Министерство образования Республики Беларусь

Учреждение образования Белорусский государственный университет информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Объектно-ориентированное программирование (ООП)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту на тему

«Игра с бесконечной генерацией уровня»

Минск

2021

**Содержание**

[Введение 5](#_Toc73792177)

[1. Постановка задачи 6](#_Toc73792178)

[1.1 Описание предметной области 6](#_Toc73792179)

[1.2 Цели и задачи проектирования 7](#_Toc73792180)

[1.3 Требования к разрабатываемому программному обеспечению 8](#_Toc73792181)

[2 Проектирование задачи 8](#_Toc73792182)

[2.1 Проектирование архитектуры приложения 9](#_Toc73792183)

[2.2 Обоснование выбора технологий 10](#_Toc73792184)

[2.3 Описание сред реализации задачи 10](#_Toc73792185)

[3 Программная реализация 16](#_Toc73792186)

[3.1 Физическая структура приложения 16](#_Toc73792187)

[3.2 Архитектура приложения 17](#_Toc73792188)

[4. Описание применения 19](#_Toc73792189)

[Заключение 22](#_Toc73792190)

[Список использованных источников 23](#_Toc73792191)

[Приложения 23](#_Toc73792192)

# Введение

Первые компьютеры были созданы человеком для упрощения сложных и монотонных расчётов. Ив те далёкие года никто даже не мог себе представить другого использования этих, не побоюсь этого слова, величественных машин. А сегодня практически каждый человек когда-либо играл в какую-то игру на своём ЭВМ.

Для многих людей это хобби. Во время ковидных ограничений, людям было невозможно заниматься большинством досуга. Кто-то после тяжелого рабочего дня брал книжку и сидя в кресле наблюдал за приключениями главного героя, кто-то смотрел укутавшись в тёплый плед сериал, а кто-то улетал в другие миры и вселенные играя в компьютерные игры.

На сегодняшний день компьютерная техника достигла такого уровня развития, что позволяет программистам разрабатывать очень реалистичные игры с хорошим графическим и звуковым оформлением, сложными механиками и глубокой проработкой мира.

# 1. Постановка задачи

Наверно каждому хотелось почувствовать себя не в роли игрока, а в качестве разработчика какой-либо игры.

## 1.1 Описание предметной области

Для создания игры необходимо:

* Идея
* Знание кого-то ни было языка программирования
* Так называемый игровой движок[[1]](#endnote-1)

Идея появилась достаточно быстро. Сеттинг[[2]](#endnote-2) игры проходит в лабиринте Минотавра на острове Крит. Задача игрока пробежать как можно дальше, увариваясь от атак быков. Игровой уровень должен быть разнообразным и генерироваться автоматически, чтобы не надоедать игроку и быть реиграбельным.

Для решения поставленных задач был выбран язык программирования C# и игровой движок Unity.

Unity позволяет создавать как 2d так и 3d игры. Я выбрал 2d, так как на создание простой 2д графики уйдёт гораздо меньше ресурсов чем для 3д.

Карта будет состоять из клеточек, тайлов воды, дорог и травы.

Тип генерации карты выбран коллапс волновой функции WFC, который подбирает соседние тайлы из набора всех тайлов и проверяя могут ли они соприкасаться.

## 1.2 Цели и задачи проектирования

Игровая карта:

* Создать Алгоритм WFC
* Нарисовать тайлы
* Создать Фасад для Алгоритма WFC
* Создать генератор карт
* Создать границы карты

Добавить каллайдеры для границы карты

Добавить триггеры для тайлов карты

Игрок:

* Нарисовать игрока
* Добавить управление для игрока WSAD + left Shift
* Добавить камеру
* Добавить коллайдер

Противник:

* Нарисовать противника
* Создать алгоритм атаки игрока
* Добавить коллайдер

UI:

* Создать счёт
* Создать рекорд
* Создать окно проигрыша
* Добавить пасхальное яйцо

Сцена:

* Создать завершение игры
* Добавить перезапуск сцены после проигрыша

## 1.3 Требования к разрабатываемому программному обеспечению

UE:

Удобное управление

Сложность уклонения от противников выше среднего, что повышает интерес к игре на рекорд

Минимальная графика

Сцена:

Генерация карты с помощью WFC

Камера следует за игроком

Скорость игрока зависит от тайла на котором он находиться

Игрок не может выходить за границы карты

2 Проектирование задачи

## 2.1 Проектирование архитектуры приложения

Перед началом работы была разработана данная диаграмма классов, которая являлась шаблоном для игры, и в процессе разработки менялась и дополнялась.

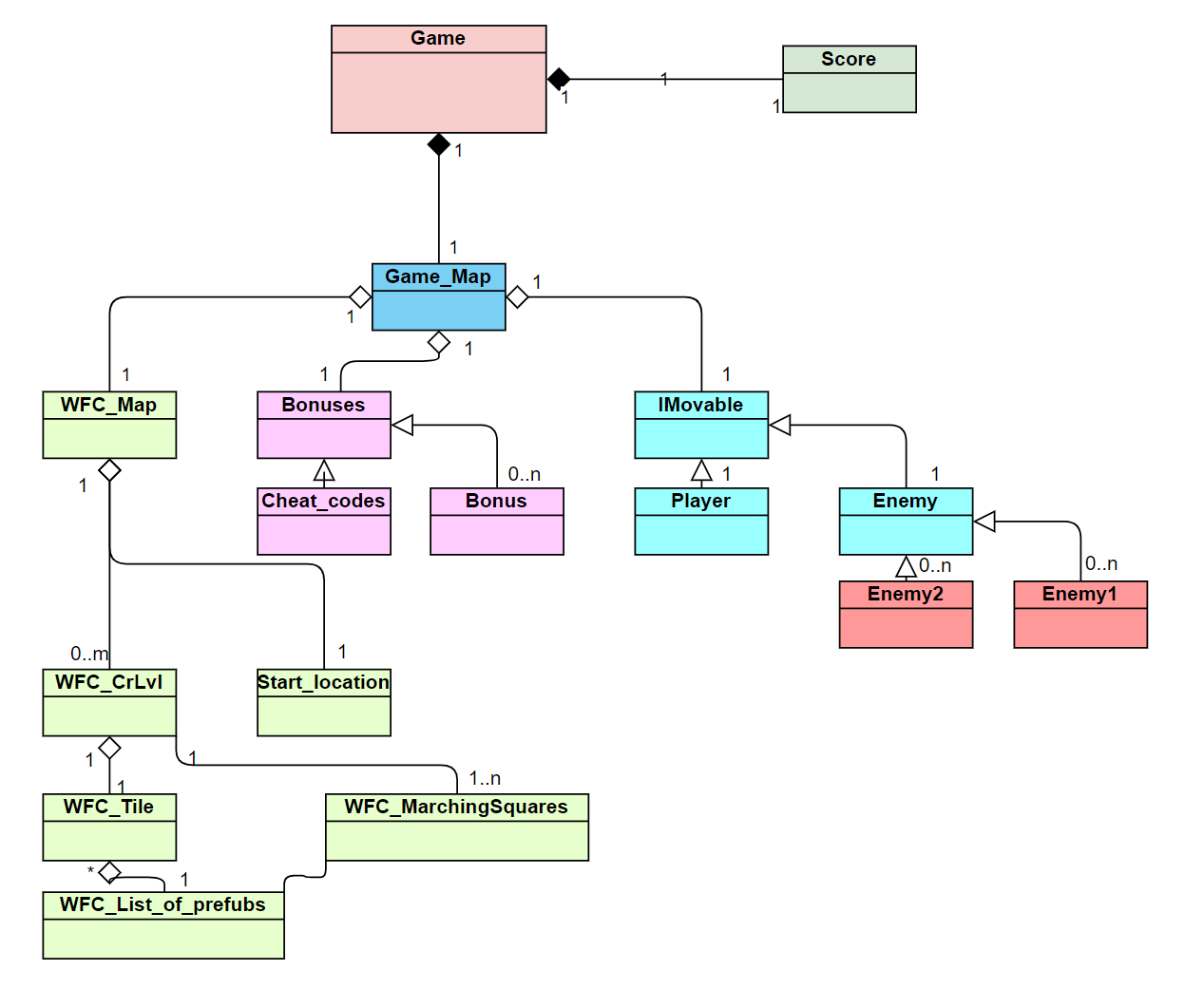


Рис.1 Начальная диаграмма классов

* Класс Game является основным объектом игры. Он включает в себя поля жизней, бонусов и счёта, методы запуска и остановки игры, а также связывает между собой классы score и map.
* Класс Score отображает счёт, бонусы.
* Класс Game\_Map включает в себя основную игровую логику, просчитывает колизии, перемещает объекты.
* Класс WFC\_Map создаёт карту из нескольких уровней WFC\_CrLvl и Start\_location и др.
* WFC\_Tile является объектом который создаётся в каждой клеточке WFC\_CrLvl.possible[x, y] и содержит все необходимые свойства для коллапсирования тайлов
* WFC\_List\_of\_prefabs это список всех тайлов со скриптом WFC\_MarchingSquares, который является описанием каждого тайла-префаба по-отдельности
* Bonuses это класс от которого наследуется читкоды и все остальные бонусы
* Интерфейс IMovable включает в себя объекты которые могут двигаться по карте.
* Player игрок за кого и происходит игра
* Enemy класс который наследуют все враги в игре

## 2.2 Обоснование выбора технологий

Unity и язык C# одна из самых популярных технологий в игровой индустрии. Язык C# изучался в предыдущих семестрах, что значительно ускорила ознакомление с игровым движком и позволила с минимальными временными затратами начать делать данный проект. Большое количество обучающих материалов помогло быстро решать появляющиеся проблемы во время процесса разработки игры.

## 2.3 Описание сред реализации задачи

**Visual Studio**

Интегрированная среда разработки Visual Studio — это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Интегрированная среда разработки (IDE) представляет собой многофункциональную программу, которую можно использовать для различных аспектов разработки программного обеспечения. Помимо стандартного редактора и отладчика, которые существуют в большинстве сред IDE, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для упрощения процесса разработки.

