Taller de Desarrollo de Videojuegos

ComCom

DC - FCEyN - UBA

May 8, 2026

¡Antes de empezar!

Y de que pregunten...

Todo el material del taller (tanto la demo del final como estas mismas diapos) puede ser descargado usando el siguiente link:

https://comcom.dc.uba.ar/talleres/gamedev

¿Que vamos a ver?

Pequeño indice

- Recorrido historico (mas no exhaustivo)
- Que es (y que no) un motor grafico
- Conceptos de Godot
- Interfaz y el uso
- Ejercicio controles
- Mas conceptos de Godot
- Ejercicio señales
- Explorar demo

Un poco de historia...

Los primeros años...

Los videojuegos tempranos (si es que podemos llamarlos así) eran más parecidos a experimentos entre científicos o estudiantes avanzados de computación y matemática. Piensen que las primeras computadoras eran calculadoras enormes, como Clementina.

Un poco de historia...

Los primeros años...

Los videojuegos tempranos (si es que podemos llamarlos así) eran más parecidos a experimentos entre científicos o estudiantes avanzados de computación y matemática. Piensen que las primeras computadoras eran calculadoras enormes, como Clementina.

En la foto se ve una reconstrucción de un osciloscopio reproduciendo el juego *Tennis For Two* (1958), considerado uno de los primeros videojuegos en la historia.

Un poco de historia...

Nacen las consolas

Cuando el alcance económico se empieza a ver, aparecen los primeros dispositivos dedicados al juego: las maquinitas de arcade, primero, y las consolas personales, luego.

Un poco de historia...

Nacen las consolas

Cuando el alcance económico se empieza a ver, aparecen los primeros dispositivos dedicados al juego: las maquinitas de arcade, primero, y las consolas personales, luego.
Las empresas que diseñaban este hardware armaban también los manuales o librerías que permiten utilizar los dispositivos.

Atari popularizó esta idea de tercerizar el diseño del juego a un equipo (generalmente compuesto por una persona) de desarrolladores.
Nintendo no fue muy fan de esa idea.

Un poco de historia...

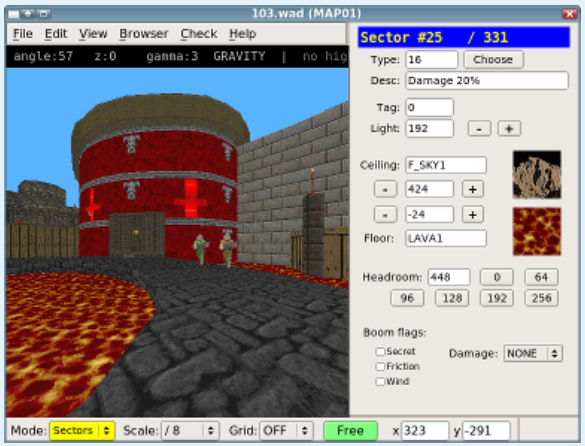
¿Jugar en PC?

A mediados de los 80 's el uso de las computadoras para jugar se empezó a hacer popular.

Un poco de historia...

¿Jugar en PC?

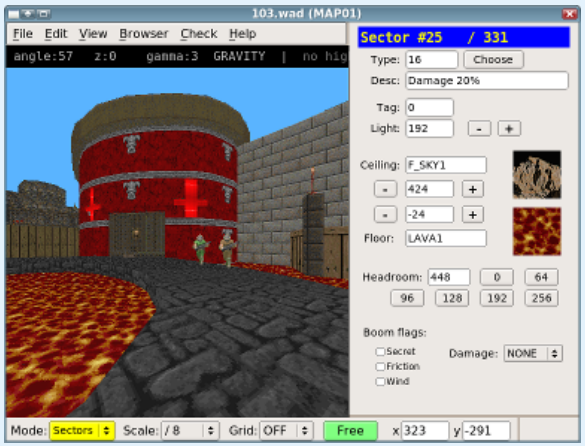
A mediados de los 80 's el uso de las computadoras para jugar se empezó a hacer popular.
El equipo de *ID Software*, un año después de la salida oficial de *Doom*, publicaron el código fuente del juego. La gente empezó a modificar este código para crear algo nuevo (lo que hoy se conoce como *modding*).



Un poco de historia...

¿Jugar en PC?

A mediados de los 80 's el uso de las computadoras para jugar se empezó a hacer popular.
El equipo de *ID Software*, un año después de la salida oficial de *Doom*, publicaron el código fuente del juego. La gente empezó a modificar este código para crear algo nuevo (lo que hoy se conoce como *modding*).



Para los curiosos, la cuenta de github de *ID Software*, donde estan los codigos de todos sus juegos: https://github.com/id-software

¿Como se hacen los videojuegos?

Desarrollo multifacetica

¿Como se hacen los videojuegos?

Desarrollo multifacetica

La tarea de hacer un videojuego involucra distintas áreas:

- El apartado técnico (lógica del juego, renderizado, optimización)
- El apartado visual (diseños, UI, animaciones)
- El apartado sonoro (música, efectos de sonidos)
- El apartado narrativo (guión, diálogos, personajes)
- El apartado lúdico (diseño de niveles, tutoriales)

¿Como se hacen los videojuegos?

Desarrollo multifacetica

La tarea de hacer un videojuego involucra distintas áreas:

- El apartado técnico (lógica del juego, renderizado, optimización)
- El apartado visual (diseños, UI, animaciones)
- El apartado sonoro (música, efectos de sonidos)
- El apartado narrativo (guión, diálogos, personajes)
- El apartado lúdico (diseño de niveles, tutoriales)

Nos vamos a quedar con el apartado técnico, así que en el mejor caso terminamos con algo feo pero divertido.

¿Como se hacen los videojuegos?

Tareas de un desarrollador

Con lo visto anteriormente, podemos separar al desarrollo de un videojuego en dos grandes tareas.

¿Como se hacen los videojuegos?

Tareas de un desarrollador

Con lo visto anteriormente, podemos separar al desarrollo de un videojuego en dos grandes tareas.

I/O

La entrada y la salida de la computadora, podríamos mencionar:

- Input del teclado o controles.
- La placa de video y la pantalla.
- Guardado y lectura de archivos en el sistema.

¿Como se hacen los videojuegos?

Tareas de un desarrollador

Con lo visto anteriormente, podemos separar al desarrollo de un videojuego en dos grandes tareas.

