



"El saber de mis hijos
hará mi grandeza"

UNIVERSIDAD DE SONORA

División de Ingeniería

Departamento de Ingeniería Industrial

Videojuego educativo

Desarrollo de videojuegos 2

Grupo 1

Profesor

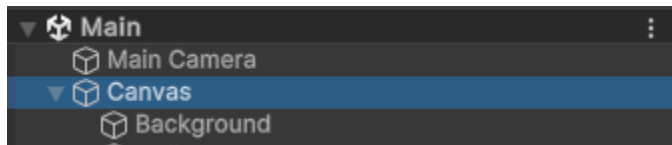
Álvaro Alfonso Valenzuela Figueroa

Alumno

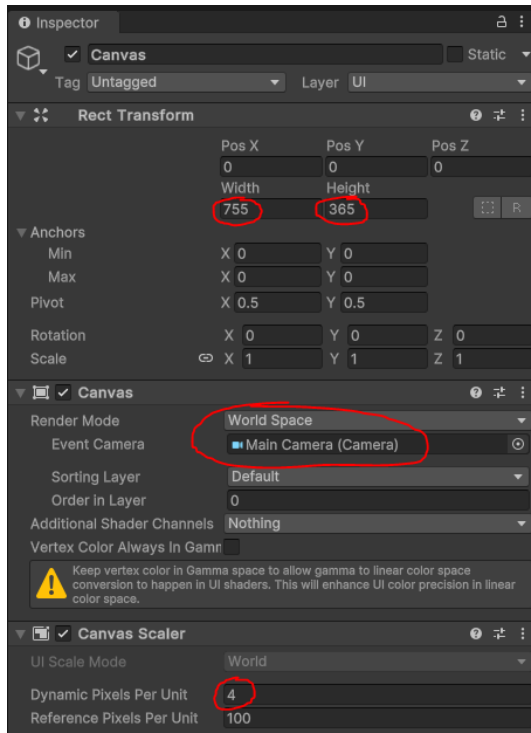
Chávez Pérez Javier Ariel

Hermosillo, Sonora a 23 de noviembre de 2023.

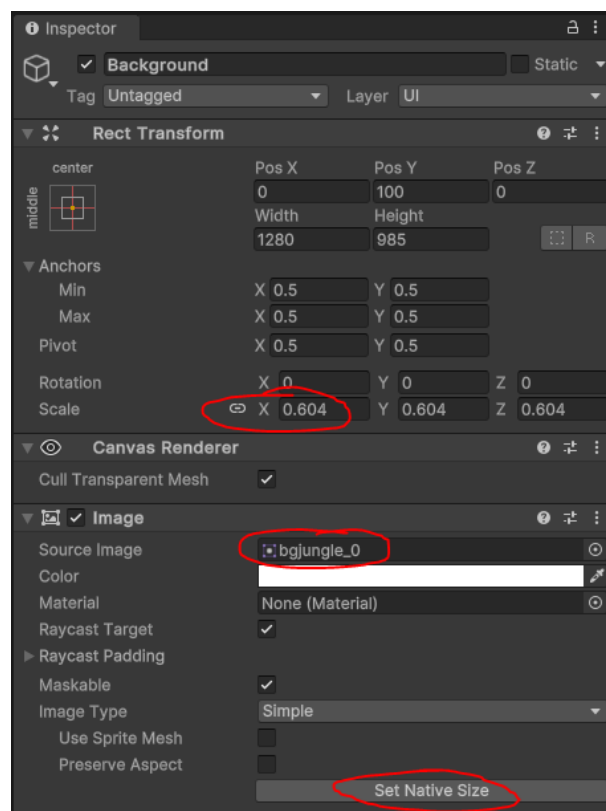
Despues de haber creado nuestro proyecto 2D en unity, creamos un canvas, dentro de este solamente agregamos un objeto imagen



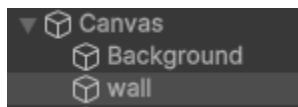
en el inspector del canvas modificamos estos valores.



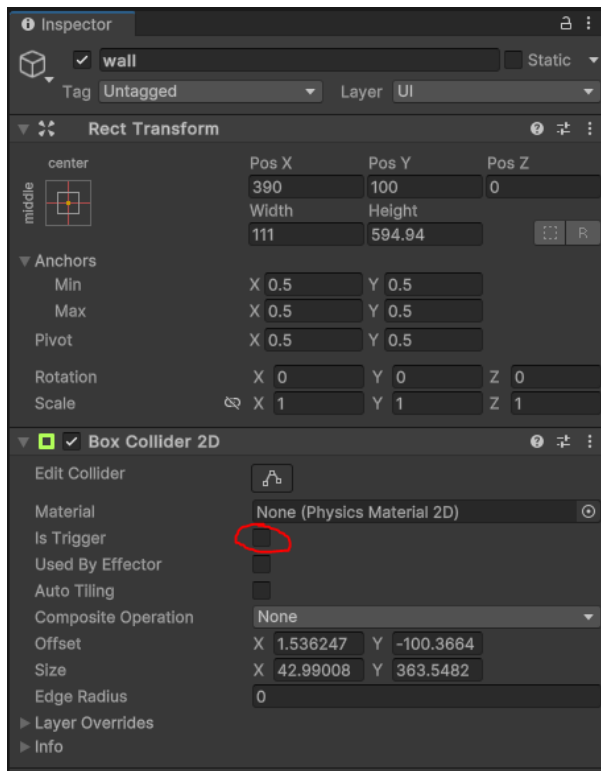
En el inspector de imagen estos valores, no olvides darle click en set native size



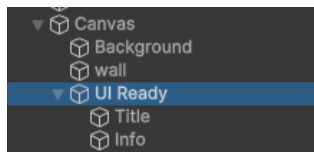
Despues ahí mismo en canvas creas un empty con el nombre de wall, esto nos servirá mas tarde para delimitar el recorrido de nuestros dinos.



