Programación
1º DAM

Proyecto Final

(Modelo de) Memoria

I.E.S. San Vicente
San Vicente del Raspeig (Alicante)
Curso 2018/2019

Alumno:
Sergio Ruescas

Profesor:
Nacho Cabanes

1. Introducción

Nombre del proyecto
Mario

Desarrollado por
Sergio Ruescas

Descripción breve del proyecto
El proyecto consiste en el juego arcade de Mario desarrollado en Unity.

2. Funcionalidad del proyecto
Al entrar al programa, aparecerá un pantalla de bienvenida, en la que el usuario podrá escoger entre:

- 1 Player
- 2 Players
- Scoreboards.
- Leave the game.

La partida consiste en sobrevivir a diferentes niveles donde aparecerán enemigos y bolas de fuego. Para derrotar a un enemigo hay que golpear a la plataforma donde está, y una vez haya "caido" habrá que "patearlos" para derrotarlos definitivamente, si no se les patea volverán con mas poder.

En ScoreBoards se podrá ver las mejores puntuaciones (Se guardarán en un fichero)

3. Prototipo de la pantalla
La apariencia que se persigue es ésta:

4. Entregas previstas

(**Nota:** Debes incluir al menos 15 entregas, de 2 horas de trabajo cada una)

1. Creación del script del enemigo (tortuga) Movimiento, “stun” al golpear la plataforma donde se encuentra y aumento de velocidad si sobreviven.
2. Creación del script del generador de enemigos y script de la bola de fuego.
3. Creacion del script del bloque POW y añadir al script del personaje las vidas, la muerte y la puntuación.
4. Creación de la pantalla de la demostración (tortuga) y la bola de fuego verde.
5. Modificar el script del enemigo (tortuga) para crear el prefab del enemigo (cangrejo) al que se le debe golpear dos veces en la plataforma.
6. Modificar el script del enemigo (tortuga) para crear el prefab del enemigo (mosca) al que saltará por lo que solo se le puede golpear cuando esté tocando la plataforma.
7. Creacion de la pantalla de la demostración del enemigo (cangrejo) y (mosca).
8. Modificacion del script de generador para implementarla en las diferentes fases
9. Crear un script para el control de las diferentes fases.
10. Modificaciones necesarias para el modo 2 jugadores
11. Pantalla para ver las puntuaciones cargados desde un fichero.
12. Pantalla para guardar la puntuación.
13. Pantalla del menu inicial y la animación de inicio del programa.
14. Añadir sonido de salto, muerte y música de fondo.
15. Script de unión de las diferentes escenas.

5. Trabajo diario realizado

- Creado el enemigo (tortuga), Se mueve, se da la vuelta al chocar con otro, y se puede matar.
- Creado el script de la bola de fuego y el script de el generador de enemigos. Se ha añadido la animación de quemado a los personajes.
- Creado el bloque Pow y su interaccion con los enemigos
- Creado la demostracion de la tortuga y arreglados errores pendientes de la entregas anteriores
- Creado el enemigo (cangrejo). Se mueve, se da la vuelta al chocar con otro y se puede matar.
- Creada parte del script de union entre escenas
- Clase ScoreBoard que maneja el fichero con las puntuaciones, terminado el script de union entre escenas y arreglado un bug de la bola de fuego que aun no estndo activa podías chocar con ella.
- Hecho script del enemigo (mosca) y del generador arreglado bug (No se desactivaban los colliders del jugador cuando moria por la bola de fuego) y casi terminado el prefab de la bola de fuego verde.
- Creado script para crear mensajes con sprites.
- Creada la demostracion del enemigo 2 (cangrejo), añadidos mensajes de explicacion en las demostraciones.
- Terminada la pantalla de inicio y comenzada la pantalla de puntuacion
- Terminada la pantalla de score y modificaciones para los dos jugadores
- Terminada la ultima escena con varios tipos de enemigos

6. Problemas encontrados durante el desarrollo y sus soluciones

- Durante la segunda entrega alguna modificación ha estropeado el funcionamiento de matar enemigos (Al golpearlo muere tanto el jugador como el enemigo)
  ARREGLO: Separar el collider de la cabeza en un gameobject diferente del cuerpo para poder diferenciar cuando mario ataca por debajo de la plataforma o por el frente.
- La bola de fuego al salir por un lado de la pantalla y al aparecer por el otro la posicion del collider no va concorde con la posicion del sprite provocando que puedra atravesar a otros objectos.
  ARREGLO: En el script de los colliders en el FixedUpdate detectar cuando no es igual la posicion del sprite y de los colliders y en ese caso actualizar su posicion.

Checklist

✔ if
- ✔ else
- ✔ Conectores: && y/o || y/o !

✔ switch

✔ ?
- ✔ while
- ✔ for
- ✔ foreach
- ✔ try-catch
- ✔ array

✔ struct
- ✔ (clases + herencia)
- ✔ (propiedades o getters y setters)
- ✔ public, protected, (opcional) private
- ✔ ArrayList o List<>
- ✔ HashTable o ShortedList o Dictionary
- ✔ StreamReader o FileStream o BinaryReader

✗ ref o out
- ✔ manejo avanzado de cadenas
- ✔ consola avanzada o SDL o Windows Form o Unity

OPCIONALES:
- ✗ Operaciones a nivel de bits
- ✗ Constantes
- ✔ Enumeraciones
- ✗ Interfaces, clases selladas

- xManejo de fechas
- xManejo de directorios
- xInformación del sistema
- xLINQ
- xExpresiones regulares
- xSerialización
- xConexión a bases de datos

7. Mejoras o restricciones respecto a la idea inicial

-

8. Capturas de pantalla del proyecto final

(…)


PRESS SPACE TO START
MARIO BROS.
ONE PLAYER
TWO PLAYERS
SCOREBOARD
LEAVE
PRESS C TO INSERT COIN
COINS 0

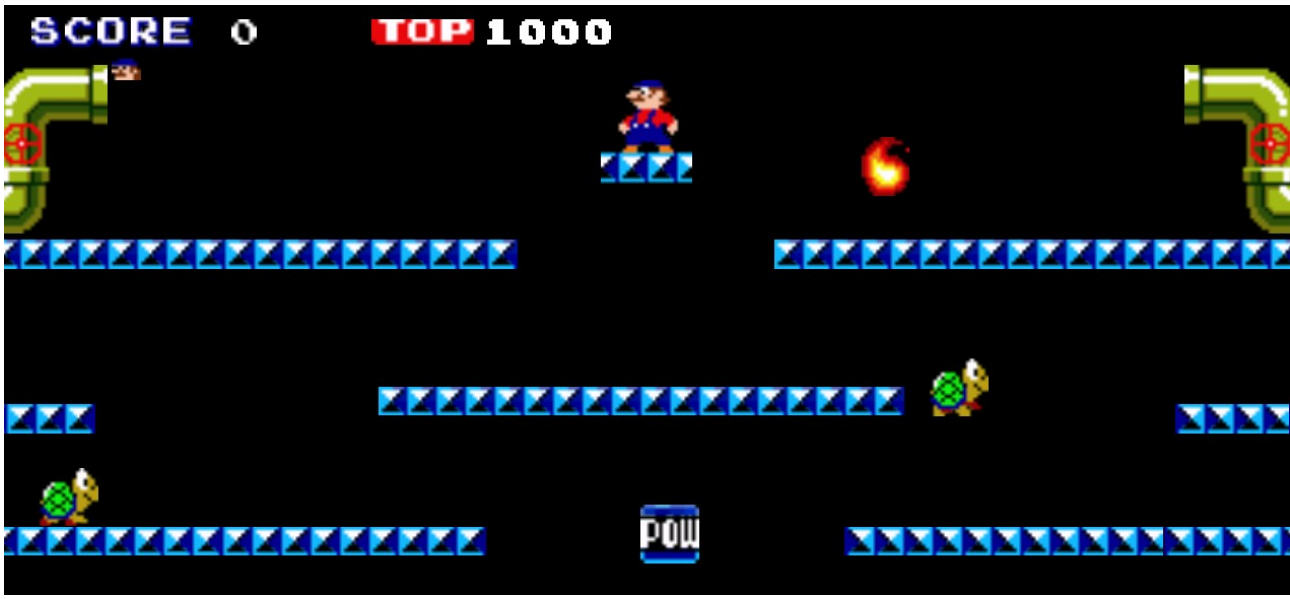
9. Código fuente del proyecto final

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ColliderDetectDeath : MonoBehaviour
{

    void OnTriggerEnter2D(Collider2D trigger)
    {
        if (trigger.tag == "PlayerHead" )
        {
            transform.parent.SendMessage("StunController");
        }
    }
}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ColliderX : MonoBehaviour
{
    Transform trans;
```




```
    void Start()
    {
        trans = GetComponent<Transform>();
    }

    void Update()
    {
```

```
    }

    void FixedUpdate()
    {
        if (trans.position.x != transform.parent.position.x ||
                trans.position.y != transform.parent.position.y)
            trans.position = new Vector2(0, 0);
    }

    void OnCollisionEnter2D()
    {
        transform.parent.SendMessage("CollideX");
    }
}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ColliderY : MonoBehaviour
{
    Transform trans;

    void Start()
    {
        trans = GetComponent<Transform>();
    }


    void Update()
    {

    }

    void FixedUpdate()
    {
        if (trans.position.x != transform.parent.position.x ||
                trans.position.y != transform.parent.position.y)
            trans.position = new Vector2(0, 0);
    }

    void OnCollisionEnter2D()
    {
        transform.parent.SendMessage("CollideY");
    }
}
using System.Collections;
using System.Collections.Generic;
using System.Threading;
using UnityEngine;
```

