!!заполнить гугл-форму!!

Тайлы

Никакая игра не может существовать без дизайна. Хотя может, к примеру змейка. Но это был 1978 год. И то минимальный дизайн там был.

Знакомство с системой тайлов в Unity даёт отличную возможность экономии времени разработчикам игр.

Что такое тайловая игра?

Тайловая 2D-игра — это любая игра, в которой уровни или игровые области состоят из множества небольших плиток (тайлов), вместе образующих сетку тайлов. Иногда различия между тайлами могут быть очевидными, а иногда они кажутся игрокам сплошными и неразличимыми.

Коллекция имеющихся в игре тайлов называется «тайлсетом», и каждый тайл обычно является спрайтом
обычно тайлы являются квадратами. Но они могут принимать и другую форму — прямоугольники, параллелограммы или шестигранники. В играх обычно

используется вид сверху или сбоку, но иногда в тайловых играх применяется и 2.5D.

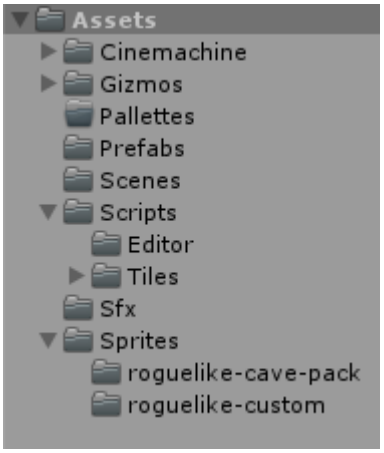
Возможно, вам уже известны две самые популярные игры, в которых используется система тайловых карт: *Starbound* и *Terraria*.

Практическое задание:

Скачайте материалы проекта для этого урока и распакуйте файл .zip.

Запустите редактор Unity и загрузите проект *starter* из распакованных материалов проекта.

Вот, с чем вы будете работать в этом проекте:



- Cinemachine/Gizmos: Unity Cinemachine добавлен только для реализации простой в использовании камеры, следующей за игроком.
- Palettes: в этой папке мы будем хранить собственные палитры тайлов.
- Prefabs: несколько заранее созданных заготовок-префабов, которые мы позже используем в игре.
- Scenes: здесь мы будем открывать и сохранять сцены.
- Scripts: несколько простых скриптов для управления движением игрока, логики коллизии ловушек и победы/проигрыша в игре. Также в этой папке содержатся скрипты Unity Tilemap из Github-репозитория Unity 2D Extras, которые мы применим позже.
- Sfx: звуки для игры.
- Sprites: 2D-спрайты, из которых мы будем создавать палитры тайлов.

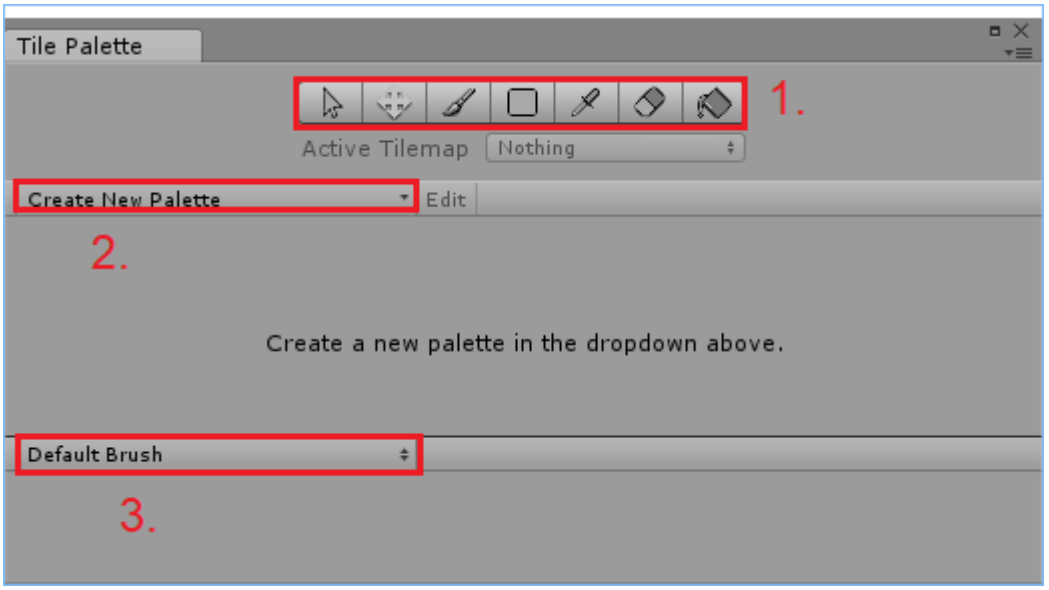
Задание:
Откройте сцену *Game* из папки *Scenes*.

Нажмите на кнопку *Play* в редакторе, чтобы запустить игру. В окне *Game* перемещайте героя клавишами WASD или «стрелками».

герой бродит по кажущемуся бесконечным фону камеры с цветом #00000, потерявшись в пустоте.
Необходимо это исправить с помощью тайлов.

Знакомимся с палитрой тайлов

В редакторе нажмите *Window -> Tile Palette*, чтобы открыть окно 2D Tile Palette.



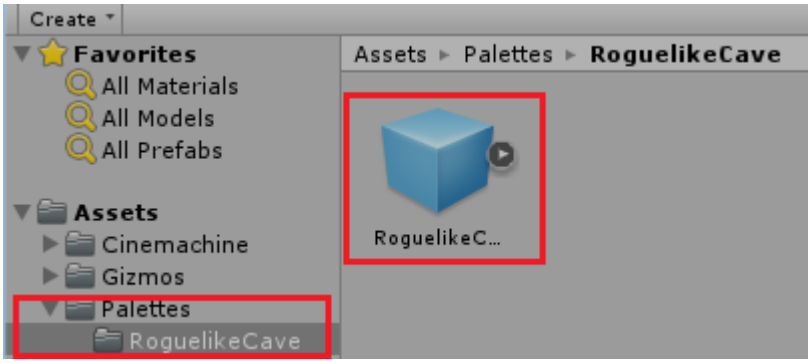
Это окно станет вашим лучшим другом при работе над тайловыми играми в Unity.

1. Этот ряд значков предоставляет основные инструменты манипуляций тайлами, рисования и удаления

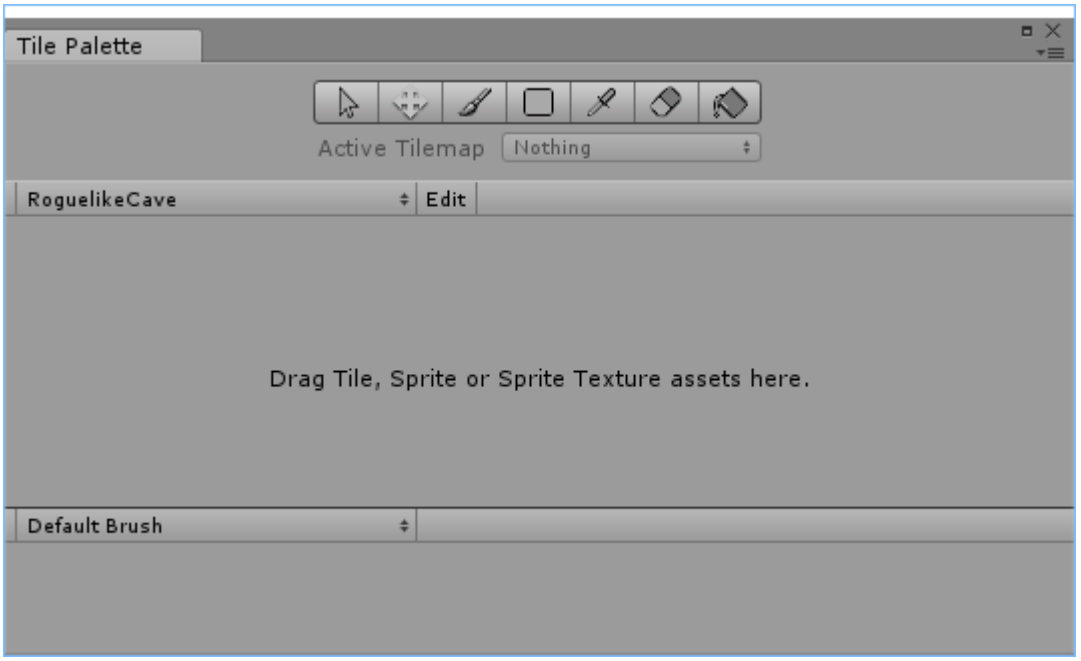
2. Этот селектор позволяет создавать разные палитры, которые можно воспринимать как палитры для рисования, в которых мы располагаем «цвета», а в этом случае — тайлы.
3. Этот селектор позволяет создавать кисти с различными поведениями. Можно добавлять собственные кисти, поведение которых отличается от кисти по умолчанию (например, рисующую тайлы-префабы с дополнительным функционалом)

Нажмите на *Create New Palette* и назовите палитру *Roguelike Cave*. Опции сетки и ячеек не изменяйте.
Нажмите на *Create* и выберите сохранение новой палитры в папке *Assets\Palettes* проекта. В ней создайте новую папку *RoguelikeCave*.

Теперь структура папок вашего проекта должна выглядеть так:



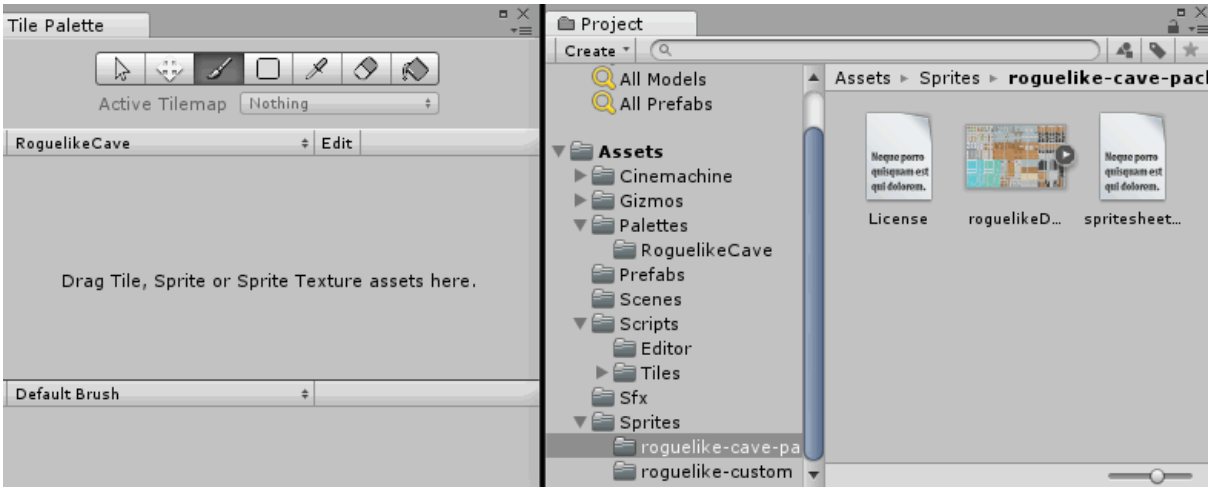
В окне редактора *Tile palette* должна быть выбрана *RoguelikeCave*; на этом этапе у нас всё ещё нет никаких тайлов:



Как художник может творить свои шедевры, если у него нет материалов?

