**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ**

**Кафедра веб-технологий и компьютерного моделирования**

**Создание игры в жанре Roguelike**

Курсовой проект

Красина Федора Тимуровича, студента 3 курса, специальность “веб-программирование”

Научный руководитель:

Д. В. Филимонов

Минск, 2022

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ](#_30j0zll) 3

ГЛАВА 1. СОВРЕМЕННЫЕ ТЕНДЕНЦИИ В РАЗРАБОТКЕ ИГР. ОБЗОР ПРЕДСТАВИТЕЛЕЙ ЖАНРА “ROGUELIKE”. ИНСТРУМЕНТЫ РАЗРАБОТКИ (ЯЗЫК C#, UNITY, ZENJECT)

1.1. Современные тенденции в разработке игр. 4

1.2. Жанр “Roguelike”. 7

1.3. Игровой движок Unity. 9

1.4. Язык программирования C#. 11

1.5. Фреймворк Zenject 12

ГЛАВА 2. РАЗРАБОТКА СПОСОБА РЕАЛИЗАЦИИ ПРОЦЕДУРНОЙ ГЕНЕРАЦИИ УРОВНЯ. РАЗБОР АЛГОРИТМОВ И РЕАЛИЗАЦИЯ ПРОЦЕДУРНОЙ ГЕНЕРАЦИИ УРОВНЯ. ДОБАВЛЕНИЕ ГРАФИЧЕСКИХ РЕСУРСОВ

2.1. Разработка способа реализации процедурной генерации уровня. 13

2.2. Разбор алгоритмов и реализация процедурной генерации уровня. 15

2.3. Добавление графических ресурсов. 20

ЗАКЛЮЧЕНИЕ 25

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 26

# 

**ВВЕДЕНИЕ**

Создание компьютера и развитие цифровых технологий повлекло за собой необратимое изменение образа жизни человека. Сначала компьютеры использовались лишь для вычисления сложных математических задач, однако с течением времени они стали все больше влиять на ежедневную деятельность людей: поиск информации, хранение информации, общение, обучение и многое другое.

Одной из таких областей использования компьютера стал досуг и развлечение. Мы каждый день используем современные технологии для отдыха и для смены деятельности: смотрим кино, слушаем музыку, читаем книги, а также играем в компьютерные игры.

Разработка игр в настоящее время является одной из наиболее развитых областей IT: на сегодняшний день, игры - это одно из самых массовых применений компьютера.

Первые игры отличались простотой идеи, механики и реализации, однако с течением времени технологии стали развиваться, а потребности пользователя расти. Современные игры отличаются высоким качеством графики, сюжетом, а также сложной механикой, способной заинтересовать пользователя.

Современные игры могут занимать десятки гигабайтов, они требуют высокой производительности компьютера.

Разработка новых игр, в частности в жанре головоломка, представляет собой актуальную задачу.

Цель данной работы – реализация алгоритмов, используемых при создании игр в жанре “Roguelike” и разработка процедурно генерируемого уровня в Unity.

Для реализации поставленной цели необходимо решить следующие задачи:

1. Проанализировать современные тенденции в разработке игр.
2. Разработать способ реализации процедурно генерируемого уровня.
3. Разобрать алгоритмы, используемые при разработке игры.
4. Реализовать алгоритмы для генерации и механику игры с помощью движка Unity и языка C#.

**ГЛАВА 1. СОВРЕМЕННЫЕ ТЕНДЕНЦИИ В РАЗРАБОТКЕ ИГР. ОБЗОР ПРЕДСТАВИТЕЛЕЙ ЖАНРА “ROGUELIKE”. ИНСТРУМЕНТЫ РАЗРАБОТКИ (ЯЗЫК C#, UNITY, ZENJECT)**

**1.1. Современные тенденции в разработке игр**

Еще пару десятилетий назад любители компьютерных игр воспринимались как отдельная часть социума, состоящая преимущественно из детей и подростков. Однако с течением времени молодое население выросло, а технологии развились настолько, что мощность вычислительной техники тех дней не идет ни в какое сравнение с современной. Люди все больше времени проводят за такими девайсами, как смартфон, ноутбук, персональный компьютер, игровая консоль и другими. И, конечно, определенную часть этого времени люди играют в игры.

Таким образом, мы можем разделить все игры на две крупные составляющие: мобильные и не мобильные.

**Мобильные игры**

В последние несколько лет заметен бурный рост рынка мобильных игр. Одна из причин этого явления – пандемия коронавируса и всеобщий локдаун. Но также важными факторами являются мощность современных мобильных устройств и удобство такого формата гейминга.



*Рис. 1.1 – рост рынка игр за 2012-2021 гг.*

Лидирующим жанром среди всех доступных на мобильных устройствах можно назвать Hyper Casual.

Гипер-казуальные игры – это простые игры с элементарной механикой, в которых происходит мгновенный переход к геймплею. Дословно «нажми, и играй». Благодаря своей фундаментальной «простоте» гипер-казуальные игры не только легки в прохождении, но в них хочется играть снова и снова.

Комбинация простой механики с минималистичным UI обеспечивает доступный и невероятно увлекательный пользовательский опыт без какого либо обучения. В отличии от других жанров игроки могут мгновенно погрузиться в процесс и «подсесть» на игру.

Помимо этого игры данного жанра невероятно прибыльны для их создателей. Разработка таких игр обычно не требует больших ресурсов, в них часто присутствует огромное количество рекламы, а самое главное – людям действительно нравится в них играть.

Наиболее популярные игры последний лет в жанре Hyper Casual: Subway Surfers, Cut the Rope, Flappy Bird, Angry Birds.

Другой тенденцией индустрии мобильных игр можно назвать AR-игры. Технология дополненной реальности (AR) позволяет использовать мобильные устройства, чтобы встраивать цифровой контент в реальный мир. Таким образом, игрок может взаимодействовать с цифровыми объектами так, будто бы они находятся в реальном мире. Наиболее яркими представителями игр такого типа являются Pokemon Go, Minecraft Earth, ARZombi.

AR-приложения также используются и в других сферах, например как путеводитель по интересным местам в городе, помощь при выборе одежды, запись видео.

**Киберспорт**

Особую популярность в наши дни составляют многопользовательские игры соревновательного жанра. Чаще всего этими жанрами являются MOBA (многопользовательская онлайн боевая арена) и шутеры.

Даже не играя активно, пользователи общаются, взаимодействуют и наблюдают за другими.

Уже в 2018 году финал турнира по League of Legends смотрели 200 миллионов человек. Ежегодный турнир по игре Dota 2 “The International” собирает все больший призовой фонд, в 2021 году он составлял более 40 миллиардов долларов.