```
public class Demonstration1Scene : MonoBehaviour
{
    public GameObject enemy;
    public GameObject player;
    public MessageController messageDemonstration;
    public MessageController messageArrows;
    public MessageController messageJump;
    public MessageController messageInstruccions1;
    public MessageController messageInstruccions2;
    public MessageController messageInstruccions3;
    protected PlayerDemonstration playerScript;
    protected Enemy enemyScript;
    protected enum ScenePhase { Waiting, MovePlayer, JumpPlayer, Attack, Kill };
    protected ScenePhase scenePhase;

    void Start()
    {
        scenePhase = ScenePhase.Waiting;
        playerScript = player.GetComponent<PlayerDemonstration>();
        enemyScript = enemy.GetComponent<Enemy>();
        playerScript.SetIdle();
        messageDemonstration.SetMessage("demonstration");
        messageArrows.SetMessage("to move use left and right arrow");
        messageJump.SetMessage("to jump use espace");
        messageInstruccions1.SetMessage("one hit flip it over");
        messageInstruccions2.SetMessage("jump up");
        messageInstruccions3.SetMessage("kick off when upside down");
        messageArrows.Draw();
        messageJump.Draw();
        messageDemonstration.Draw();
    }


    void FixedUpdate()
    {
        if (Time.time > 4 && scenePhase == ScenePhase.Waiting)
        {
            scenePhase = ScenePhase.MovePlayer;
        }

        if (scenePhase == ScenePhase.MovePlayer)
        {
            enemy.SetActive(true);
            messageArrows.Hide();
            messageJump.Hide();
            playerScript.MoveLeft();
        }
        else if (scenePhase == ScenePhase.JumpPlayer)
        {
            if (playerScript.IsGrounded() && !enemyScript.GetStun()
```

```
            && enemy.GetComponent<Transform>().position.x < 2)
        {
            playerScript.Jump();
        }
        else if (enemyScript.GetStun())
        {
            messageInstruccions1.Draw();
            scenePhase = ScenePhase.Attack;
        }
        else
            playerScript.SetIdle();
    }
    else if (scenePhase == ScenePhase.Attack)
    {
        if (player.GetComponent<Transform>().position.x > -4.5f)
        {
            playerScript.MoveLeft();
        }
        else if (playerScript.IsGrounded())
        {
            messageInstruccions1.Hide();
            messageInstruccions2.Draw();
            playerScript.Jump();
        }
        else
        {
            scenePhase = ScenePhase.Kill;
        }
    }
    else if (enemyScript.GetStun())
    {
        messageInstruccions3.Draw();
        playerScript.MoveRight();
    }


    if (player.GetComponent<Transform>().position.x < -1f
            && scenePhase == ScenePhase.MovePlayer)
        scenePhase = ScenePhase.JumpPlayer;
  }

}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Demonstration2Scene : MonoBehaviour
{
  public GameObject enemy;
```

```
public GameObject player;
public MessageController messageDemonstration;
public MessageController messageInstruccions1;
public MessageController messageInstruccions2;
public MessageController messageInstruccions3;
protected PlayerDemonstration playerScript;
protected Enemy2 enemyScript;
protected enum ScenePhase {Waiting, Level2, Level1, MovePlayer, Kill};
protected ScenePhase scenePhase;

void Awake()
{
   scenePhase = ScenePhase.Waiting;
}
void Start()
{
   playerScript = player.GetComponent<PlayerDemonstration>();
   enemyScript = enemy.GetComponent<Enemy2>();
   messageDemonstration.SetMessage("demonstration");
   messageInstruccions1.SetMessage("first hit makes it mad");
   messageInstruccions2.SetMessage("second hit makes it over");
   messageInstruccions3.SetMessage("kick off when upside down");
   playerScript.SetIdle();
   messageDemonstration.Draw();
}


void FixedUpdate()
{
   ControllePlayer();

   ChangeScenePhase();
}

private void ControllePlayer()
{
   float posXEnemy = enemy.GetComponent<Transform>().position.x;
   float posYPlayer = player.GetComponent<Transform>().position.y;
   if (scenePhase == ScenePhase.Level2)
   {
      enemy.SetActive(true);
      if(posXEnemy < 4)
      {
         playerScript.Jump();
         messageInstruccions1.Draw();
      }
   }
   else if (scenePhase == ScenePhase.MovePlayer)
   {
      playerScript.MoveLeft();
```

```
        }
        else if (scenePhase == ScenePhase.Level1)
        {
            messageInstruccions1.Hide();
            messageInstruccions2.Draw();
            playerScript.Jump();
        }
        else if (scenePhase == ScenePhase.Kill)
        {
            if(posYPlayer  < -1.5)
            {
                playerScript.Jump();
                messageInstruccions2.Hide();
                messageInstruccions3.Draw();
            }

            playerScript.MoveRight();
        }
    }

    private void ChangeScenePhase()
    {
        float posXPlayer = player.GetComponent<Transform>().position.x;

        if (scenePhase == ScenePhase.Waiting)
        {
            scenePhase = ScenePhase.Level2;
        }
        else if (enemyScript.GetLevel() < 2 && scenePhase == ScenePhase.Level2)
        {
            scenePhase = ScenePhase.MovePlayer;
        }
        else if (posXPlayer < 0 && posXPlayer > -0.6 &&
                    scenePhase == ScenePhase.MovePlayer)
        {
            scenePhase = ScenePhase.Level1;
        }
        else if (enemyScript.GetLevel() < 1 && scenePhase == ScenePhase.Level1)
        {
            scenePhase = ScenePhase.MovePlayer;
        }
        else if (posXPlayer < -4)
        {
            scenePhase = ScenePhase.Kill;
        }

    }
}
using System.Collections;
using System.Collections.Generic;
```

```
using UnityEngine;

public class Demonstration3Scene : MonoBehaviour
{
    public GameObject enemy;
    public GameObject player;
    public MessageController messageDemonstration;
    public MessageController messageInstruccions1;
    public MessageController messageInstruccions2;
    public MessageController messageInstruccions3;
    protected PlayerDemonstration playerScript;
    protected Enemy3 enemyScript;
    protected enum ScenePhase { Waiting, Jump, MovePlayer, Stunned, Kill };
    protected ScenePhase scenePhase;
    protected bool changePhase;

    void Awake()
    {
        scenePhase = ScenePhase.Waiting;
    }

    void Start()
    {
        changePhase = false;
        playerScript = player.GetComponent<PlayerDemonstration>();
        enemyScript = enemy.GetComponent<Enemy3>();
        messageDemonstration.SetMessage("demonstration");
        messageInstruccions1.SetMessage("one hit flip it over");
        messageInstruccions2.SetMessage("only when touching floor");
        messageInstruccions3.SetMessage("kick off when upside down");
        playerScript.SetIdle();
        messageDemonstration.Draw();
    }


    void FixedUpdate()
    {
        ControllePlayer();

        ChangeScenePhase();
    }

    private void ControllePlayer()
    {
        float posXEnemy = enemy.GetComponent<Transform>().position.x;
        float posYPlayer = player.GetComponent<Transform>().position.y;
        float posXPlayer = player.GetComponent<Transform>().position.x;
        if (scenePhase == ScenePhase.Jump)
        {
            if (posXEnemy < 2.3)
```

```
        {
            if(posXEnemy > 2)
                playerScript.Jump();
            if(posXEnemy < 1)
                changePhase = true;
        }
    }
    else if (scenePhase == ScenePhase.MovePlayer)
    {
        if (posXEnemy < -1.2)
        {
            playerScript.Jump();
            messageInstruccions1.Draw();
            messageInstruccions2.Draw();
            if(enemy.GetComponent<Enemy3>().GetStun())
                changePhase = true;
        }
        else if (posXEnemy > -0.7)
        {
            playerScript.MoveLeft();
        }
        else
            playerScript.SetIdle();
    }
    else if(scenePhase == ScenePhase.Stunned)
    {
        if(posXPlayer > -5)
        {
            playerScript.MoveLeft();
        }
        else
        {
            changePhase = true;
        }
    }
    else if (scenePhase == ScenePhase.Kill)
    {
        if (posYPlayer < -1.5)
        {
            playerScript.Jump();
            messageInstruccions2.Hide();
            messageInstruccions3.Draw();
        }
        playerScript.MoveRight();
    }
}

private void ChangeScenePhase()
{
    float posXPlayer = player.GetComponent<Transform>().position.x;
```

