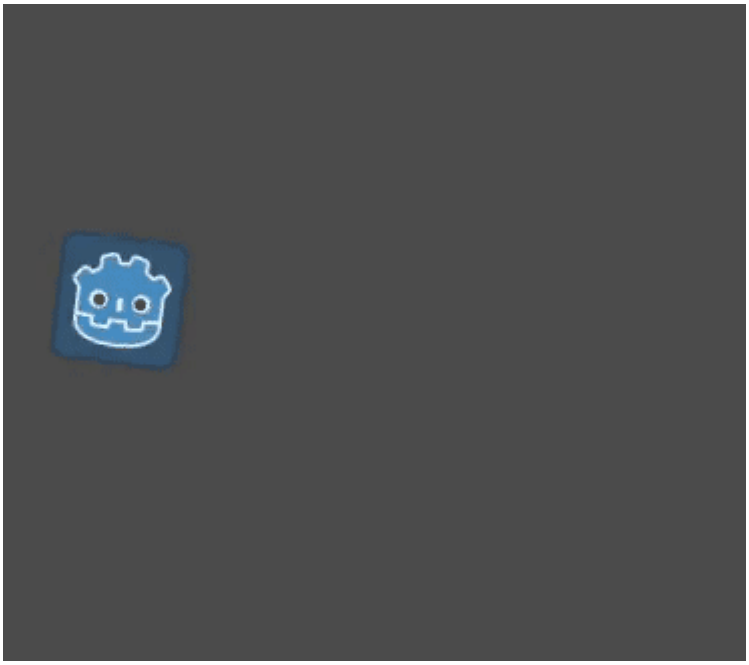
Создание вашего первого скрипта

В этом уроке вы напишете свой первый скрипт, чтобы значок Godot вращался по кругу, используя GDScript. Как мы упоминали во введении , мы предполагаем, что у вас есть основы программирования. Эквивалентный код C# для удобства включен в другую вкладку.



См.также

Подробнее о GDScript, его ключевых словах и синтаксисе, читайте в Основы GDScript.

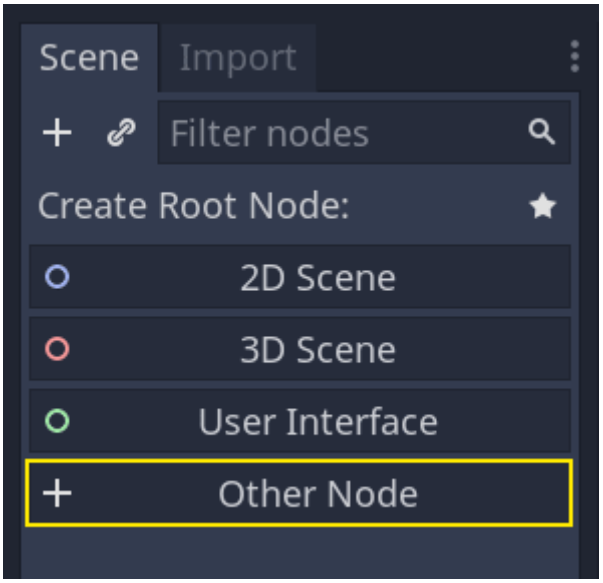
См.также

Чтобы узнать больше о C#, перейдите на страницу основ C# .

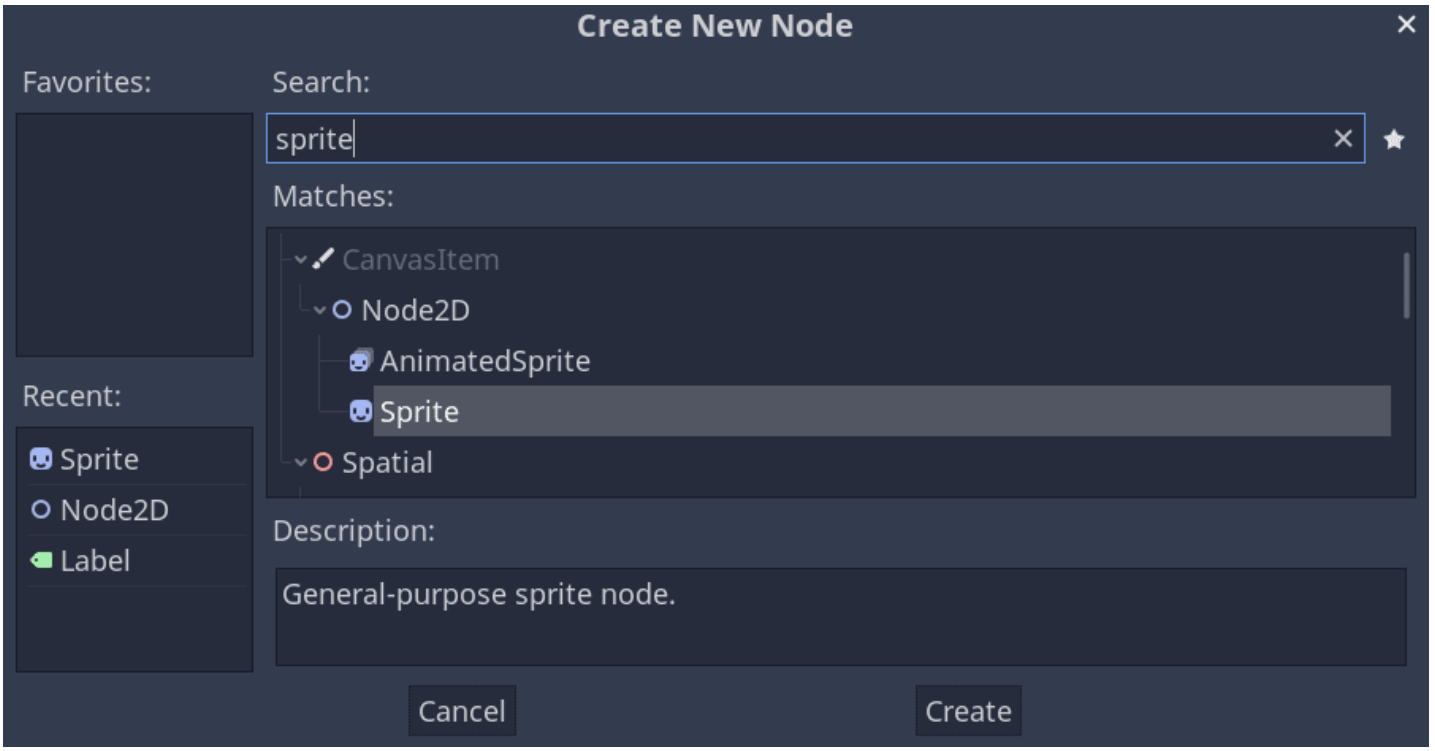
Настройка проекта

Пожалуйста, создайте новый проект, чтобы начать с чистого листа. Ваш проект должен содержать одно изображение: иконку Godot, которую мы в сообществе часто используем для прототипирования.

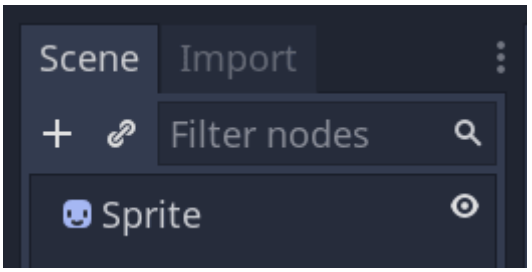
Нам нужно создать узел Sprite для отображения в игре. В панели Scene, кликните на кнопку Other Node (Другой Узел).



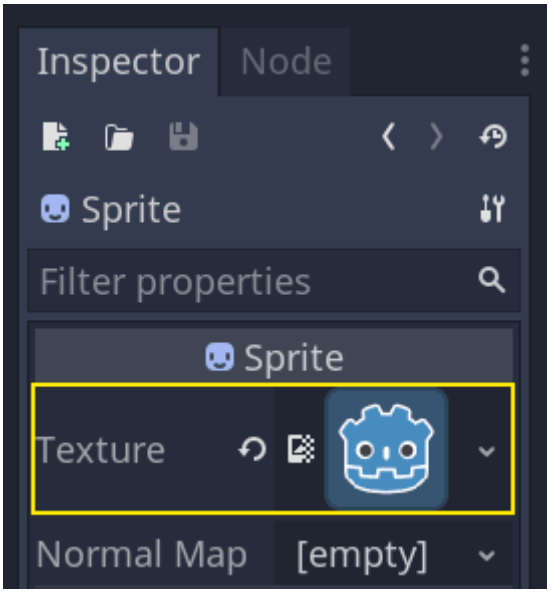
Наберите "Sprite" в поисковой строке, чтобы отобразились нужные узлы, затем дважды кликните на Sprite, чтобы создать узел.



Вкладка Scene теперь должна содержать только узел Sprite.

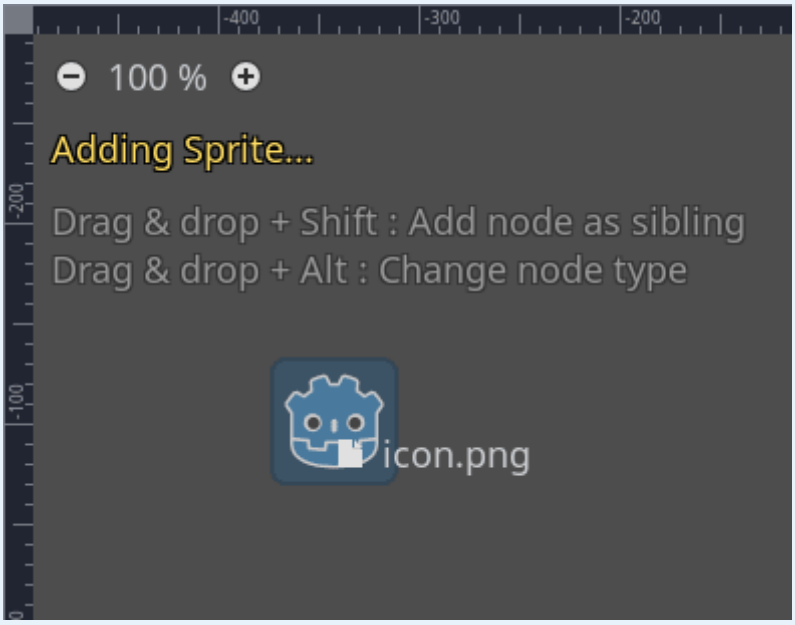


Узлу Sprite нужна текстура для отображения. В Инспекторе справа вы можете увидеть, что в свойстве текстуры указано "[пусто]". Чтобы отобразить иноку Godot, кликните и перетащите файл ``icon.png``из файловой системы панели на слот текстуры.

