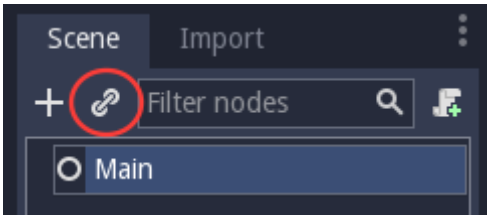
Главная сцена игры

Итак, настало время перенести всё, что мы сделали вместе, на играбельную игровую сцену.

Создайте новую сцену и добавьте Node с именем `Main`. (Причиной, по которой мы используем Node, а не Node2D, является то, что узел будет контейнером для обработки игровой логики. Это не требует именно двумерного функционала.)

Нажмите кнопку **«Экземпляр»** (представленную значком звена цепи) и выберите сохраненный файл `Player.tscn`.



Теперь добавьте следующие узлы в виде дочерних элементов `Main` и назовите их как показано ниже (значения указаны в секундах):

- Timer (назвать `MobTimer`) - чтобы контролировать частоту появления мобов
- Timer (назвать `ScoreTimer`) - чтобы каждую секунду увеличивать счет
- Timer (назвать `StartTimer`) - чтобы дать задержку перед стартом игры
- Position2D (назвать `StartPosition`) - чтобы указать начальную позицию игрока

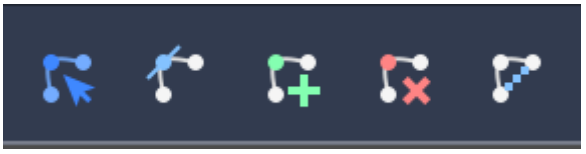
Задайте значение `Wait Time` для каждого из узлов `Timer` следующим образом:

- `MobTimer`: `0.5`
- `ScoreTimer`: `1`
- `StartTimer`: `2`

Кроме того, установите для свойства `One Shot` узла `StartTimer` значение "Вкл" и для свойства `Position` узла `StartPosition` установите значение `(240, 450)`.

Добавление мобов

Узел Main будет порождать новых мобов, и мы хотим, чтобы они появлялись в случайном месте на краю экрана. Добавьте узел Path2D с именем `MobPath` как дочерний элемент узла `Main`. Когда вы выберете `Path2D`, вы увидите несколько новых кнопок в верхней части редактора:



Выберите среднюю ("Add Point") и нарисуйте путь щелчками мыши, чтобы добавить точки в показанных углах. Чтобы точки привязывались к сетке, убедитесь, что выбраны "Use Grid Snap" и "Use Snap". Эти опции можно найти слева от кнопки "Lock", они отображаются в виде магнита рядом с точками и пересекающимися линиями соответственно.

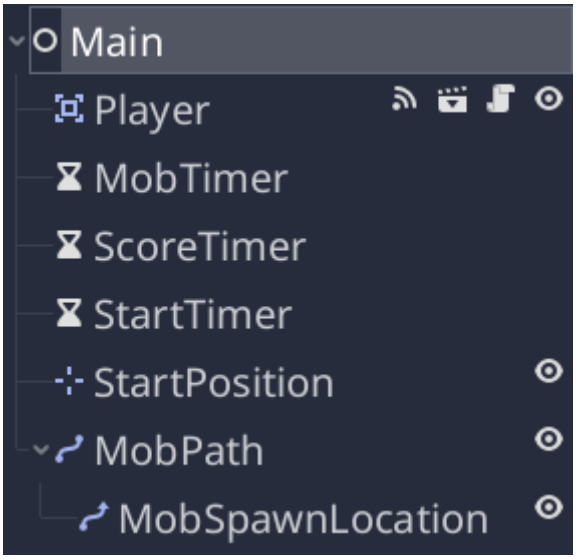
Важный

Нарисуйте путь в порядке *по часовой стрелке*, иначе ваши мобы будут появляться с направлением *наружу*, а не *внутрь*!

Поместив точку "4" на изображение, нажмите кнопку "Сомкнуть кривую", и она будет завершена.

Теперь, когда путь определен, добавьте узел PathFollow2D как дочерний элемент `MobPath` и назовите его `MobSpawnLocation`. Этот узел будет автоматически вращаться и следовать по пути при его перемещении, поэтому мы можем использовать его для выбора случайной позиции и направления вдоль пути.

Ваша сцена должна выглядеть так:



Главный скрипт

Добавьте скрипт к узлу `Main`. В верхней части скрипта мы пишем `export (PackedScene)`, что позволяет нам выбрать сцену Mob, экземпляр которой мы хотим сделать.

