Простая 2D игра

понг

В этом уроке будет создана базовая игра в понг. В демонстрациях, включенных в движок, есть множество более сложных примеров, но это дол

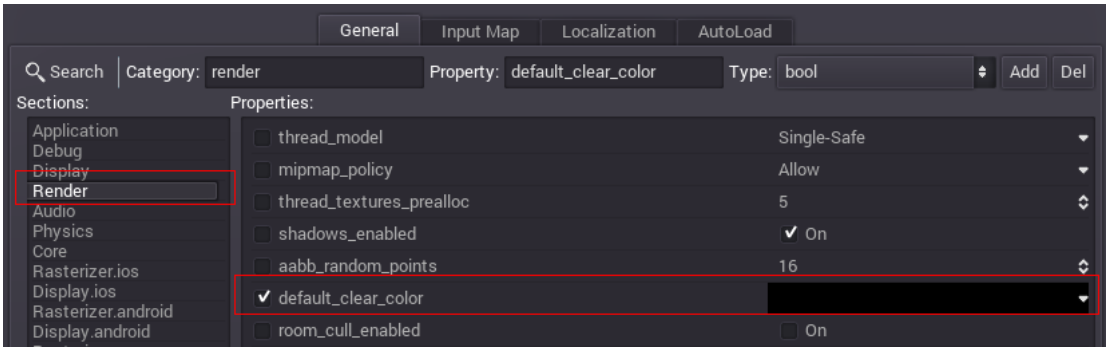
Для начала запустите Godot Engine и начните новый проект.

Ресурсы

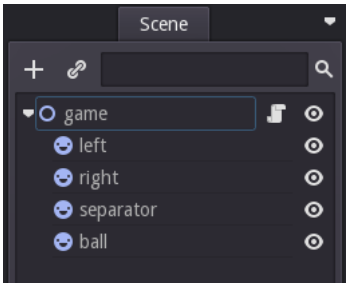
Некоторые активы включены в этот учебник: pong_assets.zip. Разархивируйте его содержимое в папку вашего проекта.

Настройка сцены

По старинке игра будет в разрешении 640x400 пикселей. Это можно настроить в настройках проекта (см. Настройка проекта) в меню настроек



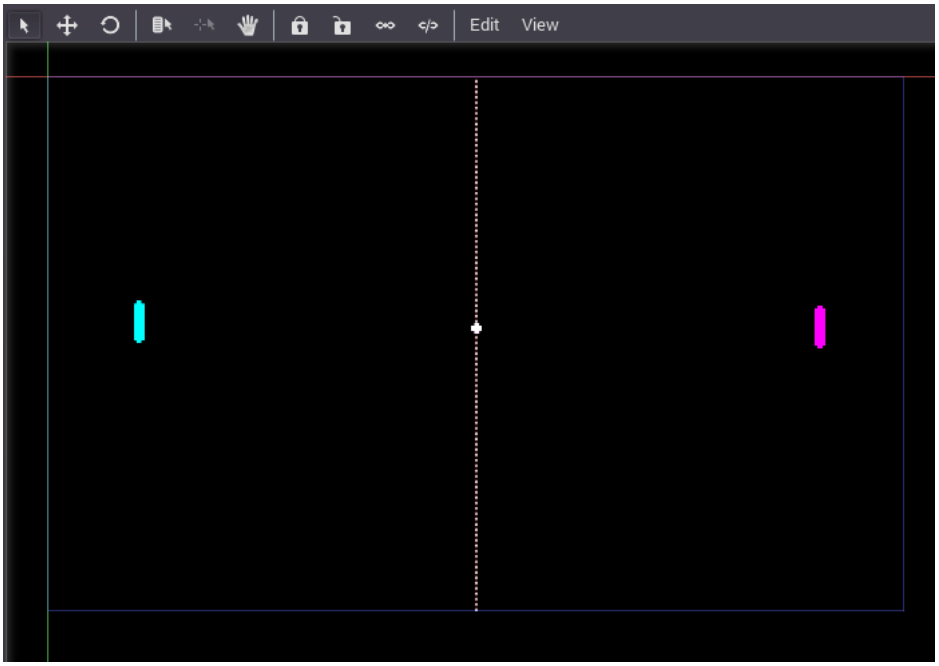
Создайте узел Node2D для корня проекта. Node2D — это базовый тип для движка 2D. После этого добавьте несколько спрайтов (узел Sprite) д имя для каждого узла и установить текстуру для каждого спрайта в Инспекторе.



