Principio de Testing

Taller 3 GoDot

Parte 1

- Primer Juego: Animaciones y movimientos

Ing. Eduardo Kunysz
EPET12
San Martín de los Andes

Introducción

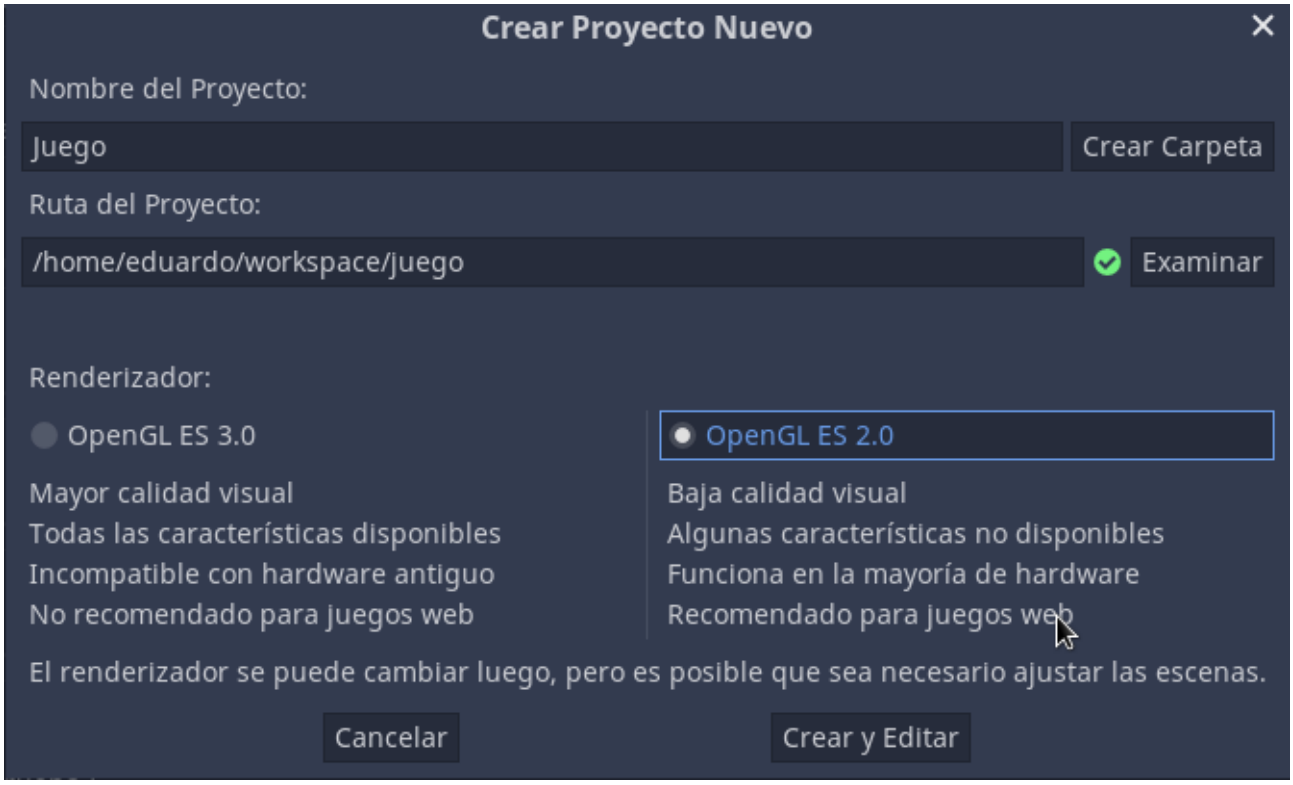
En este proyecto, que va a ser nuestro segundo proyecto en la interfaz, desarrollaremos un simple juego. Nos va a permitir, conocer algunos de los nodos de interacción. Y continuaremos practicando GDscript.

Sumario

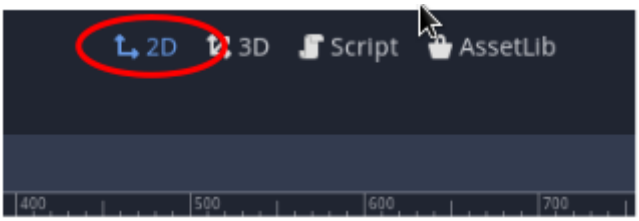
Comenzando el proyecto

Creando el proyecto

Abrir Godot y crear un nuevo proyecto (recuerda que debes elegir una carpeta vacía para guardarlo, de preferencia una creada exclusivamente para este proyecto).



Para este proyecto, al ser simple y no requerir calidad visual, podemos elegir la opción de OpenGL 2.0.



Luego presionando Crear y Editar entramos en nuestra interfaz de GoDot.

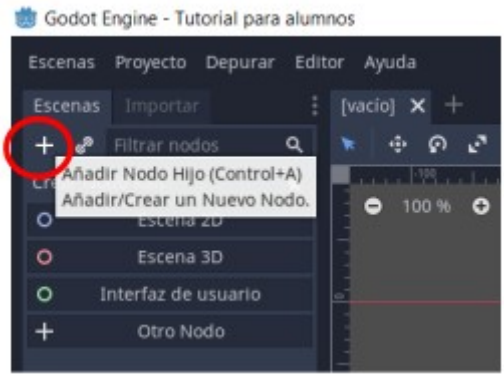
Pasamos la Interfaz a 2D en el menú superior.

