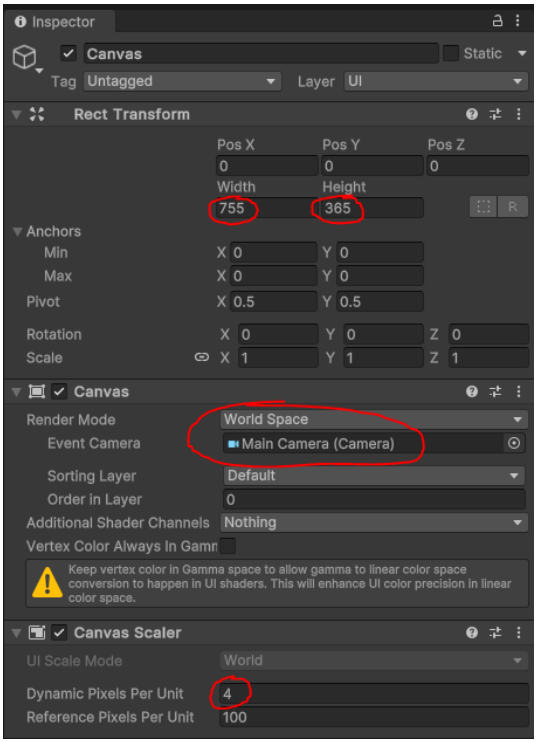
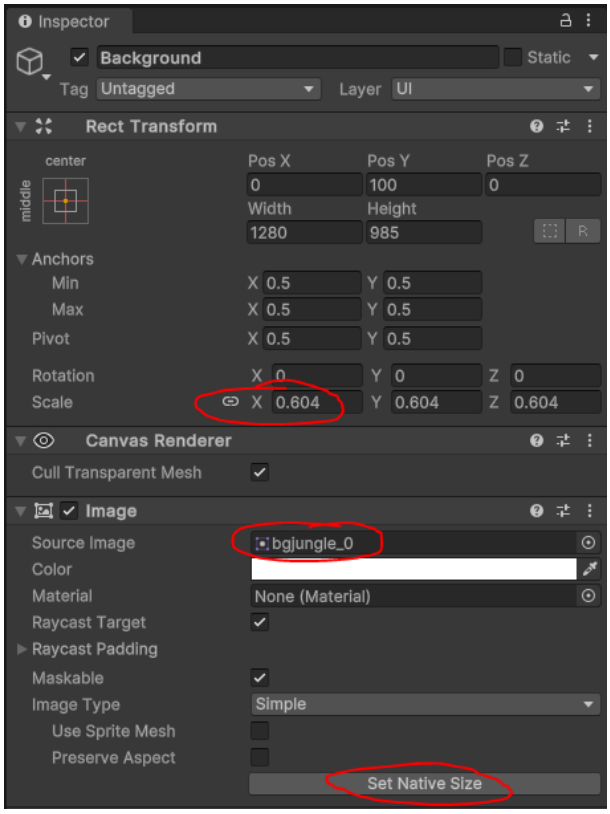
Después de haber creado nuestro proyecto 2D en unity, creamos un canvas, dentro de este solamente agregamos un objeto imagen

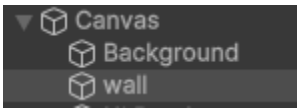
en el inspector del canvas modificamos estos valores.



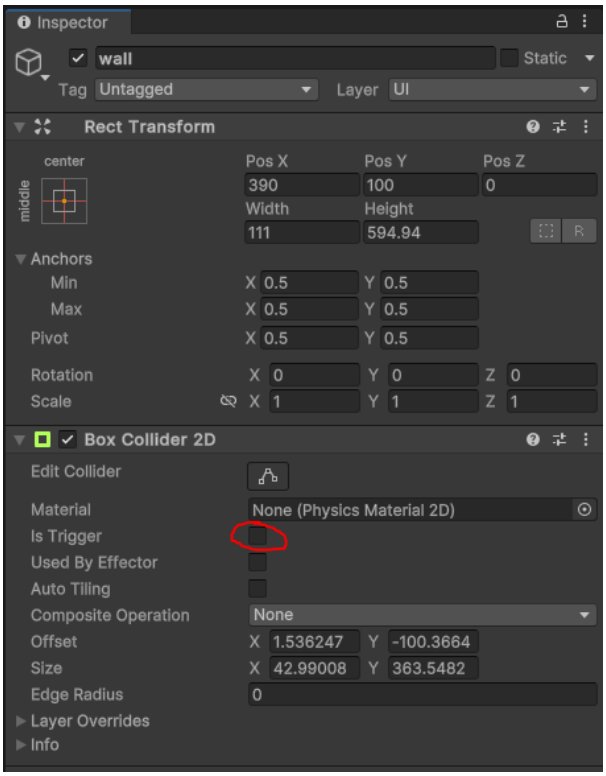
En el inspector de imagen estos valores, no olvides darle click en set native size



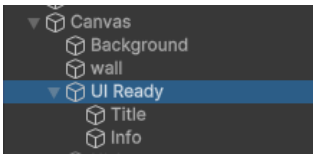
Después ahí mismo en canvas creas un empty con el nombre de wall, esto nos servirá mas tarde para delimitar el recorrido de nuestros dinos.



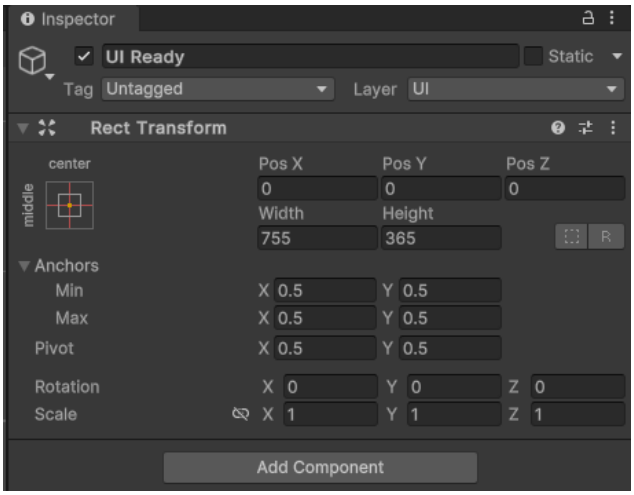
Después crearemos un boxcollider para darle colision a este objeto que como su nombre lo indica será un muro delimitante para nosotros. (asegúrate de que este desactivado trigger). se le agregan estos valores:



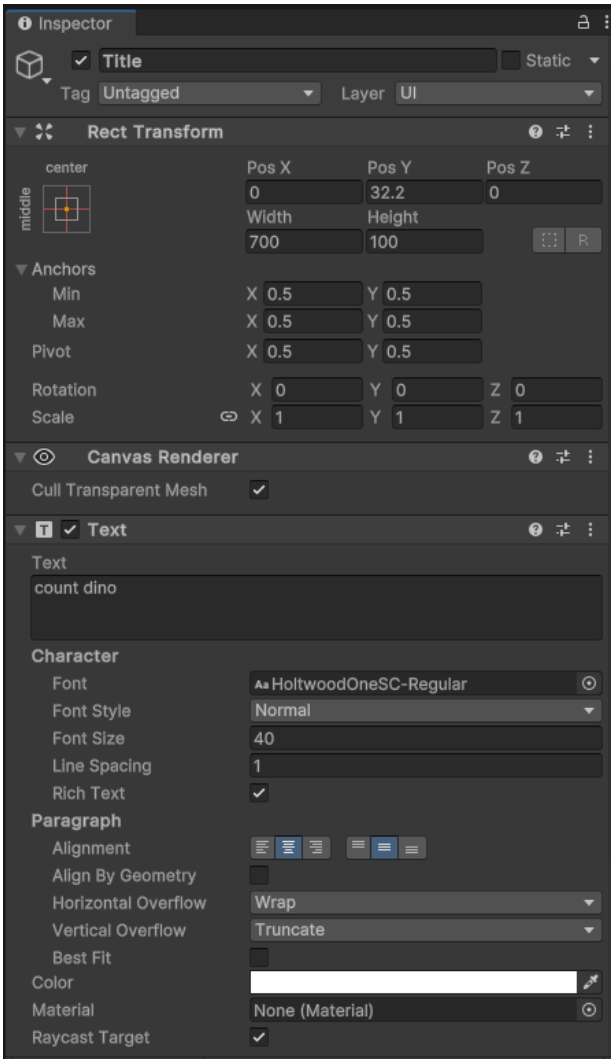
Debajo de esto, pero aun dentro de la carpeta canvas creamos una subcarpeta llamada “ui ready” con 2 elementos tipo texto dentro con sus respectivos nombres

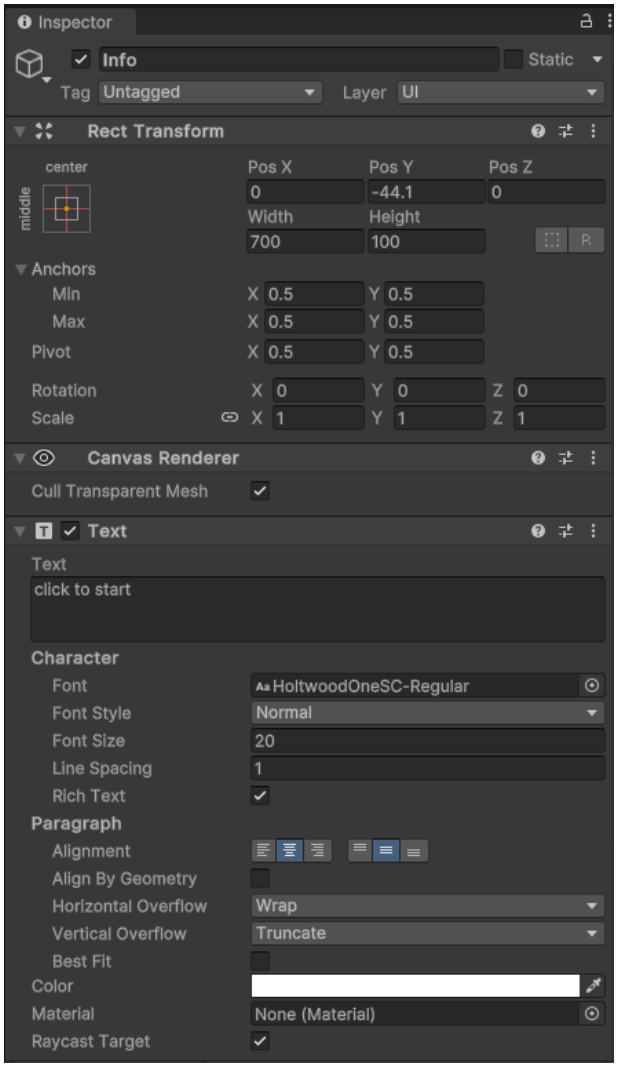


con estos valores:

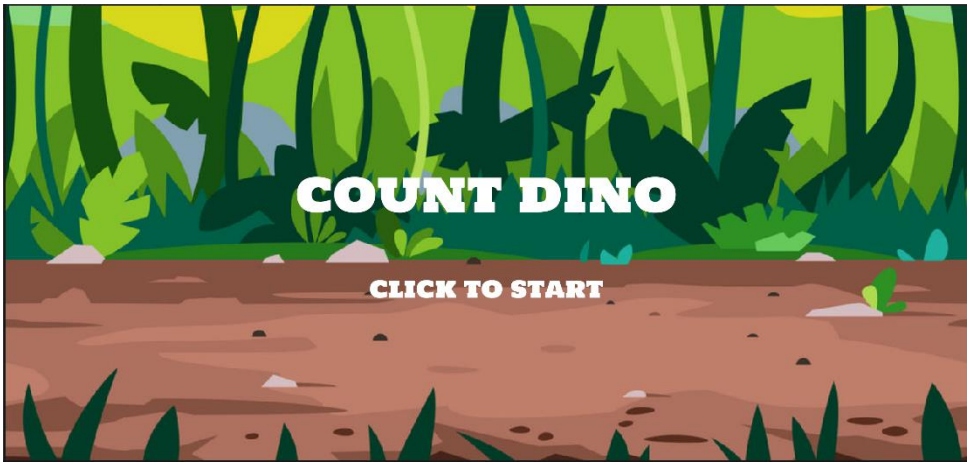


La configuración de ambas es la siguiente:

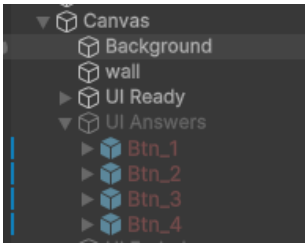




Esta es la pantalla que deberíamos tener:

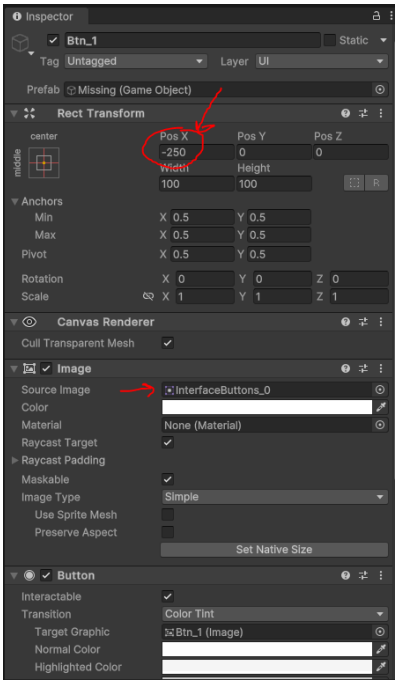


Aun dentro de canvas creas una carpeta extra llamada “ui answers”, ahí mismo creamos 4 botones.



