Documentación de Diseño del Videojuego PERSIANAS

Equipo de Desarrollo:

Jose Manuel Gallardo Del Águila

Julia Muñoz Tejera

Nikola Semerdzhiev

Alfredo García Gómez

Paula Venegas Roldán

Índice

1 Introducción

Persianas es un remake del clásico *Prince of Persia* (1989), actualizado para las plataformas modernas como PC y consolas, específicamente PlayStation. Se ha adaptado el sistema de control para ser completamente compatible con joystick, ofreciendo una experiencia intuitiva y fluida. La esencia del juego original se mantiene gracias a la recuperación de los sonidos clásicos, mientras que la experiencia visual se renueva con gráficos modernos, mejorados y estilizados para atraer a las nuevas generaciones. Esta documentación recoge el diseño, desarrollo y planificación del proyecto, reflejando nuestra intención de combinar nostalgia con innovación.

2 Enlace al proyecto en Github

Puedes acceder al repositorio del proyecto en el siguiente enlace: Repositorio en GitHub

3 Descripción General del Juego

Persianas es un juego de plataformas en 2D que combina acción, exploración y combate dinámico. Se ha implementado un mapa de controles que conserva las funcionalidades del original, adaptado a la ergonomía y características de los mandos modernos. El juego incluye varios mapas con temáticas distintas que enriquecen la variedad visual y jugabilidad, desde mazmorras hasta ambientes naturales. El personaje principal tiene un diseño personalizado, creado íntegramente por nuestro equipo, que le confiere una identidad única dentro del género. Aunque mantiene la atmósfera y sonidos del juego original, se ha modernizado para ofrecer una experiencia más atractiva y actual.

4 Público Objetivo

El juego está pensado para:

- Fans nostálgicos de la saga *Prince of Persia* que buscan revivir los clásicos con una experiencia renovada y mejoras técnicas.
- Nuevos jugadores interesados en plataformas con ambientaciones variadas y estética moderna.

La dificultad aumenta progresivamente conforme se superan los niveles.

5 Inspiración y Referencias

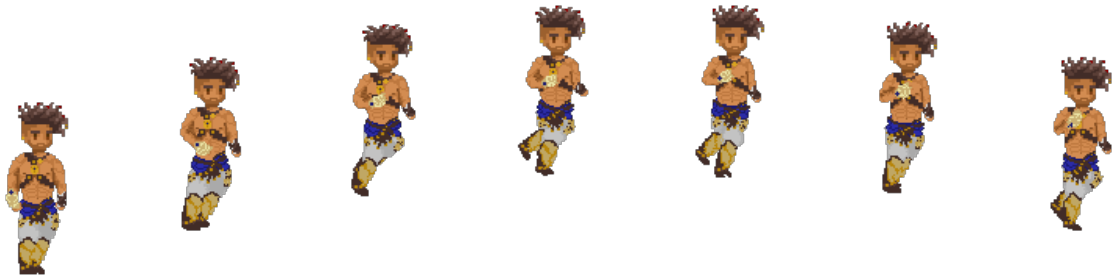
Nuestra principal inspiración es el *Prince of Persia* original (1989), tanto por su jugabilidad como por su atmósfera distintiva. También hemos tomado como referencia los efectos de sonido y la banda sonora de Prince of Persia: The Two Thrones y Prince of Persia: The sands of time. Además, hemos integrado nuevas temáticas para los mapas para tener una propuesta visual renovada y atractiva.

6 Estilo Visual y Arte

Nuestro juego presenta un estilo visual completamente en 2D, con gráficos diseñados manualmente que mezclan lo clásico y lo moderno. Todos los sprites han sido creados desde cero con el programa de dibujo **aseprite**, dando lugar a un entorno personalizado y distintivo.

A continuación, se muestran las animaciones del personaje principal (Figura 1):

(a) Animación correr



(b) Animación saltar

(c) Animación idle

(d) Animación ataque

(e) Animación escalar

Figura 1: Ejemplos de animaciones del personaje diseñadas desde cero para el juego.

7 Controles y Experiencia de Usuario

El juego ha sido diseñado para ofrecer una experiencia coherente tanto en PC como en consolas, adaptando los controles a cada plataforma.