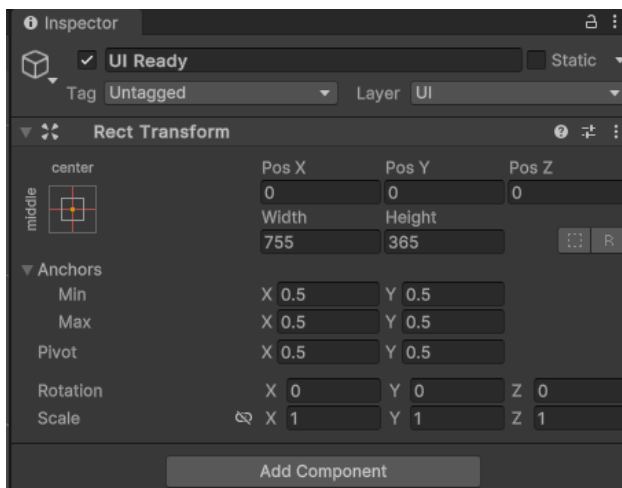
Despues crearemos un boxcollider para darle colision a este objeto que como su nombre lo indica será un muro delimitante para nosotros.(asegurate de que este desactivado trigger). se le agregan estos valores:



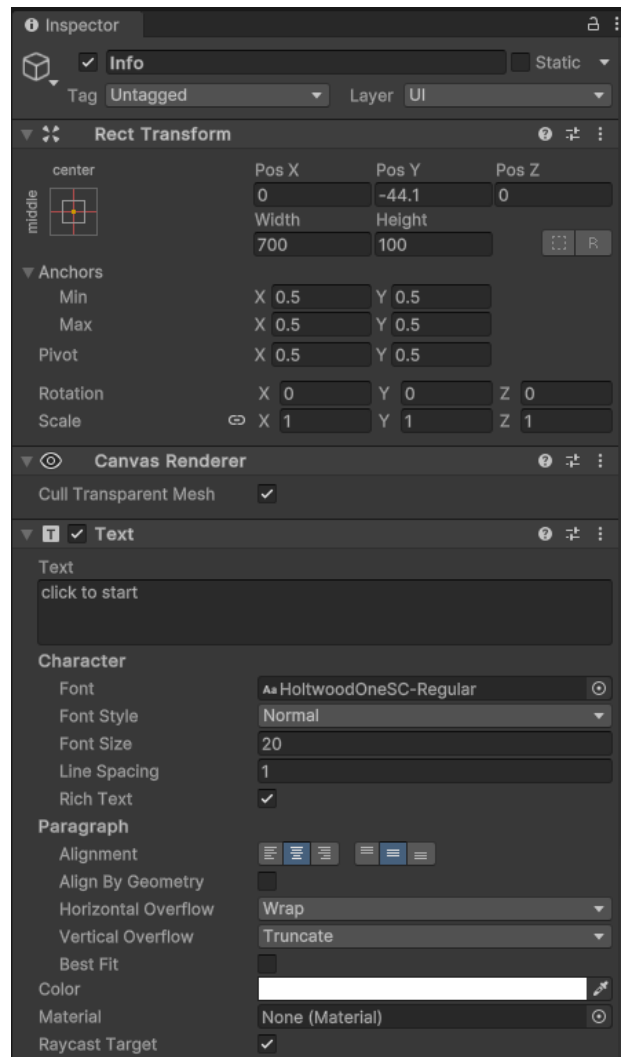
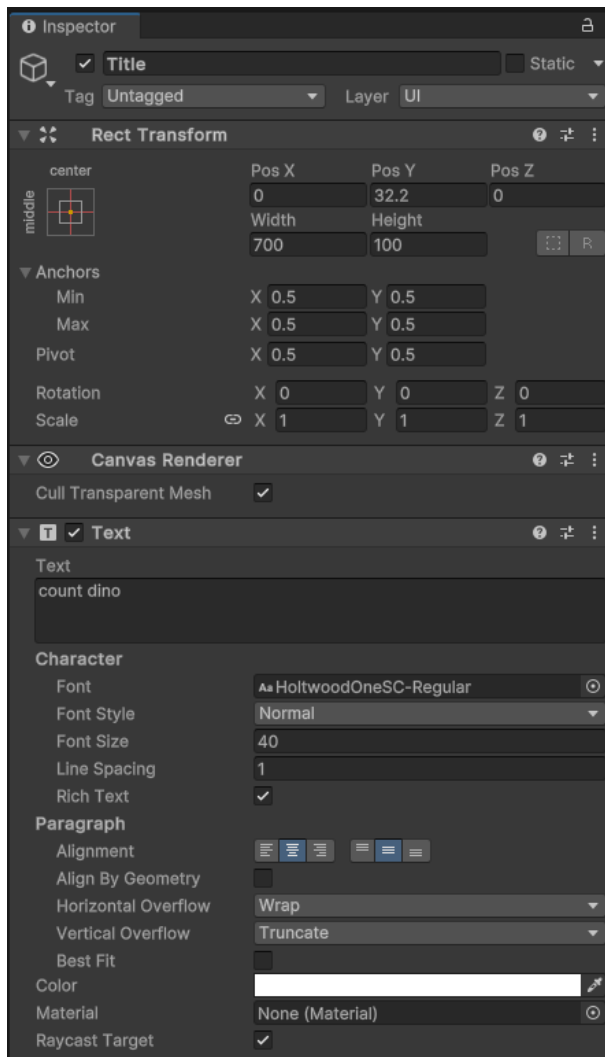
Debajo de esto pero aun dentro de la carpeta canvas creamos una subcarpeta llamada “ui ready” con 2 elementos tipo texto dentro con sus respectivos nombres