У обычных игроков появляются свои кумиры, системы турниров приобретают все более элегантный вид, крупные организации вкладывают огромные деньги в развитие этого явления – все это говорит о том, что киберспорт набирает обороты огромными темпами и в наше время его можно сравнивать с привычными всеми видами командного спорта, такими как футбол или волейбол.

Наиболее крупными киберспортивными дисциплинами можно назвать такие игры как Dota 2, Fortnite, CS:GO, Hearthstone.

**Виртуальная реальность**

Отличие виртуальной реальности от дополненной в том, что для VR-игр людям необходимо специальное оборудование в виде VR-шлема и контроллеров.

Доступность VR-гарнитур медленно, но растет, однако контент, который будет действительно популярным среди игроков пока что так и не был разработан. Это создает некий замкнутый круг: разработчики не хотят создавать качественные продукты для виртуальной реальности, поскольку в них не будут играть, а производители устройств не хотят выпускать качественные и дешевые модели, которые будут пользоваться популярностью у большого числа пользователей, потому что пользователям они незачем. Однако аналитики считают, что эту проблему позволит решить концепция так называемых метавселенных.

**Метавселенная**

Термин «метавселенная» придумал писатель-фантаст Нил Стивенсон в 1992 году в романе «Лавина» (“Snow Crash”). Метавселенная по Стивенсону — это следующая стадия развития интернета: общий цифровой мир, объединяющий «физическую», дополненную и виртуальную реальности.

Люди могут подключаться к нему в виде аватаров и делать все, что и в реальном мире: искать информацию, общаться, ходить по магазинам и работать — но при этом уйти от реальности и жить в виртуальной вселенной. Аватар человека в метавселенной может быть кем захочет и владеть чем угодно, а смерть не означает смерть в реальном мире.

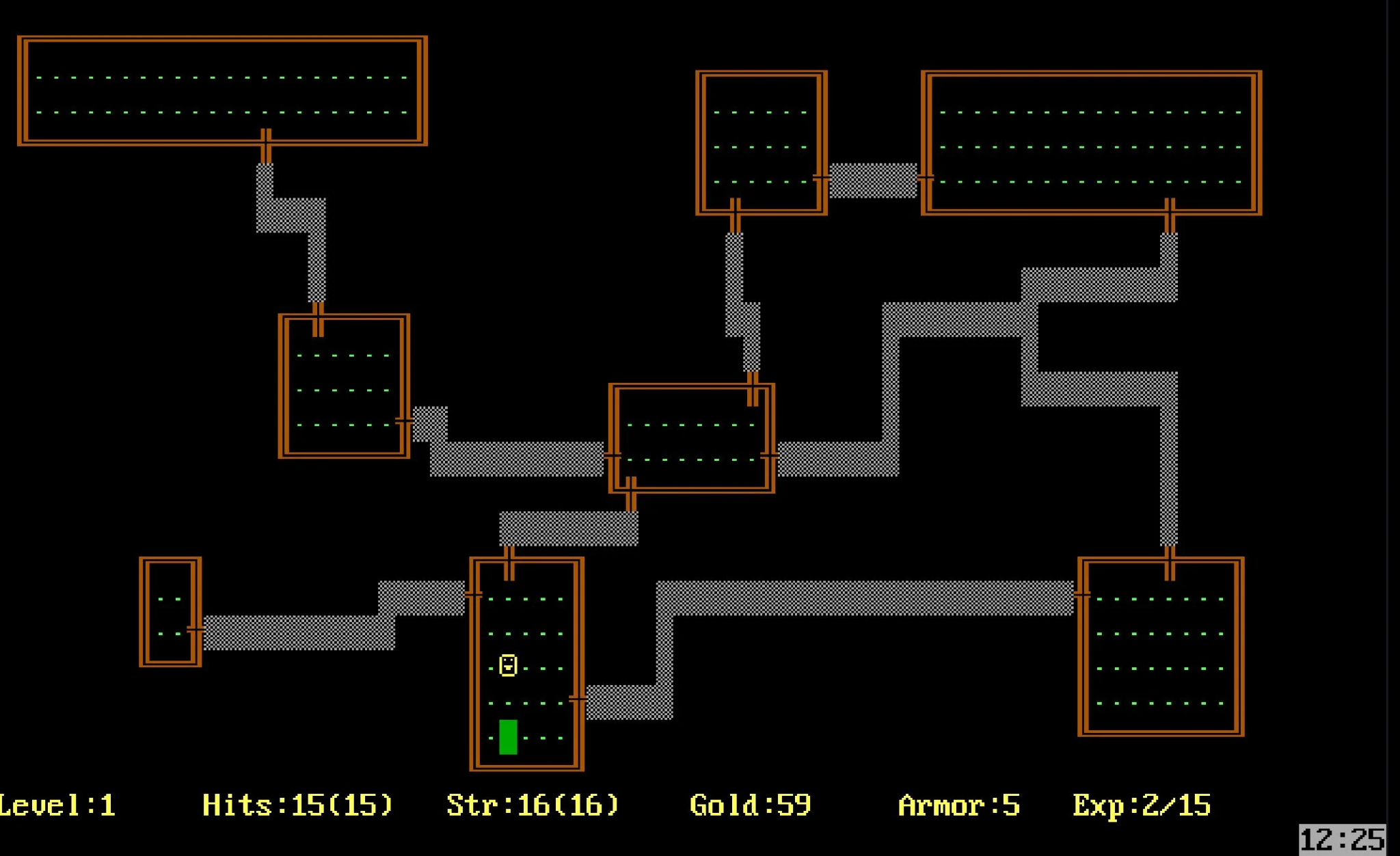
За последние несколько лет термин «метавселенная» стал сверхпопулярным. Например, в 2021 году о развитии своей метавселенной одновременно заговорили главы Microsoft, Epic Games и Facebook (ныне Meta). И каждая компания по-своему видит образ виртуальной вселенной будущего.

Метавселенная напрямую связана с другими популярными на сегодняшний день вещами: блокчейн, NFT, криптовалюта. Но поскольку все эти явления недавно начали входить в нашу жизнь, можно сказать, что идеалистическая метавселенная появится лишь спустя годы — для этого нужны одновременное развитие во многих ИТ-областях. Ее можно представить как «преемника» мобильного интернета: у пользователей будут устройства и платформы, через которые они взаимодействуют с метавселенной.

**1.2. Жанр “Roguelike”**

Roguelike (сленг “рогалик”) — жанр [компьютерных игр](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0). Характерными особенностями классического roguelike являются генерируемые случайным образом уровни, [пошаговость](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%88%D0%B0%D0%B3%D0%BE%D0%B2%D0%B0%D1%8F_%D1%82%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D0%B0) и необратимость смерти персонажа — в случае его гибели игрок не может загрузить игру и должен начать ее заново. Многие roguelike выполнены в декорациях [эпического фэнтези](https://ru.wikipedia.org/wiki/%D0%AD%D0%BF%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D1%84%D1%8D%D0%BD%D1%82%D0%B5%D0%B7%D0%B8) под сильным влиянием [настольных ролевых игр](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%81%D1%82%D0%BE%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%80%D0%BE%D0%BB%D0%B5%D0%B2%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0) наподобие [Dungeons & Dragons](https://ru.wikipedia.org/wiki/Dungeons_%26_Dragons).