I/O

La entrada y la salida de la computadora, podríamos mencionar:

- Input del teclado o controles.
- La placa de video y la pantalla.
- Guardado y lectura de archivos en el sistema.

Logica

Cuando hablamos de lógica nos referimos a cómo la computadora responde a las acciones del jugador.
Calcular fuerzas como la gravedad, el movimiento de objetos, interacción entre los mismos, IAs, entre otras.

Entonces, un engine es...

Motores gráficos

Podríamos definir, finalmente, a un engine como la parte del software que se encarga de la primera tarea, la de entrada y salida.
Luego, un desarrollador utiliza las funcionalidades y abstracciones del motor para implementar la lógica de su juego.

Entonces, un engine es...

Motores gráficos

Podríamos definir, finalmente, a un engine como la parte del software que se encarga de la primera tarea, la de entrada y salida.
Luego, un desarrollador utiliza las funcionalidades y abstracciones del motor para implementar la lógica de su juego.
Ejemplos de motores modernos, sin intentar ser exhaustivo, son:

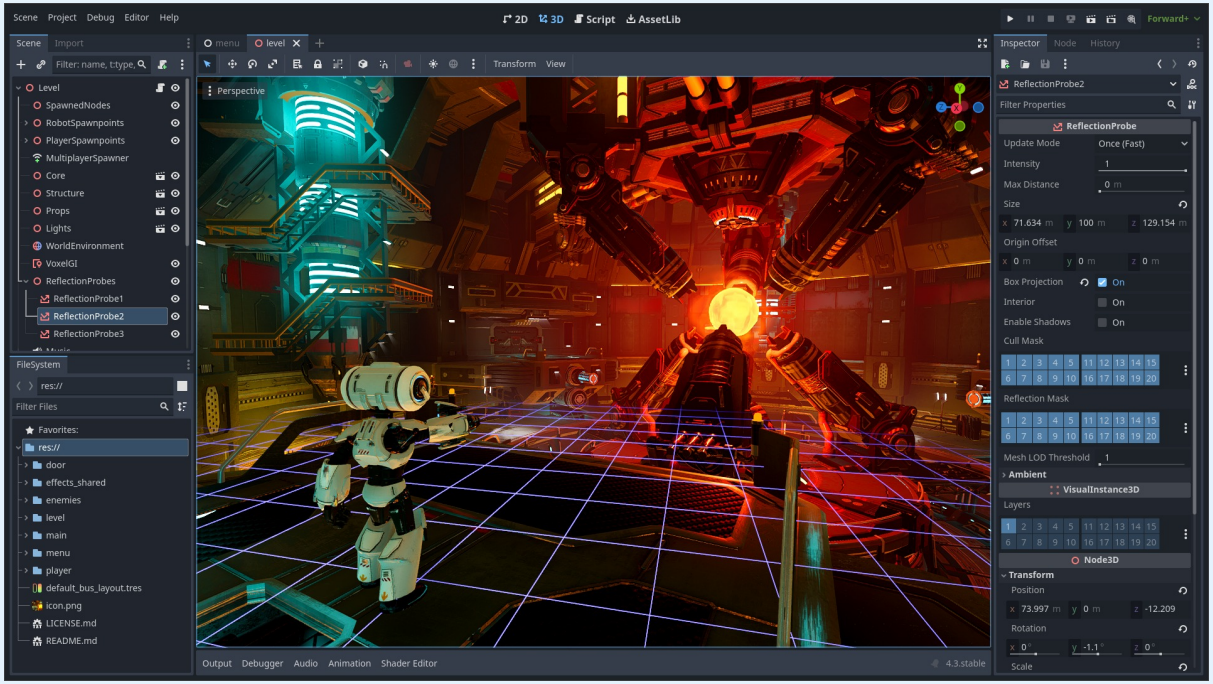
- Unreal
- Unity
- Godot

En este curso, nos enfocaremos en el uso de Godot.