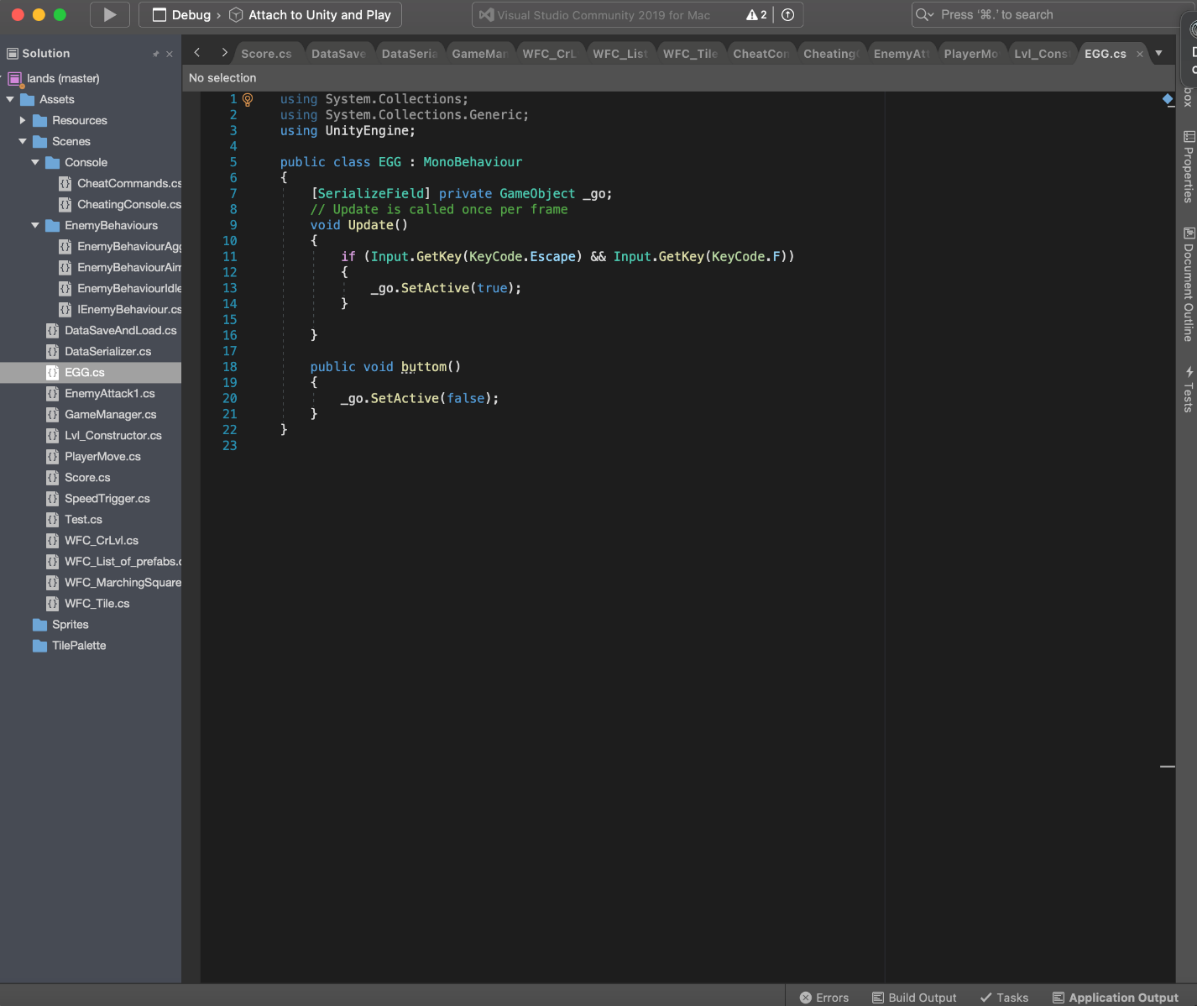


Рис.2 Visual Studio

**Unity**

Unity — межплатформенная среда разработки компьютерных игр. Unity позволяет создавать приложения, работающие под более чем 20 различными операционными системами, включающими персональные компьютеры, игровые консоли, мобильные устройства, интернет-приложения и другие. Выпуск Unity состоялся в 2005 году и с того времени идёт постоянное развитие.

Основными преимуществами Unity являются наличие визуальной среды разработки, межплатформенной поддержки и модульной системы компонентов. К недостаткам относят появление сложностей при работе с многокомпонентными схемами и затруднения при подключении внешних библиотек.

На Unity написаны тысячи игр, приложений и симуляций, которые охватывают множество платформ и жанров. При этом Unity используется как крупными разработчиками, так и независимыми студиями

***Возможности***

Редактор Unity имеет простой Drag&Drop интерфейс, который легко настраивать, состоящий из различных окон, благодаря чему можно производить отладку игры прямо в редакторе. Движок поддерживает два скриптовых языка: C#, JavaScript (модификация). Ранее была поддержка Boo (диалект Python), но его убрали в 5-й версии. Расчёты физики производит физический движок PhysX от NVIDIA.

Проект в Unity делится на сцены (уровни) — отдельные файлы, содержащие свои игровые миры со своим набором объектов, сценариев, и настроек. Сцены могут содержать в себе как, собственно, объекты (модели), так и пустые игровые объекты — объекты, которые не имеют модели («пустышки»). Объекты, в свою очередь содержат наборы компонентов, с которыми и взаимодействуют скрипты. Также у объектов есть название (в Unity допускается наличие двух и более объектов с одинаковыми названиями), может быть тег (метка) и слой, на котором он должен отображаться. Так, у любого объекта на сцене обязательно присутствует компонент Transform — он хранит в себе координаты местоположения, поворота и размеров объекта по всем трём осям. У объектов с видимой геометрией также по умолчанию присутствует компонент Mesh Renderer, делающий модель объекта видимой.

К объектам можно применять коллизии (в Unity т. н. коллайдеры — collider), которых существует несколько типов.

Также Unity поддерживает физику твёрдых тел и ткани, а также физику типа Ragdoll (тряпичная кукла). В редакторе имеется система наследования объектов; дочерние объекты будут повторять все изменения позиции, поворота и масштаба родительского объекта. Скрипты в редакторе прикрепляются к объектам в виде отдельных компонентов.