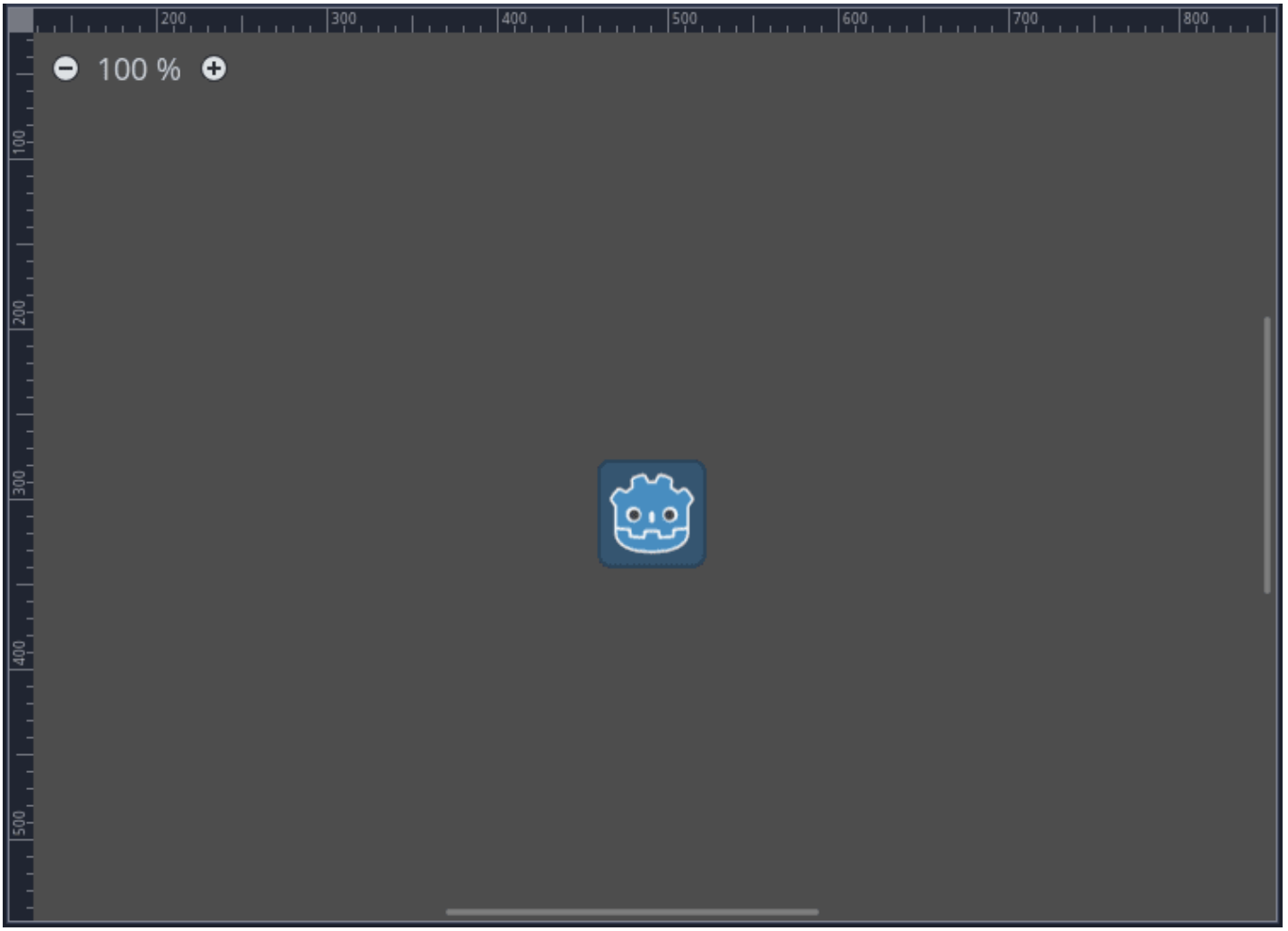


Примечание

Также вы можете создавать узлы Sprite автоматически, перетаскивая изображения в Окно просмотра.