Comenzando con una Animación Sprite2D

En este tutorial veremos cómo crear personajes 2D animados con la clase AnimatedSprite. Típicamente, cuando creas o descargas un personaje animado, éste viene en uno de estos dos modos: como imágenes individuales o como una sola hoja de sprites conteniendo todos los frames de la animación. Ambos pueden ser animados en Godot con la clase AnimatedSprite.

Colocando los Nodos en el árbol

En el menú contextual de Escena, seleccionamos el “+”



Creando Aread2D

Vamos a agregar un nodo del tipo “Area2D”

Area2D

Imagina el nodo Area2D como un "detector de colisiones" en tu juego. Es como si fuera un sensor que puede sentir cuando otros objetos o personajes chocan o entran en su área.

Descripción

1. Área: El nodo Area2D crea una especie de área invisible alrededor de sí mismo. Esta área tiene una forma específica, como un círculo o un rectángulo, que tú puedes configurar. El área puede ser más grande o más pequeña dependiendo de tus necesidades.
2. Detección de colisiones: Cuando otro objeto, como un personaje o un enemigo, entra o sale del área del nodo Area2D, este "sensor" lo detecta y te avisa a través de eventos en el juego.
3. Eventos: Los eventos son acciones especiales que ocurren cuando suceden ciertas cosas en el juego. Cuando el nodo Area2D detecta una colisión, se activa un evento que tú puedes programar para que ocurra algo específico, como hacer que el personaje se detenga, hacer sonar un efecto de sonido o activar una animación.
4. Interacciones: Puedes utilizar el nodo Area2D para crear interacciones interesantes en tu

juego. Por ejemplo, podrías usarlo para detectar cuando el jugador toca un objeto importante o cuando dos personajes chocan entre sí.

Para utilizar las clases AnimatedSprite como veremos a continuación, se pueden utilizar también (además de Area2D que es la mas simple) como nodos raíz: *KinematicBody2D* o *RigidBody2D*.

Haremos doble click sobre el nombre y lo renombraremos como “Player” (se le puede poner cualquier nombre, pero para este ejemplo respetemos estos nombres, porque luego cuando realicemos el código GDScript, tal vez debamos llamar a algún nodo especifico).

Agregando AnimatedSprite

Ahora, vamos a agregar otro nodo llamado “AnimatedSprite”. Para ello primero seleccionamos el nodo “Player” y luego clickeamos en “+”. Esto lo hacemos para que el nuevo nodo quede como hijo del nodo anterior.

AnimatedSprite

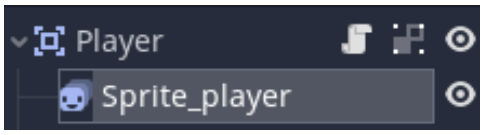
Imagina el nodo AnimatedSprite como un "dibujante animado" en tu juego. Es como si tuvieras un artista que puede dibujar diferentes cuadros de animación y mostrarlos uno tras otro para crear una animación en movimiento.

Descripción:

1. Sprite: El nodo AnimatedSprite es un tipo especial de sprite, que es una imagen o dibujo que se muestra en la pantalla del juego. Sin embargo, en lugar de mostrar una sola imagen estática, este sprite puede cambiar su apariencia mostrando diferentes cuadros de animación.
2. Cuadros de animación: Un cuadro de animación es una imagen individual que forma parte de una secuencia animada. Puedes tener varios cuadros que representen diferentes poses o estados de un personaje o objeto en movimiento.
3. Animación: Cuando el nodo AnimatedSprite está en funcionamiento, va cambiando rápidamente entre los diferentes cuadros de animación, creando la ilusión de movimiento. Esto se llama animación y es como si el dibujante estuviera mostrando una serie de dibujos en rápida sucesión para que veamos algo en movimiento.
4. Cambio de animación: Tú puedes programar el nodo AnimatedSprite para que cambie entre diferentes secuencias de animación dependiendo de lo que esté ocurriendo en el juego. Por ejemplo, puedes tener una animación para caminar, otra para saltar, otra para atacar, etc.

El nodo AnimatedSprite es un poderoso recurso para dar vida a tus personajes y objetos en el juego. Con su ayuda, puedes crear animaciones interesantes y hacer que tu juego se vea más vivo y dinámico. Es como si tuvieras a un talentoso artista animando tus personajes para que se muevan y

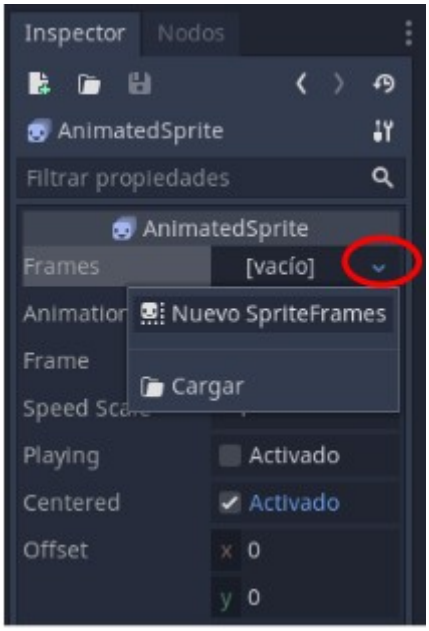
actúen en el mundo del juego.



