Asignatura	Datos del alumno	Fecha
Desarrollo de Videojuegos I	Apellidos:	20/02/2026
	Nombre:	

Actividad 3. Grupal. Top-Down completo

Integrantes

- Jorge Blacio
- Luis Diaz
- Gustavo Totoy

Introducción

El prototipo de top down se desarrolló usando el motor Unity en su versión 6.3.8f1, y se realizó un build web que se encuentra disponible en la plataforma Itchio.

Enlaces del proyecto

- Enlace de GitHub
  - https://github.com/gtotoy/unir-top-down
- Enlace de Itchio
  - https://jorgeblacio.itch.io/the-legend-of-top-down
- Enlace a video de gameplay y organización de proyecto Unity
  - https://alumnosunir-my.sharepoint.com/:v:/g/personal/luis_diaz609_comunidadunir_net/IQAe6U6kUtmPSKlNf-WncstRAdPOvf9GlgGtoUVLcdL82so?e=kZvz3J

Listado de Implementaciones Extra

- **Menús de juego**
  - Menú principal con acceso directo a la partida.
  - Menú de pausa accesible en cualquier momento durante el juego.

- Menú de Game Over con opción de reinicio.

- **Mecánicas adicionales del jugador**
  - Bloqueo (clic derecho): reduce el daño recibido a la mitad y elimina el tinte rojo de impacto.
  - Dash (barra espaciadora): movimiento rápido con cooldown, permite esquivar proyectiles o reposicionarse.
- **Sistema de enemigos con comportamientos diferenciados**
  - Hachero: persigue y ataca en rango corto con mecánica de cuerpo a cuerpo directa.
  - Lancero: mantiene distancia prudencial y ataca en rango medio; retrocede si el jugador se acerca demasiado.
  - Arquero: huye del jugador y dispara flechas como proyectil físico con detección de impacto.
- **Sistema de spawning de enemigos**
  - Puntos configurables de spawn por grupos de combinaciones de tipos de enemigo, parametrizables desde el inspector.
- **Elementos de recuperación**
  - Spawning automático de med kits (representados como filetes) con radio y frecuencia configurables.
- **Diseño sonoro**
  - Música de fondo en loop para cada nivel.
  - Efectos de sonido por acción: ataque, bloqueo, dash, recibir daño, recoger med kit, abrir puerta.
- **Efectos visuales y de retroalimentación**
  - Animaciones de sprites para dash, impacto, consumo de med kit y apertura de puerta.
  - Tinte rojo al recibir daño, tinte alternativo al bloquear.
  - Efecto de cámara (avatar UI) con sacudida y flash al recibir daño.
- **Diseño de niveles**

- Nivel 1: tres tipos de enemigos, puerta con cerradura, llave otorgada al eliminar todos los enemigos.
- Nivel 2: jefe final con máquina de estados que combina ataque, bloqueo y dash de forma aleatoria ponderada.

Desarrollo del proyecto

A continuación, se encuentran capturas de la configuración del proyecto de Unity:

