FORO NACIONAL DE INGENIERÍA UVM 2024

RETROGAME DE 8 BITS:

¡ALCÁNZALOS, HUESITOS!

CREADO POR:

Cervantes Gutiérrez Arturo Gabriel

Dávila Vázquez Lilia Charlotte

Flores Gómez Mónica Nayely

Contenido del juego

Historia

Huesitos es un gatito blanco y delgado que perdió el rumbo de su hogar. Se encuentra atrapado en un espacio desconocido, rodeado por paredes, plataformas flotantes, monedas y *slimes* nada amigables. Huesitos deberá escalar el lugar usando las plataformas móviles, y tendrá que evitar los peligros para alcanzar el único alimento que encontró en ese extraño mundo: un pescado.

Controles

- Moverse lateralmente: Flecha derecha e izquierda, o A y D
- Saltar: Espacio, flecha arriba, W.
- Menú de pausa y selección: Enter

Mecánicas

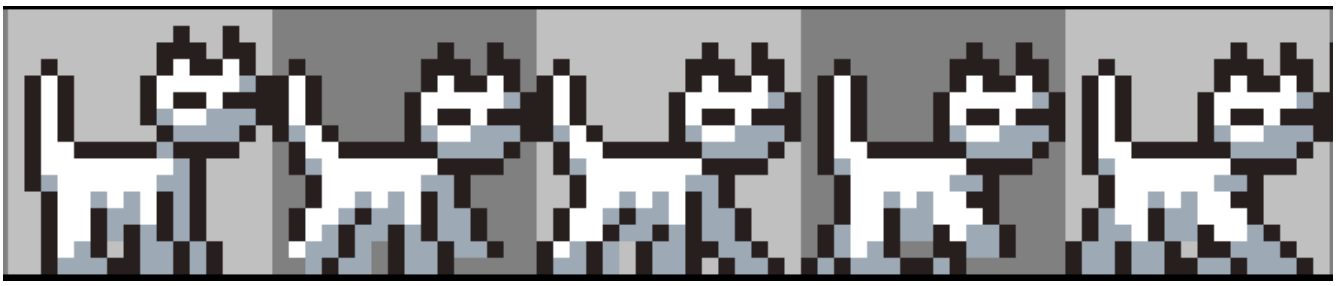
- Moverse
- Saltar

Diseño de personajes y assets

El juego se desarrolló principalmente utilizando herramientas de software libre. Todos los *sprites* y *tiles* para los personajes y escenarios se realizaron con Aseprite y Librespite. Se utilizaron tamaños de 16x16 píxeles.

Para crear animación y movimiento se utilizan conjuntos de sprites llamados *spritesheets*. En un *spritesheet* se encuentran todas las imágenes usadas para representar un elemento dentro de un solo archivo, que luego son acomodadas y asociadas a un evento o momento durante el juego.

Por ejemplo, cuando Huesitos corre, se le asigna el ciclo compuesto de los *sprites* que representan su movimiento.