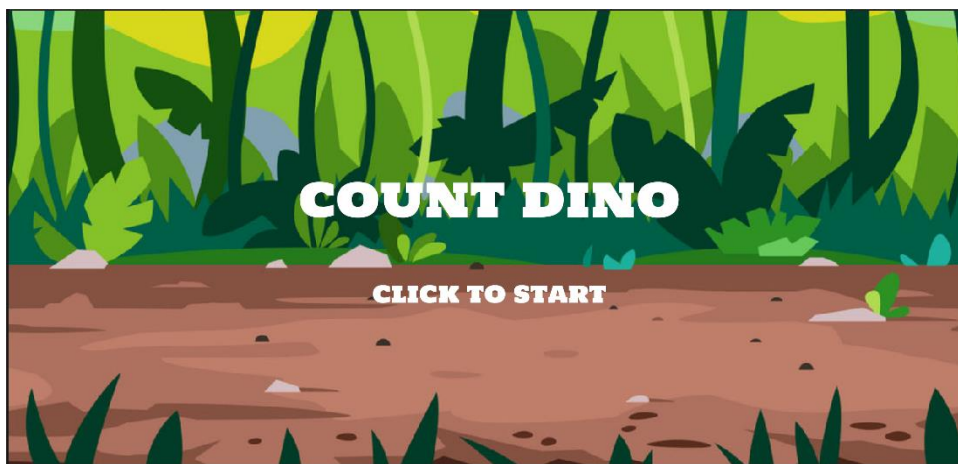
con estos valores:



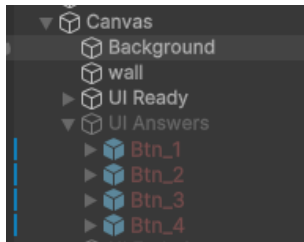
La configuración de ambas es la siguiente:



Esta es la pantalla que deberíamos tener:

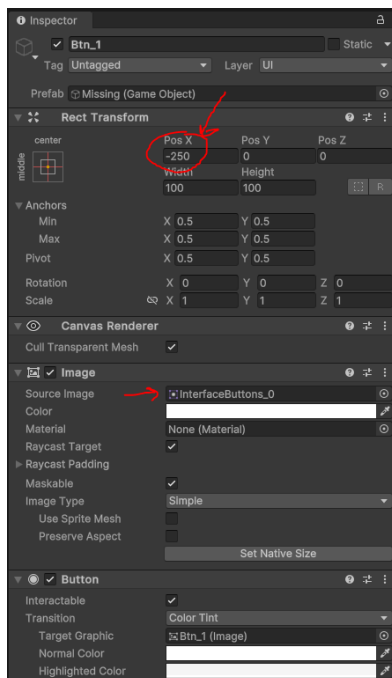


Aun dentro de canvas creas una carpeta extra llamada “ui answers”, ahí mismo creamos 4 botones.



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:

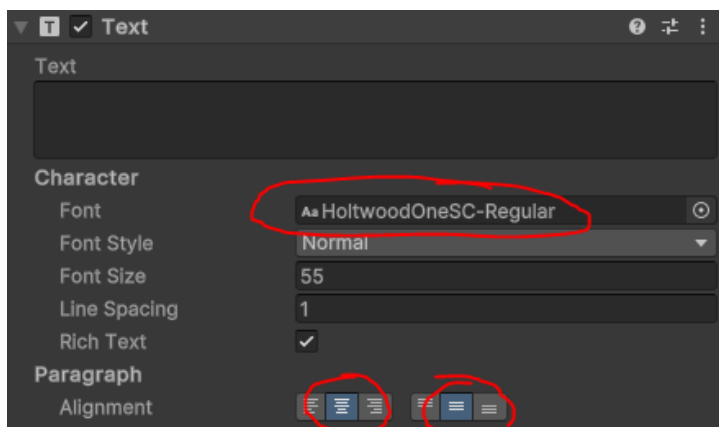
btn1 = -250 btn2=-75 btn3=75 btn4=250



Los colores y sombreado de los colores son libre, realmente eso no influye, solamente el tamaño para poder mantener un 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:



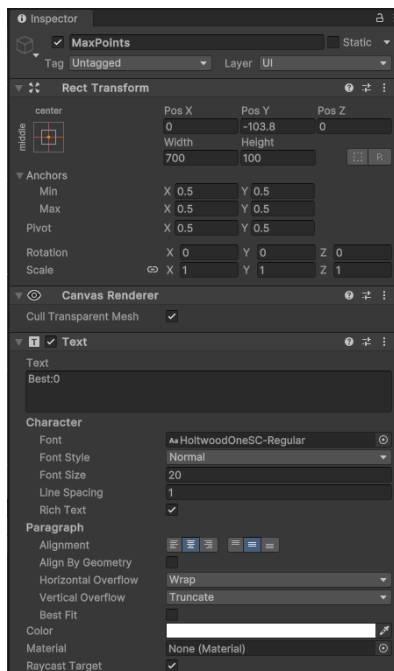
Creemos de nuevo una carpeta más llamada ahora “ui ended”, que fácilmente podemos duplicar la de ui ready, solamente cambiando los textos por otros, así lo haremos, esto tenemos esperado encontrar en el nuevo ui ended



Por última carpeta que crearemos en canvas esta es la nueva llamada ui score.