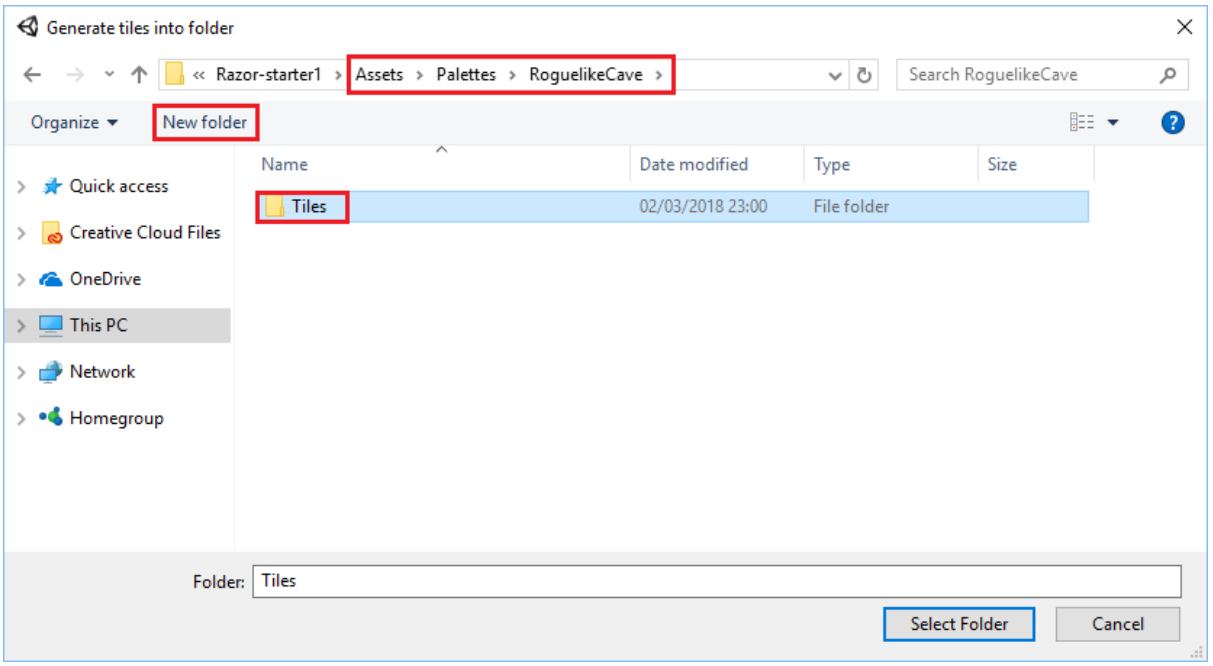
Не закрывая окно Tile Palette, выберите папку проекта *Sprites/roguelike-cave-pack* и разверните ассет *roguelikeDungeon transparent.png*. Затем выделите все спрайты в этом листе спрайтов: выберите первый спрайт, зажмите shift и выберите последний спрайт.

Перетащите все выбранные спрайты в окно Tile Palette Roguelike Cave:



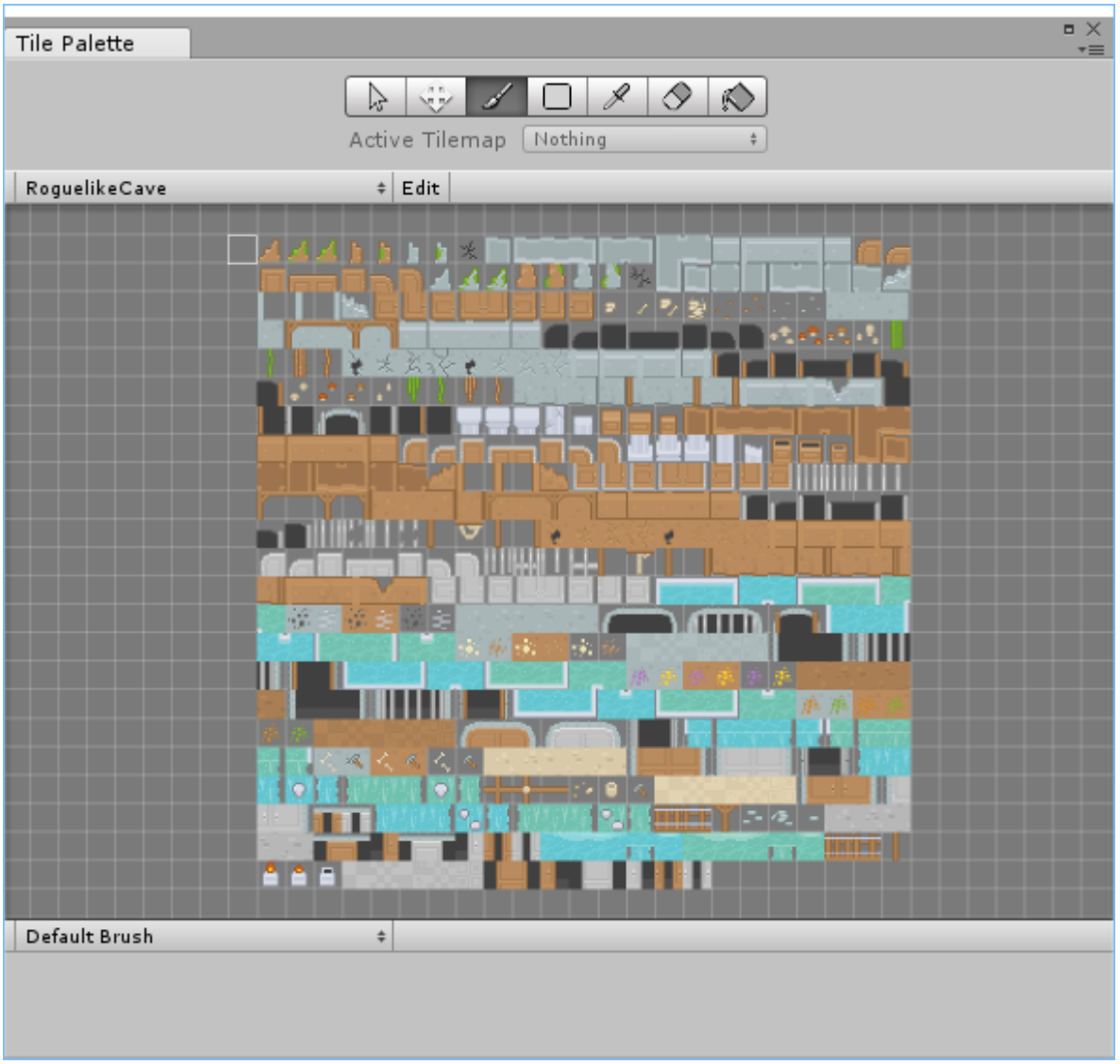
Перетащив спрайты в окно Tile Palette, выберите в Unity место для хранения ассетов.

Создайте в *Assets/Palettes/RoguelikeCave* новую папку *Tiles* и выберите эту папку в качестве места хранения:



Unity сгенерирует тайловый ассет для каждого спрайта, добавленного из листа спрайтов. Дождитесь завершения процесса, затем увеличьте размер окна Tile

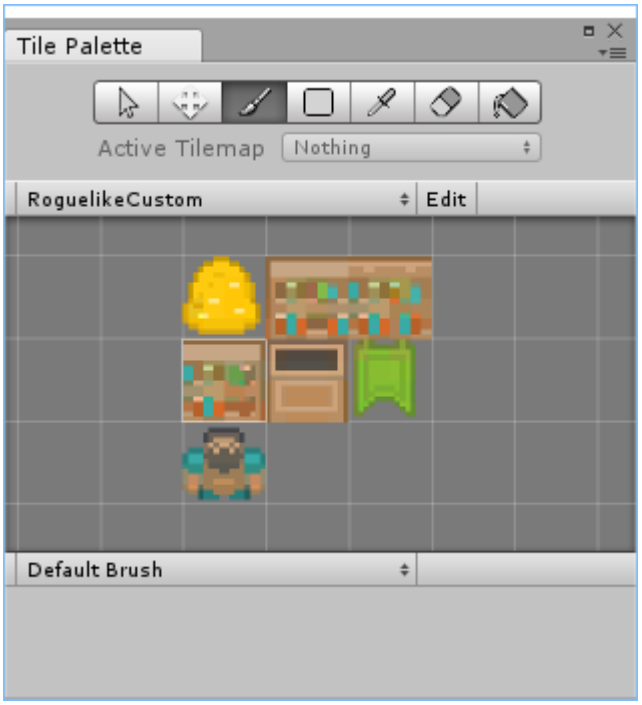
Palette и полюбуйтесь на ровные ряды красивых новых тайлов, расположившихся в палитре RoguelikeCave:



Повторите описанный выше процесс для создания палитры тайлов с помощью окна *Tile Palette*, но на этот раз назовите новую палитру *RoguelikeCustom*.

Поместите новую палитру в новую папку. Назовите папку *RoguelikeCustom* и переместите её в папку *Assets/Palettes* проекта.

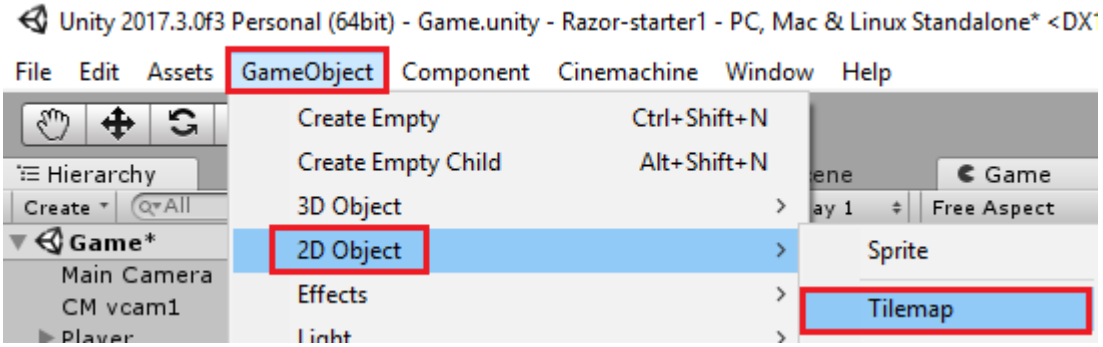
На этот раз, воспользовавшись описанным выше процессом, используйте спрайты из листа *Assets/Sprites/roguelike-custom/roguelike-normal-cutdown-sheet.png* для заполнения тайлами новой палитры. Создайте внутри папки палитры *RoguelikeCustom* папку *Tiles* и переместите ассеты тайлов туда:



Поздравляю, теперь вам известна магия создания тайловой палитры!