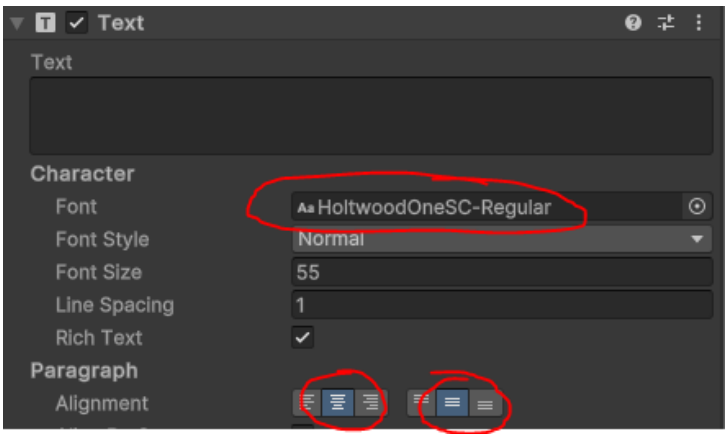
Lo que esta señalado es uno de los mayores cambios que tendrán uno con otro botón, todos son a la misma altura solamente se modifica la posición en x espaciada entre si y las imágenes porque cada botón tiene un color diferente, si se cambia de color el orden no afecta en nada mientras se llamen de la misma manera. las posiciones en x de cada uno son:

btn1 = -250 btn2=-75 btn3=75 btn4=250



Los colores y sombreado de los colores son libre, realmente eso no influye, solamente el tamaño para poder mantener una orden y estética.

No olvidemos que cada botón tiene su texto, este solamente lo modificaremos de esta manera.

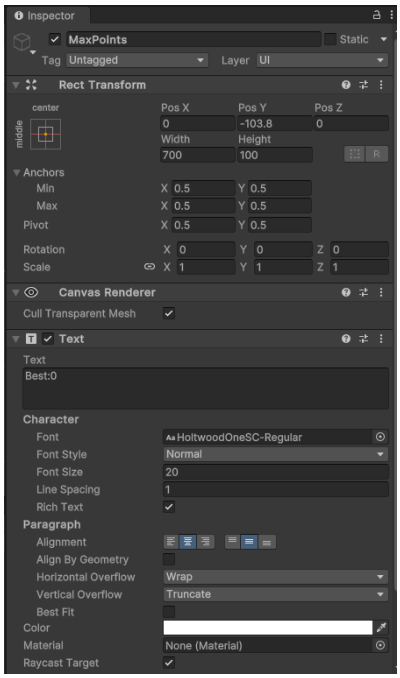


Así debe quedar la pantalla de los botones:

Creamos de nuevo una carpeta más llamada ahora “ui ended”, que fácilmente podemos duplicar la de ui ready, solamente cambiando los textos por otros, así lo haremos, esto tenemos esperado encontrar en el nuevo ui ended

Por ultima carpeta que crearemos en canvas esta esa nueva llama ui score.

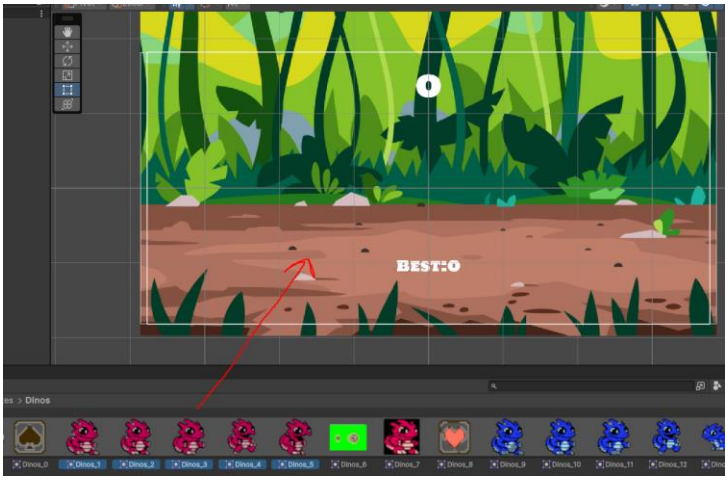
En ambos igual son textos, pero estos son más importantes la dirección donde estén, verificamos que cumpla correctamente:



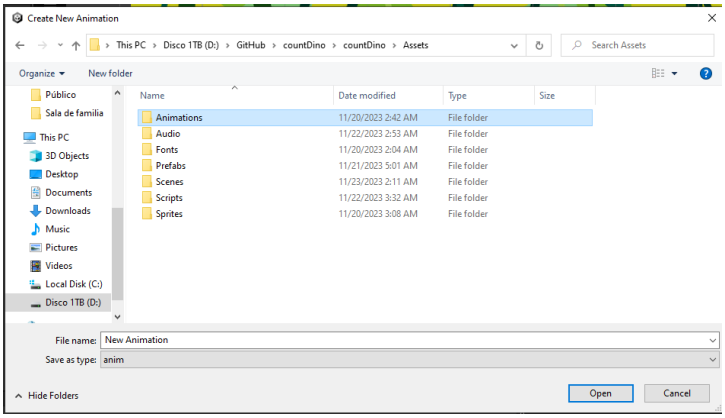
Así quedo la distribución deseada

Ahora tenemos que crear a nuestros personajes que recorrerán el mapa, los Dinos… para esto contamos con 4 diferentes estilos de Dinos por lo tanto crearemos 4 prefabs.

Seleccionamos los sprites que conformaran nuestra animación.