```
        if (scenePhase == ScenePhase.Waiting)
        {
            enemy.SetActive(true);
            scenePhase = ScenePhase.Jump;
        }
        else if(scenePhase == ScenePhase.Jump && changePhase)
        {
            scenePhase = ScenePhase.MovePlayer;
            changePhase = false;
        }
        else if(scenePhase == ScenePhase.MovePlayer && changePhase)
        {
            scenePhase = ScenePhase.Stunned;
            changePhase = false;
        }
        else if(scenePhase == ScenePhase.Stunned && changePhase)
        {
            scenePhase = ScenePhase.Kill;
            changePhase = false;
        }
    }
}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class Enemy : MonoBehaviour
{
    protected float speed;
    protected float limitSpeed;
    bool isGrounded;
    bool isStunned;
    bool isDeath;
    bool isTurning;
    int level;
    EnemyGenerator1 enemyGenerator;
    MultiEnemyGeneratorController multiEnemyGenerator;
    Animator anim;
    Collider2D col2D;
    public enum Direction {Left, Right};
    protected Direction direction;
    protected Transform transformEnemy;
    protected SpriteRenderer sprite;

    void Awake()
    {
        limitSpeed = 0.11f;
        speed = 0.05f;
```

```
    level = 1;

    transformEnemy = GetComponent<Transform>();
    anim = GetComponentInChildren<Animator>();
    sprite = GetComponentInChildren<SpriteRenderer>();
    col2D = GetComponent<Collider2D>();
    enemyGenerator = FindObjectOfType<EnemyGenerator1>();
    multiEnemyGenerator =
        FindObjectOfType<MultiEnemyGeneratorController>();
}
// Start is called before the first frame update
void Start()
{
    isStunned = false;
    isDeath = false;
    isTurning = false;
    if (transformEnemy.position.x < 0)
        direction = Direction.Right;
    else
        direction = Direction.Left;
}

// Update is called once per frame
void Update()
{
    if(direction == Direction.Right)
    {
        sprite.flipX = true;
    }
    else
    {
        sprite.flipX = false;
    }


    if (isDeath)
    {
        anim.SetBool("Death", true);
    }
    else
    {
        anim.SetBool("Death", false);
    }

    if (isStunned)
    {
        anim.SetBool("Stun", true);
    }
    else
    {
```

```
            anim.SetBool("Stun", false);
        }

        if(isTurning)
        {
            anim.SetBool("Turn", true);
        }
        else
        {
            anim.SetBool("Turn", false);
        }
    }

    void FixedUpdate()
    {
        if (isStunned == false && isTurning == false)
        {
            if(direction == Direction.Right)
            {
                transformEnemy.position = new Vector2(
                    transformEnemy.position.x + speed,
                        transformEnemy.position.y);
            }
            else
            {
                transformEnemy.position = new Vector2(
                    transformEnemy.position.x - speed,
                        transformEnemy.position.y);
            }
        }
        else if (isDeath)
        {
            transformEnemy.position = new Vector2(
                transformEnemy.position.x, transformEnemy.position.y-0.07f);
            col2D.enabled = false;
        }

        if ((transformEnemy.position.x < -8.8
                || transformEnemy.position.x > 8.8)
                && isTurning == false)
        {
            if (transformEnemy.position.y < -2.5)
            {
                if (multiEnemyGenerator == null)
                    enemyGenerator.CreateEnemy(this.level, this.speed, 1);
                else
                    multiEnemyGenerator.CreateEnemy(this.level, this.speed, 1);
                Destroy(gameObject);
            }
            else if (transformEnemy.position.x < 0)
```

```
            transformEnemy.position = new Vector2(
                8.7f, transform.position.y);
        else
            transformEnemy.position = new Vector2(
                -8.7f, transform.position.y);
    }
    else if (transformEnemy.position.y < -5.5)
    {
        Destroy(gameObject);
        if(SceneManager.GetActiveScene().name.StartsWith("Scene"))
            GameSceneController.EnemyKilled(50);
        else
            GameSceneController.EnemyKilled(0);
    }

}

void OnTriggerStay2D(Collider2D trigger)
{
    if (trigger.tag == "ground")
    {
        isGrounded = true;
    }

}

void OnTriggerEnter2D(Collider2D trigger)
{
    if (trigger.tag == "Player" && isGrounded && isStunned)
    {
        isDeath = true;
    }
    else if ((trigger.tag == "Enemy" && isTurning == false) ||
             (trigger.tag == "Player" && isStunned == false) ||
                (trigger.tag =="FireBall" && isTurning == false))
    {
        if(direction == Direction.Right)
        {
            direction = Direction.Left;
        }
        else
        {
            direction = Direction.Right;
        }
        isTurning = true;
    }
}

void OnTriggerExit2D(Collider2D trigger)
{
```

```
    if (trigger.tag == "ground")
    {
        isGrounded = false;
    }
}

public void LevelUp()
{
    speed += speed/2;
    if (speed > limitSpeed)
        speed = limitSpeed;
    isStunned = false;
    level++;
}

public void StunController()
{
    if (isGrounded && isStunned == false)
    {
        isStunned = true;
    }
    else if (isGrounded && isStunned)
    {
        isStunned = false;
        isTurning = false;
    }
}

public void EndTurn()
{
    isTurning = false;
}

public bool GetStun()
{
    return isStunned;
}

public bool GetTurn()
{
    return isTurning;
}

public void SetSpeed(float speed)
{
    this.speed = speed;
}

public void SetLevel(int level)
{
```

```
        this.level = level;
    }
}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class Enemy2 : MonoBehaviour
{
    protected float speed;
    protected float limitSpeed;
    bool isGrounded;
    bool isStunned;
    bool isDeath;
    bool isTurning;
    bool reduceLevel;
    int level;
    EnemyGenerator2 enemyGenerator;
    MultiEnemyGeneratorController multiEnemyGenerator;
    Animator anim;
    Collider2D[] col2D;
    public enum Direction { Left, Right };
    protected Direction direction;
    protected Transform transformEnemy;
    protected SpriteRenderer sprite;

    void Awake()
    {
        level = 2;
        speed = 0.05f;
        limitSpeed = 0.12f;

        transformEnemy = GetComponent<Transform>();
        anim = GetComponentInChildren<Animator>();
        sprite = GetComponentInChildren<SpriteRenderer>();
        col2D = GetComponents<Collider2D>();
        enemyGenerator = FindObjectOfType<EnemyGenerator2>();
        multiEnemyGenerator =
            FindObjectOfType<MultiEnemyGeneratorController>();
    }
    // Start is called before the first frame update
    void Start()
    {
        isStunned = false;
        isDeath = false;
        isTurning = false;
        reduceLevel = false;
        if (transformEnemy.position.x < 0)
            direction = Direction.Right;
```

```
    else
        direction = Direction.Left;
}

// Update is called once per frame
void Update()
{
    if (direction == Direction.Right)
    {
        sprite.flipX = true;
    }
    else
    {
        sprite.flipX = false;
    }


    if (isDeath)
    {
        anim.SetBool("Death", true);
    }
    else
    {
        anim.SetBool("Death", false);
    }

    if (isStunned)
    {
        anim.SetBool("Stun", true);
    }
    else
    {
        anim.SetBool("Stun", false);
    }

    if (isTurning)
    {
        anim.SetBool("Turn", true);
    }
    else
    {
        anim.SetBool("Turn", false);
    }

    if(reduceLevel)
    {
        anim.SetBool("Reduce", true);
    }
    else
    {
```

```
            anim.SetBool("Reduce", false);
        }
        anim.SetInteger("Level", level);
    }

    void FixedUpdate()
    {
        if (isStunned == false && isTurning == false && reduceLevel == false)
        {
            if (direction == Direction.Right)
            {
                transformEnemy.position = new Vector2(
                    transformEnemy.position.x + speed,
                        transformEnemy.position.y);
            }
            else
            {
                transformEnemy.position = new Vector2(
                    transformEnemy.position.x - speed,
                        transformEnemy.position.y);
            }
        }
        else if (isDeath)
        {
            transformEnemy.position = new Vector2(
                transformEnemy.position.x, transformEnemy.position.y - 0.07f);
            for (int i = 0; i < col2D.Length; i++)
            {
                col2D[i].enabled = false;
            }
        }

        if ((transformEnemy.position.x < -8.8
                || transformEnemy.position.x > 8.8)
                && isTurning == false)
        {
            if (transformEnemy.position.y < -2.5)
            {
                if (multiEnemyGenerator == null)
                    enemyGenerator.CreateEnemy(this.level, this.speed, 2);
                else
                    multiEnemyGenerator.CreateEnemy(this.level, this.speed, 2);
                Destroy(gameObject);
            }
            else if (transformEnemy.position.x < 0)
                transformEnemy.position = new Vector2(
                    8.7f, transform.position.y);
            else
                transformEnemy.position = new Vector2(
                    -8.7f, transform.position.y);
```

```
    }
    else if (transformEnemy.position.y < -5.5)
    {
        Destroy(gameObject);
        if (SceneManager.GetActiveScene().name.StartsWith("Scene"))
            GameSceneController.EnemyKilled(100);
        else
            GameSceneController.EnemyKilled(0);
    }

}

void OnTriggerStay2D(Collider2D trigger)
{
    if (trigger.tag == "ground")
    {
        isGrounded = true;
    }

