Práctica 1 de PANG

Fecha límite de entrega: 24 de octubre

En esta práctica se va a programar un juego simplificado basada en Pang. Habrá unas pompas que rebotarán en el suelo y el objetivo del jugador es dispararlas para destruirlas evitando ser alcanzadas por estas pompas. El juego original lo podéis observar en el siguiente vídeo:

https://www.youtube.com/watch?v=w701xE60HtE

En esta práctica sólo se implementará una versión muy simple del juego, y sólo debes implementar lo que se pide en esta práctica de la forma que se pide.

Descripción desglosada de la práctica

En este juego se hará en dos dimensiones. Por tanto, todos los elementos que se creen y se usen deberán ser en dos dimensiones, tales como (2D Object->Sprites -> (Circle, Square, Capsule).

Configuración de la Cámara

La cámara no se moverá, dado que el escenario es fijo. Configura los siguientes valores:

Aspect ratio: 16:10 (En pestaña superior entre Display y Scale)

Size: 5 (En inspector en la cámara)

Creación de la escena:

Se crearán los bordes de la escena. Tendrá que haber un suelo abajo, un techo arriba, y paredes a los lados. Todos estos cuerpos deben tener su BoxCollider2D y RigidBody2D para permitir que las pompas no puedan traspasarlas (mirar tema de "Física"). No deben moverse. Se puede conseguir con "constraints"-> Freeze position

Todos los límites del escenario de juego se recomiendan agrupar apropiadamente en un EmptyObject llamado "SceneBorders". Se usará un Sprite->Square y escalarlo a medida. Una vez que te funcione el suelo, se puede duplicar para obtener las paredes

El grosor de paredes y suelos será de 0.5 unidades Unity. Esto tendrá que ponerse en el Transform>Scale del inspector en X o Y dependiendo de si el elemento es horizontal o vertical. Los suelos y paredes deben ajustarse a los bordes de la parte visible. Este se puede conseguir con una longitud de 16 en el caso del suelo y el techo, y una longitud de 10 en el caso de las paredes. Las posiciones de suelo y techo se consiguen con Y = -5, Y= +5, y las posiciones de las paredes se consiguen con X = -8 y X=8. Observa la relación de estas medidas con las explicadas en la configuración de la cámara.

Creación del jugador:

Inicialmente puede ser cualquier forma geométrica (e.g. Sprite->capsule), pero posteriormente se puede mejorar. Para permitir futuras extensiones, se pondrán el comportamiento en un empty GameObject (llamado Player) con su collider2D. En el interior se usarán la figura(s) geométrica(s) que se deseen, sin usar los colliders dado que este se pondrá en el objeto en el Player.

El personaje del jugador no usará las leyes de la física para el movimiento. NO usará por tanto RigidBody2D. Tendrá BoxCollider2D para las colisiones.

Implementad el componente para que se pueda mover en base a una velocidad. Recuerda multiplicar la velocidad por Time.deltaTime. Para ello, se tomará la dirección de la entrada horizontal. Se calculará nueva posición como la suma de vectores.

Sólo se cambiará la nueva posición si está dentro de los límites jugables, es decir si x está en el intervalo visible y dentro de las paredes. Para ello deberías calcularlo en código a partir del tamaño visible según la cámara (16 de ancho entre -8 y 8), el groso de las paredes y el grosor del personaje.

Para esta práctica, se establecerán todos esos tamaños usados como atributos de la clase, pero porque se está en un contexto de aprendizaje. Posteriormente, se podrían obtener estos valores de otra manera (e.g. con referencias a los objetos implicados), pero esto no se pide en esta práctica por simplificación.

Creación de las Pompas (Bubble):

Inicialmente se creará un círculo (Sprite->Circle). Sigue las instrucciones de “Rebote” del apartado de física. Recuerda usar los componentes BoxCollider2D, RigidBody2D, Physics Material2D. Se debe crear y asignar el material físico con la propiedad “Bounciness” para que rebote y “Friction” a cero.

Asigna un material para que tenga color.

Para la creación de más pompas se creará un prefab a partir de dicha pompa. Para eso, se establecerá en el editor las pompas en diferentes posiciones (transform) y con diferentes fuerzas iniciales, establecidas con un nuevo atributo.

Se creará un componente “Blowup” para establecer la fuerza inicial que se aplicará al inicio desde el script. De esta forma, tendrá un impulso inicial. Se usará “[SerializeField]” para dos atributos, uno con la dirección inicial “initalForceDirection” que se normalizará y otro “initialForceMagnitude” con la magnitud. De esa manera, se podrán configurar desde el editor el nivel inicial.