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

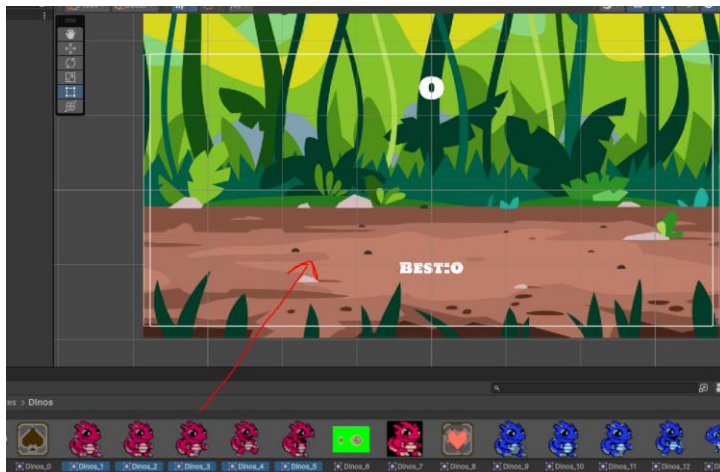


Así quedo la distribución deseada

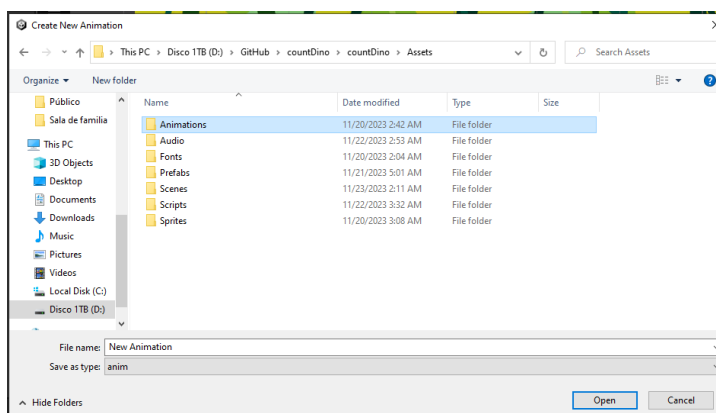


Ahora tenemos que crear a nuestros personajes que recorrerán el mapa, los Dinos... para esto contamos con 4 diferentes estilos de Dinos por lo tanto crearemos 4 prefabs.

Seleccionamos los sprites que conformaran nuestra animación.

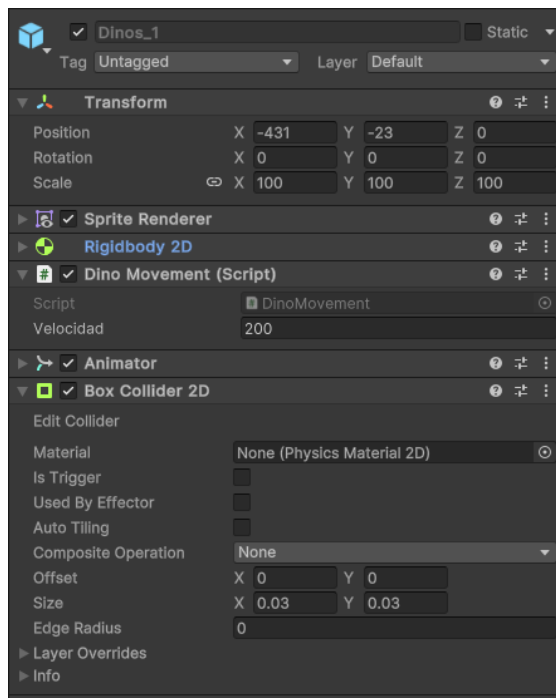
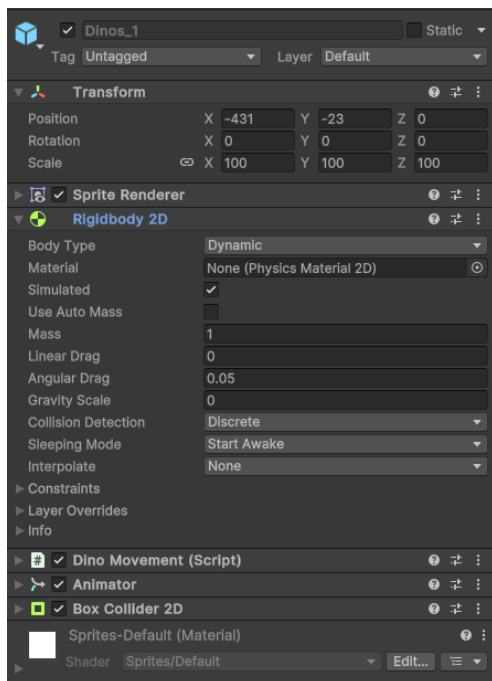


Guarda la animación dentro de una carpeta de Animations, si no existe créala.



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.

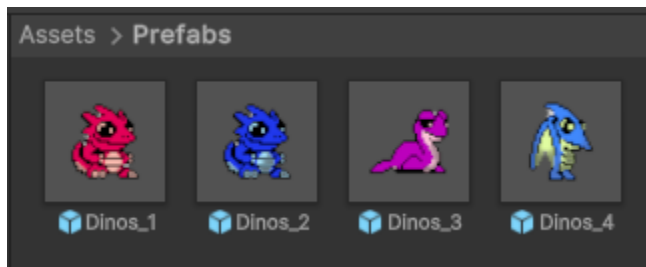
```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class DinoMovement : MonoBehaviour
6 {
7     public float velocidad = 200.0f; // Velocidad de movimiento del sprite
8     private Rigidbody2D rb;
9     private bool right = true;
10    private SpriteRenderer spriteRenderer;
11    private Vector3 posInicial;
12    @ Unity Message | 0 references
13    void Start()
14    {
15        rb = GetComponent<Rigidbody2D>();
16        spriteRenderer = GetComponent<SpriteRenderer>();
17        posInicial = Camera.main.WorldToScreenPoint(transform.position);
18    }
19    @ Unity Message | 0 references
20    void Update()
21    {
22        rb.velocity = right ? new Vector2(velocidad, rb.velocity.y) : new Vector2(-velocidad, rb.velocity.y);
23        Vector3 pantallaPos = Camera.main.WorldToScreenPoint(transform.position);
24        if (pantallaPos.x < posInicial.x)
25        {
26            Destroy(this.gameObject);
27        }
28    }
29    @ Unity Message | 0 references
30    void OnCollisionEnter2D(Collision2D collision)
31    {
32        right = false;
33        spriteRenderer.flipX = true;
34    }
35 }

