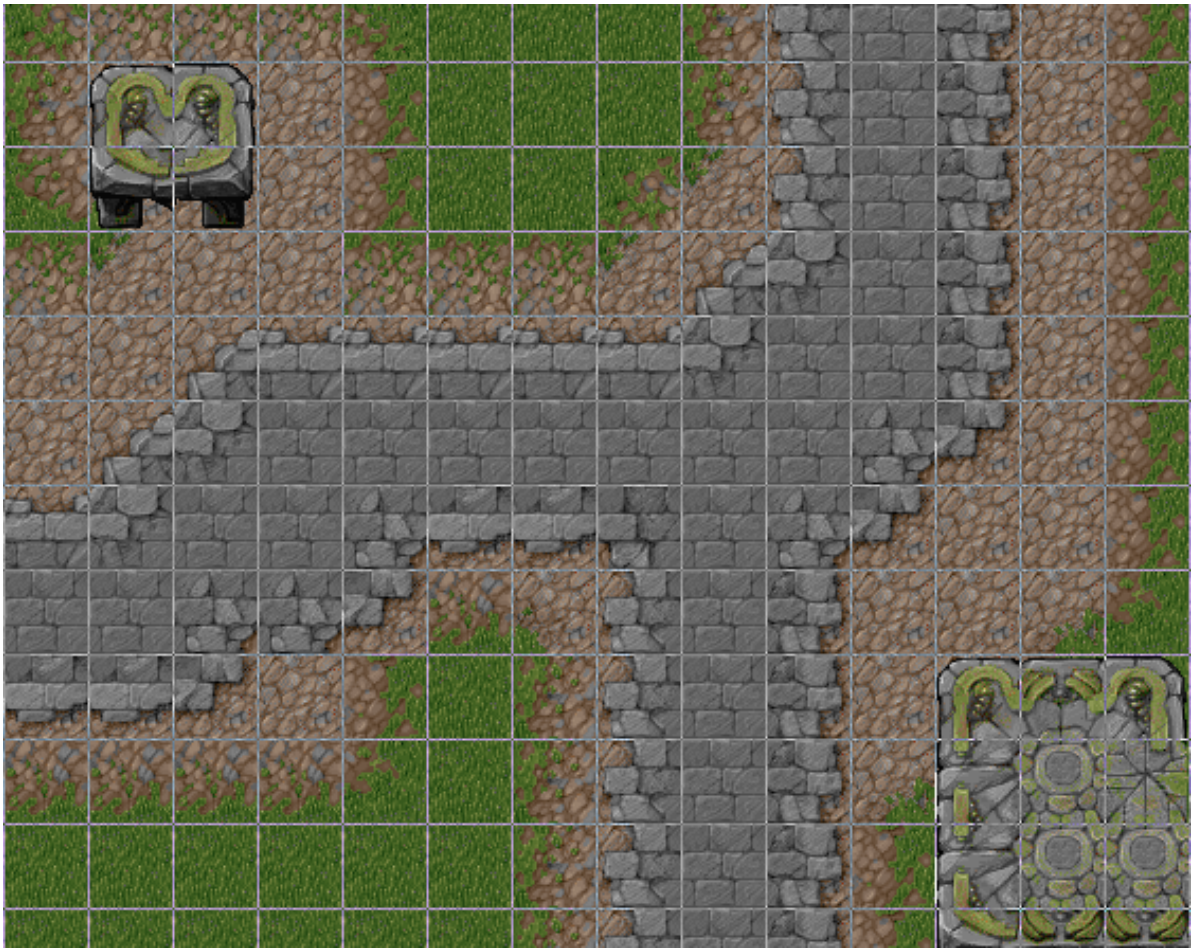

Рисунок 2 — Пример использования тайловой графики

ASCII — «Американский Стандартный Код для информационного Обмена» (American Standard Code for Information Interchange). Был разработан в начале 60-х годов 20-го века как стандартная кодировка для компьютеров и аппаратных устройств. Данная кодировка использовалась как для передачи данных, так и для создания изображений (рисунок 3) из символов в так называемой ASCII-графике (или «псевдографике»), что помогало создавать псев-

дографические изображения для отображения их там, где невозможно отобразить обычную графику.

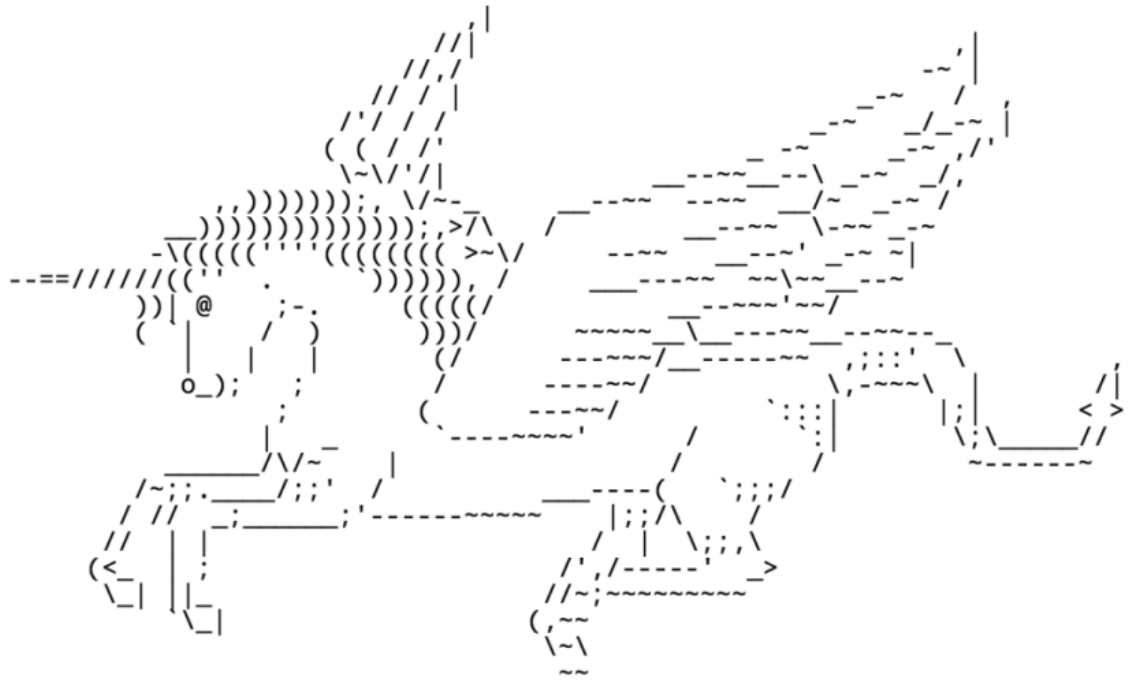

Рисунок 3 — Пример псевдографики

1.2 Анализ существующих разработок

Жанр Roguelike — жанр компьютерных игр, является поджанром ролевых пошаговых игр. Данный жанр имеет некоторые отличительные особенности, среди которых можно выделить следующие:

- случайная генерация игрового окружения (локация, враги, бонусы и прочее), которая уникальна для каждого нового прохождения;
- пошаговость действий, каждая команда должна быть равна одному ходу или одному действию;
- необратимость действий, любое действие может привести к фатальной ошибке, вплоть до окончательной гибели персонажа и невозможности впоследствии продолжить игру;

- полная доступность всех игровых действий с самого начала игры, то есть у игрока изначально есть весь доступный функционал и он может им пользоваться;
- самостоятельное исследование окружающего мира, игрок должен сам исследовать и разобраться в работе всех игровых механик;
- игрок имеет полную свободу действий, в игре не предусмотрено линейного прохождения.

Жанр roguelike назван в честь игры-прародителя, Rogue и означает «rogue-подобные». Игра Rogue вышла в 1980 году на операционную систему Unix, а впоследствии была портирована на множество других популярных на то время систем, среди которых были IBM PC, Amstrad CPC, ZX Spectrum, Commodore 64 и другие.

Суть игры заключается в исследовании подземелий. Локация игры полностью отображалась на экране и представляла собой несколько небольших комнат (общим количеством не более 9). Целью игры является нахождение сокровища — амулета Йендора (Amulet of Yendor), ставший впоследствии одной из самых распространённых конечных целей игр данного жанра. Амулет спрятан на самом нижнем этаже подземелий. По мере прохождения игроку встречаются различные монстры и ловушки.

Внешне игра является псевдографической (рисунок 4), каждый символ или буква обозначает тот или иной объект (например, враг соответствует заглавной буквы своего имени, т.е. B — Bat, Z — Zombie и т.д.), значение всех символов было описано в инструкции, шедшей в комплекте с игрой.

Данная игра очень сильно повлияла на игровую индустрию, породив собой новый жанр. К числу наиболее известных продолжателей жанра принадлежат такие игры как NetHack, Ancient Domians of Mystery, Moria и другие. Как правило, большинство классических roguelike игр используют псевдографику, но некоторые используют тайловую графику.

Рисунок 4 — Скриншот игры Rogue

NetHack — одна из наиболее известных roglike игр. Вышла в 1987 году. «Net» означает не многопользовательский режим, как может показаться на первый взгляд, а то, что, во-первых, при разработке игры команда разработчиков общается по сети, а во-вторых, успешное прохождение игры подразумевает обсуждение игры с другими игроками, обычно происходящее в интернете. «Hack» в названии отсылает к одному из ответвлений жанра ролевых игр — hack and slash (уничтожение большого количества врагов в ближнем бою) — что неплохо описывает игровой процесс.

NetHack является старейшей из игр, которая до сих пор поддерживается и развивается разработчиками (последнее обновление игры вышло весной 2018 года). Благодаря открытому исходному данная игра была портирована на огромное количество платформ (Linux, FreeBSD, MacOS, iOS, Android и другие). Игра выполнена с использованием псевдографической графики (рисунок 5), однако некоторые порты имеют тайловую графику.

Рисунок 5 — Скриншот игры NetHack

Перед началом игрок может выбрать для своего персонажа расу, пол, мировоззрение и роль (роли есть как классические, вроде мага или воина, так и необычные, например, археолог). От выбора роли и мировоззрения будет зависеть то, в какое божество верит персонаж, и как к нему будут относиться различные существа на протяжении игры.

Целью игры является поиск амулета Йендора для последующей его передачи своему божеству, что позволит получить герою статус полубога.

Встречается большое разнообразие существ, причем как враждебных, так и дружелюбных (игрок может начать игру со своим домашним животным — собакой или кошкой). Существа обозначены буквами английского алфавита (как и в Rogue) и некоторыми символами.

В игре насчитывается около 50 уровней, разделенных на несколько зон (к тому же, существует еще около 30 уровней различных ответвлений, необязательных для прохождения).

Игровой процесс, в целом, схож с игрой Rogue. Здесь у игрока есть возможность пройти через подземелья и замки, побывать в местном аналоге ада, через который, путем проведения ритуала, можно попасть в святилище и добыть у главного священника искомый амулет. После получения амулета открывается несколько уровней, где на последнем из них предстоит сражение с тремя всадниками Апокалипсиса — Голодом, Чумой и Смертью. Войной —

четвертым всадником — считается сам игрок. После этого можно пожертвовать амулет своему божеству и получить бессмертие, тем самым пройдя игру.

Игра до сих пор некоторыми игроками считается одной из самых интересных и проработанных за всю историю видеоигр.

Вследствие расцвета разработки инди-игр и развития жанра roguelike появился поджанр roguelike-like игр, т.е. «подобные Rogue-подобным». Игры жанра roguelike-like сохраняют основные особенности жанра roguelike (например, процедурную генерацию и необратимую смерть персонажа), но при этом не имеют пошаговости и включают в себя элементы других жанров.

Был проведен анализ нескольких игр данного жанра разных годов выпуска.

Pixel Dungeon — кроссплатформенная игра, выпущенная в 2013 году. Данная игра (рисунок 6) представляет собой классический рогалик, соответствующий всем канонам родоначальника жанра. В данной игре 25 уровней, каждый пятый из которых — битва с боссом. Конечной целью игры является поиск амулета Йендора, как в оригинальном Rogue.

Игрок перед началом игры может выбрать один из четырех классов персонажа — воин, маг, охотник и разбойница (отличаются начальными параметрами и стартовой экипировкой).

В игре всего 3 параметра героя — здоровье, сила (влияющая на эффективность экипировки) и голод.

На каждом уровне герой должен найти переход на следующий уровень. При исследовании локации герой может встретить врагов (в игре 25 видов врагов), после смерти, которые могут дать различные предметы.

Среди предметов есть зелья и свитки, эффект которых можно узнать только после применения, одежда (доспехи и кольца) оружие ближнего боя, метательное оружие, волшебные палочки, еда (необходима для утоления голода персонажа, иначе персонаж при перемещении будет не восстанавливать

здоровье, а получать урон) и золото. Один из свитков позволяет накладывать на вещи чары, дающие какие-либо бонусы.

Рисунок 6 — Скриншот игры Pixel Dungeon

При убийстве врагов персонаж получает опыт, повышающий уровень персонажа (что дает больший запас здоровья и очко силы).

Иногда на уровне встречаются магазины, где герой может потратить добытое золото на различные предметы.

Сама игра имеет простое и понятное управление, но при этом высокую игровую сложность. В игре имеется большое количество достижений.

DRL (ранее DoomRL) — кроссплатформенная игра 2013 года выпуска. Игра выполнена в сеттинге вселенной игр Doom (рисунок 7) и много заимствует именно оттуда, например, врагов и вооружение.

При запуске игры доступно несколько уровней сложности, затем игроку предлагается 3 класса персонажа на выбор, отличающихся характеристиками.

Игра состоит из 25 уровней (без учета секретных уровней). По сюжету, персонаж игрока это последний выживший из отряда космического спецназа, который был отправлен проверить сигнал бедствия с завода на планете Фобос, спутнике Марса (там же разворачивались действия в оригинальной игре).

Рисунок 7 — Скриншот из игры Doom

Отличием от оригинального Doom является то, что в DRL (рисунок 8) боевая система имеет большое тактическое значение, в то время как Doom является очень динамичным шутером.

Одна игровая сессия (полное прохождение) DRL не очень долгая, относительно многих игр данного жанра. Игра имеет простое и понятное управление.

Рисунок 8 — Скриншот игры DRL

Powder (рисунок 9) — идейный продолжатель NetHack. Разработчик игры хотел сделать игру, похожую на NetHack, но при этом для системы GameBoy Advance. Выбор платформы и ее портативность сыграли большую роль в истории развития и популярности игры.

Рисунок 9 — Скриншот игры Powder

Так как игра разрабатывалась для платформы GameBoy Advance (рисунок 10), можно отметить, что в игре крайне простое и понятное управление, поскольку консоль обладает малым количеством кнопок.

Рисунок 10 — Консоль GameBoy Advance

В Powder также 25 уровней. Главной целью игры является победа над монстром по имени Baezl'bub и завладение его сердцем.