Lo renombraremos como **Sprite_player**

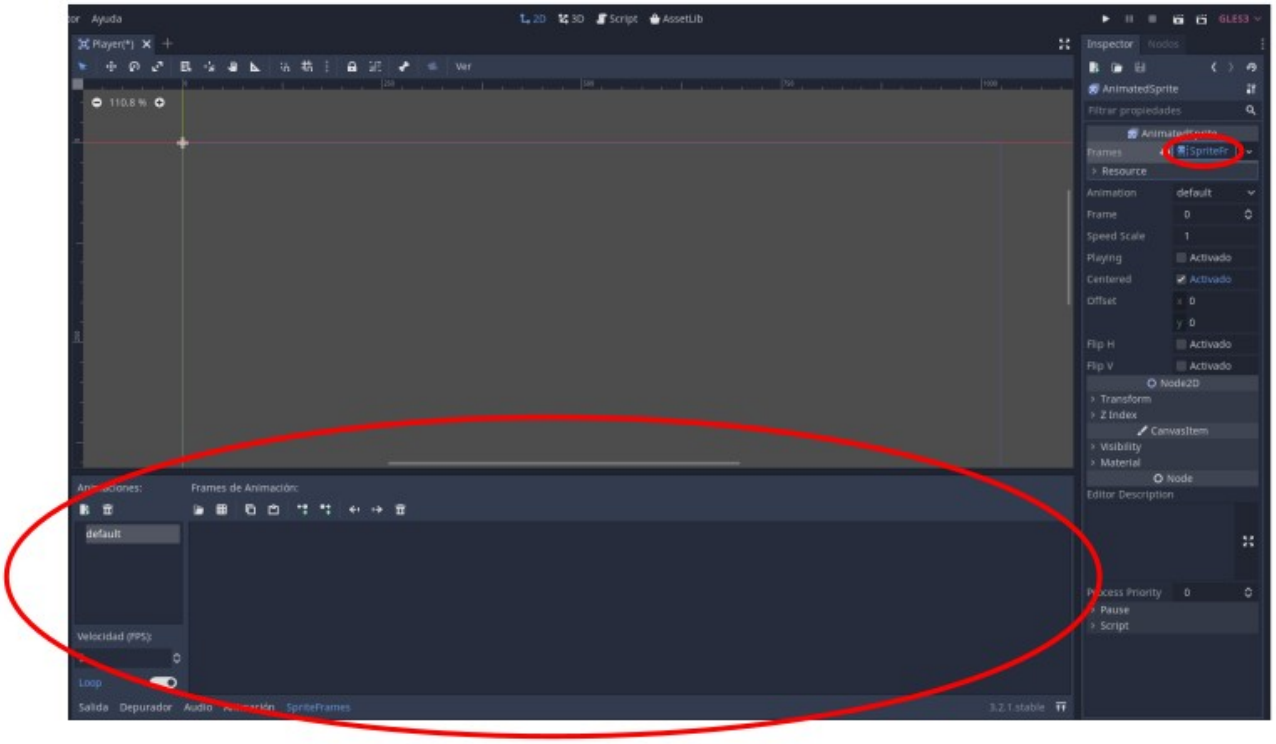
Configurando AnimatedSprite

Luego, seleccionamos Sprite_player y en el menú contextual de la derecha, vemos las propiedades del mismo. En la opción de Frame seleccionamos Nuevo Sprite Frames, de tal forma que se vea de la siguiente forma:



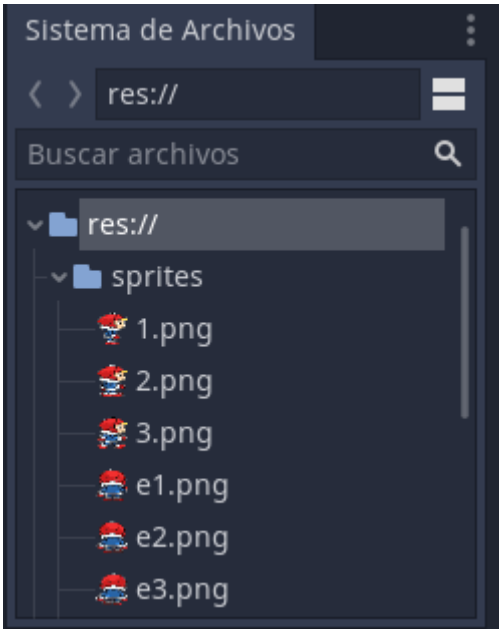
Panel de animaciones.

Presiona sobre la opción SpriteFrame que acabas de agregar, para que se despliegue en la parte inferior de la ventana el panel para agregar animaciones.



Carga de los sprites.

Descomprimimos el archivo adjunto con los sprites dentro de la carpeta del proyecto en na carpeta llamada “Sprites”.



De tal forma que la estructura del proyecto se vea de la siguiente forma.