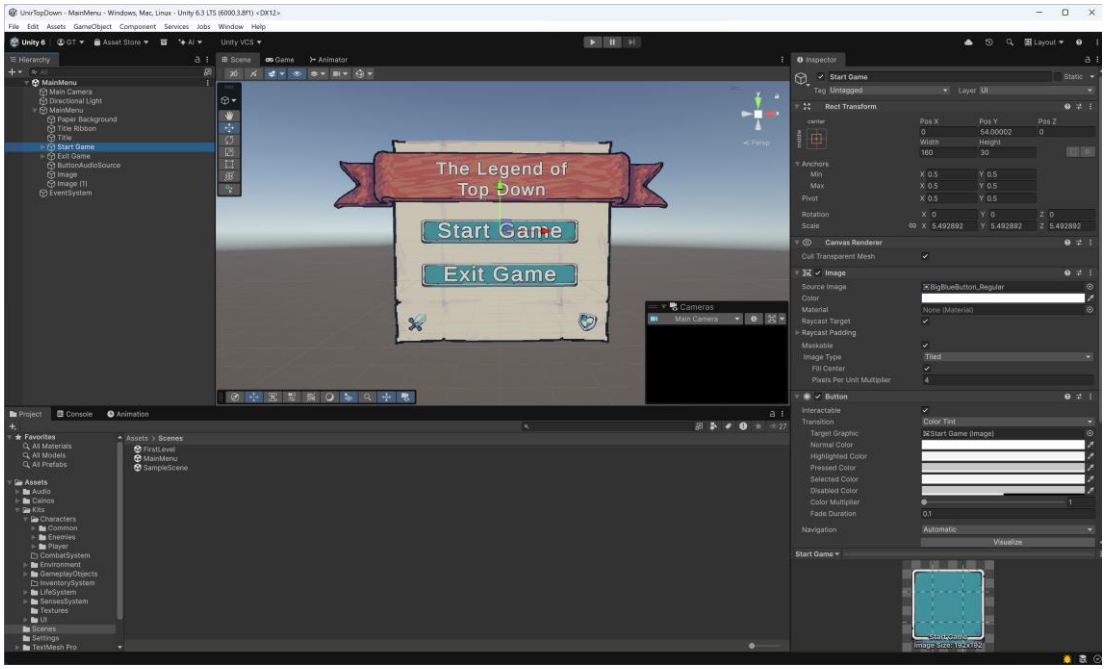

Figura 1. Escena del menú principal. Fuente: Elaboración propia.

Asignatura	Datos del alumno	Fecha
Desarrollo de Videojuegos I	Apellidos:	20/02/2026
	Nombre:	

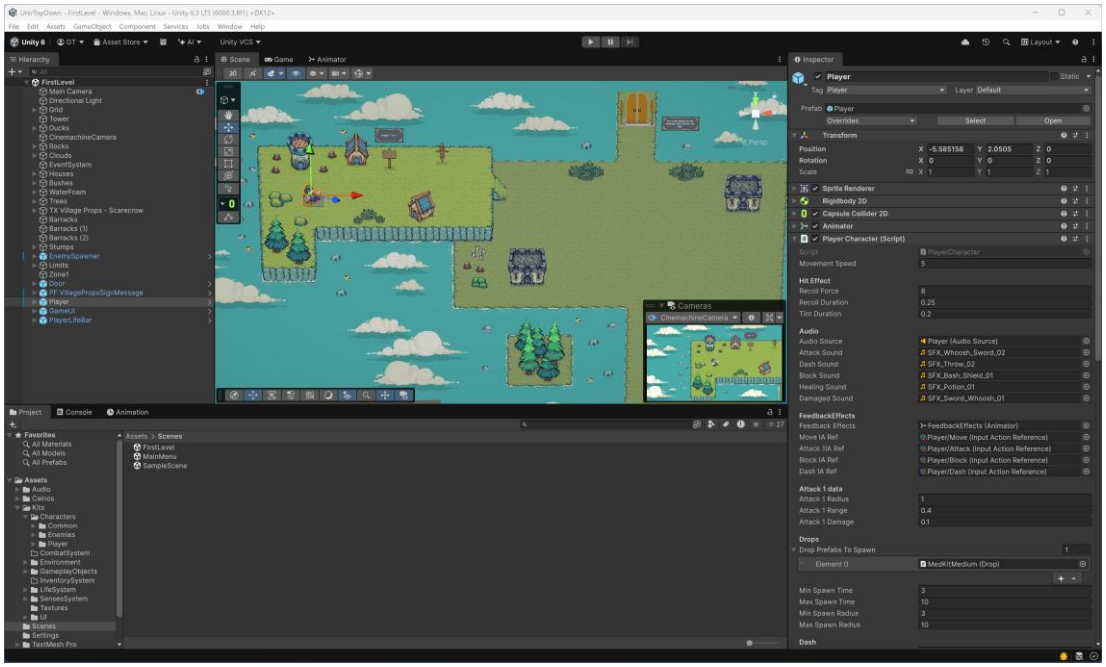

Figura 2. Escena del primer nivel que tiene un spawner con los 3 tipos de enemigos y requiere de eliminarlos a todos para obtener la llave de la puerta al castillo del jefe. Fuente: Elaboración propia.