El motorcito nacional

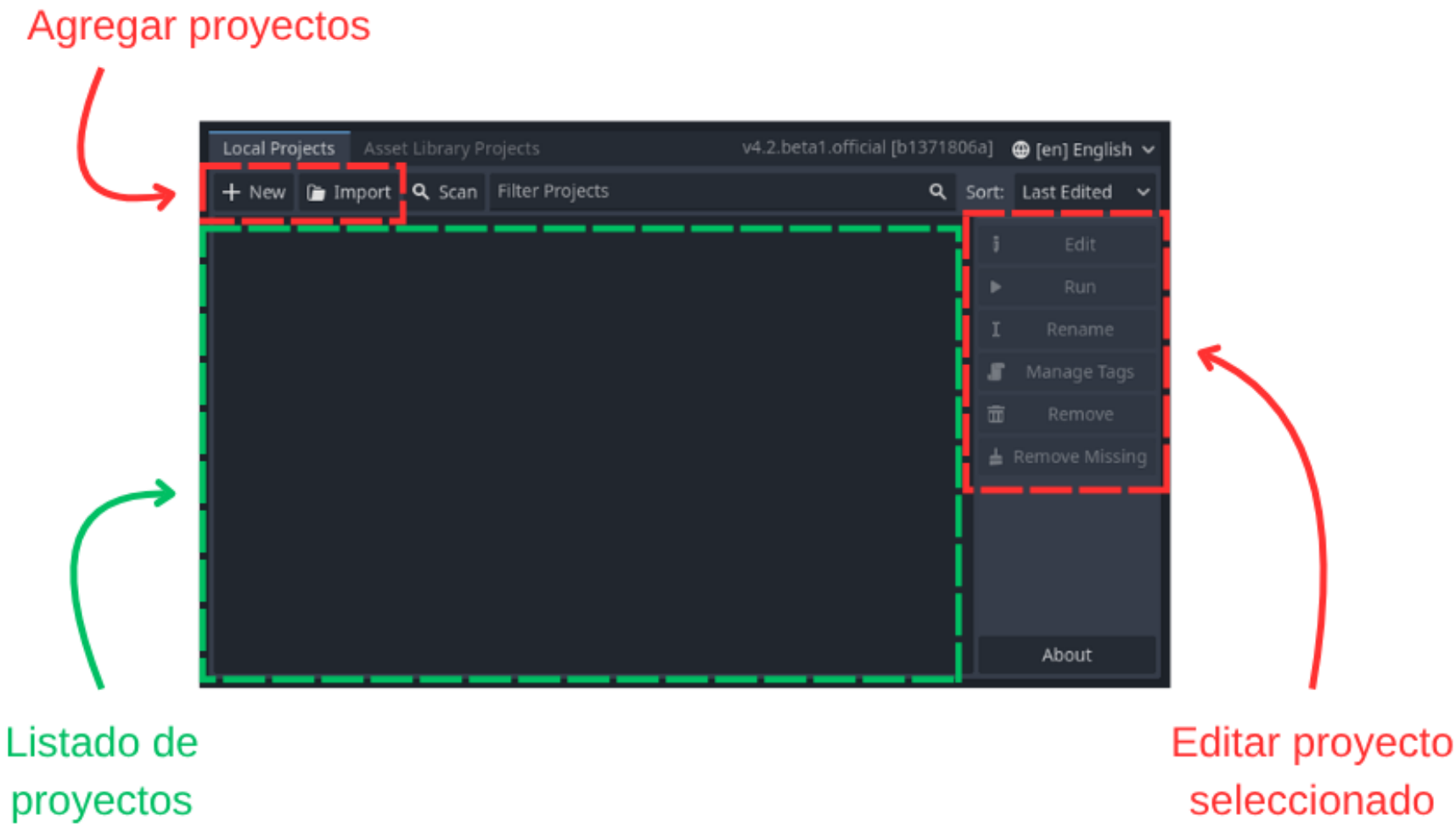
Godot

Godot es un motor gráfico, tanto para 2D como 3D. Es multiplataforma, gratuito y de código abierto desde 2014.

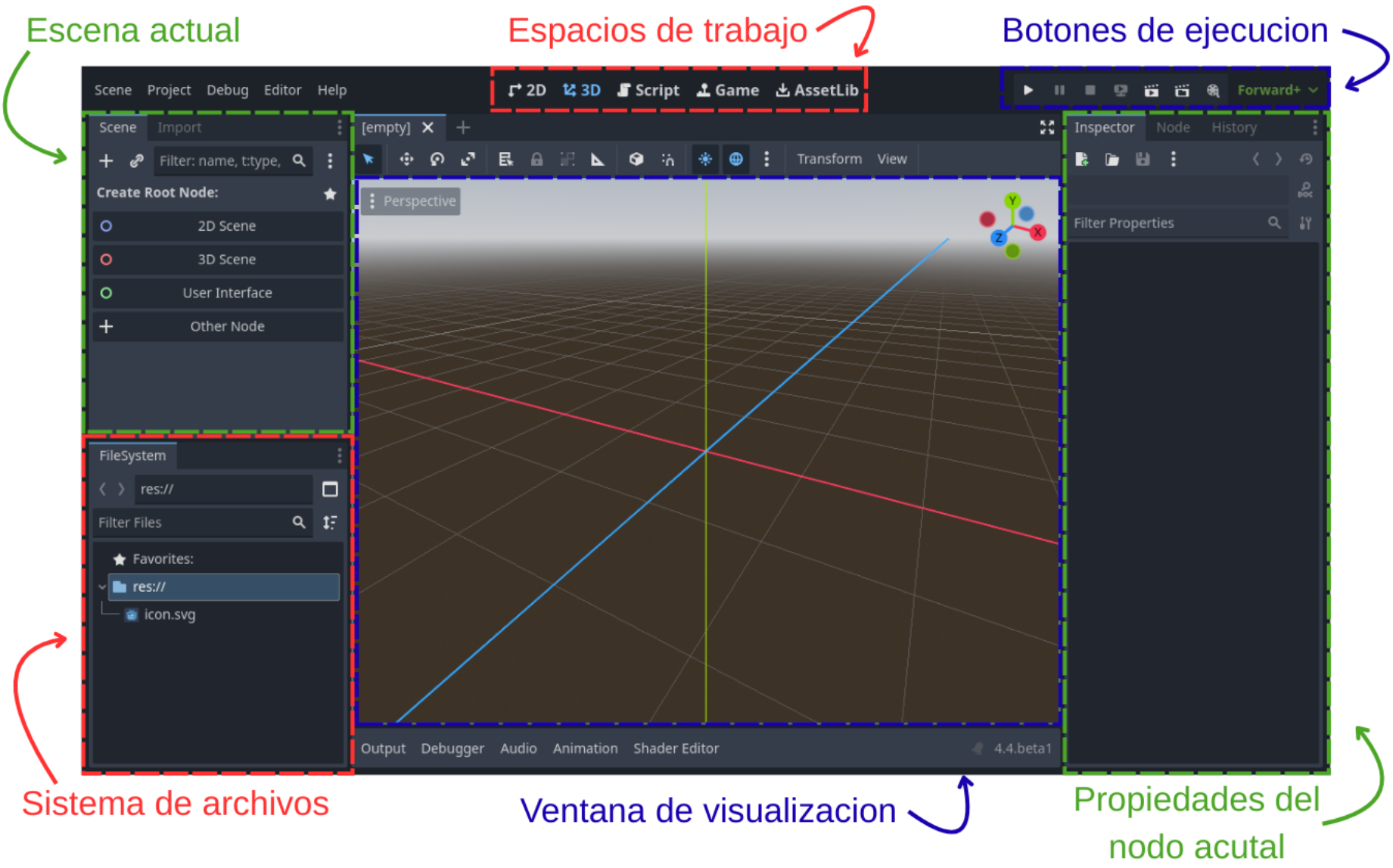


- Pagina oficial de Godot: https://godotengine.org/
- Repo de github de Godot: https://github.com/godotengine/godot

Conociendo el editor...



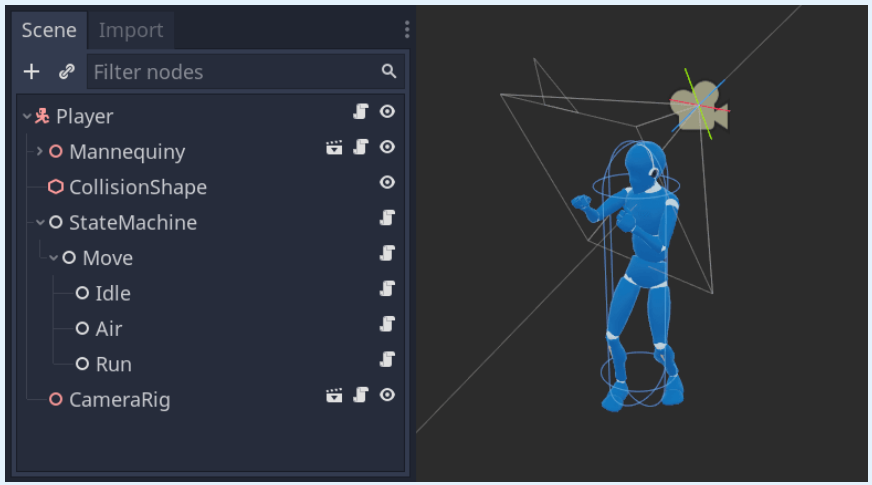
Conociendo el editor...