Создание сетки карты тайлов

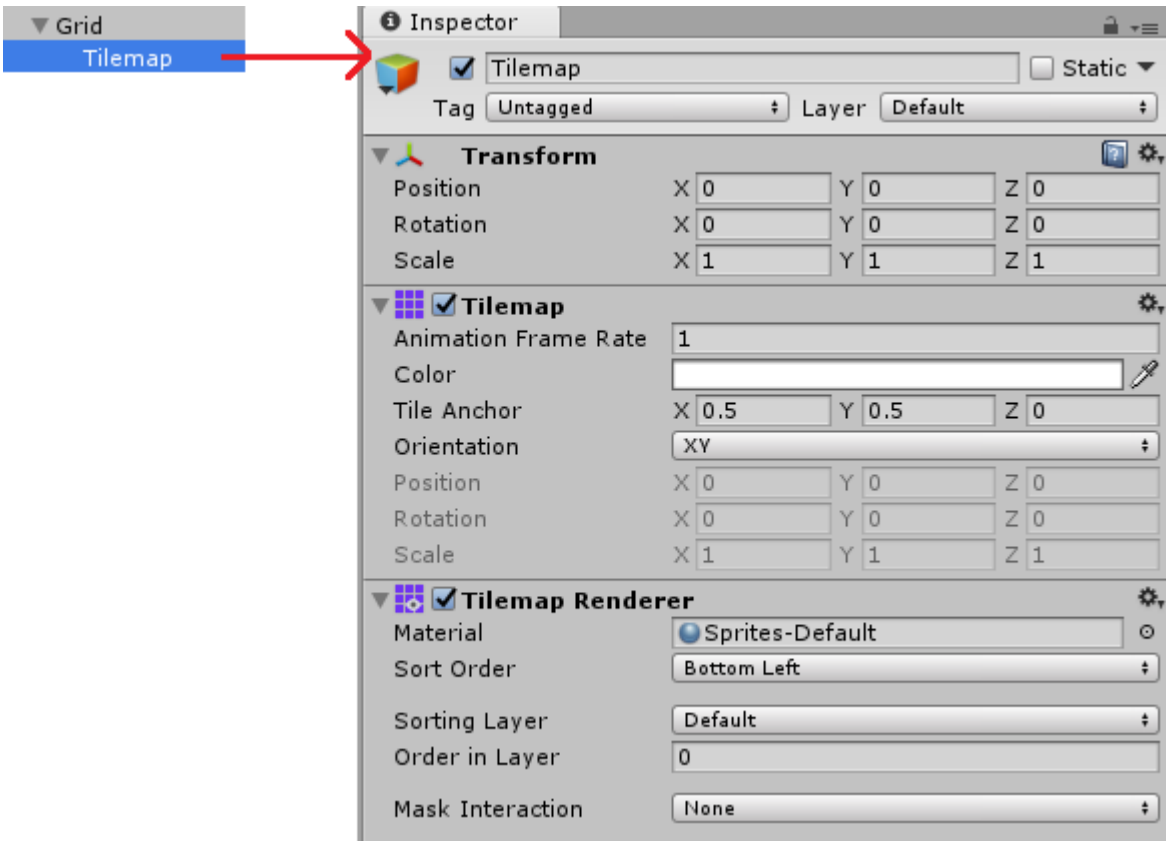
Откройте меню *GameObject* в верхней части редактора Unity , нажмите на *2D Object*, а затем *Tilemap*, чтобы создать новую сетку Tilemap:



Вы должны увидеть, что в иерархию сцены добавился новый GameObject *Grid*. Разверните его и выберите встроенный GameObject *Tilemap*.

Воспринимайте этот объект *Tilemap* как слой (возможно, один из многих) вашей игры. Можно добавить новые объекты для создания дополнительных слоёв Tilemap.

В инспекторе вы увидите два компонента, которые Unity автоматически добавила к этому GameObject:



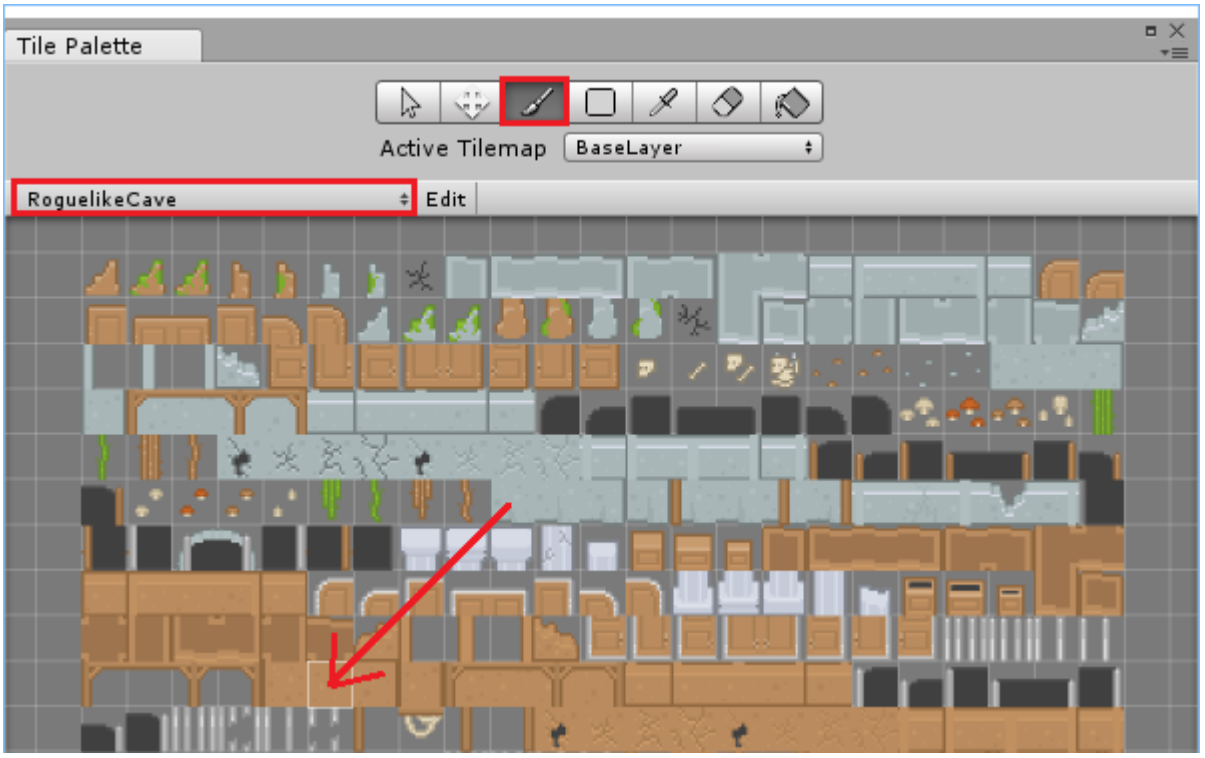
- Компонент `Tilemap` используется движком Unity для хранения спрайтов в схеме, помеченной компонентом `Grid` — в нашем случае это GameObject *Grid*. При первом создании Tilemap не стоит особо беспокоиться о всех технических особенностях того, как Unity связи между всеми этими компонентами.
- `Tilemap Renderer` назначает материал, который будет использоваться для рендеринга тайлов в Tilemap. Также он позволяет настроить свойства сортировки этого слоя Tilemap.

Переименуйте GameObject *Tilemap* в *BaseLayer*.

Использование различных инструментов рисования палитры тайлов

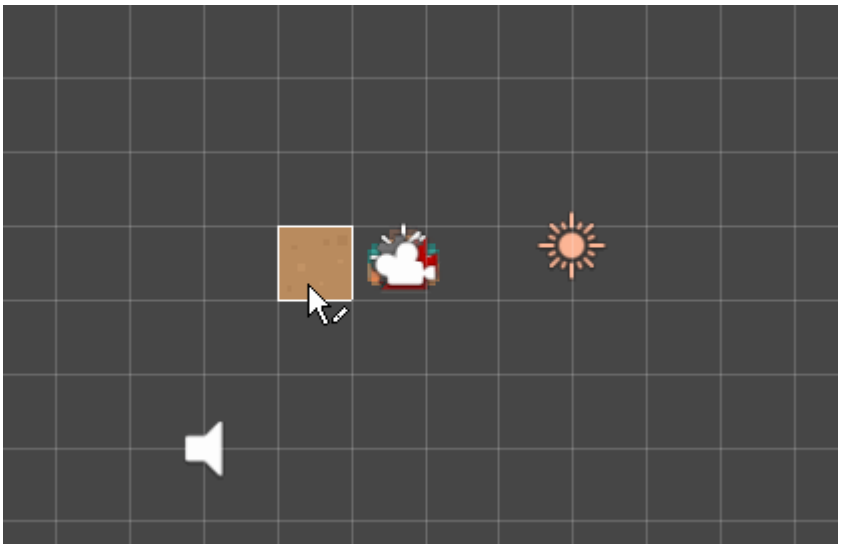
Переключитесь в редакторе на режим Scene.

Не закрывая окно Tile Palette, выберите палитру RoguelikeCave, а затем выберите инструмент *brush* (или нажмите *B*). Выберите тайл песка, как показано ниже:



В окне Scene переместите курсор на сетку рядом с игроком. Кисть с тайлом песка будет привязываться к сетке.

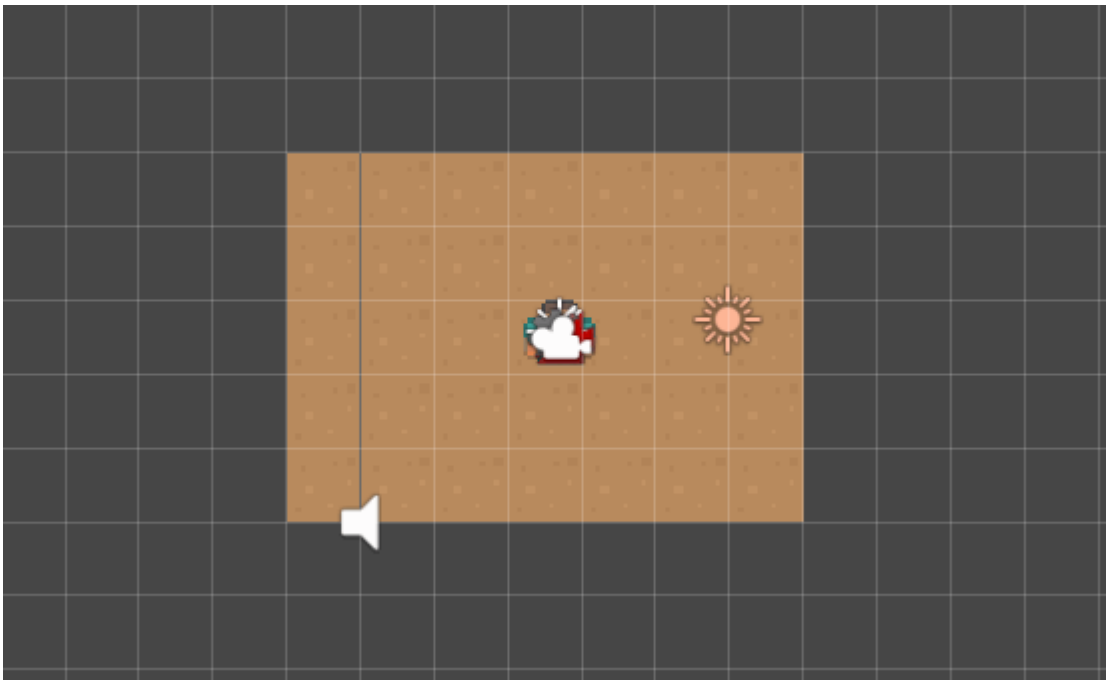
Зажав и удерживая левую клавишу мыши, нарисуйте вокруг игрока прямоугольную область. Она будет отрисована на слое Tilemap BaseLayer:



Рисование больших областей может оказаться монотонным занятием, поэтому существует кисть *Filled Box*, которую можно использовать для закрашивания крупных площадей. В окне Tile Palette нажмите на *квадратный значок кисти* (или нажмите *U*).

Вернитесь в редактор и нарисуйте вокруг игрока прямоугольник ещё большего размера, нажав и удерживая левую клавишу мыши, перетаскивая курсор из

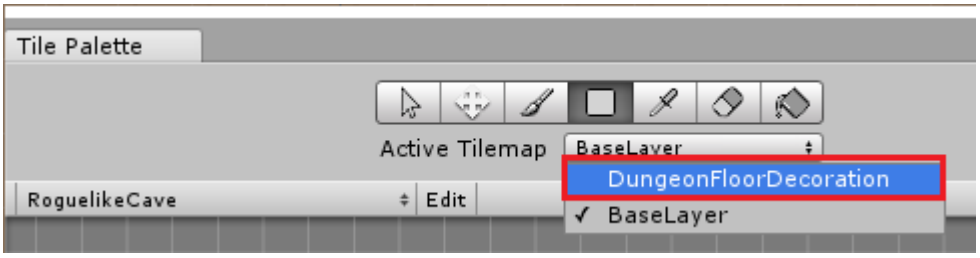
верхнего левого в нижний правый угол:



Хотя мы добавили в игру немного цвета, это песчаное подземелье выглядит уныло. Настало время добавить немного деталей!

