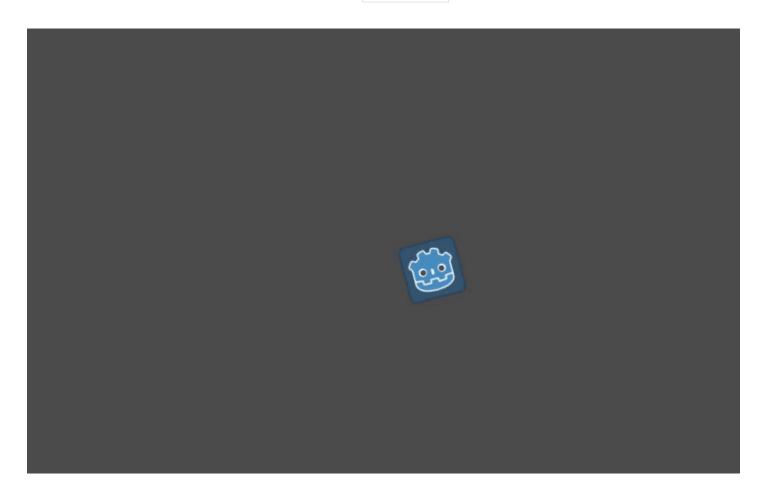
# Отслеживание ввода игрока

Основываясь на предыдущем уроке Создание вашего первого скрипта, давайте рассмотрим еще одну важную особенность любой игры: предоставление контроля игроку. Чтобы добавить это, нам нужно изменить наш код Sprite.gd.



У вас есть два важных инструмента для обработки пользовательского ввода в Godot:

- 1. Встроенные обратные вызовы ввода, в основном \_\_unhandled\_input() . Например, \_\_process() , это встроенная виртуальная функция, которую Godot вызывает каждый раз, когда игрок нажимает клавишу. Это инструмент, который вы захотите использовать, чтобы реагировать на события, которые не происходят каждый кадр, например, нажатие \_\_Space \_\_для прыжка. Чтобы узнать больше об обратных вызовах ввода, посмотрите \_\_Использование InputEvent.
- 2. Singleton``Input``. Singleton это глобально доступный объект. Godot предоставляет доступ к нескольким из них в скриптах. Это подходящий инструмент для проверки ввода в каждом кадре.

Здесь мы будем использовать синглтон <u>Input</u>, поскольку нам нужно знать в каждом кадре, хочет ли игрок поворачиваться или двигаться.

Для поворота мы должны использовать новую переменную: direction. В нашей функции \_process() замените строку rotation += angular\_speed \* delta на приведенный ниже код.

#### **GDScript**

C#

```
var direction = 0
if Input.is_action_pressed("ui_left"):
    direction = -1
if Input.is_action_pressed("ui_right"):
    direction = 1

rotation += angular_speed * direction * delta
```

Наша локальная переменная direction - это множитель, представляющий собой направление, в котором игрок хочет повернуть. Значение 0 означает, что игрок не нажимает ни стрелку влево, ни стрелку вправо. Значение 1 говорит, что игрок хочет повернуть вправо, а -1 - что игрок хочет повернуть влево.

Для обработки этих значений мы вводим условие и используем Input. Условие в GDScript начинается с ключевого слова if и заканчивается двоеточием. Условие - это выражение между ключевым словом и концом строки.

Чтобы проверить, была ли нажата клавиша в этом кадре, мы вызываем

Input.is\_action\_pressed(). Метод принимает текстовую строку, представляющую входное действие, и возвращает true, если клавиша нажата, и false в противном случае.

Два действия, которые мы использовали выше, «ui\_left» и «ui\_right», предопределены в каждом проекте Godot. Они соответственно срабатывают, когда игрок нажимает стрелки влево и вправо на клавиатуре или же влево и вправо на крестовине геймпада.

#### Примечание

Вы можете просматривать и редактировать действия ввода в своем проекте, перейдя в «Проект» -> «Настройки проекта» и щелкнув вкладку «Список действий».