У персонажа есть всего 2 параметра — жизнь и мана. При убийстве монстров герой получает опыт, необходимый для повышения уровня. При переходе на новый уровень игра предлагает на выбор несколько богов, за которыми вы можете последовать. Разные боги дают разные бонусы.

В игре большое количество различных предметов, способностей и заклинаний.

Elona — roguelike JRPG 2008 года, причем в данной игре упор делается именно на ролевую составляющую, нежели на roguelike. Вопреки жанру roguelike, в данной игре смерть не является необратимой, и лишь снижает имеющиеся ресурсы и способности, а случайная генерация происходит лишь в части мест.

В Elona (рисунок 11) имеется большое количество квестов и персонажей, что более присуще обычным RPG играм, а не roguelike.

Персонаж попадает в страну, которая называется Северный Тайрис, где периодически распространяется болезнь, заставляющая живых существ мутировать. Данные мутанты, в том числе, будут являться врагами во время путешествия главного героя.

Рисунок 11 — Скриншот игры Elona

Для исследования в Северном Тайрисе персонажу доступны несколько городов и ряд других локаций. Есть возможность создать свои собственные локации. Дополнительно доступны как заранее созданные, так и процедурно генерируемые подземелья, на нижнем уровне которых есть боссы, дающие игроку награду за победу над ними.

Хоть в игре и есть основное задание, необходимое для прохождения игры, разработчик задумал игру именно не как сложную и запутанную, а простую и забавную, в которой игрок может спокойно исследовать мир в свое удовольствие.

Игроку доступно огромное множество разнообразных навыков и умений для освоения, среди которых есть как боевые и магические навыки, так и различные профессии, в том числе шитье одежды, рыболовство, садоводство, шахтерское дело и другие.

Поскольку игра представляет собой, прежде всего PRG, то игровая сессия может затянуться на многие часы.

Lost Labyrinth — данная игра (рисунок 12) была создана в 2001 году. Данная игра имеет возможность игры вчетвером на одном компьютере.

Рисунок 12 — Скриншот игры Lost Labyrinth

Во время создания персонажа игрок может выбрать несколько навыков из огромного списка (при этом навыки могут быть как положительные, так и отрицательные), причем эти навыки напрямую повлияют на последующую генерацию уровней. У героя есть более 10 различных характеристик, среди которых есть как стандартные здоровье и мана, так и восприятие, голод, жажда или нагрузка.

Целью игры является нахождение девяти фрагментов Посоха.

Во время изучения локаций игрок может встретить торговцев, у которых можно купить снаряжение или еду.

В игре отсутствует получение опыта за убийство врагов. Игрок получает опыт только при переходе на следующий уровень.

Lost Labyrinth может похвастаться тем, что игровая сессия быстрая — в среднем занимает 10–40 минут.

1.3 Анализ средств разработки и обоснование выбора технологии проектирования для всех элементов проекта

Игровой движок (game engine) — программное обеспечение, предназначенное для разработки компьютерных игр и прочих интерактивных программ, обрабатывающих графику в реальном времени.

Для анализа было взято и рассмотрено несколько игровых движков — Unity 2018, Unreal Engine 4, CryEngine V.

Unity 2018 — кроссплатформенный движок для разработки игр, разработанный компанией Unity Technologies. Список поддерживаемых платформ крайне большой — Windows, Mac OS, Android, iOS, Fire OS, Linux, PS4, PS Vita, Xbox One, Nintendo 3DS, Nintendo Switch, платформы виртуальной реальности (Oculus Rift, Steam VR, Gear VR, PlayStation VR), телевизионные операционные системы (Android TV, Samsung SMART TV, tvOS) и некоторые другие.

Данный движок обладает довольно низкими системными требованиями: Windows 7/8/10 64x или Mac OS X 10.9+, процессор с поддержкой набора инструкций SSE2, видеокарта с поддержкой DirectX 10.

Движок Unity (рисунок 13) обладает удобным и понятным интерфейсом и имеет богатый функционал.

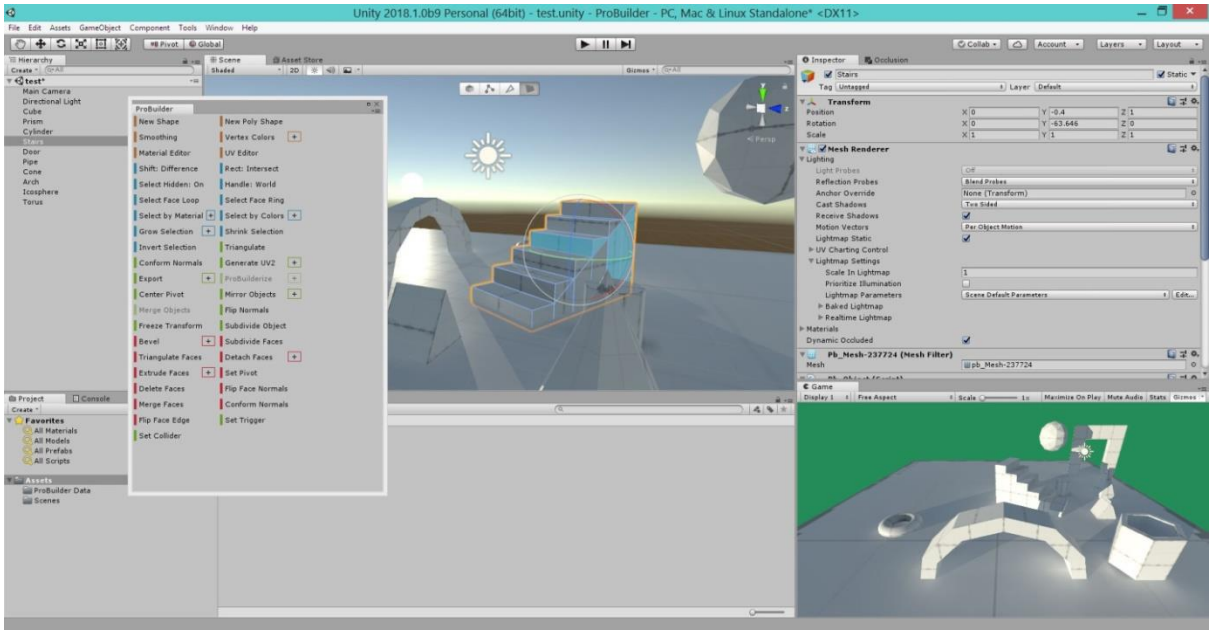

Рисунок 13 — Интерфейс Unity 2018

Имеет инструменты, как для художников-разработчиков (например, Timeline для разработки анимационных сцен, Cinemachine — набор «умных» и динамических камер, Progressive Lightmapper для работы с освещением и отличная поддержка программы Autodesk Maya для работы с 3D-анимацией и моделированием), так и для программистов-разработчиков. Движок позволяет вести разработку как 2D, так и 3D проектов.

Имеется поддержка движков Box2D (движок для работы с двумерными объектами) и NVIDIA PhysX (движок для работы с физикой объектов).

Данный движок имеет отличную оптимизацию, что сказывается на скорости и качестве проекта.

В Unity поддерживает такие языки программирования как C# и UnityScript (основанный на JavaScript).

Unity имеет свой собственный магазин Asset Store, в котором представлен огромный каталог материалов для разработки, как платных, так и бес-

платных: можно найти все необходимое — рисунки, модели, скрипты, различные дополнения и расширения и прочее.

Имеется возможность разрабатывать сетевые игры с помощью сервиса Unity Multiplayer. Данные игры используют серверы Unity Matchmaker, что сильно упрощает задачу соединения пользователей.

Unity поддерживает одновременную работу над одним проектом несколькими разработчиками.

На официальном сайте движка можно найти огромное количество документации (рисунок 14), в том числе и обучающей, что, несомненно, является неплохим подспорьем начинающего разработчика.

Обучающая информация представлена как отдельными уроками, подробно разбирающими какие-либо темы, так и проектами по созданию небольших игр.

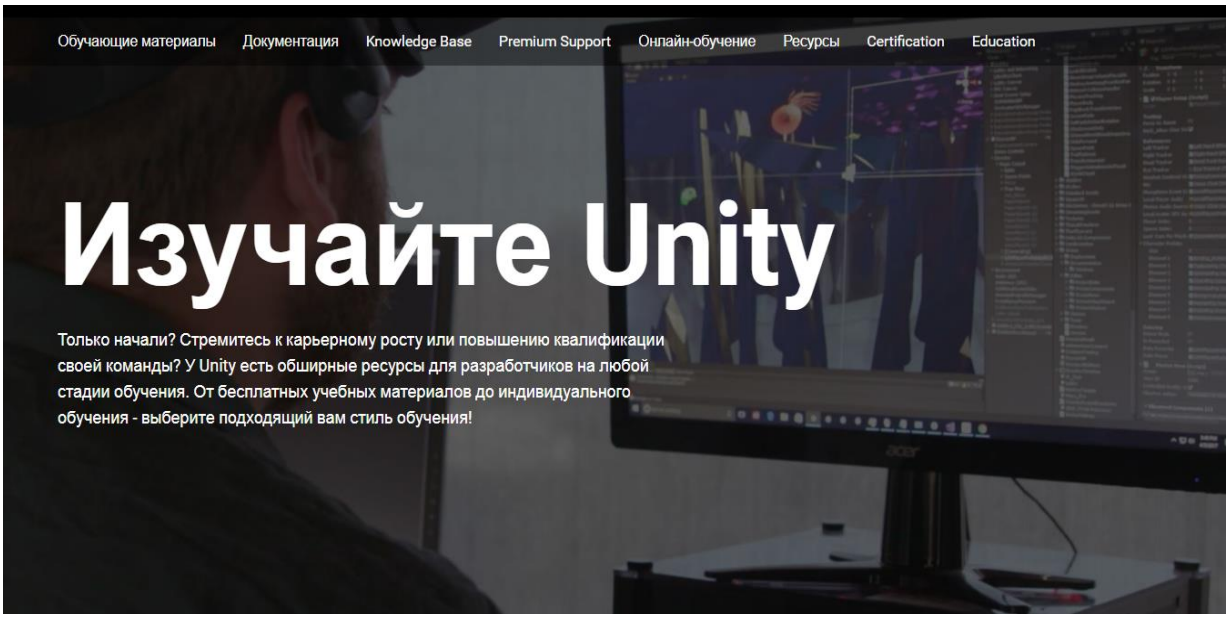

Рисунок 14 — Раздел официального сайта движка с документацией

Поскольку это один из самых распространенных игровых движков, в интернете представлено большое количество неофициальной обучающей информации.

Данный движок имеет 3 версии — Personal (бесплатно), Plus (35$ в месяц), Pro (125$ в месяц). Данные версии отличаются лишь предлагаемым

функционалом и максимально допустимым годовым доходом (100 тыс. долларов, 200 тыс. долларов и без ограничений соответственно).

Движок является очень универсальным и позволяет использовать его для разработки самых разнообразных продуктов.

CryEngine V — самая последняя версия игрового движка от компании Crytek. Данный движок может похвастаться отличной графикой. Первая версия движка CryEngine использовалась компанией при разработке первой части игры Far Cry (рисунок 15).

Рисунок 15 — Скриншот игры Far Cry

CryEngine V хоть и является кроссплатформенным движком, поддерживает всего лишь 4 платформы — Windows, Playstation 4, Xbox One, Oculus Rift.

Минимальные системные требования не очень высокие, хоть и выше, чем у Unity: Windows 7/8/10 64x, процессор Intel Dual-Core от 2GHz (Core 2 Duo и выше)/AMD Dual-Core от 2GHz (Phenom II X2 и выше), оперативная память 4 GB, видеокарта NVIDIA GeForce 450 и выше/AMD Radeon HD 5750 и выше, поддержка DirectX 11.

Движок имеет простой и понятный интерфейс (рисунок 16), во многом схожий с интерфейсом Unity.

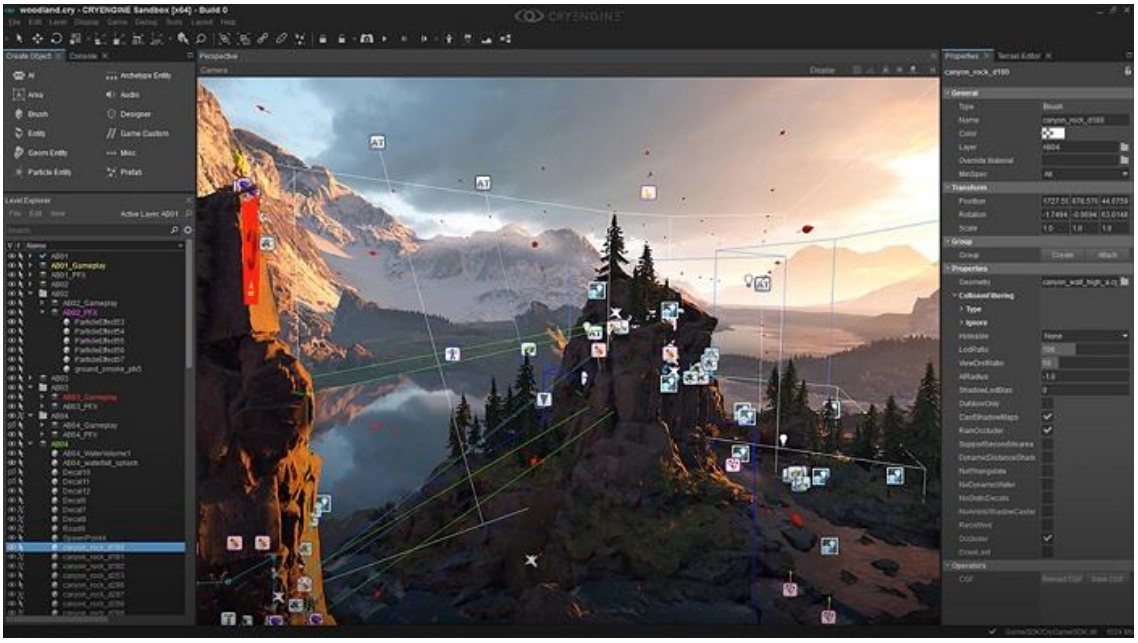

Рисунок 16 — Интерфейс CryEngine V

Движок CryEngine делает большой упор на визуальную составляющую проекта, поэтому имеет большое количество функций и инструментов для достижения превосходного качества изображения. Имеется поддержка DirectX 12.

Система Physically Based Rendering позволяет имитировать взаимодействие света и материалов с использованием физики реального мира, что помогает добиться более правдоподобного вида объектов. Высокое качество изображения воды достигается путем динамической обработки водных каустик в режиме реального времени. Данный движок обладает эффективным сглаживанием, что позволяет снизить пикселизацию изображения. Наличие теневых карт с широким диапазоном настроек помогает добиться наилучшего отображения теней. Имеется HDR-кривая для настройки освещения, позволяющая использовать широкий тональный диапазон, с помощью которого можно улучшить определение темных, средних и светлых тонов без особых усилий.