Conceptos básicos de Godot

Nodos y escenas

En Godot nuestra pieza de construcción van a ser las *escenas*. Las escenas van a ser todas nuestras entidades, niveles, *menus*, o prácticamente cualquier cosa interactuable (o no) del juego. Las escenas pueden estar constituidas por otras escenas o, en su defecto, por *nodos*.

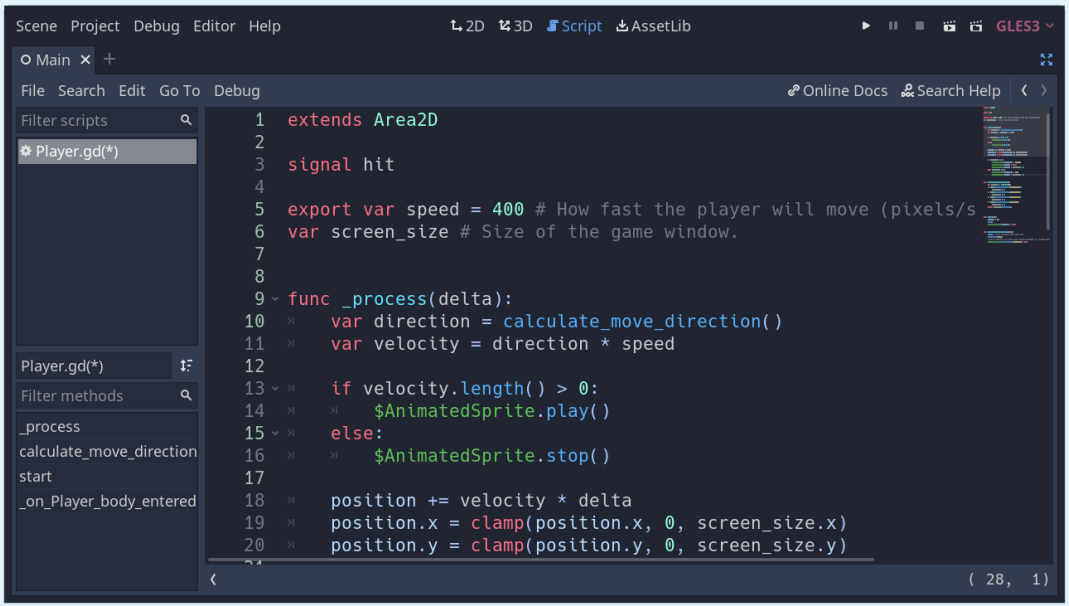


Los nodos, básicamente, son como "mini escenas" que Godot ya nos provee y tienen algún tipo de comportamiento básico, es decir, tienen propiedades y funciones que podemos utilizar.

Conceptos básicos de Godot

¡Código!

A su vez, a cada escena podemos vincularle un archivo de código para modificar su comportamiento. Godot tiene un lenguaje de programación propio, *GDScript*, bastante similar en sintaxis a Python.



Algo a destacar es que cada código de Godot debe empezar con la línea `extends` y el tipo del nodo del que venimos.

Ejercicio 1

Moviendo cosas

Ejercicio 1

Moviendo cosas

1. Una vez abramos Godot, creamos un proyecto nuevo.

Ejercicio 1

Moviendo cosas

1. Una vez abramos Godot, creamos un proyecto nuevo.
2. Vamos a crear una nueva escena, un `CharacterBody2D` con el logo de Godot para arrancar.

Ejercicio 1

Moviendo cosas

1. Una vez abramos Godot, creamos un proyecto nuevo.
2. Vamos a crear una nueva escena, un `CharacterBody2D` con el logo de Godot para arrancar.
3. Le agregamos un script nuevo, que debería por defecto tener la línea `extends CharacterBody2D` y no mucho más.

Ejercicio 1

Moviendo cosas

1. Una vez abramos Godot, creamos un proyecto nuevo.
2. Vamos a crear una nueva escena, un `CharacterBody2D` con el logo de Godot para arrancar.
3. Le agregamos un script nuevo, que debería por defecto tener la línea `extends CharacterBody2D` y no mucho más.
4. Iniciamos la función `_process`, en la que movemos la escena a gusto. Acá hay un ejemplo para "que dé vueltas".

```
extend CharacterBody2D

const SPEED : int = 10

func _process(delta):
    self.rotation = self.rotation + SPEED * delta
```

Habemus teclado

Botones...

Si quisiéramos, en un frame dado, chequear si se presionó un botón en el teclado podemos escribir la siguiente línea.

```
    if Input.get_action_pressed("ui_right"):
```

Ese texto es un renombre, pero no para una tecla específica, si no para una *acción*.

Habemus teclado

Botones...

Si quisiéramos, en un frame dado, chequear si se presionó un botón en el teclado podemos escribir la siguiente línea.