En PC, el teclado y ratón son el sistema principal de control:

- **A,D:** movimiento del personaje.
- **Espacio:** salto.
- **Clic izquierdo del ratón:** ataque principal.
- **Clic derecho del ratón:** embestida temporal
- **Butón C:** crear clones temporales
- **Salto + dirección hacia plataforma:** escalada si hay un borde disponible.

Para las consolas(por ejemplo PlayStation), se implementa un mapa de controles, manteniendo las mismas funcionalidades:

- Stick izquierdo: movimiento.
- Botón Equis de salto.
- Botón Cuadrado de ataque.
- Botón Círculo para embestida temporal.
- Botón R1 para crear clonos temporales

La interfaz (HUD) cambia dinámicamente según la plataforma, garantizando accesibilidad, buena visibilidad y una experiencia de usuario fluida. Todos los iconos están optimizados para pantallas táctiles sin perder claridad en monitores de escritorio.

7.1 Mecánicas Nuevas

- **Sistema de combate:** Una embestida temporal que mueve al personaje rápidamente para evitar ataques.
- **Manipulación de Tiempo:** Habilidad de crear clonos temporales que imitan acciones realizadas en los ultimos segundos.
- Enemigos con patrones de ataque variados y animaciones avanzadas.

7.2 Progresión del Jugador

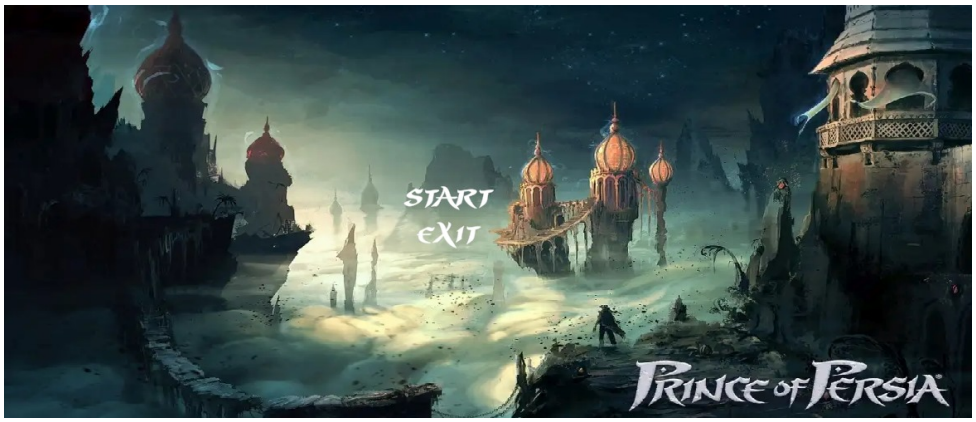
El avance del jugador se realiza a través de niveles con dificultad creciente, poniendo presión al jugador con un tiempo de límite. El jugador debe salir del nivel antes de un límite de tiempo o falla el nivel y vuelve a empezar este. A continuación se puede observar cómo maneja las vidas y dónde aparece el cronómetro en el juego:

(a) Pantalla de inicio

8 Diseño de Niveles

Los niveles que hemos diseñado varían en estilo: algunos evocan directamente el diseño clásico de los escenarios originales, mientras que otros presentan propuestas totalmente nuevas, reforzando la originalidad del proyecto.