Установить позиции узлов:

- «левый» узел: (67, 183)
- «правый» узел: (577, 187)
- узел «разделитель»: (320, 200)
- узел «шарик»: (320, 188)

Окончательный макет сцены должен выглядеть примерно так (обратите внимание: шар находится посередине!):



Сохраните сцену как «pong.tscn» и установите ее как основную сцену в свойствах проекта.

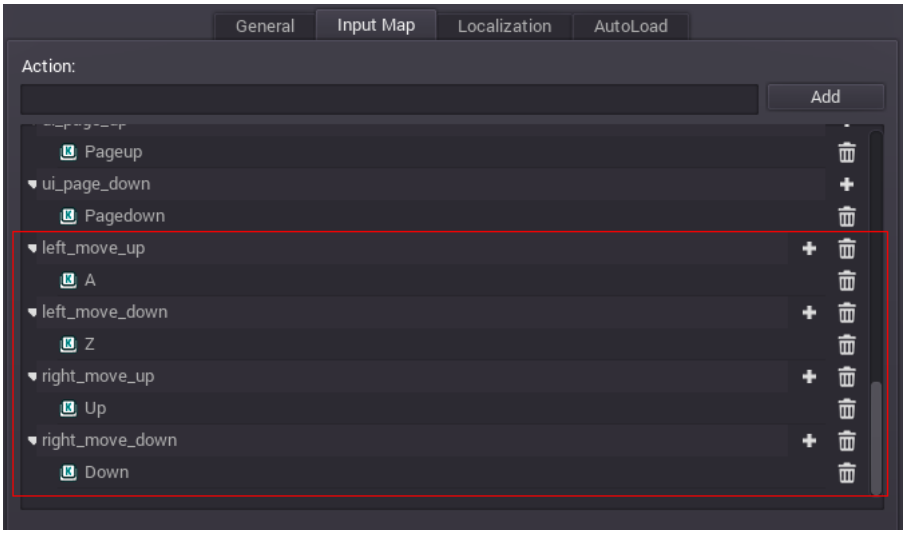
Настройка входных действий

В видеоигры можно играть, используя различные способы ввода: клавиатура, джойстик, мышь, сенсорный экран (мультитач)... Godot может исп
вместо аппаратных действий, которыми вы бы управляли отдельно. Таким образом, можно использовать любой метод ввода: каждый из них тр

Это Понг. Единственный ввод, который имеет значение, - это движение пэдов вверх и вниз.

Снова откройте диалоговое окно свойств проекта (настройки сцены/проекта), но на этот раз перейдите на вкладку «Входная карта».

В этой вкладке добавьте 4 действия: `left_move_up` , `left_move_down` , `right_move_up` , `right_move_down` . Назначьте ключи, которые вы хотите. A/Z (д
большинстве случаев.



Скрипт

Создайте сценарий для корневого узла сцены и откройте его (как описано в разделе Добавление сценария). Этот скрипт наследует Node2D:

```
расширяет  Node2D

функция  _ready ():
    пройти
```

Прежде всего, нам нужно определить некоторые элементы для нашего скрипта, чтобы он мог хранить полезные значения. Такими значениями

```
расширяет  Node2D

# Переменные-члены
var  screen_size
var  pad_size
var  direction  =  Vector2 ( 1.0 ,  0.0 )

функция  _ready ():
    пройти
```

Как вы знаете, `_ready()` функция — это первая вызываемая функция (после `_enter_tree()` которой мы здесь не нуждаемся). В этой функции нуж
Второй — инициализировать две наши переменные-члены.

```
расширяет  Node2D

# Переменные-члены
var  screen_size
var  pad_size
var  direction  =  Vector2 ( 1.0 ,  0.0 )

func  _ready ():
    screen_size  =  get_viewport_rect () . размер
    pad_size  =  get_node ( "левый" ) . получить_текстуру () . get_size ()
    set_process ( истина )
```

Мы инициализируем `pad_size` переменную, получая один из узлов контактных площадок (левый здесь) и получаем размер его текстуры. Иници
Rect, соответствующий окну игры, и мы сохраняем его размер.