}

void OnTriggerEnter2D(Collider2D trigger)
{
    if (trigger.tag == "Player" && isGrounded && isStunned)
    {
        isDeath = true;
    }
    else if ((trigger.tag == "Enemy" && isTurning == false) ||
              (trigger.tag == "Player" && isStunned == false) ||
                 (trigger.tag == "FireBall" && isTurning == false))
    {
        if (direction == Direction.Right)
        {
            direction = Direction.Left;
        }
        else
        {
            direction = Direction.Right;
        }
        isTurning = true;
    }
}

void OnTriggerExit2D(Collider2D trigger)
{
    if (trigger.tag == "ground")
    {
        isGrounded = false;
    }
}
```

```
public void StunController()
{
   if (isGrounded && isStunned == false)
   {
      if (level == 2)
      {
         reduceLevel = true;
         level--;
         speed += speed / 2;
         if (speed > limitSpeed)
            speed = limitSpeed;
      }
      else if(level == 1)
      {
         level--;
         isStunned = true;
      }
   }
   else if (isGrounded && isStunned)
   {
      isStunned = false;
      isTurning = false;
      level = 2;
      speed = 0.05f;
   }
}

public void EndTurn()
{
   isTurning = false;
}

public void LevelUp()
{
   isStunned = false;
   level = 2;
}

public void DeactiveReduceLevel()
{
   reduceLevel = false;
}

public bool GetStun()
{
   return isStunned;
}

public bool GetTurn()
```

```
    {
        return isTurning;
    }

    public int GetLevel()
    {
        return level;
    }

    public void SetSpeed(float speed)
    {
        this.speed = speed;
    }

    public void SetLevel(int level)
    {
        this.level = level;
    }
}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class Enemy3 : MonoBehaviour
{
    protected bool isGrounded;
    protected bool isStunned;
    protected bool isDeath;
    protected bool isReadyToJump;
    protected int level;
    protected float jumpForce;
    protected float speed;
    EnemyGenerator3 enemyGenerator;
    MultiEnemyGeneratorController multiEnemyGenerator;
    Animator anim;
    Collider2D col2D;
    Rigidbody2D rb2d;
    protected Transform transformEnemy;
    protected SpriteRenderer sprite;
    public enum Direction { Left, Right };
    protected Direction direction;

    void Awake()
    {
        level = 1;

        transformEnemy = GetComponent<Transform>();
        anim = GetComponentInChildren<Animator>();
        sprite = GetComponentInChildren<SpriteRenderer>();
```

```
    col2D = GetComponent<Collider2D>();
    rb2d = GetComponent<Rigidbody2D>();
    enemyGenerator = FindObjectOfType<EnemyGenerator3>();
    multiEnemyGenerator =
        FindObjectOfType<MultiEnemyGeneratorController>();
}
// Start is called before the first frame update
void Start()
{
    jumpForce = 3.4f;
    speed = 2f;
    isStunned = false;
    isDeath = false;
    isReadyToJump = false;
    if (transformEnemy.position.x < 0)
        direction = Direction.Right;
    else
        direction = Direction.Left;
}

// Update is called once per frame
void Update()
{
    if (isDeath)
    {
        anim.SetBool("Death", true);
    }
    else
    {
        anim.SetBool("Death", false);
    }

    if (isStunned)
    {
        anim.SetBool("Stun", true);
    }
    else
    {
        anim.SetBool("Stun", false);
    }

    if (isGrounded)
    {
        anim.SetBool("Ground", true);
    }
    else
    {
        anim.SetBool("Ground", false);
    }
}
```

```
void FixedUpdate()
{
   if (isStunned == false && isReadyToJump)
   {
      if (direction == Direction.Right)
      {
         rb2d.AddForce(new Vector2(speed, jumpForce), ForceMode2D.Impulse);
      }
      else
      {
         rb2d.AddForce(new Vector2(-speed, jumpForce), ForceMode2D.Impulse);
      }
      isReadyToJump = false;
   }
   else if (isDeath)
   {
      transformEnemy.position = new Vector2(
         transformEnemy.position.x, transformEnemy.position.y - 0.07f);
      col2D.enabled = false;
   }

   if ((transformEnemy.position.x < -8.8
         || transformEnemy.position.x > 8.8))
   {
      if (transformEnemy.position.y < -2.5)
      {
         if (multiEnemyGenerator == null)
            enemyGenerator.CreateEnemy(this.level, this.speed, 3);
         else
            multiEnemyGenerator.CreateEnemy(this.level, this.speed, 3);
         Destroy(gameObject);
      }
      else if (transformEnemy.position.x < 0)
         transformEnemy.position = new Vector2(
            8.7f, transform.position.y);
      else
         transformEnemy.position = new Vector2(
            -8.7f, transform.position.y);
   }
   else if (transformEnemy.position.y < -5.5)
   {
      Destroy(gameObject);
      if (SceneManager.GetActiveScene().name.StartsWith("Scene"))
         GameSceneController.EnemyKilled(200);
      else
         GameSceneController.EnemyKilled(0);
   }

}
```

```
void OnTriggerStay2D(Collider2D trigger)
{
   if (trigger.tag == "ground")
   {
      isGrounded = true;
   }

}

void OnTriggerEnter2D(Collider2D trigger)
{
   if (trigger.tag == "Player" && isGrounded && isStunned)
   {
      isDeath = true;
   }
   else if ((trigger.tag == "Enemy" && isStunned == false) ||
            (trigger.tag == "Player" && isStunned == false) ||
               (trigger.tag == "FireBall" && isStunned == false))
   {
      if (direction == Direction.Right)
      {
         direction = Direction.Left;
      }
      else
      {
         direction = Direction.Right;
      }
   }
}

void OnTriggerExit2D(Collider2D trigger)
{
   if (trigger.tag == "ground")
   {
      isGrounded = false;
   }
}

public void LevelUp()
{
   speed += speed / 5;
   isStunned = false;
   level++;
}

public void StunController()
{
   if (isGrounded && isStunned == false)
   {
```

```
            isStunned = true;
        }
        else if (isGrounded && isStunned)
        {
            isStunned = false;
        }
    }

    public bool GetStun()
    {
        return isStunned;
    }

    public void SetSpeed(float speed)
    {
        this.speed = speed;
    }

    public void SetLevel(int level)
    {
        this.level = level;
    }

    public void ReadyToJump()
    {
        this.isReadyToJump = true;
    }

    public void Revive()
    {
        this.isStunned = false;
        LevelUp();
    }
}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class EnemyBody1 : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
```

```
    }

    void SendLevelUp()
    {
        transform.parent.SendMessage("LevelUp");
    }

    void EndTurn()
    {
        transform.parent.SendMessage("EndTurn");
    }
}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class EnemyBody2 : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }

    void SendLevelUp()
    {
        transform.parent.SendMessage("LevelUp");
    }

    void EndTurn()
    {
        transform.parent.SendMessage("EndTurn");
    }

    void FinishedJump()
    {
        transform.parent.SendMessage("DeactiveReduceLevel");
    }
}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class EnemyBody3 : MonoBehaviour
```

```
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }

    void SendLevelUp()
    {
        transform.parent.SendMessage("LevelUp");
    }

    void EndTurn()
    {
        transform.parent.SendMessage("EndTurn");
    }

    void SendReadyToJump()
    {
        transform.parent.SendMessage("ReadyToJump");
    }

    void SendRevive()
    {
        transform.parent.SendMessage("Revive");
    }
}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public abstract class EnemyGenerator : MonoBehaviour
{
    protected float generatorTimer;
    private float initialTimer;
    public int maxEnemies;

    protected AudioSource audioSource;
    public AudioClip generateEnemyAudio;

    // Start is called before the first frame update
    void Awake()
    {
```

```
        maxEnemies =
System.Convert.ToInt32(SceneManager.GetActiveScene().name.Substring(
            SceneManager.GetActiveScene().name.LastIndexOf("l") + 1));
        audioSource = GetComponent<AudioSource>();
        audioSource.clip = generateEnemyAudio;
    }

    void Start()
    {
        initialTimer = Random.Range(0, 3);
        generatorTimer = Random.Range(5, 10);
        StartGenerator();
    }

    // Update is called once per frame
    void Update()
    {
        if (maxEnemies <= 0)
            CancellGenerator();
    }

    public void SetMaxEnemies(int maxEnemies)
    {
        this.maxEnemies = maxEnemies;
    }

    public abstract void CreateStartEnemies();

    public void StartGenerator()
    {
        InvokeRepeating("CreateStartEnemies", initialTimer, generatorTimer);
    }

    public void CancellGenerator()
    {
        CancelInvoke("CreateStartEnemies");
    }

    public abstract void CreateEnemy(int level, float speed, int type);
}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class EnemyGenerator1 : EnemyGenerator
{
    public Enemy prefabEnemy;

    public override void CreateEnemy(int level, float speed, int type)
    {
```

```
        audioSource.Play();
        Enemy enemyCreated =
            Instantiate(prefabEnemy, transform.position, Quaternion.identity);

        enemyCreated.SetSpeed(speed);
        enemyCreated.SetLevel(level);
    }

    public override void CreateStartEnemies()
    {
        audioSource.Play();
        Instantiate(prefabEnemy, transform.position, Quaternion.identity);
        maxEnemies--;
    }
}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class EnemyGenerator2 : EnemyGenerator
{
    public Enemy2 prefabEnemy;

    public override void CreateEnemy(int level, float speed, int type)
    {
        audioSource.Play();
        Enemy2 enemyCreated =
            Instantiate(prefabEnemy, transform.position, Quaternion.identity);

        enemyCreated.SetSpeed(speed);
        enemyCreated.SetLevel(level);
    }

    public override void CreateStartEnemies()
    {
        audioSource.Play();
        Instantiate(prefabEnemy, transform.position, Quaternion.identity);
        maxEnemies--;
    }
}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class EnemyGenerator3 : EnemyGenerator
{
    public Enemy3 prefabEnemy;

    public override void CreateEnemy(int level, float speed, int type)
    {
```

```
        audioSource.Play();
        Enemy3 enemyCreated =
            Instantiate(prefabEnemy, transform.position, Quaternion.identity);

        enemyCreated.SetSpeed(speed);
        enemyCreated.SetLevel(level);
    }

    public override void CreateStartEnemies()
    {
        audioSource.Play();
        Instantiate(prefabEnemy, transform.position, Quaternion.identity);
        maxEnemies--;
    }
}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class FireBallBody : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }

    public void StartMovement()
    {
        transform.parent.SendMessage("StartMovement");
    }
}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class FireBallController : MonoBehaviour
{
    protected Transform transformFireBall;
    protected Animator anim;
    protected SpriteRenderer bodySprite;
    public GameObject collX;
    public GameObject collY;
    protected Collider2D[] collidersX;
```