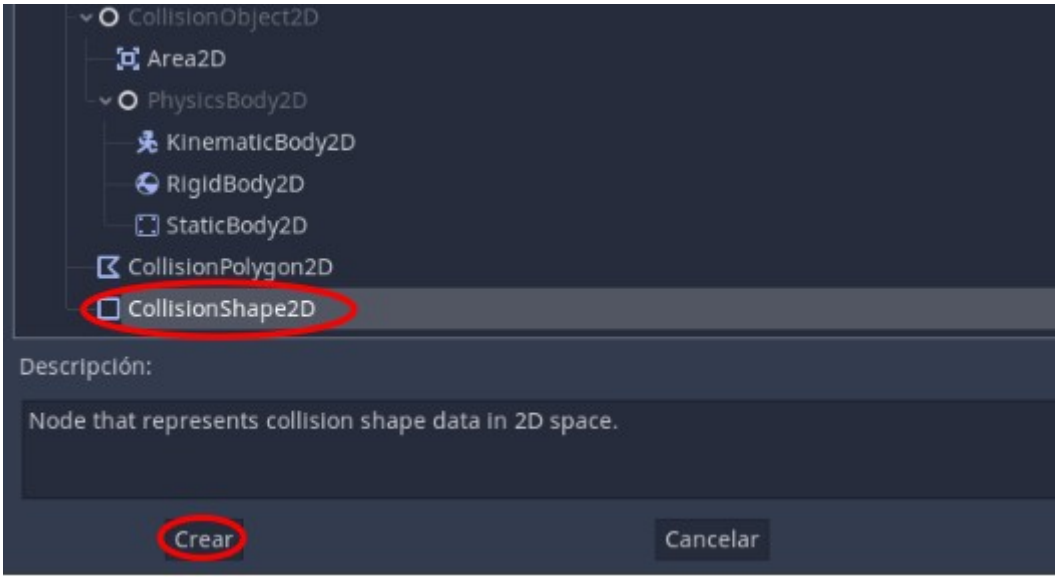
En el sistema de archivos ingresas a la carpeta “Sprites” (doble clic sobre dicha carpeta) para que muestre las imágenes que agregaste. Luego editas el nombre de la animación “default” y le colocas de nombre “**Frente**”, también agregas otras dos animaciones, las cuales llamaremos “**atrás**”, “**lado**” y “**quieto**”

Agregando los Sprites a las Animaciones

Arrastras las imágenes (Sprites) correspondientes a cada movimiento, dentro de cada animación.

Nodo CollisionShape2D

Agregar Nodo (hijo de Player) de tipo CollisionShape2D (antes puedes cambiar el tamaño de tu Player para que sea más visible, utilizando Shift y Clic sobre sostenido sobre una de las esquinas)



CollisionShape2D

Imagina el nodo CollisionShape2D como una "cápsula de colisión invisible" que rodea a tus objetos en el juego. Es como si crearas un escudo protector alrededor de ellos para que sepan cuándo chocan con algo más en el mundo del juego.

Descripción:

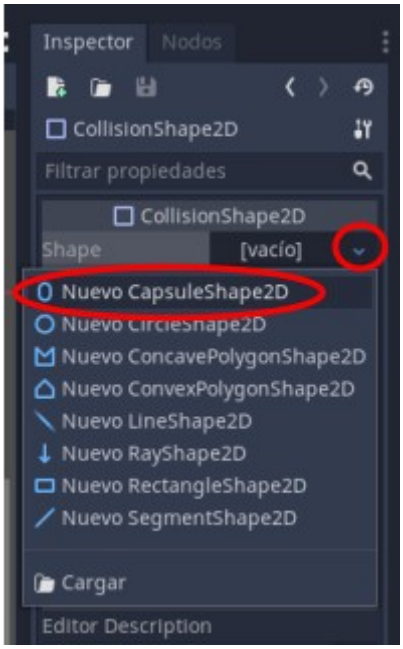
1. Protección invisible: El nodo CollisionShape2D no es visible en el juego, lo cual significa que no se ve, pero está ahí. Es como una cápsula que rodea a tus personajes u objetos, pero nadie puede verla.
2. Detección de colisiones: Cuando otro objeto o personaje entra en contacto con la cápsula de colisión del nodo CollisionShape2D, ¡boom!, se detecta una colisión. Es como si la cápsula fuera sensible al tacto y puede sentir cuando algo más está cerca.
3. Eventos de colisión: Cuando ocurre una colisión, puedes programar el juego para que suceda algo específico. Por ejemplo, puedes hacer que el personaje se detenga cuando choque con una pared o que recoja un objeto cuando entre en contacto con él.

4. Forma de colisión: La cápsula de colisión puede tener diferentes formas, como un círculo, un rectángulo o un polígono. Esto te permite ajustar la forma de la cápsula para que se ajuste mejor al contorno de tu personaje u objeto.

El nodo CollisionShape2D es una herramienta muy útil para manejar colisiones en tu juego. Te permite proteger tus objetos y personajes con cápsulas invisibles que detectan cuándo entran en contacto con otros objetos. Es como un "escudo protector" que ayuda a tu juego a saber cuándo ocurren colisiones y qué hacer en esos momentos.