```

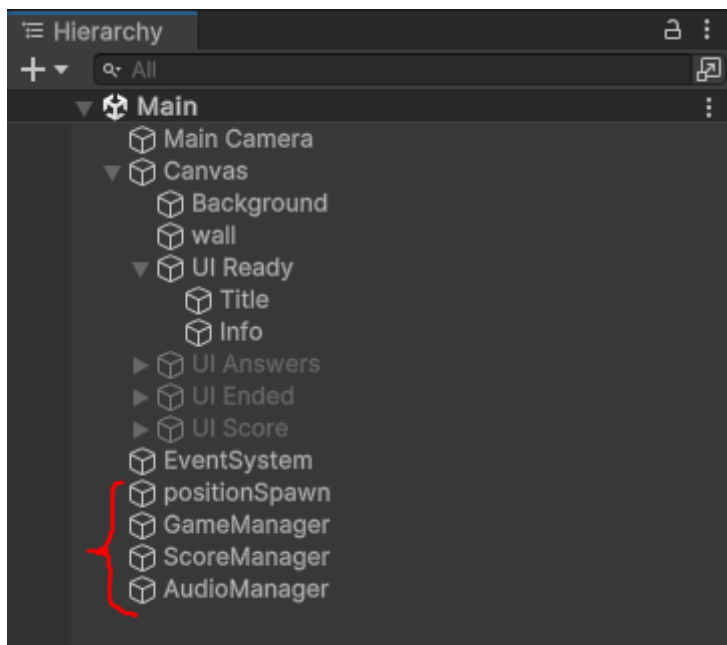
Este es el código, como observamos al final contiene una función en caso de colisiones, es por esto que creamos el primer wall en su momento, para que nuestros Dinos chocaran con el. El código simplemente hace que avancen a la derecha hasta chocar, regresar y al salir de la pantalla autodestruirse.

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.

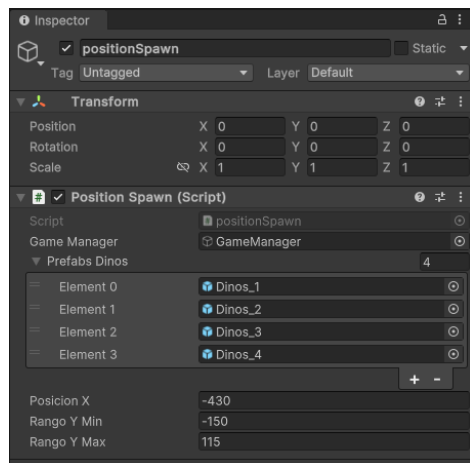


Puedes crearlos de una vez para no olvidarlos.

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

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using System.Linq;
4 using UnityEngine;
5 using UnityEngine.UI;
6 using UnityEngine.UIElements;
7
8 public class positionSpawn : MonoBehaviour
9 {
10     public GameObject gameManager;
11     private GameManager gameManagerScript;
12     public GameObject[] prefabsDinos;
13     public float posicionX, rangoYMin, rangoYMax;
14     private float manyDinos;
15     void Start()
16     {
17         gameManagerScript = gameManager.GetComponent<GameManager>();
18     }
19     void Update()
20     {
21         SpawnDinos();
22     }
23     void SpawnDinos()
24     {
25         if (gameManagerScript.gameState == GameState.StartRun)
26         {
27             manyDinos = UnityEngine.Random.Range(1, 6);
28             gameManagerScript.correct = (int)manyDinos;
29             float[] listY = new float[(int)manyDinos];
30             int contadorSprite = 0;
31             for (int i = 0; i < manyDinos; i++)
32             {
33                 rangoYMax = (contadorSprite == 3) ? 130 : 0;
34                 rangoYMin = (contadorSprite == 3) ? 0 : -130;
35
36                 float posicionY;
37                 do
38                 {
39                     posicionY = Mathf.Round(UnityEngine.Random.Range(rangoYMin, rangoYMax) / 20f) * 20f;
40                 } while (listY.Contains(posicionY));
41
42                 listY[i] = posicionY;
43                 Vector3 setPosition = new Vector3(posicionX, posicionY, 0f);
44                 GameObject instanciaPrefab = Instantiate(prefabsDinos[contadorSprite], setPosition, Quaternion.identity);
45
46                 contadorSprite = (contadorSprite != 3) ? contadorSprite + 1 : 0;
47             }
48             gameManagerScript.gameState = GameState.Running;
49             StartCoroutine(EsperarYCambiarEstado());
50         }
51     }
52     IEnumerator EsperarYCambiarEstado()
53     {
54         yield return new WaitForSeconds(8); //Espera 8 segundos.
55         gameManagerScript.gameState = GameState.StartAnswer;
56     }
57 }
58
```

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 aun no esta configurado el script del game manager, obvio no funcionara hasta que creamos los demás codigos



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

```

1  using NUnit.Framework;
2  using System.Collections;
3  using System.Collections.Generic;
4  using System.Linq;
5  using UnityEngine;
6  using UnityEngine.UI;
7  using UnityEngine.UIElements;
8  using Button = UnityEngine.UI.Button;
9
10  18 references
11  public enum GameState { Ready, StartRun, Running, StartAnswer, Answering, Lose, Ended };
12  1 Unity Script (1 asset reference) | 4 references
13  public class GameManager : MonoBehaviour
14  {
15      public GameState gameState = GameState.Ready;
16      public GameObject uiReady;
17      public GameObject UIScore;
18      public GameObject UIBtns;
19      public GameObject UIEnd;
20      public Button[] btns;
21      public Text[] txts;
22      public int correct;
23      1 Unity Message | 0 references
24      void Update()
25      {
26          ifStart();
27          SpawnButtons();
28          methodExit();
29      }
30      1 reference
31      void ifStart()
32      {
33          if (gameState == GameState.Ready && Input.GetMouseButtonDown(0))
34          {
35              gameState = GameState.StartRun;
36              uiReady.SetActive(false);
37              UIScore.SetActive(true);
38          }
39          if (gameState == GameState.Ended && Input.GetMouseButtonDown(0))
40          {
41              gameState = GameState.StartRun;
42              UIEnd.SetActive(false);
43              AudioManager.Instance.PlayMusic();
44          }
45      }
46  }

```