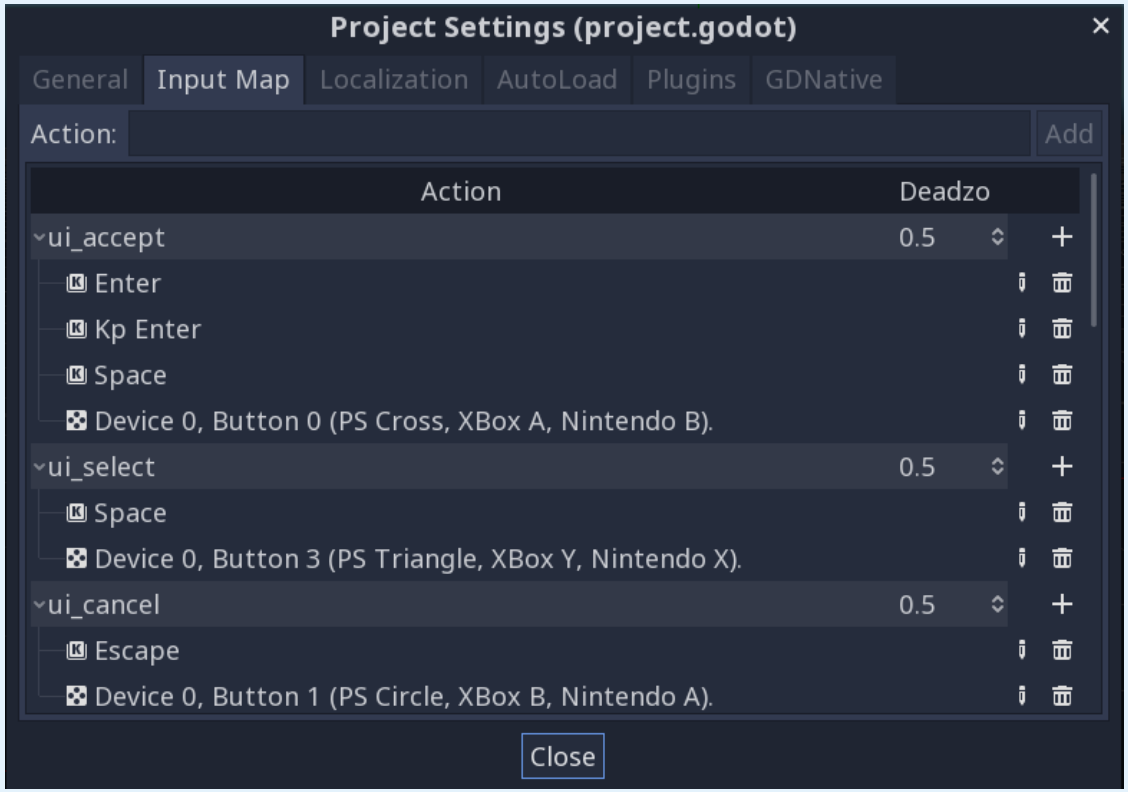
```
    if Input.get_action_pressed("ui_right"):
```

Ese texto es un renombre, pero no para una tecla específica, si no para una *acción*.
Godot nos permite crear acciones para una cantidad indefinida de botones. Esto es muy útil para lograr que distintos periféricos (como teclados, controles de consolas o celulares) se comporten, en principio, de la misma manera.

Habemus teclado

Mapeo de acciones

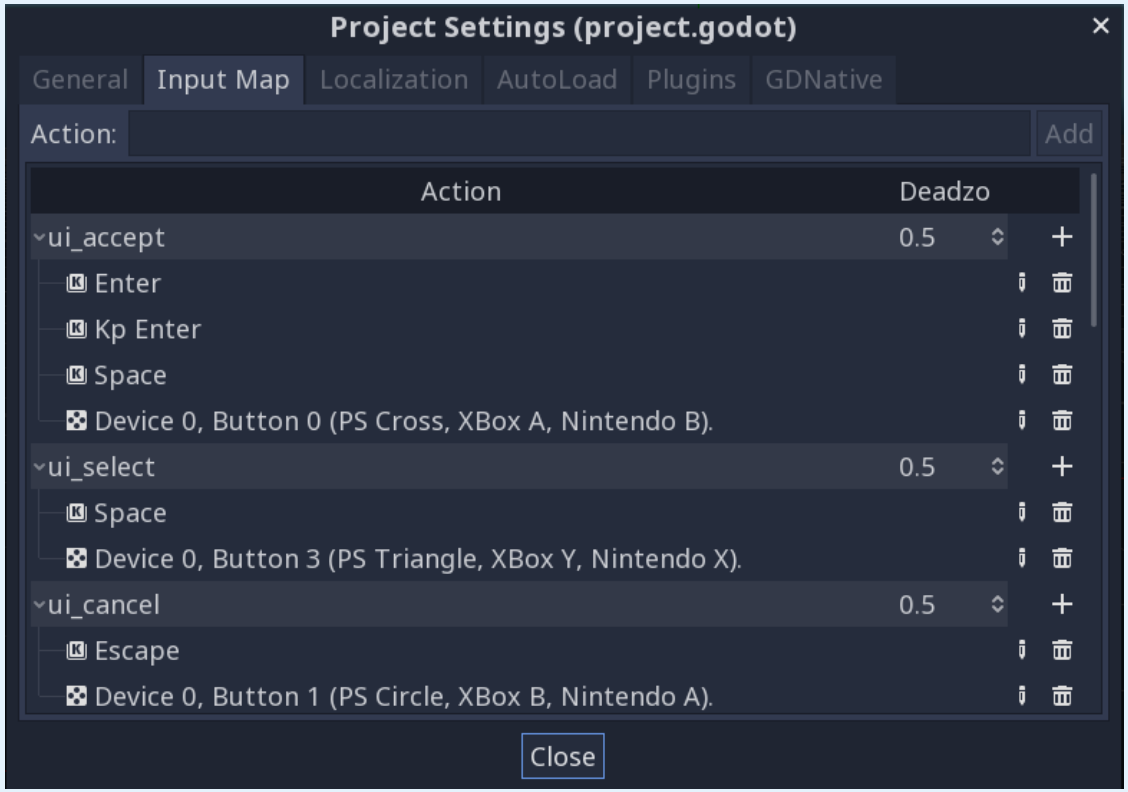
El mapa de input se encuentra en `Project -> Project Settings -> Input map`.



Habemus teclado

Mapeo de acciones

El mapa de input se encuentra en `Project -> Project Settings -> Input map`.



¡Prueben crear una acción nueva!

Detectando input...

Usamos el teclado

Ahora, hagamos algo parecido a un juego. Intentemos que el Godot se mueva con una tecla para una dirección (por ejemplo, con la `a` hacia la izquierda), y con otra tecla hacia la dirección opuesta (continuando con el ejemplo, con la `d` a la derecha).

Detectando input...

Usamos el teclado

Ahora, hagamos algo parecido a un juego. Intentemos que el Godot se mueva con una tecla para una dirección (por ejemplo, con la a hacia la izquierda), y con otra tecla hacia la dirección opuesta (continuando con el ejemplo, con la d a la derecha).

```
extends CharacterBody2D

const SPEED : int = 300

func _process(delta):
    if Input.get_action_pressed("ui_right"):
        self.velocity.x = SPEED
    elif Input.get_action_pressed("ui_left"):
        self.velocity.x = SPEED
    else:
        self.velocity.x = 0
    move_and_slide() #<-- ¡No se olviden de esta función!
```

Los nodos hablan...