Жанр восходит к игре 1980 года [Rogue](https://ru.wikipedia.org/wiki/Rogue). Хотя и до нее выходили подобные игры, такие как [Beneath Apple Manor](https://ru.wikipedia.org/w/index.php?title=Beneath_Apple_Manor&action=edit&redlink=1), именно [Rogue](https://ru.wikipedia.org/wiki/Rogue) стала образцом для бесчисленных подражаний. Подобные игры, распространявшиеся в виде [открытого кода](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), были крайне популярны в среде американских программистов и студентов в 1980-х — 1990-х годах. При наличии множества вариантов и различий между отдельными играми основные принципы геймплея roguelike оставались неизменными. В 2008 году эти консервативные принципы были описаны в рамках так называемой “Берлинской интерпретации”.



*Рис. 1.2 – игра “Rogue”.*

Ряд фактов, выделенных Берлинской интерпретацией:

* Игра должна быть [пошаговой](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%88%D0%B0%D0%B3%D0%BE%D0%B2%D0%B0%D1%8F_%D1%82%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D0%B0), каждая команда должна соответствовать одному действию и одному ходу;
* [Игровые уровни](https://ru.wikipedia.org/wiki/%D0%A3%D1%80%D0%BE%D0%B2%D0%B5%D0%BD%D1%8C_(%D0%B8%D0%B3%D1%80%D1%8B)) должны генерироваться случайным образом, будучи уникальными для каждого прохождения;
* Игра должна содержать «[перманентную смерть](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%BC%D0%B0%D0%BD%D0%B5%D0%BD%D1%82%D0%BD%D0%B0%D1%8F_%D1%81%D0%BC%D0%B5%D1%80%D1%82%D1%8C)», не позволяя игроку продолжить прохождение после гибели персонажа;
* Игра должна иметь единый режим и единый набор команд для всех игровых ситуаций, не допуская каких-либо дополнительных меню, головоломок и [мини-игр](https://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BD%D0%B8-%D0%B8%D0%B3%D1%80%D0%B0);
* Игра должна предоставлять игроку не какой-то единый линейный путь, а свободу со множеством вариантов прохождения;
* Игрок должен самостоятельно исследовать найденные предметы и открывать их свойства.

Новое обращение к жанру roguelike произошло в 2000-е годы с расцветом разработки [инди-игр](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B4%D0%B8-%D0%B8%D0%B3%D1%80%D0%B0) и появлению «roguelike-подобных игр» с нечеткой жанровой принадлежностью. Такие игры представляли собой не собственно roguelike в классическом понимании этого термина, но соединение элементов roguelike и игр других жанров.

Наиболее популярными представителями этого жанра являются The Binding Isaac, Hades, Slay the Spire, Darkest Dungeon. Другие игры, такие как [Diablo](https://ru.wikipedia.org/wiki/Diablo_(%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0)) и [UnReal World](https://ru.wikipedia.org/w/index.php?title=UnReal_World&action=edit&redlink=1) — важные вехи в эволюции жанров [Action/RPG](https://ru.wikipedia.org/wiki/Action/RPG) и [симуляторов выживания](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BC%D1%83%D0%BB%D1%8F%D1%82%D0%BE%D1%80_%D0%B2%D1%8B%D0%B6%D0%B8%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) соответственно — создавались под влиянием roguelike.

**1.3. Игровой движок Unity**

Unity — межплатформенная [среда разработки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) компьютерных игр, разработанная американской компанией [Unity Technologies](https://ru.wikipedia.org/wiki/Unity_Technologies). Unity позволяет создавать приложения, работающие на более чем 25 различных [платформах](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B0), включающих персональные компьютеры, [игровые консоли](https://ru.wikipedia.org/wiki/%D0%98%D0%B3%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BA%D0%BE%D0%BD%D1%81%D0%BE%D0%BB%D1%8C), мобильные устройства, [интернет-приложения](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) и другие. Выпуск Unity состоялся в [2005](https://ru.wikipedia.org/wiki/2005_%D0%B3%D0%BE%D0%B4_%D0%B2_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D1%85_%D0%B8%D0%B3%D1%80%D0%B0%D1%85) году и с того времени идет постоянное развитие.

Основными преимуществами Unity являются наличие визуальной среды разработки, [межплатформенной](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C) поддержки и модульной системы компонентов.

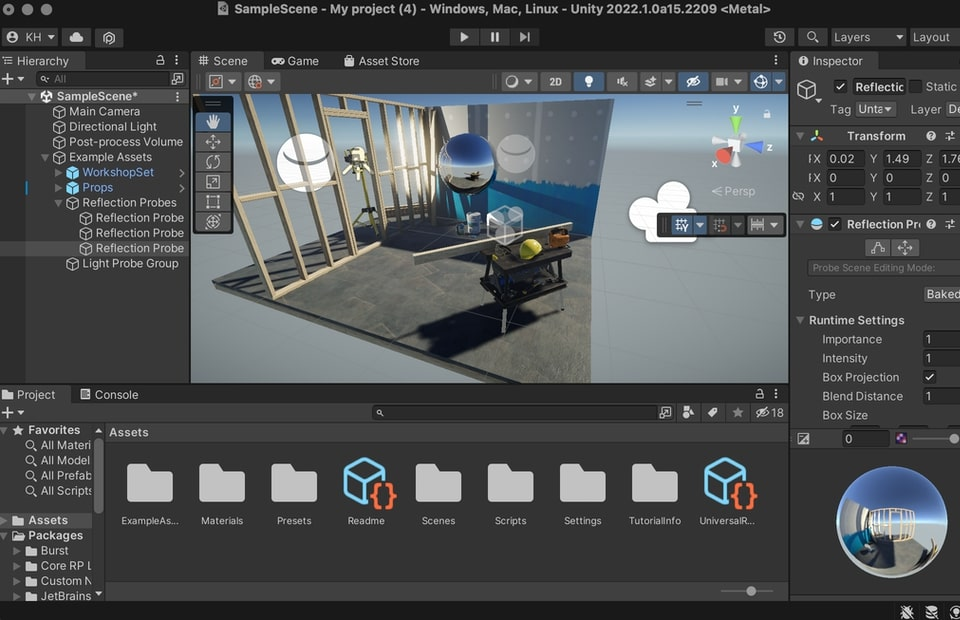
На Unity написаны тысячи игр, приложений, визуализации математических моделей, которые охватывают множество платформ и жанров. При этом Unity используется как крупными разработчиками, так и [независимыми студиями](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B4%D0%B8-%D0%B8%D0%B3%D1%80%D0%B0).