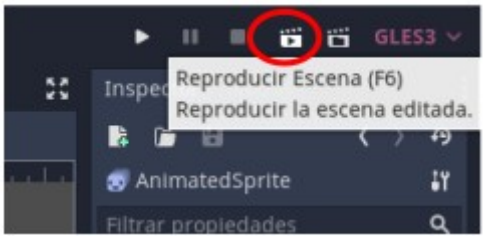
Configurar la cápsula de colisión

En el Inspector, selecciona el Shape CapsuleShepe2D. Ahora mueve y cambia el tamaño de la Cápsula de Colisión para que encaje con el personaje lo mejor posible.



Previsualizando la Escena

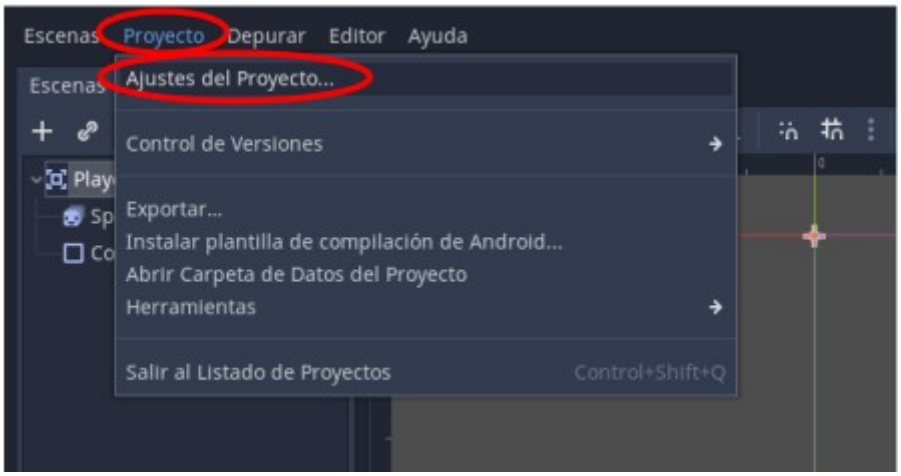
Este sería el equivalente a “Ejecutar” un programa, en esta ocasión, al presionar F6 (o el botón encerrado en un círculo) cargaría la escena completa con lo que llevamos configurado (únicamente la pantalla con el personaje principal).