В движке присутствует встроенный редактор материалов. Система CryEngine Sandbox предлагает большое количество инструментов для быст-

рого создания игровых уровней и миров. Редактор TrackView представляет собой встроенный инструмент редактирования видеозаписей для создания интерактивных сцен с зависящим от времени управлением объектами и событиями. Инструмент Designer Tool является редактором моделей и позволяет экспортировать созданные объекты во внешние инструменты

Хотя CryEngine и поддерживает языки программирования C++ и Lua, в нем доступна система визуальных скриптов Flowgraph, которая позволяет создавать и контролировать игровую логику и события без необходимости писать скрипты вручную.

Движок имеет большое количество настроек анимации персонажей, в том числе параметрическую скелетную анимацию. Имеется встроенная расширенная система искусственного интеллекта, позволяющая настроить реалистичное поведение неигровых персонажей.

CryEngine представляет инструментарий для анализа производительности игры — можно отслеживать проблемы производительности прямо во время игры, а инструмент Statoscope (рисунок 17) позволяет получать всю информацию по расходу ресурсов компьютера в графическом виде.

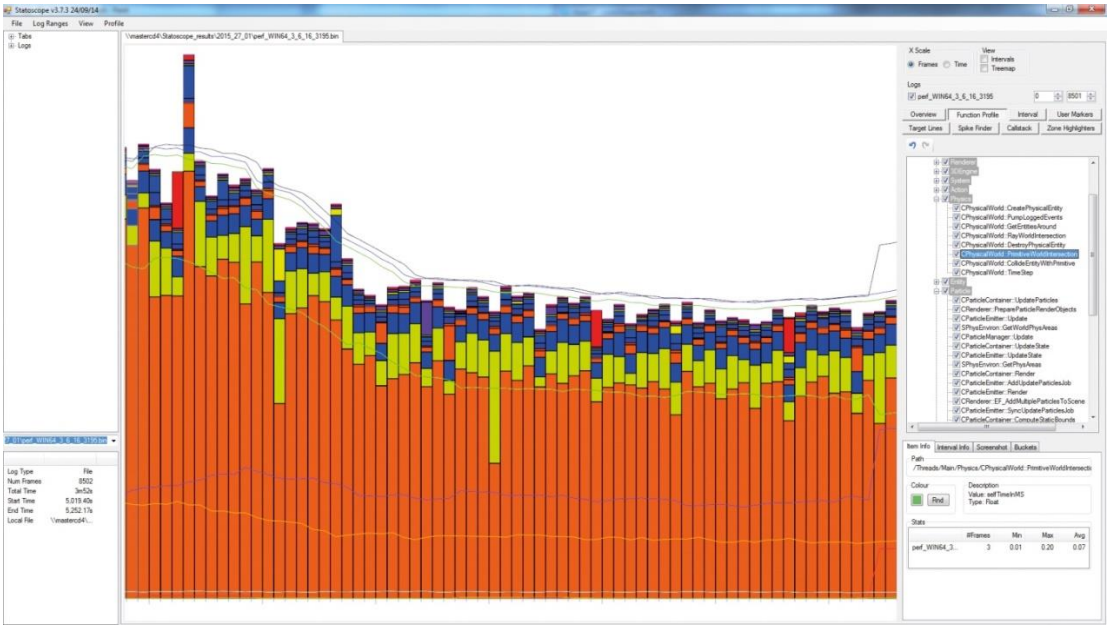

Рисунок 17 — Инструмент Statoscope

Проекты, созданные на CryEngine, имеют открытый исходный код.

На официальном сайте имеется обучающая информация и магазин с различными материалами для разработки — модели, скрипты, анимации, звуки и прочее (как платные, так и бесплатные).

Движок распространяется со свободной лицензией, однако, если продукт, созданный на движке, приносит прибыль более 5 тысяч долларов в год, то 5 % прибыли отчисляется Crytek.

Данный движок является отличным инструментом для разработки 3D проектов.

Unreal Engine 4 — игровой движок, разработанный компанией Epic Games. Первая версия Unreal Engine была разработана для игры этой компании под названием Unreal (рисунок 18).

Рисунок 18 — Скриншот игры Unreal

Движок Unreal Engine поддерживает следующие платформы — Windows, Mac OS, Android, iOS, Nintendo Switch, Linux, PS4, Xbox One, Oculus Rift, PlayStation VR, Samsung Gear VR, Viveport, Daydream, HTML5.

Данный движок обладает следующими рекомендуемыми системными требованиями: Windows 7/8/10 64x или Mac OS 10.13 или Linux Ubuntu 15.04, процессор Intel Quad-core или AMD (2.5 GHz и выше), оперативная память 8 GB (16 GB для Linux), видеокарта с поддержкой DirectX 11 (для Windows) или Metal 1.2 Compatible Graphics Card (для Mac OS) или NVIDIA GeForce 470 GTX и выше (для Linux). Минимальные требования ограничены лишь версией операционной системы.

Движок имеет удобный, настраиваемый интерфейс (рисунок 19), сильно отличающийся от интерфейсов предыдущих движков.

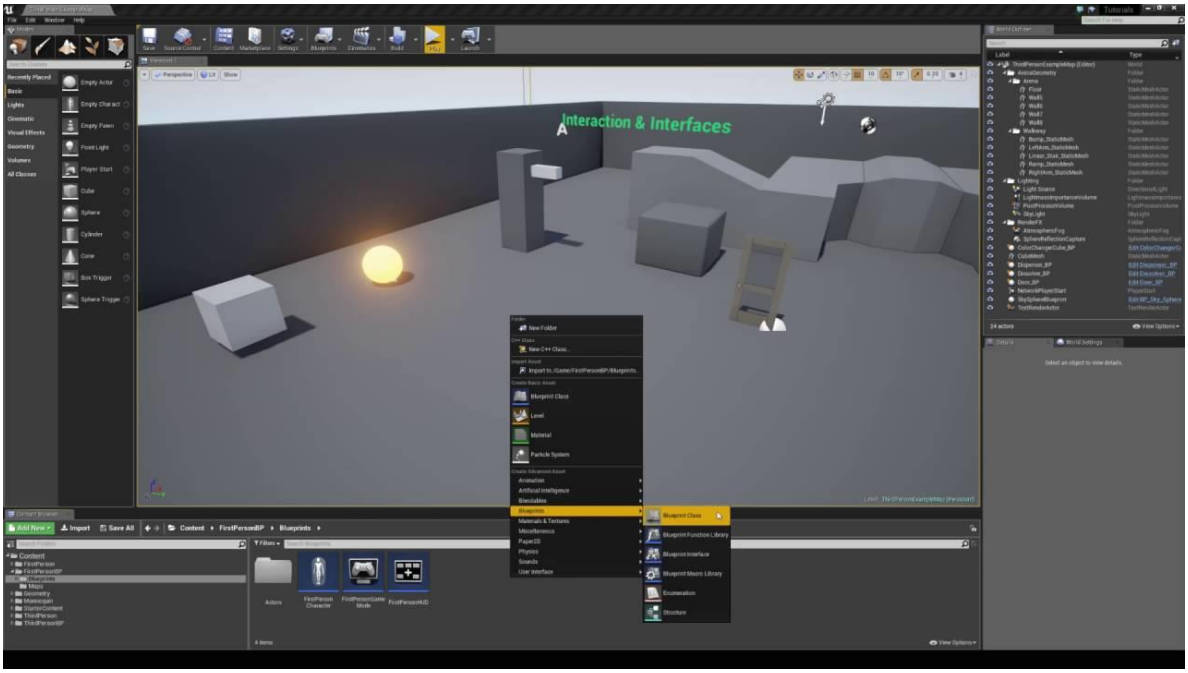

Рисунок 19 — Интерфейс Unreal Engine 4

Движок позволяет добиться фотореалистичной графики с помощью большого количества настроек рендера, динамических теней, отражений каналов освещения.

Инструментарий движка позволяет существенно упростить работу с ландшафтом и растительностью.

Имеется полный доступ к исходному коду на языке C++.

Благодаря системе визуальных скриптов Blueprint есть возможность управлять игровой логикой без написания скриптов вручную, хоть такая возможность и имеется (на том же языке C++).

Встроенный редактор визуальных эффектов Cascade позволяет настраивать системы частиц с использованием самых разных модулей. В движок встроен редактор материалов, использующий искусственное затенение и дающий беспрецедентный контроль над внешним обликом персонажей и объектов.

Используется обширный набор анимационных инструментов, позволяющих редактировать сетку и анимацию персонажей. Кроме того, получившийся результат можно сразу же посмотреть и оценить.

Для создания анимированных сцен и видеороликов используется инструмент Sequencer, позволяющий настраивать освещение, камеру и персонажей.

Unreal Engine имеет широкую поддержку разработки мультиплеерных игр. Данный движок поставляется с масштабируемой и проверенной архитектурой клиент/сервер.

Unreal Engine позволяет работать в режиме виртуальной реальности, с расширенными элементами управления движением. Благодаря тесному сотрудничеству компании Epic Games с мировыми лидерами в области аппаратного и программного обеспечения, Unreal Engine обеспечивает высокое качество взаимодействия с системами виртуальной и дополненной реальности.

Расширенная система искусственного интеллекта позволяет настраивать реалистичное поведение неигровых персонажей.

На официальном сайте имеется официальная документация и обучающая информация. Имеется магазин с различными материалами для разработки — модели, скрипты, анимации, звуки, плагины и прочее (как платные, так и бесплатные).

Данный движок наиболее подходит для разработки 3D игр, нежели 2D, однако имеет широкий выбор платформ, как для разработки, так и для готовых проектов.

Так как для реализации данные проекта не требуется подключение каких-либо дополнительных инструментов, главными критериями выбора среды разработки являются следующие: бесплатная лицензия, наличие обучающей документации, низкие системные требования, удобный инструментарий для разработки 2D проектов. Данным критериям более всего соответствует движок Unity.

Для реализации программного кода в движке Unity используется Microsoft Visual Studio 2017 (рисунок 20), который имеет возможность интеграции в Unity.

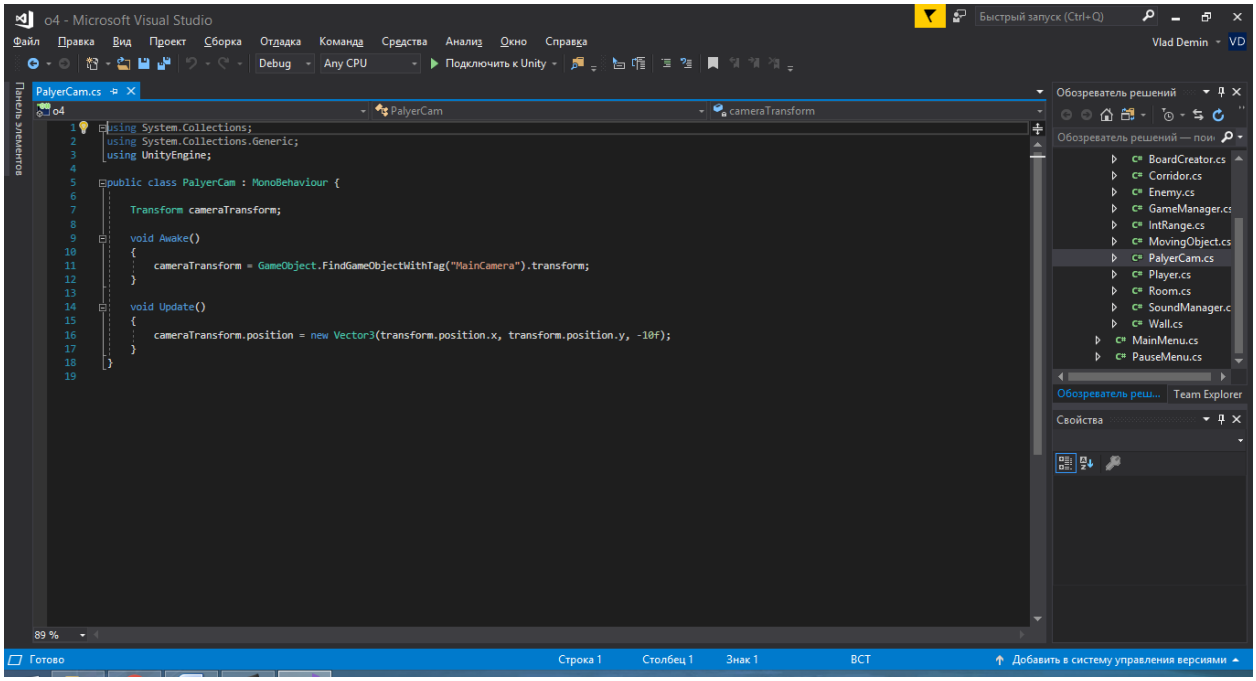

Рисунок 20 — Интерфейс Microsoft Visual Studio 2017

Visual Studio 2017 — программный продукт компании Microsoft, включающий в себя интегрированную среду разработки программного обеспечения. Включает в себя редактор исходного кода и имеет возможность рефакторинга кода. Обладает редактором форм для упрощения создания графического интерфейса приложений. Имеется возможность расширенной настройки продукта под свои нужны, доступно подключение различных сторонних дополнений.

Данный продукт является удобным инструментом программирования и отлично подходит для написания кода.

Одним из важнейших этапов разработки компьютерной игры является работа с визуальной частью. В этом могут помочь графические редакторы. Есть большое количество различных графических редакторов, обладающих различным функционалом и предназначенных для различных задач, но среди них имеется один, который одновременно обладает внушительным инструментарием, но при этом является легким в изучении.

Adobe Photoshop CC 2017 (рисунок 21) — растровый графический редактор, распространяемый компанией Adobe Systems. Данный редактор поддерживает как растровые, так и векторные форматы изображений и поддерживает все самые распространенные цветовые модели (RGB, SMYK, LAB и другие). Имеется поддержка графических планшетов и расширенный функционал работы как с графикой, так и со скриптами и анимацией. Adobe Photoshop распространяется по платной лицензии (от 1288 рублей в месяц).