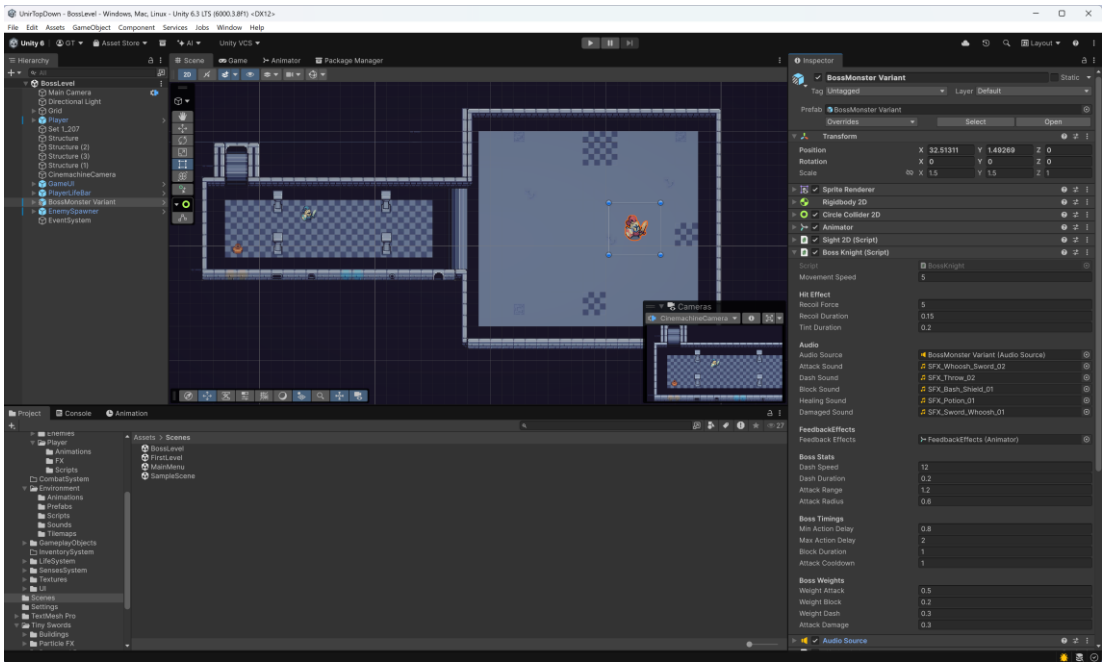

Figura 3. Escena del segundo nivel que se realiza al interior del castillo del jefe final rodeado de sus aliados. Fuente: Elaboración propia.

Asignatura	Datos del alumno	Fecha
Desarrollo de Videojuegos I	Apellidos:	20/02/2026
	Nombre:	