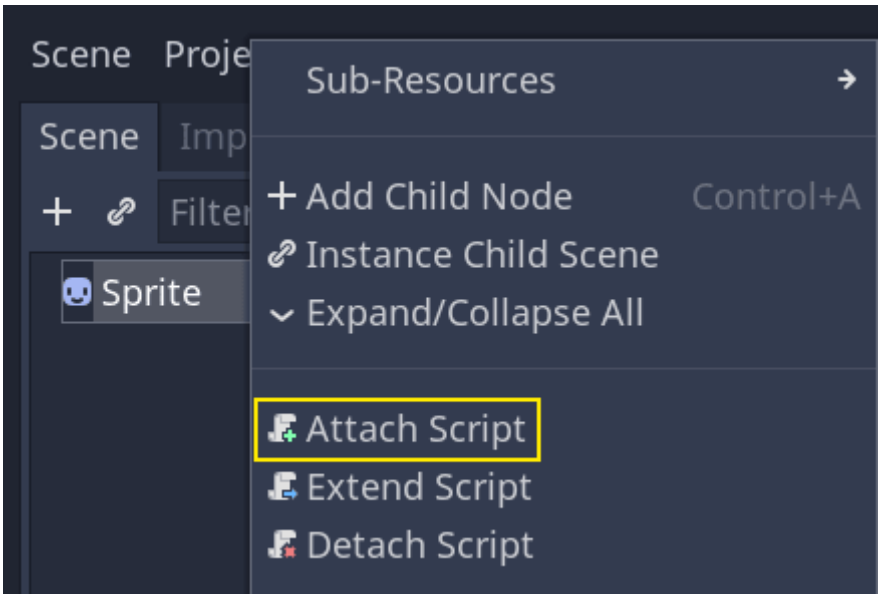


Затем нажмите и перетащите значок в окне просмотра, чтобы отцентрировать его в игровом представлении.



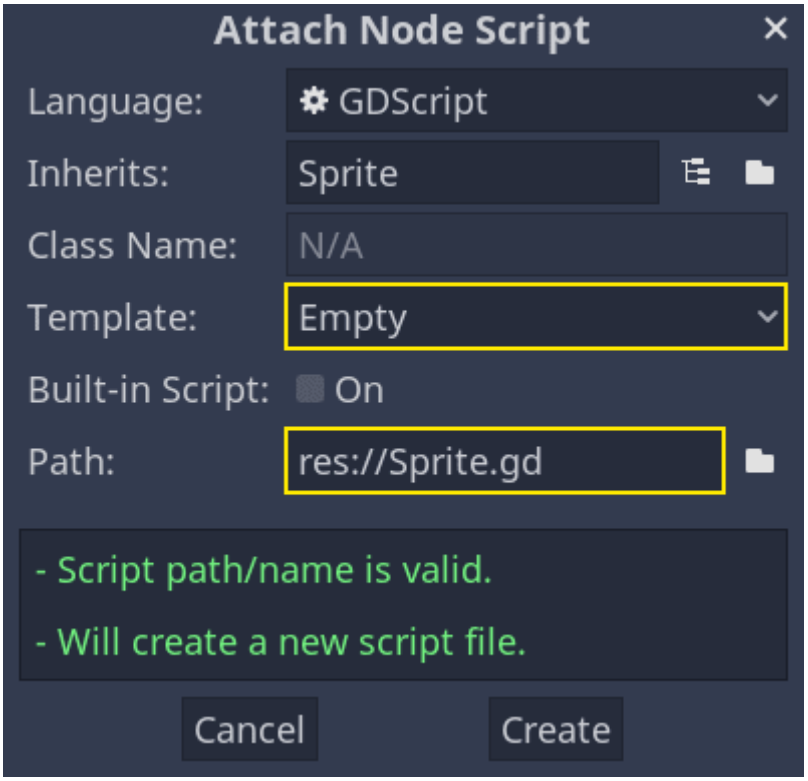
Создание нового скрипта

Чтобы создать и прикрепить новый скрипт к нашему узлу, нажмите правой кнопкой мыши на Sprite в панели сцены и выберите "Attach Script" (Прикрепить Скрипт).



Откроется окно Attach Node Script. В этом окне вы можете указать язык программирования скрипта, путь в корневой папке и другие параметры.

Измените Template (Шаблон) с Default на Empty, чтобы начать с чистого файла. Оставьте остальные параметры по умолчанию и нажмите кнопку Create (Создать), чтобы создать сценарий.



В рабочей области Script должен появиться новый файл `Sprite.gd` и следующая строка кода:

GDScript C#

```
extends Sprite
```

Каждый файл GDScript неявно является классом. Ключевое слово `extends` определяет класс, который наследует или расширяет данный скрипт. В этом случае `Sprite` означает, что наш скрипт получит доступ к свойствам и функциям узла `Sprite`, включая классы, которые он расширяет, такие как `Node2D`, `CanvasItem` и``Node``.