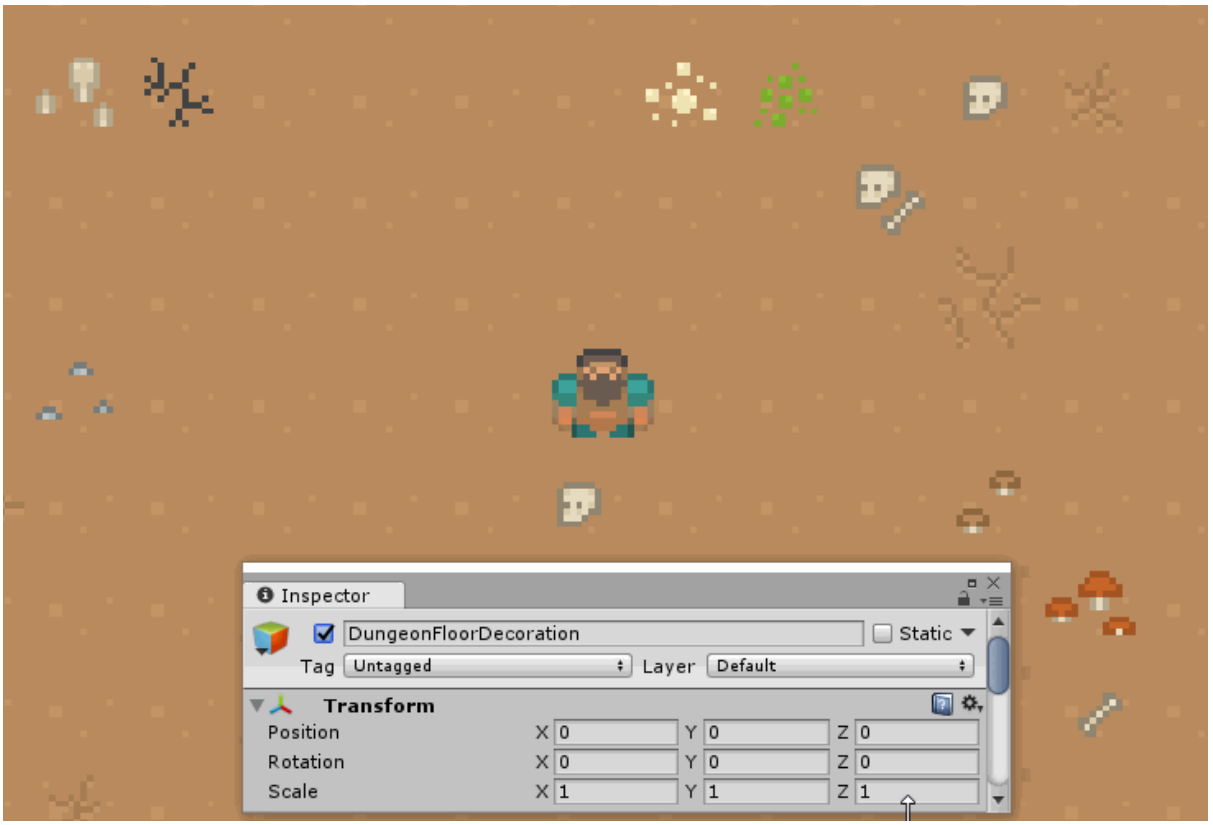
Используйте опцию меню *GameObject -> 2D Object -> Tilemap* для создания нового слоя Tilemap. На этот раз это будет единственный созданный в иерархии объект, потому что у нас уже есть подходящая Grid. Переименуйте этот слой в *DungeonFloorDecoration*:

В окне Tile Palette переключите *Active Tilemap* на слой *DungeonFloorDecoration*:



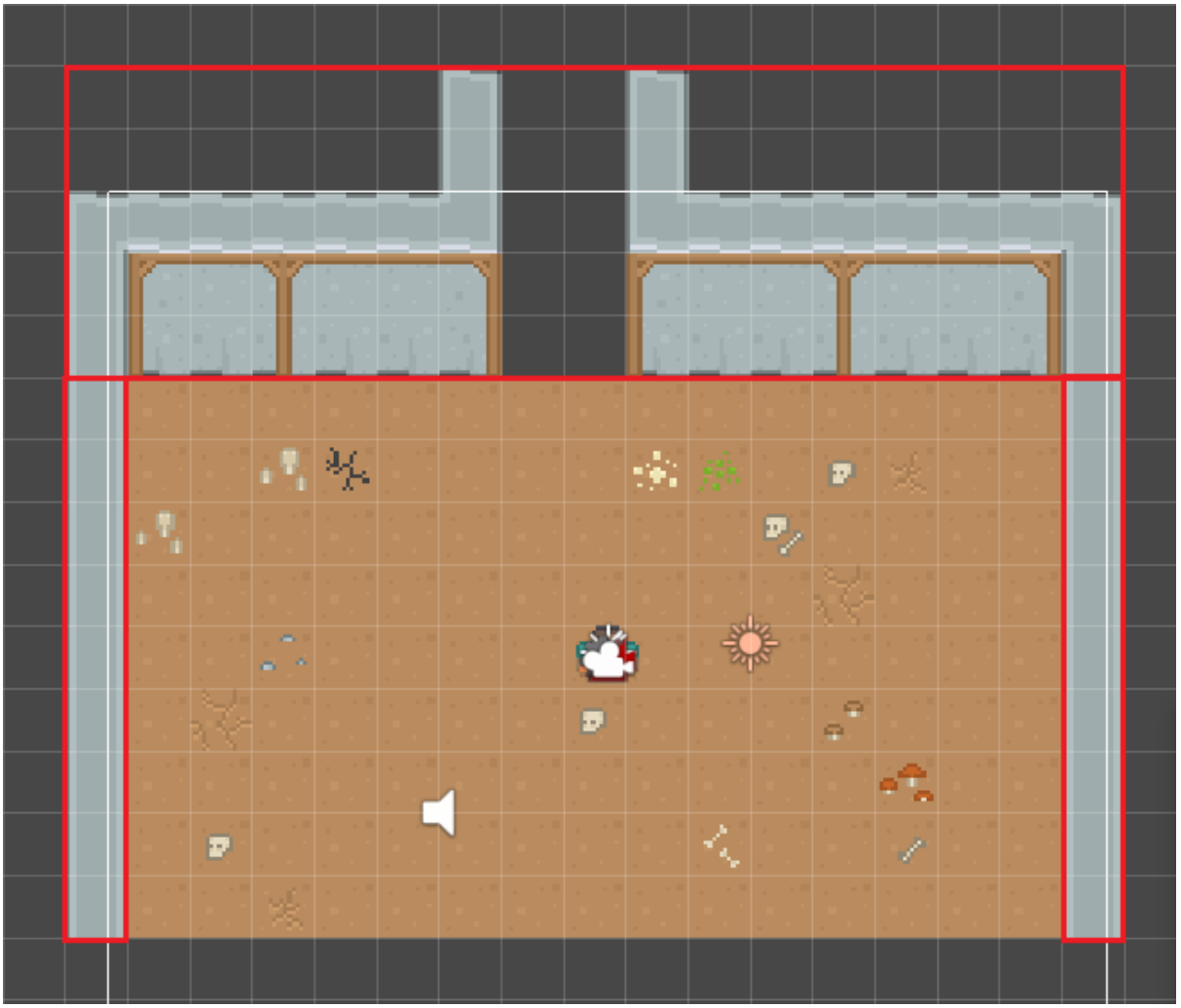
Выберите инструмент *brush* (*B*), затем нарисуйте в окне Scene разбросанные на карте объекты:

Отключите, а затем снова включите в иерархии GameObject *DungeonFloorDecoration*, чтобы увидеть, как отрисовка на активном Tilemap изменяет слой DungeonFloorDecoration, а все отрисованные тайлы попадают на этот новый слой:



Создайте новый слой Tilemap, снова воспользовавшись опцией *GameObject -> 2D Object -> Tilemap*. Назовите его *Collideable*. В дальнейшем мы используем его для создания стен и границ.

Переключите Active Tilemap в окне Tile Palette на *Collideable*. Выберите инструмент brush (*B*), а затем нарисуйте следующие тайлы, чтобы построить вокруг игровой области стену. Выделенные красным зоны на изображении ниже — это новые части, которые нужно добавить:

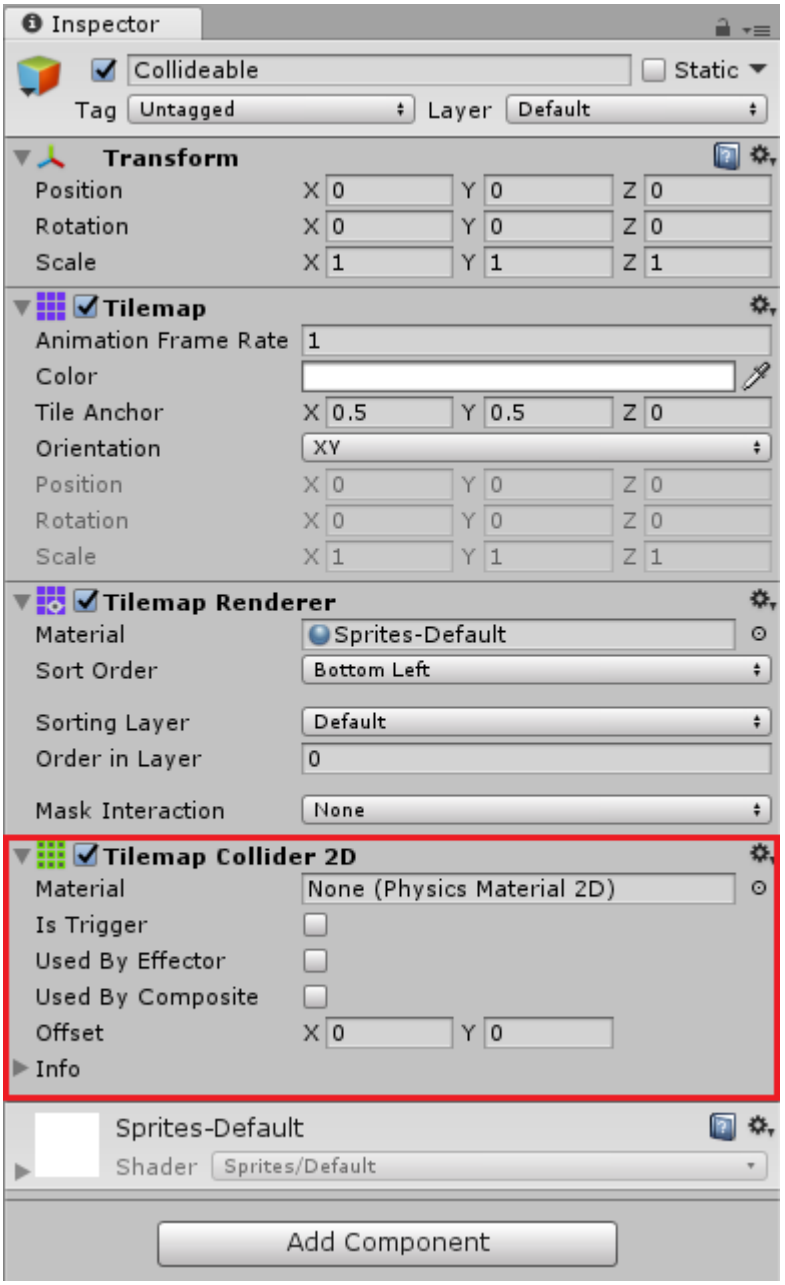


Посмотрите на показанный ниже скриншот окна Tile Palette, чтобы разобраться, где найти тайлы, необходимые для постройки стены. Не забывайте, что можно использовать сочетания CTRL-Z или CMD-Z для отмены действия или стирать ошибки с помощью текущей кисти (удерживая *Shift*):

Запустите игру в редакторе и попытайтесь пройти сквозь стену

Что-то не так.
Проблема в том, что мы просто нарисовали стандартные тайлы и пока не применяли к слою Tilemap волшебную физику Unity.

