Collect Carrot

Proyecto SMX2B

Presentado por Mario Alcaraz Corbacho y Jon Bornás Cunillera

Índice

01

Portada

Qué es el proyecto y primer impacto

Portada

Collect Carrot: Videojuego de plataformas interactivo en 2D.
Inspiración en mecánicas clásicas de movimiento y recolección.
Progresión visual dividida en dos entornos diferenciados (Bosque y Cueva).

02

Introducción

El motor de juego y el lenguaje de programación que usamos.

Introducción

El Reto Técnico: Unificar sistemas aislados en un entorno de ejecución simultánea en tiempo real.
Arquitectura Coherente: Integración síncrona de físicas del personaje, rutas de IA y persistencia de datos.
Metodología de Equipo: Control de versiones profesional mediante repositorio remoto en GitHub.

Objetivos

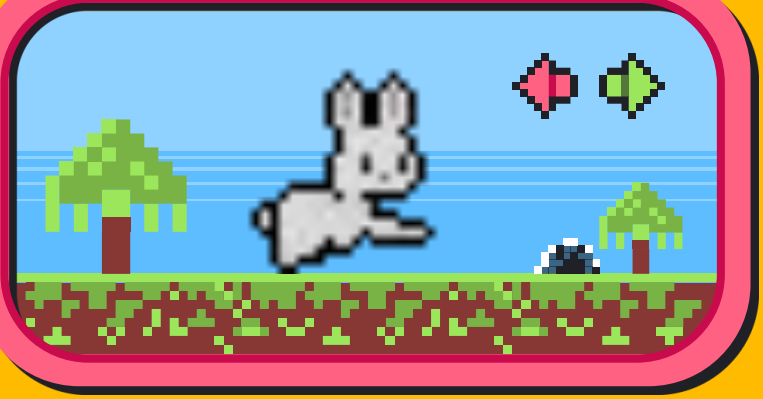
Lo que queríamos conseguir para que el juego funcionara al 100%.

Objetivos

Meta Principal: Entrega de un software finalizado, pulido y jugable de 10 niveles sin excepciones lógicas.
Indicadores Técnicos de Rendimiento:

- *Interactividad:* Latencia mínima y respuesta fluida en los controles del personaje.
- *Autonomía:* Patrullas enemigas automatizadas confinadas a sus límites físicos.
- *Persistencia:* Retención e integridad de datos globales (vidas/zanahorias) entre escenas.

04



Tecnología y Metodología

Motor de desarrollo Godot 4 y herramientas de software.

Tecnología y Metodología

Entorno de Desarrollo: Selección de **Godot Engine 4** frente a Unity/Unreal por optimización de hardware.
Lenguaje Nativo: Implementación de lógica orientada a objetos mediante **GDScript**.
Metodología Modular: Arquitectura basada en scripts independientes (Jugador / Enemigos / Coleccionables).
Beneficios: Escalabilidad del sistema, limpieza de código y mantenimiento eficiente.

GDScript

05

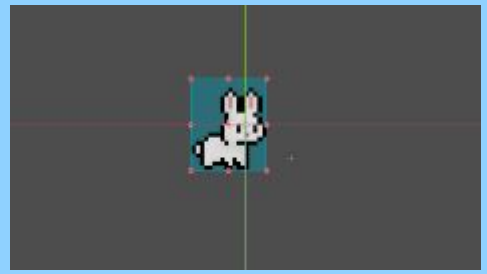
Arquitectura

Organización por fases y flujo de trabajo del equipo.

Conejos (Fisicas, animaciones, audios)

Nodo Raíz: CharacterBody2D (Esqueleto físico y control cinemático).
Bucle de Físicas: Función nativa _physics_process(delta).
Detección de Suelo: Uso de is_on_floor() para el cálculo de gravedad.
Caja de Colisión: CollisionShape2D en forma de cápsula (Optimización de salto).

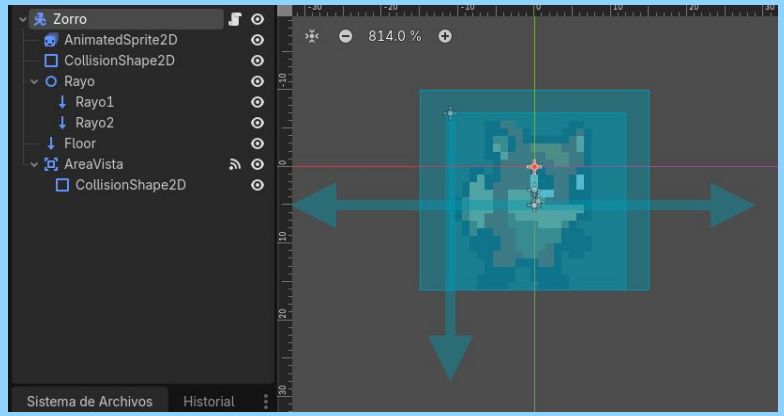
```
13
14  func _physics_process(delta: float) -> void:
15      if esta_muerto:
16          velocity.x = 0
17          if not is_on_floor():
18              velocity += get_gravity() * delta
19          move_and_slide()
20          return
21
22      if not is_on_floor():
23          velocity += get_gravity() * delta
```