Для разработки моего проекта мной была выбрана именно среда разработки Unity, поскольку она делает процесс создания игр максимально удобным. Unity обладает огромным количеством преимуществ перед альтернативными способами разработки компьютерных игр.

Одной из таких особенностей является огромное количество плагинов, поддерживаемых Unity. Плагины значительно ускоряют процесс создания игр.

Unity позволяет создавать приложения для разных платформ, среди которых:

* Десктопные устройства (под управлением Windows, MacOS, Linux)
* Android
* iOS
* Веб-версии для браузеров, поддерживающих WebGL.



*Рис. 1.3* *– интерфейс среды разработки Unity.*

Наконец, Unity доступен бесплатно, что открывает перед независимыми разработчиками дверь в игровую индустрию.

Unity – это наиболее удобная среда разработки для меня, поскольку я уже имею немалый опыт работы как с ней, так и с языком C#, используемым для разработки игр на Unity.

**1.4. Язык программирования C#**

C# — современный объектно-ориентированный и типобезопасный язык программирования. Он позволяет разработчикам создавать множество типов безопасных и надежных приложений. C# относится к широко известному семейству языков C, и покажется хорошо знакомым любому, кто работал с C, C++, Java или JavaScript.

C# поддерживает наследование, перегрузку операторов, полиморфизм и продолжает активно развиваться, с каждой новой версией появляются новые функциональности, как, например, динамической связывание, лямбды, асинхронные методы и так далее.

Немаловажно, что С# является не учебным, а профессиональным языком, предназначенным для решения широкого спектра задач, и в первую очередь - в быстро развивающейся области создания распределенных приложений. Поэтому базовый курс программирования, построенный на основе языка С#, позволит студентам быстрее стать востребованными специалистами-профессионалами.

На данный момент, C# является единственным языком, поддерживаемым средой разработки Unity. Дело в том, что C# обладает различными преимуществами перед другими языками программирования, которые полезны именно для Unity:

* C# – это объектно-ориентированный, простой и в то же время мощный язык программирования, который позволяет разработчикам создавать многофункциональные приложения.
* C# относится к языкам компилируемого типа, поэтому он обладает всеми преимуществами таких языков.
* C# объединяет лучшие идеи современных языков программирования Java, C++, Visual Basic и т.д.
* Из-за большого разнообразия синтаксических конструкций и возможности работать с платформой .Net, C# позволяет быстрее, чем любой другой язык, разрабатывать программные решения.
* C# отличается надежностью и элегантностью.

**1.5. Фреймворк Zenject**

Zenject — контейнер внедрения зависимостей с открытым исходным кодом, нацеленный на применение с игровым движком Unity, обеспечивающий работу на большинстве платформ, поддерживаемых Unity. Стоит заметить, что Zenject можно применять и для С# приложений, разработанных без Unity. Этот контейнер является довольно популярным среди Unity разработчиков, активно поддерживается и развивается. Кроме того, Zenject обладает всем необходимым контейнеру DI функционалом.

Внедрение зависимостей — это стиль настройки объекта, при котором поля объекта задаются внешней сущностью. Другими словами, объекты настраиваются внешними объектами. DI — это альтернатива самонастройке объектов.

Использование фреймворка Zenject при разработки приложений в Unity позволяет значительно упростить процесс написания кода, вывести архитектуру приложения на новый уровень.

Многие крупные проекты были разработаны с использованием данного фреймворка, например Pokemon Go.

Также можно сказать, что умение пользоваться Zenject при создании приложений в Unity является важной характеристикой достаточного опытного Unity-разработчика.

**ГЛАВА 2. РАЗРАБОТКА СПОСОБА РЕАЛИЗАЦИИ ПРОЦЕДУРНОЙ ГЕНЕРАЦИИ УРОВНЯ. РАЗБОР АЛГОРИТМОВ И РЕАЛИЗАЦИЯ ПРОЦЕДУРНОЙ ГЕНЕРАЦИИ УРОВНЯ. ДОБАВЛЕНИЕ ГРАФИЧЕСКИХ РЕСУРСОВ**

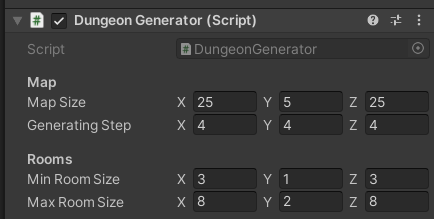
**2.1. Разработка способа реализации процедурной генерации уровня**

Существует множество способов решения задачи реализации процедурной генерации уровня. В качестве генерируемого уровня мною был выбран классический вариант – подземелье, состоящее из комнат, коридоров и лестниц.

За основу я взял трехмерный мир, разделенный в виде прямоугольной сетки. Одна ячейка этой сетки – куб размером (1, 1, 1).

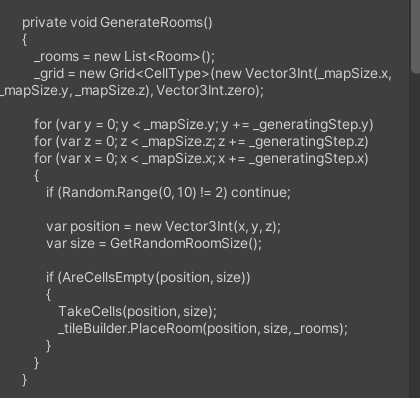
Основная характеристики карты, на которой будет генерироваться уровень, это ее размер.

Для того, чтобы каким-то образом сгенерировать подземелье, нам изначально необходимо сгенерировать комнаты этого подземелья, поскольку именно они и являются ключевыми точками на полученном уровне. У комнат также есть свои характеристики: их минимальный и максимальный размер и минимально возможный отступ между различными комнатами.