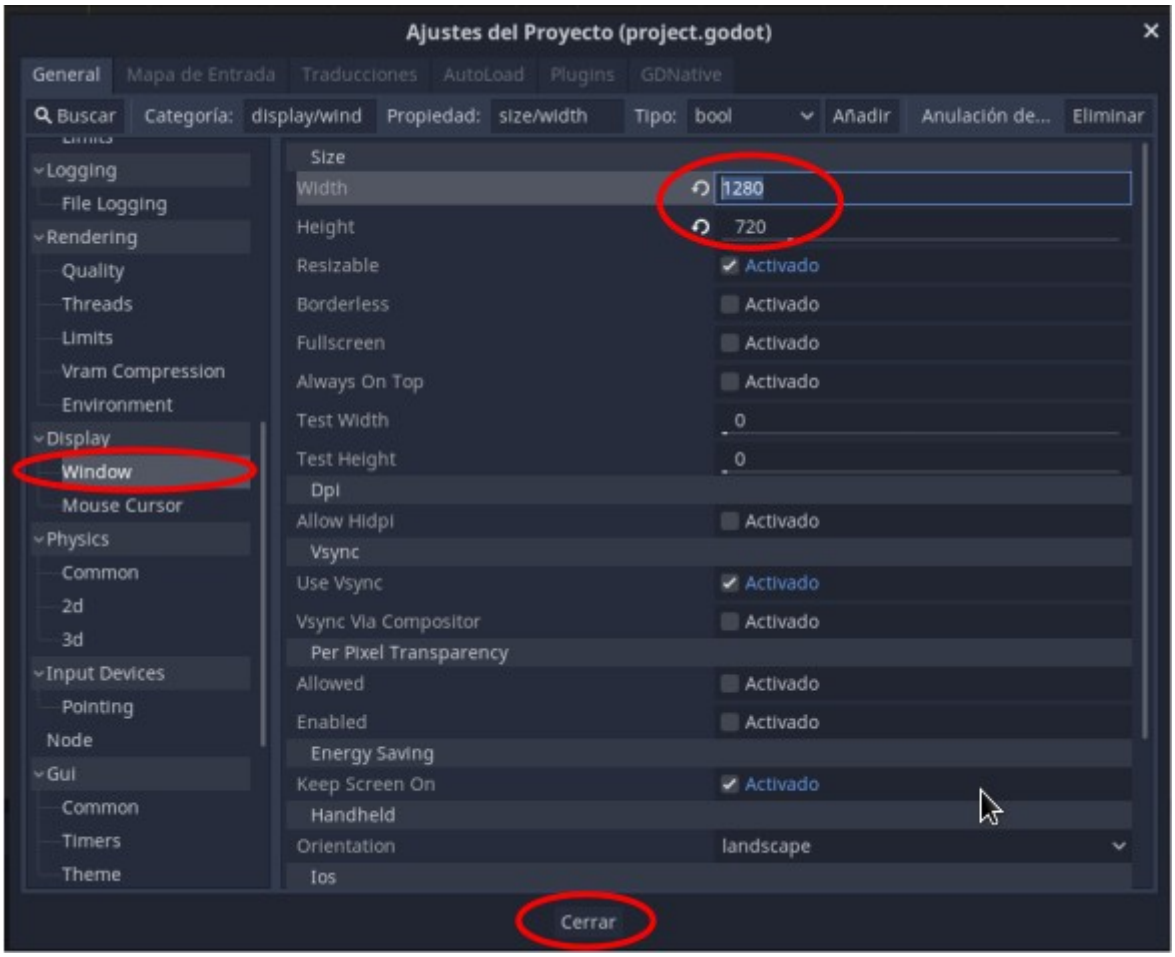
Dimensiones de la pantalla del juego

Ahora cambiaremos el tamaño de la ventana en que se ejecuta el juego. Normalmente, esta resolución dependerá de varios factores, por ejemplo, de en qué tipo de dispositivo se ejecutará el juego y de los elementos de diseño gráfico que incluirá (fondos, personajes, elementos del escenario, etc.).

Para cambiar esta configuración, accedemos al menú Proyecto y luego a la opción Ajustes del Proyecto



Y en la ventana que te aparece, buscas la opción Window (en el panel lateral, debajo de Display) y asignas las dimensiones que consideres correctas. Para este proyecto, aplicaremos un ancho (Width) de 1280 y un alto (Height) de 720. Y una vez configurados estos valores, cerramos esta ventana.



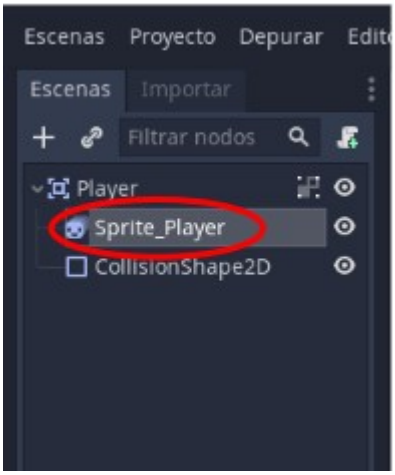
Agrupando a los hijos del nodo

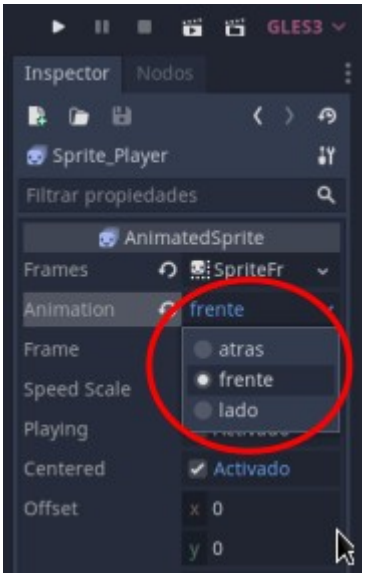
Si intentamos mover a nuestro personaje, notaremos que la forma de colisión y el sprite se mueven por separado, lo que puede dificultar tu trabajo, entonces, lo que haremos será “agrupar” ambos objetos, haciendo que los nodos hijos no sean seleccionables.

Con el nodo Player seleccionado, presiona sobre el botón que se indica en la imagen:

Seleccionando la animación por defecto

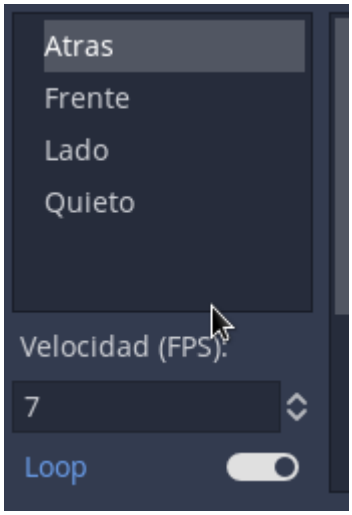
Debes seleccionar el nodo Sprite_Player y luego dirigirte al inspector, en donde encontrarás la opción de Animation con su respectivo menú desplegable. Puedes elegir cualquiera de las animaciones pero en este proyecto elegiremos la frontal.





Cambiando la velocidad de la animación

Los Frames Per Second (FPS / Fotogramas por segundo) son los encargados de reglar la velocidad de la animación, para cambiarlos y colocarle una velocidad correcta, debes habilitar el panel de SpriteFrames y modificar el campo de Velocidad (FPS). Por ejemplo, en este proyecto asignaremos 7 FPS.



Hacemos lo mismo en las tres animaciones (atrás, frente y lado)

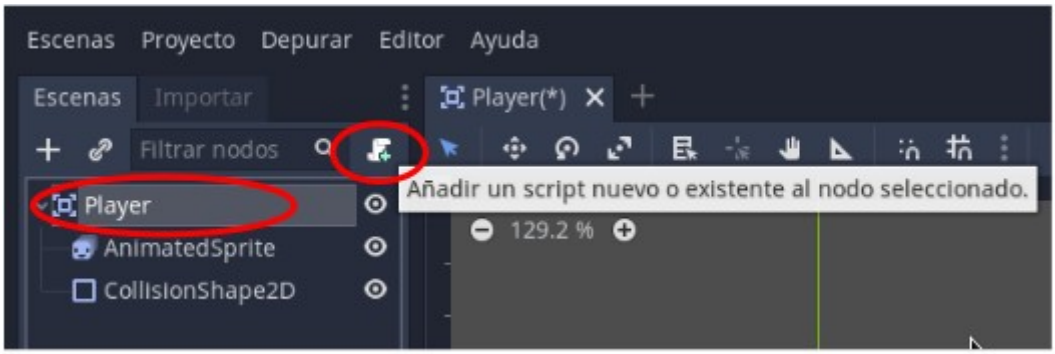
GdScript

Hasta ahora, solo hemos preparado la escena, y organizado los nodos. Es momento de darle vida a nuestra animación.