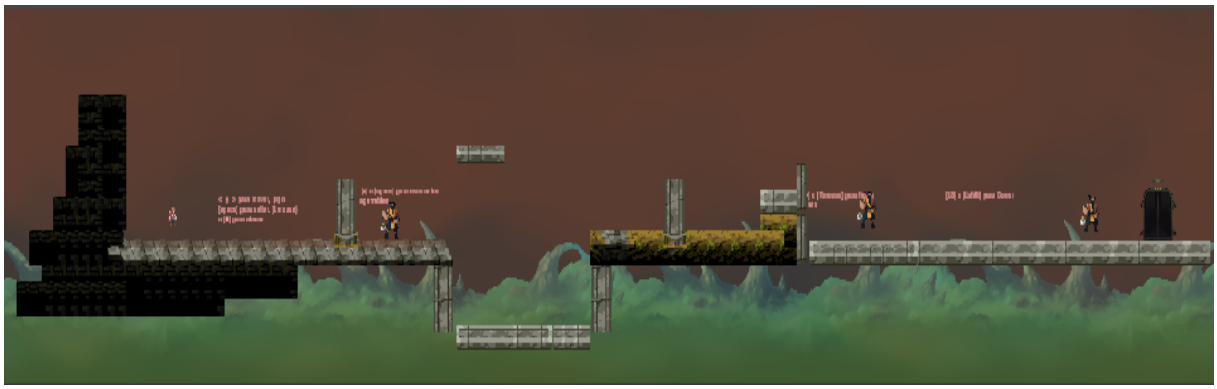
8.1 Pantalla inicial



(a) Pantalla de inicio

8.2 Nivel 0 – Tutorial

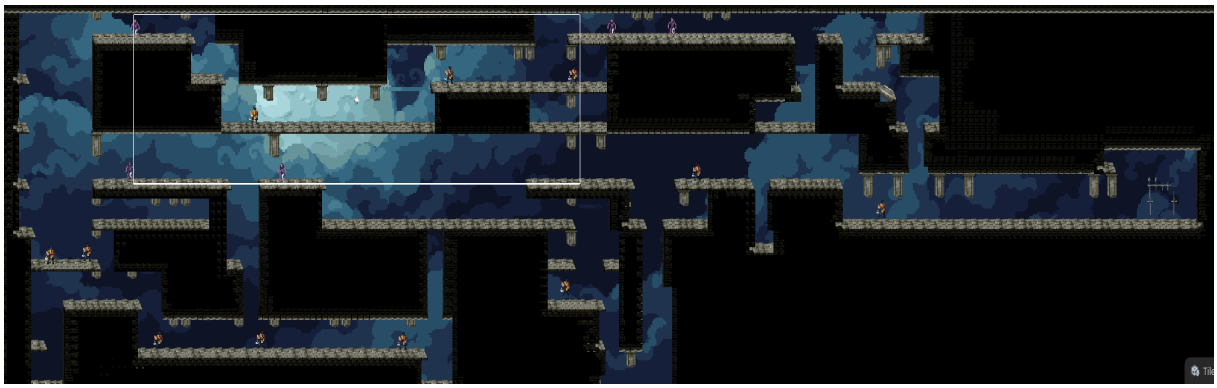
Primer nivel diseñado como tutorial, donde el jugador aprende controles básicos, mecánicas de movimiento, salto y combate contra enemigos simples.
El entorno es una dimensión con trampas sencillas y pocos enemigos, ideal para la familiarización con el joystick.



(a) Mapa del tutorial.

8.3 Nivel 1 – Las Mazmorras del Despertar

El primer nivel recrea una mazmorra siguiendo como línea el primer nivel del juego *Prince of Persia* de 1989. En este nivel el jugador aparece encerrado en una mazmorra donde tendrá que encontrar una salida en el intervalo de 3 minutos.



(a) Mapa del nivel 1.

8.4 Nivel 2 – La naturaleza engaña

Una vez nuestro protagosnista escapa de la mazmorra lo encontramos en la salida de las ruinas en un nivel inspirado en ruinas boscosas. Este nivel propone un desafío similar cronometrado por 3 minutos. En este nivel el foco se centra mas en las plataformas y menos en una estructura laberintica.

(a) Mapa del nivel 2.

8.5 Nivel 3 – La verdad

El nivel final del juego. Tras atravesar el las ruinas del bosque el protagonista se adentra en las ruinas del desierto donde se encuentra custodiada la princesa. Este nivel esta más centrado en el combate, donde toman mas protagonismo los enemigos que guardan la sala de la princesa.

(a) Mapa del nivel 3.

9 Audio y Música

El apartado sonoro de **Persianas** ha sido diseñado cuidadosamente para evocar la nostalgia del juego original mientras se adapta a las expectativas actuales de inmersión y calidad.

Hemos recuperado y remasterizado parte del audio original de *Prince of Persia* (1989), en particular los sonidos asociados a las animaciones del personaje principal —como correr y saltar. Esto refuerza el vínculo emocional con los jugadores veteranos, transportándolos al pasado a través del sonido.