Enemigos, IA y Música

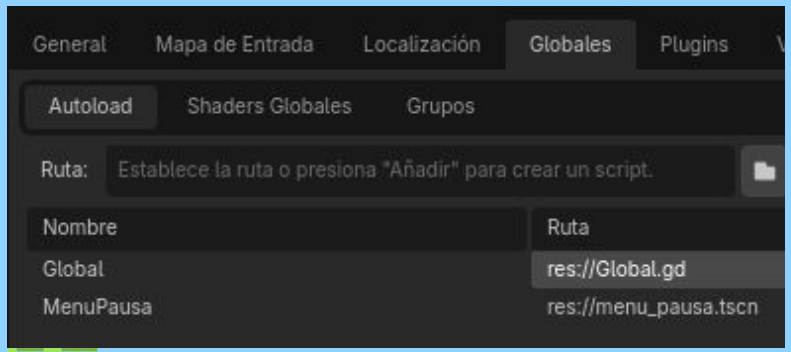
Inteligencia de Patrulla: Comportamiento automático para vigilar plataformas y rebotar al llegar a los extremos.
Sensor Dinámico (Floor): Nodo RayCast2D que actúa como un láser invisible hacia el suelo y se recoloca a izquierda (-8) o derecha (8) según la marcha del zorro.
Detección de Vacío (!ray_suelo.is_colliding()): Condición lógica en el script que avisa al instante cuando el nodo Floor deja de tocar suelo firme.
Inversión de Marcha (velocity.x *= -1): Multiplicación que cambia el signo de la velocidad para dar la vuelta inmediatamente al tocar paredes o detectar abismos.

```
if is_on_wall() or !ray_suelo.is_colliding():
    velocity.x *= -1
```



Vidas, Zanahoriass y Datoss Globales

Persistencia de Datos: Script permanente (Global.gd / Autoload) para evitar el reseteo de marcadores al cambiar de nivel.
Zona de Detección: Nodo Area2D invisible para registrar la recolección sin frenar el movimiento del protagonista.
Efecto de Recolección: Sumador automático de puntos en el almacén global tras interactuar con el objeto.
Optimización del Sistema: Comando queue_free() para eliminar el objeto recogido y liberar recursos de forma limpia.

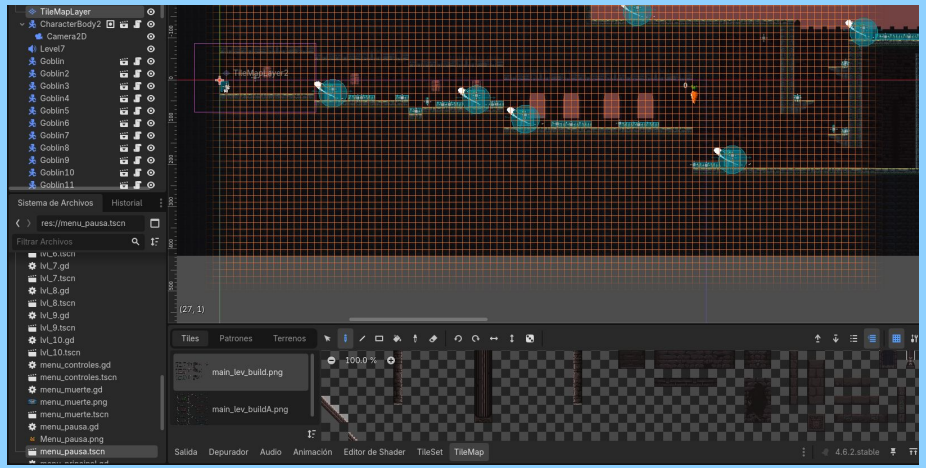


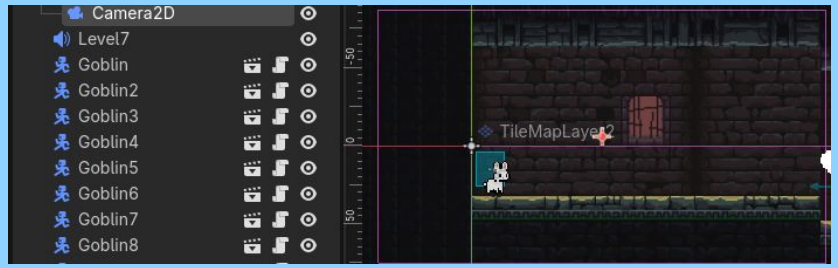
Menús de Pausa, Victoria y Muerte

Tránsito entre Escenas: Método change_scene_to_file() para conectar de forma secuencial los niveles del 1 al 10.
Menú de Pausa: Congelación absoluta de físicas mediante get_tree().paused con tres botones interactivos de madera.
Pantalla de Victoria: Activación lógica al pisar la meta final, desplegando un mensaje de éxito y animaciones de celebración.
Menú de Muerte: Sistema de reintento enlazado al script global para restaurar la salud y recargar la escena desde cero.

Diseño del Mundo

Construcción por Matriz (TileMapLayer): Edición sistemática de niveles que optimiza los procesos del juego.
Seguimiento Óptico (Camera2D): Nodo de encuadre centrado con desplazamiento suavizado del protagonista.
Profundidad Visual (ParallaxBackground): Sistema de capas infinitas con movimiento asíncrono para generar realismo.





06

Evaluación

Resultado del juego y lo que hemos aprendido programando

Evaluación

Producto Terminado: Videojuego de plataformas robusto, divertido y 100% estable en sus 10 niveles consecutivos.
Cero Fallos Lógicos: Funcionamiento totalmente fluido, garantizado y libre de cuelgues o bloqueos de código.
Arquitectura Modular: Scripts completamente desacoplados por componentes independientes para facilitar la limpieza y el mantenimiento.
Escalabilidad Garantizada: Estructura base altamente eficiente que permite añadir decenas de mapas nuevos sin romper el núcleo.
Persistencia Externa (Mejora): Futuro sistema de archivos locales para guardar el progreso del juego al cerrar la aplicación.
IA Avanzada (Mejora): Futuras mecánicas de persecución activa por rango de visión para los enemigos.

Muchas Gracias

https://projectes.elpuig.xeill.net/projecte/43