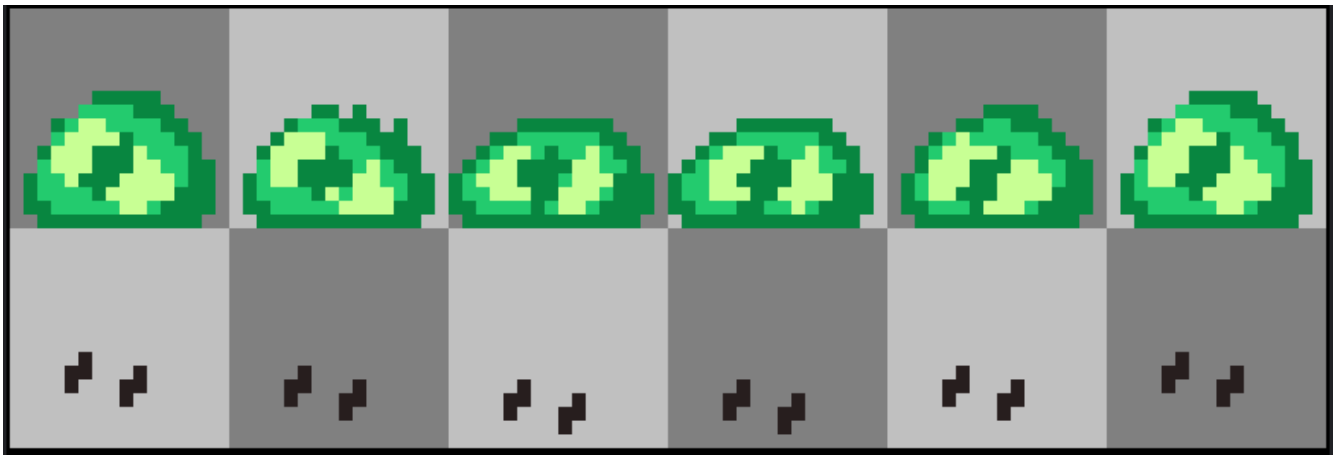


Spritesheet del ciclo de movimiento de Huesitos

Cabe recalcar que los sprites de la NES sólo permiten 4 colores: 3 a elegir, y el cuarto es la transparencia, que "rellena" el resto del sprite sin colorear.

Si queremos un modelo más complejo, podemos colocar dos *sprites* juntos, como es el caso del enemigo "Mucu", que se compone de dos partes: el cuerpo principal y los ojos. Esto permite más detalle y complejidad en un modelo, pudiendo superar un poco las limitaciones

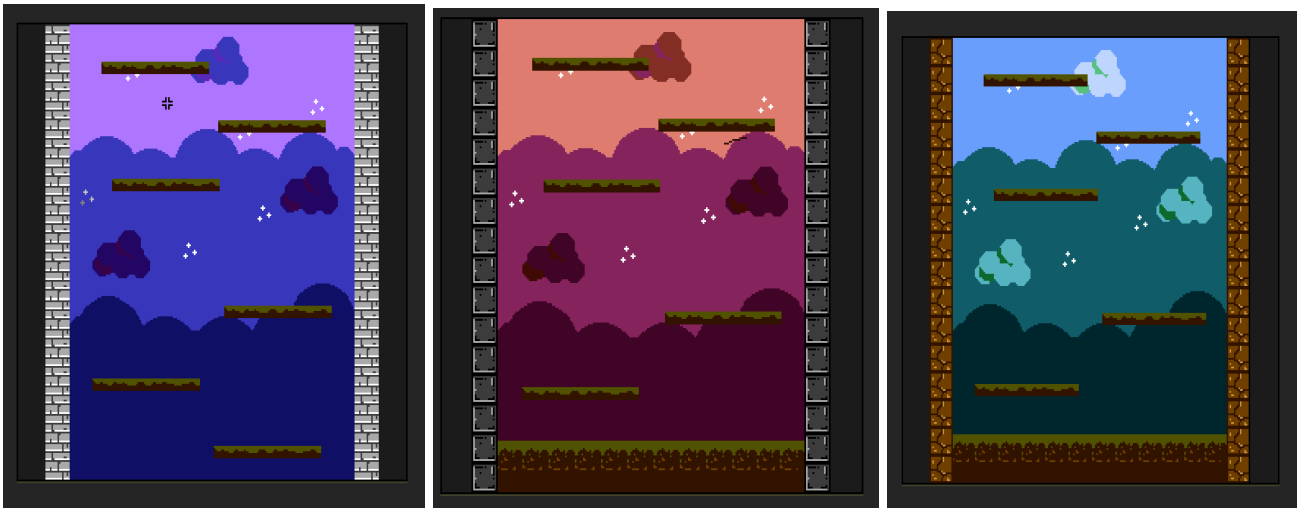
técnicas.



En la parte superior, el cuerpo principal del enemigo "Mucu", usando tres colores. En la parte inferior, los ojos de "Mucu"

Creación de los tiles

Los tiles permiten diseñar libremente los mapas en los videojuegos. Antes de crear los tiles se dibujaron bocetos en LibreSprite para probar distintas paletas de colores, formas y composiciones.



Una vez satisfechos con los bocetos, se creó el tilemap

Música y SFX

La música y los efectos de sonido se realizaron con LMMS, utilizando un plugin llamado 'Nescaline', que es capaz de replicar el sonido de los 4 canales de audio de la NES junto con algunos efectos que la consola era capaz de reproducir, tales como el *sweep*, distintas duraciones de *envelope* y posibilidades de transposición de tonos.