При импорте текстуры в Unity можно сгенерировать alpha-канал, mip-уровни, normal-map, light-map, карту отражений, однако непосредственно на модель текстуру прикрепить нельзя — будет создан материал, которому будет назначен шейдер, и затем материал прикрепится к модели. Редактор Unity поддерживает написание и редактирование шейдеров. Редактор Unity имеет компонент для создания анимации, но также анимацию можно создать предварительно в 3D-редакторе и импортировать вместе с моделью, а затем разбить на файлы.

Unity 3D поддерживает систему Level Of Detail (сокр. LOD), суть которой заключается в том, что на дальнем расстоянии от игрока высокодетализированные модели заменяются на менее детализированные, и наоборот, а также систему Occlusion culling, суть которой в том, что у объектов, не попадающих в поле зрения камеры не визуализируется геометрия и коллизия, что снижает нагрузку на центральный процессор и позволяет оптимизировать проект. При компиляции проекта создается исполняемый (.exe) файл игры (для Windows), а в отдельной папке — данные игры (включая все игровые уровни и динамически подключаемые библиотеки).

Движок поддерживает множество популярных форматов. Модели, звуки, текстуры, материалы, скрипты можно запаковывать в формат .unityassets и передавать другим разработчикам, или выкладывать в свободный доступ. Этот же формат используется во внутреннем магазине Unity Asset Store, в котором разработчики могут бесплатно и за деньги выкладывать в общий доступ различные элементы, нужные при создании игр. Чтобы использовать Unity Asset Store, необходимо иметь аккаунт разработчика Unity. Unity имеет все нужные компоненты для создания мультиплеера. Также можно использовать подходящий пользователю способ контроля версий. К примеру, Tortoise SVN или Source Gear.

В Unity входит Unity Asset Server — инструментарий для совместной разработки на базе Unity, являющийся дополнением, добавляющим контроль версий и ряд других серверных решений.

***Достоинства и недостатки***

Как правило, игровой движок предоставляет множество функциональных возможностей, позволяющих их задействовать в различных играх, в которые входят моделирование физических сред, карты нормалей, динамические тени и многое другое. В отличие от многих игровых движков, у Unity имеется два основных преимущества: наличие визуальной среды разработки и межплатформенная поддержка. Первый фактор включает не только инструментарий визуального моделирования, но и интегрированную среду, цепочку сборки, что направлено на повышение производительности разработчиков, в частности, этапов создания прототипов и тестирования. Под межплатформенной поддержкой предоставляется не только места развертывания (установка на персональном компьютере, на мобильном устройстве, консоли и т. д.), но и наличие инструментария разработки (интегрированная среда может использоваться под Windows и Mac OS).

Третьим преимуществом называется модульная система компонентов Unity, с помощью которой происходит конструирование игровых объектов, когда последние представляют собой комбинируемые пакеты функциональных элементов. В отличие от механизмов наследования, объекты в Unity создаются посредством объединения функциональных блоков, а не помещения в узлы дерева наследования. Такой подход облегчает создание прототипов, что актуально при разработке игр.

В качестве недостатков приводятся ограничение визуального редактора при работе с многокомпонентными схемами, когда в сложных сценах визуальная работа затрудняется. Вторым недостатком называется отсутствие поддержки Unity ссылок на внешние библиотеки, работу с которыми программистам приходится настраивать самостоятельно, и это также затрудняет командную работу. Ещё один недостаток связан с использованием шаблонов экземпляров (англ. prefabs). С одной стороны, эта концепция Unity предлагает гибкий подход визуального редактирования объектов, но с другой стороны, редактирование таких шаблонов является сложным. Также, WebGL-версия движка, в силу специфики своей архитектуры (трансляция кода из C# в С++ и далее в JavaScript), имеет ряд нерешённых проблем с производительностью, потреблением памяти и работоспособностью на мобильных устройствах.