```
protected Collider2D[] collidersY;
protected Collider2D coll;

protected bool movementActived;
protected float speedX, speedY;
protected int timer;

public AudioClip rebound;
protected AudioSource audioSource;


// Start is called before the first frame update
void Start()
{
    transformFireBall = GetComponent<Transform>();
    anim = GetComponentInChildren<Animator>();
    bodySprite = gameObject.GetComponentInChildren<SpriteRenderer>();
    coll = GetComponent<Collider2D>();
    collidersX = collX.GetComponents<Collider2D>();
    collidersY = collY.GetComponents<Collider2D>();
    audioSource = GetComponent<AudioSource>();

    speedX = 0.025f;
    speedY = 0.025f;
    movementActived = false;
    timer = 400;
    audioSource.clip = rebound;
}

void Update()
{
    if (timer > 0)
        timer--;
    else
        Active();
}

void FixedUpdate()
{
    if(movementActived)
    {
        transform.position = new Vector2(transform.position.x+speedX,
            transform.position.y + speedY);
    }

    if (transformFireBall.position.x < -9 || transformFireBall.position.x > 9)
    {
        CollideX();
    }
    if (transformFireBall.position.y > 5 )
```

```
            {
                CollideY();
            }
        }

        public void Active()
        {
            bodySprite.enabled = true;
            anim.enabled = true;
            for (int i = 0; i < collidersX.Length; i++)
            {
                collidersX[i].enabled = true;
                collidersY[i].enabled = true;
            }
            coll.enabled = true;
            collX.GetComponent<ColliderX>().enabled = true;
            collY.GetComponent<ColliderY>().enabled = true;
    }

        public void StartMovement()
        {
            movementActived = true;
        }

        void CollideX()
        {
            speedX = -speedX;
            audioSource.Play();
        }

        void CollideY()
        {
            speedY = -speedY;
            audioSource.Play();
        }
    }
    using System.Collections;
    using System.Collections.Generic;
    using UnityEngine;

    public class FireBallGreenController : MonoBehaviour
    {
        protected Transform transformFireBall;
        protected Animator anim;
        protected SpriteRenderer bodySprite;
        protected ColliderY collY;
        protected CircleCollider2D coll;

        public CapsuleCollider2D[] colliders;
```

```
protected bool movementActived;
protected float speedX, speedY;
protected int timer;

// Start is called before the first frame update
void Start()
{
   transformFireBall = GetComponent<Transform>();
   anim = GetComponentInChildren<Animator>();
   bodySprite = gameObject.GetComponentInChildren<SpriteRenderer>();
   collY = gameObject.GetComponentInChildren<ColliderY>();
   coll = GetComponent<CircleCollider2D>();

   speedX = 0.025f;
   speedY = 0.025f;
   movementActived = false;
   timer = 100;
}

void Update()
{
   if (timer > 0)
      timer--;
   else
      Active();
}

void FixedUpdate()
{
   if (movementActived)
   {
      transform.position = new Vector2(transform.position.x + speedX,
         transform.position.y + speedY);
   }

   if (transformFireBall.position.x < -9 ||
         transformFireBall.position.x > 9)
   {
      transformFireBall.position =
         new Vector2(-transform.position.x, transform.position.y);
   }

   if (transformFireBall.position.y < 0.3 ||
         transformFireBall.position.y > 0.8f)
   {
      CollideY();
   }
}

public void Active()
```

```
    {
        bodySprite.enabled = true;
        anim.enabled = true;
        collY.enabled = true;
        coll.enabled = true;
    }

    public void StartMovement()
    {
        movementActived = true;
    }

    void CollideY()
    {
        speedY = -speedY;
    }
}
using System.Collections;
using System.Collections.Generic;
using UnityEngine.SceneManagement;
using UnityEngine;

public class GameSceneController : MonoBehaviour
{
    public static int score = 0;
    public static int level = 1;
    public static int enemiesDeath;
    public static int playersDeath;

    private int enemies;
    private string quantityPlayers;
    protected MultiEnemyGeneratorController[] multiEnemyGenerators;

    public MessageController topScoreMessage;
    public MessageController actualScoreMessage;
    public GameObject topSprite;

    public AudioClip startScene;
    protected AudioSource audioSource;

    private static GameSceneController instance = null;

    void Awake()
    {
        if (instance == null)
            instance = this;
        else if (instance != this)
            Destroy(gameObject);

        DontDestroyOnLoad(gameObject);
```

```
}
// Start is called before the first frame update
void Start()
{
    quantityPlayers = "OnePlayer";
    enemies = 0;
    enemiesDeath = -1;
    playersDeath = 0;
    audioSource = GetComponent<AudioSource>();
}

// Update is called once per frame

void Update()
{
    if (enemies == enemiesDeath)
    {
        ChangeScene();
    }

    SetActiveScoreMessage();

    if (SceneManager.GetActiveScene().name.Contains("Level"))
    {
        if (quantityPlayers == "OnePlayer" && playersDeath == 1)
        {
            GoToSceneScore();
        }
        else if (quantityPlayers == "TwoPlayers" &&
                playersDeath == 2)
        {
            GoToSceneScore();
        }
    }

    //If press Q you can advance to the next Scene
    if (Input.GetKeyDown(KeyCode.Q))
    {
        enemiesDeath = enemies;
    }

    if (Input.GetKeyDown(KeyCode.Escape))
    {
        Application.Quit();
    }

}

public void GoToSceneScore()
{
```

```
    SceneManager.LoadScene("SceneScore");
}

public static void EnemyKilled(int punctuation)
{
    if(score < 99999)
        score += punctuation;
    enemiesDeath++;
}

public static void PlayerDeath()
{
    playersDeath++;
}

public void ChangeScene()
{
    if (level <= 3 &&
            SceneManager.GetActiveScene().name.StartsWith("Scene"))
    {
        enemiesDeath = -1;
        enemies = 0;
        SceneManager.LoadScene("Demonstration" + level);
    }
    else if (SceneManager.GetActiveScene().name.
        StartsWith("Demonstration"))
    {
        enemies = 2 * level;
        enemiesDeath = 0;
        SceneManager.LoadScene("Scene" + quantityPlayers +
            "Level" + level);
        level++;
    }
    else
    {
        enemies = 2 * level;
        enemiesDeath = 0;
        SceneManager.LoadScene("Scene" + quantityPlayers +
            "Level4");
        multiEnemyGenerators =
            FindObjectsOfType<MultiEnemyGeneratorController>();
        for (int i = 0; i < multiEnemyGenerators.Length; i++)
        {
            multiEnemyGenerators[i].SetMaxEnemies(level);
        }
    }
    audioSource.clip = startScene;
    audioSource.Play();
}
```

```
public void SetActiveScoreMessage()
{
   if (SceneManager.GetActiveScene().name.Contains("Level"))
   {
      LoadTopScore();
      UpdateScoreMessage();
      topSprite.SetActive(true);
   }
   else
   {
      topScoreMessage.Hide();
      actualScoreMessage.Hide();
      topSprite.SetActive(false);
   }
}

public void GoBack()
{
   SceneManager.LoadScene("InitialScene");
   score = 0;
   level = 1;
   quantityPlayers = "OnePlayer";
   enemies = 0;
   playersDeath = 0;
   enemiesDeath = -1;
}

public void SetQuantityPlayers(string quantityPlayers)
{
   this.quantityPlayers = quantityPlayers;
}

public void StartDemonstration1()
{
   SceneManager.LoadScene("Demonstration1");
}

public string GetQuantityPlayers()
{
   return quantityPlayers;
}

public void LoadTopScore()
{
   ScoreBoard sc = new ScoreBoard("scoreboard.txt");
   int topScore = sc.GetScore(sc.GetNamesSorted()[0]);
   topScoreMessage.SetMessage(System.Convert.ToString(topScore));
   topScoreMessage.Draw();
}
```

```
    public void UpdateScoreMessage()
    {
        actualScoreMessage.Hide();
        actualScoreMessage.SetMessage("score " +
            System.Convert.ToString(score));
        actualScoreMessage.Draw();
    }