Примечание

В GDScript, если вы опустите строку с ключевым словом `extends`, ваш класс будет неявно расширять Reference, который Godot использует для управления памятью вашего приложения.

К унаследованным свойствам относятся те, которые вы можете видеть в инспекторе, например, `texture` узла.

Примечание

По умолчанию Инспектор отображает свойства узла в «Заглавном регистре» с заглавными буквами, разделенными пробелом. В коде GDScript эти свойства находятся в «snake_case», то есть в нижнем регистре со словами, разделенными символом подчеркивания.

Вы можете навести курсор на имя любого свойства в инспекторе, и увидеть его описание и идентификатор в коде.

Привет мир!

Наш скрипт ничего не делает. Давайте заставим его вывести "Hello, world!" в панель вывода в нижней части экрана.

Добавьте следующий код в ваш скрипт:

GDScript C#

```
func _init():
    print("Hello, world!")
```

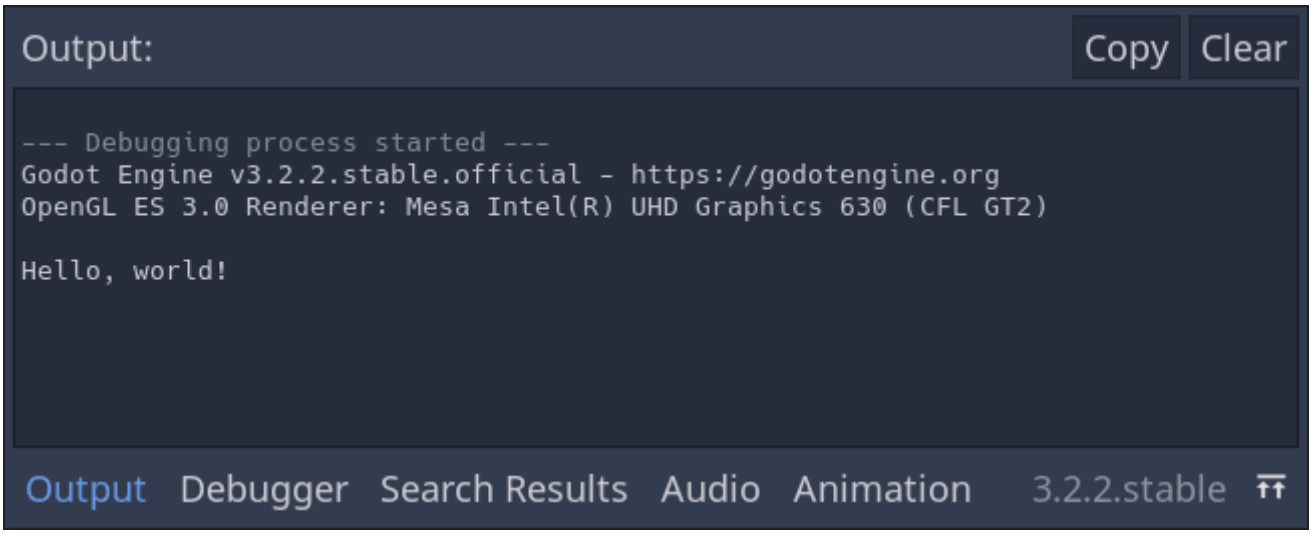
Давайте разберём это. Слово `func` объявляет новую функцию `_init`. Это конструктор нашего класса. Движок вызывает `_init` при создании объекта в памяти, если вы определили эту функцию.

Примечание

GDScript — это язык, основанный на отступах. Вкладка в начале строки с надписью `print()` необходима для работы кода. Если вы опустите его или сделаете неправильный

отступ строки, редактор выделит ее красным цветом и отобразит следующее сообщение об ошибке: «Ожидается блок с отступом».

