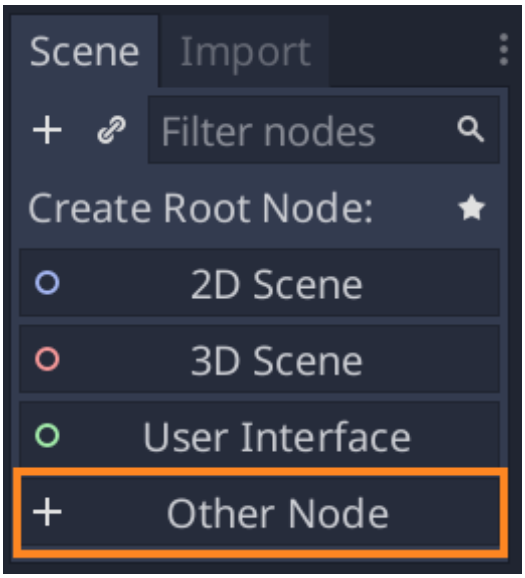
# Создание сцены игрока

С установленными настройками проекта, мы можем начать работу над персонажем, управляемым игроком.

Первая сцена будет определять объект `Player`. Одним из преимуществ создания отдельной сцены Player является то, что мы можем протестировать ее отдельно, даже до того, как создадим другие части игры.

## Структура узла

Для начала нам нужно выбрать корневой узел для объекта игрока. Как правило, корневой узел сцены должен отображать желаемую функциональность объекта - *чем* объект является. Нажмите кнопку "Другой узел" (Other Node) и добавьте узел Area2D в сцену.

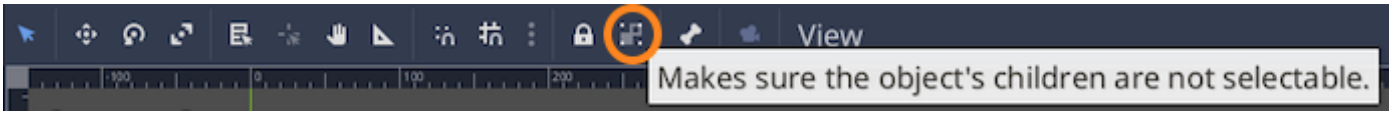



Godot отобразит значок предупреждения рядом с этим узлом в древе сцены. Пока что не обращайте внимание на это. Мы рассмотрим это позже.

С помощью `Area2D` мы можем обнаруживать объекты, которые перекрывают или сталкиваются с игроком. Измените имя узла на `Player`, дважды щёлкнув по нему. Теперь, когда мы установили корневой узел сцены, мы можем добавлять дополнительные узлы, чтобы привнести больше функционала.

Прежде чем мы добавим потомков на узел `Player`, мы хотим убедиться, что нажав на них, мы не переместим их и не изменим их размер. Выберите узел и нажмите на значок справа

от блокировки; его всплывающая подсказка гласит: "Делает потомков объекта невыбираемыми."




Сохраните сцену. Нажмите "Сцена" -> "Сохранить сцену" в верхней панели или нажмите сочетание клавиш `Ctrl + S` на Windows/Linux или `Cmd + S` на Mac.

