SESSIÓ 4.(2)

Últims detalls:
rondes, nivells de dificultat i pantalla d'inici.

Per acabar el petit joc d'exemple, hi afegirem elements que són típics d'aquest tipus de jocs: diferents rondes de joc amb diferent nivell de dificultat. També li posarem una pantalla inicial.

1 Rondes

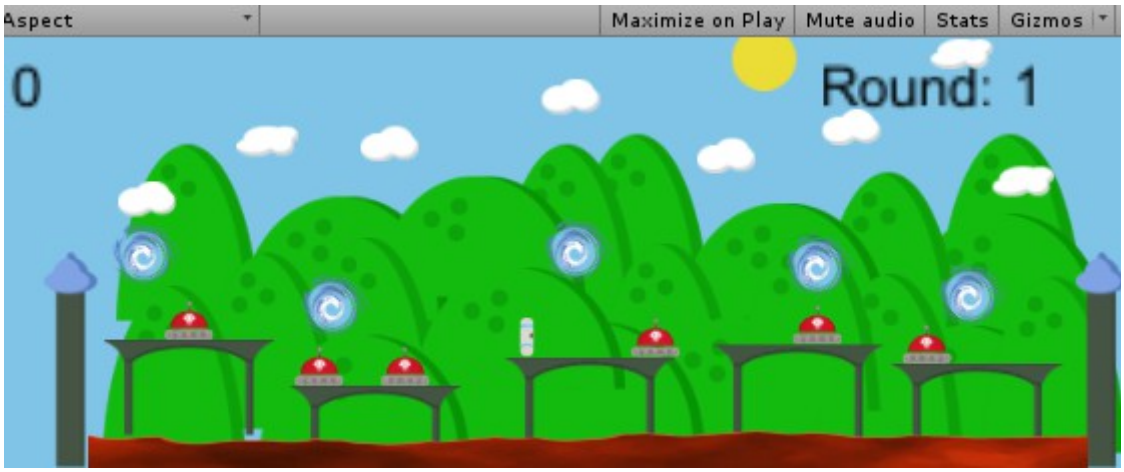
Cada vegada que aconseguim derrotar el **Boss**, passarem a la següent ronda. Per implementar això, simplement afegirem a la cantonada superior dreta un nou indicador **Round**. La feina necessària és molt semblant a la feta per visualitzat la puntuació amb l'objecte **Score**.

Creem un objecte buit de tipus "**3D Text**", fill de **Main Camera**, que anomenem **Round**. Establim el **Color** del text com negre, **Anchor** com Upper Right i **Alignment** com Right. Si cal, modifiquem el vector de **position** del component **Transform** perquè el text quedi a dalt a la dreta. Per tal d'actualitzar correctament el **Round**, creem el nou script **RoundWatcher**, que assignem a l'objecte. El codi de l' script és:

```
using UnityEngine;
using System.Collections;
[RequireComponent(typeof(TextMesh))]
public class RoundWatcher : MonoBehaviour
{
	public int currRound = 1;
	private TextMesh roundDisplayMesh = null;

	void Start ()
	{
		roundDisplayMesh = gameObject.GetComponent<TextMesh>();
		roundDisplayMesh.text = "Round: " + currRound.ToString();
	}

	void OnEnable()
	{
		BossEventController.bossDied += increaseRound;
	}

	void OnDisable()
	{
		BossEventController.bossDied -= increaseRound;
	}

	void increaseRound(int ignore)
	{
		currRound += 1;
		roundDisplayMesh.text = "Round: " + currRound.ToString();
	}
}
```

Comentaris al codi:

- El mètode **Start()** inicialitza el valor del text.
- Al mètode **OnEnable()** l'objecte **Round** s'apunta a escoltar, per mitjà del mètode **increaseRound()**, l'event **bossDied**, que generarà l'script **BossEventController**, de seguida que es detecti la destrucció del **Boss**.
- Al mètode **OnDisable()** es realitza l'operació contraria a **OnEnable()**.
- El mètode **increaseRound()** el programem nosaltres (no s'hereda de **MonoBehaviour**) i és el **listener** que proporciona l'objecte **Roun** per l'event **bossDied**. La seva funció és incrementar el nombre de Round, per passar a la següent ronda.



2 Augmentar la dificultat del joc

Per modificar la dificultat del joc, augmentarem la velocitat de desplaçament dels enemics senzills (**Enemy**) cada vegada que passem a la ronda següent.