Выберите GameObject *Collideable* и добавьте новый компонент, нажав кнопку *Add Component* в окне *Inspector*; в поле поиска введите *Tilemap Collider 2D*:

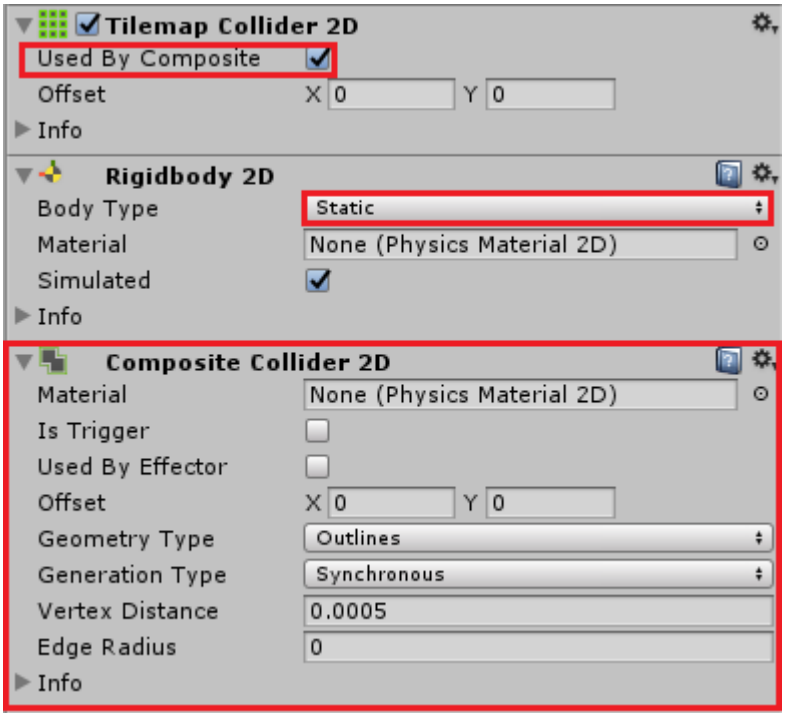


Этот компонент был создан специально для тайловых 2D-игр на Unity. Он просто применяет форму физического коллайдера ко всем тайлам слоя, к которому он был добавлен, не выполняя никакой другой работы.

Снова запустите игру и попробуйте пройти сквозь стену.

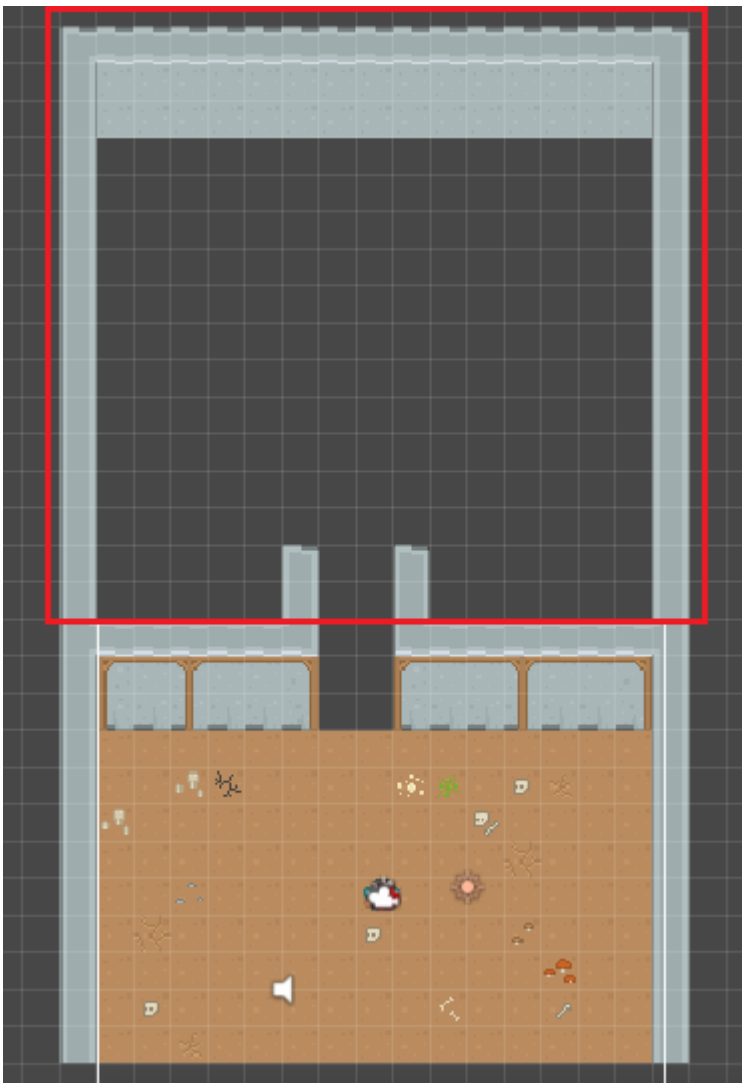
Выбрав GameObject *Collideable*, добавьте к нему компонент *Composite Collider 2D*. Это также автоматически добавит RigidBody2D.

Задайте параметру *BodyType* RigidBody2D значение *Static*, а затем поставьте флажок *Used by Composite* в компоненте *Tilemap Collider 2D*:



После этого вы заметите, что эти ненужные квадратные коллайдеры посередине стен исчезнут.

Завершите создание стен, достроив их вверх и замкнув наверху, высотой примерно в 16 тайлов. Не забывайте. что в качестве Active Tilemap окна Tile Palette должен быть выбран *Collideable*:

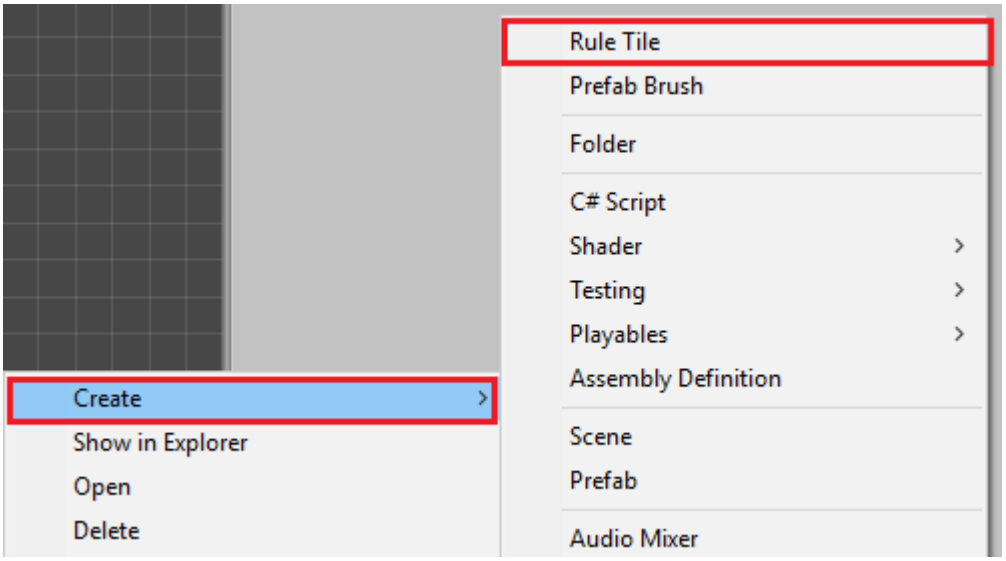


Участок подземелья не будет представлять никакой сложности для нашего героя без препятствий. Теперь мы начнём работу над созданием комнаты смерти, дополненной красивыми древними мраморными коридорами. После преодоления всех этих препятствий игрока ждёт награда — гора золота.

Для отрисовки этих коридоров мы воспользуемся специальной тайловой кистью *Rule Tile*. Как вы видели в начале туториала, в проект уже добавлены дополнительные тайловые скрипты из Github-репозитория Unity 2D Extras. Одним из них является Rule Tile.

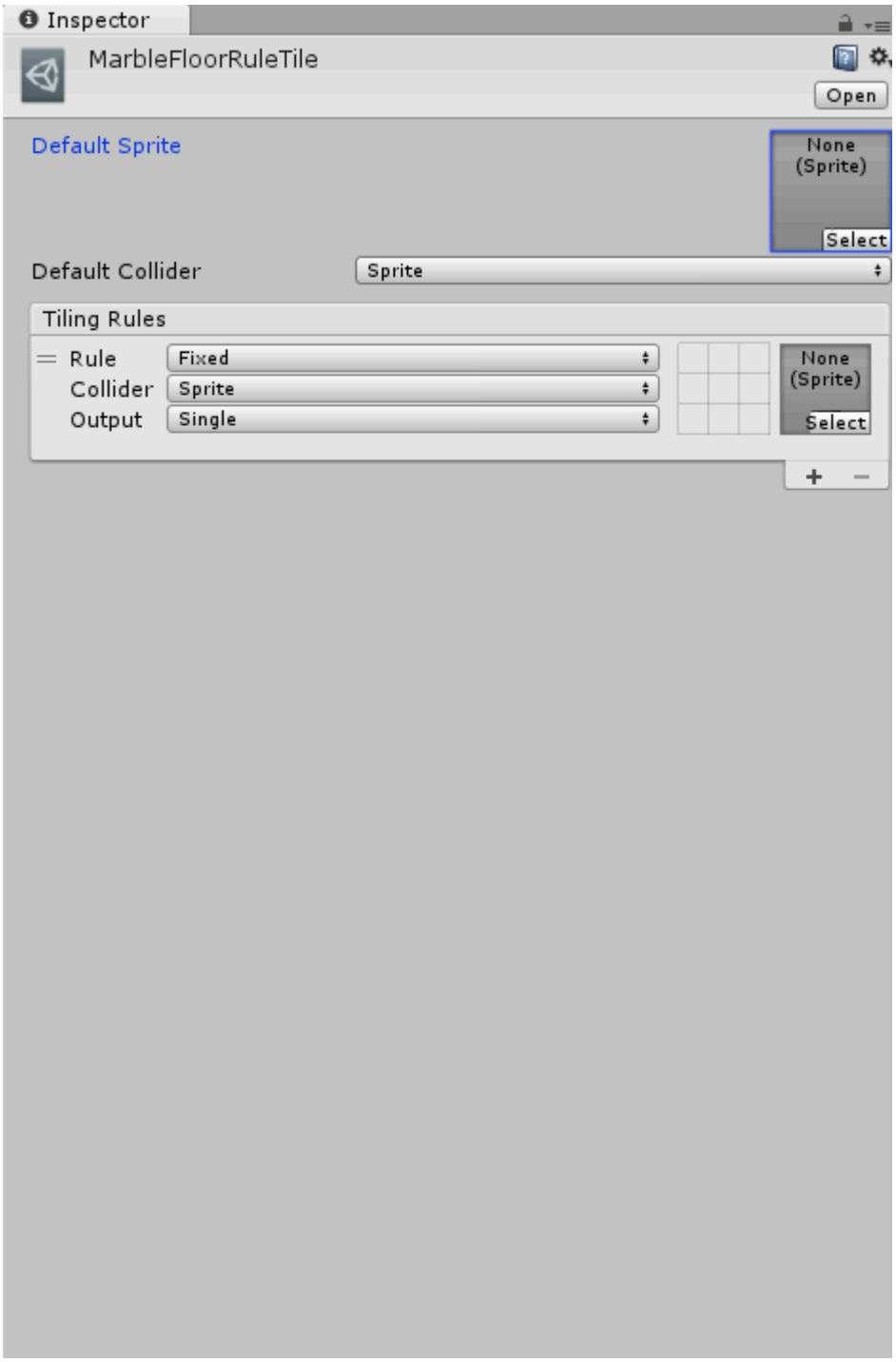
Rule Tile позволяет нам задавать правила о том, какие тайлы нужно отрисовывать в зависимости от соседних располагаемых нами тайлов.