Señales

Godot nos da la posibilidad de que las distintas escenas que formemos interactúen entre sí. Para lograr esto se utiliza el concepto de *señal*, que podemos pensarlo como un mensaje que un nodo le manda a otro.

Los nodos hablan...

Señales

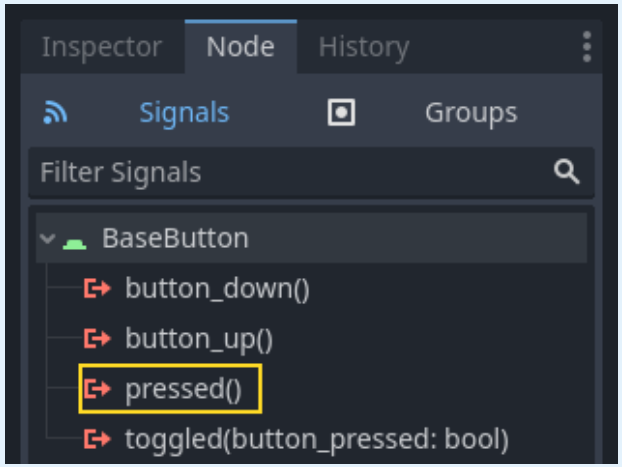
Godot nos da la posibilidad de que las distintas escenas que formemos interactúen entre sí. Para lograr esto se utiliza el concepto de *señal*, que podemos pensarlo como un mensaje que un nodo le manda a otro. Ejemplos de señales podrían ser:

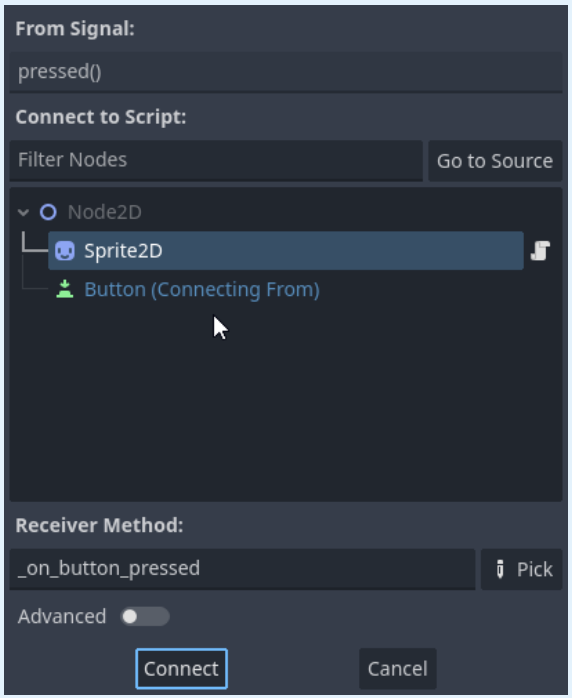
- Un botón es presionado
- Un timer termina
- Dos objetos colisionan
- Una escena es creada

Los nodos hablan...

Usando señales

Para conectar una señal, primero seleccionamos el nodo que la va a mandar, y abrimos el menú de señales. Qué señales vamos a poder mandar está determinado por el tipo del nodo.
Al elegir una señal, Godot debería preguntarnos con qué nodo queremos conectarlo, como se ve en el menú de la derecha.





Ejercicio 2

Señalar es de mala educación

Ejercicio 2

Señalar es de mala educación

1. Crear una escena nueva que sea un `Area2D`, le agregamos su debida caja de colisión y un script nuevo.

Ejercicio 2

Señalar es de mala educación

1. Crear una escena nueva que sea un `Area2D`, le agregamos su debida caja de colisión y un script nuevo.
2. Seleccionamos la `CollisionShape2D`, y modificamos su atributo `debug_color` por un color de nuestro agrado.

Ejercicio 2

Señalar es de mala educación

1. Crear una escena nueva que sea un `Area2D`, le agregamos su debida caja de colisión y un script nuevo.
2. Seleccionamos la `CollisionShape2D`, y modificamos su atributo `debug_color` por un color de nuestro agrado.
3. Teniendo seleccionado el nodo del `Area2D`, vamos a contectar la señal `on_body_entered` al script que acabamos de crear.

Ejercicio 2

Señalar es de mala educación

1. Crear una escena nueva que sea un `Area2D`, le agregamos su debida caja de colisión y un script nuevo.
2. Seleccionamos la `CollisionShape2D`, y modificamos su atributo `debug_color` por un color de nuestro agrado.
3. Teniendo seleccionado el nodo del `Area2D`, vamos a contectar la señal `on_body_entered` al script que acabamos de crear.

El código completo podría verse así:

```
extends Area2D

func _on_body_entered(body):
    body.modulate = Color.RED
```

Los juegos son como las cebollas

¿Te hacen llorar?

Los juegos son como las cebollas

¿Te hacen llorar?

¡NO! (generalmente) ¡Tienen capas!

¿Capas? ¿Y ahora mascaras?