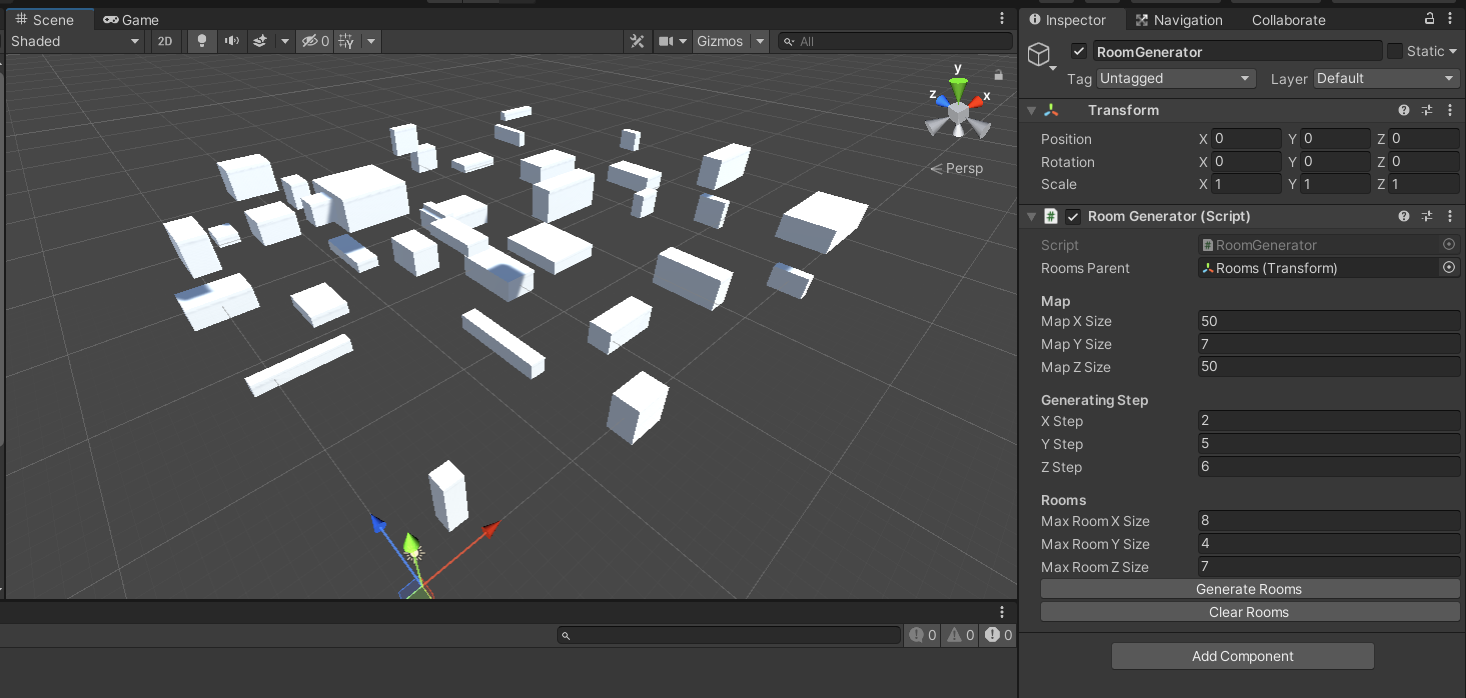
*Рис 2.1 – компонент Dungeon Generator.*

Класс DungeonGenerator впоследствии и будет отвечать за генерацию всего нашего подземелья. На данный момент добавим в него метод, отвечающий за случайную генерацию наших будущих комнат.



*Рис. 2.2 – метод GenerateRooms().*

Результатом данного метода является генерация на уровне комнат случайного размера и со случайной позицией:



*Рис. 2.3 – случайно сгенерированные комнаты.*

Пока что “комнаты” имеют вид белых параллелепипедов различного размера, однако после реализации всей механики мы будем добавлять им графические ассеты.

Следующим шагом для реализации процедурной генерации будет проведение между комнатами некоторых отрезков, вдоль которых впоследствии будут строится коридоры и лестницы. Реализовать такую механику без знания существующих алгоритмов было бы не так просто, благо многие вещи в математике уже были придуманы до нас.

**2.2. Разбор алгоритмов**

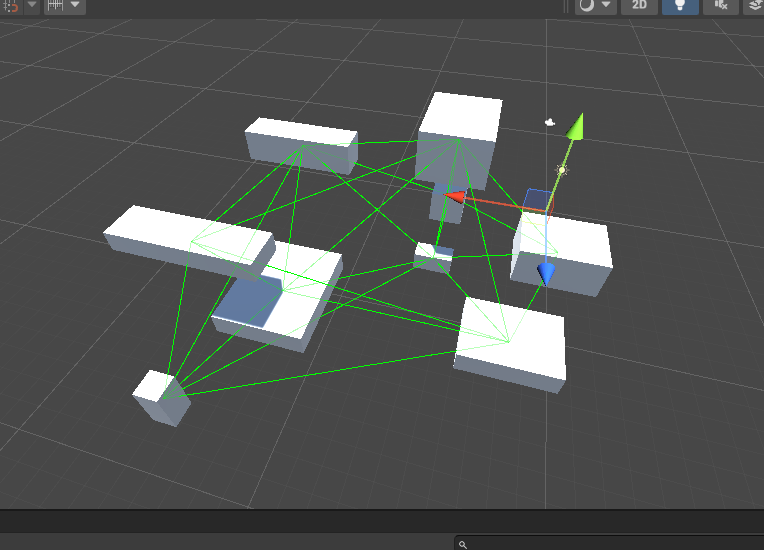
**Тетраэдрация Делоне и алгоритм Боуэра-Ватсона**

В качестве объединения всех комнат (то есть вершин) в некоторую сетку мною был выбран граф триангуляции Делоне, а точнее его трехмерная вариация – граф тетраэдрации Делоне. Для его создания я использовал алгоритм Боуэра-Ватсона. В различных ресурсах можно найти множество версий его реализации для двухмерного пространства, но для трехмерного пространства информации совсем мало.

Тетраэдрация – разбиение пространства на тетраэдры.

При использовании данного алгоритма нам необходимо иметь множество всех точек, для которых мы будем создавать тетраэдрацию. Изначально мы создаем так называемый “супертетраэдр”, который будет включать в себя все точки заданного графа, после чего на каждой итерации мы постепенно исключаем из имеющегося тетраэдра вершины и создаем новые новые тетраэдры между исключенными вершинами. Алгоритм будет завершен, когда все созданные тетраэдры не будут содержать в своих описанных сферах никаких вершин графа, кроме как тех, что являются вершинами соответствующего тетраэдра.