Нажмите правой клавишей мыши на папке *Prefabs* проекта и выберите *Create -> Rule Tile* (этот пункт должен быть в верхней части меню). Назовите новый элемент *MarbleFloorRuleTile*:



Выберите этот новый *MarbleFloorRuleTile* и используйте инспектор, чтобы присвоить *Default Sprite* значение `roguelikeDungeon_transparent_335`. Затем добавьте новое правило Tiling Rule, нажав на значок +. Выберите для *Sprite* этого правила значение `roguelikeDungeon_transparent_339` и нажмите на все внешние квадраты в схеме правила, чтобы на каждом была

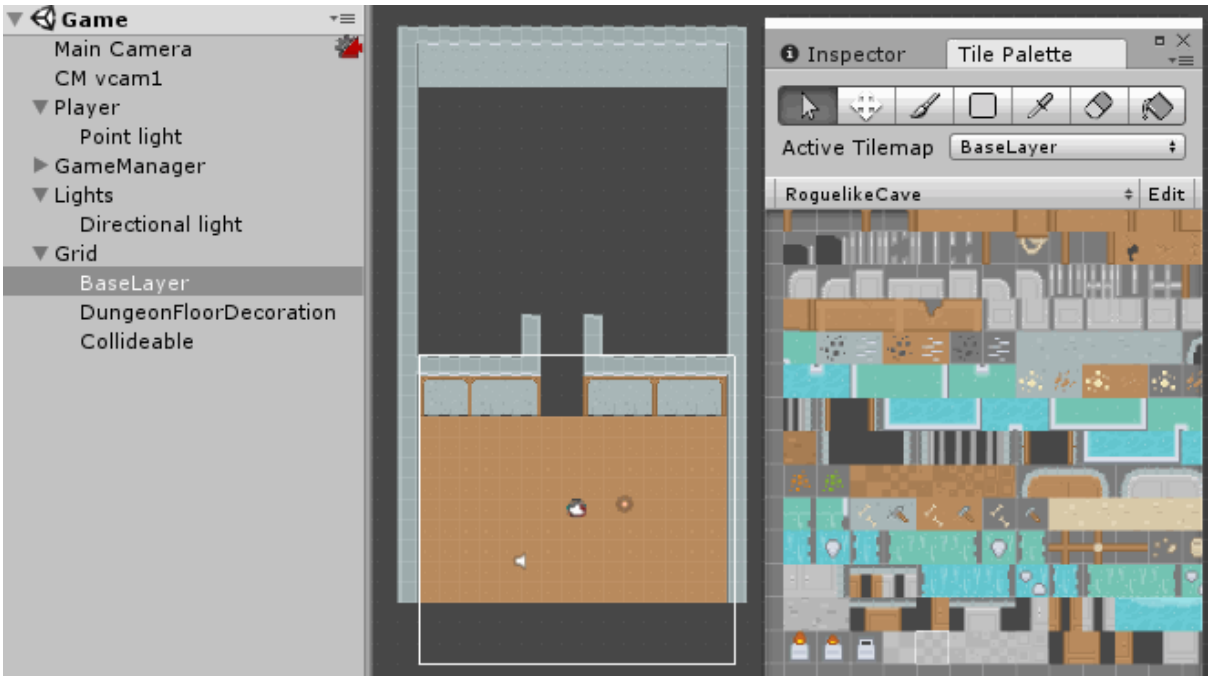
указывающая наружу зеленая стрелка:



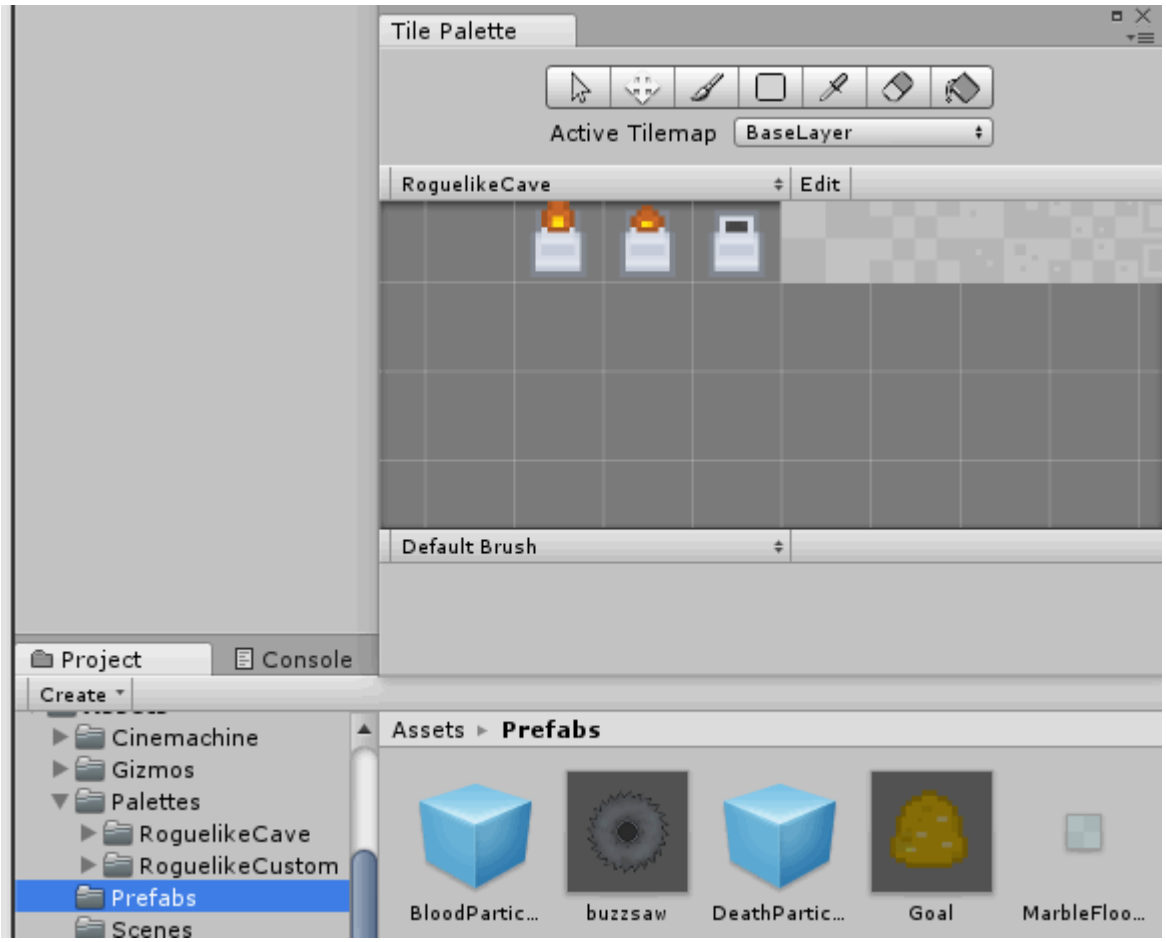
Воспользовавшись инструментом box fill brush (*B*) в окне Tile Palette и выбрав слой Tilemap *BaseLayer*, нарисуйте прямую секцию мраморной стены. Нужно, чтобы она закрывала всё пока свободное пространство пола.

Можно заметить, что когда мы будем это делать, слой будет закрывать тайлы стен с коллайдерами, потому что пока не задан порядок слоев. Это легко исправить, выбрав GameObject *Collideable* и изменив *Order in Layer* компонента

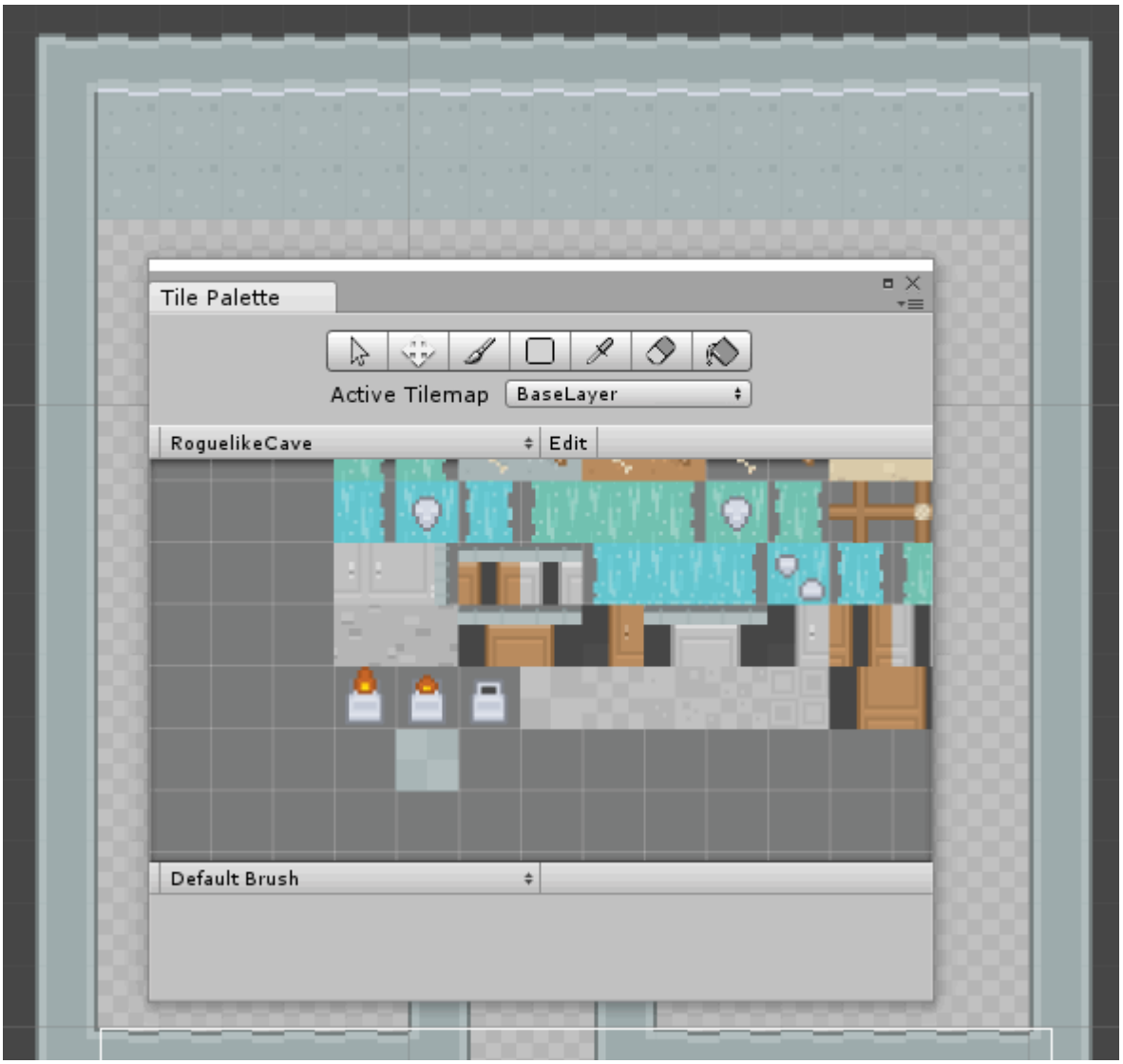
Tilemap Renderer на более высокое значение (достаточно будет *5*):