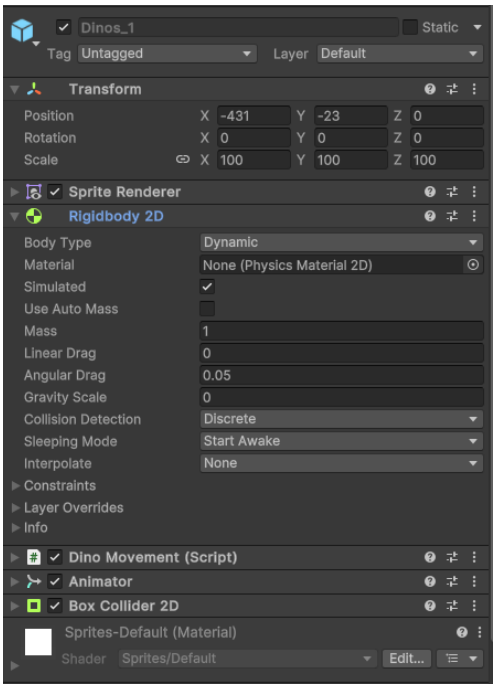


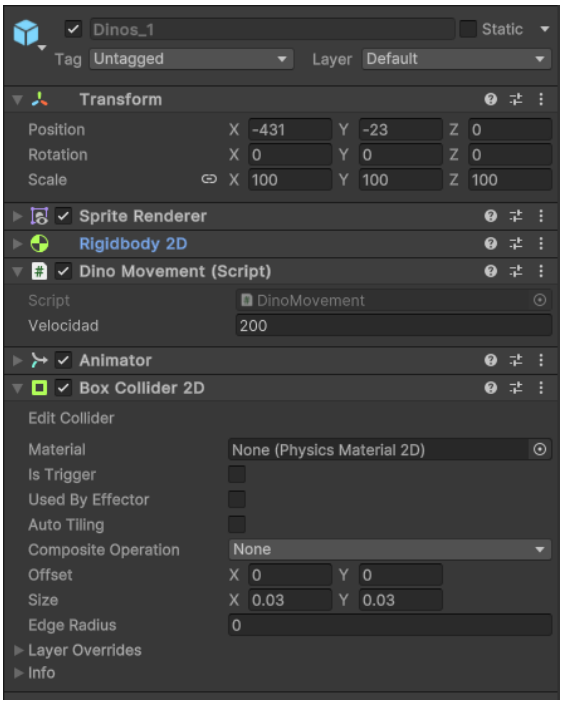
Guarda la animación dentro de una carpeta de Animations, si no existe créala.



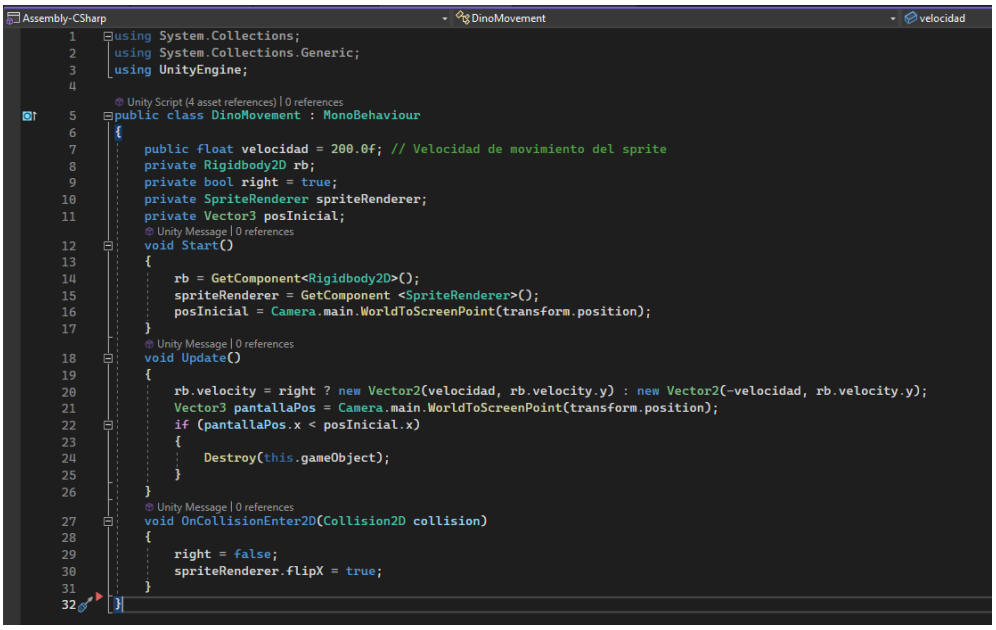
Al hacer esto nuestro personaje siempre se percibe en animación de movimiento.

Una de las partes importantes es darle los valores correctos, todos los modelos deben de mantener la mayor similitud posible ya que compartirán código de ejecución.



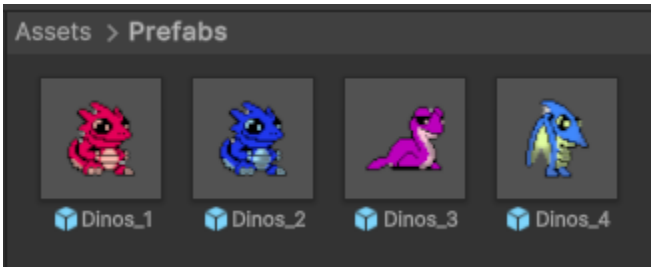


En este momento seguramente te preguntes en que momento creamos un script para nuestros Prefabs, ahora agregaremos nuestro script llamado “DinoMovement”. Y se lo asignamos a cada uno de nuestros prefabs, con esos valores que vimos anteriormente.



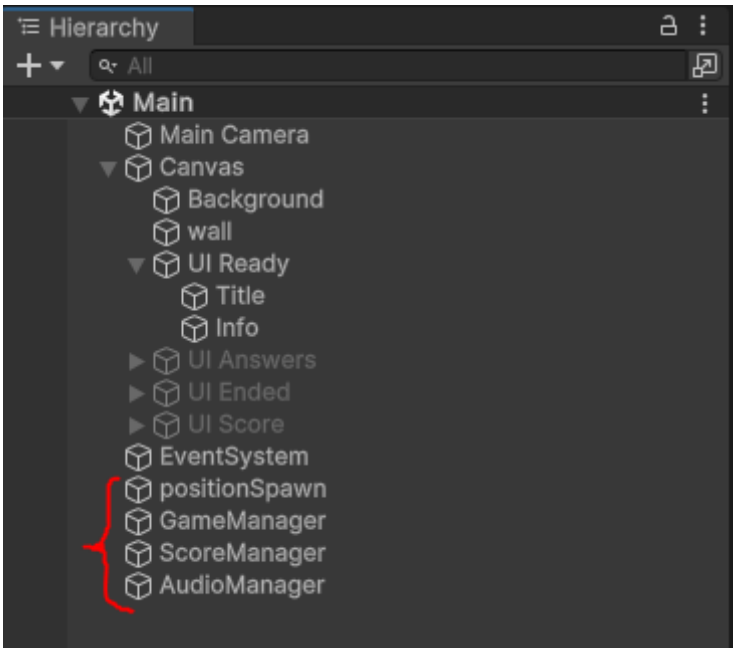
Este es el código, como observamos al final contiene una función en caso de colisiones, es por esto que creamos el primer wall en su momento, para que nuestros Dinos chocaran con él. El código simplemente hace que avancen a la derecha hasta chocar, regresar y al salir de la pantalla autodestruirse.