Para la música ambiental, se han incorporado fragmentos de la banda sonora de *Prince of Persia: The Two Thrones* (2005), reconocida por su atmósfera envolvente. Estas piezas contribuyen a crear un entorno auditivo épico y coherente con la estética del juego.

Además, se ha llevado a cabo una implementación completa del audio para todos los movimientos del jugador, enemigos y elementos del entorno. Los efectos sonoros son dinámicos y adaptativos: responden en tiempo real a las acciones del jugador, como el uso de habilidades y los combates. Esta sonoridad reactiva incrementa la sensación de estar en un mundo vivo y mejora significativamente la inmersión.

10 Tecnología Utilizada

Para el desarrollo hemos utilizado Unity en su modo 2D, que proporciona herramientas optimizadas para trabajar con sprites, físicas y animaciones bidimensionales. Esta elección técnica permite una implementación visual precisa de los escenarios laterales y las animaciones del personaje.

- **Motor:** Unity 2D, por su versatilidad para desarrollo multiplataforma.
- **Lenguaje de scripting:** C#, facilitando desarrollo y mantenimiento.
- **Herramientas de diseño gráfico:** Photoshop y Aseprite para sprites y animaciones.
- **Control de versiones:** GitHub para gestión colaborativa del proyecto.

11 Manual de Usuario

Este manual está diseñado para guiar a los jugadores en la instalación, configuración y uso de **Persianas**, proporcionando una experiencia de juego fluida y sin inconvenientes.

11.1 Configuración Inicial

Al abrir el juego por primera vez, se recomienda realizar las siguientes configuraciones para optimizar la experiencia:

- **Controles:** Personaliza las teclas o botones táctiles según tu comodidad. Para usuarios de joystick, configura sensibilidad y asignación de botones.
- **Gráficos:** Ajusta la resolución, calidad de texturas y efectos visuales según las capacidades de tu dispositivo para equilibrar calidad y rendimiento.

11.2 Controles

PC / Consola:

- **A / D:** Movimiento lateral.
- **W / S o flechas arriba/abajo:** Interacción vertical o selección en menús.
- **Espacio:** Salto simple.
- **Clic izquierdo:** Ataque básico.
- [**Rmouse**] Embestida temporal
- ***[C]*:** crear clones
- **Tecla P o pausa:** Pausar el juego.

Consolas:

- Stick *L3* para movimiento.
- *x* de salto.
- *Cuadrado* de ataque.
- *o* Embestida
- *R1* Crear clones
- Menú accesible mediante icono en pantalla.

11.3 Interfaz de Usuario (HUD)

El HUD incluye indicadores claros para:

- Contador de vidas.
- Cronómetro

11.4 Solución de Problemas Comunes

- **Controles no responden o funcionan mal:** Revisa la configuración de entrada y conecta/desconecta dispositivos periféricos.
- **Rendimiento bajo o lag:** Reduce la calidad gráfica, cierra aplicaciones en segundo plano y actualiza drivers.
- **Errores de sonido:** Asegúrate que el dispositivo de audio esté correctamente configurado y que no haya conflictos con otras aplicaciones.
- **Problemas al guardar la partida:** Comprueba espacio en disco y permisos de escritura.