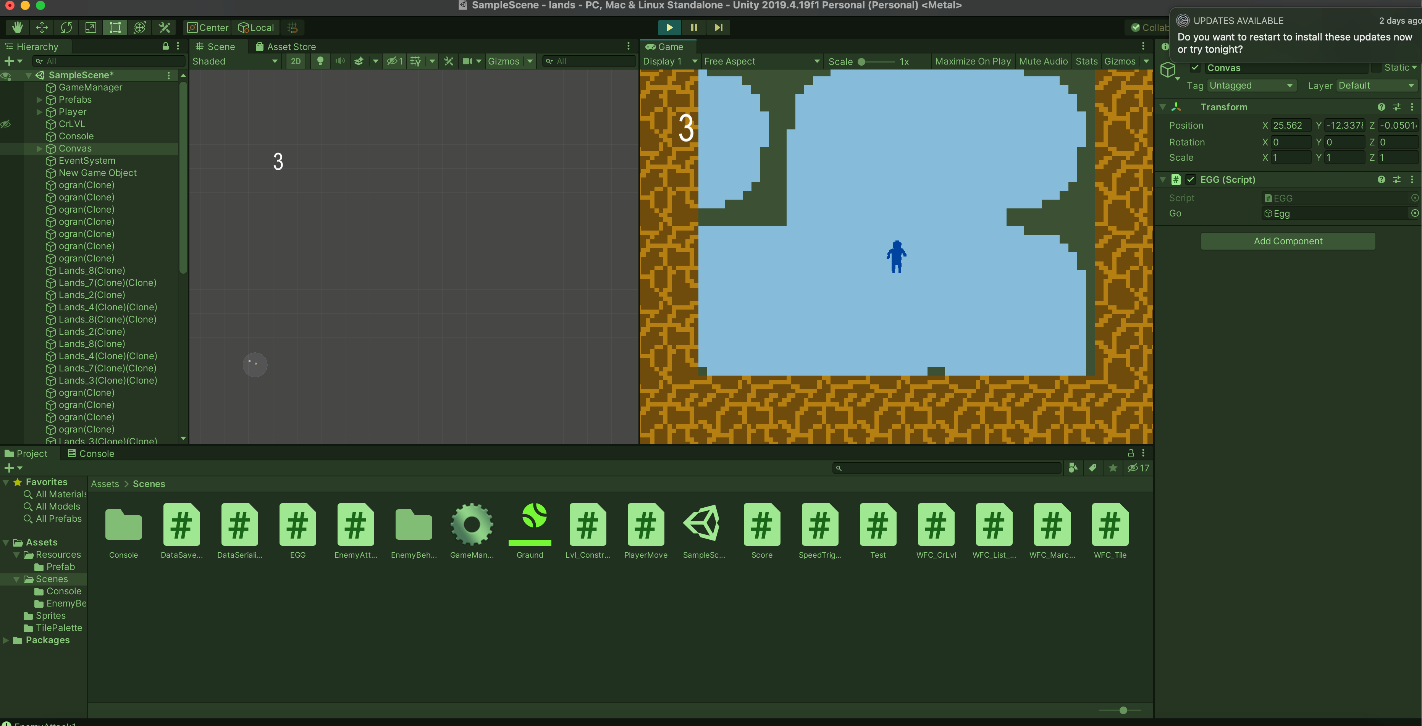


Рис.3 Unity

# 3 Программная реализация

## 3.1 Физическая структура приложения

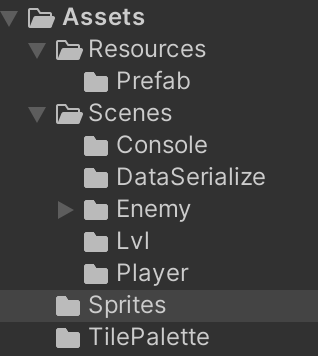


Рис.4 Структура приложения

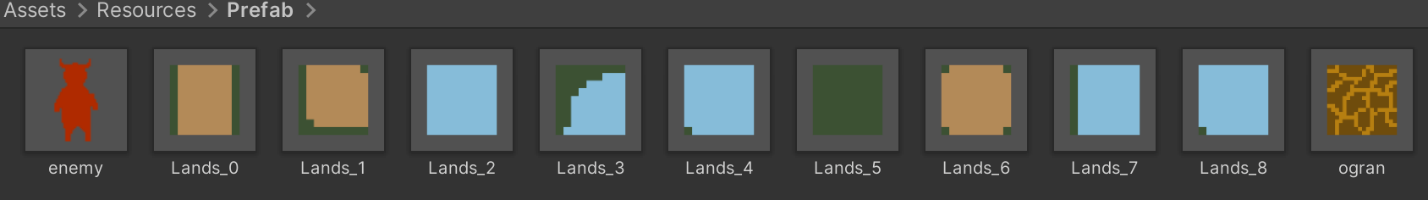
В папке Resourses хранится префабы тайлов, игрока и противника. 

Рис.5 Папка Resourses

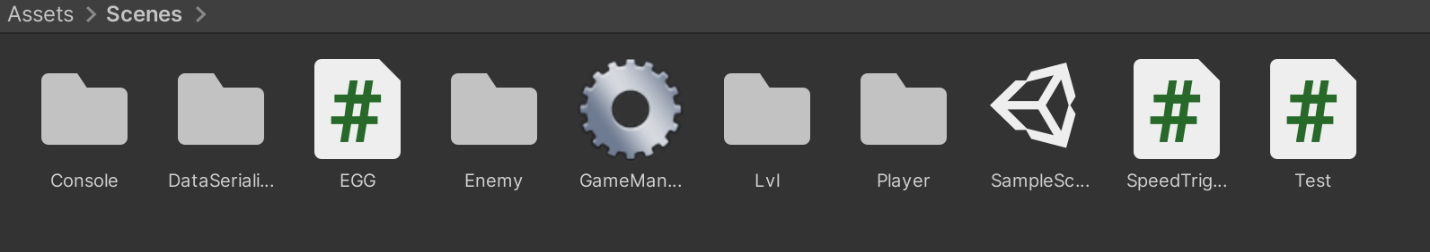
Папка сцена является хранилищем всех скриптов разделённых на категории 