En este momento ya debemos tener nuestros prefabs.



Ahora que entraremos en los códigos habrá situaciones donde no podrá ejecutarse hasta haber completado varios códigos, ya que dependen unos de otros. No olvides hacer las referencias que se mencionan aquí de las variables públicas.

Estos son los objetos vacíos que crearemos para referenciarles un script solamente.

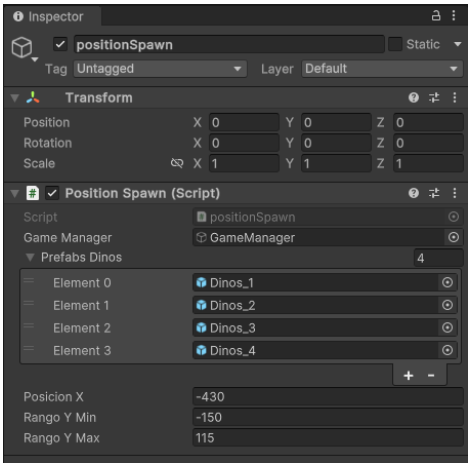


Puedes crearlos de una vez para no olvidarlos.

Empezamos por el primero, positionSpawn se encargará de alinear los prefabs de Dinos dentro del espacio para que estén bien distribuidos sin que se empalmen unos con otros.

```
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.UIElements;

Unity Script (1 asset reference) | 0 references
public class positionSpawn : MonoBehaviour
{
    public GameObject gameManager;
    private GameManager gameManagerScript;
    public GameObject[] prefabsDinos;
    public float posicionX, rangoYMin, rangoYMax;
    private float manyDinos;
    Unity Message | 0 references
    void Start()
    {
        gameManagerScript = gameManager.GetComponent<GameManager>();
    }
    Unity Message | 0 references
    void Update()
    {
        SpawnDinos();
    }
    1 reference
    void SpawnDinos()
    {
        if (gameManagerScript.gameState == GameState.StartRun)
        {
            manyDinos = UnityEngine.Random.Range(1, 6);
            gameManagerScript.correct = (int)manyDinos;
            float[] listY = new float[(int)manyDinos];
            int contadorSprite = 0;
            for (int i = 0; i < manyDinos; i++)
            {
                rangoYMax = (contadorSprite == 3) ? 130 : 0;
                rangoYMin = (contadorSprite == 3) ? 0 : -130;

                float posicionY;
                do
                {
                    posicionY = Mathf.Round(UnityEngine.Random.Range(rangoYMin, rangoYMax) / 20f) * 20f;
                } while (listY.Contains(posicionY));

                listY[i] = posicionY;
                Vector3 setPosition = new Vector3(posicionX, posicionY, 0f);
                GameObject instanciaPrefab = Instantiate(prefabsDinos[contadorSprite], setPosition, Quaternion.identity);

                contadorSprite = (contadorSprite != 3) ? contadorSprite + 1 : 0;
            }
            gameManagerScript.gameState = GameState.Running;
            StartCoroutine(EsperarYCambiarEstado());
        }
    }
    1 reference
    IEnumerator EsperarYCambiarEstado()
    {
        yield return new WaitForSeconds(8);//Espera 8 segundos.
        gameManagerScript.gameState = GameState.StartAnswer;
    }
}

```

En el inspector tenemos referenciado en un arreglo de 4 los distintos modelos de Dinos en Prefabs, lo único importante es que el ultimo Dino sea el **volador**, de ahí en fuera los demás no importa como estén ordenados. La referencia del GameManager la puedes hacer ya incluso cuando aún no está configurado el script del game manager, obvio no funcionara hasta que creemos los demás códigos.



Ahora mostrare el código de GameManager, se mostrará en partes para que sea más legible.

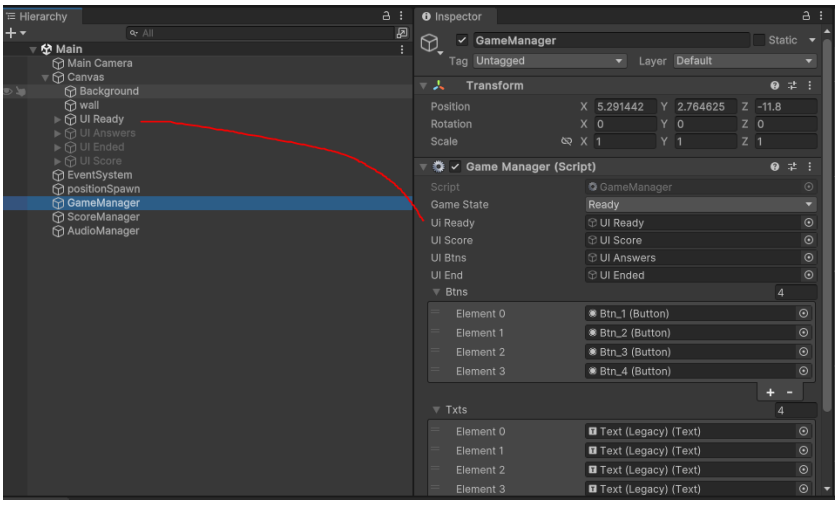
```
using NUnit.Framework;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.UIElements;
using Button = UnityEngine.UI.Button;

18 references
public enum GameState { Ready, StartRun, Running, StartAnswer,Answering, Lose,Ended };
Unity Script (1 asset reference) | 4 references
public class GameManager : MonoBehaviour
{
    public GameState gameState = GameState.Ready;
    public GameObject uiReady;
    public GameObject UIScore;
    public GameObject UIBtns;
    public GameObject UIEnd;
    public Button[] btns;
    public Text[] txts;
    public int correct;
    Unity Message | 0 references
    void Update()
    {
        ifStart();
        SpawnButtons();
        methodExit();
    }
    1 reference
    void ifStart()
    {
        if (gameState==GameState.Ready && Input.GetMouseButtonDown(0))
        {
            gameState = GameState.StartRun;
            uiReady.SetActive(false);
            UIScore.SetActive(true);
        }
        if (gameState == GameState.Ended && Input.GetMouseButtonDown(0))
        {
            gameState = GameState.StartRun;
            UIEnd.SetActive(false);
            AudioManager.Instance.PlayMusic();
        }
    }
```

```
    void SpawnButtons()
    {
        if (gameState == GameState.StartAnswer)
        {
            UIBtns.SetActive(true);
            foreach (Button btn in btns)
            {
                btn.interactable = true;
            }
            string [] listTxt = new string[4];
            // Seleccionar aleatoriamente el índice del botón correcto
            int indiceBotonCorrecto = Random.Range(0, btns.Length);
            txts[indiceBotonCorrecto].text = correct.ToString();
            btns[indiceBotonCorrecto].onClick.AddListener(BotonCorrecto);
            listTxt[indiceBotonCorrecto] = txts[indiceBotonCorrecto].text;

            for (int i = 0; i < btns.Length; i++)
            {
                btns[i].interactable = true;
                if (i != indiceBotonCorrecto)
                {
                    do
                    {
                        string randomNum;
                        if (correct<4)
                        {
                            randomNum = Random.Range(Mathf.Abs(correct - 2), correct + 4).ToString();
                        }
                        else if (correct>7)
                        {
                            randomNum = Random.Range(correct - 3, correct + 2).ToString();
                        }
                        else
                        {
                            randomNum = Random.Range(correct - 3, correct + 3).ToString();
                        }
                        txts[i].text = randomNum;
                    } while (listTxt.Contains(txts[i].text));
                    listTxt[i] = txts[i].text;
                    btns[i].onClick.AddListener(BotonIncorrecto);
                }
            }
            gameState = GameState.Answering;
        }else if (gameState == GameState.StartRun)
        {
            UIBtns.SetActive(false);
        }else if (gameState == GameState.Ended)
        {
            UIBtns.SetActive(false);
            UIEnd.SetActive(true);
        }
    }
```