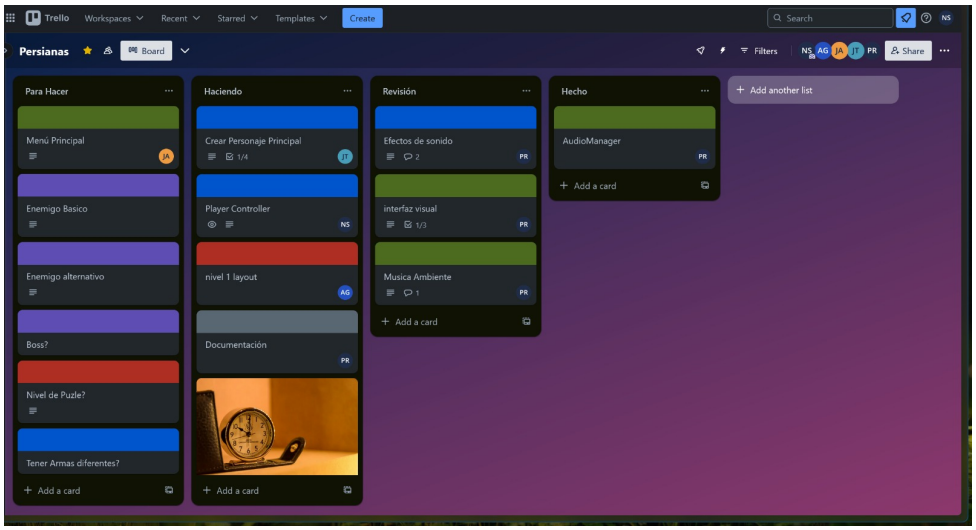
12 Planificación y Equipo

Se designó a Nikola como responsable de la planificación del proyecto, la delegación de tareas y la elaboración de un calendario para llevar a cabo el desarrollo del juego. La distribución de roles entre los miembros del equipo fue la siguiente:

- Jose Manuel Gallardo Del Águila: Desarrollo del menú y conexión entre niveles.
- Julia Muñoz Tejera: Animación y creación del personaje principal.
- Nikola Semerdzhiev: Gestión de tareas, implementación del *Player Controller*, animación del protagonista en Unity, creación e inteligencia artificial de los enemigos.
- Alfredo García Gómez: Diseño de niveles.
- Paula Venegas Roldán: Diseño de la interfaz de usuario (UI), lifemanager,documentación y apartado sonoro .

12.1 Trello

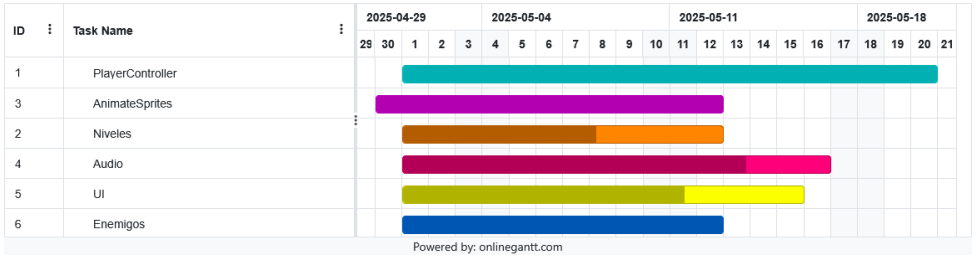
Se utilizó la plataforma Trello para representar visualmente las tareas pendientes, asignadas y finalizadas mediante tarjetas y categorías. Gracias a ello, cada integrante del equipo podía conocer en todo momento qué tareas estaban en curso, quién las estaba realizando y cuáles quedaban por hacer.



(a) Ejemplo de Trello durante el proyecto

12.2 Diagrama de Gantt

También se elaboró un diagrama de Gantt con el fin de proporcionar a cada miembro una estimación del plazo disponible para completar sus respectivas tareas.



(a) El diagrama usado

13 Desglose del desarrollo

13.1 Personaje jugable

Al tratarse de un juego en 2D, todos los elementos visuales están basados en *sprites*. El *prefab* del personaje contiene diversos scripts encargados de gestionar el movimiento, la animación y la orientación del mismo, en función de las acciones ejecutadas.

- `TouchingDirections`: garantiza una correcta interacción del personaje con el suelo, paredes, techo, etc.
- `Damageable`: script de uso general que gestiona los valores de vida y daño de las entidades vivas.
- `PlayerInput`: detecta las entradas del usuario desde el teclado o el mando mediante eventos.
- `PlayerController`: controla todos los movimientos posibles del personaje principal.

13.2 Animación del personaje en Unity

La animación del personaje se gestiona mediante una máquina de estados, cuya lógica se coordina con los parámetros activados desde el script `PlayerController`. Con vistas a futuras ampliaciones del proyecto, se implementaron submáquinas de estados que facilitan la expansión de los movimientos disponibles.