Сохраните сцену, если вы еще этого не сделали, затем нажмите `F6` (в macOS), чтобы запустить ее. Посмотрите на нижнюю панель **вывода** , которая расширяется. Он должен отображать «Привет, мир!». `Cmd + R`



Удалите функцию `_init()` , оставив только строку `extends Sprite` .

Поворот вокруг

Теперь сделаем наш узел двигающимся и вращающимся. Для этого добавим две переменных-члена в наш скрипт: скорость перемещения в пикселях в секунду и угловую скорость в радианах в секунду.

GDScript C#

```
var speed = 400
var angular_speed = PI
```

Переменные-члены располагаются в верхней части скрипта после любых строк «расширения», но перед функциями. Каждый экземпляр узла с прикрепленным к нему скриптом будет иметь собственную копию свойств `speed` и `angular_speed` .

Примечание

Углы в Godot по умолчанию работают в радианах, но у вас есть встроенные функции и свойства, если вы предпочитаете вместо этого вычислять углы в градусах.

Для перемещения иконки мы должны обновлять её позицию и вращение каждый кадр игрового цикла. Для этого используем виртуальную функцию `_process` класса `Node`. При объявлении этой функции в любом классе, наследуемом от `Node` (например, `Sprite`), Godot будет вызывать её каждый кадр и передавать в аргументе `delta` время, прошедшее от предыдущего кадра.

Примечание

Игры работают в цикле, отображая множество изображений в секунду, каждое из которых называется кадром. Скорость, с которой игра создаёт эти изображения, измеряется в кадрах в секунду (Frame Per Second). Большинство игр нацелены на 60 FPS, хотя вы можете найти такие цифры, как 30 кадров в секунду на более медленных мобильных устройствах или от 90 до 240 для игр виртуальной реальности.

Движок и разработчики игры делают все возможное, чтобы обновлять игровой мир и рендерить изображения с постоянным интервалом времени, но всегда существуют небольшие отклонения во времени рендеринга кадров. Поэтому движок предоставляет нам это значение дельта-времени, делая наше движение независимым от частоты кадров.

В нижней части скрипта определите функцию:

GDScript C#

```
func _process(delta):
    rotation += angular_speed * delta
```

Ключевое слово func определит (создаст) новую функцию. После него в нужно написать имя функции и в скобках аргументы, которые она принимает. Двоеточие завершает определение, а следующие за ним блоки с отступом представляют собой содержимое или инструкции функции.

Примечание

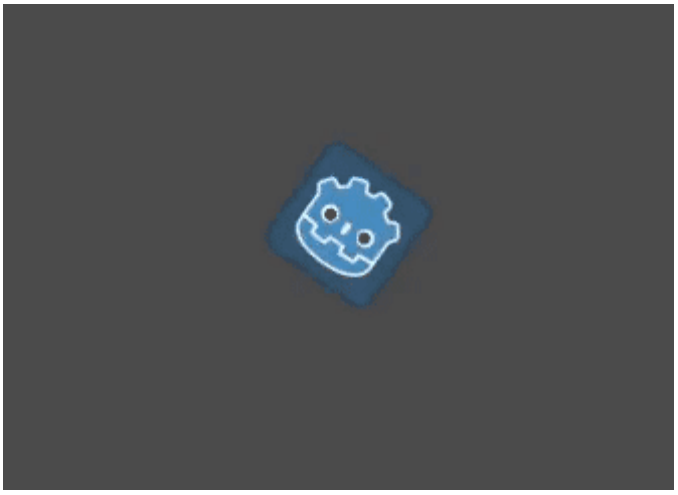
Обратите внимание, что `_process()`, как и `_init()`, начинается с символа подчёркивания. По соглашению, виртуальные функции, то есть встроенные функции Godot, которые вы можете переопределить - начинаются с символа подчёркивания.

Строка внутри функции, `rotation += angular_speed * delta`, увеличивает угол вращения нашего спрайта каждый кадр. Здесь `rotation` — это свойство, унаследованное от класса `Node2D`, который расширяет класс `Sprite`. Именно он управляет вращением узла, используя радианы для измерения угла поворота.