Рис.6 Папка Scenes

Папка Sprites используется для хранения спрайтов

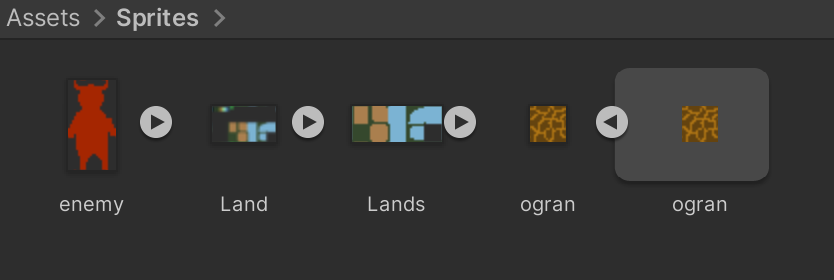


Рис.7 Папка Sprites

## 3.2 Архитектура приложения

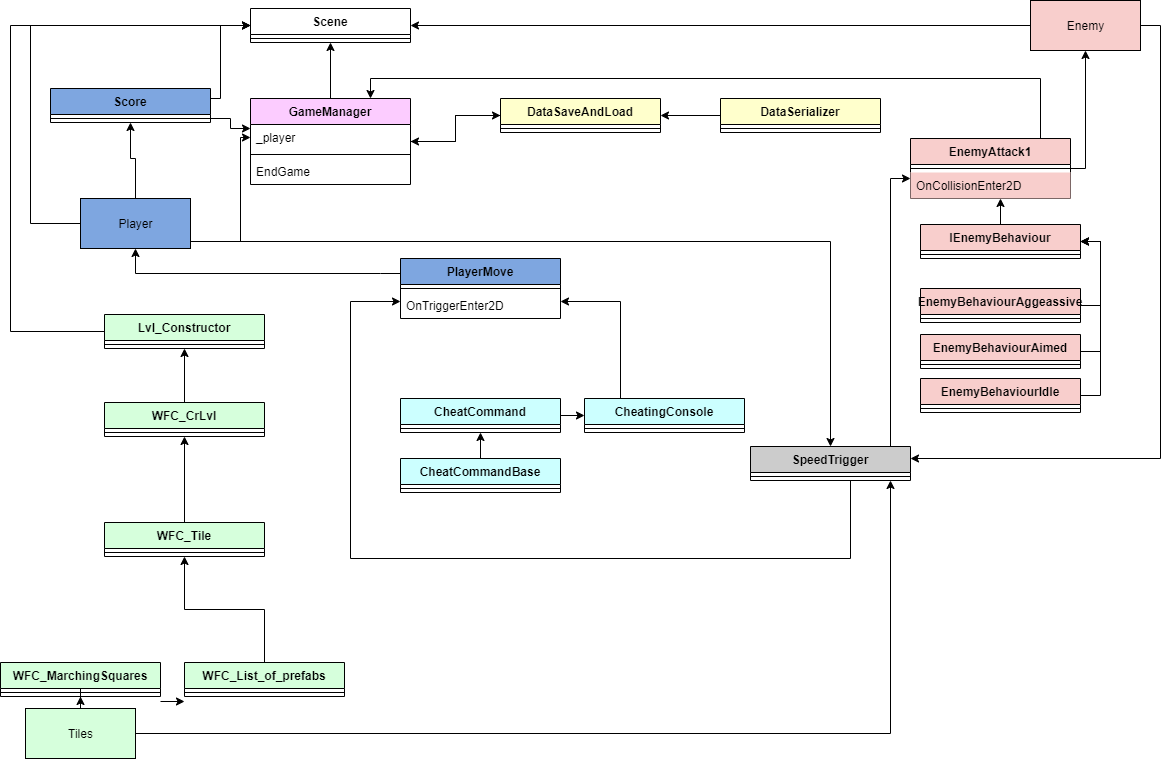


Рис.8 Конечная диаграмма классов.

В которой:

* Класс GameManager является основным классом игры. Он перезапускает игру после проигрыша, управляет UI, сериализует и десериализует рекорды.
* Класс Score отображает счёт.
* Класс Game\_Map включает в себя основную игровую логику, просчитывает колизии, перемещает объекты.
* Класс Lvl\_Constructor создаёт карту с помощью фасада WFC\_CrLvl.CreateLvl
* WFC\_Tile является объектом который создаётся в каждой клеточке WFC\_CrLvl.possible[x, y] и содержит все необходимые свойства для коллапсирования тайлов
* WFC\_List\_of\_prefabs это список всех тайлов со скриптом WFC\_MarchingSquares, который является описанием каждого тайла-префаба по-отдельности
* Player игрок за кого и происходит игра
* Enemy префаб противников

# 4. Описание применения

Благодаря тому, что карта генерируется «до бесконечности» и противники появляются в случайных местах игра идеально подходит для соревнований между людьми.

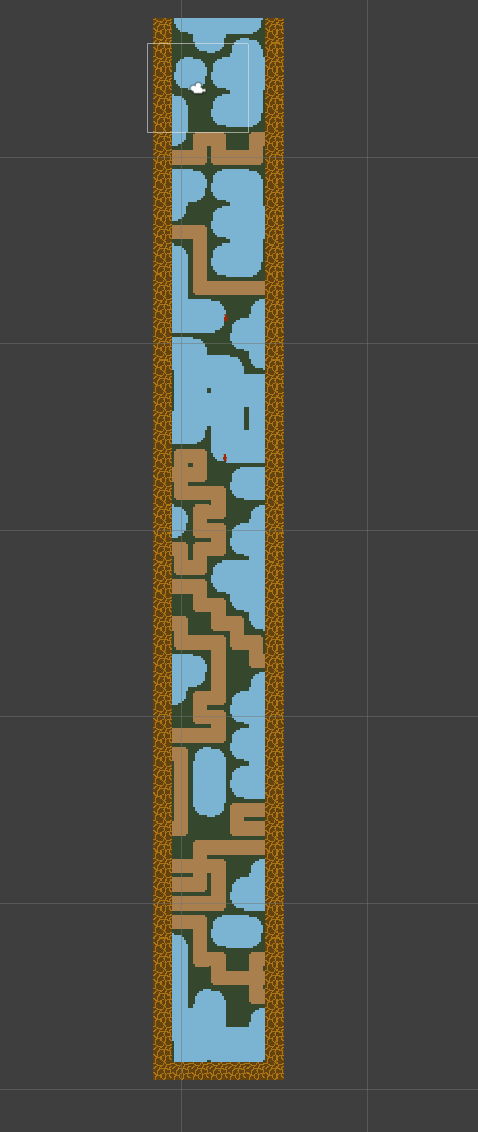
****

Рис.9 Сгенерированная карта

Для игры с другом на одном телефоне была добавлена строка со счётом прошлой игры.