Podem utilitzar el nombre de Ronda (**currRound**) per incrementar d'alguna manera **walkingSpeed** de l'enemic, definit a l'script **EnemyControllerScript.**

Afegim el següent codi al final del mètode **Start()** de **EnemyControllerScript.**

```
// Obtenir nombre de ronda actual
GameObject roundWatcherObject =
      GameObject.FindGameObjectWithTag("RoundWatcher");

if (roundWatcherObject != null)
{
      RoundWatcher roundWatcherComponent =
                  roundWatcherObject.GetComponent<RoundWatcher>();
      // Assignar velocitat segons ronda en curs
      walkingSpeed = walkingSpeed * roundWatcherComponent.currRound;
}
```

Comentaris al codi:

- Recordem que el mètode **Start()** s'executa una sola vegada quan apareix en escena l'objecte. És un lloc adequat per assignar la velocitat del tanc que s'acaba de crear.

- El número de ronda es troba a la propietat **currRound** del component **RoundWatcher()** de l'objecte **Round**. Per obtenir aquesta dada, primer de tot recuperem l'objecte **Round** utilitzant el mètode **FindGameObjectWithTag**, al qual indiquem el tag de l'objecte que busquem. Previament, hem de crear el tag **RoundWatcher** i assignar-lo a l'objecte **Round**.
  Si hem trobat l'objecte (`roundWatcherObject != null`), obtenim el seu component **RoundWatcher()** amb el mètode **GetComponent** i, finalment, utilitzem **currRound**.

S'ha de buscar un valor inicial de velocitat, p.e. 0.25, que permeti anar-la pujant sense arribar de seguida a valors excessius. També es pot fer més gradual l'augment de velocitat, p.e.

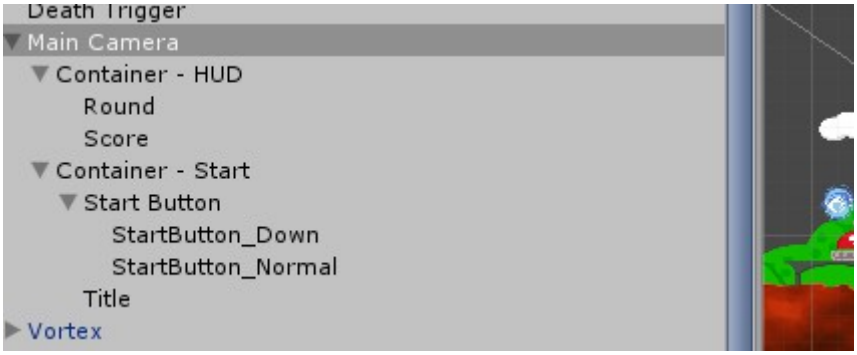
```
walkingSpeed = walkingSpeed + walkingSpeed * (roundWatcherComponent.currRound - 1)/ 5;
```

3 Crear la pantalla incial del joc

Tal com s'ha fet per implementar el marcador i l'indicador de rondes, crearem la pantalla d'inici amb objectes de **Unity**, sense utilitzar el sistema nadiu de generació de **GUI**s. Fins fa poc, aquest sistema era molt incòmode d'utilitzar i generava interfícies molt ineficients per la quantitat desmesurada de crides de rendering. El sistema de generació de **GUI** que ve amb les últimes versions de **Unity** sembla haver corregit els antics defectes, però en aquest document encara no l'utilitzarem.

Creem dos objectes contenidors, fills de **MainCamera**, **Container–HUD** i **Container-Start** que situem a X:0, Y:0, Z:0. Els objectes actuals **Score** i **Round** els col·loquem dins de **Container–HUD**.

Dins de Textures creem la carpeta **Title Screen** i hi importem els sprites **Title.png**, **StartButton_Normal.png** i **StartButton_Down.png**. Els situem sota **Container-Start**, i els botons els agrupem en un nou objecte **StartButton**:



A l'objecte **Title** li assignem la posició i escala que ens agradi i un **Order in Layer** = 100, perquè es visualitzi per devant de tot.

Els objectes **StartButton_Down** i **StartButton_Normal** tindran sempre la mateixa localització perquè, de cara a l'usuari, representen dos estats diferents (polsat i no polsat) d'un sol botó (botó Start). Per això han de tenir la mateixa position i scale.

A l'objecte **StartButton** li assignem un **BoxCollider2D** que ens permetrà capturar events del mouse. Ens hem d'assegurar que **StartButton** i el seu collider envolten completament els botons amb imatge **StartButton_Down** i **StartButton_Normal**.

Hem d'evitar que la resta d'objectes del joc col·lisionin amb els nous objectes de la **UI**. Per això creem un nou layer, que anomenem UI, i hi assignem tots els objectes de que hem creat per implementar la interface. Finalment, **editem la matriu de col·lisions** (Edit > Project Settings > Phisics 2D) i indiquem que els elements de la capa UI no col·lisionen amb les altres capes.

L'aspecte del joc en aquest moment serà semblant a

Ara, per fer-ho funcionar, afegim un nou script associat a la **MainCamera**, que anomenem **GameStates**.

```
using UnityEngine;
using System.Collections;

public class GameStates : MonoBehaviour
{
	public GameObject hudContainer;
	public GameObject titleContainer;
	public static bool gameActive = false;

	public enum displayStates
	{
		titleScreen = 0,
		hudScreen
	}
```

```
	void Start()
	{
		changeDisplayState(displayStates.titleScreen);
	}

	public void changeDisplayState(displayStates newState)
	{
		hudContainer.SetActive(false);
		titleContainer.SetActive(false);

		switch(newState)
		{
			case displayStates.titleScreen:
				gameActive = false;
				titleContainer.SetActive(true);
			break;

			case displayStates.hudScreen:
				gameActive = true;
				hudContainer.SetActive(true);
			break;
		}
	}

	public void startGame()
	{
		changeDisplayState(displayStates.hudScreen);
	}
}
```

Segons l'estat (de **displayStates**), es visualitzaran els element d'UI que mostren les puntuacions etc (hud: heads-up display) o els elements de pantalla inicial. El booleà **gameActive** indica si estem jugant o no.

Ara falta implementar el comportament del botó Start. Creem un nou script **StartButtonController** que assignem a l'objecte **StartButton**. El codi és:

```
using UnityEngine;
using System.Collections;

public class StartButtonController : MonoBehaviour
{
	public GameObject upSprite;
	public GameObject downSprite;
	public float downTime = 0.1f;
	public GameStates stateManager = null;

	private enum buttonStates
	{
		up = 0,
		down
	}

	private buttonStates currentState = buttonStates.up;
	private float nextStateTime = 0.0f;

	void Start()
	{
		upSprite.SetActive(true);
```

```
            downSprite.SetActive(false);
      }

      void OnMouseDown()
      {
            if(nextStateTime == 0.0f
               &&
               currentState == StartButtonController.buttonStates.up)
            {
                  nextStateTime = Time.time + downTime;
                  upSprite.SetActive(false);
                  downSprite.SetActive(true);
                  currentState = StartButtonController.buttonStates.down;
            }
      }

      void Update()
      {
            if(nextStateTime > 0.0f)
            {
                  if(nextStateTime < Time.time)
                  {
                        // Retornar el botó a estat "no polsat"
                        nextStateTime = 0.0f;
                        upSprite.SetActive(true);
                        downSprite.SetActive(false);
                        currentState = StartButtonController.buttonStates.up;

                        // Començar el joc
                        stateManager.startGame();
                  }
            }
      }
}
```

Downtime és el temp que esperem per reaccionar a la polsació del botó.

Veure **http://docs.unity3d.com/ScriptReference/MonoBehaviour.OnMouseDown.html**

Com sempre, les variables públiques cal informar-les des de l'**Object Inspector** de **Unity**.

A partir d'ara quan arrenquem el joc veurem els nous objectes de la **GUI** i, quan cliquem sobre el botó Start, veurem el joc com estem acostumats.

Hi ha un petit detall que podem arreglar: el **Player** es pot moure en la pantalla inicial, quan el joc encara no ha començat. Per evitar-ho, en el script **PlayerStateController**, al començament del mètode **LateUpdate()** afegim la línia:

```
      if(!GameStates.gameActive) return;
```

A partir d'ara, el joc no admet cap input abans d'estar actiu.
Podriem aconseguir el mateix d'altres maneres: esperar a l'inici del joc per generar el Player i els Enemy o "parar el temps" posant Timescale = 0 (veure **http://docs.unity3d.com/ScriptReference/Time-timeScale.html**)

Activitats

1. Fer un llistat amb els conceptes introduits en aquest document, indicar la pàgina i explicar què són i per a què s'utilitzen.