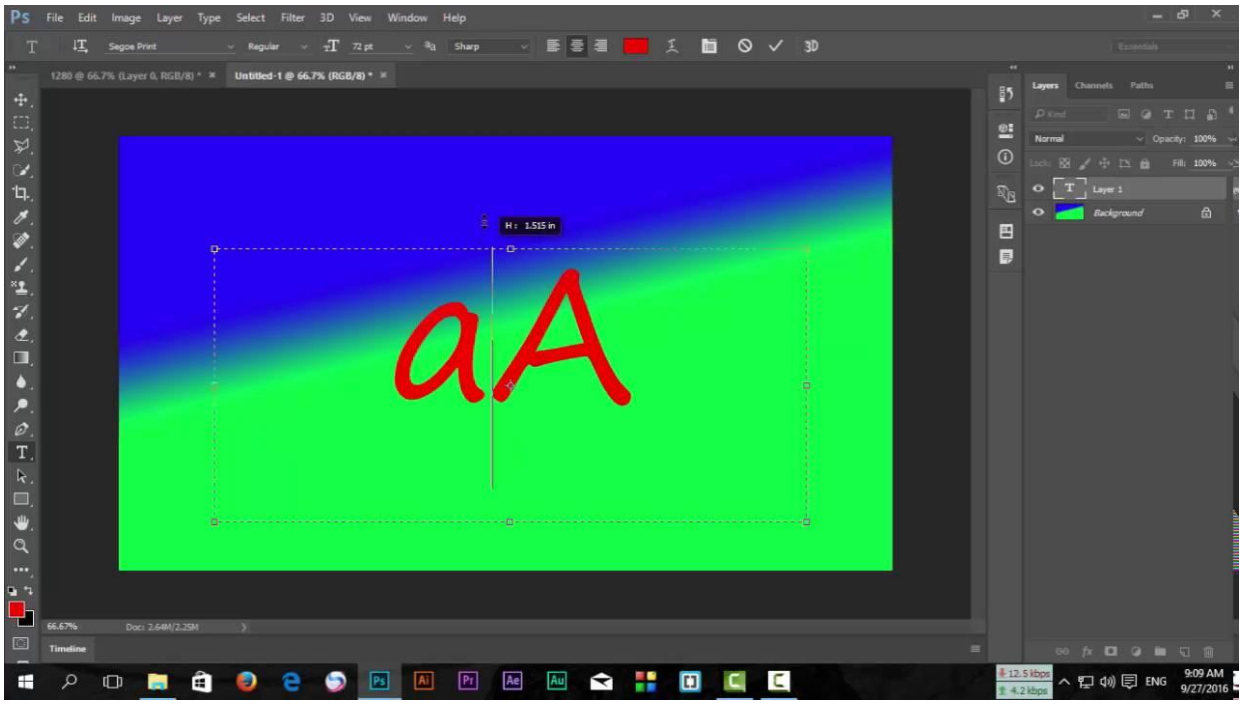

Рисунок 21 — Интерфейс Adobe Photoshop CC 2017

Поскольку данный графический редактор еще и изучался во время обучения в университете, именно он был выбран для работы с графической частью игры.

1.4 Общий алгоритм реализации проекта

Этап 1 — разработка концепта игры. На данном этапе выбирается жанр игры, ее сеттинг, ее вида (2D/3D), вида камеры (от первого лица/от третьего лица), пишется сценарий. На этом же этапе, выбирается среда разработки игры и платформа и определяется целевая аудитория.

Этап 2 — написание скриптов. Сюда входят скрипты процедурной генерации и обработчика уровня, скрипт передвижения (персонажа и врагов), скрипт персонажа и врагов (взаимодействие героя с бонусами и врагами).

Этап 3 — отрисовка графики. Рисуются тайлы для генерации уровня и спрайты бонусов, спрайты героя и врагов, с последующей их анимацией.

Этап 4 — тестирование продукта. Здесь происходит проверка работоспособности продукта и отдельных его механик.

Этап 5 — запаковка проекта и выпуск игры.

2 ПРОЕКТНАЯ ЧАСТЬ

2.1 Характеристика потенциальной аудитории потребителей проекта, ориентированность работы

Характеристика целевой аудитории является одним из значительных этапов разработки любого проекта, так как это поможет разработчику лучше понять целевую аудиторию и, следовательно, более внимательно отнестись к тем или иным деталям разрабатываемого продукта. Поэтому для проекта была описана его целевая аудитория.

Типичный представитель целевой аудитории разрабатываемой выглядит так: подросток 11–19 лет, преимущественно мужского пола (поскольку жанр ролевых игр более распространен у лиц мужского пола) доход низкий или отсутствует. Место жительство любое — от небольших населенных пунктов до крупных городов. Род занятий — учеба или работа. Свободное время предпочитает проводить, играя в игры или сидя в интернете. Готовы узнавать/пробовать новое.

Поскольку жанр roguelike является довольно специфичным и не самым распространенным, его целевая аудитория не так обширна, как у более популярных жанров. Также этому способствует довольно высокая сложность игр данного жанра, игр в жанре roguelike с низкой сложностью крайне мало. Именно поэтому целевой аудиторией разрабатываемой игры были выбраны людей, которые хотят познакомиться с данным жанром. Разрабатываемая игра не обладает такой высокой сложностью, как большинство представителей данного жанра, но при этом позволяет понять основные механики, присущие жанру roguelike.

Поскольку игровые сессии крайне быстрые, то данная игра может приглянуться людям с небольшим количеством свободного времени.

Игра обладает пиксельной графикой, что может привлечь любителей проектов такого вида.

В общем, данный проект может заинтересовать самую разнообразную аудиторию.

2.2 Постановка задачи проекта

2.2.1 Актуальность проекта

В наши дни игровая индустрия развивается быстро, с каждым годом — растет количество активных игроков и расширяется их аудитория, появляются новые студии и новые проекты. Все это приносит огромную прибыль. Со временем стало развиваться направления инди-разработки (благодаря доступности средств разработки и сервисам цифровой дистрибуции), что способствовало появлению большого количества независимых разработчиков, многие из которых могут представить различные интересные проекты.

Жанр roguelike является не особо распространенным среди игроков, потому что такие игры, как правило, являются очень сложными и нередко требуют времени для того, чтобы разобраться даже в основных механиках. Многие из людей бросают знакомство с жанром именно по этим причинам.

Данный проект отличается тем, что здесь жанр roguelike (несмотря на наличие основных для жанра механик) немного упрощен, что способствует более быстрому освоению основных механик. Данная игра может служить не только для знакомства с жанром, но и при этом не отпугнуть потенциального игрока, а может и привлечь к данному жанру.

2.2.2 Цель и назначение проекта

Можно выделить следующую цель проекта — анализ средств разработки компьютерных игр с последующей разработкой компьютерной игры в

жанре roguelike. Создание уникального продукта использующего основные отличительные особенности жанра. Знакомство потенциальной целевой аудитории с не самым распространенным, но интересным жанром.

Разработка игры помогает получить ценный опыт, который может пригодиться не только при разработке игр, но и во многих других областях, а опыт работы с движком, полученный во время разработки, позволяет сократить время работы при работе со следующими проектами.

2.2.3 Функционал проекта

Проект представляет собой двумерную компьютерную игру в жанре roguelike.

Проект должен отвечать следующим требованиям:

1. Игра должна обладать простым и понятным интерфейсом.
2. В игре должно быть простое управление.
3. Игра должна включать в себя основные геймплейные особенности жанра roguelike, а именно:
   - случайная генерация игрового окружения, которая уникальна для каждого нового прохождения;
   - пошаговость действий;
   - необратимость действий;
   - полная доступность всех игровых действий с самого начала игры;
   - самостоятельное исследование окружающего мира.
4. Враги ходят реже игрока и все время движутся в его направлении.
5. На уровнях должны иметься собираемые объекты, увеличивающие игровой счет игрока или восстанавливающие здоровье игроку.
6. Должна быть реализована прогрессия развития игрока — получение очков опыта за убийство врагов и повышение уровня за очки опыта. Повы-

шение уровня влечет за собой повышение урона и максимального здоровья игрока.

7. С увеличением уровня игры игровой процесс должен усложняться (увеличиваться характеристики врагов и их количество).

2.2.4 Входные данные к проекту

В качестве входных данных были использованы три мелодии (одна воспроизводится в меню, вторая во время игры, третья, победная, во время подбирания игроком Кубка Света), находящиеся в свободном доступе на сайте opengameart.org.

Были использованы звуки, находящиеся в свободном доступе на сайте freesound.org, среди которых были следующие:

- шаги;
- удар оружием по врагу;
- удар по камню;
- глотание воды;
- звон мешка с монетами;
- рев зомби;
- шипение змеи;
- вой привидения.

В игре используется шрифт Press Start 2P (рисунок 22). Данный шрифт распространяется по свободной лицензии автором. Шрифт Press Start 2P основан на шрифтах из ряда классических игр аркадных игр компании Namco, благодаря чему этот шрифт отлично подходит для игр, выполненных в ретро стиле. Кроме того, данный шрифт поддерживает кириллицу, позволяя писать на русском языке, что, несомненно, является плюсом.

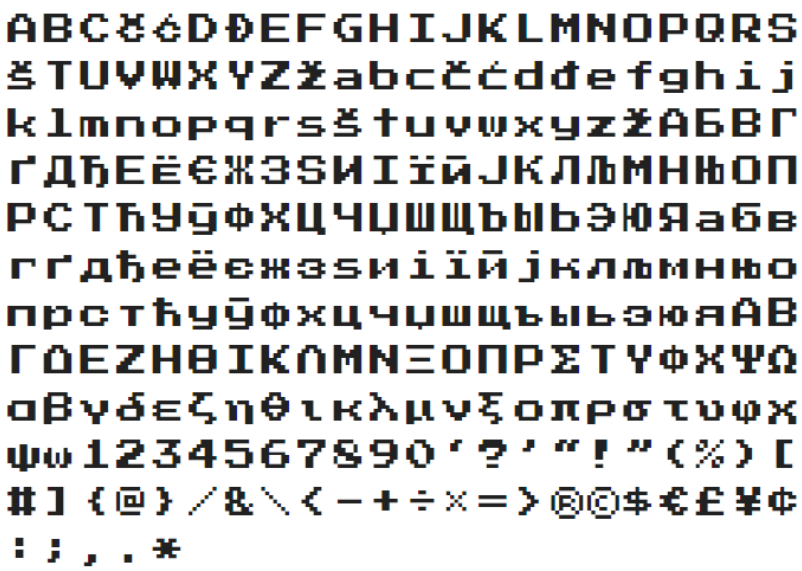

Рисунок 22 — Шрифт Press Start 2P

2.2.5 Характеристики оборудования для реализации проекта

Для реализации проекта был использован ноутбук Acer Aspire E1-570G со следующими техническими характеристиками:

- операционная система: Windows 8.1 (64x);
- процессор: Intel Core i3-3217U, 1.8 ГГц;
- объем оперативной памяти: 4 Гб;
- видеокарта: NVIDIA GeForce 720M 1 Гб VRAM.

2.3 Жизненный цикл проекта, описание поэтапной реализации проекта с указанием средств реализации

2.3.1 Этап эскизного проектирования и разработки элементов дизайна

Первым этапом в разработке игры стал этап эскизного проектирования. В первую очередь на данном этапе разработки необходимо выбрать сеттинг, то есть принадлежность к какой-либо сюжетной теме или миру (реальному или виртуальному). В качестве сеттинга было выбрано приключение в сред-

невековом фэнтезийном мире. Именно в таком сеттинге было выполнено подавляющее большинство классических roguelike игр.

Поскольку многие из классических roguelike игр были выполнены либо в псевдографическом стиле, либо использовали простую и примитивную графику, то было решено использовать в игре пиксельную графику. Во-первых, такой вид графики пользуется высоким спросом в наши дни, а во-вторых, пиксельная графика внешне близка к графике игр жанра roguelike. Для игры было выбрано разрешение 32x32 pxl, то есть все тайлы и объекты будут иметь размеры 32 на 32 пикселя.

В качестве главного героя было принято решение сделать рыцаря. Поскольку было решено создать для героя анимацию получения урона, то вид рыцаря в шлеме не рассматривался. После чего было сделано несколько набросков рыцаря (рисунок 23) и несколько вариантов лица и оружия.

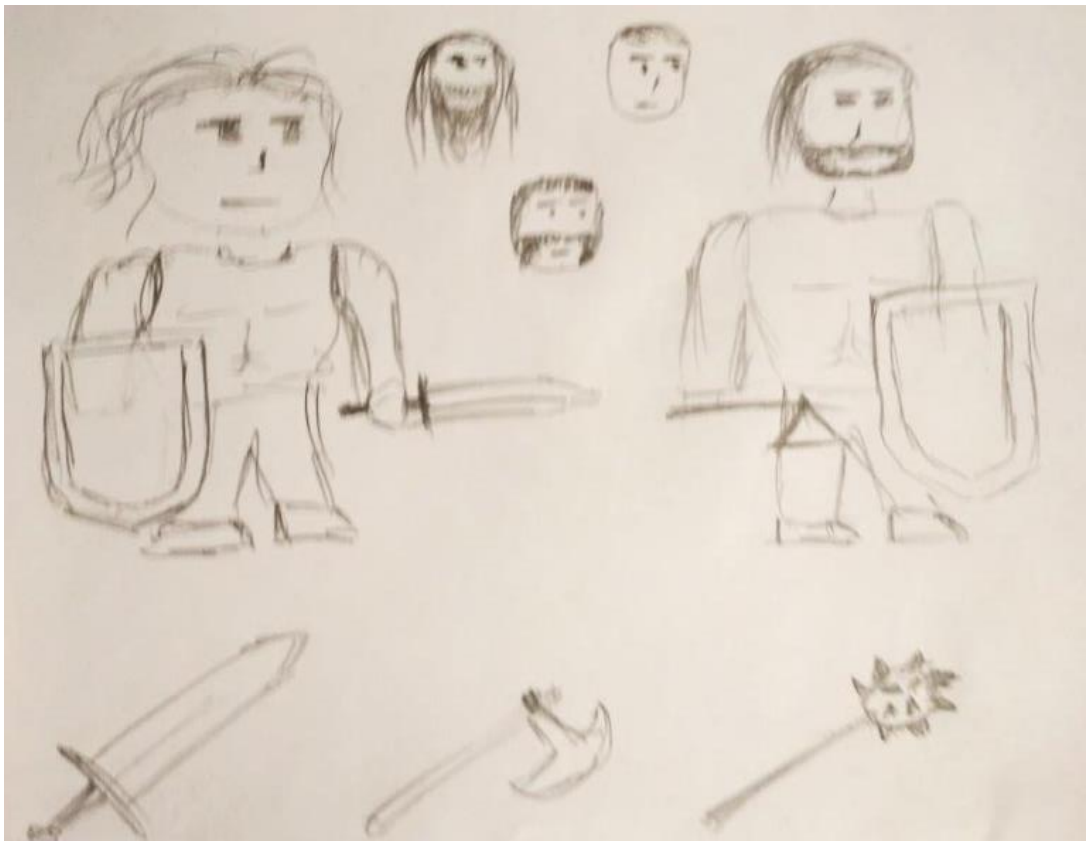

Рисунок 23 — Набросок внешнего вида главного героя и его оружия

По данным эскизам было создано два разных спрайта рыцаря (рисунок 24), из которых был выбран второй.