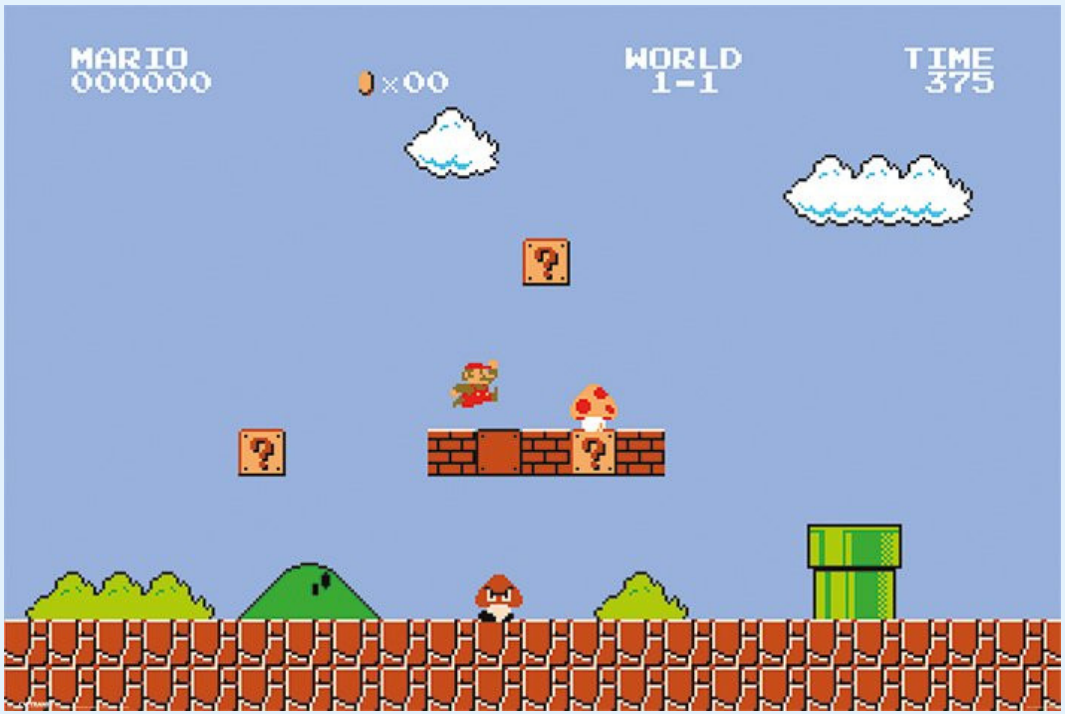
Un ejemplo poco conocido

Las capas de colision existen desde siempre. Es normal que en un juego queramos que ciertas cosas interactuen entre si, pero que otras se ignoren.

¿Capas? ¿Y ahora mascaras?

Un ejemplo poco conocido

Las capas de colision existen desde siempre. Es normal que en un juego queramos que ciertas cosas interactuen entre si, pero que otras se ignoren.

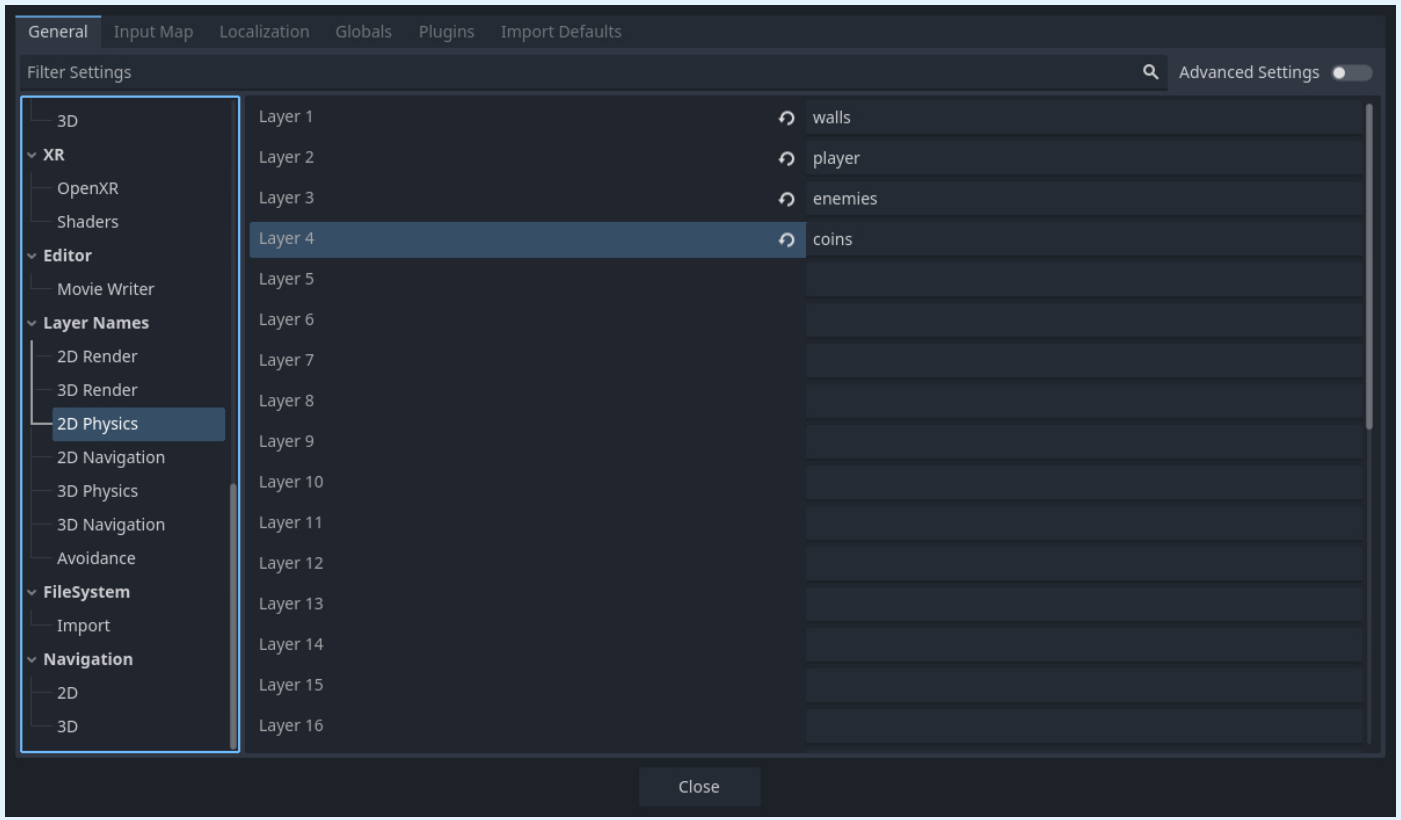


Por ejemplo, ¿tiene sentido que los *goomba* detecten las monedas?

Capas y mascaras

Siendo invisible

Godot ya tiene diseñado un sistema de capas y mascaras, todas numeradas del 1 al 32. Además de esto, podemos nombrarlas dentro de `Project -> Project Settings -> General`.



Ejercicio 3

Veo, veo...

Ejercicio 3

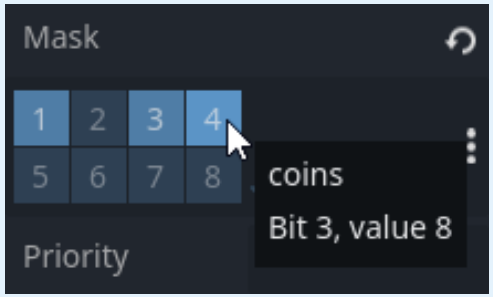
Veo, veo...

1. Vamos a crear una nueva escena que consista en un `StaticBody2D`, que tenga una caja de colisión y un `ColorRect` (para que se pueda ver, puede ser otra vez el logo de Godot si sienten el atrevimiento). Esta va a ser nuestro objeto "solido", como una pared.