```

42 void SpawnButtons()
43 {
44     if (gameState == GameState.StartAnswer)
45     {
46         UIBtns.SetActive(true);
47         foreach (Button btn in btns)
48         {
49             btn.interactable = true;
50         }
51         string [] listTxt = new string[4];
52         // Seleccionar aleatoriamente el indice del botón correcto
53         int indiceBotonCorrecto = Random.Range(0, btns.Length);
54         txts[indiceBotonCorrecto].text = correct.ToString();
55         btns[indiceBotonCorrecto].onClick.AddListener(BotonCorrecto);
56         listTxt[indiceBotonCorrecto] = txts[indiceBotonCorrecto].text;
57
58         for (int i = 0; i < btns.Length; i++)
59         {
60             btns[i].interactable = true;
61             if (i != indiceBotonCorrecto)
62             {
63                 do
64                 {
65                     string randomNum;
66                     if (correct < 4)
67                     {
68                         randomNum = Random.Range(Mathf.Abs(correct - 2), correct + 4).ToString();
69                     }
70                     else if (correct > 7)
71                     {
72                         randomNum = Random.Range(correct - 3, correct + 2).ToString();
73                     }
74                     else
75                     {
76                         randomNum = Random.Range(correct - 3, correct + 3).ToString();
77                     }
78                     txts[i].text = randomNum;
79                 } while (listTxt.Contains(txts[i].text));
80                 listTxt[i] = txts[i].text;
81                 btns[i].onClick.AddListener(BotonIncorrecto);
82             }
83         }
84         gameState = GameState.Answering;
85     } else if (gameState == GameState.StartRun)
86     {
87         UIBtns.SetActive(false);
88     } else if (gameState == GameState.Ended)
89     {
90         UIBtns.SetActive(false);
91         UIEnd.SetActive(true);
92     }
93 }

```

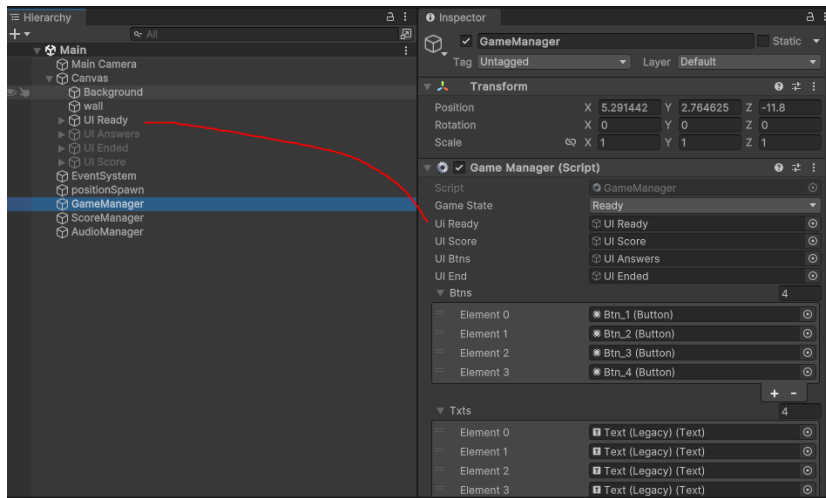
```

94 void BotonCorrecto()
95 {
96     Debug.Log("¡Correcto!");
97     foreach (Button btn in btns)
98     {
99         btn.onClick.RemoveAllListeners();
100     }
101     scoreManager.Instance.IncreasePoints();
102     AudioManager.Instance.PlaySound("Point");
103     gameState = GameState.StartRun;
104 }
1 reference
105 void BotonIncorrecto()
106 {
107     Debug.Log("Incorrecto.");
108     foreach (Button btn in btns)
109     {
110         btn.onClick.RemoveAllListeners();
111     }
112     AudioManager.Instance.PlaySound("spongebob-fail");
113     AudioManager.Instance.StopMusic();
114     gameState = GameState.Lose;
115 }
1 reference
116 void methodExit()
117 {
118     if (Input.GetKeyDown("escape")) {
119         Application.Quit();
120         Debug.Log("Exit App");
121     }
122 }
123 }