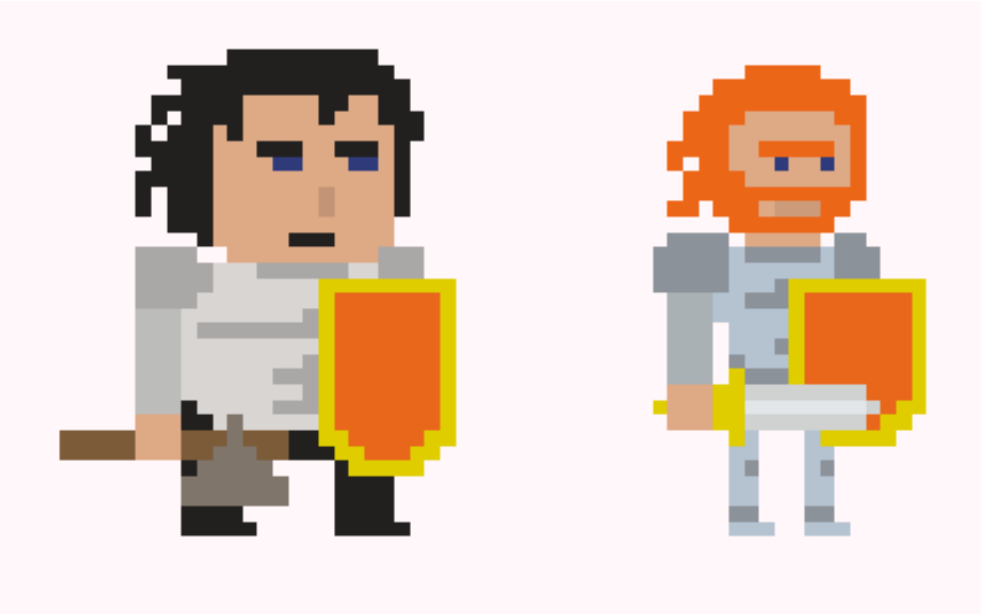

Рисунок 24 — Варианты спрайта героя

Для этого спрайта было покадрово отрисовано 3 разных анимации (рисунок 25) — анимация простоя, то есть анимация объекта по умолчанию, имитирующая стояние на месте и дыхания персонажа, а также анимации атаки и получения урона.

Рисунок 25 — Лист анимации персонажа

Фэнтезийный сеттинг позволяет использовать различных мифических существ, поэтому в качестве врагов были выбраны следующие существа — зомби, привидение и гигантская змея. Для них были сделаны наброски (рисунок 26), по которым были созданы спрайты врагов.

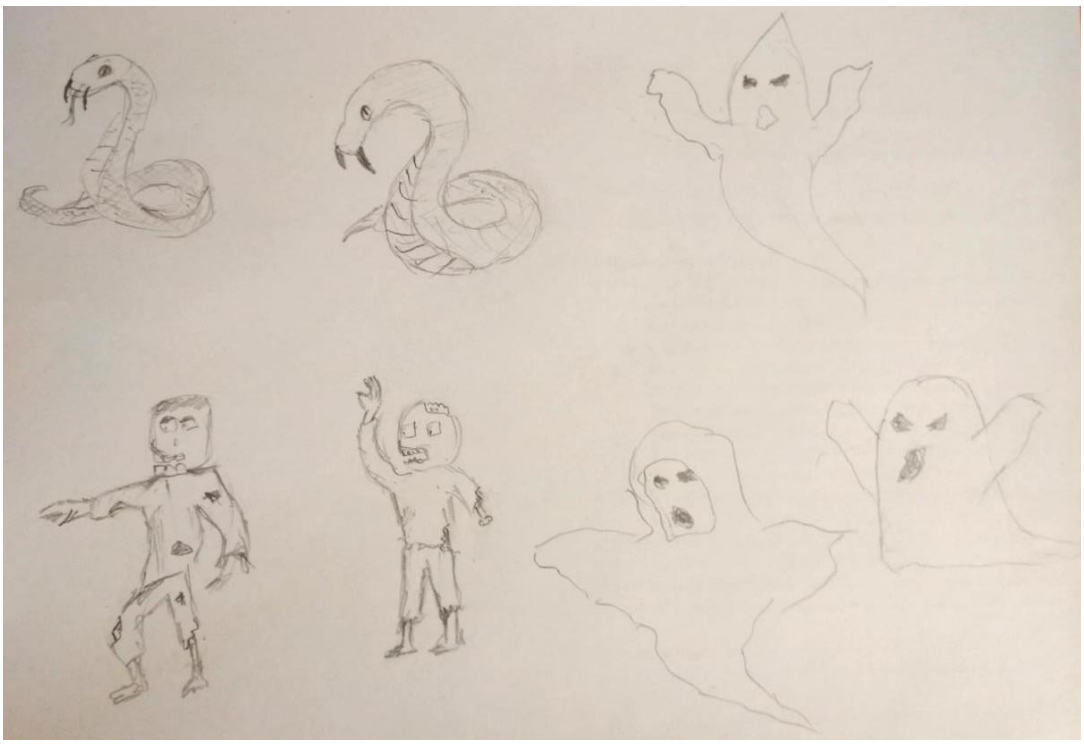

Рисунок 26 — Наброски врагов для игры

Для данных спрайтов (рисунок 27) были нарисованы кадры анимации простоя и атаки.

Рисунок 27 — Спрайты змеи, зомби, привидения

Были отрисованы тайлы окружения (стены и пол) трех видов (рисунок 28) для создания разного внешнего вида уровней: каменное подземелье, песчаное подземелье, ледяное подземелье, и тайл перехода на следующий уровень.

Данные тайлы являются бесшовными, это позволяет составлять из них локации любого размера, и при этом стыки спрайтов не будут бросаться в глаза

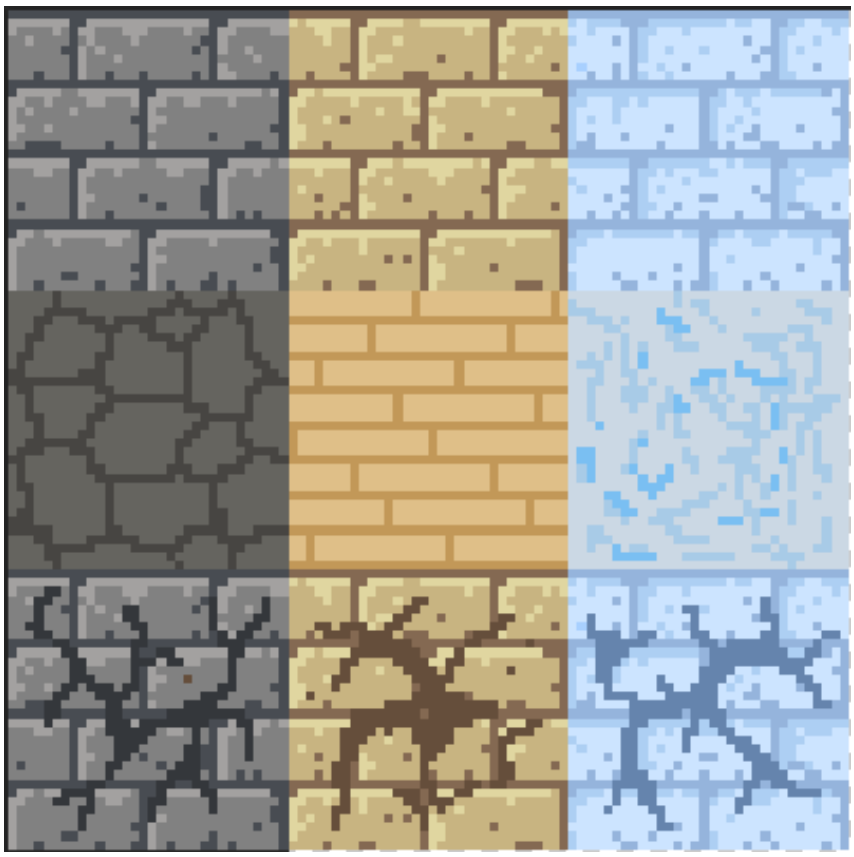

Рисунок 28 — Тайлы уровней трех видов

Были отрисованы спрайты зелий (большого и маленького), мешка с золотом (рисунок 29) и кубка.

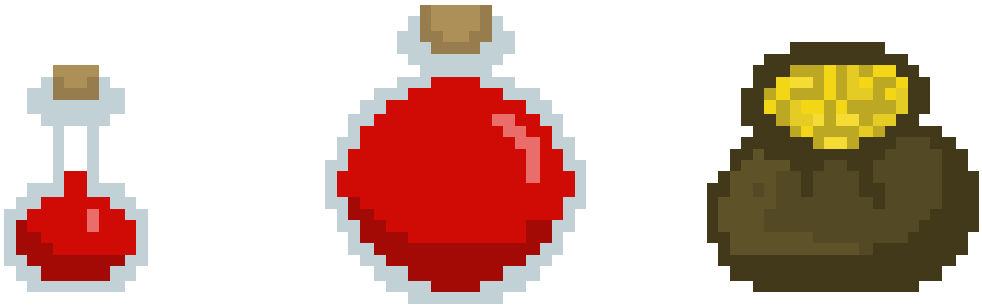

Рисунок 29 — Спрайты зелий и золота

Необходимо на данном этапе описать сюжет. Игры в жанре roguelike, как правило, не имеют глубокого сюжета и обычно его описание ограничено кратким введением игрока в события игры в самом начале. Такой же подход к рассказу сюжета был использован и в данном проекте.

Игроку предстоит взять на себя роль искателя приключений, отправляющегося на поиски волшебного артефакта Кубка Света (рисунок 30), кото-

рый находится в глубинах подземелий и, дающий владельцу невиданную мощь.

Рисунок 30 — Главная цель игры Кубок Света

Использованный в игре шрифт имитирует шрифты аркадных автоматов и является пиксельным шрифтом, что в целом вписывается в стилистику игры.

2.3.2 Этап работы с визуальной частью игры

После импорта спрайтов в игровой движок необходимо с помощью движка нарезать лист спрайтов на отдельные спрайты, а из них создать игровые объекты — префабы (Prefabs). Для создания героя и врагов необходимо создать пустой объект (рисунок 31), выделить кадры одной из анимаций и перетащить их в созданный пустой объект.

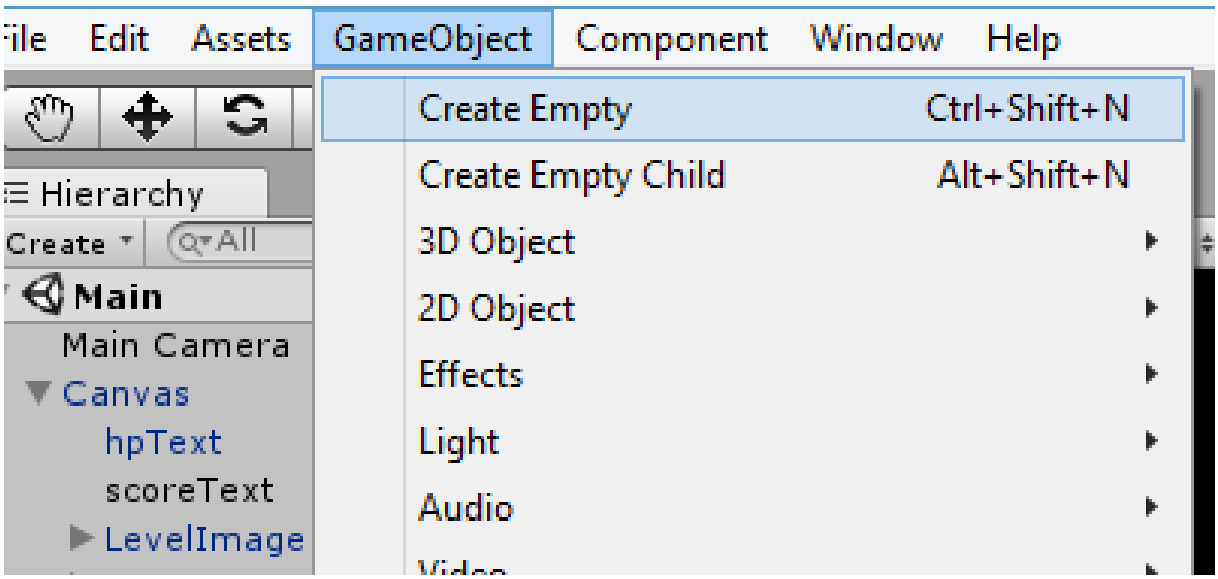

Рисунок 31 — Создание пустого объекта

Движок предложит сохранить анимацию, вместе с анимацией для объекта создается контроллер анимации (рисунок 32), позволяющий настроить переходы анимации из одного состояния в другое.

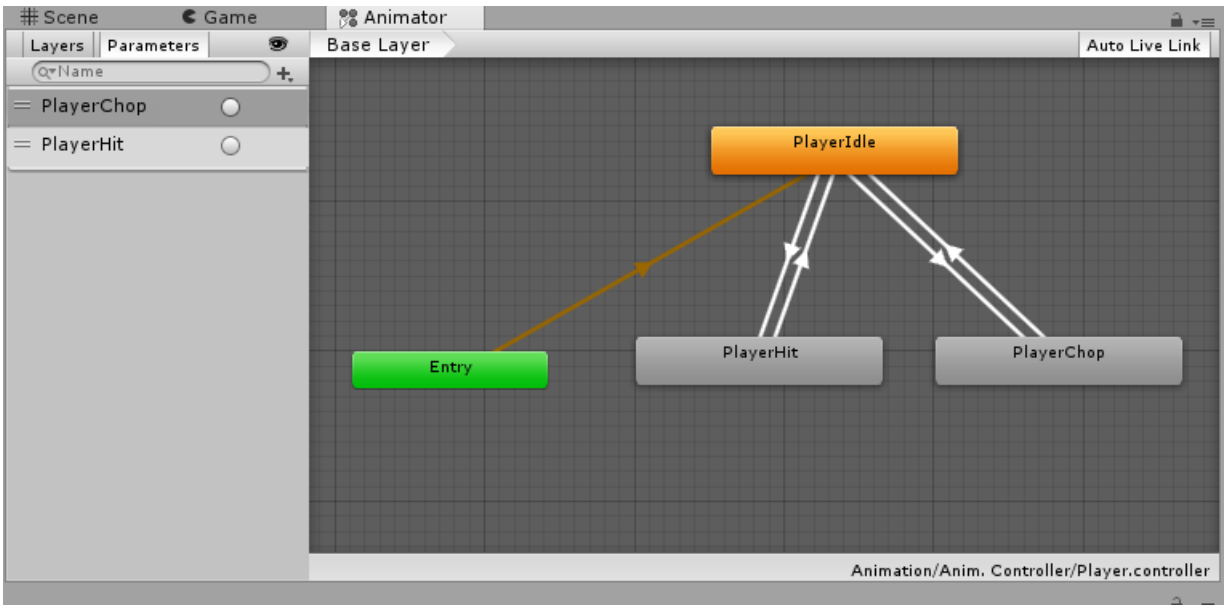

Рисунок 32 — Контроллер анимации объекта Player

Затем необходимо добавить объектам компоненты Rigidbody 2D и Box Collider 2D (рисунок 33). Компонент Rigidbody 2D позволяет настроить перемещение объекта, а Box Collider 2D отвечает за настройку коллизии объекта (то есть столкновение с другими объектами), что позволит объектам взаимодействовать друг с другом.