Ejercicio 3

Veo, veo...

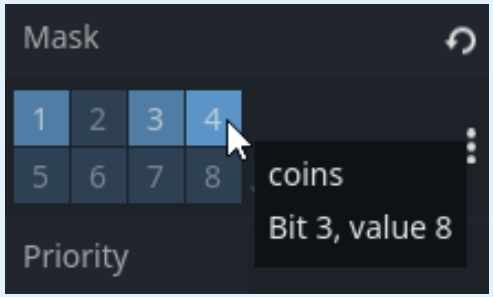
1. Vamos a crear una nueva escena que consista en un `StaticBody2D`, que tenga una caja de colisión y un `ColorRect` (para que se pueda ver, puede ser otra vez el logo de Godot si sienten el atrevimiento). Esta va a ser nuestro objeto "solido", como una pared.
2. En la escena principal, poner una o varias paredes, y buscar en el inspector el apartado de mascaras y capas de colisión.



Ejercicio 3

Veo, veo...

1. Vamos a crear una nueva escena que consista en un `StaticBody2D`, que tenga una caja de colisión y un `ColorRect` (para que se pueda ver, puede ser otra vez el logo de Godot si sienten el atrevimiento). Esta va a ser nuestro objeto "solido", como una pared.
2. En la escena principal, poner una o varias paredes, y buscar en el inspector el apartado de mascaras y capas de colisión.



3. Prueben editar los distintos valores de las capas de colisión, tanto del jugador como de las paredes, o también de las `Area2D` que agregamos antes.

Tenemos un juego, pero... ¿y ahora que?

¿Como se expande? ¿Que agregamos? ¿Que falta?

Ya llegamos a algo que *parece* una demo. Pero esta carente de varias cosas para llegar a ser un juego, ¿no?
Algunas de las cosas que podriamos tener en cuenta para el proximo desarrollo son:

- **Mecanicas**: un set básico de movimiento esta bien, pero que tal enemigos, coleccionables, estadisticas para el jugador.
- **Sonido y musica**: bastante autodescriptivo, es de lo mas importante a la hora de traer vida al juego.
- **Arte**: seguro lo que tienen frente a ustedes no es muy amigable a los ojos, por lo que buscar algun *asset* de sprites o incluso hacer uno propio seria una gran actualización.
- **Menu y UI**: suele ser la presentación al jugador, no es facil saber que mostrar y como.

Y mucho mas, realmente. ¡Lo que se les ocurra!

¿Necesitan inspiración?

La GameJam de Exactas

Durante el verano de 2026 se hizo la primera edición de la GameJam de Exactas, que les dio a varios desarrolladores 2 semanas para hacer un juego segun un tema dado.

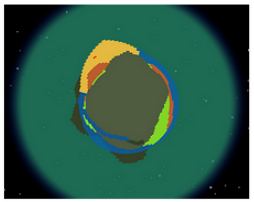
Planeta Olvidado
tomobossi
¿Cómo podrías recordar algo que no sabías que olvidaste?

Quilombo Cósmico
t0m45DEV
Un plataformas 2D ridiculo ambientado en el espacio.

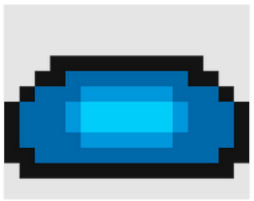
Forged Planet
BerTobi
Un juego de minar un asteroide y construir torretas

planetRhythmXD
valenrc, Batuel
Un juego de ritmo sobre un planeta al borde de ser olvidado.

Torneo del Artificiero
lenethxcv, JuanPaT
Combate por turnos estilo slay the spire con reacciones elementales



pnalet
Koliao



Baballero
EmpanadasCursed
Para el uba exactas gamejam 2026

El Errante
fabri_frito

Project: OPHANIM
Fidel_Barreiros, Thelpix, Divitrion

La Goma Gómez
Project909
Una goma perdida se adentra en lo más profundo de la caja de objetos perdidos

Exactas Roads
Radiante Perdido, alplex
1er Exactas Game Jam

Fracti Dialogus
kigoyuto, LucaPancetitA
Explora este misterioso planeta mientras descubres el conocimiento absurdo de los habitantes

Buscando a Dios
Torbernight, makiniqa
Un jueguito donde seguis la palabra de dios

UBA-O-MATIC
colltoaction
Edición Bicentenario: El Artefacto Absurdo

¡Son todos juegos hechos por gente de la facultad!
Para jugar y ver los resultados: https://itch.io/jam/dcuba-jam-01

Me prometieron una demo...

¡Hora libre!

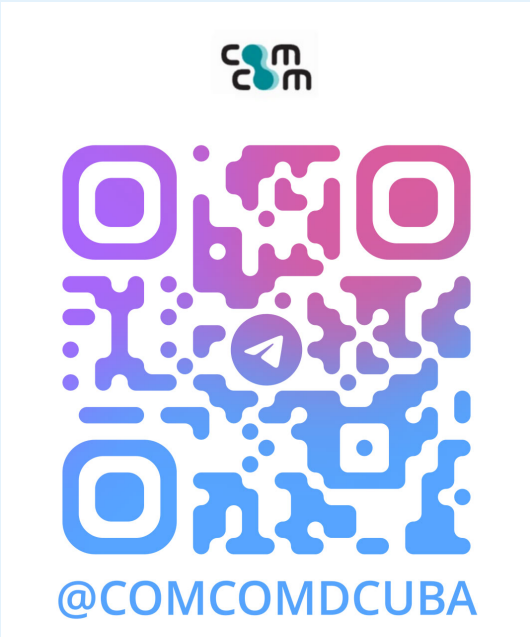
En la página de CubaWiki, en el apartado de CubaWeeki del taller de GameDev, o entrando al link de abajo, pueden descargar una demo jugable de un juego de plataformas en 2D.
Pueden editarla, agregar mecánicas u obstáculos nuevos, o simplemente jugar.

https://comcom.dc.uba.ar/talleres/gamedev

¿Quien organizo todo esto?

¡La comisión de estudiantes!

La ComCom (Comisión de estudiantes de Computación) es la encargada de diseñar, organizar y dictar talleres y actividades para los estudiantes, impulsadas por los mismos estudiantes. Todos formamos parte de la ComCom.



¡Sumate para darnos tu feedback o proponer ideas nuevas!