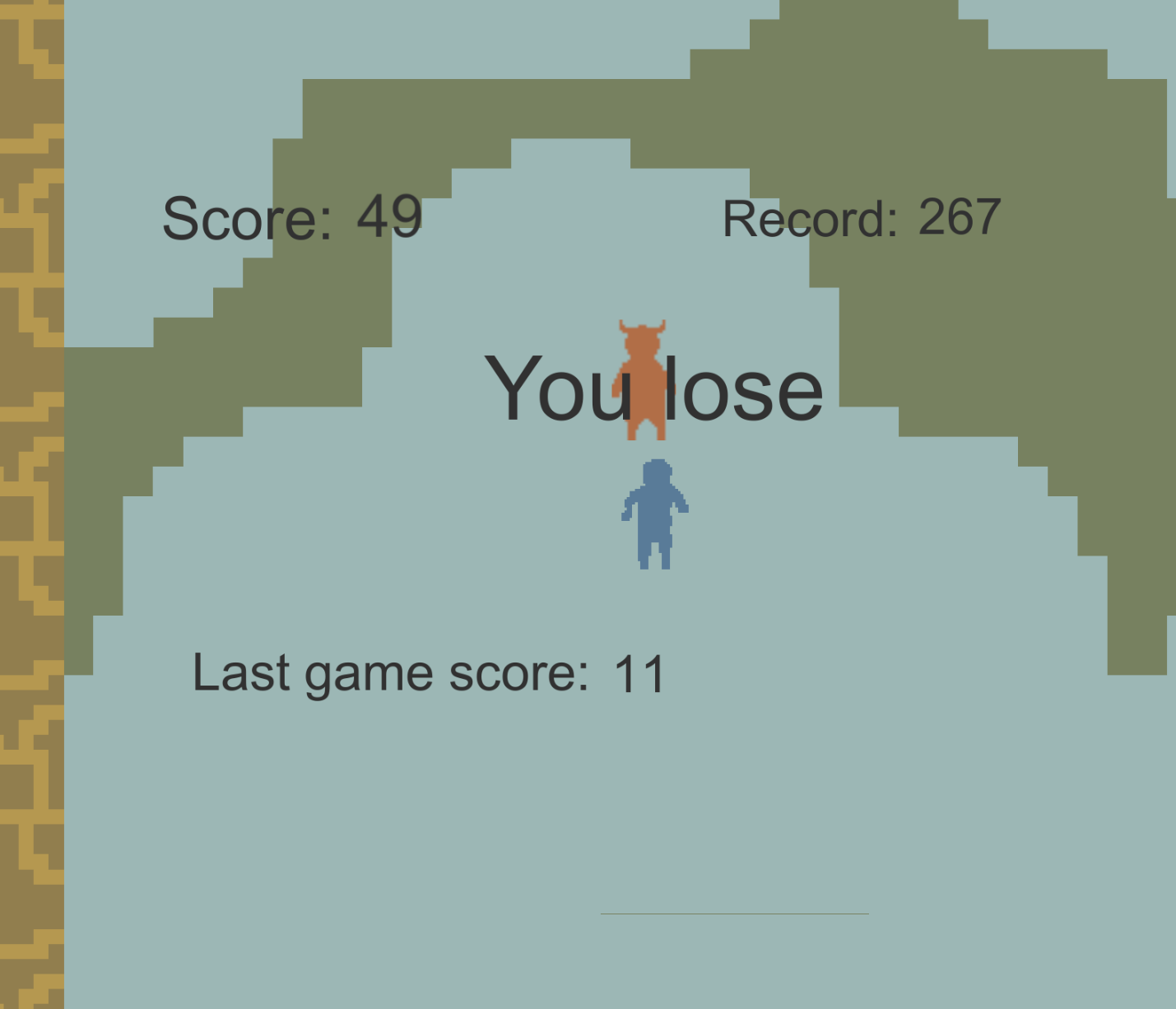
****

Рис.10 Окно проигрыша

Скорость игрока и противников зависит от поверхности, на которой они находятся, что является одной из дополнительных механик в игре с помощью которой можно балансировать удобство и сложность, а также это подспорье для добавления новых поверхностей.