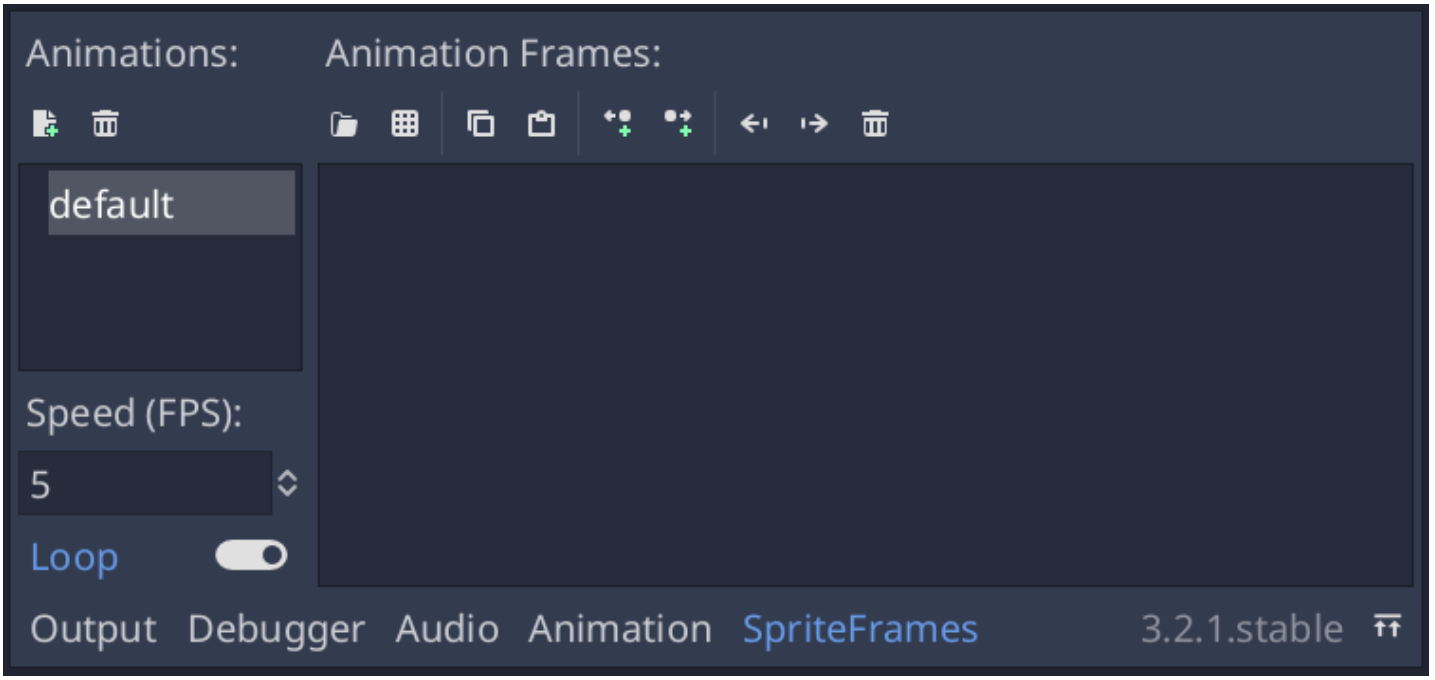
**Примечание**

Для этого проекта мы будем следовать правилам именования Godot.

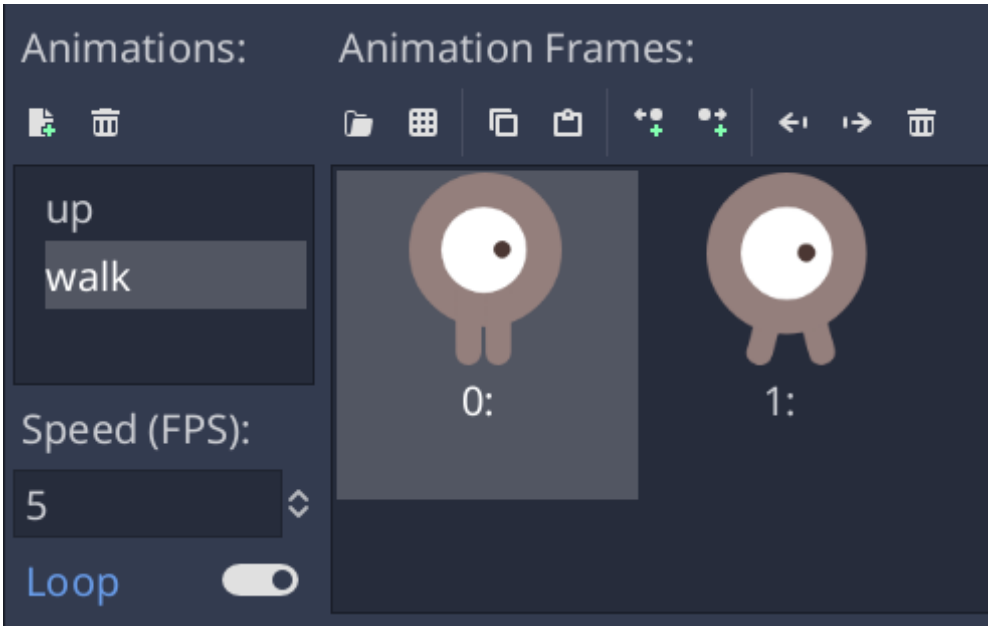
- **GDScript**: Классы (узлы) используют PascalCase, переменные и функции - snake_case, константы - ALL_CAPS (см. Руководство по стилю GDScript).
- **C#**: Классы, экспортные переменные и методы используют PascalCase, приватные поля используют _camelCase, локальные переменные и параметры используют camelCase (См. Руководство по стилю C#). Будьте внимательны, чтобы точно набирать имена методов при подключении сигналов.

## Анимация Спрайтов

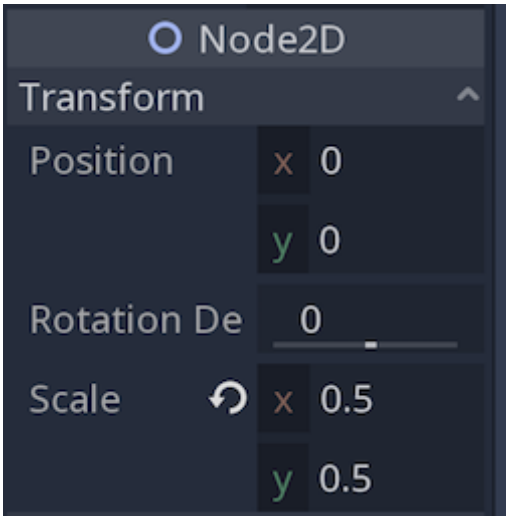
Кликните на узел `Player` и добавьте дочерний узел AnimatedSprite. `AnimatedSprite` будет обрабатывать внешний вид и анимации для нашего игрока. Обратите внимание на символ предупреждения рядом с узлом. `AnimatedSprite` требует ресурс SpriteFrames, который представляет собой список отображаемых анимаций. Чтобы его создать, найдите свойство `Frames` в окне инспектора и кликните на «[пусто]» -> «Новый SpriteFrames». Снова кликните в том же месте на `SpriteFrames`, чтобы открыть панель:



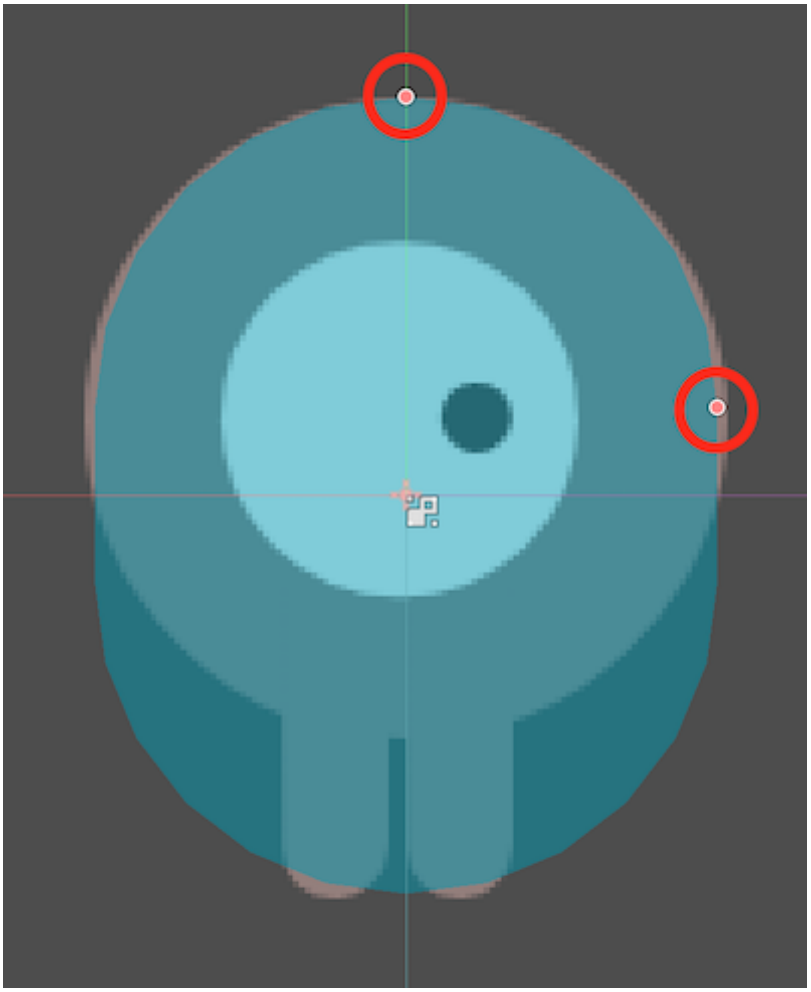

Слева находится список анимаций. Нажмите на "default" и переименуйте на "walk". Затем щелкните по кнопке "Новая анимация" для создания второй анимации с именем "up". Найдите изображения игрока на вкладке "Файловая система" - они находятся в папке "art", которую вы разархивировали ранее. Перетащите два изображения с названиями `playerGrey_up[1/2]` и `playerGrey_walk[1/2]` на сторону панели "Кадры анимации" для каждой анимации соответственно:



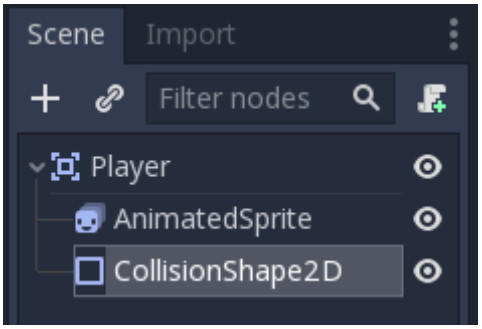

Изображения игрока немного великоваты для игрового окна, потому нам надо их слегка уменьшить. Нажмите на узел `AnimatedSprite` и установите параметр `Scale` в `(0.5, 0.5)`. Вы можете найти его в Инспекторе под заголовком `Node2D`.




Наконец, добавьте CollisionShape2D в качестве дочернего элемента `Player`. Это определит "хитбокс" игрока, или границы его области столкновения. Для этого персонажа лучше всего подходит узел `CapsuleShape2D`, поэтому рядом с "Shape" в Инспекторе нажмите "[empty]" -> "New CapsuleShape2D". Используя две ручки размера, измените размер фигуры, чтобы она покрывала спрайт:



Когда вы закончите, сцена с вашим `Player` будет выглядеть вот так:




После этих изменений обязательно сохраните сцену еще раз.

В следующей части мы добавим скрипт к узлу игрока, чтобы перемещать и анимировать его. Позже, мы настроим обнаружение столкновений, чтобы знать, в какой момент игрока что-то ударило.