Таким образом мы получили сетку, в узлах которой лежат комнаты генерируемого подземелья:



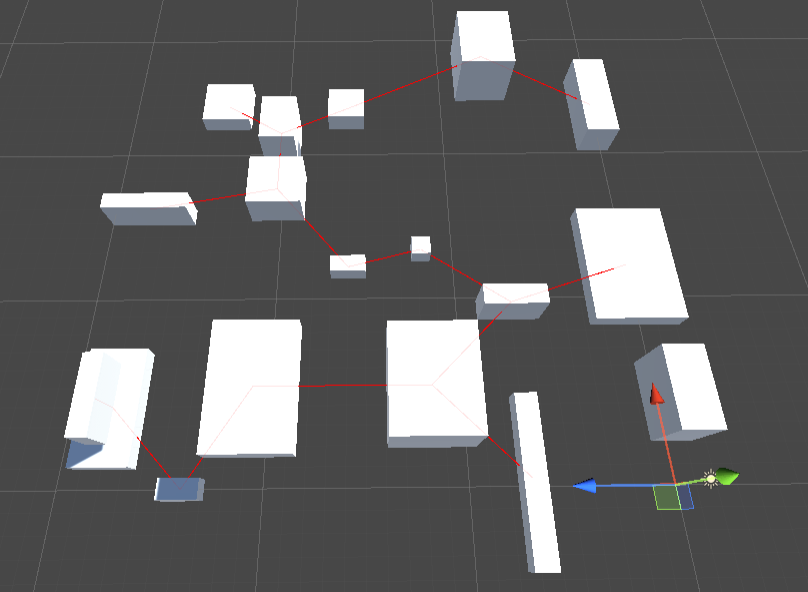
*Рис. 2.4 – тетраэдрация Делоне для комнат генерируемого уровня.*

**MST и алгоритм Прима**

Имея граф, объединяющий в себя все комнаты, мы можем создать минимальное остовное дерево (minimum spanning tree, MST). Это необходимо, для будущего соединения комнат коридорами наиболее короткой длины, что будет выглядеть естественно.

Для создания минимального остовного дерева я использовал алгоритм Прима. Его реализация для трехмерного пространства несильно отличается от реализации для двухмерного.

Сначала берется произвольная вершина и находится ребро, инцидентное данной вершине и обладающее наименьшей стоимостью. Найденное ребро и соединяемые им две вершины образуют дерево. Затем, рассматриваются ребра графа, один конец которых — уже принадлежащая дереву вершина, а другой — нет; из этих ребер выбирается ребро наименьшей стоимости. Выбираемое на каждом шаге ребро присоединяется к дереву. Рост дерева происходит до тех пор, пока не будут исчерпаны все вершины исходного графа. Результатом работы алгоритма является остовное дерево минимальной стоимости. В нашем случае это коридоры, соединяющие между собой все комнаты минимальным путем, причем без образования каких-либо петель.



*Рис. 2.5 – MST сгенерированных комнат.*

**

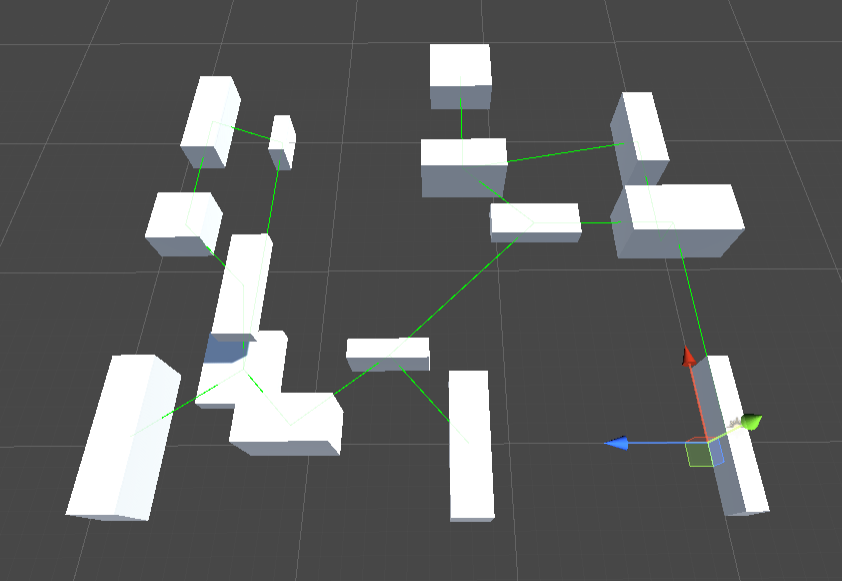
*Рис. 2.6 – код, реализующий алгоритм Прима.*

**Создание дополнительных коридоров между комнатами**

Для образования дополнительных коридоров (петель) между комнатами мы можем добавить некоторое случайное количество ребер графа из тетраэдрации Делоне, которые не входят в полученное остовное дерево.

Для этого необходимо перебрать каждого ребро тетраэдрации, проверить, включено ли оно в минимальное остовное дерево и, если нет, то со случайной вероятностью добавить его к будущим коридорам.

В результате мы получим граф, соединяющий между собой комнаты сгенерированного уровня как необходимыми коридорами, так и дополнительными, разнообразывающими общий ландшафт:



*Рис. 2.7 – полученный граф.*

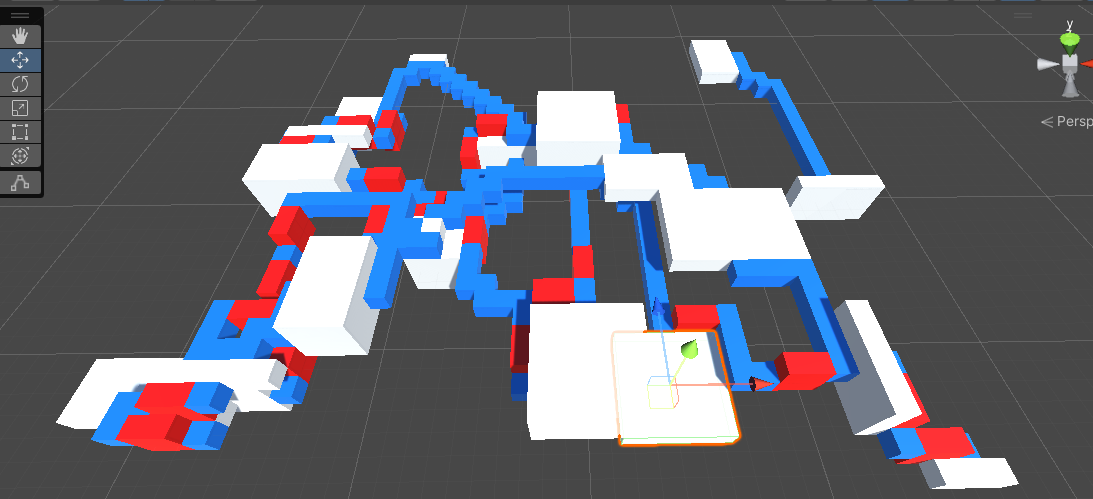
**Построение коридоров и алгоритм А\***

Для создания более красивых и реалистичных подземелий у меня возникла идея объединения некоторых полученных коридоров в общие и, как следствие, их увеличения, а не использования одного пути для построения разных коридоров. Кроме этого, существовала проблема того, что все комнаты находятся на разной высоте, и необходимо каким-то образом построить такие проходы, которые не уничтожали бы друг друга.