Para evitar las colisiones entre las pompas, crearemos los siguiente cuatro layers:

- Bubble
- Scenario
- Bullet
- Player

Luego marcaremos los elementos con los layers adecuados. Podemos marcar los elementos padres y aceptar que se trasladen a sus hijos.

En Edit->Project Settings->Physics 2D-> Collision Matrix: desmarcar la colisión de Bubble consigo misma.

Para agrupar los comportamientos de los elementos inflables (componente “Blowup”), se establecerá un script que contenga la dirección y magnitud de la fuerza inicial, y aplicarla a su RigidBody2D. También tendrá un método “Burst”, para gestionar cuando se pincha.

Se da libertad para que diseñéis un escenario con la cantidad de pompas necesarias para que sea suficientemente difícil de ganar, pero no imposible. Para corregir, se ejecutará con dos pompas con posiciones y fuerzas iniciales diferentes.

Disparos:

Crea un objeto de bala (Bullet). Le asignas una velocidad hacia arriba. Se debe asignar una velocidad muy alta y destruirse cuando su altura supere cierto valor (fuera del campo de visión). Los movimientos deben calcularse con la velocidad multiplicado por Delta Time para que se vea en todas las máquinas a la misma velocidad y no varíe según se cambie el juego. Se deberá usar vectores para calcular y asignar la nueva posición (en concreto se usará Vector2.up).

Para implementar todo lo anterior, puedes crear el Script “BulletMovement”. Este script gestionará las colisiones.

Cuando entre en colisión con una pompa la debe destruir. Para ello usaremos colliders y gestionaremos el evento “OnCollisionDetection2D”.

Para ver si entra en colisión con una pompa, usaremos el DuckTyping. Preguntará a la Pompa tiene el Script característico, por ejemplo “Blowup”, que pincha la pompa con método “Burst”.

Salud del jugador

Las pompas tendrán un componente de script llamado “Harmful” (dañino). Este script detectará las colisiones con OnCollisionEnter2D. Cuando tenga una colisión, usará DuckTyping con el script Health. El script Health será asociado al objeto Player que podrá ser herido (método “Harm”) En el método “Harm” se llamará al método “OnPlayerDamaged” de GameManager (mira siguiente apartado). Cuando esto ocurre, también el elemento dañino se destruye.

Game Manager y Detección de fin de partida:

Haz un game manager (componente GameManager asociado a objeto vacío “GameManager”) usando el patrón Singleton.

Cuando una de las pompas toca al jugador (recibe llamada “OnPlayerDamaged”), se acaba la partida, se muestra un mensaje de que el usuario ha perdido, y se destruye el jugador.

Si todas las pompas han desaparecido, entonces se muestra el mensaje de que ha ganado la partida. Para ello, el game manager usará un contador del número de pompas.

Cada pompa informará en el Start que ha sido creada llamando a un método de GameManager (OnBubbleCreated). También informará cuándo ha sido destruida llamando a otro método del GameManager (OnBubbleDamaged). El GameManager en Awake, inicializará a cero el número de pompas e irá actualizando el número de pompas. Cuando se destruya una pompa se comprueba si era la última (i.e. se ha llegado a cero pompas). El GameManager contendrán una referencia con el UIManager. Sólo el GameManager accederá al UIManager.

UIManager

Toda la gestión del interfaz gráfico la gestionares desde el UIManager. Este componente se asignará al Canvas correspondiente. Tendrá un letrero, donde se pondrá “Has Ganado” o “Has perdido” dependiendo. El método para informar con texto al usuario se llamará “Inform (String message)”

Estructura de carpetas:

Mantén por separado en carpetas diferentes los siguientes elementos:

- Scenes
- Materials
- Scripts
- Prefabs

Normas para nombrar todo en scripts y elementos de Unity

No se puede usar acentos, ñ, espacios ni caracteres especiales en los nombres ni métodos de los scripts, ni en los elementos de Unity. Se recomienda poner nombres en inglés para evitar usar caracteres españoles, entre otros aspectos.

Normas de Entrega

Se subirá al campus virtual un archivo comprimido zip sólo con el contenido de las carpetas “Assets”, “Packages” y “Project Settings” antes de la fecha límite.

Una vez se haya entregado, se procederá a la defensa ante el profesor de vuestro laboratorio. Tenéis que tener todo funcionando y mostrarle lo que el profesor os pida. El profesor podrá hacer una o varias preguntas a cada estudiante para comprobar su grado de participación y conocimiento en la práctica.