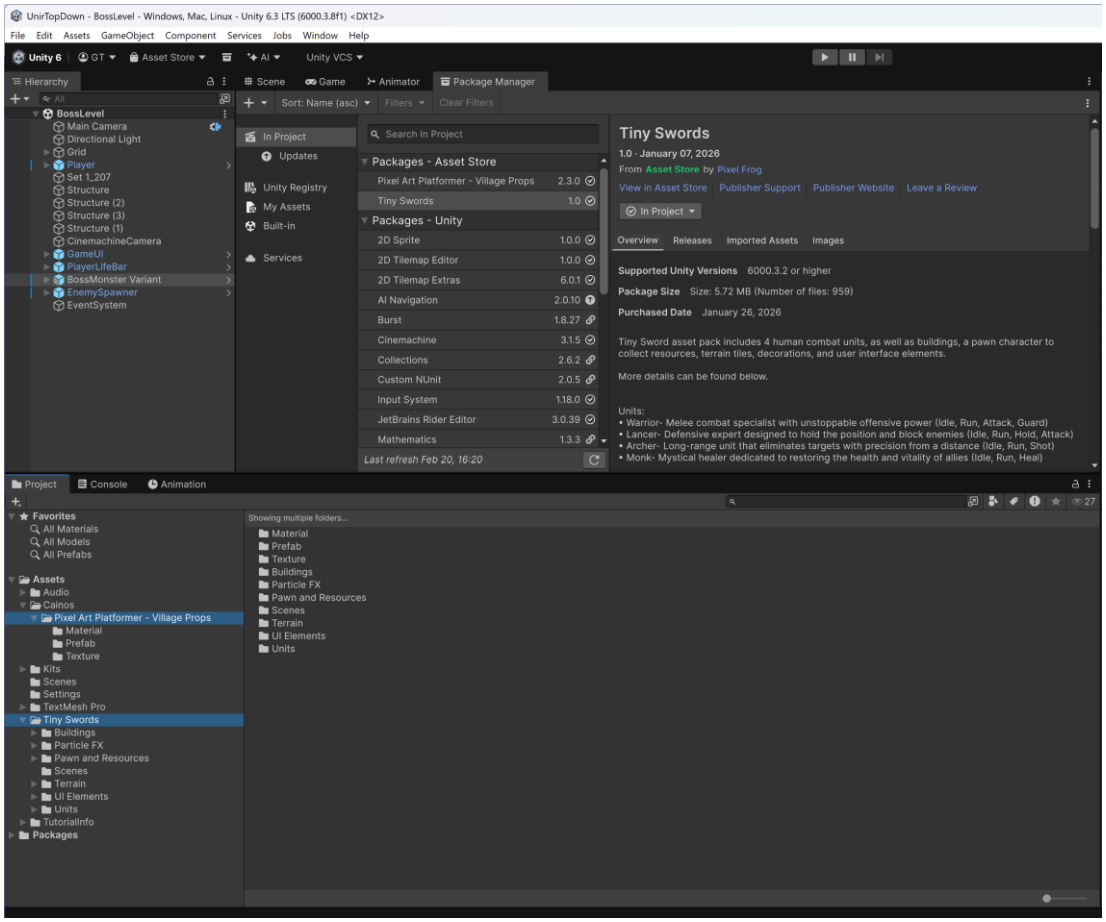

Figura 4. Assets packages obtenidos del Asset Store. Fuente: Elaboración propia.

A continuación, se encuentran capturas del código C# programado:

Asignatura	Datos del alumno	Fecha
Desarrollo de Videojuegos I	Apellidos:	20/02/2026
	Nombre:	