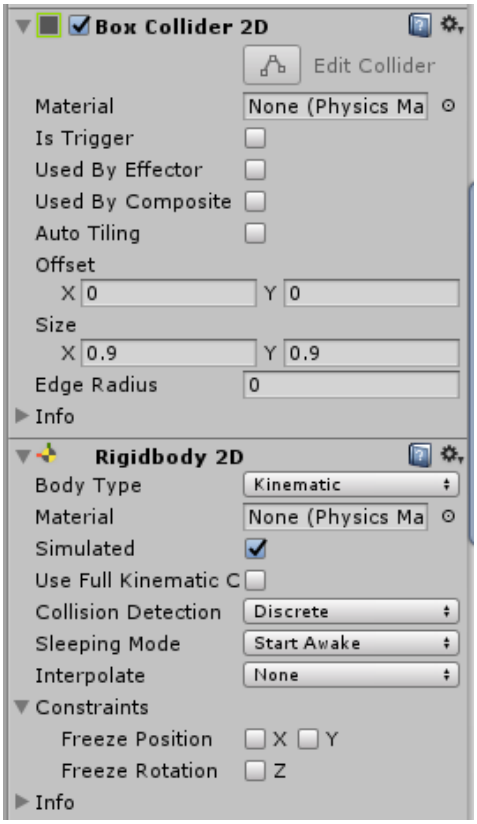

Рисунок 33 — Компоненты Box Collider 2D и Rigidbody 2D объекта Player

Для создания тайлов нужно также создать пустой объект, но добавить ему компонент Sprite Renderer (рисунок 34). В данном компоненте необходимо выбрать спрайт нужного тайла. Если подразумевается взаимодействие с объектом, то необходимо еще добавить компонент Box Collider 2D.

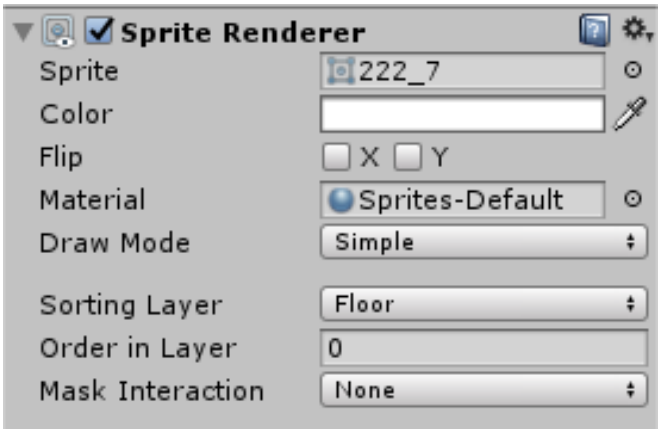

Рисунок 34 — Компонент Sprite Renderer тайла пола Floor 1

Были созданы элементы интерфейса — количество жизней героя, количество набранных очков (рисунок 35), уровень персонажа и количество опыта персонажа.

Рисунок 35 — Элемент интерфейса, отвечает за отображение очков игрока

Был создан элемент интерфейса Canvas (рисунок 36), в котором были добавлены текстовые поля (для каждого значения свое поле), обновляемые с помощью скриптов.

Рисунок 36 — Элемент интерфейса Canvas

С помощью элемента было настроено и отображение экрана уровня и меню паузы (рисунок 37).

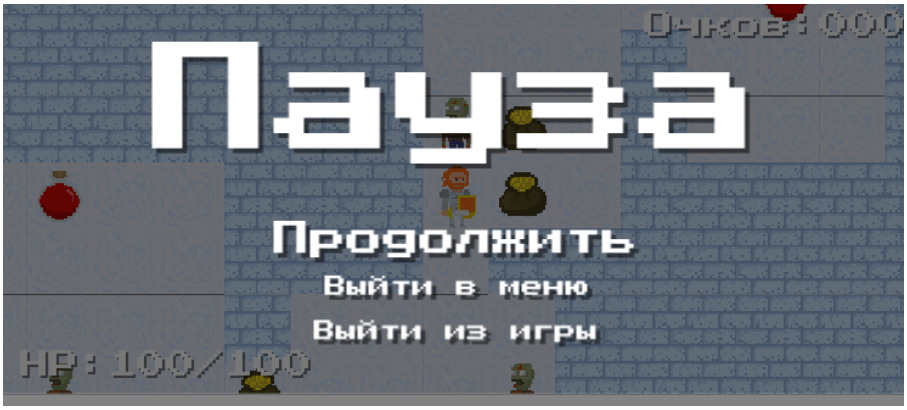

Рисунок 37 — Меню паузы игры

2.3.3 Этап программирования скриптов

Среда разработки Unity хоть и обладает массой различных компонентов с огромным количеством настроек, их функционала явно недостаточно для реализации задуманных механик, поэтому необходимо обратиться к программированию скриптов. Поскольку средой разработки является Unity, то

программирование скриптов будет осуществляться на языке программирования C#.

Всего игра содержит 13 скриптов и далее будет описан функционал каждого из них и методы основных скриптов.

Игра использует один и тот же объект GameManager, в котором генерируется каждый новый уровень.

К объекту GameManager прикреплены 2 скрипта: GameManager и BoardCreator.

Скрипт GameManager отвечает за переходы между уровнями, создание листа врагов и пошаговость игры. Данный скрипт содержит следующие методы:

1. Awake. Метод вызывается перед загрузкой игры, загружает лист врагов для данного уровня. Вызывает метод InitGame для первого отображения экрана уровня (надпись «Уровень 1»).

2. OnLevelWasLoaded. Метод считает уровни и вызывает метод InitGame для отображения следующих экранов уровня, начиная со второго.

3. InitGame. Данный метод отвечает за вывод на экран названия уровня в соответствующий текстовый элемент интерфейса (LevelText) во время смены уровня игры и очищает лист врагов.

4. HideLevelImage. Скрывает экран смены уровней.

5. Update. Метод вызывается каждый кадр игры и вызывает интерфейс IEnumerator, отвечающий за поочередное передвижение врагов.

6. AddEnemyToList. Метод добавляет в лист врагов, генерируемых с помощью скрипта Enemy.

7. GameOver. Отвечает за вывод на экран сообщения о проигрыше игрока, показывая при этом небольшую статистику по прохождению (количество пройденных уровней и набранных очков).

Кроме данных методов, скрипт GameManager содержит интерфейс IEnumerator под именем MoveEnemies. Данный интерфейс после хода игрока

позволяет каждому врагу из листа сделать ход. После хода каждого врага право хода переходит к игроку.

Скрипт Player отвечает за персонажа и все с ним связанное. Содержит следующие методы:

1. Start. Во время загрузки уровня загружает данные об игроке (жизни, очки и т.д.).
2. OnDisable. Сохраняет информацию об игроке во время переходов между уровнями.
3. Update. Проверяет каждый кадр, наступила ли очередь игрока ходить, запрещает игроку двигаться по диагонали.
4. AttemptMove. Дочерний метод скрипта MovingObject. Метод позволяет игроку перемещаться, воспроизводит звук перемещения игрока.
5. OnCantMove. Дочерний метод скрипта MovingObject. Метод отвечает за разрушение стен игроком.
6. EnemyAttack. Дочерний метод скрипта MovingObject. Метод отвечает за атаку игроком врагов.
7. OnTriggerEnter2D. Данный метод отвечает за срабатывание триггеров при взаимодействии с различными игровыми объектами (например, зелья лечения или золото).
8. Restart. Метод вызывает генерацию нового уровня.
9. LoseHP. При получении урона игроком вычитает здоровье игрока.
10. UpdatehpText. Реагирует на изменения количества здоровья игрока и изменяет ответственный за вывод на экран количества здоровья элемент интерфейса на соответствующее количество.
11. UpdateScoreText. Аналогичен предыдущему, за исключением того, что отвечает не за здоровье игрока, а за количество набранных им очков.
12. CheckIfGameOver. При снижении здоровья игрока до нуля вызывает экран с соответствующим сообщением.

Скрипт Enemy отвечает за врагов и содержит следующие методы:

1. Start. Добавляет врагов в лист, содержащийся в скрипте GameManage.

2. Awake. Содержит ссылку на объект GameManager.

3. AttemptMove. Дочерний метод скрипта MovingObject. Определяет будет ли ходить каждый отдельно взятый враг (или сколько ходов этот враг пропустит).

4. MoveEnemy. Выделяет героя целью врага и заставляет двигаться врага в направлении героя.

5. OnCantMove. Дочерний метод скрипта MovingObject. Отвечает за атаку врагом героя.

6. TakeDamage. Следит за изменением здоровья врага при получении им урона от героя. При достижении нулевой отметки здоровья уничтожает этот объект с игрового поля.

Скрипт BoardCreator (приложение Б) отвечает за генерацию игрового уровня и всех объектов на нем. Содержит следующие классы:

1. Awake. Содержит ссылку на GameManager.

2. Start. Определяет, какой из трех типов уровней будет создан. Запускает генерацию уровня.

3. CreateRoomsAndCorridors. Создание массивов с комнатами и коридорами, генерируемыми в скриптах Room и Corridor соответственно и построение подземелья из этих массивов.

4. SetTilesValuesForRooms. Задает клеткам комнаты тип тайла пола Floor.

5. SetTilesValuesForCorridors. Задает клеткам корридора тип тайла пола Floor.

6. InstantiateTiles. Генерирует тайлы на уровне, соответствующего вида тайла (пол или стена) из массива.

7. InstantiateOuterTiles. Создает дополнительную неразрушаемую стену из тайлов вокруг игрового поля.

8. InstantiateVerticalOuterWall. Генерация тайлов на вертикальных неразрушаемых стенах.

9. InstantiateHorizontalOuterWall. Генерация тайлов на горизонтальных неразрушаемых стенах.

10. InstantiateFromArray. Выдает случайный объект или тайл из списка доступных.

11. LayoutObjectsAtRandom. Метод, использующийся для случайной генерации юнитов и собираемых объектов на уровне.

12. CreateEnemies. С помощью метода LayoutObjectsAtRandom генерирует какие враги и в каком количестве появятся на уровне.

13. CreatePotions. С помощью метода LayoutObjectsAtRandom генерирует какие зелья и в каком количестве появятся на уровне.

14. CreateGold С помощью метода LayoutObjectsAtRandom генерирует каком количестве золото появится на уровне.

15. CreateAllObjectsInRoom. Создает врагов, зелья и золото на игровом поле.

Скрипт Corridor (приложение В) отвечает за генерацию коридоров для массива коридоров в скрипте BoardCreator.

Скрипт Room (приложение Г) отвечает за генерацию комнат для массива комнат в скрипте BoardCreator.

Скрипт Wall отвечает за отображение тайлов поврежденной стены при повреждении стены игроком.

Скрипт IntRange генерирует случайные диапазоны чисел, использующиеся для генерации игрового поля.

Скрипт MovingObject содержит родительские классы, связанные с перемещением объектов и использующиеся в скриптах Player и Enemy.

Скрипт PlayerCam отвечает за наблюдением камеры за персонажем игрока. При помощи данного скрипта герой всегда находится в центре экрана.

Скрипт SoundManager отвечает за воспроизведение звуков и музыки во время игры.

Скрипт PauseMenu отвечает за срабатывание кнопок в меню паузы.

Скрипт MainMenu отвечает за срабатывание кнопок в главном меню.

2.3.4 Этап тестирования

После разработки и сборки проекта необходимо провести этап тестирования.

Данный этап необходим для проверки работоспособности проекта и выявления различных ошибок.

Также нужно предоставить возможность другим людям ознакомиться с проектом. Это позволит протестировать удобство и понятность интерфейса для обычных игроков и поможет выявить другие, ранее не найденные ошибки.

При самостоятельном тестировании были выявлены следующие ошибки:

- анимации удара и получения урона у объекта Player вместо однократного срабатывания зацикливаются и не прекращаются;
- объекты двух из трех врагов были развернуты в противоположные их движению стороны;
- при переходе на другой уровень сбрасываются данные об игроке (количество жизней, очков и т.д.);
- при первом убийстве врага игра зависала;
- игра не считала количество очков за убийство врагов;
- генерация двух объектов Player и Exit;
- прочие мелкие ошибки во время генерации игрового уровня.

Все ошибки были исправлены и при последующих запусках не были обнаружены.

После чего продукт был передан для тестирования нескольким людям для оценки удобства и понятности интерфейса. В тестировании принимали участие люди как хорошо разбирающиеся в играх и постоянно в них играющие, так и те, кто почти или совсем не играет в компьютерные игры.

В ходе тестирования все люди быстро разобрались в управлении и основных механиках игры. Дополнительных ошибок в ходе тестирования другими людьми обнаружено не было.

2.3.5 Запаковка и выпуск игры

Финальным этапом разработки компьютерной игры является ее запаковка. Во время подготовки к сборке игры необходимо очистить проект от ненужных файлов — это необходимо для того, чтобы итоговый продукт занимал меньше места на жестком диске.

После очистки проекта необходимо открыть настройки проекта Build Settings (рисунок 38). В окне настроек проекта необходимо выбрать сцены, которые будут в итоговой игре и желаемую платформу. В данном случае — Windows.

Остается лишь нажать кнопку Build, после чего движок начнет сборку проекта. Можно нажать кнопку Build And Run и тогда проект запустится сразу же после сборки.

После завершения сборки будет доступен готовый продукт, который можно запустить с помощью файла с разрешением .exe.

Не стоит забывать, что для некоторых платформ будет нужно дополнительно доработать проект. Например, для платформ с сенсорным управлением (телефоны или планшеты) необходимо адаптировать проект под сенсорное управление. Но не смотря на это, возможность адаптации на другие платформы является неоспоримым плюсом.