}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class InitialScene : MonoBehaviour
{
    public MessageController messageOnePlayer;
    public MessageController messageTwoPlayers;
    public MessageController messageScoreBoard;
    public MessageController messageLeave;
    public MessageController messageCoins;
    public MessageController messageInsertCoins;
    public MessageController messageCoinsNeeded;
    public MessageController messageHelp;
    public Transform messageTitleTransform;
    public GameObject pointer;
    public GameSceneController gameSceneController;
    private Transform pointerTransform;
    private int coins;
    private bool started;
    private float timer;

    // Start is called before the first frame update
    void Start()
    {
        gameSceneController =
            GameObject.FindGameObjectWithTag("MainCamera").
                GetComponent<GameSceneController>();
        pointerTransform = pointer.GetComponent<Transform>();
        started = false;
        coins = 0;
        timer = 0;
    }

    // Update is called once per frame
    void FixedUpdate()
    {
        if (started)
        {
            if (Input.GetKeyDown(KeyCode.DownArrow) &&
```

```
pointerTransform.position.y > -3.5)
{
   gameSceneController.SetQuantityPlayers("TwoPlayers");
   pointerTransform.position =
      new Vector2(pointerTransform.position.x,
         pointerTransform.position.y - 0.65f);
}
else if (Input.GetKeyDown(KeyCode.UpArrow) &&
   pointerTransform.position.y < -2.2)
{
   pointerTransform.position =
      new Vector2(pointerTransform.position.x,
         pointerTransform.position.y + 0.65f);
   if(pointerTransform.position.y > -2.25)
      gameSceneController.SetQuantityPlayers("OnePlayer");
}

if (Input.GetKeyDown(KeyCode.Space))
{
   if(pointerTransform.position.y > -2.25 &&
      gameSceneController.GetQuantityPlayers().Contains("One")
         && coins >= 1)
      gameSceneController.StartDemonstration1();
   else if (pointerTransform.position.y > -2.25 &&
         gameSceneController.GetQuantityPlayers().Contains("One"))
   {
      DrawWarningMessage(1 - coins);
   }
   else if (pointerTransform.position.y < -2.2 &&
       pointerTransform.position.y > -2.5  &&
       gameSceneController.GetQuantityPlayers().Contains("Two")
         && coins >= 2)
      gameSceneController.StartDemonstration1();
   else if (pointerTransform.position.y < -2.2 &&
       pointerTransform.position.y > -2.5 &&
       gameSceneController.GetQuantityPlayers().Contains("Two"))
   {
      DrawWarningMessage(2 - coins);
   }
   else if (pointerTransform.position.y < -2.4 &&
       pointerTransform.position.y > -3.1)
   {
      gameSceneController.GoToSceneScore();
   }
   else if (pointerTransform.position.y < -3.25)
   {
      Application.Quit();
   }
}
else if (Input.GetKeyDown(KeyCode.C))
```

```
        {
            SetCoins(coins+1);
        }
    }
    else
    {
        messageTitleTransform.position = new Vector2(
            messageTitleTransform.position.x,
                messageTitleTransform.position.y - 0.05f);
        if(messageTitleTransform.position.y < 2.5)
        {
            started = true;
            Unlock();
        }
    }

    if(timer > 0)
    {
        timer -= Time.deltaTime;
    }
    else if (timer < 0)
    {
        messageCoinsNeeded.Hide();
        timer = 0;
    }

}

private void Unlock()
{
    messageInsertCoins.SetMessage("press c to insert coin");
    messageLeave.SetMessage("leave ");
    messageScoreBoard.SetMessage("scoreboard");
    messageCoins.SetMessage("coins " + coins);
    messageOnePlayer.SetMessage("one player");
    messageTwoPlayers.SetMessage("two players");
    messageHelp.SetMessage("press space to start");

    messageInsertCoins.Draw();
    messageOnePlayer.Draw();
    messageTwoPlayers.Draw();
    messageScoreBoard.Draw();
    messageLeave.Draw();
    messageCoins.Draw();
    messageHelp.Draw();
}

public void SetCoins(int coins)
{
    this.coins = coins;
```

```
        messageCoins.SetMessage("coins " + coins);
        messageCoins.Hide();
        messageCoins.Draw();
    }

    public void DrawWarningMessage(int coinsNeeded)
    {
        messageCoinsNeeded.SetMessage("you need " + coinsNeeded + " more coins");
        messageCoinsNeeded.Draw();
        timer = 1;
    }

}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MessageController : MonoBehaviour
{
    protected Dictionary<char, Sprite> letters;
    public GameObject prefabLetter;
    protected string message;

    // Start is called before the first frame update
    void Awake()
    {
        letters = new Dictionary<char, Sprite>();
        string path = "letters";

        Sprite[] lettersSprite = Resources.LoadAll<Sprite>(path);

        for (int i = 97; i <= 122; i++)
        {
            foreach(Sprite s in lettersSprite)
            {
                if(s.name.Contains("("+(i-96)+")"))
                {
                    letters.Add(System.Convert.ToChar(i), s);
                }
            }
        }

        path = "numbers";

        Sprite[] numbersSprite = Resources.LoadAll<Sprite>(path);
        foreach (Sprite s in numbersSprite)
        {
            letters.Add(System.Convert.ToChar(s.name), s);
        }
    }
```

```
    // Update is called once per frame
    void Update()
    {

    }

    public void SetMessage(string message)
    {
        this.message = message;
    }

    public string GetMessage()
    {
        return message;
    }

    public void Draw()
    {
        float offset = 0;
        foreach(char c in message)
        {
            if(c != ' ')
            {
                prefabLetter.GetComponent<SpriteRenderer>().sprite = letters[c];
                prefabLetter.transform.position =
                    new Vector2(this.transform.position.x + offset,
                        this.transform.position.y);
                GameObject letter = GameObject.Instantiate(prefabLetter);
                letter.transform.parent = this.gameObject.transform;

            }
            offset += 0.45f;
        }
    }

    public void Hide()
    {
        for (int i = 0; i < this.transform.childCount; i++)
        {
            this.transform.GetChild(i).gameObject.SetActive(false);
        }
    }
}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MultiEnemyGeneratorController : EnemyGenerator
{
```

```
public Enemy prefabEnemy1;
public Enemy2 prefabEnemy2;
public Enemy3 prefabEnemy3;

public override void CreateEnemy(int level, float speed, int type)
{
   audioSource.Play();
   if (type == 1)
   {
      Enemy enemyCreated =
         Instantiate(prefabEnemy1, transform.position, Quaternion.identity);
      enemyCreated.SetSpeed(speed);
      enemyCreated.SetLevel(level);
   }
   else if (type == 2)
   {
      Enemy2 enemyCreated =
         Instantiate(prefabEnemy2, transform.position, Quaternion.identity);
      enemyCreated.SetSpeed(speed);
      enemyCreated.SetLevel(level);
   }
   else
   {
      Enemy3 enemyCreated =
         Instantiate(prefabEnemy3, transform.position, Quaternion.identity);
      enemyCreated.SetSpeed(speed);
      enemyCreated.SetLevel(level);
   }

}

public override void CreateStartEnemies()
{
   audioSource.Play();
   int type = Random.Range(1, 4);
   if(type == 1)
   {
      Instantiate(prefabEnemy1, transform.position, Quaternion.identity);
   }
   else if (type == 2)
   {
      Instantiate(prefabEnemy2, transform.position, Quaternion.identity);
   }
   else
   {
      Instantiate(prefabEnemy3, transform.position, Quaternion.identity);
   }
   maxEnemies--;
}
```

```
}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class OnePlayerController : PlayerController
{
    protected override void GetInput()
    {
        horizontalInput = Input.GetAxisRaw("Horizontal");
        jumpInput = Input.GetAxisRaw("Jump");
    }

    protected override void SetPositionLives()
    {
        coordinatesLives = new Coordinates(-7.12f, 4.09f);
    }
}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public abstract class PlayerController : MonoBehaviour
{
    protected Transform transformPlayer;
    protected SpriteRenderer sprite;
    protected Rigidbody2D rb2d;
    protected Animator anim;
    protected Collider2D[] colliders2D;

    public GameObject objectLives;
    protected SpriteRenderer[] spritesLives;
    protected struct Coordinates
    {
        public float posX, posY;
        public Coordinates(float positionX, float positionY)
        {
            posX = positionX;
            posY = positionY;
        }
    }

    protected Coordinates coordinatesLives;

    protected float horizontalInput;
    protected float jumpInput;

    protected float speed;
    protected float maxSpeed;
```

```
protected float jumpForce;
protected bool isGrounded;
protected bool isIdle;
protected bool isDamaged;
protected bool isFalling;
protected bool isBurned;
protected bool isEnemyStunned;
protected bool alive;

protected int lives;

protected AudioSource audioSource;
public AudioClip jumpSound;
public AudioClip deathSound;
public AudioClip reviveSound;

private void Awake()
{
   rb2d = GetComponent<Rigidbody2D>();
   anim = GetComponent<Animator>();
   transformPlayer = GetComponent<Transform>();
   sprite = GetComponent<SpriteRenderer>();
   colliders2D = GetComponentsInChildren<Collider2D>();
   spritesLives = objectLives.GetComponentsInChildren<SpriteRenderer>();
   audioSource = GetComponent<AudioSource>();
}

private void Start()
{
   SetPositionLives();

   maxSpeed = 4;
   speed = 5;
   jumpForce = 3.4f;
   lives = 3;

   alive = true;
}

private void Update()
{
   if(horizontalInput > 0)
   {
      sprite.flipX = true;
      anim.SetBool("Idle", false);
   }
   else if(horizontalInput < 0)
   {
      sprite.flipX = false;
      anim.SetBool("Idle", false);
```

