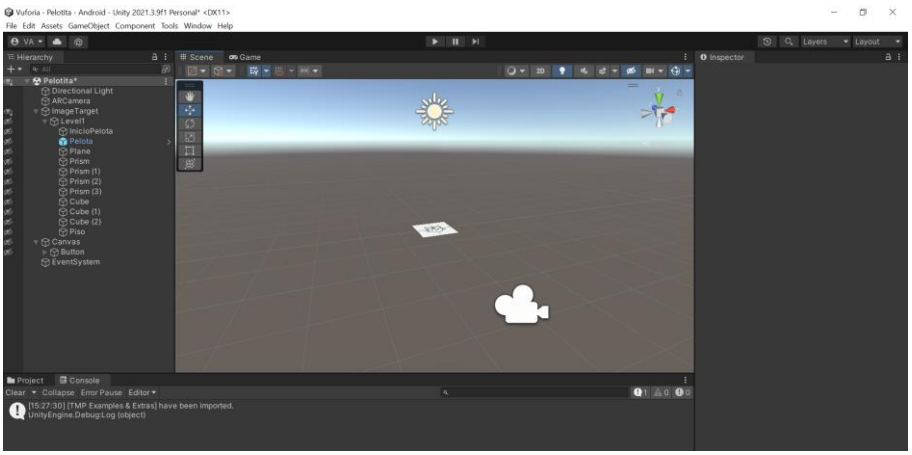
Mario Alberto Vásquez Cancino
315021963

Reporte

Juego de Realidad Aumentada con Vuforia

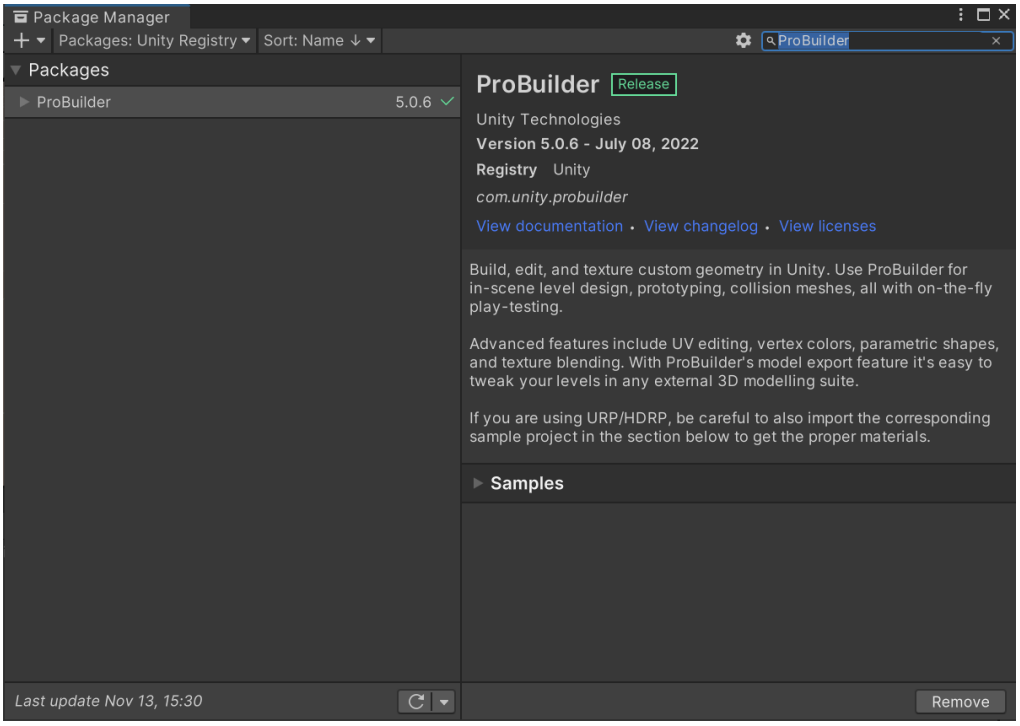
Para empezar, debemos crear un nuevo proyecto con Vuforia y realizar todo lo realizado en la práctica anterior:

- Crear Base de Datos en Vuforia.
- Crear Marcadores.
- Importar la Base de Datos a Unity.
- Poner la Cámara de Vuforia y el marcador.
- Todas las configuraciones pertinentes.