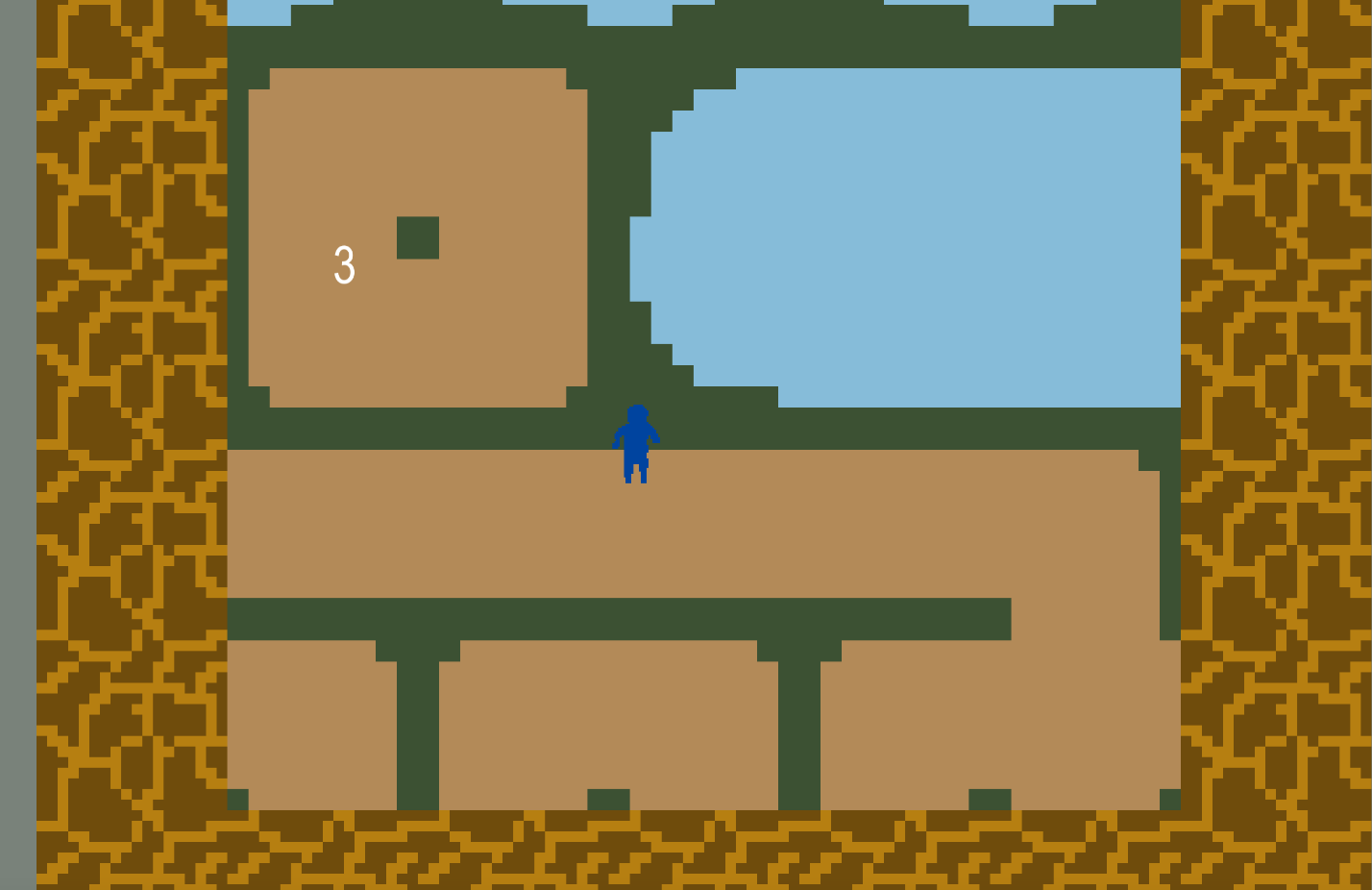
****

Рис.11 Окно игры

# Заключение

Игра получилась очень затягивающей, хоть она и имеет очень простую графику. Сложность выше среднего постоянно бросает вызов игроку и он пытается из раза в раз побить свой предыдущий рекорд. При этом если играть с другом, то состязание становиться ещё более захватывающим.

Все поставленные цели были реализованы.

И в процессе разработки появлялись новые идеи, что является индикатором масштабируемого приложения. Архитектура позволяет без каких-либо трудностей добавлять новые системы, механики и противников. Изменяя параметры скорости, дальности обнаружения противников и игрока, можно как усложнять игру со временем, так и наоборот облегчать игру для начинающих игроков.

# Список использованных источников

1. Нейгел К. И др. C# 5.0 и платформа .NET 4.5 для профессионалов, М.: Диалектика, 2013
2. Рендольф Н., Гарднер Д., Минутилло М., Андерсон К. Microsoft Visual Studio 2010 для профессионалов, М.: Диалектика, 2011
3. Шилдт Г. C# 4.0: полное руководство, М.: ООО «И.Д. Вильямс», 2011
4. habr.com
5. wikipedia.org
6. docs.unity3d.com

# Приложения

1. Термин «игровой движок» появился в середине 1990-х в контексте компьютерных игр жанра шутер от первого лица, похожих на популярную в то время Doom. Архитектура программного обеспечения Doom была построена таким образом, что представляла собой разумное и хорошо выполненное разделение центральных компонентов игры (например, подсистемы трёхмерной графики, расчёта столкновений объектов, звуковой и других) и графических ресурсов, игровых миров, формирующие опыт игрока игровые правила и другое. Как следствие, это получило определённую ценность за счёт того, что начали создаваться игры с минимальными изменениями, когда при наличии игрового движка компании создавали новую графику, оружие, персонажей, правила игры и тому подобное. (Jason, 2009) [↑](#endnote-ref-1)
2. Се́ттинг (англ. setting «помещение, установка, обстановка») — среда, в которой происходит действие; место, время и условия действия. Сеттинг может рассматриваться в настольных и компьютерных играх, в фильмах, художественных произведениях, новостях и др. Описывая сеттинг, пользователь определяет свойства реальности. (Wikipedia) [↑](#endnote-ref-2)