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
  + TODO

**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:

The Legend of Vanguard

AI-generated content may be incorrect.

Main menu

The image depicts a digital game interface, featuring a scene with a tower, various game objects, a character, and interactive elements like a player life bar and sound effects.

AI-generated content may be incorrect.

Primer nivel

Segundo nivel

Packages from package manager

The image appears to be a snippet of a software interface, possibly a version control system, with various tabs and elements such as 'File', 'View', 'Debug', and code listings including classes like `BaseCharacter` and `PlayerCharacter` in a C# scripting environment.

AI-generated content may be incorrect.Player Character

Monsters

Jefe

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.

Objetivos

Con esta actividad vas a conseguir afianzar los conocimientos vistos durante los temas en los que hemos desarrollado el prototipo de un Top-Down y desarrollarás un carácter creativo e independiente al introducir mecánicas y dinámicas nuevas de juego dentro del prototipo desarrollado en clase. Con ello, aprenderás a manejarte con mayor soltura con el editor, aprenderás nuevos conceptos por tu cuenta y aprenderás a enfrentarte a problemas reales.

Pautas de elaboración

En base al prototipo desarrollado en clase de un Top-Down, los integrantes del grupo han de completar este prototipo añadiendo mecánicas y dinámicas nuevas de elección totalmente **libre**, siempre y cuando estas no «rompan» con la idea original del prototipo. Algunas ideas podrían ser:

* Utilización de los items que hemos obtenido**.**
* Toma de decisiones en los diálogos**.**
* Sistema de misiones simples (utiliza eventos)**.**
* Transición entre escenas (entrar en casas, cuevas, etc**.**).
* Mecanismos (puertas, movimiento de bloques, botones, etc**.**).
* Entrar en escena donde se desarrolle un combate por turnos.
* Sistemas de partículas**.**
* Iluminación**.**
* Sistema de compra con los puntos para adquirir nuevas habilidades**.**

Extensión y formato

Para completar la actividad, se deberá entregar un **único documento en formato pdf (máximo 10 páginas)** que incluya lo siguientes apartados:

* Introducción
* Enlaces del proyecto: **enlace al GitHub** del proyecto y enlace al **ejecutable de Itchio**.
* Incluir un vídeo (**2-5 minutos**) en formato .MOV o .MP4 a modo de gameplay donde se observen las implementaciones realizadas en el proyecto.
* Listado de **implementaciones extra**. Una enumeración de las implementaciones extra añadidas al proyecto con una breve descripción de cada una.
* Desarrollo del proyecto, es decir, un desarrollo de los pasos realizados, incluyendo **capturas de código**, **captura de la configuración en Unity** etc.
* Conclusiones y problemas encontrados en el desarrollo de la actividad.

En el caso de que alguno de los archivos (por ejemplo, el vídeo de gameplay) sea demasiado pesado para poder ser subido al campus, se deberá incluir en el archivo en PDF *(portable document format)* que incluya un enlace al OneDrive de la cuenta institucional del estudiante de UNIR.

Cualquier otro archivo enviado a través de un enlace de Drive, WeTransfer, etc., no será evaluado y la calificación de dicho apartado será 0.

Rúbrica

|  |  |  |  |
| --- | --- | --- | --- |
| Actividad:  Top-Down completo | Descripción | Puntuación máxima  (puntos) | Peso  % |
| Criterio 1 | Se ha creado un concepto de juego **creativo** y **divertido** con una **jugabilidad** y un **desafío** adecuados en base al prototipo desarrollado en clase. | 1,5 | 15% |
| Criterio 2 | El **apartado artístico** del juego es el adecuado. Se han cuidado los detalles y la apariencia del proyecto. | 1,5 | 15% |
| Criterio 3 | El juego compila, funciona **sin errores** por consola y se han seguido las convenciones de escritura de código. | 1 | 10% |
| Criterio 4 | Se incluyen **todas las funcionalidades** tal y como se han hecho en clase | 1 | 10% |
| Criterio 5 | Se han incluido **implementaciones extra** de calidad y de implementación compleja. El alumno ha usado sus conocimientos del programa de forma creativa. | 5 | 50% |
|  |  | **10** | **100 %** |