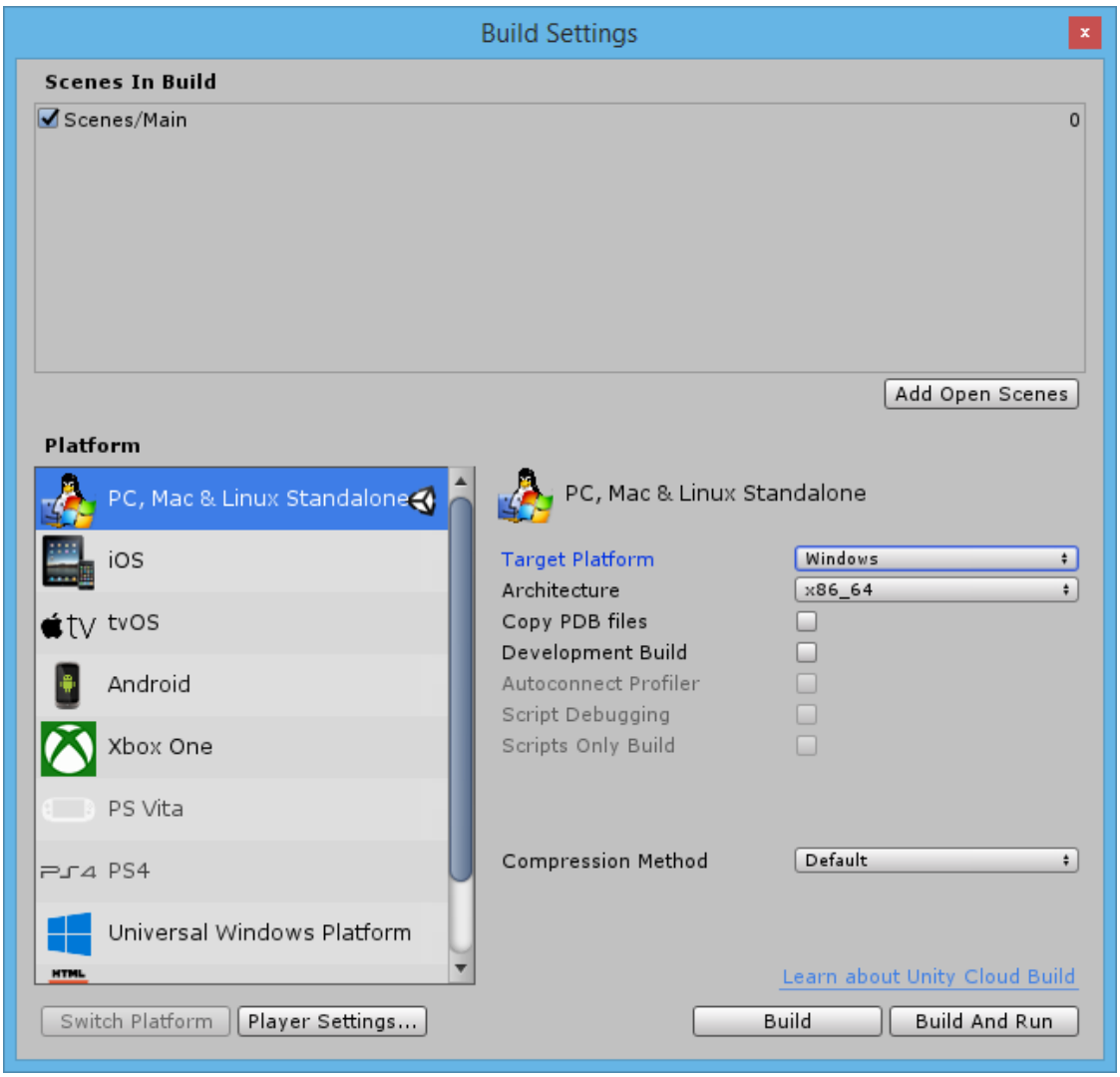

Рисунок 38 — Окно настройки проекта

2.4 Технические требования к проекту

Для запуска игры нужен компьютер или ноутбук со следующими минимальными системными требованиями:

- операционная система: Windows Vista SP1+;
- процессор: любой, поддерживающий набор инструкций SSE2;
- видеокарта: любая, с поддержкой DirectX 10;
- место на жестком диске: 200 Мб.

2.5 Калькуляция проекта

Во время разработки было разработано некоторое количество элементов, представленных в таблице 1.

Таблица 1 — Количество созданных элементов с описанием

Элемент	Количество	Комментарий
Сцены	2	Сцена меню и сцена игры
Тайлы уровней	10	3 набора тайлов (пол, стена, поврежденная стена) для трех вариантов уровня и 1 тайл перехода на следующий уровень
Персонаж	1	Содержит 3 анимации — анимация простоя (5 кадров), анимация атаки (3 кадра) и анимация получения урона (3 кадра)
Враги	3	3 врага по 2 анимации у каждого — анимация простоя (3-5 кадров), анимация атаки (3-4 кадра)
Собираемые объекты	4	3 спрайта элементов (золото, большое и маленькое зелья) и кубок, обладающий своей анимацией (4 кадра)
Скрипты	13	13 скриптов с общим количеством строк 1351
Элементы интерфейса	6	3 элемента для отображения параметров игрока (количество очков, жизней, уровня и опыта персонажа), 1 экран смены уровня, 2 меню (пауза и главное меню)
Прочие элементы	1	Фон для меню

Времени потрачено:

- на работу с графикой: около 30 часов;
- на программирование скриптов: около 200 часов;
- на тестирование и исправление ошибок: около 20 часов.

ЗАКЛЮЧЕНИЕ

Инди-разработка в наши дни являются одним из популярных и перспективных направлений работы.

Во время выполнения дипломной работы было произведено исследование понятия «компьютерная игра» и различных представителей компьютерных игр жанра roguelike, а также различных технологий и сред разработки компьютерных игр. В ходе исследования были выявлены особенности, присущие жанру roguelike и изучен функционал игрового движка Unity.

Были отрисованы необходимые тайлы и спрайты. С помощью движка Unity и Visual Studio были написаны необходимые скрипты.

Готовый продукт был протестирован на предмет ошибок, выявленные ошибки были исправлены.

В результате выполнения работы цель — разработка компьютерной игры с помощью средств разработки компьютерных игр была выполнена.

В дальнейшем предполагается разнообразить геймплей: увеличить количество врагов и добавить боссов, усовершенствовать систему прокачки персонажа, боевую систему и систему процедурной генерации уровней.

Не стоит забывать, что игровой движок Unity является мультиплатформенным, а это значит, что разработанный продукт в любое время можно адаптировать практически под любую платформу из списка поддерживаемых движком платформ.

Разработка собственного проекта позволила получить большое количество опыта в различных областях, связанных с разработкой компьютерных игр, а именно: разработка концепции игры, разработка игровых механик, программирование и реализация игровой логики, сборка и настройка проектов в среде разработки.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Жанр Roguelike / Персональный блог имени меня ^_^ [Электронный ресурс]. — Режим доступа: https://stopgame.ru/blogs/topic/48196 (дата обращения: 10.05.2018).
2. Значение слова ЖАНР. Что такое ЖАНР? [Электронный ресурс]. — Режим доступа: https://kartaslov.ru/значение-слова/жанр (дата обращения: 09.05.2018).
3. ИГРА это что такое ИГРА: определение — Педагогика.НЭС [Электронный ресурс]. — Режим доступа: http://didacts.ru/termin/igra.html (дата обращения: 09.05.2018).
4. Каталог API (Microsoft) и справочных материалов [Электронный ресурс]. — Режим доступа: https://msdn.microsoft.com/ru-ru/library/ (дата обращения: 10.03.2018).
5. Кодировка ASCII [Электронный ресурс]. — Режим доступа: http://wm-school.ru/html/html_ascii_codes.html (дата обращения: 10.05.2018).
6. Компьютерная игра — Циклопедия [Электронный ресурс]. — Режим доступа: http://cyclowiki.org/wiki/Компьютерная_игра/ (дата обращения: 09.05.2018).
7. Лучший игровой движок по версии пользователей хабра [Электронный ресурс]. — Режим доступа: https://habr.com/post/307952/ (дата обращения: 12.05.2018).
8. Рогалики — Lurkmore [Электронный ресурс]. — Режим доступа: https://lurkmore.to/Рогалики (дата обращения: 10.05.2018).
9. Сеттинг — ролевая энциклопедия [Электронный ресурс]. — Режим доступа: http://wiki.rpgverse.ru/wiki/Сеттинг (дата обращения: 09.05.2018).
10. Уровень 5. Механика и динамика — Аус Хестов [Электронный ресурс]. — Режим доступа: (дата обращения: 10.05.2018).

11. Что такое Инди-игры? Значение термина Инди-игры [Электронный ресурс]. — Режим доступа: https://animatika.ru/info/gloss/indie-games.html (дата обращения: 10.05.2018).

12. 2D игры: тайловая графика и почему ее применяют до сих пор [Электронный ресурс]. — Режим доступа: https://cadelta.ru/games/id800 (дата обращения: 12.05.2018).

13. 2D Roguelike tutorial — Unity [Электронный ресурс]. — Режим доступа: https://unity3d.com/ru/learn/tutorials/s/2d-roguelike-tutorial (дата обращения: 10.03.2018).

14. 63 Best roguelikes/roguelites on PC as of 2018 — Slant [Электронный ресурс]. — Режим доступа: https://www.slant.co/topics/1657/~roguelikes-roguelites-on-pc (дата обращения: 12.05.2018).

15. Buy Adobe Photoshop CC | Best photo, image, and design editing software [Электронный ресурс]. — Режим доступа: https://www.adobe.com/products/photoshop.html?promoid=MLR7SFRV&mv=other (дата обращения: 11.05.2018).

16. DRL - D**m, the Roguelike [Электронный ресурс]. — Режим доступа: https://drl.chaosforge.org/ (дата обращения: 10.05.2018).

17. Econ Dude: Что такое тайлы? Тайловая графика в играх [Электронный ресурс]. — Режим доступа: https://www.econdude.pw/2017/03/tiles.html (дата обращения: 12.05.2018).

18. Elona: A Special Roguelike [Электронный ресурс]. — Режим доступа: https://www.reddit.com/r/Elona/ (дата обращения: 11.05.2018).

19. Game Engines. How do they work? [Электронный ресурс]. — Режим доступа: https://www.giantbomb.com/profile/michaelenger/blog/game-engines-how-do-they-work/101529/ (дата обращения: 10.05.2018).

20. GDC 2016: CryEngine 5 vs Unreal Engine 4 vs Unity 5 • Sgamers [Электронный ресурс]. — Режим доступа: https://sgamers.ru/novosti/gdc-2016-cryengine-5-vs-unreal-engine-4-vs-unity-5.html (дата обращения: 12.05.2018).

21. Hardware & Software Specifications [Электронный ресурс]. — Режим доступа: https://docs.unrealengine.com/en-us/GettingStarted/RecommendedSpecifications (дата обращения: 12.05.2018).

22. Lost Labyrinth ~ The best Roguelike on the Net [Электронный ресурс]. — Режим доступа: https://www.lostlabyrinth.com/index.php?p=start (дата обращения: 10.05.2018).

23. PAUSE MENU in Unity [Электронный ресурс]: видеоролик. — Режим доступа: https://www.youtube.com/watch?v=JivuXdrIHK0 (дата обращения: 31.05.2018).

24. Pixel Dungeon [Электронный ресурс]. — Режим доступа: http://pixeldungeon.watabou.ru/ (дата обращения: 11.05.2018).

25. POWDER [Электронный ресурс]. — Режим доступа: http://www.zincland.com/powder/?pagename=about (дата обращения: 10.05.2018).

26. Rogue (1984-DOS). Ссылки, описание, обзоры, скриншоты, видеоролики на Old-Games.RU [Электронный ресурс]. — Режим доступа: https://www.old-games.ru/game/1206.html (дата обращения: 11.05.2018).

27. RogueBasin [Электронный ресурс]. — Режим доступа: http://roguebasin.com/index.php?title=Main_Page (дата обращения: 28.05.2018).

28. START MENU in Unity [Электронный ресурс]: видеоролик. — Режим доступа: https://www.youtube.com/watch?v=zc8ac_qUXQY (дата обращения: 28.05.2018).

29. System Requirements — CRYENGINE V Manual — Documentation [Электронный ресурс]. — Режим доступа: http://docs.cryengine.com/display/CEMANUAL/System+Requirements (дата обращения: 12.05.2018).

30. Unity — System Requirements [Электронный ресурс]. — Режим доступа: https://unity3d.com/ru/unity/system-requirements (дата обращения: 12.05.2018).

ПРИЛОЖЕНИЕ А

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Российский государственный профессионально-педагогический университет»

Институт инженерно-педагогического образования
Кафедра информационных систем и технологий
Направление подготовки 09.03.02 Информационные системы и технологии
Профиль подготовки «Информационные технологии в медиаиндустрии»

УТВЕРЖДАЮ
Заведующий кафедрой
_____ Н.С. Толстова
подпись и.о. фамилия
« ___ » _____ 2018 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра

студента (ки) *4* курса группы *ИТм-402*
Демина Владислава Вячеславовича
фамилия, имя, отчество полностью

1. Тема ***2D игра в жанре roguelike***

утверждена распоряжением по институту от «_____» _____ 20 г. № ___

2. Руководитель ***Садчиков Илья Александрович***
фамилия, имя, отчество полностью

_____ _____ *ст. преподаватель* *РГППУ*
ученая степень ученое звание должность место работы

3. Место преддипломной практики *РГППУ*

4. Исходные данные к ВКР *Шрифт Press Start 2P, звуки с сайта freesound.com, музыка с сайта opengameart.org*

5. Содержание текстовой части ВКР (перечень подлежащих разработке вопросов) *Анализ представителей жанра roguelike и выявление особенностей жанра, анализ сред разработки компьютерных игр. Разработка концепта игры и реализация этого концепта с помощью сред разработки компьютерных игр.*

6. Перечень демонстрационных материалов *презентация выполненная в MS Power Point, обзорный видеоролик с геймплеем разработанной игры, 2D-игра в жанре roguelike*

7. Календарный план выполнения выпускной квалификационной работы

№ п/п	Наименование этапа дипломной работы	Срок выполнения этапа	Процент выполнения ВКР	Отметка руководителя о выполнении
1	Сбор информации по выпускной квалификационной работе	23.04.2018	10%	подпись
2	Выполнение работ по разрабатываемым вопросам и их изложение в пояснительной записке:	03.05.2015	60%	подпись
2.1	Анализ жанра roguelike и его представителей и выявление особенностей жанра	04.05.2014	5%	подпись
2.2	Анализ средств разработки компьютерных игр и изучение их функционала	05.05.2014	10%	подпись
2.3	Разработка графической составляющей игры	07.05.2014	10%	подпись
2.4	Программирование скриптов для игры	08.05.2014	30%	подпись
2.5	Тестирование и исправление ошибок	14.05.2014	5%	подпись
3	Оформление текстовой части ВКР	15.05.2018	10%	подпись
4	Выполнение демонстрационных материалов к ВКР	01.06.2018	10%	подпись
5	Нормоконтроль	06.06.2018	5%	подпись
6	Подготовка доклада к защите в ГЭК	13.06.2018	5%	подпись

8. Консультанты по разделам выпускной квалификационной работы

Наименование раздела	Консультант	Задание выдал		Задание принял	
		подпись	дата	подпись	дата

Руководитель _____ _____ *Задание получил* _____

подпись дата

подпись студента дата

9. Дипломная работа и все материалы проанализированы.

Считаю возможным допустить ***Демина В.В.*** к защите выпускной квалификационной работы в государственной экзаменационной комиссии.