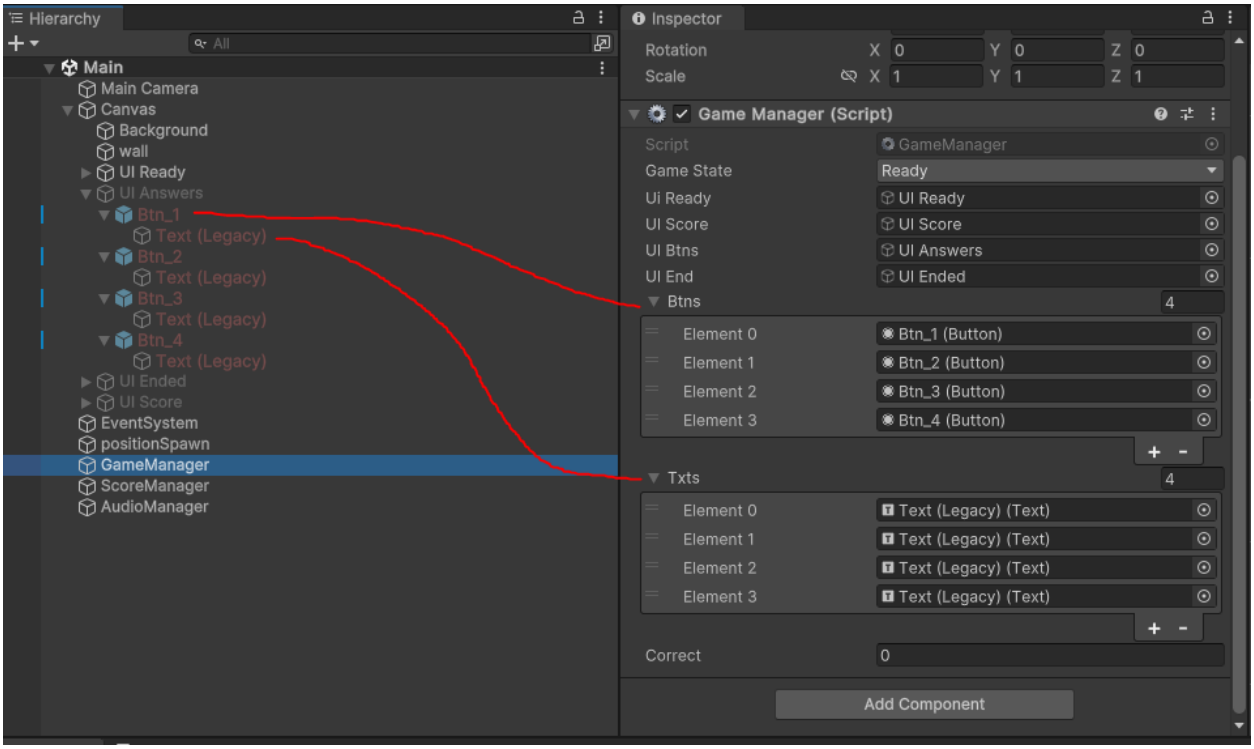
```
    void BotonCorrecto()
    {
        Debug.Log("¡Correcto!");
        foreach (Button btn in btns)
        {
            btn.onClick.RemoveAllListeners();
        }
        scoreManager.Instance.IncreasePoints();
        AudioManager.Instance.PlaySound("Point");
        gameState = GameState.StartRun;
    }
    1 reference
    void BotonIncorrecto()
    {
        Debug.Log("Incorrecto.");
        foreach (Button btn in btns)
        {
            btn.onClick.RemoveAllListeners();
        }
        AudioManager.Instance.PlaySound("spongebob-fail");
        AudioManager.Instance.StopMusic();
        gameState = GameState.Lose;
    }
    1 reference
    void methodExit()
    {
        if (Input.GetKeyDown("escape")) {
            Application.Quit();
            Debug.Log("Exit App");
        }
    }
}
```

Este es el código que cuenta con más referencias.

