Proyecto - Videojuego en Godot

Diplomatura en Desarrollo de Videojuegos - UNQui

Objetivo del Proyecto

El objetivo de este proyecto es desarrollar un videojuego funcional en Godot, reutilizando como base el código del “juego espejo” trabajado en clase. Los alumnos deberán implementar una versión alternativa del juego, respetando sus mecánicas centrales y estructura, pero adaptando la lógica de eje y vista.

El proyecto será además una instancia de evaluación conjunta entre la materia “Elementos Básicos de Programación de Videojuegos” e “Introducción a los Videojuegos”, ya que los alumnos deberán diseñar y aplicar sus propios recursos visuales originales para el juego.

Descripción General

Los estudiantes utilizarán como punto de partida el proyecto trabajado en clase (versión Frogger de gatito), replicando su funcionamiento básico. Sin embargo, esta versión del alumno requiere un cambio fundamental en la forma de presentar y gestionar la lógica del juego: deberán invertir el eje principal de interacción, modificando tanto el movimiento como la disposición visual.

Este desafío deberá ser abordado por cada alumno sin instrucciones explícitas en este documento. La lógica requerida se puede deducir a partir del código del “juego espejo” y su análisis.

Requisitos Técnicos y Funcionales

Resolución y Sistema de Celdas

- El proyecto base está diseñado con una resolución de 208 x 288 píxeles (vertical), utilizando assets de 16x16 píxeles.

- Esta estructura responde a una lógica de movimiento por celdas de 16 píxeles, similar al Frogger clásico.

- El movimiento del jugador y la colocación de obstáculos están sincronizados con esta grilla.

Importante:

A los alumnos se le dará la libertad de modificar el tamaño de los assets y la resolución del juego si lo desean (por ejemplo, trabajar con 32x32 o con resoluciones diferentes), pero deben tener en cuenta que esto puede implicar reajustar completamente:

- La lógica del movimiento.

- Las posiciones de los objetos en pantalla.

- La resolución del mapa y HUD.

Se recomienda mantener la resolución original y el sistema de celdas 16x16, especialmente si no se tiene experiencia previa modificando este tipo de sistemas.

Funcionalidad General

- El juego debe ser completamente jugable desde inicio a fin.

- Deben respetarse las mecánicas base del juego original (movimiento, objetivos, obstáculos, condiciones de victoria/derrota).

- El personaje debe poder llegar a una zona objetivo (casas) evitando obstáculos.

Adaptación del Eje

- El juego debe adaptar su lógica de movimiento y presentación del eje Y (vertical) al eje X (horizontal).

- La relación de aspecto debe pasar de **208x288 (alto)** a **288x208 (ancho)**.

- Los movimientos de enemigos/obstáculos deben ahora producirse de forma horizontal.

Buenas Prácticas de Programación

- Uso correcto de nombres de archivos y nodos.
- Estructura clara de scripts: sin lógica duplicada ni funciones sin uso.
- Separación adecuada de responsabilidades entre scripts.
- Comentarios claros cuando sea necesario para explicar funciones no evidentes.
- Evitar código hardcodeado innecesariamente.

Requisitos de Arte y Assets

Todos los recursos gráficos del proyecto deben ser originales y creados por los alumnos, se podrán utilizar recursos como referencias o adaptaciones. No se aceptarán assets descargados de bancos externos. En caso de utilizar assets externos estos tendrán que verse afectados por un proceso de modificación sustancial por parte del alumnado.

Assets mínimos obligatorios:

1. **Sprites para el Personaje Principal**
    - Sprite sheet de **3 estados distintos**:
        - Estado idle (quieto).
        - Estado en movimiento.
        - Estado golpeado/perdiendo vida.
2. **Asset de vidas**
    - Debe representar visualmente una vida (ej: corazón, ícono, símbolo personalizado).
3. **Asset de casa (meta)**
    - Imagen que represente la casa a la que debe llegar el jugador.

4. **Asset de casa ocupada**
    - Puede ser una versión modificada del asset de casa original o un asset completamente distinto que indique que ya fue ocupada.
5. **Asset de obstáculo / auto**
    - Puede ser un solo sprite o varios, dependiendo del diseño del juego.
6. **Tilemap completo para el mapa**
    - Debe incluir los distintos tiles del mapa: caminos, fondo, terreno, etc.
    - Se pueden incluir aquí también los assets de casa, casa ocupada u obstáculo si no cuentan con animación.

Sugerencia: mantener una coherencia visual entre los diferentes elementos del juego, y consideren el uso de una paleta base limitada para facilitar el diseño pixel-art.

Estructura Esperada del Proyecto

Scripts:

- `jugador.gd`: Movimiento y control del personaje.
- `hud.gd`: Interfaz gráfica y control de puntajes.
- `game_state_manager.gd`: Control de estados del juego.
- `casas_manager.gd` y `casa.gd`: Lógica de objetivos.
- `linea_obstaculo.gd` y `spawner_obstaculos.gd`: Generación y control de obstáculos.

Escenas:

- `main.tscn`: Escena raíz del juego.
- `jugador.tscn`: Nodo con sprite y comportamiento del personaje.
- `hud.tscn`: Interfaz gráfica.
- `mapa.tscn`, `casa.tscn`, `obstáculo.tscn`, etc.

Entregables

- Carpeta del proyecto completa en formato `.zip`.
- Debe incluir todos los assets y scripts necesarios para su funcionamiento.
- Debe abrirse correctamente en Godot sin errores.
- Archivo README.txt/.docx/.pdf con:
    - Nombre/apellido del alumno.
    - Créditos de autores de los assets (si corresponde), aunque deben ser originales.
    - De ser necesario, cualquier decisión técnica destacable.

Fechas de Entregas

Primera entrega intermedia/check: 26 de Mayo

Segunda entrega intermedia/check: 2 de Junio

Entrega final: 9 de Junio

Recuperatorio: 16 de Junio