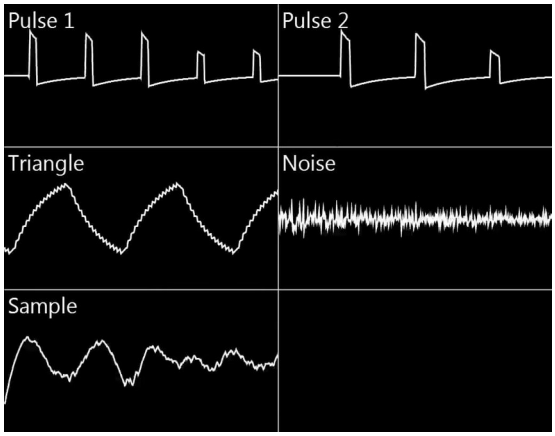
Nescaline, con las configuraciones para cada canal de la NES

Mezcla de pistas de Nescaline para la creación de la melodía completa en LMMS

La NES cuenta con cuatro canales de audio disponibles, los cuales pueden producir únicamente un sonido a la vez (explod2A03, 2012):

- Canales 1 y 2: Los primeros dos canales generan una onda de forma cuadrada, llamada "*pulse wave*", los cuales tienen cuatro "voces", que son ciclos de trabajo de 12.5%, 25%, 50% y 75%. Una *voz* de 50% da una onda totalmente cuadrada, mientras que las *voces* de 25% y 75% son exactamente iguales. Es posible aplicar efectos de *sweep*, *envelop* y *loop*, y también se puede alterar el volumen de cada canal. Estos dos canales se usan principalmente para la melodía y armonía.
- Canal 3: El tercer canal genera una onda de forma triangular. Sólo puede hacer una voz, y no tiene control de volumen; una nota o está encendida o está apagada. Se utiliza principalmente para las líneas de bajo.
- Canal 4: Es el *canal de ruido*, generando un sonido similar al de estática en una televisión antigua. Tiene un total de 32 voces, y tampoco tiene control de volumen. Se utiliza principalmente para la percusión, aunque en ocasiones puede formar parte de la melodía.

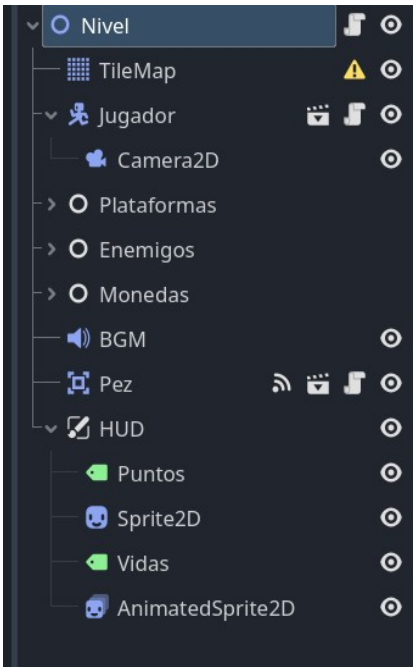
Adicionalmente, la NES cuenta con un quinto canal que se utiliza para *samples*. En nuestro caso no utilizamos dicho canal, ya que nos es innecesario.



Representación de los canales de la NES

Programación del juego

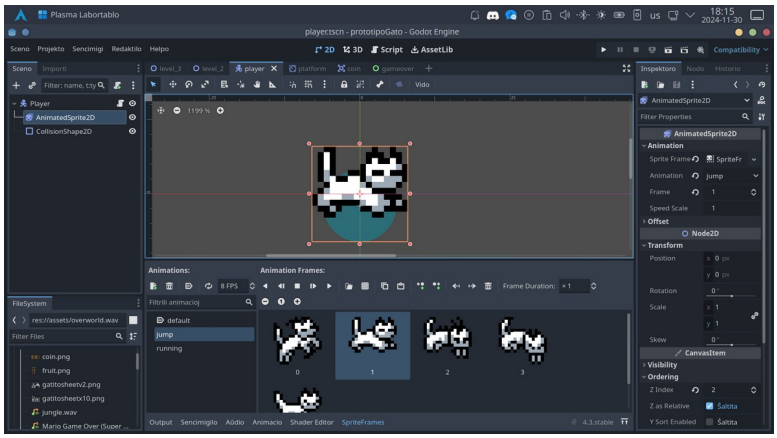
Godot proporciona una interfaz muy sencilla para la implementación del funcionamiento del juego. En Godot cada nodo es una 'escena', que puede contener a otras *escenas* y que se puede reutilizar en cualquier momento.



Árbol de nodos para cada nivel.

El jugador, los coleccionables y los enemigos consisten de un *sprite* animado en 2D y una

CollisionShape, que es el objeto encargado de detectar colisiones.