Руководитель _____ _____

подпись дата

10. Допустить ***Демина В.В.*** к защите выпускной квалификационной работы

фамилия и. о. студента

в государственной экзаменационной комиссии (протокол заседания кафедры от «_____» _____ 20_____г., № _____)

Заведующий кафедрой _____ _____

подпись дата

ПРИЛОЖЕНИЕ Б

Скрипт BoardCreator

```
using System.Collections;
using UnityEngine;
using System.Collections.Generic;

public class BoardCreator : MonoBehaviour
{
    public enum TileType //типы тайлов, которые будут положены в том или ином месте.
    {
        Wall, Floor,
    }

    public int columns = 100; //длина и высота игрового поля
    public int rows = 100;
    public IntRange numRooms = new IntRange(15, 20); //диапазон создаваемых комнат
    public IntRange roomWidth = new IntRange(3, 10);  //диапазон ширин комнаты
    public IntRange roomHeight = new IntRange(3, 10); //диапазон высот комнаты
    public IntRange corridorLength = new IntRange(6, 10); //диапазон длины коридоров
    public GameObject[] floorTiles; //массив доступных тайлов пола
    public GameObject[] wallTiles; //массив доступных тайлов стен
    public GameObject[] outerWallTiles; //массив доступных тайлов наружных стен
    public GameObject[] enemyTiles; //массив доступных тайлов врагов
    public GameObject[] foodTiles; //массив доступных тайлов еды
    public GameObject[] gold; //массив доступных тайлов золота
    public GameObject cup; //объект кубок
    public GameObject player; //объект
    public GameObject exit; //объект
    public GameManager gameManage;
    public int randomIndex;

    private TileType[][] tiles; //массив для определения типа тайла в каждой ячейке
    private Room[] rooms; //массив для комнат
    private Corridor[] corridors; //массив для коридоров
    private GameObject boardHolder; //объект, хранит в себе копии объектов во время игры
    private List<Vector3> gridPositions = new List<Vector3>();

    void Awake()
    {
        gameManage = GetComponent<GameManager>(); //ссылка на объект GameManager
    }

    private void Start()
    {
        randomIndex = Random.Range(0, 3); // выдает какой тип уровня будет создан
        boardHolder = new GameObject("BoardHolder");
        SetupTilesArray();      //запуск всех методов в скрипте
        CreateRoomsAndCorridors();
        SetTilesValuesForRooms();
        SetTilesValuesForCorridors();
        InstantiateTiles();
        InstantiateOuterWalls();
        CreateAllObjectsInRoom();
    }


    void SetupTilesArray()  //заполнение массива ячейками для дальнейшего создания тайлов
    {
        tiles = new TileType[columns][];
```

```
        for (int i = 0; i < tiles.Length; i++)
        {
            tiles[i] = new TileType[rows];
        }
    }


    void CreateRoomsAndCorridors()
    {
            rooms = new Room[numRooms.Random]; //массив комнат случ. размера из доступных
            corridors = new Corridor[rooms.Length - 1]; //массив коридоров на 1 меньше
            rooms[0] = new Room();//первая комната
            corridors[0] = new Corridor();//первый коридор из нее
            rooms[0].SetupRoom(roomWidth, roomHeight, columns, rows);
            corridors[0].SetupCorridor(rooms[0], corridorLength, roomWidth, roomHeight,
columns, rows, true);

            for (int i = 1; i < rooms.Length; i++)//создание последующих комнат
            {
                rooms[i] = new Room();
                rooms[i].SetupRoom(roomWidth, roomHeight, columns, rows, corridors[i -
1]);
                if (i < corridors.Length) //создание коридоров пока не кончатся комнаты
                {
                    corridors[i] = new Corridor();
                    corridors[i].SetupCorridor(rooms[i], corridorLength, roomWidth, room-
Height, columns, rows, false);
                }

                if (i == 1) //создание игрока в первой комнате
                {
                    Vector3 playerPos = new Vector3(rooms[i].xPos, rooms[i].yPos, 0);
                    Instantiate(player, playerPos, Quaternion.identity);
                }

                if (i == rooms.Length - 1 && gameManage.level != 10) //выхода в последней
                {
                    Vector3 exitPos = new Vector3(rooms[i].xPos, rooms[i].yPos, 0);
                    Instantiate(exit, exitPos, Quaternion.identity);
                }
                else if (i == rooms.Length - 1 && gameManage.level == 10)
                {                           //создание кубка на последнем уровне
                    Vector3 cupPos = new Vector3(rooms[i].xPos, rooms[i].yPos, 0);
                    Instantiate(cup, cupPos, Quaternion.identity);
                }
            }
        }


    void SetTilesValuesForRooms()          //установка типов тайлов в комнатах
    {
        for (int i = 0; i < rooms.Length; i++) //для каждой комнаты
        {
            Room currentRoom = rooms[i];
            for (int j = 0; j < currentRoom.roomWidth; j++) //через всю ширину
            {
                int xCoord = currentRoom.xPos + j;
                for (int k = 0; k < currentRoom.roomHeight; k++)  //для каждого гориз.
                {
                    int yCoord = currentRoom.yPos + k;    //идти вверх через всю комнату
                    tiles[xCoord][yCoord] = TileType.Floor;
                }
            }
        }
    }
```

```
void SetTilesValuesForCorridors()       //установка типа тайла для корридоров
{
    for (int i = 0; i < corridors.Length; i++)  //для каждого корридора
    {
        Corridor currentCorridor = corridors[i];

        for (int j = 0; j < currentCorridor.corridorLength; j++) //пройти его длину
        {
            int xCoord = currentCorridor.startXPos;
            int yCoord = currentCorridor.startYPos;

            switch (currentCorridor.direction)    //в зависимости от направления
            {                                     //двигаться в соответств. Направ.
                case Direction.North:
                    yCoord += j;
                    break;
                case Direction.East:
                    xCoord += j;
                    break;
                case Direction.South:
                    yCoord -= j;
                    break;
                case Direction.West:
                    xCoord -= j;
                    break;
            }
            tiles[xCoord][yCoord] = TileType.Floor;
        }
    }
}


void InstantiateTiles()     //отрисовка тайлов на уровне в соответствии с типом
{
    for (int i = 0; i < tiles.Length; i++)
    {
        for (int j = 0; j < tiles[i].Length; j++)
        {
            InstantiateFromArray(floorTiles, i, j, randomIndex);

            if (tiles[i][j] == TileType.Wall)
            {
                InstantiateFromArray(wallTiles, i, j, randomIndex);
            }
            else if (tiles[i][j] == TileType.Floor)
            {
                gridPositions.Add(new Vector3(i, j));
            }
        }
    }
}


void InstantiateOuterWalls()           //отрисовка тайлов внешних стен(вокруг уровня)
{                                      //при помощи двух следующих методов
    float leftEdgeX = -1f;
    float rightEdgeX = columns + 0f;
    float bottomEdgeY = -1f;
    float topEdgeY = rows + 0f;

    InstantiateVerticalOuterWall(leftEdgeX, bottomEdgeY, topEdgeY);
    InstantiateVerticalOuterWall(rightEdgeX, bottomEdgeY, topEdgeY);
```

```
        InstantiateHorizontalOuterWall(leftEdgeX + 1f, rightEdgeX - 1f, bottomEdgeY);
        InstantiateHorizontalOuterWall(leftEdgeX + 1f, rightEdgeX - 1f, topEdgeY);
    }


    void InstantiateVerticalOuterWall(float xCoord, float startingY, float endingY)
    {                                       //метод для вертикальных стен
        float currentY = startingY;

        while (currentY <= endingY)
        {
            InstantiateFromArray(outerWallTiles, xCoord, currentY, randomIndex);
            currentY++;
        }
    }


    void InstantiateHorizontalOuterWall(float startingX, float endingX, float yCoord)
    {                                       //метод для горизонтальных стен
        float currentX = startingX;

        while (currentX <= endingX)
        {
            InstantiateFromArray(outerWallTiles, currentX, yCoord, randomIndex);
            currentX++;
        }
    }


    void InstantiateFromArray(GameObject[] prefabs, float xCoord, float yCoord, int i)
           //отрисовка объектов из массива
    {
        Vector3 position = new Vector3(xCoord, yCoord, 0f);
        GameObject tileInstance = Instantiate(prefabs[i], position, Quaternion.identity)
as GameObject;
        tileInstance.transform.parent = boardHolder.transform; //помещение объектов в
    }                                                           //BoardHolder

    Vector3 RandomPosition()     //генерация случайной позиции на поле для объектов
    {
        int randomIndex = Random.Range(0, gridPositions.Count);
        Vector3 randomPosition = gridPositions[randomIndex];
        gridPositions.RemoveAt(randomIndex);
        return randomPosition;
    }

    void LayoutObjectsAtRandom(GameObject[] tileArray, int minimum, int maximum)
    {                 //генерация случайных объектов для каждого типа
        int objectCount = Random.Range(minimum, maximum + 1);
        for (int i = 0; i < objectCount; i++)
        {
            Vector3 randomPosition = RandomPosition();
            GameObject tileChoice = tileArray[Random.Range(0, tileArray.Length)];
            GameObject tileCreated = Instantiate(tileChoice, randomPosition, Quaterni-
on.identity) as GameObject;
            tileCreated.transform.parent = boardHolder.transform;
        }
    }

    void CreateEnemies(int level) //создание случайных врагов из доступного списка с
    {                 //количеством, заданном диапазоном, которые были сгенерированы
        LayoutObjectsAtRandom(enemyTiles, 7, 10);
    }
```

```
    void CreatePotion()//создание случайных зелий врагов из доступного списка с
    {                  //количеством, заданном диапазоном, которые были сгенерированы
        LayoutObjectsAtRandom(potionTiles, 5, 10);
    }

    void CreateGold() //создание золота врагов из доступного списка с
    {                 //количеством, заданном диапазоном, которые были сгенерированы
        LayoutObjectsAtRandom(gold, 5, 10);
    }

    void CreateAllObjectsInRoom()//объединение всех методов для метода Start
    {
        CreateEnemies(gameManage.level);
        CreateFood();
        CreateGold();
    }
}
```

ПРИЛОЖЕНИЕ В

Скрипт Corridor

```
using UnityEngine;

public enum Direction //список доступных направлений
{
    North, East, South, West,
}

public class Corridor
{
    public int startXPos; //X координата для левого нижнего тайла коридора
    public int startYPos; //У координата для левого нижнего тайла коридора
    public int corridorLength;   //длина коридора
    public Direction direction; //направление коридора

    public int EndPositionX //определение конечной X координаты коридора
    {
        get
        {
            if (direction == Direction.North || direction == Direction.South)
                return startXPos;
            if (direction == Direction.East)
                return startXPos + corridorLength - 1;
            return startXPos - corridorLength + 1;
        }
    }


    public int EndPositionY //определение конечной У координаты коридора
    {
        get
        {
            if (direction == Direction.East || direction == Direction.West)
                return startYPos;
            if (direction == Direction.North)
                return startYPos + corridorLength - 1;
            return startYPos - corridorLength + 1;
        }
    }


    public void SetupCorridor(Room room, IntRange length, IntRange roomWidth, IntRange
roomHeight, int columns, int rows, bool firstCorridor) //создание коридора

    {
        direction = (Direction)Random.Range(0, 4); //определение направления

        Direction oppositeDirection = (Direction)(((int)room.enteringCorridor + 2) % 4);
                                                    //обратное направление
        if (!firstCorridor && direction == oppositeDirection)
        {  //если не первый коридор и выбрано случ. направление, повернуть на 90 град.
            int directionInt = (int)direction;
            directionInt++;
            directionInt = directionInt % 4;
            direction = (Direction)directionInt;
        }
```

```
        corridorLength = length.Random; //задание случайной длины коридора
        int maxLength = length.m_Max;

        switch (direction) //для разных направлений разные параметры
        {
            case Direction.North:
                startXPos = Random.Range(room.xPos, room.xPos + room.roomWidth - 1);
                startYPos = room.yPos + room.roomHeight;
                maxLength = rows - startYPos - roomHeight.m_Min;
                break;

            case Direction.East:
                startXPos = room.xPos + room.roomWidth;
                startYPos = Random.Range(room.yPos, room.yPos + room.roomHeight - 1);
                maxLength = columns - startXPos - roomWidth.m_Min;
                break;

            case Direction.South:
                startXPos = Random.Range(room.xPos, room.xPos + room.roomWidth);
                startYPos = room.yPos;
                maxLength = startYPos - roomHeight.m_Min;
                break;

            case Direction.West:
                startXPos = room.xPos;
                startYPos = Random.Range(room.yPos, room.yPos + room.roomHeight);
                maxLength = startXPos - roomWidth.m_Min;
                break;
        }
        corridorLength = Mathf.Clamp(corridorLength, 1, maxLength); //чтоб длина коридора
    }                                                               //не вышла за пределы уровня
}
```

ПРИЛОЖЕНИЕ Г

Скрипт Room

```
using UnityEngine;

public class Room
{
    public int xPos; //левая нижняя X координата комнаты
    public int yPos; //левая нижняя У координата
    public int roomWidth; //ширина комнаты
    public int roomHeight; //высота
    public Direction enteringCorridor; //с какой стороны коридор в нее входит

    public void SetupRoom(IntRange widthRange, IntRange heightRange, int columns, int
rows) //метод для стартовой комнаты, создается примерно в середине поля
    {
        roomWidth = widthRange.Random;
        roomHeight = heightRange.Random;

        xPos = Mathf.RoundToInt(columns / 2f - roomWidth / 2f);
        yPos = Mathf.RoundToInt(rows / 2f - roomHeight / 2f);
    }

    public void SetupRoom(IntRange widthRange, IntRange heightRange, int columns, int
rows, Corridor corridor) //метод для остальных комнат
    {
        enteringCorridor = corridor.direction; //входящий коридор

        roomWidth = widthRange.Random; //случайные высота и ширина комнаты
        roomHeight = heightRange.Random;

        switch (corridor.direction) //в зависимости от направления входящего коридора
        {             //настройки комнат (чтоб не выйти за пределы уровня)
            case Direction.North:
                roomHeight = Mathf.Clamp(roomHeight, 1, rows - corridor.EndPositionY);

                yPos = corridor.EndPositionY;
                xPos = Random.Range(corridor.EndPositionX - roomWidth + 1, corri-
dor.EndPositionX);
                xPos = Mathf.Clamp(xPos, 0, columns - roomWidth);
                break;

            case Direction.East:
                roomWidth = Mathf.Clamp(roomWidth, 1, columns - corridor.EndPositionX);

                xPos = corridor.EndPositionX;
                yPos = Random.Range(corridor.EndPositionY - roomHeight + 1, corri-
dor.EndPositionY);
                yPos = Mathf.Clamp(yPos, 0, rows - roomHeight);
                break;

            case Direction.South:
                roomHeight = Mathf.Clamp(roomHeight, 1, corridor.EndPositionY);

                yPos = corridor.EndPositionY - roomHeight + 1;
                xPos = Random.Range(corridor.EndPositionX - roomWidth + 1, corri-
dor.EndPositionX);
                xPos = Mathf.Clamp(xPos, 0, columns - roomWidth);
                break;
```

```
            case Direction.West:
                roomWidth = Mathf.Clamp(roomWidth, 1, corridor.EndPositionX);
                xPos = corridor.EndPositionX - roomWidth + 1;

                yPos = Random.Range(corridor.EndPositionY - roomHeight + 1, corri-
dor.EndPositionY);
                yPos = Mathf.Clamp(yPos, 0, rows - roomHeight);
                break;
        }
    }
}
```