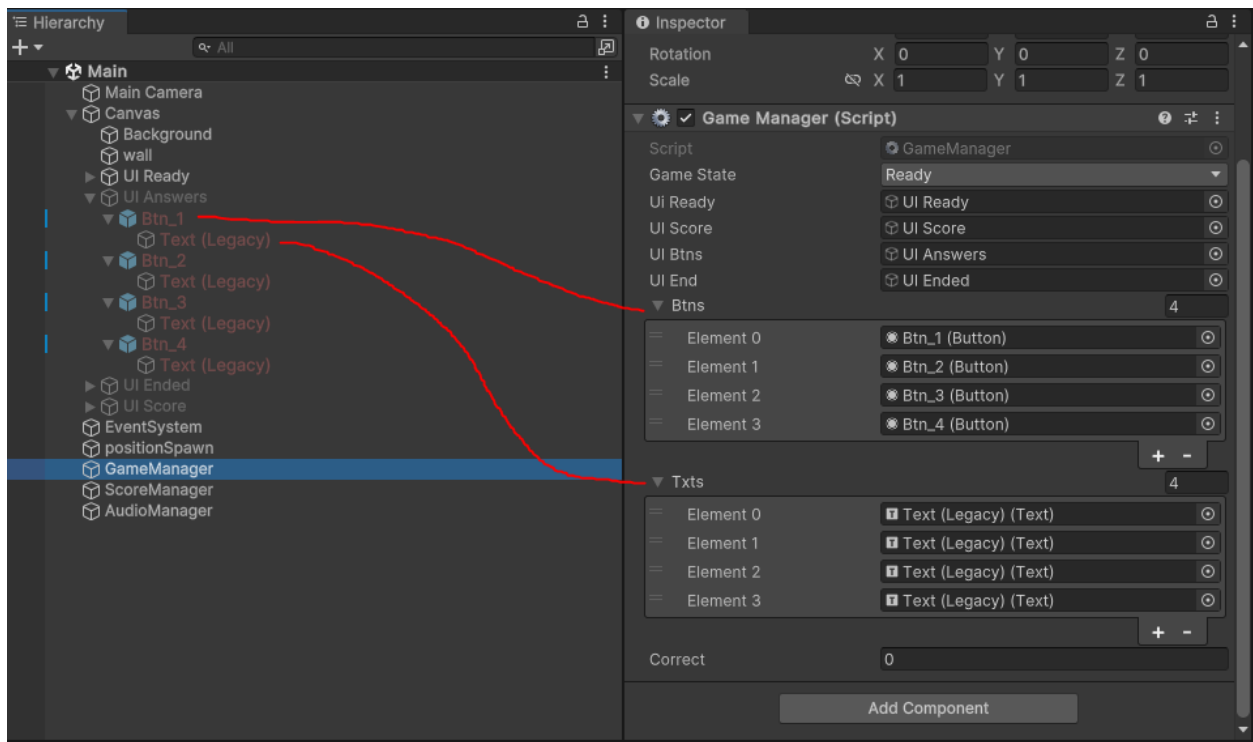
```

Este es el código que cuenta con mas referencias.

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



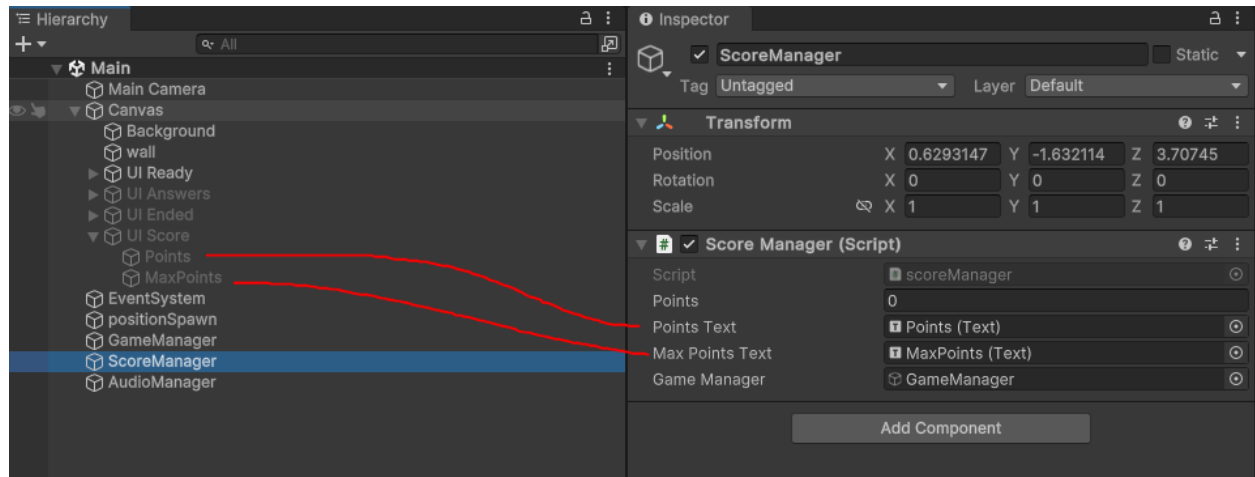
Aquí puede existir confusión... en la primera lista se agregan los botones tal cual en orden como se encuentran, y en el **MISMO ORDEN** debemos de agregar los Text de cada uno de los botones, esto nos servirá para modificar el texto de los botones.



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

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5
6  [Unity Script (1 asset reference) | 2 references]
7  public class ScoreManager : MonoBehaviour
8  {
9      2 references
10     public static ScoreManager Instance { get; private set; }
11     [SerializeField] private int points;
12     public Text pointsText, maxPointsText;
13     public GameObject gameManager;
14     private GameManager gameManagerScript;
15
16     [Unity Message | 0 references]
17     private void Awake()
18     {
19         Instance = this;
20     }
21
22     [Unity Message | 0 references]
23     private void Update()
24     {
25         UpdateMaxPoints();
26     }
27
28     [Unity Message | 0 references]
29     void Start()
30     {
31         gameManagerScript = gameManager.GetComponent<GameManager>();
32     }
33
34     1 reference
35     public void IncreasePoints()
36     {
37         points++;
38         pointsText.text = points.ToString();
39     }
40
41     1 reference
42     public void UpdateMaxPoints()
43     {
44         if (gameManagerScript.gameState == GameState.Lose)
45         {
46             int maxPoints = PlayerPrefs.GetInt("Max", 0);
47             if (points >= maxPoints)
48             {
49                 maxPoints = points;
50                 PlayerPrefs.SetInt("Max", maxPoints);
51             }
52             maxPointsText.text = "BEST: " + maxPoints.ToString();
53             points = 0;
54             pointsText.text = points.ToString();
55             gameManagerScript.gameState = GameState.Ended;
56         }
57         if (gameManagerScript.gameState == GameState.Ready)
58         {
59             int maxPoints = PlayerPrefs.GetInt("Max", 0);
60             if (points >= maxPoints)
61             {
62                 maxPoints = points;
63                 PlayerPrefs.SetInt("Max", maxPoints);
64             }
65             maxPointsText.text = "BEST: " + maxPoints.ToString();
66             points = 0;
67             pointsText.text = points.ToString();
68         }
69     }
70 }
```