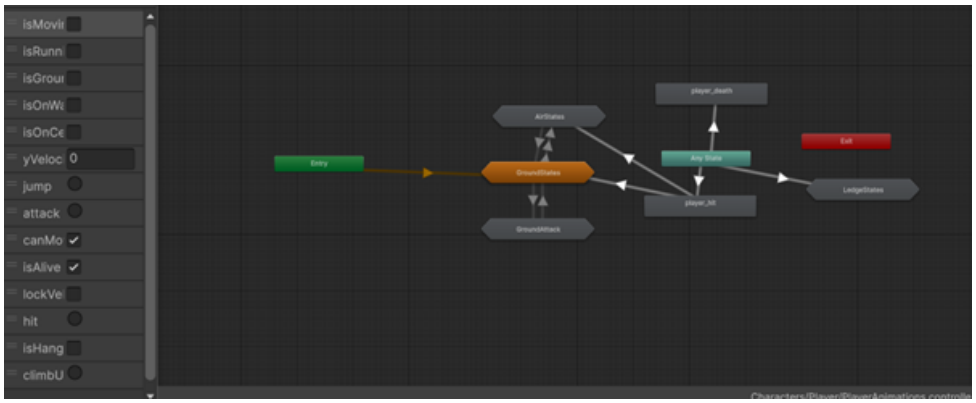
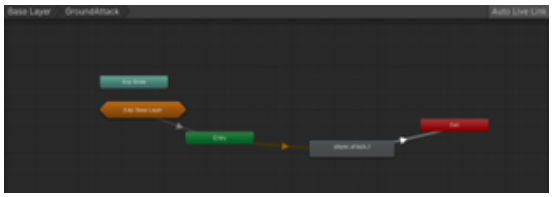
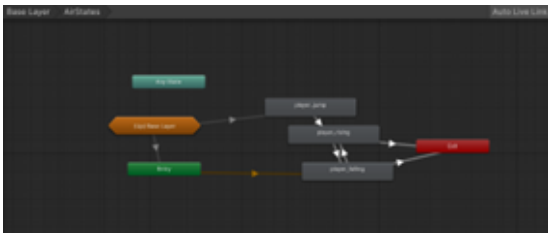


Figura 10: Máquina de estado principal



(a) Maquina de estado en el suelo



(b) Maquina de estado en el aire

13.3 Manipulación del tiempo

Embestida

La embestida es una mecánica novedosa implementada para dotar al personaje de una ventaja estratégica frente a los enemigos. Aprovechando la manipulación temporal, permite realizar un desplazamiento rápido hacia delante para esquivar ataques.

Durante la ejecución de esta acción, el tiempo se ralentiza y el personaje se lanza hacia delante, adoptando un tono azul que indica visualmente la activación del poder temporal.

```
private IEnumerator Dash()
{
    isDashing = true;
    canDash = false;
    spriteRenderer.color = new Color(0f, 0.9f, 5f);

    float originalGravity = rb.gravityScale;
    rb.gravityScale = 0;

    float dashDirection = isFacingRight ? 1f : -1f;
    rb.linearVelocity = new Vector2(dashDirection * dashSpeed, 0f);

    //slowing time
    Time.timeScale = 0.5f;
    Time.fixedDeltaTime = 0.02f * Time.timeScale;

    yield return new WaitForSeconds(dashDuration);

    rb.linearVelocity = Vector2.zero;
    rb.gravityScale = originalGravity;
    isDashing = false;

    Time.timeScale = 1f;
    Time.fixedDeltaTime = 0.02f;
    spriteRenderer.color = originalColor;

    yield return new WaitForSeconds(dashCooldown);
    canDash = true;
}
```

(a) Script que ejecuta la acción

Clones temporales

Nos sentimos especialmente orgullosos de esta mecánica, cuyo desarrollo supuso un gran esfuerzo, pero que finalmente resultó muy enriquecedora para la experiencia de juego.