```
}
else
{
   anim.SetBool("Idle", true);
}

if(isGrounded)
{
   anim.SetBool("Grounded", true);
}
else
{
   anim.SetBool("Grounded", false);
}

if (isDamaged && isBurned)
{
   anim.SetBool("Damaged", true);
   anim.SetBool("Burned", true);
}
else if (isDamaged)
{
   anim.SetBool("Damaged", true);
}
else
{
   anim.SetBool("Damaged", false);
   anim.SetBool("Burned", false);
}

if(isFalling)
{
   anim.SetBool("Falling", true);
}
else
{
   anim.SetBool("Falling", false);
}

if (alive)
{
   anim.SetBool("Alive", true);
}
else
{
   anim.SetBool("Alive", false);
}

objectLives.transform.position = new Vector2(coordinatesLives.posX,
      coordinatesLives.posY);
```

```
    }

    private void FixedUpdate()
    {
        GetInput();

        if(!isDamaged)
            rb2d.AddForce(Vector2.right * speed * horizontalInput);

        if (rb2d.velocity.x > maxSpeed)
        {
            rb2d.velocity = new Vector2(maxSpeed, rb2d.velocity.y);
        }
        else if(rb2d.velocity.x < -maxSpeed)
        {
            rb2d.velocity = new Vector2(-maxSpeed, rb2d.velocity.y);
        }

        if (jumpInput != 0 && isGrounded)
        {
            speed = 5;
            rb2d.velocity = new Vector2(0,rb2d.velocity.y);
            rb2d.AddForce(new Vector2(0, jumpForce), ForceMode2D.Impulse);
            audioSource.clip = jumpSound;
            audioSource.Play();
        }
        else
        {
            speed = 15;
        }
        if (transformPlayer.position.x < -9 || transformPlayer.position.x > 9)
        {
            transformPlayer.position = new Vector2(-transform.position.x,
transform.position.y);
        }

        if(isFalling)
        {
            transform.position = new Vector2(
                transform.position.x, transform.position.y - 0.07f);
        }

        if(alive && transform.position.y < -5.5)
        {
            lives--;
            if(lives > 0)
            {
                alive = false;
                spritesLives[lives - 1].enabled = false;
```

```
            Revive();
        }
        else
        {
            GameSceneController.PlayerDeath();
            Destroy(gameObject);
        }
    }
    else if (!alive && transform.position.y > 3.0f)
    {
        if(transform.position.y > 3.3f)
        {
            transform.position = new Vector2(0, transform.position.y - 0.001f);
        }
        else if (Input.anyKey)
        {
            Restart();
        }
        else
        {
            transform.position = new Vector2(0, 3.2f);
        }
    }

}

protected abstract void GetInput();

protected abstract void SetPositionLives();


private void OnCollisionExit2D(Collision2D collision)
{
    isGrounded = false;
}

private void OnTriggerEnter2D(Collider2D trigger)
{
    if(trigger.tag == "Enemy")
    {
        isEnemyStunned =
            trigger.GetComponentInChildren<Animator>().GetBool("Stun");
        if(!isEnemyStunned)
        {
            Damaged();
        }
    }

    else if (trigger.tag == "FireBall")
    {
```

```
            Burned();
        }
    }

    private void OnTriggerStay2D(Collider2D trigger)
    {
        if (trigger.tag == "ground" || trigger.tag == "Pow")
        {
            isGrounded = true;
        }
    }

    private void OnTriggerExit2D(Collider2D trigger)
    {
        if (trigger.tag == "ground" || trigger.tag == "Pow")
        {
            isGrounded = false;
        }
    }

    private void Fall()
    {
        for (int i = 0; i < colliders2D.Length; i++)
        {
            colliders2D[i].enabled = false;
        }
        isFalling = true;
    }

    private void Revive()
    {
        audioSource.clip = reviveSound;
        audioSource.Play();
        transform.position = new Vector2(0, 5);
        rb2d.bodyType = RigidbodyType2D.Static;
    }

    private void Restart()
    {
        for (int i = 0; i < colliders2D.Length; i++)
        {
            colliders2D[i].enabled = true;
        }
        isFalling = false;
        isDamaged = false;
        isBurned = false;
        rb2d.bodyType = RigidbodyType2D.Dynamic;
        alive = true;
    }
```

```
    public void Damaged()
    {
        audioSource.clip = deathSound;
        audioSource.Play();
        isDamaged = true;
        rb2d.velocity = new Vector2(0, 0);
    }

    public void Burned()
    {
        isBurned = true;
        Damaged();
    }

    public int GetLives()
    {
        return lives;
    }
}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerDemonstration : MonoBehaviour
{
    Transform transformPlayer;
    SpriteRenderer sprite;
    Rigidbody2D rb2d;
    Animator anim;
    Collider2D[] colliders2D;

    float speed;
    float jumpForce;

    bool isGrounded;

    protected AudioSource audioSource;
    public AudioClip jump;

    private void Awake()
    {
        rb2d = GetComponent<Rigidbody2D>();
        anim = GetComponent<Animator>();
        transformPlayer = GetComponent<Transform>();
        sprite = GetComponent<SpriteRenderer>();
        colliders2D = GetComponentsInChildren<Collider2D>();
        audioSource = GetComponent<AudioSource>();
    }

    private void Start()
```

```
    {
        speed = 0.06f;
        jumpForce = 3.6f;
    }

    private void Update()
    {

        if (isGrounded)
        {
            anim.SetBool("Grounded", true);
        }
        else
        {
            anim.SetBool("Grounded", false);
        }

    }

    public void MoveRight()
    {
        anim.SetBool("Idle", false);
        transformPlayer.position = new Vector3(transformPlayer.position.x + speed,
transformPlayer.position.y, 0);
        sprite.flipX = true;
    }

    public void MoveLeft()
    {
        anim.SetBool("Idle", false);
        transformPlayer.position = new Vector3(transformPlayer.position.x - speed,
transformPlayer.position.y, 0);
        sprite.flipX = false;
    }

    public void Jump()
    {
        if (isGrounded)
        {
            rb2d.AddForce(new Vector2(0, jumpForce), ForceMode2D.Impulse);
            audioSource.clip = jump;
            audioSource.Play();
        }
    }

    public void SetIdle()
    {
        anim.SetBool("Idle", true);
    }
```

```
    private void OnCollisionExit2D(Collision2D collision)
    {
        isGrounded = false;
    }

    private void OnTriggerStay2D(Collider2D trigger)
    {
        if (trigger.tag == "ground")
        {
            isGrounded = true;
        }
    }

    private void OnTriggerExit2D(Collider2D trigger)
    {
        if (trigger.tag == "ground")
        {
            isGrounded = false;
        }
    }

    public bool IsGrounded()
    {
        return isGrounded;
    }

}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Pow : MonoBehaviour
{
    protected GameObject[] Enemies;
    protected int hits;
    // Start is called before the first frame update
    void Start()
    {
        hits = 0;
    }

    // Update is called once per frame
    void Update()
    {
        if(hits == 3)
        {
            gameObject.SetActive(false);
        }
    }
```

```
    void OnTriggerEnter2D(Collider2D trigger)
    {
        if (trigger.tag == "PlayerHead")
            Active();
    }

    void Active()
    {
        hits++;
        Enemies = GameObject.FindGameObjectsWithTag("Enemy");
        foreach(GameObject e in Enemies)
        {
            e.SendMessage("StunController");
        }
    }
}
//Sergio Ruescas
using System;
using System.Collections.Generic;
using System.IO;

class ScoreBoard
{
    private Dictionary<string, int> scores;
    private List<int> punctuations;
    private string path;

    public ScoreBoard(string path)
    {
        punctuations = new List<int>();
        scores = new Dictionary<string, int>();
        this.path = path;
        Load(path);
    }

    public Dictionary<string, int> GetScores()
    {
        return scores;
    }

    public List<string> GetNamesSorted()
    {
        punctuations.Sort();
        List<string> namesSorted = new List<string>();
        for (int i = punctuations.Count - 1; i >= 0; i--)
        {
            foreach (KeyValuePair<string, int> keyValuePair in scores)
            {
                if (keyValuePair.Value == punctuations[i] &&
                        !namesSorted.Contains(keyValuePair.Key))
```