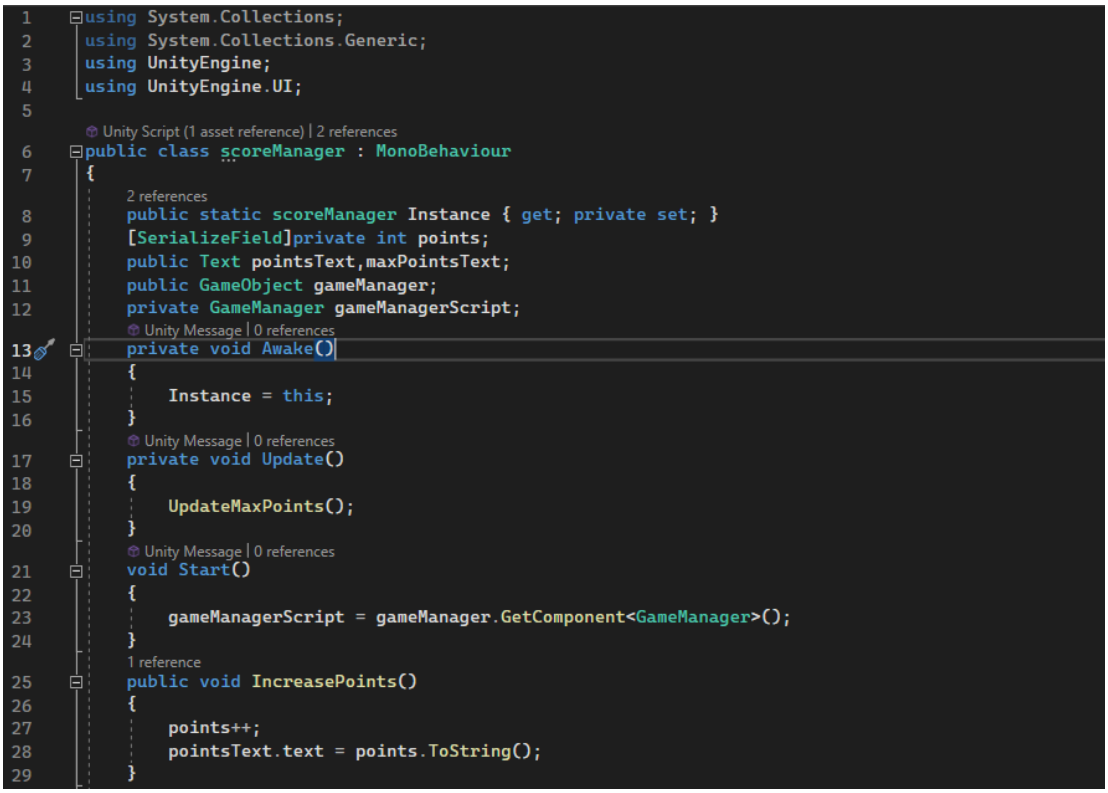
Aquí la primera referencia es de las carpetas de UI's que creamos al principio, se agrega cada una respectivamente en su nombre.



Aquí puede existir confusión… en la primera lista se agregan los botones tal cual en orden como se encuentran, y en el **MISMO ORDEN** debemos de agregar los Text de cada uno de los botones, esto nos servirá para modificar el texto de los botones.



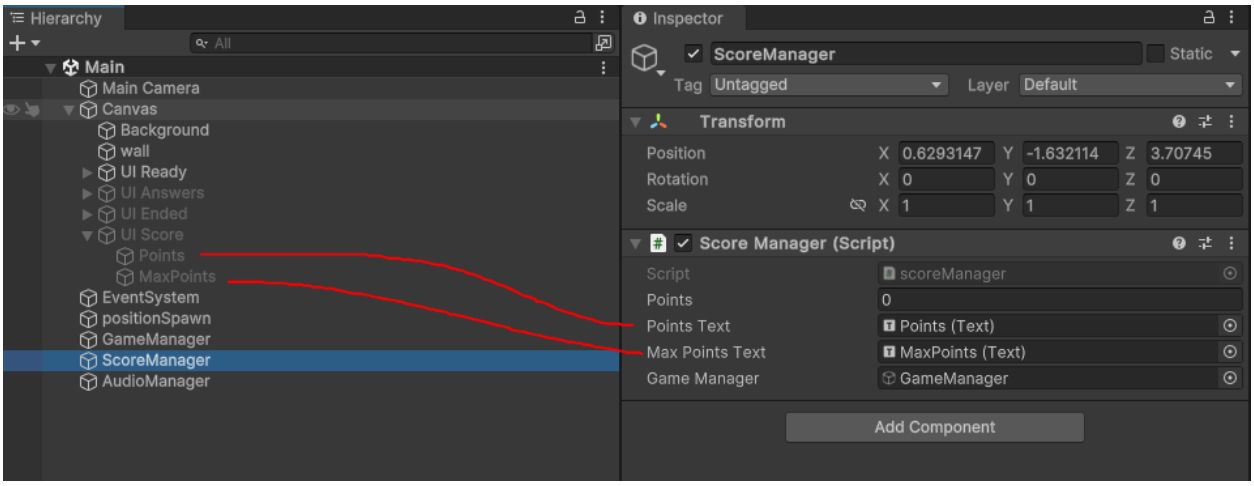
Proseguiremos con el siguiente Script que sería ScoreManager, con este tendremos aplicada la persistencia.



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

Unity Script (1 asset reference) | 2 references
public class scoreManager : MonoBehaviour
{
    2 references
    public static scoreManager Instance { get; private set; }
    [SerializeField]private int points;
    public Text pointsText,maxPointsText;
    public GameObject gameManager;
    private GameManager gameManagerScript;
    Unity Message | 0 references
    private void Awake()
    {
        Instance = this;
    }
    Unity Message | 0 references
    private void Update()
    {
        UpdateMaxPoints();
    }
    Unity Message | 0 references
    void Start()
    {
        gameManagerScript = gameManager.GetComponent<GameManager>();
    }
    1 reference
    public void IncreasePoints()
    {
        points++;
        pointsText.text = points.ToString();
    }
```