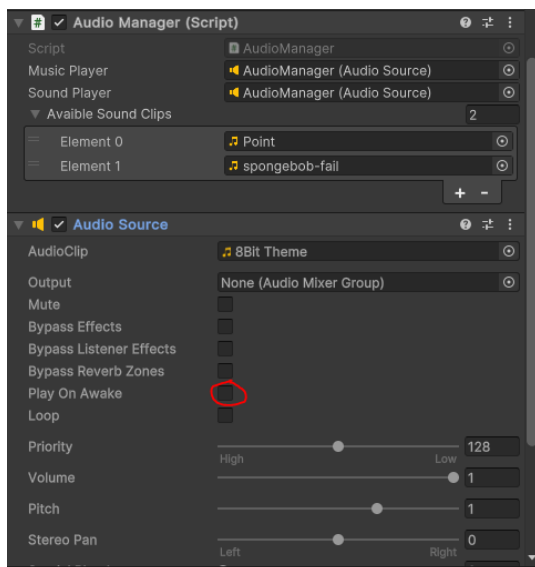
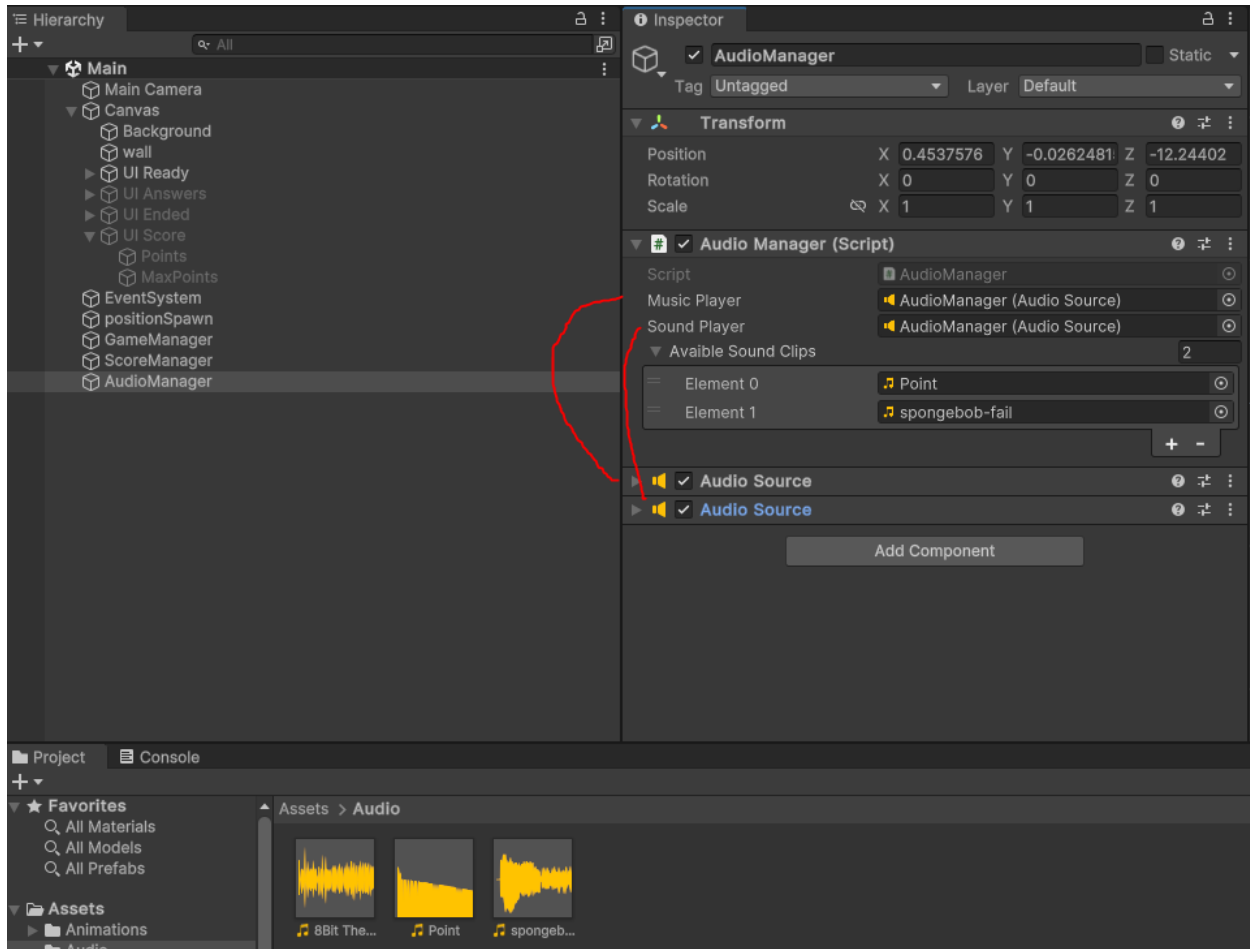

Ahora las referencias que necesita son más sencillas en este caso, y claro la referencia a GameManager como ya es habitual.



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

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class AudioManager : MonoBehaviour
6 {
7     public static AudioManager Instance { get; private set; }
8     public AudioSource musicPlayer, soundPlayer;
9     public AudioClip[] availableSoundClips;
10    private Dictionary<string, AudioClip> loadedAudioClips;
11    private void Awake()
12    {
13        Instance = this;
14    }
15    void Start()
16    {
17        loadedAudioClips = new Dictionary<string, AudioClip>();
18        foreach (AudioClip audio in availableSoundClips)
19        {
20            loadedAudioClips.Add(audio.name, audio);
21        }
22        PlayMusic();
23    }
24    public void PlaySound(string name)
25    {
26        soundPlayer.clip = loadedAudioClips[name];
27        soundPlayer.Play();
28    }
29    public void StopMusic()
30    {
31        musicPlayer.Stop();
32    }
33    public void PlayMusic()
34    {
35        musicPlayer.loop = true;
36        musicPlayer.Play();
37    }
38 }
39
```

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.