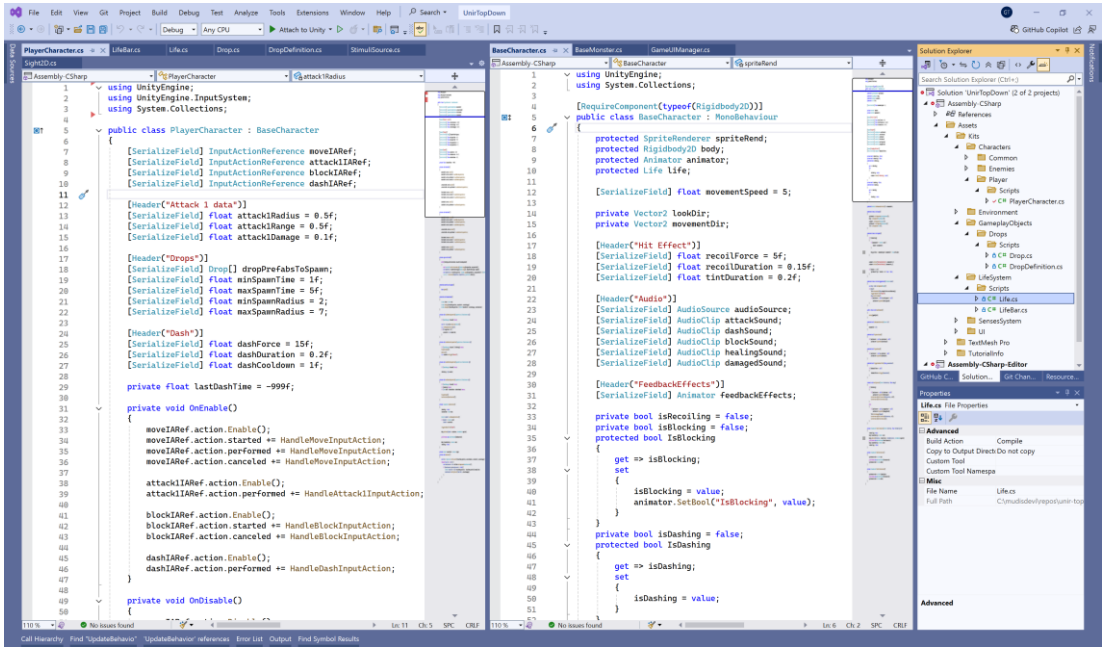

Figura 5. Snippet de código del PlayerCharacter que muestra a su lado su clase base BaseCharacter siguiendo la arquitectura definida en clases. Fuente: Elaboración propia.

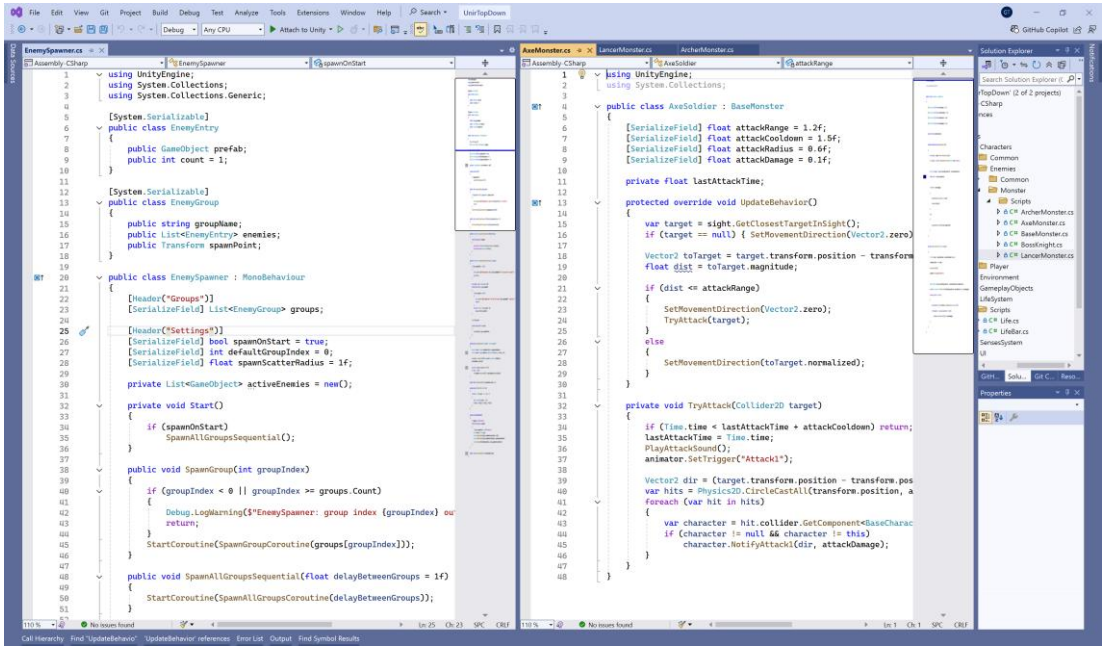

Figura 6. Snippet de código del EnemySpawner + AxeMonster. Adicional a AxeMonster también constan LancerMonster y ArcherMonster con comportamientos distintos. Fuente: Elaboración propia.

