Informe de funcionamiento de Software (Proyecto Arkmed)

Santiago Castañeda Pérez

Universidad Tecnologica de Pereira

Computación Grafica

Profesor: Andrés Felipe Ramírez

22 de mayo de 2025

Índice

1. AudioManager.cs

Este script gestiona la música de fondo y los efectos de sonido globales. Implementa el patrón Singleton para asegurar que solo exista una instancia en toda la aplicación. Permite reproducir un sonido de click y detener la música de fondo. Se utiliza principalmente en menús y transiciones de escenas.

2. MainMenuManager.cs

Controla la lógica del menú principal, incluyendo la navegación entre el menú, el panel de controles y la salida del juego. También se encarga de detener la música de fondo al iniciar el juego y de limpiar los datos guardados del jugador.

3. IntroManager.cs

Gestiona la pantalla de introducción del juego. Muestra texto con efecto de tipeo, reproduce un sonido por cada letra y realiza un fade-in en el panel de fondo. Inicializa los datos del jugador en `PlayerPrefs` si no existen y al finalizar la introducción, carga la escena principal del juego.

4. FinalManager.cs

Similar a `IntroManager`, pero se utiliza para mostrar el texto final del juego antes de pasar a los créditos. Incluye efecto de tipeo, fade-in y sonido de tipeo.

5. Credits.cs

Muestra los créditos del juego con el mismo sistema de tipeo y fade-in que la introducción y el final. Al terminar, regresa al menú principal.

6. TextoParpadeante.cs

Hace que un texto (usualmente una instrucción o mensaje) parpadee cambiando su transparencia con una función seno, para llamar la atención del jugador.

7. PoliceSiren.cs

Simula el parpadeo alternado de las luces de una sirena policial, activando y desactivando dos GameObjects (luces roja y azul) a intervalos regulares.

8. PauseManager.cs

Gestiona el menú de pausa. Permite pausar y reanudar el juego, ocultando o mostrando la UI del jugador según corresponda. También permite salir al menú principal destruyendo el objeto del jugador.

9. DeathMenu.cs

Controla el menú que aparece al morir el jugador. Permite reiniciar el nivel o volver al menú principal, asegurando que solo exista una instancia de este menú.

10. PlayerMovement.cs

Script principal del jugador. Gestiona el movimiento, salto, disparo, recarga, uso de botiquines, recogida de ítems, daño, muerte y actualización de la UI. Implementa la lógica de interacción con el entorno y la persistencia de datos del jugador.

11. PlayerInventory.cs

Gestiona el inventario de llaves del jugador. Permite tomar y usar llaves, y lleva la cuenta de cuántas tiene.

12. PlayerDoorRaycast.cs

Permite al jugador interactuar con puertas usando un raycast. Si el jugador tiene una llave, puede abrir la puerta y cambiar de escena. Muestra mensajes de UI según si tiene o no llaves.

13. DoorScript.cs

Script para puertas (3D). Permite cambiar de escena si el jugador tiene una llave, usando colisiones en vez de raycast.

14. KeyPickup.cs

Permite recoger llaves en el entorno 3D. Al recoger una llave, la añade al inventario del jugador y destruye el objeto llave.

15. ItemPickup.cs

Permite recoger ítems (botiquines, munición, etc.) en el entorno 2D. Muestra un mensaje de recogida cuando el jugador está cerca y lo oculta al alejarse.

16. PlayerSystemRaycast.cs

Permite al jugador interactuar con sistemas especiales (como consolas o paneles) usando un raycast, siempre que tenga al menos una llave. Muestra un mensaje de interacción y abre el menú del sistema si es posible.

17. SystemInteractionMenu.cs

Gestiona el menú de interacción con sistemas especiales. Permite elegir entre dos opciones, cada una de las cuales carga una escena diferente. Solo se puede abrir si el jugador está cerca y tiene una llave.

18. PlayerDocumentRaycast.cs

Permite al jugador leer documentos en el entorno usando un raycast. Muestra un mensaje de lectura y permite abrir/cerrar el panel de lectura con una tecla.

19. PlayerDocumentInteraction.cs

Alternativa para la interacción con documentos usando zonas de colisión en vez de raycast. Permite abrir y cerrar el panel de lectura con la tecla E.

20. InteractionZone.cs

Zona de interacción para documentos (3D). Detecta cuando el jugador entra o sale de la zona de un documento y actualiza la referencia al documento actual.

21. DocumentReader.cs

Contiene el texto de un documento y un sonido opcional de apertura. Permite obtener el texto y reproducir el sonido al abrir el documento.

22. ZombieAI.cs

Controla el comportamiento de los enemigos tipo zombie. Incluye movimiento hacia el jugador, ataque, recibir daño, muerte, soltar botín y llaves, y actualizar la barra de vida.

23. BoosFinalAI.cs

Controla el jefe final. Incluye movimiento, ataques normales y especiales, disparo de bolas de energía, recibir daño, muerte, soltar llave y actualizar la barra de vida del jefe.

24. BulletScript.cs

Gestiona el comportamiento de las balas disparadas por el jugador. Detecta colisiones con zombies y el suelo, aplica daño y muestra una explosión.

25. FloatingText.cs

Muestra textos flotantes (por ejemplo, puntuación o mensajes críticos) que se desvanecen y se mueven hacia arriba.

26. CameraFollow.cs

Hace que la cámara siga al jugador dentro de unos límites definidos, con un offset configurable.

27. GameManager.cs

Clase estática para almacenar variables globales como munición, puntuación y botiquines.

28. PlayerData.cs

Clase estática para guardar y cargar los datos del jugador entre escenas, como vida, munición, botiquines y puntuación.

29. DialogoManager.cs

Gestiona la visualización de diálogos con efecto de tipeo y permite avanzar entre líneas con una tecla.

30. MissingScripts.cs

Script de utilidad para el editor de Unity. Permite buscar componentes faltantes (scripts perdidos) en todos los GameObjects de la escena.