```
    1 reference
    public void UpdateMaxPoints()
    {
        if (gameManagerScript.gameState == GameState.Lose)
        {
            int maxPoints = PlayerPrefs.GetInt("Max", 0);
            if (points >= maxPoints)
            {
                maxPoints = points;
                PlayerPrefs.SetInt("Max", maxPoints);
            }
            maxPointsText.text = "BEST: " + maxPoints.ToString();
            points = 0;
            pointsText.text = points.ToString();
            gameManagerScript.gameState = GameState.Ended;
        }
        if (gameManagerScript.gameState == GameState.Ready)
        {
            int maxPoints = PlayerPrefs.GetInt("Max", 0);
            if (points >= maxPoints)
            {
                maxPoints = points;
                PlayerPrefs.SetInt("Max", maxPoints);
            }
            maxPointsText.text = "BEST: " + maxPoints.ToString();
            points = 0;
            pointsText.text = points.ToString();
        }
    }
}
```

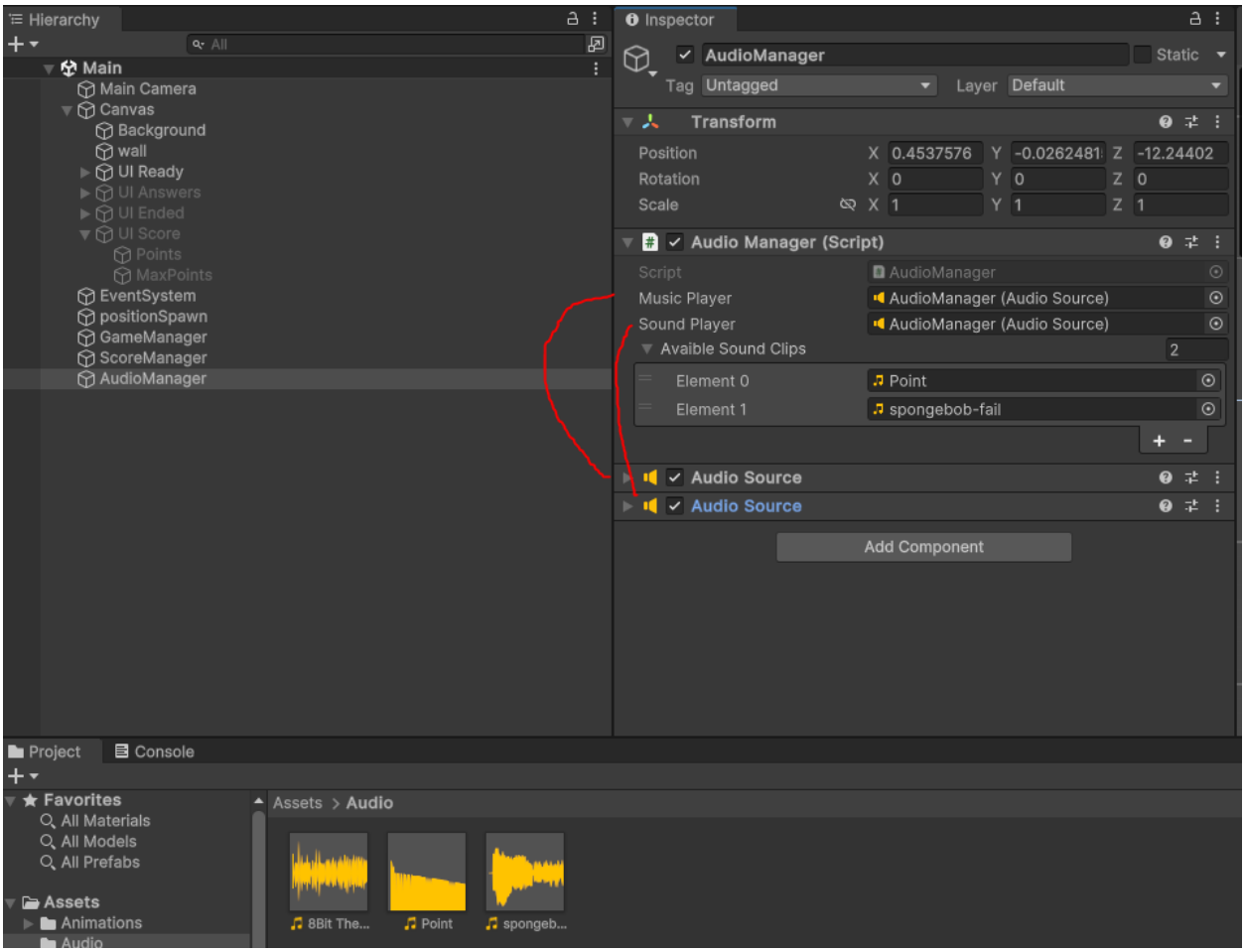
Ahora las referencias que necesita son más sencillas en este caso, y claro la referencia a GameManager como ya es habitual.

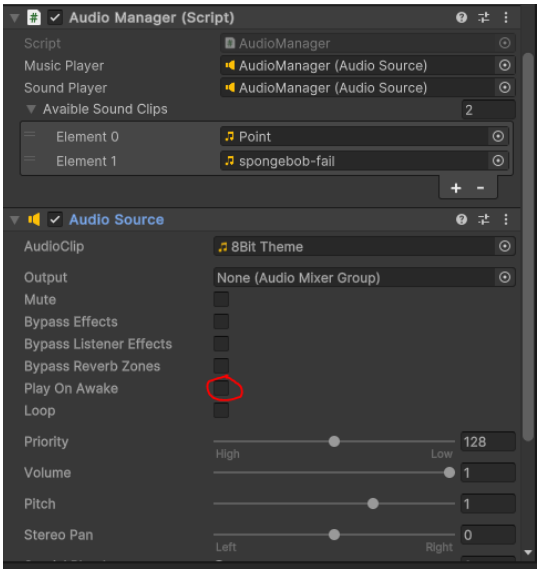


Ahora por último el código de AduioManager el cual es el más sencillo.

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

Unity Script (1 asset reference) | 5 references
public class AudioManager : MonoBehaviour
{
    5 references
    public static AudioManager Instance { get; private set; }
    public AudioSource musicPlayer, soundPlayer;
    public AudioClip[] avaibleSoundClips;
    private Dictionary<string, AudioClip> loadedAudioClips;
    Unity Message | 0 references
    private void Awake()
    {
        Instance = this;
    }
    Unity Message | 0 references
    void Start()
    {
        loadedAudioClips = new Dictionary<string, AudioClip>();
        foreach (AudioClip audio in avaibleSoundClips)
        {
            loadedAudioClips.Add(audio.name, audio);
        }
        PlayMusic();
    }
    2 references
    public void PlaySound(string name)
    {
        soundPlayer.clip = loadedAudioClips[name];
        soundPlayer.Play();
    }
    1 reference
    public void StopMusic()
    {
        musicPlayer.Stop();
    }
    2 references
    public void PlayMusic()
    {
        musicPlayer.loop = true;
        musicPlayer.Play();
    }
}
```

Las únicas referencias con las que cuenta con propiamente los archivos de audio, como observamos se tienen 2 Audio Source, el primero lo ocupara la MusicPlayer mientras que el segundo será el SoundPlayer, hay un arreglo el cual agregaremos los 2 sonidos que se usaran en este caso, un sonido para cuando se sume puntos y otro para cuando pierdas.





Tienes que tener desactivada esta opción al momento de agregar los AudioSource ya que nosotros los activaremos mediante código.