Asignatura	Datos del alumno	Fecha
Desarrollo de Videojuegos I	Apellidos:	20/02/2026
	Nombre:	

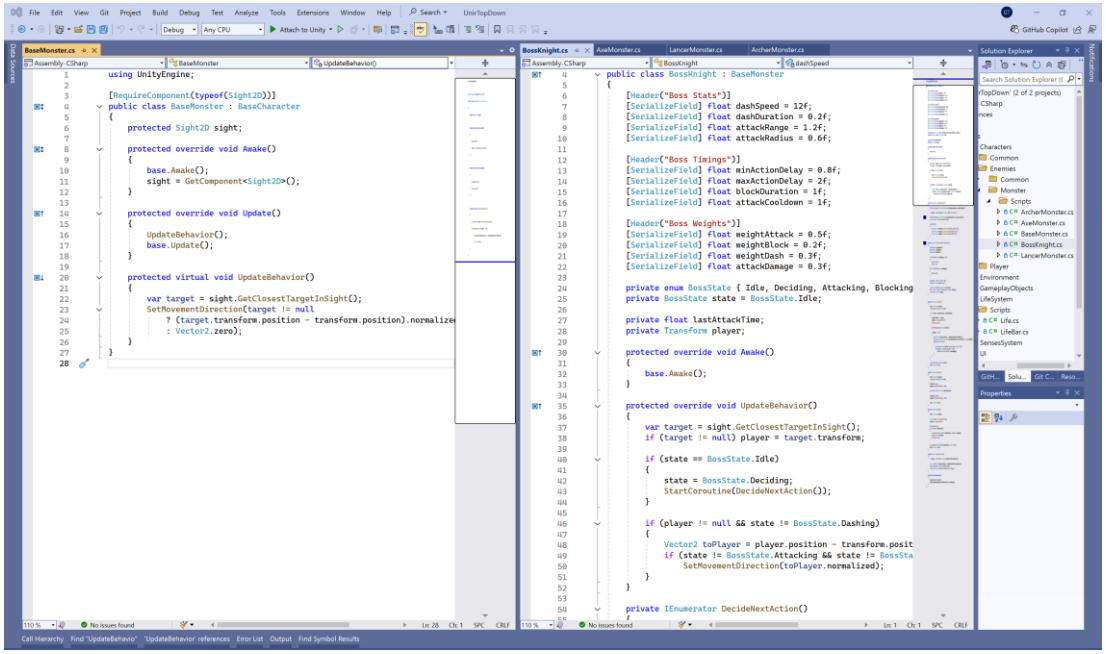

Figura 7. Snippet de código del BaseMonster con BossKnight a su lado. Fuente: Elaboración propia.