Como vimos en el talles de la calculadora, GoDot soporta una gran variedad de lenguajes de programación. Seguiremos utilizando el lenguaje nativo del motor de desarrollo, **GdScript**.

Para ello necesitamos seleccionar un nodo sobre el que desarrollaremos nuestro script. Por lo tanto seleccionaremos el nodo Player.

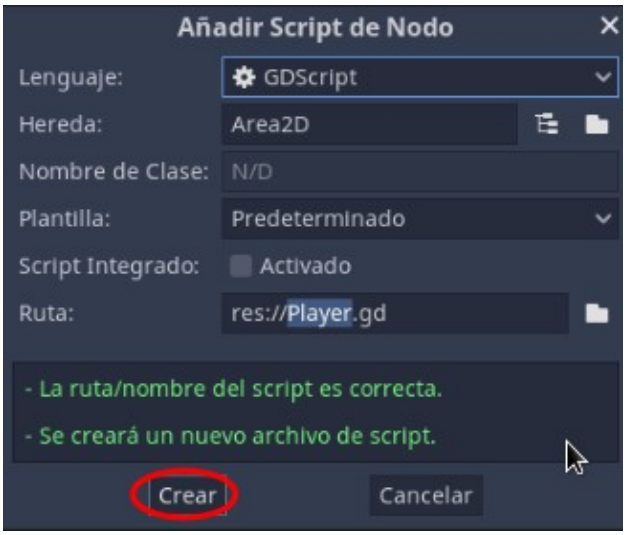
Una vez seleccionado el nodo, clickeamos en el botón de script, ubicado en la interfaz de escena a la derecha arriba.



Con esto, se nos desplegará un menú que ya vimos durante el taller de la calculadora, en donde nos preguntará información como:

- Lenguaje con el que vamos a programar.
- Cual es el tipo de nodo del cual vamos a tomar los métodos y propiedades.
- Nos preguntará sobre que plantilla queremos utilizar.
- Finalmente el nombre que le deseamos poner.

Si la información mas abajo está en verde, significa que está todo correcto. Si algún campo está mal, la herramienta nos lo marcará en rojo.



Dejaremos las opciones por defecto y presionaremos Crear.

Nos mostrará la ventana de Script y un template de código. Vamos a limpiar todo e ir agregando las siguientes lineas

Definiendo variables y nodos

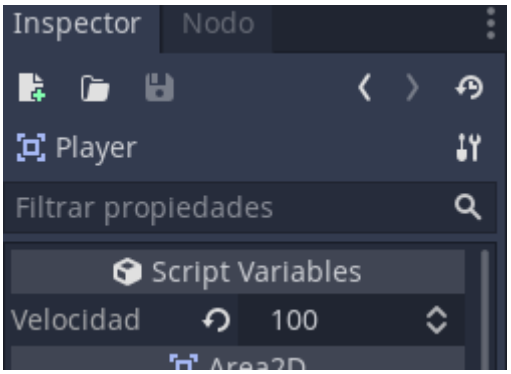
```
extends Area2D
```

La línea de código "extends Area2D" significa que el script está creando un nuevo tipo de nodo que se comporta como un área de detección de colisiones en el juego. Al extender el nodo "Area2D", el script hereda todas las características y funcionalidades que ofrece este tipo de nodo.

```
export (int) var Velocidad
```

Esta línea crea una variable exportada llamada "Velocidad". Al ser exportada, la velocidad será visible y ajustable desde el inspector de nodos en el editor de Godot. Permite controlar la velocidad de movimiento del área en el juego.

Si ahora tocamos en el nodo Player, veremos en el menú contextual de la derecha la variable velocidad. Le podemos poner un valor, por ejemplo 100. Que será la velocidad con la que se mueva nuestro personaje dentro del Area.




```
var Movimiento = Vector2()
var limite
```

Se declara la variable "Movimiento" como un Vector2 (un tipo de dato que almacena una dirección y magnitud en 2D). Representa la dirección y magnitud del movimiento del área.

Se declara la variable "limite" para almacenar el tamaño de la ventana del juego.

Definiendo el limite del Area

```
func _ready():
	#Defino limite de la pantalla
	limite = get_viewport_rect().size
```

Esta es una función especial llamada "_ready()". Se llama automáticamente cuando el nodo (objeto) al que está vinculado este script se añade al árbol de escenas y está listo para ser utilizado.

Se obtiene el tamaño de la ventana del juego (el viewport) y se almacena en la variable "limite". Esto permite definir el límite máximo de movimiento del área.

Ahora vamos a crear una función llamada: **_process(delta)**. Esta función se llama en cada fotograma del juego y se utiliza para realizar actualizaciones de lógica y movimiento.