Objeto del jugador, con su *sprite* y su *CollisionShape*

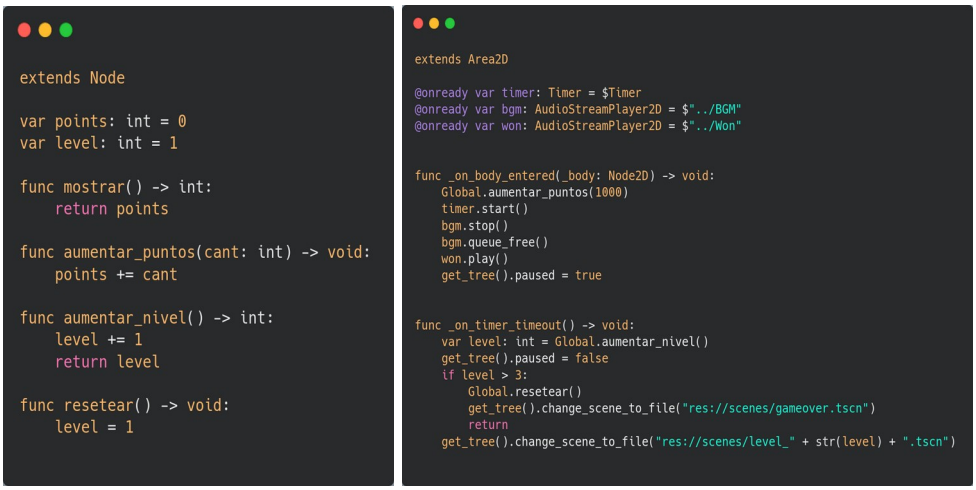
Mediante programación en GDScript se pueden especificar las acciones que se llevarán a cabo cuando se detecte una colisión, además de que se puede implementar el movimiento del personaje con sus animaciones. También se pueden realizar cosas como pausar o reproducir música, modificar las físicas o pausar la ejecución del juego.

```gdscript
func _physics_process(delta: float) -> void:

    if not is_on_floor():
        if animated_sprite.animation != "jump":
            animated_sprite.play("jump")
        velocity += get_gravity() * delta

    if Input.is_action_just_pressed("space") and is_on_floor():
        velocity.y = JUMP_VELOCITY

    var direction := Input.get_axis("ui_left", "ui_right")

    if direction == 0 and is_on_floor():
        animated_sprite.play("default")
    if direction != 0 and is_on_floor():
        animated_sprite.play("running")

    if direction > 0:
        animated_sprite.flip_h = false
    elif direction < 0:
        animated_sprite.flip_h = true

    if direction:
        velocity.x = direction * SPEED

    else:
        velocity.x = move_toward(velocity.x, 0, SPEED)

    move_and_slide()
```

Función de movimiento del personaje

También es posible utilizar variables y métodos globales, que son particularmente útiles para

cosas como la puntuación, la cantidad de vidas, el nivel en el que se encuentra el jugador, etcétera. Estos objetos globales pueden accederse desde cualquier otra parte, y se pueden modificar.



```
extends Node

var points: int = 0
var level: int = 1

func mostrar() -> int:
    return points

func aumentar_puntos(cant: int) -> void:
    points += cant

func aumentar_nivel() -> int:
    level += 1
    return level

func resetear() -> void:
    level = 1
```

```
extends Area2D

@onready var timer: Timer = $Timer
@onready var bgm: AudioStreamPlayer2D = $"../BGM"
@onready var won: AudioStreamPlayer2D = $"../Won"


func _on_body_entered(_body: Node2D) -> void:
    Global.aumentar_puntos(1000)
    timer.start()
    bgm.stop()
    bgm.queue_free()
    won.play()
    get_tree().paused = true


func _on_timer_timeout() -> void:
    var level: int = Global.aumentar_nivel()
    get_tree().paused = false
    if level > 3:
        Global.resetear()
        get_tree().change_scene_to_file("res://scenes/gameover.tscn")
        return
    get_tree().change_scene_to_file("res://scenes/level_" + str(level) + ".tscn")
```

Ejemplo de variables globales, y su acceso desde otra parte del código

Para el resto de cosas se puede usar el editor visual de Godot, que no requiere programación. En el documento que se nos proporcionó se solicitó que se utilizara una resolución de 256x240 píxeles, por lo que ese es el tamaño utilizado.

Referencias

Brackeys. (2024, 28 abril). How to make a Video Game - Godot Beginner Tutorial [Video]. YouTube. https://www.youtube.com/watch?v=LOhfqjmasi0

explod2A03. (2012, 6 marzo). NES Audio: Brief Explanation of Sound Channels [Video]. YouTube. https://www.youtube.com/watch?v=la3coK5pq5w

Gwizz. (2023, 25 marzo). Godot 4 Main Menu Beginner Tutorial [Video]. YouTube. https://www.youtube.com/watch?v=vsKxB66_ngw