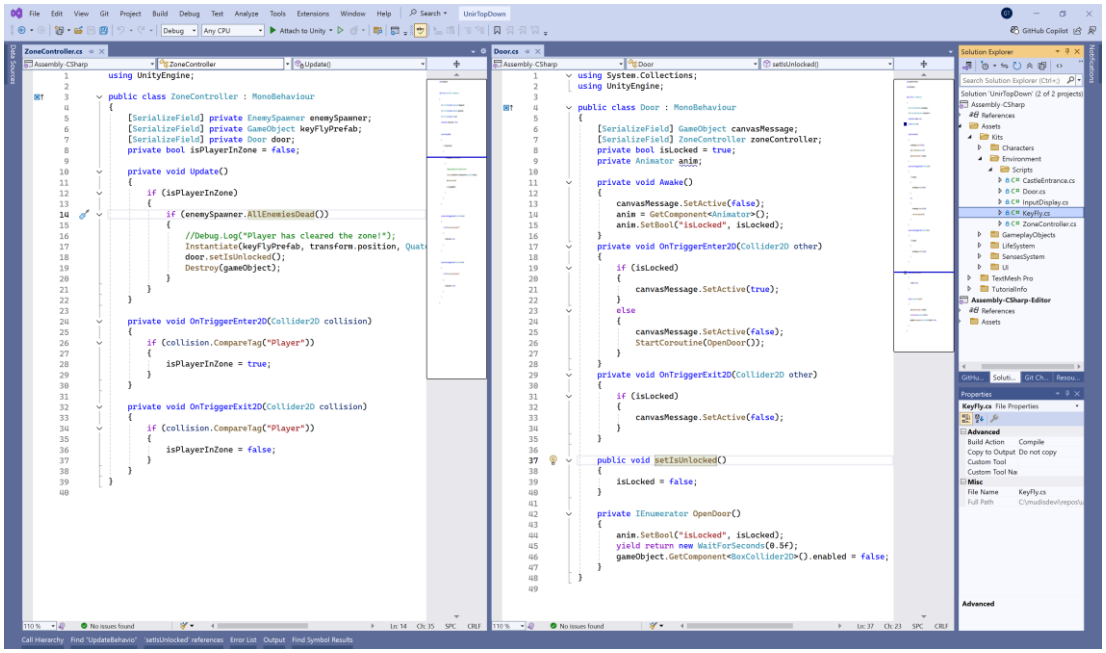

Figura 8. Snippet de código de ZoneController + Door. Responsables de instanciar la llave y controlar el acceso de la puerta para continuar al siguiente nivel. Fuente: Elaboración propia.

Conclusiones

El desarrollo del prototipo ha resultado una experiencia formativa completa que ha obligado al equipo a investigar y resolver problemas reales de diseño y programación de videojuegos.

Arquitectura extensible: la decisión de centralizar la lógica común en BaseCharacter y delegar el comportamiento específico a los subtipos mediante UpdateBehavior() permitió añadir nuevos arquetipos de enemigo sin modificar código existente, siguiendo el principio Open/Closed.

Problemas con la física de proyectiles: el principal problema técnico fue la aplicación continua de fuerza al jugador cuando una flecha colisionaba. La causa raíz fue la interacción entre el *Rigidbody2D* del proyectil y el del jugador antes de que Destroy surtiera efecto. La solución fue desactivar body.simulated inmediatamente tras el impacto y usar un flag hasHit para evitar re-entradas.

Sincronización de animaciones y lógica: ajustar el timing del ataque del jugador con el momento real de detección de daño requirió varios ciclos de iteración, resolviéndose con un mejor balance de los tiempos de animación y de los rangos y radios configurables que definimos para el golpe.

Máquina de estados del jefe: un desafío de diseño fue evitar que el jefe se quedara bloqueado en un estado indefinidamente. Se resolvió garantizando que cada corrutina de acción siempre establece state = BossState.Idle en su bloque final, incluso si el jugador sale del rango de visión.

Balance de jugabilidad: encontrar el equilibrio correcto en los pesos de las acciones del jefe y los cooldowns de dash y bloqueo del jugador requirió muchas sesiones de prueba. Los parámetros finales se expusieron en el inspector de Unity para facilitar el ajuste sin recompilar.

Trabajo en equipo y control de versiones: el uso de GitHub con Unity requirió configurar correctamente el *.gitignore* y establecer convenciones claras para evitar conflictos en archivos de escena y prefabs.

En conjunto, el proyecto ha permitido al equipo consolidar conocimientos sobre arquitectura de software orientada a juegos, inteligencia artificial básica, diseño de niveles y flujo de trabajo colaborativo con control de versiones. Las implementaciones extra han supuesto un reto estimulante que ha impulsado la búsqueda activa de soluciones fuera del temario impartido en clase.