Для решения этой задачи был выбран поиск А\* – [алгоритм](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) поиска [по первому наилучшему совпадению](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B8%D1%81%D0%BA_%D0%BF%D0%BE_%D0%BF%D0%B5%D1%80%D0%B2%D0%BE%D0%BC%D1%83_%D0%BD%D0%B0%D0%B8%D0%BB%D1%83%D1%87%D1%88%D0%B5%D0%BC%D1%83_%D1%81%D0%BE%D0%B2%D0%BF%D0%B0%D0%B4%D0%B5%D0%BD%D0%B8%D1%8E) на [графе](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), который находит маршрут с наименьшей стоимостью от одной вершины (начальной) к другой (целевой, конечной).

A\* пошагово просматривает все пути, ведущие от начальной вершины в конечную, пока не найдет минимальный. Как и все [информированные алгоритмы поиска](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BF%D0%BE%D0%B8%D1%81%D0%BA), он просматривает сначала те маршруты, которые «кажутся» ведущими к цели.

В начале работы просматриваются узлы, смежные с начальным; выбирается тот из них, который имеет минимальное значение f(x), после чего этот узел раскрывается. На каждом этапе алгоритм оперирует с множеством путей из начальной точки до всех еще не раскрытых (листовых) вершин графа — множеством частных решений, — которое размещается в [очереди с приоритетом](https://ru.wikipedia.org/wiki/%D0%9E%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C_%D1%81_%D0%BF%D1%80%D0%B8%D0%BE%D1%80%D0%B8%D1%82%D0%B5%D1%82%D0%BE%D0%BC_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)).



*Рис. 2.8 – сгенерированное подземелье.*

После выполнения алгоритма мы получили “подземелье”: белые параллелепипеды – будущие комнаты, синие – коридоры, красные – лестницы.

Как можно заметить на рисунке, лестницы представляют собой группу кубов размером 2х2х1. Сама лестница будет занимать два нижних блока, то есть при прохождении на две клетки вперед, игрок будет подниматься на одну клетку. Два дополнительных блока сверху – это пустое пространство над лестницей.

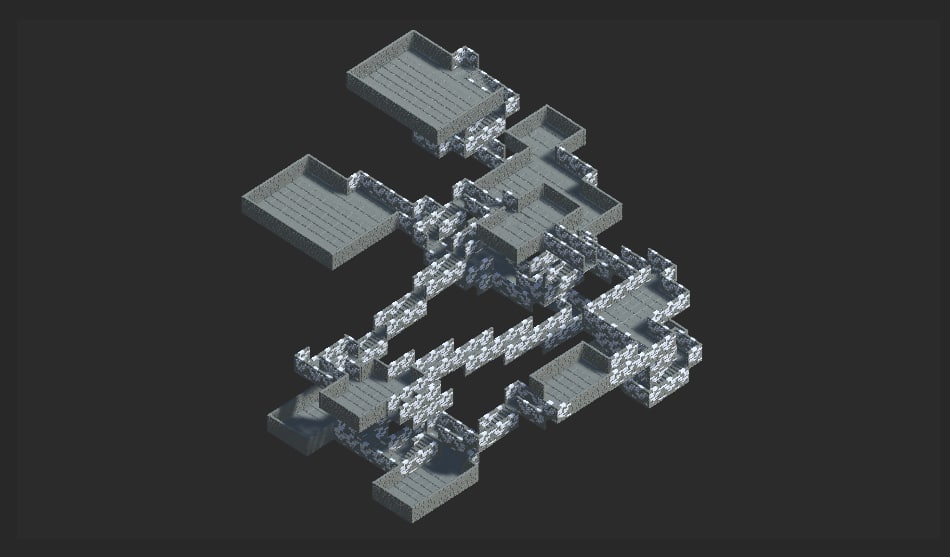
**2.3. Добавление графических ресурсов**

Поскольку изначальная задумка генерируемого уровня – это подземелье, в открытом доступе мною был найден бесплатный графический ассет трехмерных модульных объектов для окружения некоторого подземелья. Модели в данном наборе представляют собой клетки напольной плитки, стен, лестницы, окон, а также предметов интерьера.



*Рис. 2.9 – Modular Ruins Pack.*

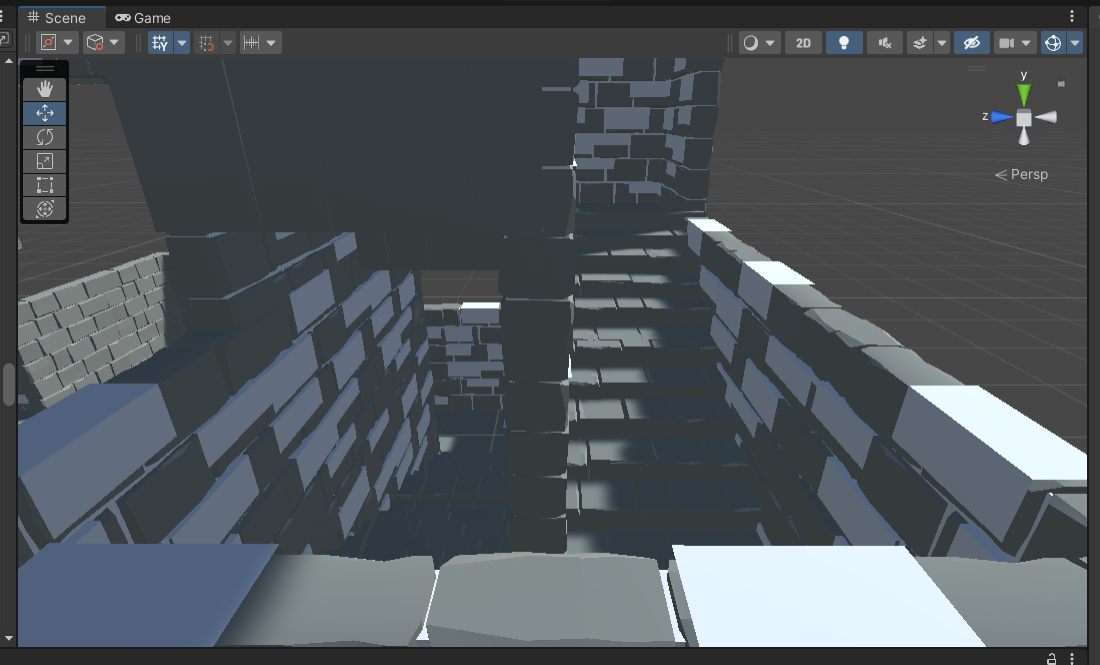
Заменив имеющиеся кубы на соответствующие для них модели, а также настроив генерацию таким образом, чтобы учитывался поворот коридоров и лестниц, я получил первоначальную версию подземелья:



*Рис. 2.10 – первоначальная версия подземелья.*

В полученном подземелье можно заметить некоторые интересные моменты, сгенерированные случайным образом.