El script `ghostPlayerController` gestiona una instancia "fantasma" del personaje que registra las últimas acciones ejecutadas por el `PlayerController`. Al pulsar la tecla `R1` o la tecla `[c]`, se genera una copia que reproduce dichas acciones en el entorno del juego, lo que aporta tanto un efecto visual satisfactorio como una clara ventaja táctica.

```
public class GhostPlayerController : Playercontroller
{

    1 reference
    private IEnumerator ReplayInputs()
    {
        while (currentIndex < replayInputs.Count)
        {
            var input = replayInputs[currentIndex];

            // Feed inputs directly
            moveInput = input.move;
            IsMoving = moveInput != Vector2.zero;
            SetFacingDirection(moveInput);

            runHeld = input.run;
            IsRunning = runHeld;

            if (input.jump)
            {
                // Simulate jump logic
                if (IsAlive && !isHanging && touchingDirections.IsGrounded && CanMove)
                {
                    animator.SetTrigger(AnimationStrings.jumpTrigger);
                    rb.linearVelocity = new Vector2(rb.linearVelocity.x, jumpImpulse);
                }
            }

            if (input.attack)
            {
                animator.SetTrigger(AnimationStrings.attackTrigger);
            }

            currentIndex++;
            yield return new WaitForSeconds(replayInterval);
        }

        // Wait a bit before disappearing
        yield return new WaitForSeconds(0.5f);
        Destroy(gameObject);
    }
}
```

(a) Script que lanza el fantasma

13.4 Herramientas adicionales

Cinemachine Camera

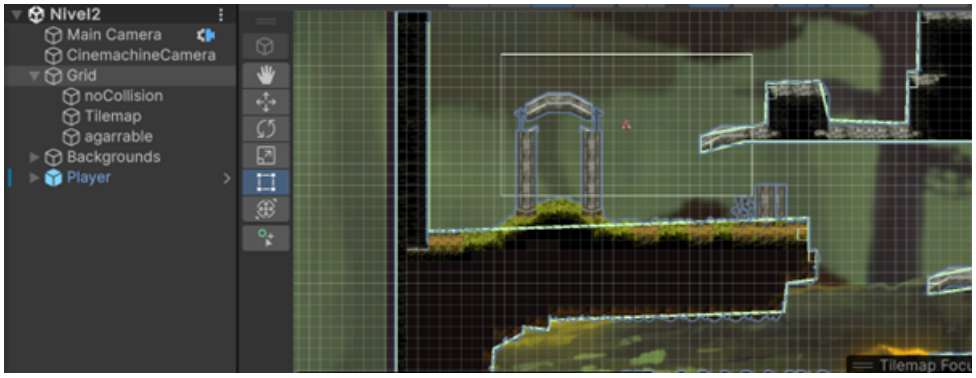
Se utilizó el paquete externo gratuito `Cinemachine`, que simplifica notablemente la implementación de una cámara que sigue al personaje principal. Además, incorpora la función `Impulse()`, que genera un efecto de sacudida de la cámara. Esta funcionalidad se integró como detalle final en la ejecución de la mecánica de los clones temporales.

13.5 Diseño de niveles

El juego ha sido diseñado para funcionar con una resolución base de 16x16 píxeles. Para el diseño de niveles, se empleó la herramienta `TileMap2D`, con la que se fueron dibujando manualmente los escenarios sobre una cuadrícula.

Se utilizaron tres capas de `Tilemaps`: una para los elementos puramente estéticos que no colisionan con el personaje, otra para los que sí lo hacen, y una tercera para los bloques

que pueden ser agarrados o escalados. Los fondos también se diseñaron por capas, con el objetivo de dotarlos de mayor profundidad y atractivo visual. La capa colisionable emplea tanto un `Tilemap Collider` como un `Composite Collider` para fusionar los elementos en una sola malla de colisión.



(a) Los 3 Tilemaps

14 Problemas encontrados

14.1 Colisiones entre personajes y niveles

El error más persistente y frustrante estuvo relacionado con la interacción entre las entidades del juego y los `Colliders` de los niveles. A menudo quedaban atascadas en zonas inesperadas o giraban de forma errática, entre otros problemas.

Para solucionarlo, se intentó suavizar al máximo las texturas. Las formas irregulares se trasladaron a la capa estética, y se ajustó el parámetro `Edge Radius` del `Composite Collider`, que permite ampliar el margen de detección del colisionador.

(a) Nivel solucionado

14.2 Problemas generales con GitHub

El sistema de control de versiones de Unity permite un máximo de tres colaboradores en su versión gratuita. Al ser cinco miembros en el equipo, se presentaron varias dificultades al integrar los cambios en las distintas ramas del repositorio. Descubrimos demasiado tarde la importancia de los archivos `.meta`, lo cual provocó múltiples conflictos al no incluirlos en los `commits`.