```
            namesSorted.Add(keyValuePair.Key);
        }
    }
    return namesSorted;
}

public int GetScore(string name)
{
    return scores[name];
}

public bool Add(string name, int punctuation)
{
    if (!scores.ContainsKey(name))
    {
        if (punctuations.Count < 5)
        {
            scores.Add(name, punctuation);
            punctuations.Add(punctuation);
        }
        else
        {
            punctuations.Sort();
            string keyToRemove = "";
            for (int i = 0; i < punctuations.Count; i++)
            {
                if (punctuation > punctuations[i] && keyToRemove == "")
                {
                    foreach (KeyValuePair<string, int> keyValuePair
                                                in scores)
                    {
                        if (keyValuePair.Value == punctuations[i] &&
                            keyToRemove == "")
                        {
                            keyToRemove = keyValuePair.Key;
                        }
                    }
                    scores.Remove(keyToRemove);
                    scores.Add(name, punctuation);

                    punctuations.RemoveAt(i);
                    punctuations.Insert(i, punctuation);
                }
            }
        }
        return true;
    }
    else //The name exists
        return false;
}
```

```
public bool Save()
{
   try
   {
      StreamWriter scoreWriter = new StreamWriter(path);
      foreach (KeyValuePair<string, int> keyValuePair in scores)
      {
         scoreWriter.WriteLine(keyValuePair.Key + ";" + keyValuePair.Value);
      }
      scoreWriter.Close();
      return true;
   }
   catch (PathTooLongException)
   {
      return false;
   }
   catch (IOException)
   {
      return false;
   }
   catch (Exception)
   {
      return false;
   }
}

public bool Load(string path)
{
   if (File.Exists(path))
   {
      try
      {
         string nameToAdd;
         int punctuationToAdd;
         StreamReader scoreReader = new StreamReader(path);
         string linea;
         do
         {
            linea = scoreReader.ReadLine();
            if (linea != null)
            {
               nameToAdd = linea.Substring(0, linea.IndexOf(';'));
               punctuationToAdd = Convert.ToInt32(
                  linea.Substring(linea.IndexOf(';') + 1));
               scores.Add(nameToAdd, punctuationToAdd);
               punctuations.Add(punctuationToAdd);
            }
         } while (linea != null);
         scoreReader.Close();
```

```
            return true;
        }
        catch (PathTooLongException)
        {
            return false;
        }
        catch (IOException)
        {
            return false;
        }
        catch (Exception)
        {
            return false;
        }
    }
    return false;
  }
}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ScoreSceneController : MonoBehaviour
{
  public MessageController[] keyboard;
  public MessageController[] scoreBoardMessages;
  public MessageController nameMessage;
  public MessageController finalMessage;
  public MessageController tooLongMessage;
  public GameObject pointer;
  private ScoreBoard scoreBoard;
  private int pointerPosX;
  private int pointerPosY;
  private string newName;
  private bool finished;
  private float timer;
  // Start is called before the first frame update
  void Awake()
  {
    if(GameSceneController.score > 0)
    {
      finished = false;
    }
    else
    {
      finished = true;
    }
    timer = 0;
  }
  void Start()
```

```
{
    pointerPosX = pointerPosY = 0;
    newName = "";

    finalMessage.SetMessage("press space to go back");
    if( finished)
    {
        finalMessage.Draw();
    }
    tooLongMessage.SetMessage("name too long");

    LoadKeyboard();

    scoreBoard = new ScoreBoard("scoreboard.txt");

    DrawScoreboard();


}

// Update is called once per frame
void Update()
{
    if (finished && Input.GetKeyDown(KeyCode.Space))
    {
            GameObject.FindGameObjectWithTag("MainCamera").
                GetComponent<GameSceneController>().GoBack();
    }
    else if (!finished)
    {
        GetHorizontalInput();
        GetVerticalInput();
        GetModifyInput();
    }

    if (timer > 0)
    {
        timer -= Time.deltaTime;
    }
    else if (timer < 0)
    {
        tooLongMessage.Hide();
        timer = 0;
    }
}

public void HideScoreboard()
{
    for (int i = 0; i < scoreBoardMessages.Length; i++)
    {
```

```
            scoreBoardMessages[i].Hide();
        }
    }

    public void GetHorizontalInput()
    {
        if (Input.GetKeyDown(KeyCode.LeftArrow))
        {
            pointerPosX--;
            pointer.transform.position = new Vector2(pointer.transform.position.x - 0.9f,
                pointer.transform.position.y);
            if (pointerPosX < 0)
            {
                pointerPosX = 9;
                pointer.transform.position = new Vector2(4.15f,
                    pointer.transform.position.y);
            }
        }
        else if (Input.GetKeyDown(KeyCode.RightArrow))
        {
            pointerPosX++;
            pointer.transform.position = new Vector2(pointer.transform.position.x + 0.9f,
                pointer.transform.position.y);
            if (pointerPosX > 9)
            {
                pointerPosX = 0;
                pointer.transform.position = new Vector2(-3.99f,
                    pointer.transform.position.y);
            }
        }
    }

    public void GetVerticalInput()
    {
        if (Input.GetKeyDown(KeyCode.UpArrow))
        {
            pointerPosY--;
            pointer.transform.position = new Vector2(pointer.transform.position.x,
                pointer.transform.position.y + 1);
            if (pointerPosY < 0)
            {
                pointerPosY = 2;
                pointer.transform.position = new Vector2(pointer.transform.position.x,
                    1);
            }
        }
        else if (Input.GetKeyDown(KeyCode.DownArrow))
        {
            pointerPosY++;
            pointer.transform.position = new Vector2(pointer.transform.position.x,
```

```
            pointer.transform.position.y - 1);
        if (pointerPosY > 2)
        {
            pointerPosY = 0;
            pointer.transform.position = new Vector2(
                pointer.transform.position.x, 3);
        }
    }
}

public void GetModifyInput()
{
    if (Input.GetKeyDown(KeyCode.Space))
    {
        WriteName();
    }
    else if (Input.GetKeyDown(KeyCode.Backspace))
    {
        DeleteFromName();
    }
}

public void LoadKeyboard()
{

    for (int i = 1; i < keyboard.Length; i++)
    {
        keyboard[i].transform.position = new Vector2(keyboard[0].transform.position.x,
            keyboard[i - 1].transform.position.y - 1);
    }

    keyboard[0].SetMessage("a b c d e f g h i j");
    keyboard[1].SetMessage("k l m n o p q r s t");
    keyboard[2].SetMessage("u v w x y z 1 2 3 end");

    for (int i = 0; i < keyboard.Length; i++)
    {
        keyboard[i].Draw();
    }
}

public void DrawScoreboard()
{
    List<string> namesSorted = scoreBoard.GetNamesSorted();
    Dictionary<string, int> scores = scoreBoard.GetScores();
    for (int i = 0; i < scores.Count; i++)
    {
        if(i > 0)
            scoreBoardMessages[i].transform.position =
                new Vector2(scoreBoardMessages[0].transform.position.x,
```

```
                scoreBoardMessages[i - 1].transform.position.y - 0.7f);
        switch (i)
        {
            case 0:
                scoreBoardMessages[i].SetMessage("1st   " +
                    namesSorted[i].PadRight(9, ' ') +
                        scores[namesSorted[i]]);
                break;
            case 1:
                scoreBoardMessages[i].SetMessage("2nd   " +
                     namesSorted[i].PadRight(9, ' ') +
                        scores[namesSorted[i]]);
                break;
            case 2:
                scoreBoardMessages[i].SetMessage("3rd   " +
                     namesSorted[i].PadRight(9, ' ') +
                        scores[namesSorted[i]]);
                break;
            default:
                scoreBoardMessages[i].SetMessage((i+1) + "th   " +
                     namesSorted[i].PadRight(9, ' ') +
                        scores[namesSorted[i]]);
                break;
        }
        scoreBoardMessages[i].Draw();
    }
}

public void WriteName()
{
    int posLetter = 0;
    if (pointerPosY != 2 || pointerPosX != 9)
    {
        if (newName.Length < 7)
        {
            foreach (char c in keyboard[pointerPosY].GetMessage())
            {
                if (c != ' ')
                {
                    if (posLetter == pointerPosX && (pointerPosY != 2 || pointerPosX != 9))
                    {
                        newName += c;
                        nameMessage.SetMessage(newName);
                        nameMessage.Hide();
                        nameMessage.Draw();
                    }
                    posLetter++;
                }
            }
        }
```

```
            else
            {
                tooLongMessage.Draw();
                timer = 2;
            }
        }
        else
        {
            bool added =
                scoreBoard.Add(newName, GameSceneController.score);
            if (added)
            {
                finished = true;
                HideScoreboard();
                nameMessage.Hide();
                DrawScoreboard();
                finalMessage.Draw();
                scoreBoard.Save();
            }
        }
    }

    public void DeleteFromName()
    {
        if(newName.Length > 0)
        {
            newName = newName.Substring(0, newName.Length - 1);
            nameMessage.SetMessage(newName);
            nameMessage.Hide();
            nameMessage.Draw();
        }
    }

}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class TwoPlayersController1 : PlayerController
{
    protected override void GetInput()
    {
        horizontalInput = Input.GetAxisRaw("Horizontal2");
        jumpInput = Input.GetAxisRaw("Jump");
    }

    protected override void SetPositionLives()
    {
        coordinatesLives = new Coordinates(-7.12f, 4.09f);
    }
```

```
}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class TwoPlayersController2 : PlayerController
{
    protected override void GetInput()
    {
        horizontalInput = Input.GetAxisRaw("Horizontal");
        jumpInput = Input.GetAxisRaw("Jump2");
    }

    protected override void SetPositionLives()
    {
        coordinatesLives = new Coordinates(-7.12f, 3.60f);
    }
}
```