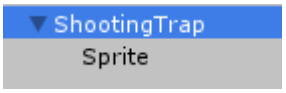
Вернитесь в папку *Prefabs* проекта, откройте окно Tile и выберите палитру RoguelikeCave, а затем перетащите *MarbleFloorRuleTile* в пустое место на палитре:



Воспользуйтесь box fill brush и нарисуйте в комнате несколько секций мраморного пола:

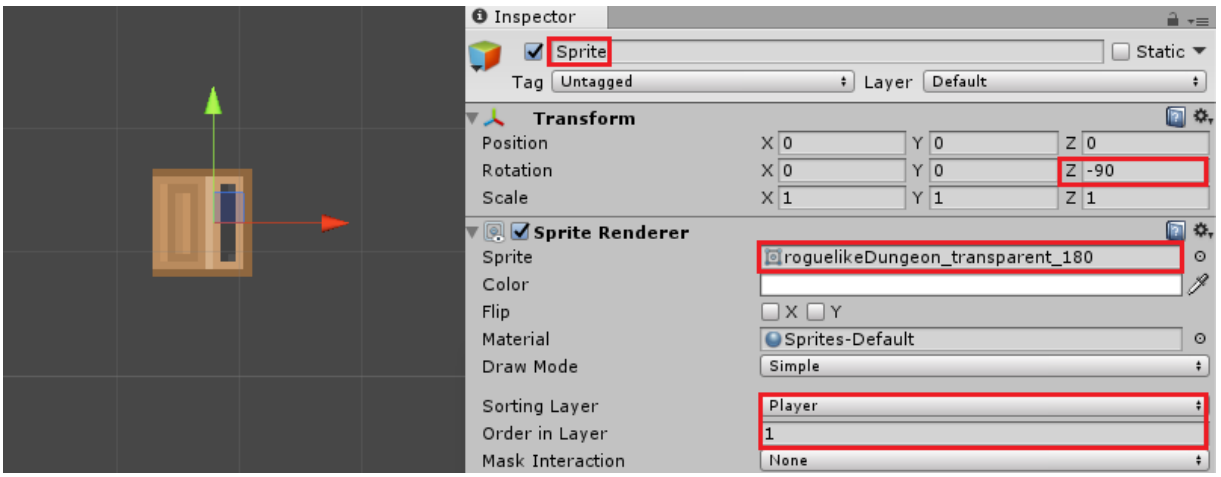


Создайте в иерархии новый пустой GameObject и назовите его *ShootingTrap*. Создайте в ShootingTrap пустой дочерний GameObject. Назовите его *Sprite*:



Выберите *Sprite* и добавьте к нему компонент *Sprite Renderer*. Задайте *Sorting Layer* значение *Player*, а *Order in Layer* значение *1*, чтобы он рендерился поверх остальных слоёв. Выберите поле *Sprite*, а в качестве спрайта поставьте *roguelikeDungeon_transparent_180*.

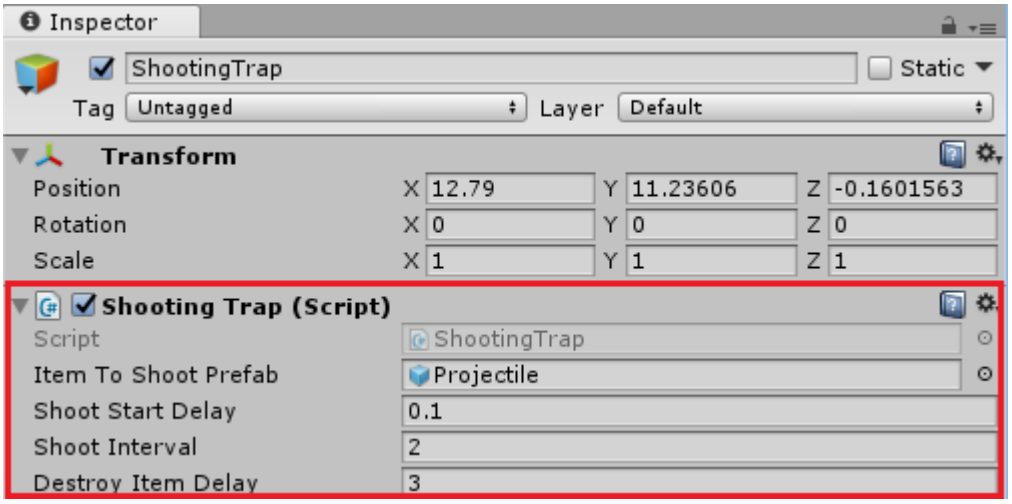
Теперь поверните Transform объекта *Sprite* на *-90* по оси *Z*:



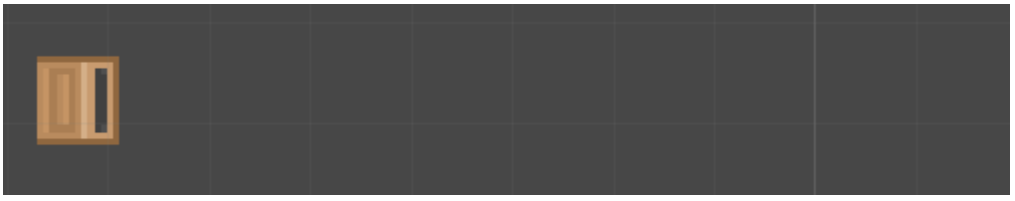
Далее вернитесь к GameObject *ShootingTrap* и добавьте с помощью инспектора новый компонент. В поле поиска найдите *Shooting Trap* и прикрепите этот скрипт.

Этот скрипт добавлен в скачанные вами файлы проекта; по сути, он каждые две секунды запускает корутину, создающую экземпляр префаба вращающегося лезвия пилы (или любого другого префаба) в текущей позиции ловушки.

Задайте параметру *Item to Shoot Prefab* компонента Shooting Trap значение *Projectile* (префаб находится в */Assets/Prefabs*):



Снова запустите игру в редакторе и воспользуйтесь режимом Scene, чтобы найти ловушку. Она работает!

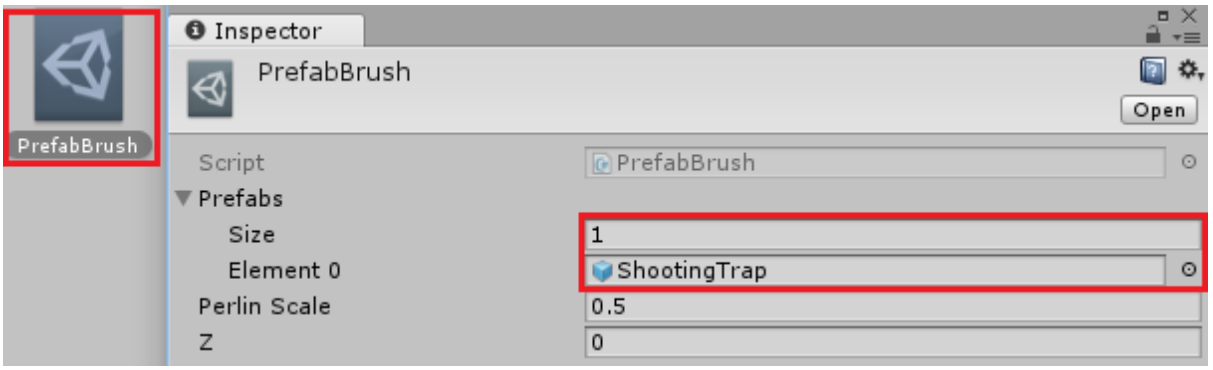


Перетащите копию *ShootingTrap* из иерархии в папку */Assets/Prefabs* проекта, чтобы создать префаб. Удалите ShootingTrap из иерархии.

Мы используем ещё один скрипт тайловой кисти под названием *PrefabBrush* для создания кисти, способной рисовать префабы на слоях Tilemap.

Нажмите правой кнопкой на папке */Assets/Prefabs* проекта и выберите *Create -> Prefab Brush*. Назовите объект *PrefabBrush*.

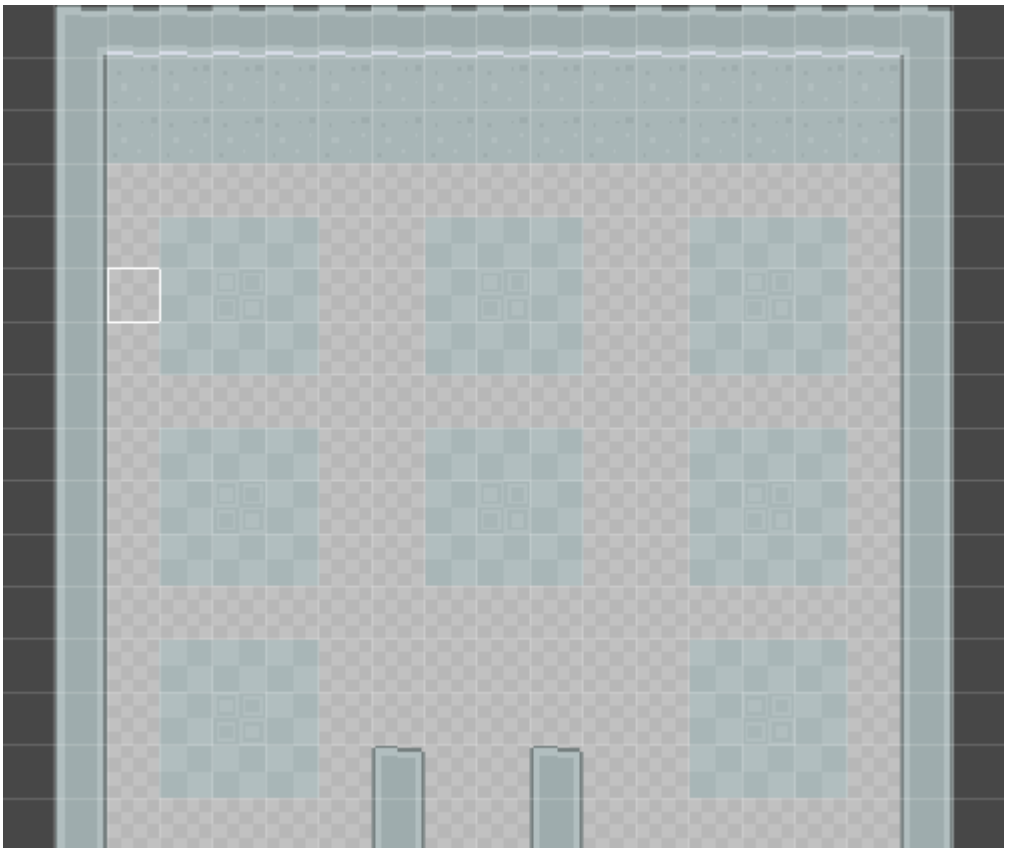
Воспользуйтесь инспектором, чтобы задать параметру *Prefabs Size* PrefabBrush значение *1*, а параметру *Element 0* — значение *ShootingTrap*.



Создайте в *Grid* новый слой Tilemap под названием *Traps* и откройте окно Tile Palette.