Например, винтовая лестница:

**

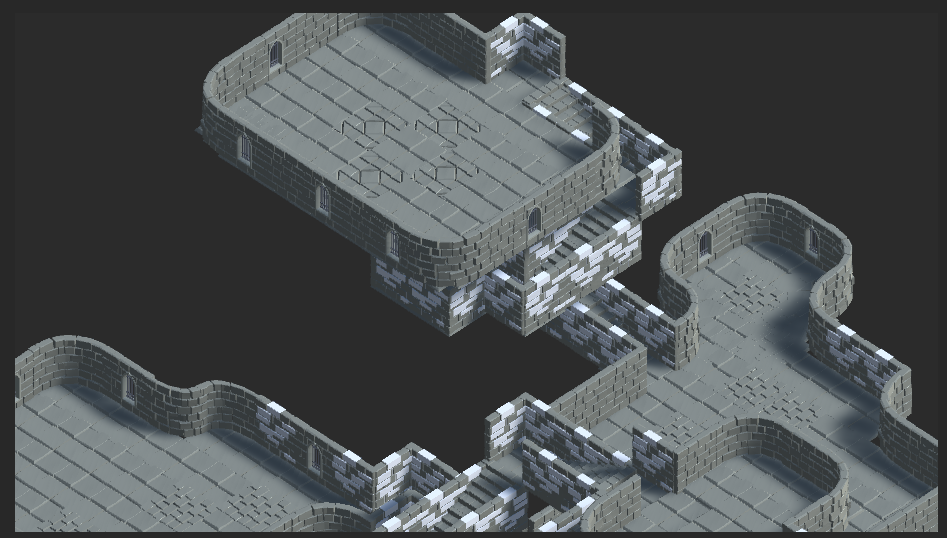
*Рис. 2.11 – винтовая лестница.*

Или место, которое объединило в себе комнату, несколько коридоров и лестницу:



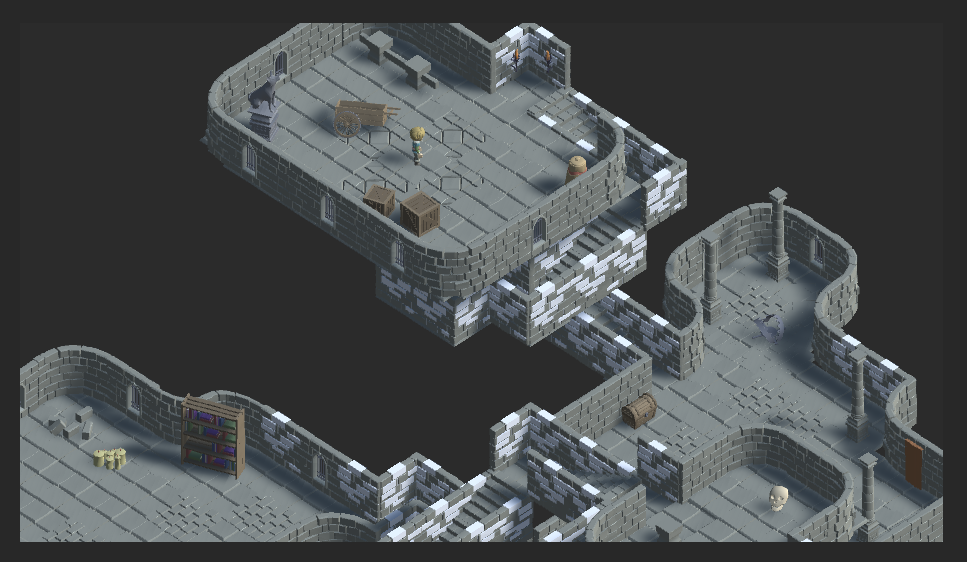
*Рис. 2.12 – интересная генерация.*

После этого я добавил случайность при выборе всех моделей: стены могут быть разными и могут содержать в себе окно, в углах комнаты стены скруглены, плитки на полу также могут различаться:



*Рис. 2.13 – обновленное подземелье.*

Для того, чтобы довести полученный уровень до ума, необходимо расставить на нем некоторые предметы, соответствующие окружению подземелья. Они также взяты из графического ассета:



*Рис. 2.14 – готовый уровень.*

**ЗАКЛЮЧЕНИЕ**

В своей курсовой работе я провел анализ современных тенденций индустрии разработки игр. Также я разобрал жанр “Roguelike”, его особенности и историю появления.

Была разработана концепция процедурной генерации уровня в стилистике подземелья, разобраны алгоритмы для ее исполнения, для некоторых алгоритмов разработана модификация для трехмерного пространства.

Программно реализована процедурная генерация уровня с использованием алгоритмов Боуэра-Ватсона, Прима и А\*.

Добавлены графические ресурсы.

Результатом данной работы является приложение, способное процедурно генерировать подземелье по заданным настройкам, что соответствует уровням в жанре игр “Roguelike”.

В будущем я хочу дополнить алгоритм генерации уровня таким образом, чтобы интерьер комнат также генерировался самостоятельно и случайным образом. Я планирую применить генетический алгоритм для того, чтобы окружение соседних комнат не совпадало, но подходило друг другу.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Обзор языка C# – <https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/>
2. Жанры компьютерных игр – <https://vlab.wikia.org/ru/wiki/%D0%96%D0%B0%D0%BD%D1%80%D1%8B_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D1%85_%D0%B8%D0%B3%D1%80>
3. Документация Zenject – <https://github.com/modesttree/Zenject>
4. Особенности движка Unity – <https://cubiq.ru/dvizhok-unity/>
5. Тренды-2020, видеоигры: по дороге с облаками: <https://vc.ru/future/109346-trendy-2020-videoigry-po-doroge-s-oblakami>
6. Что такое “метавселенная” – <https://vc.ru/future/281044-chto-takoe-metavselennaya-kotoruyu-stroyat-mark-cukerberg-satya-nadella-i-tim-suini-i-zachem-eto-nuzhno>
7. Процедурная генерация многоэтажных 3D-подземелий – <https://habr.com/ru/post/481218/>
8. Алгоритм Боуэра-Ватсона – <https://www.wikiwand.com/ru/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%91%D0%BE%D1%83%D1%8D%D1%80%D0%B0_%E2%80%94_%D0%92%D0%B0%D1%82%D1%81%D0%BE%D0%BD%D0%B0>
9. Алгоритм Прима – <https://e-maxx.ru/algo/mst_prim>
10. Алгоритм А\* – <https://habr.com/ru/post/331192/>
11. Modular Ruins Pack – <https://www.reddit.com/r/gamedev/comments/ph4yjg/ive_made_a_huge_modular_ruins_pack_you_can_use/>