GDScript C# C++

```
extends Node

export(PackedScene) var mob_scene
var score
```

Мы также добавляем здесь вызов `randomize()`, чтобы случайный генератор чисел генерировал различные случайные числа каждый раз, когда запускается игра :

GDScript C# C++

```
func _ready():
    randomize()
```

Щелкните `Main` узел, и вы увидите свойство в Инспекторе в разделе «Переменные скрипта». `Mob Scene`

Значение этого свойства можно присвоить двумя способами:

- Перетащите `Mob.tscn` его из дока «FileSystem» в свойство **Mob Scene** .
- Нажмите стрелочку вниз рядом с "[пусто]" ("[empty]") и выберите "Загрузить" ("Load"). Затем выберите `Mob.tscn` .

Затем выберите узел `Player` ("Игрок") в панели "Сцена" ("Scene") и откройте вкладку "Узел" ("Node") в инспекторе. Убедитесь, что во вкладке "Узел" выбрана вкладка "Сигналы" ("Signals").

Вы должны увидеть список сигналов для узла `Player` . В списке найдите и дважды щелкните по сигналу `hit` (или щелкните по нему правой кнопкой мыши и выберите "Присоединить..."). Это откроет диалоговое окно подключения сигнала. Мы хотим создать новую функцию с именем `game_over` , которая будет обрабатывать то, что должно произойти, когда игра заканчивается. Введите "game_over" в поле "Метод-приёмник" в нижней части диалогового окна подключения сигнала и нажмите "Присоединить". Добавьте следующий код в новую функцию, а также функцию `new_game` , которая настроит всё для новой игры:

GDScript C# C++

```
func game_over():
    $ScoreTimer.stop()
    $MobTimer.stop()

func new_game():
    score = 0
    $Player.start($StartPosition.position)
    $StartTimer.start()
```

Теперь присоедините сигнал `timeout()` каждого из узлов Timer (`StartTimer` , `ScoreTimer` и `MobTimer`) к главному скрипту. `StartTimer` запустит два других таймера. `ScoreTimer` будет увеличивать счет на 1.

GDScript C# C++

```
func _on_ScoreTimer_timeout():
    score += 1
```

```
func _on_StartTimer_timeout():
    $MobTimer.start()
    $ScoreTimer.start()
```

В `_on_MobTimer_timeout()`, мы создадим экземпляр моба, выберем случайное начальное местоположение `Path2D` и запустим моба в движение. Узел `PathFollow2D` будет автоматически вращаться по пути, поэтому мы будем использовать его для выбора направления моба, а также его положения. Когда мы создаем моба, мы выбираем случайное значение между `150.0` и `250.0` для того, как быстро будет двигаться каждый моб (было бы скучно, если бы все они двигались с одинаковой скоростью).

Обратите внимание, что новый экземпляр должен быть добавлен в сцену с помощью функции `add_child()`.

GDScript C# C++

```
func _on_MobTimer_timeout():
    # Create a new instance of the Mob scene.
    var mob = mob_scene.instance()

    # Choose a random location on Path2D.
    var mob_spawn_location = get_node("MobPath/MobSpawnLocation")
    mob_spawn_location.offset = randi()

    # Set the mob's direction perpendicular to the path direction.
    var direction = mob_spawn_location.rotation + PI / 2

    # Set the mob's position to a random location.
    mob.position = mob_spawn_location.position

    # Add some randomness to the direction.
    direction += rand_range(-PI / 4, PI / 4)
    mob.rotation = direction

    # Choose the velocity for the mob.
    var velocity = Vector2(rand_range(150.0, 250.0), 0.0)
    mob.linear_velocity = velocity.rotated(direction)

    # Spawn the mob by adding it to the Main scene.
    add_child(mob)
```

Важный

Почему `PI`? В функциях, требующих углы, Godot использует *радианы*, а не градусы. Число Пи представляет собой пол-оборота в радианах, примерно `3.1415` (также есть переменная `TAU`, которая равна `2 * PI`) Если вам удобнее работать с градусами, вам нужно использовать функции `deg2rad()` и `rad2deg()` для преобразования между ними.

Тестирование сцены

Давайте протестируем сцену, чтобы убедиться, что все работает. Добавьте в `_ready()` вызов `new_game`:

GDScript C# C++

```
func _ready():
    randomize()
    new_game()
```

Также давайте назначим сцену `Main` в качестве нашей "Главной сцены", которая запускается автоматически при запуске игры. Нажмите кнопку "Play" и выберите `Main.tscn` при появлении запроса.

Совет

Если вы уже установили другую сцену в качестве «Основной сцены», вы можете щелкнуть правой кнопкой мыши `Main.tscn` в доке файловой системы и выбрать «Установить как основную сцену».

У вас должна быть возможность перемещать игрока, видеть, как появляются мобы, и видеть, как игрок исчезает, когда его бьет моб.

Когда вы убедитесь, что всё работает, удалите вызов `new_game()` из `_ready()`.

Чего не хватает нашей игре? Какого-нибудь пользовательского интерфейса. В следующем уроке мы добавим заглавный экран и отобразим очки игрока.