A continuación agregaremos el siguiente código:

```
func _process(delta):
	Movimiento = Vector2()
	if Input.is_action_pressed("ui_right"):
		Movimiento.x += 1
	if Input.is_action_pressed("ui_left"):
		Movimiento.x -= 1
	if Input.is_action_pressed("ui_down"):
		Movimiento.y += 1
	if Input.is_action_pressed("ui_up"):
		Movimiento.y -= 1
```

```
    if Movimiento.length() > 0:
          Movimiento = Movimiento.normalized() * Velocidad
    position += Movimiento * delta
```

Movimiento = Vector2(): Al inicio del proceso, se reinicia la variable "Movimiento" a un vector nulo.

Movimiento del área: Dependiendo de las teclas presionadas por el jugador, se asignan valores a la componente "x" y "y" del vector "Movimiento" para indicar la dirección del movimiento. Nota como tenemos cuatro condiciones, una para cada tecla de dirección (derecha, izquierda, abajo y arriba), las cuales le permitirán al Player movilizarse al presionarlas, cambiando su posición aumentando o disminuyendo en 1 su valor, siguiendo algo muy similar a un plano cartesiano en donde X es el eje horizontal y Y el eje vertical (con la diferencia de que en esta programación, el movimiento positivo en el eje Y será como bajar y el movimiento negativo será como subir, contrario a lo aprendido tradicionalmente de un plano cartesiano)

Movimiento = Movimiento.normalized() * Velocidad: Si el usuario presiona al mismo tiempo, por ejemplo, la tecla de arriba y derecha, el Player se movería al doble de velocidad, ya que se estarían registrando dos entradas a la vez. Pero, este problema podemos solucionarlo agregando este código, una condición que verifique si el personaje está en movimiento y normalice dicha velocidad). Si el vector "Movimiento" tiene una magnitud mayor que cero (es decir, si el área se está moviendo), se normaliza para mantener la misma dirección pero con magnitud 1 y luego se multiplica por la velocidad definida en "Velocidad".

Configurando la variable Position para los movimientos

- **position += Movimiento * delta:** La posición del área se actualiza sumándole el vector "Movimiento" multiplicado por "delta[1]". El "delta" representa el tiempo transcurrido desde el último fotograma y se utiliza para asegurar que el movimiento sea independiente de la velocidad del hardware o de la cantidad de fotogramas por segundo.

Limitar por la ventana

```
    #Position va  estar atrapdo en el limite de la ventana
    position.x = clamp(position.x, 0, limite.x)
    position.y = clamp(position.y, 0, limite.y)
```

1 NOTA: Delta es la velocidad que el juego llevará en cada segundo y se utiliza para que funcione consistente (con el mismo tiempo), igual en cualquier computadora sin importar si es rápida o lenta.

Esta parte del código utiliza la función "clamp" para asegurarse de que la posición del área quede atrapada dentro de los límites de la ventana del juego. Es decir, si el área intenta salir de la ventana, esta función se asegura de que permanezca dentro de los límites.

Función "**clamp**":

Toma tres argumentos:

1. El primer argumento es el valor que queremos asegurarnos de que esté dentro de los límites.
2. El segundo argumento es el valor mínimo o límite inferior al que queremos restringir el valor.
3. El tercer argumento es el valor máximo o límite superior al que queremos restringir el valor.

En este caso, el código realiza lo siguiente para cada eje (x e y) de la posición del área:

1. **position.x = clamp(position.x, 0, limite.x):** Se asegura de que la posición en el eje x **(position.x)** quede atrapada entre el valor 0 (límite inferior) y el valor máximo **limite.x** (límite superior). Esto garantiza que el área no pueda moverse más allá del borde izquierdo (0) o derecho **(limite.x)** de la ventana.
2. **position.y = clamp(position.y, 0, limite.y)**: De manera similar, se asegura de que la posición en el eje y **(position.y)** quede atrapada entre el valor 0 (límite inferior) y el valor máximo **limite.y** (límite superior). Esto garantiza que el área no pueda moverse más allá del borde superior (0) o inferior **(limite.y)** de la ventana.

Mejorando los movimientos (acoplando animaciones)

```
#vamos a acoplar las animaciones
      if Movimiento.x != 0:
            $Sprite_player.animation = "Lado"
            $Sprite_player.flip_h = Movimiento.x < 0
      elif Movimiento.y < 0:
            $Sprite_player.animation = "Atras"
      elif Movimiento.y > 0:
            $Sprite_player.animation = "Frente"
      else:
            $Sprite_player.animation = "Quieto"
```

En el código anterior, creamos condiciones anidadas que definirán cuál es la animación que se reproduce al momento de presionar determinada tecla (además de hacer flip horizontal al momento de moverse en el eje X, para ahorrar sprites). Ajusta la animación del nodo Sprite del personaje

según la dirección del movimiento del área. Si el área se mueve hacia la izquierda o hacia la derecha, se muestra la animación "Lado" y se voltea horizontalmente si se mueve hacia la izquierda. Si el área se mueve hacia arriba, se muestra la animación "Atras"; si se mueve hacia abajo, se muestra la animación "Frente"; y si el área está quieta, se muestra la animación "Quieto". Esto permite que el personaje muestre animaciones adecuadas según su movimiento en el juego.

Guarda todo y prueba nuevamente la escena (F6).