Теперь нам нужно добавить некоторые другие члены в наш скрипт, чтобы заставить наш мяч двигаться.

```
расширяет  Node2D

# Переменные-члены
var  screen_size
var  pad_size
var  direction  =  Vector2 ( 1.0 ,  0.0 )

# Константа скорости мяча (в пикселях/секунду)
const  INITIAL_BALL_SPEED  =  80
# Скорость мяча (также в пикселях/секунду)
var  ball_speed  =  INITIAL_BALL_SPEED
# Константа скорости подушечек
const  PAD_SPEED  =  150

func  _ready ():
    screen_size  =  get_viewport_rect () . размер
    pad_size  =  get_node ( "левый" ) . получить_текстуру () . get_size ()
    set_process ( истина )
```

Наконец, `_process()` функция. Весь приведенный ниже код содержится в этой функции.

Мы должны ввести некоторые полезные значения для вычислений. Первый — это положение шара (от узла), второй — прямоугольник (`Rect2`) д
столкновения мяча с подушечками. Спрайты центрируют свои текстуры по умолчанию, поэтому необходимо добавить небольшую настройку .

```
func_process  ( delta ) : var ball_pos = get_node ( "ball" ) . get_pos () var left_rect = Rect2 ( get_node ( "левый" ) . get_pos () - pad_size * 0.


           pad_size  )
```

Теперь давайте добавим немного движения мячу в `_process()` функцию. Поскольку положение мяча хранится в `ball_pos` переменной, интегрир

```
# Интеграция новой позиции мяча
ball_pos  +=  direction  *  ball_speed  *  delta
```

Эта строка кода вызывается при каждой итерации функции `_process()`. Это означает, что положение мяча будет обновляться в каждом новом

Теперь, когда у мяча новое положение, нам нужно проверить, не сталкивается ли он с чем-либо, то есть с границами окна и контактными площа

```
# Перевернуть при касании крыши или пола if  ( ( ball_pos.y < 0 и direction.y < 0 ) или ( ball_pos.y > screen_size.y and direction.y > 0 ) ) : dire
```

Во-вторых, подушечки: если коснуться одной из подушечек, нам нужно инвертировать направление мяча по оси X, чтобы он вернулся назад, и с
увеличиваем его скорость.

```
# Перевернуть, изменить направление и увеличить скорость при касании пэдов
if (  ( left_rect . has_point ( ball_pos )  and  direction .x <  0  ) or ( right_rect . has_point ( ball_pos ) and direction . x > 0 )): direction


    направление  =  направление . нормализованная ()
    ball_speed  *=  1,1
```

Наконец, если мяч вышел за пределы экрана, игра окончена. То есть мы проверяем, меньше ли позиция X мяча, чем 0, или больше, чем ширина

```
# Проверяем завершение игры,
если  ( ball_pos . x  <  0  или  ball_pos . x  >  screen_size . x ):
    ball_pos  =  screen_size * 0.5
    ball_speed  =  INITIAL_BALL_SPEED
    direction  =  Vector2 ( - 1 ,  0 )
```

Как только все сделано, узел обновляется новой позицией мяча, которая была вычислена ранее:

```
get_node ( "шар" ) . set_pos ( ball_pos )
```

Затем мы позволяем контактным площадкам двигаться. Мы обновляем их позицию только в соответствии с данными игрока. Это делается с пс

```
# Переместить левую панель
var  left_pos  =  get_node ( "left" ) . получить_поз ()

if  ( left_pos.y > 0 и Input.is_action_pressed ( " left_move_up "  ) ) : left_pos  . y + = - PAD_SPEED * delta if ( left_pos.y < screen_size.y и In



get_node ( "левый" ) . set_pos ( левая_позиция )

# Переместить правую панель
var  right_pos  =  get_node ( "right" ) . получить_поз ()

if  ( right_pos.y > 0 и Input.is_action_pressed ( " right_move_up " ) ) : right_pos . y + = - PAD_SPEED * delta if ( right_pos .y < screen_size .y



get_node ( "правильный" ) . set_pos ( правая_позиция )
```

Мы используем четыре действия, ранее определенные в разделе «Настройка действий ввода» выше. Когда игрок активирует соответствующую вычисляем новую позицию контактной площадки в нужном направлении и применяем ее к узлу.

Вот и все! Простой Pong был написан с помощью нескольких строк кода.