Совет

В редакторе кода вы можете нажать Ctrl+ЛКМ на любом встроенном свойстве или функции, таких как `position`, `rotation`, или `_process` чтобы открыть соответствующую документацию в новой вкладке.

Запустите сцену, чтобы увидеть, что иконка Godot крутится на месте.



Движение вперёд

Теперь давайте заставим узел двигаться. Добавьте следующие две строки в функцию `_process()`, убедившись, что новые строки имеют такой же отступ, как и предыдущие.

GDScript C#

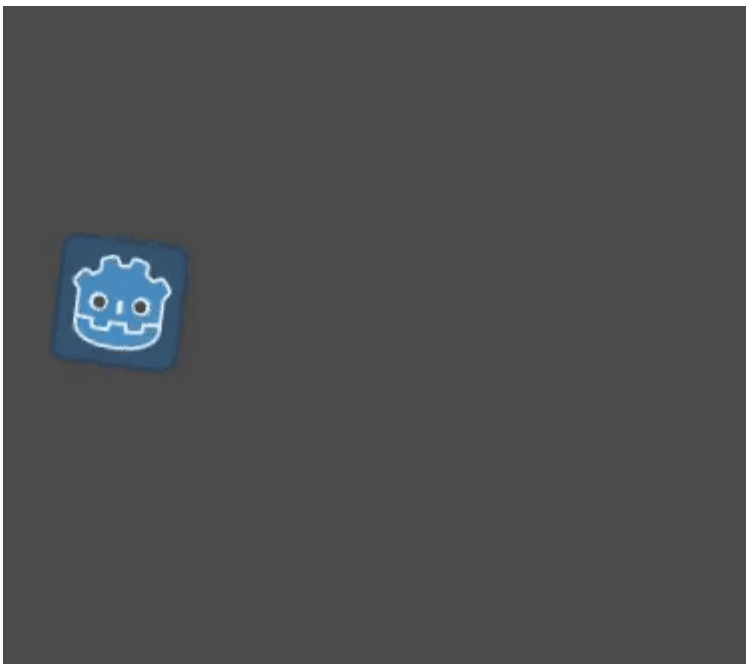
```
var velocity = Vector2.UP.rotated(rotation) * speed

position += velocity * delta
```

Как мы уже видели, ключевое слово `var` определяет новую переменную. Если вы поместите его в верхней части скрипта, оно определит свойство класса. Внутри функции оно определяет локальную переменную: данная переменная существует только в области действия функции.

Мы определяем локальную переменную `velocity`, 2D-вектор, представляющий собой направление и скорость. Чтобы заставить узел двигаться вперёд, мы начинаем с константы Vector2.UP класса Vector2, эта константа - вектор указывающий вверх и вращающийся с помощью вызова метода `Vector2.rotated()`. Данное выражение `Vector2.UP.rotated(rotation)` - вектор, указывающий вперёд относительно нашей иконки. Умноженное на наше свойство `speed`, это выражение даёт нам скорость, которую мы можем использовать для перемещения узла вперёд.

Мы добавляем `velocity * delta` к узлу `position`, чтобы переместить его. Сама позиция имеет тип Vector2, встроенный тип в Godot, представляющий двумерный вектор.

Запустите сцену, чтобы увидеть, что голова Godot движется по кругу.



Примечание

При таком перемещении узла не учитывается столкновение со стенами или полом. В Ваша первая 2D игра вы узнаете другой подход к перемещению объектов при обнаружении столкновений.

Наш узел сейчас движется самостоятельно. В следующей части Отслеживание ввода игрока мы будем использовать пользовательский ввод, чтобы контролировать его.

Готовый скрипт

Это полный файл `Sprite.gd` для справки.

GDScript C#

```
extends Sprite

var speed = 400
var angular_speed = PI


func _process(delta):
    rotation += angular_speed * delta

    var velocity = Vector2.UP.rotated(rotation) * speed

    position += velocity * delta
```