Наконец, мы используем direction как множитель, когда обновляем rotation узла: rotation += angular\_speed \* direction \* delta.

### Перемещение при нажатии "вверх"

Чтобы двигаться только при нажатии клавиши, нам нужно изменить код, который вычисляет скорость. Замените строку, начинающуюся с var velocity, на приведенный ниже код.

```
var velocity = Vector2.ZER0
if Input.is_action_pressed("ui_up"):
    velocity = Vector2.UP.rotated(rotation) * speed
```

Мы инициализируем ``velocity" со значением ``Vector2.ZERO", еще одной константой встроенного типа ``Vector", представляющей двумерный вектор нулевой длины.

Когда игрок выполняет действие "ui\_up", мы обновляем значение скорости, заставляя спрайт двигаться вперёд.

## Готовый скрипт

Это полный файл Sprite.gd для справки.

```
cytends Sprite

var speed = 400
var angular_speed = PI

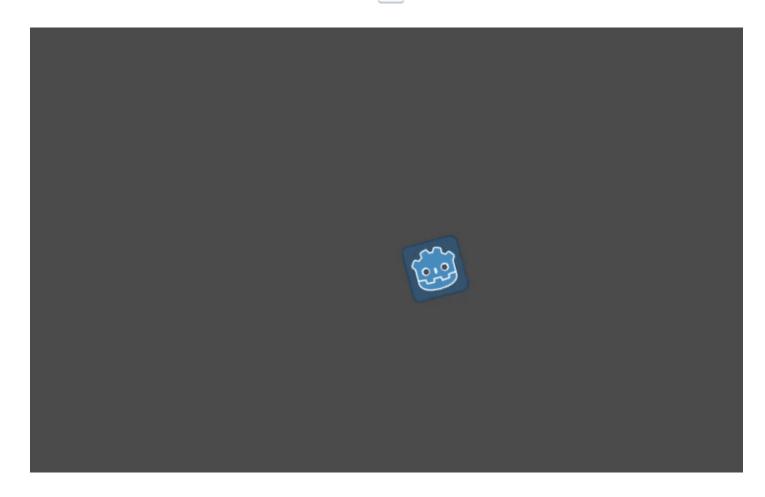
func _process(delta):
    var direction = 0
    if Input.is_action_pressed("ui_left"):
        direction = -1
    if Input.is_action_pressed("ui_right"):
        direction = 1

rotation += angular_speed * direction * delta
```

```
var velocity = Vector2.ZER0
if Input.is_action_pressed("ui_up"):
    velocity = Vector2.UP.rotated(rotation) * speed

position += velocity * delta
```

Если вы запустите сцену, вы должны иметь возможность вращаться с помощью стрелок влево и вправо, а также двигаться нажатием Up.



#### Подведение итогов

В общем, каждый скрипт в Godot представляет собой класс и расширяет один из встроенных классов движка. Типы узлов, от которых наследуются ваши классы, дают вам доступ к таким свойствам, как ``вращение" и ``позиция" в случае нашего спрайта. Вы также наследуете множество функций, которые мы не использовали в этом примере.

В GDScript переменные, которые вы помещаете в верхней части файла, являются свойствами вашего класса, также называемыми переменными-членами. Кроме переменных, вы можете определять функции, которые, по сути, являются методами ваших классов.

Godot предоставляет несколько виртуальных функций, которые вы можете определить для связи вашего класса с движком. К ним относятся \_\_process(), применяющая изменения к узлу каждый кадр, и \_\_unhandled\_input(), получающая события ввода, такие как нажатие клавиш и кнопок от пользователей. Есть еще множество других функций.

Одноэлементное множество Input``позволяет Вам реагировать на участие игрока в любом месте Вашего кода. В первую очередь Вы сможете применить его в цикле ``\_process().

В следующем уроке :ref:`doc\_signals`мы будем опираться на отношения между скриптами и узлами так, чтобы узлы приводили код в скриптах в действие.