Выберите обычную тайловую кисть (*B*) и воспользуйтесь раскрывающимся меню в нижней части окна Tile Palette для выбора *PrefabBrush*. Выберите в качестве слоя Active Tilemap *Traps* и используйте окно Scene для отрисовки

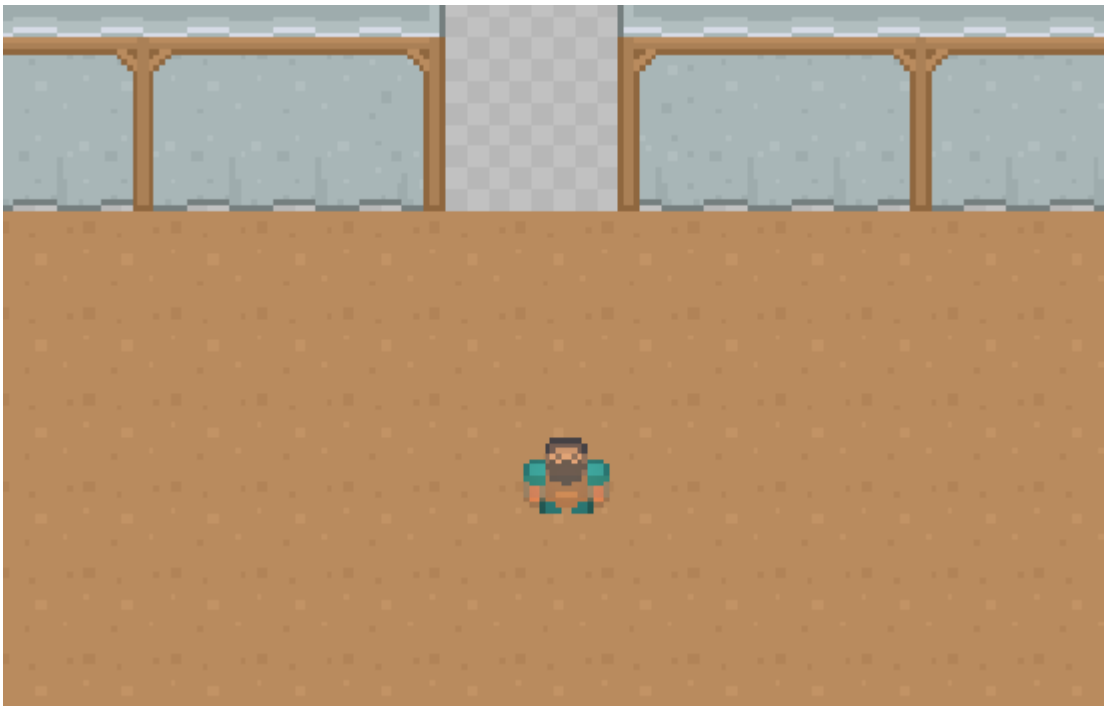
нескольких префабов ловушек вдоль левой границы комнаты.



Разверните в иерархии GameObject *Traps* и поэкспериментируйте со значением *Shoot Start Delay* для каждого value on each Gameobject *ShootingTrap* с помощью скрипта Shooting Trap в инспекторе. Добавляйте к значению каждой ловушки по 0.25, т.е.:

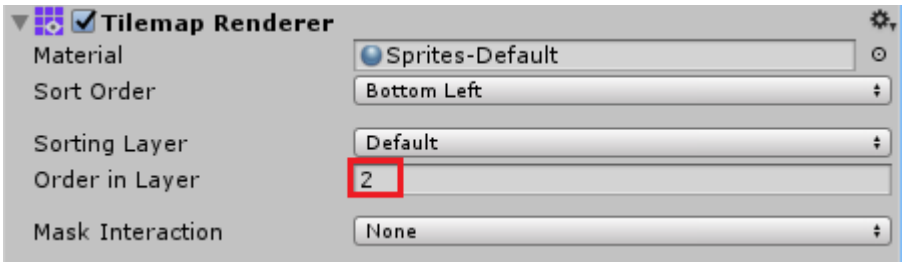
- Первая ShootingTrap -> Shoot Start Delay = 0.1
- Вторая ShootingTrap -> Shoot Start Delay = 0.35
- Третья ShootingTrap -> Shoot Start Delay = 0.6
- И так далее...

Запустите игру и пройдите испытание, если осмелитесь.



Цель этого мини-подземелья заключается в получении груды золота. Слава и богатство ждут тех, кто доберётся до неё, избежав смертельно опасных летающих лезвий.

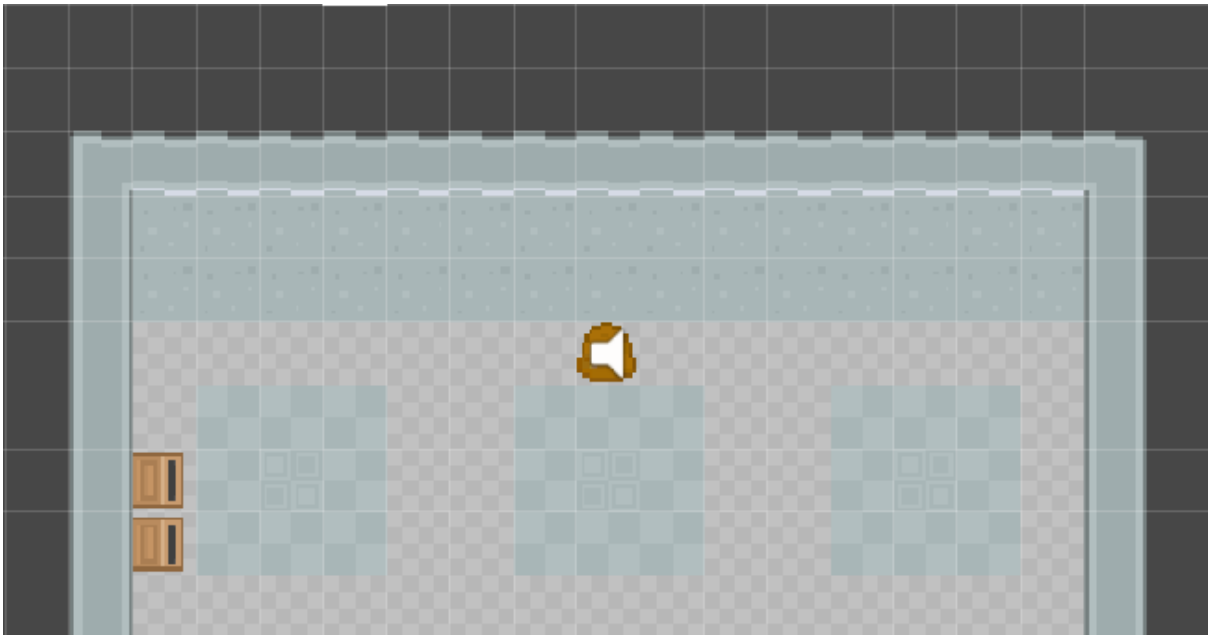
Создайте в GameObject *Grid* новый слой Tilemap под названием *Goal*. Выберите *Goal* и измените значение *Tilemap Renderer Order in Layer* на *2*:



Не закрывая окно Tile Palette, убедитесь, что всё ещё выбрана PrefabBrush. Сделайте так, чтобы *Element 0* ссылался на заготовку *Goal* из папки */Assets/Prefabs* проекта.

Это префаб со спрайтом горы золота, с Box Collider 2D со включенным режимом Is Trigger, добавленным источником звука, и с простым скриптом `Goal.cs`, воспроизводящим звук достижения цели и перезапускающим уровень, когда игрок попадает в область триггера.

Воспользуйтесь стандартной тайловой кистью для отрисовки одного тайла-префаба цели в верхней части комнаты за ловушками:

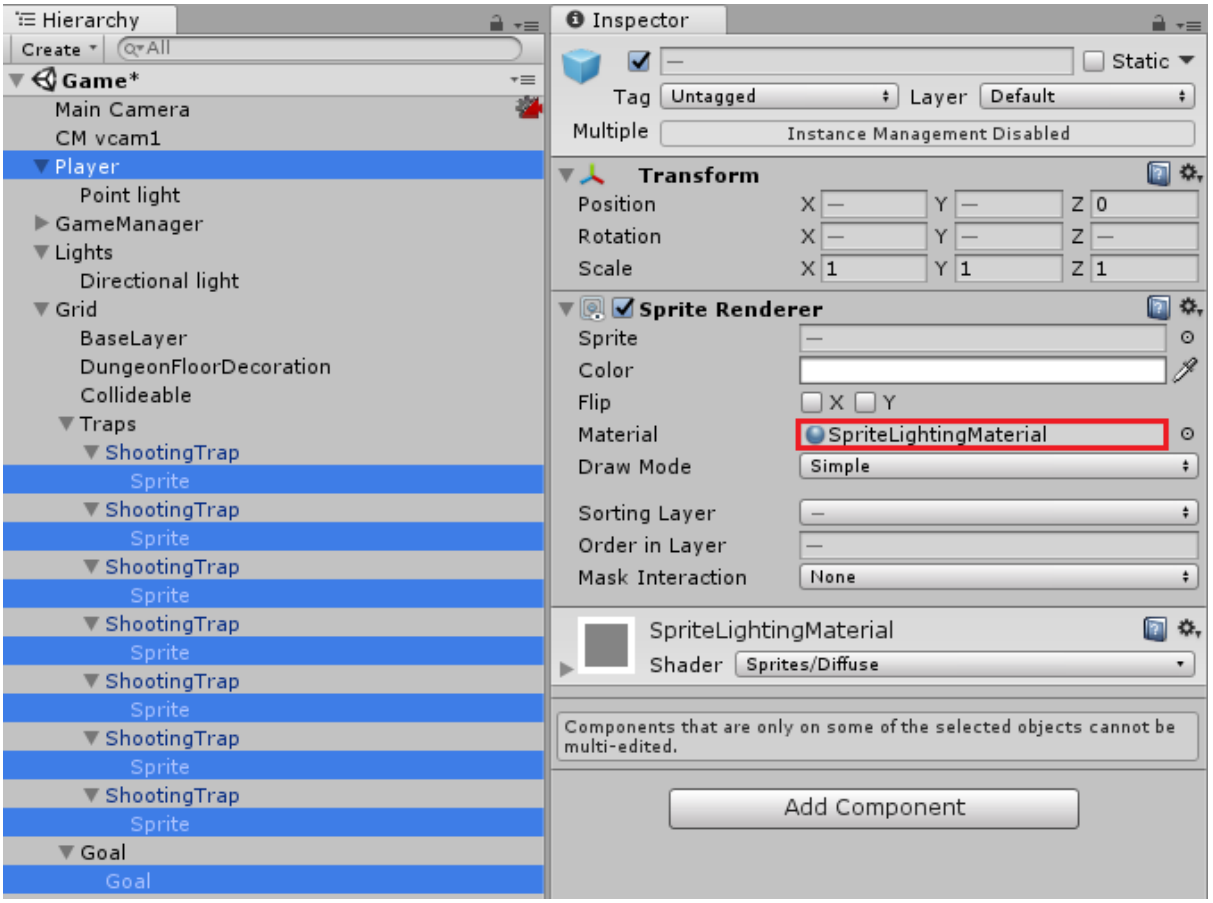


Снова запустите игру и попробуйте достичь цели. Когда вы доберётесь до этого тайла, запустится логика `OnTriggerEnter2D()` из `Goal.cs`, воспроизведя звуковой эффект и перезапустив уровень.

Последние штрихи

Сейчас подземелье слишком светлое и свободное. Мы можем добавить ему стиля, переключившись на материал 2D-спрайта, способный реагировать на свет.

Выберите Sprite объектов Player, Goal и ShootingTrap, и сделайте так, чтобы *Material* компонента Sprite Renderer использовал *SpriteLightingMaterial*:



Это материал с прикреплённым к нему шейдером *Sprite/Diffuse*. Он позволяет освещению сцены воздействовать на спрайты.

В GameObject *Grid* выберите объекты *BaseLayer, DungeonFloorDecoration, Collideable u Goal*, а потом воспользуйтесь инспектором, чтобы тоже использовать в материале *Tilemap Renderer Material SpriteLightingMaterial*.

Затем выберите в GameObject *Lights Directional light* и снизьте значение *Intensity* Light до *0.3*.

Так гораздо лучше!

Домашнее задание:

Создать карту для своей игры 🙂
Всем успехов!