Grado en diseño y desarrollo de videojuegos (Móstoles).

Grupo F

Miembros:

Mauricio Fernández Rojas

Daniel Bermejo Valverde

Javier Martín Ullán

Repartición del trabajo:

El trabajo se repartió en función de protocolos a implementar, cada miembro decidió que protocolo deseaba intentar aplicar de manera independiente, apoyado, en caso de necesitar ayuda por los otros miembros. Dada la dinámica complementaria existente en el desarrollo de un proyecto en Unity, además del sistema empleado para compartir dicho proyecto, obligó a que varios miembros trabajaran y puliesen cada implementación como conjunto.

Para desarrollar el trabajo, buscando mantener simplicidad y organización se optó por mantener copias independientes del proyecto. Una vez terminada una aplicación de un prototipo los miembros se agruparon y replicaron el trabajo realizado mientras se hacía una retrospectiva de proceso realizado, obteniendo así cada miembro una visión general del proyecto y permitiéndonos verificar la calidad del producto según se desarrollaba.

Para enumerar que parte del trabajo cada miembro realizó o participo se añadirán los miembros al final de la implantación de cada protocolo. Los miembros no mencionados participaron en esas secciones de ninguna manera.

Enumeración de los patrones y uso:

Gameloop: Este patrón de diseño no “tiene sentido” aplicarlo en Unity puesto que el motor ya lo implementa en su propia manera. Se ha buscado hacer uso de las funciónes Update y Fixed Update para controlar el paso del tiempo dentro del videojuego. Se ha necesitado realizarlo de tal forma que nos permita pausar el juego modificando el Time.timeScale cuando accedíamos al menú.

Miembros: Mauricio Fernández, Daniel Bermejo.

Component:

Este es otro patrón de diseño que no “tiene sentido” implementar de manera manual en C# en el contexto de Unity. Se ha empleado la capacidad de modulación en componentes de Unity para crear propiedades aisladas y luego implementarlas con eficiencia en cada sección del desarrollo. Se empleó principalmente para dotar de propiedades a los elementos de la escena, aunque algunas de estas fueron posteriormente sustituidas durante la implementación de otros protocolos.

Por mencionar algunos específicos esta su uso en la cámara del videojuego que se hizo que siguiese al usuario empleando un componente C# y cada propiedad específica de cada variante de los prefabricados.

Miembros: Mauricio Fernández, Daniel Bermejo.

Observer:

Este protocolo se empleó para controlar de forma aislada la llamada a eventos como sonidos y el control de datos de la UI, como las monedas recogidas y la vida del personaje. Cada uno de los elementos de Audio Manager implementa un observer, menos background music.

Por ejemplo, la barra de vida del personaje se actualiza en positivo y negativo dependiendo de si recibe daño el personaje o si ha conseguido una vida.

Miembros: Mauricio Fernández, Daniel Bermejo.

Dirty Flag:

Se ha optado por aportar el patrón de diseñ dirty flag para puntos específicos del nivel llamados checkpoints. Cuando el personaje llega a estos checkpoints se guarda la posición del personaje y la cantidad de monedas de oro que ha recogido, además las monedas de oro también se quedan desactivadas en posteriores cargas de la partida. Se ha optado por no hacerlo con las monedas de plata para incentivar que sea un reto pasarte el nivel de una sola vez.

Se permite desde el menú principal borrar los datos guardados del nivel específico, en el botón con la papelera de reciclaje.

Miembro: Daniel Bermejo.

Fly Weight:

Este protocolo se empleó para parametrizar los enemigos llamados “Roamers”, se buscaba reducir la carga de memoria creando un Scriptable Object con la vida y la velocidad de los enemigos y sus tres imágenes de sprites asociadas. De esta manera se crearon dos Scriptable Objects, uno para los Battsies (Murciélagos) y otro para los Ballies (Esferas).

Este protocolo se implementó junto a Prototype.

Miembro: Mauricio Fernández.

Command: No se implementó**.**

State:

El protocol State **NO** se pudo implementar. Se prepararon a los enemigos con un State Machine basado en if´s anidados, con tres estados identificados por los sprites establecidos en el Scriptable Object, para luego distribuir los estados en funciones específicas siguiendo el patrón, pero al final no dio tiempo para conseguir implementarlo.

Miembro: Mauricio Fernández.

Prototype:

Prototipe basado directamente en código c# no se implementó, pero si se empleó en gran medida el uso de Prefabs para la construcción de la escena, creación de disparos, tanto del jugador como estatuas, instanciado de monedas y enemigos, etc.

Como se comentó previamente se mezcló con el patrón Scriptable Objects para la implementación de los enemigos Roamers, construyendo el enemigo basado en el Scriptable Object y luego realizando instancias del Prefab.

Se implementó para crear variantes de monedas, bloques, plataformas y estatuas. Haciendo uso de variantes de prototipos para poder controlarlos de manera eficiente.

También, junto a Object Pool, se realizaron implementaciones a tiempo de ejecución de las monedas, bolas de fuego y corazones.

Miembro: Mauricio Fernández, Daniel Bermejo.

Object Pool:

Object Pool se implementó para reducir el uso de memoria en tiempo de ejecución. En Object Pools de implementaron una serie de elementos que brindan una pool de objetos desactivados que luego son llamados desde otros elementos para su activación, luego se desactivan y regresan a la pool. Las pools son dinámicas permitiendo añadir objetos a la lista en caso de ser necesitados y que el resto es estén ya activos.

Se implementó en las estatuas, para las bolas de fuego y paredes de fuego, para los bloques ? con monedas y corazones, y para los disparos del jugador.

Miembro: Mauricio Fernández.

Aspectos a destacar:

Por último comentar los elementos de terceros. No se copió y pego ningún código, por lo menos no de manera activa, si se empleó como referencia tutoriales, el manual de Unity con sus ejemplos, y algunas respuestas en redes sociales a personas con problemas similares a los que nos ocurrían. En ningún momento se copió y pego el código tal cual.

Todos los elementos visuales fueron creados por el miembro Mauricio Fernández Rojas.

Todos los audios, todos en la carpeta de SFX, son contenidos de terceros descargados de la plataforma Pixabay.com

El único elemento visual no propio es el icono de la papelera, que es editado de internet.

Original:  Editado: