SESSIÓ 5.

Lluitar contra l'enemic final: "final boss battle"

En els videojocs un **Enemic Final** (en anglès: **Boss**) és un personatge particularment desafiador que s'ha de ser vèncer al final del joc o en acabar-ne un segment, ja sigui un nivell o un episodi. En molts jocs, la batalla contra el **Boss** és realment un minijoc dins del joc, que requereix un esforç considerable de codificació.

En el joc d'exemple que estem treballant, implementarem un **Boss** senzill. Caurà del cel sobre una plataforma quan es doni alguna circumstància concreta (p.e. Haver eliminat 10 enemics). El **Player** podrà disparar contra el **Boss**, que al seu torn podrà anar-se canviant de plataforma, esclafant tot el que hi trobi a sobre (fins i tot el **Player**). Quan el **Boss** hagi rebut un nombre suficient de trets, serà vençut.

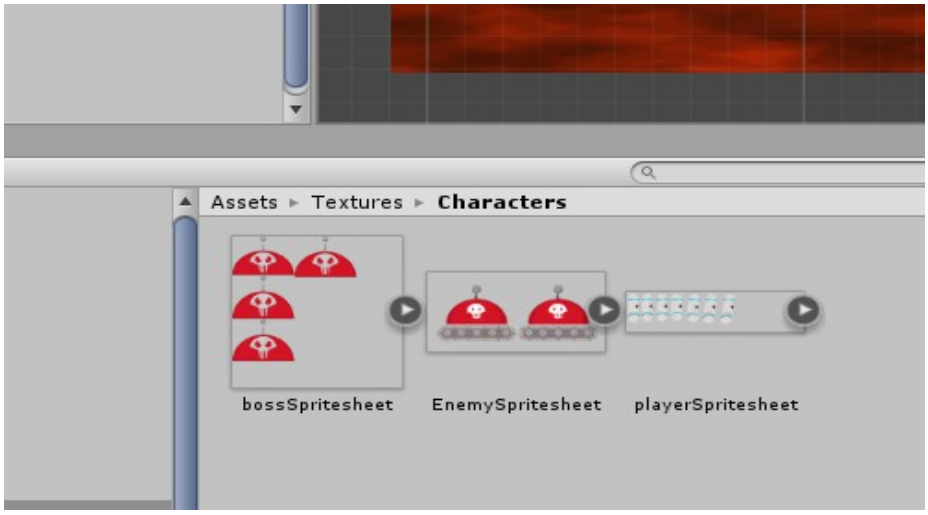
L'aspecte gràfic del Boss és:

Podem trobar el seu sprite sheet en el fitxer **bossSpritesheet.png**.

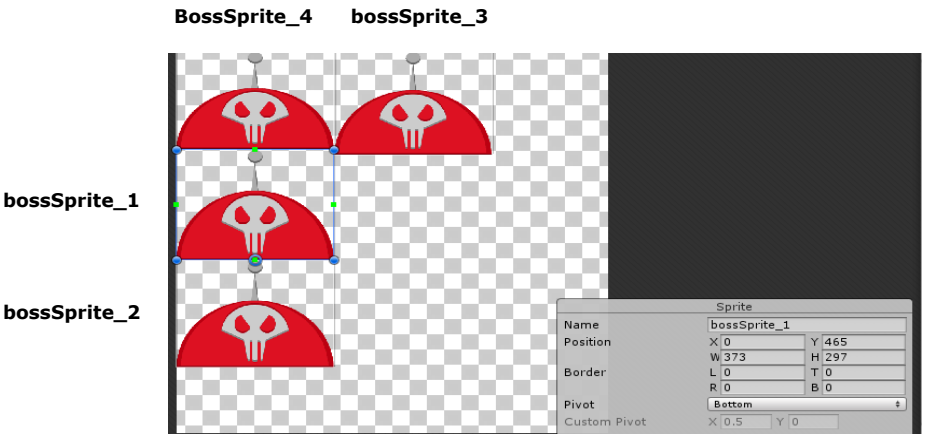
1 Introduir el nou personatge en el joc

Repetim el procés que ja coneixem per afegir un personatge al joc.

- Importem el **bossSpritesheet**,

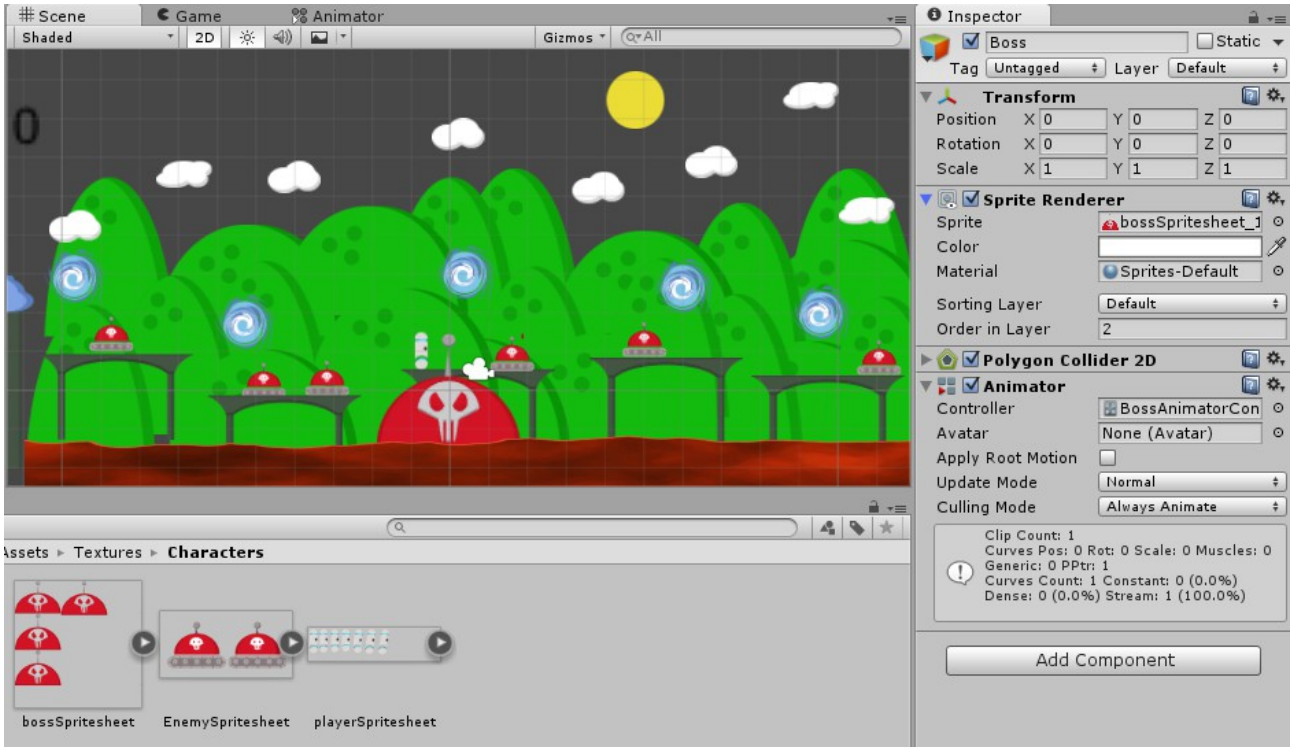


retallem les imatges amb l'**SpriteEditor** i les anomenem **bossSprite_1, bossSprite_2**, **bossSprite_3** i **bossSprite_4**.



- Creem un nou objecte al qual assignem el primer sprite de l'sprite sheet i l'anomenem **Boss**.
- Associem a l'objecte **Boss** un **Pollygon Collider 2D**.
- Creem una animació, que anomenem **BossAnimation**, amb els diferents sprites. Es tracta d'aplicar al personatge un moviment d'inflar-se i desinflar-se. Per això es proposa com a seqüència d'sprites **bossSprite_1**, **bossSprite_2**, **bossSprite_3**, **bossSprite_4**, **bossSprite_3**, **bossSprite_2**.

Per fer tot això és aconsellable revisar el que es va fer amb el **Player** i l'**Enemy**, per no oblidar cap detall.

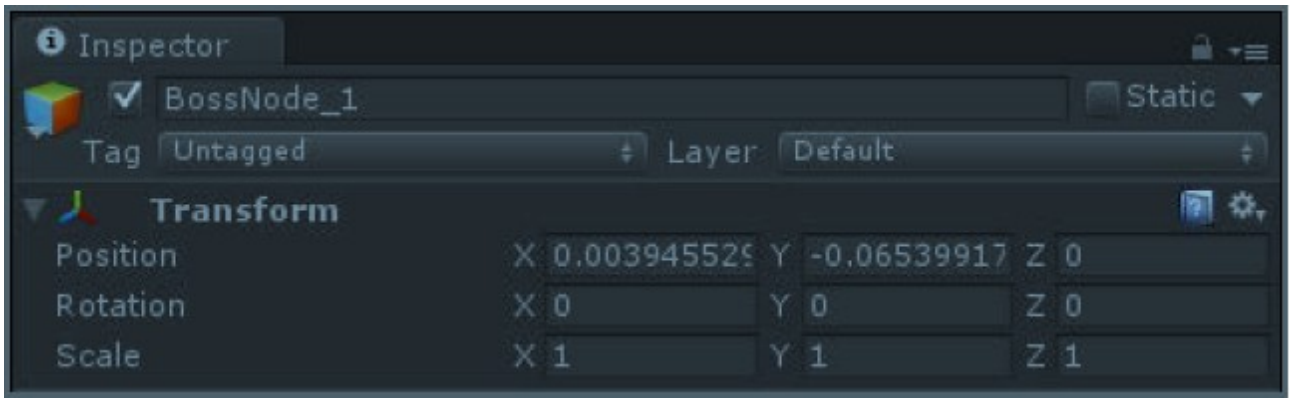


Podem canviar la posició de l'objecte Boss i situar-lo ben amunt, perquè no es vegi...

2 Objectes auxiliars per implementar els salts

Ja s'ha vist que una manera de fer saltar un objecte del joc és utilitzar un component **Rigidbody2D**, aplicar una força i deixar que el motor de físiques actui.

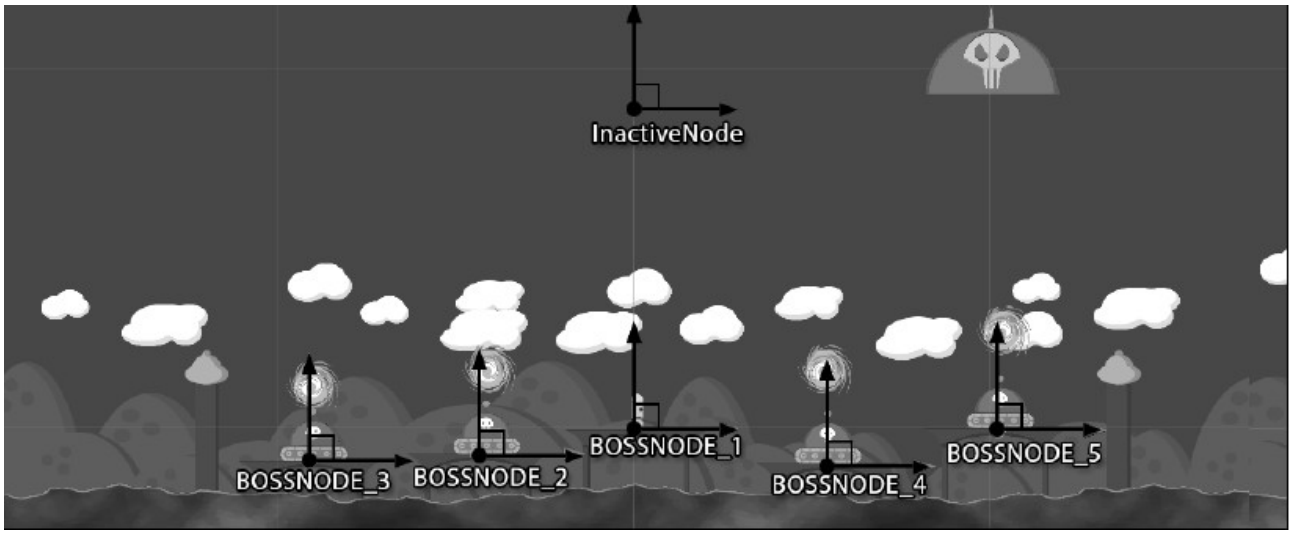
En aquest cas, com que volem que el **Boss** es situi exactament en llocs concrets ho implementarem d'una manera més manual. Crearem diferents objectes buits (que anomenarem nodes: **BossNode_1**, **BossNode_2** ...) i els situarem en els llocs que podrà ocupar el **Boss** al llarg del joc. Així, un node qualsevol serà un **GameObject** que només tindrà un component **Transform**.



Aquests objectes els carregarem, des de l'object inspector, en un objecte **List** (una de les implementacions de Col·lecció que ofereix C#).

A més dels objectes **BossNode_x**, definim un altre "node" auxiliar que anomenem **InactiveNode**, per situar-hi inicialment el **Boss**. L'**InactiveNode** haurà d'estar sobre el **BossNode_x** on apareixerà el **Boss** per primera vegada. El situarem a una alçada suficient perquè, mentre es juga, no es vegi el **Boss** esperant per aparèixer.

Si posem els objectes **BossNode_x** sobre les plataformes, la situació serà semblant a



3 Implementar el comportament del Boss

Per construir el comportament del **Boss** implementem l'script **BossEventController** (el codi del qual és més endavant) i li associem.

Descripció de les funcions de l'script **BossEventController**

F1. El **Boss** espera, situat a l'**InactiveNode**, fins que s'han matat 10 enemics. Per saber quan passa això, escolta l'event **enemyDied**.

F2. Quan el **Boss** apareix en el joc, baixa des de l'**InactiveNode** fins el node que s'ha escollit com a inicial (i que s'indica a **dropToStartNode**).

F3. Quan el **Boss** aterra sobre un node s'hi espera un temps (segons una temporització) i mentrestant pot rebre els trets del **Player**. Quan acaba l'espera, escull un altre node i s'hi desplaça, caient-hi a sobre.

La gestió dels canvis de plataforma es realitza considerant quatre estats possibles del **Boss**: waitingToFall, fallingToNode, waitingToJump i jumpingOffPlatform. Aquests estats es defineixen en una enumeració.

F4. Quan el **Boss** aterra sobre una plataforma, esclafa ("crushes") tots els personatges que hi trobi. (La gestió del "crush" es detalla més endevant).

F5. El **Boss** pot resistir un cert nombre de trets, segons un nivell inicial de vida o salut (al codi la variable **health**, inicialitzada a **startHealth**). Cada vegada que rep un tret, aquest valor es decrementa. Per gestionar això, escolta l'event **hitByBullet**, tal com fa **l'Enemy** senzill, utilitzant l'script **TakeDamageFromPlayerBullet** (revisar els detalls de com es fa a l'objecte **Enemy** i implementar-ho de manera semblant).

F6. Quan s'esgota la vida del **Boss**, mor. Això suma 1000 punts addicionals al marcador. La mort del **Boss** s'implementa de manera diferent a com ho feiem amb l'**Enemy** senzill. Ara no destruim l'objecte, sino que l'amaguem situant-lo altra vegada al node **InactiveNode**. Quan es donin les condicions necessàries, el **Boss** tornarà a aparèixer.

Codi de l'script **BossEventController**

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic; // Necesari per utilitzar llistes


public class BossEventController : MonoBehaviour
{
	public delegate void bossEventHandler(int scoreMod);
		public static event bossEventHandler bossDied;

	public delegate void bossAttack();
		public static event bossAttack crushPlayer;

	public GameObject inActiveNode = null;
	public GameObject dropToStartNode = null;
	public GameObject dropFXSpawnPoint = null;
	public List<GameObject> dropNodeList = new List<GameObject>();
```

```
public GameObject bossDeathFX = null;
public GameObject bossDropFX = null;
public TakeDamageFromPlayerBullet bulletColliderListener = null;

public float moveSpeed = 0.1f;
public float eventWaitDelay = 3f; // Temps d'espera entre events del Boss

public int enemiesToStartBattle = 10;

public enum bossEvents
{
	inactive = 0,
	fallingToNode,
	waitingToJump,
	waitingToFall,
	jumpingOffPlatform
}

// Estat actual del Boss
public bossEvents currentEvent = bossEvents.inactive;

// Node cap al qual saltara el Boss.
private GameObject targetNode = null;

// Temps a esperar fins el proper salt o caiguda.
private float timeForNextEvent = 0.0f;

// Posició de desti quan es salta a la plataforma.
private Vector3 targetPosition = Vector3.zero;

// Nivell de vida del Boss
public int health = 20;

//  Nivell de vida inicial del Boss
private int startHealth = 20;

// Indicador de si s'ha matat el Boss
private bool isDead = false;

// Enemics que s'han de matar abans que apareixi el Boss
private int enemiesLeftToKill = 0;

// Inicialitzacions. Apuntar-se a escoltar events indicant mètode per fer-ho
void OnEnable()
{
     bulletColliderListener.hitByBullet += hitByPlayerBullet;
      EnemyControllerScript.enemyDied += enemyDied;
}

void OnDisable()
{
	bulletColliderListener.hitByBullet -= hitByPlayerBullet;
	EnemyControllerScript.enemyDied -= enemyDied;
}

void Start()
{
	transform.position = inActiveNode.transform.position;
	enemiesLeftToKill = enemiesToStartBattle;
}

void Update()
{
	switch(currentEvent)
	{
		case bossEvents.inactive:
			// Not doing anything, so nothing to do.
		break;
```

```
                  case bossEvents.fallingToNode:
                        if(transform.position.y > targetNode.transform.position.y)
                        {
                              // Velocitat negativa, per desplaçar-se cap abaix
                              transform.Translate(new Vector3(0f, -moveSpeed *
Time.deltaTime, 0f));
                              if(transform.position.y <
targetNode.transform.position.y)
                              {
                                    Vector3 targetPos =
targetNode.transform.position;
                                    transform.position = targetPos;
                              }
                        }
                        else
                        {
                              // Crear efecte d'aterratge (Partícules)
                              createDropFX();

                              timeForNextEvent = 0.0f;
                              currentEvent = bossEvents.waitingToJump;
                        }
               break;

                        case bossEvents.waitingToFall:
                    // Boss esperant per caure en un altre node
                    if(timeForNextEvent == 0.0f)
                    {
                         timeForNextEvent = Time.time + eventWaitDelay;
                    }
                              else if(timeForNextEvent < Time.time)
                    {
                         // Need to find a new node!
                         targetNode = dropNodeList[ Random.Range(0,dropNodeList.Count) ];

                         // Posició del Boss SOBRE el node destí
                         transform.position = getSkyPositionOfNode(targetNode);

                         // actualitzar estat
                         currentEvent = bossEvents.fallingToNode;
                         timeForNextEvent = 0.0f;
                    }
               break;

               case bossEvents.waitingToJump:
                    // Boss espera, situat sobre plataforma, el moment de canviar
                    if(timeForNextEvent == 0.0f)
                    {
                         timeForNextEvent = Time.time + eventWaitDelay;
                    }
                              else if(timeForNextEvent < Time.time)
                    {
                         // Estableix posició objetiu per elevar-se sobre node actual
                         targetPosition = getSkyPositionOfNode(targetNode);

                         // Actualitzar estat
                         currentEvent = bossEvents.jumpingOffPlatform;
                         timeForNextEvent = 0.0f;

                         targetNode = null;
                    }
               break;

               case bossEvents.jumpingOffPlatform:
                    if(transform.position.y < targetPosition.y)
                    {
                                   // Velocitat positiva, moviment ascendent
```

```
                                        transform.Translate(new Vector3(0f, moveSpeed *
Time.deltaTime, 0f));

                        if(transform.position.y > targetPosition.y)
                                        transform.position = targetPosition;
                                }
                    else
                    {
                        timeForNextEvent = 0.0f;
                        currentEvent = bossEvents.waitingToFall;
                    }
                break;
            }
        }

        public void beginBossBattle()
        {
            // Establir node inicial i fer que el Boss hi caigui
            targetNode = dropToStartNode;
            currentEvent = bossEvents.fallingToNode;

            // Inicialitzar variables de control
            timeForNextEvent = 0.0f;
            health = startHealth;
            isDead = false;
        }

        Vector3 getSkyPositionOfNode(GameObject node)
        {
            Vector3 targetPosition = targetNode.transform.position;
            targetPosition.y += 9f;

            return targetPosition;
        }

        void hitByPlayerBullet()
        {
            health -= 1;

            // Si s'ha acabat la vida, el matem
            if(health <= 0)
                killBoss();
        }

        void createDropFX()
        {
          //Implementar sistema de partícules de caiguda sobre plataforma: bossDropFX
         // . . . .
        }

        void killBoss()
        {
                if(isDead)
                      return;

                isDead = true;
                //Implementar sistema de partícules de destrucció del Boss: bossDeathFX

                // Generar l'event "bossDied" amb una puntuació de 1000 punts
                if(bossDied != null)
                      bossDied(1000);

                // Tornar a posició inactiva inicial
                transform.position = inActiveNode.transform.position;

                //Reset de camps de control
                currentEvent = BossEventController.bossEvents.inactive;
                timeForNextEvent = 0.0f;
```

```
                enemiesLeftToKill = enemiesToStartBattle;
        }

        void enemyDied(int enemyScore)
        {
                if(currentEvent == bossEvents.inactive)
                {
                        enemiesLeftToKill -= 1;
                        Debug.Log("--- Enemics pendents per apareixer Boss: " +
enemiesLeftToKill);
                        if(enemiesLeftToKill <= 0)
                                beginBossBattle();
                }
        }

        public void playerHitByCrusher()
        {
                if(currentEvent == bossEvents.fallingToNode)
                {
                        if(crushPlayer != null)
                                crushPlayer();
                }
        }
}
```

Últims comentaris sobre l'script:

1. Perquè s'actualitzi el marcador quan mori el Boss, hem de fer que a l'script **ScoreWatcher** també s'escolti l'event bossDied. Per això completem els mètodes OnEnable() i OnDisable() amb les línies

```
BossEventController.bossDied += addScore;
```

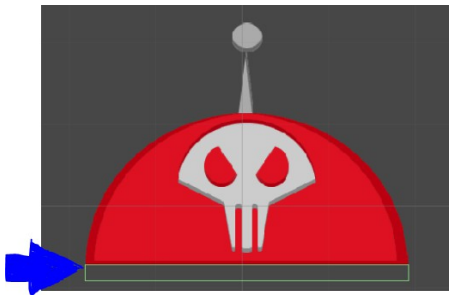
i

```
BossEventController.bossDied -= addScore;
```

respectivament.

2. Falta implementar els sistemes de partícules que es generen quan el Boss aterra sobre una plataforma i quan és vençut (es proposa com a activitat 2)

3. Falta gestionar la destrucció dels personatges que hi ha sobre la plataforma quan el Boss hi aterra. Per fer-ho, creem un objecte buit **fill de l'objecte Boss**, que anomenem **BossCrushZone** i que només té un **BoxCollider2D** (trigger!). El situem a la part inferior del Boss:



Finalment li associem l'script **BossCrushTrigger**:

```
using UnityEngine;
using System.Collections;

public class BossCrushTrigger : MonoBehaviour
{
	public BossEventController bossController;

	void OnTriggerEnter2D( Collider2D collidedObject )
	{
		if(bossController.currentEvent !=
BossEventController.bossEvents.fallingToNode)
			return;

		if(collidedObject.tag == "Player")
			collidedObject.SendMessage("hitByCrusher");
	}
}
```

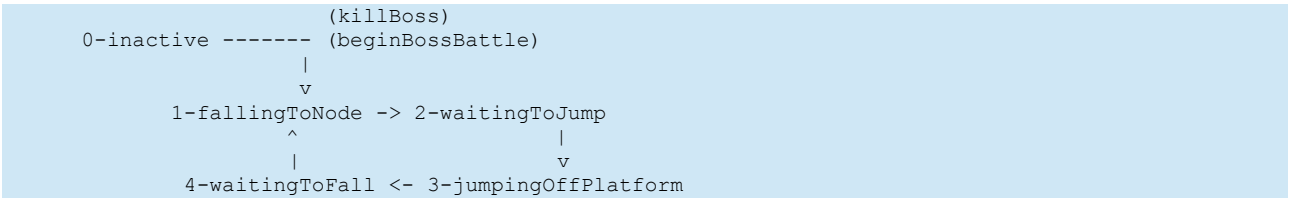
Perquè funcioni, hem de gestionar el tag del **Player** i programar el mètode **hitByCrusher()** a **PlayerStateListener.** Implementa la mort del **Player** passant-lo directament a estat Kill.

```
public void hitByCrusher()
{
	onStateChange(PlayerStateController.playerStates.kill);
}
```

Ens hem limitat a destuir el **Player**.

Activitats

1. Explicar el codi que implementa cada una de les funcions F1, F2, ... F6 de **BossEventController**



2. Implementar amb sistemes de partícules els efectes especials del Boss: aterrar sobre plataforma (amb la imatge **Particle_Smoke.png**) i destrucció (amb les imatges **Particle_EnemyDeath_1.png** i **Particle_EnemyDeath_2.png**).
Veure: http://singletechgames.com/2014/01/16/tutorial-en-espanol-de-unity-2d-6-version-4-3/

3. Implementar la destrucció dels enemics senzills quan el **Boss** aterra sobre una plataforma.

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