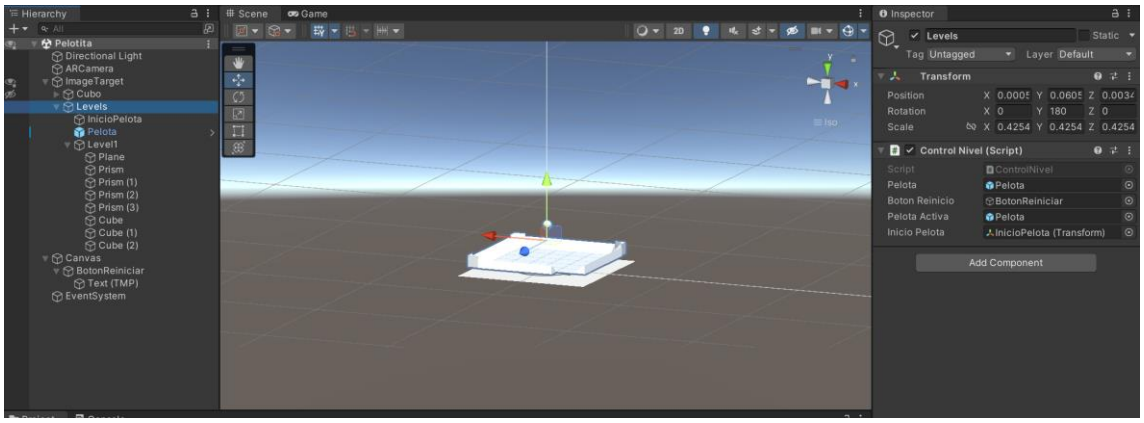
Posteriormente vamos a crear nuestro mapa de juego, para eso necesitaremos el paquete ProBuilder.

Nos vamos a Window -> Package Manager, buscamos ProBuilder y lo instalamos.



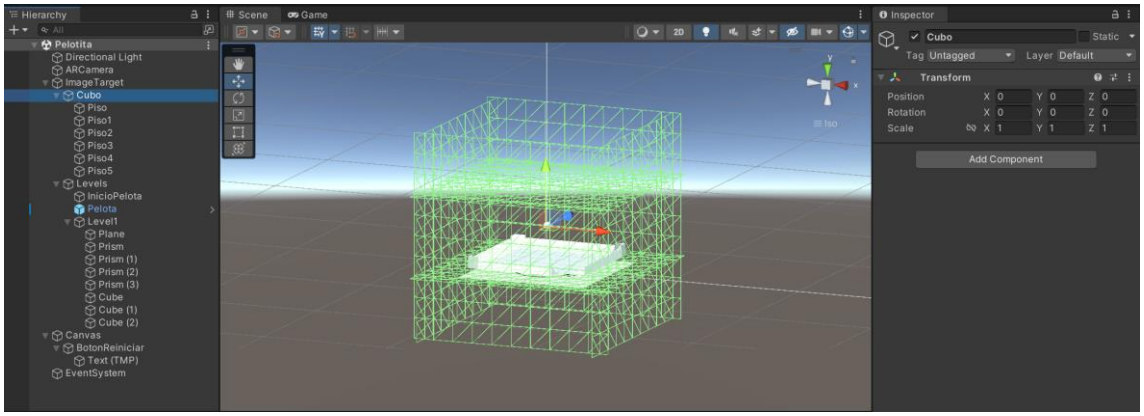
Mario Alberto Vásquez Cancino
315021963

Luego creamos nuestro tablero, pero ocupando los componentes de ProBuilder.



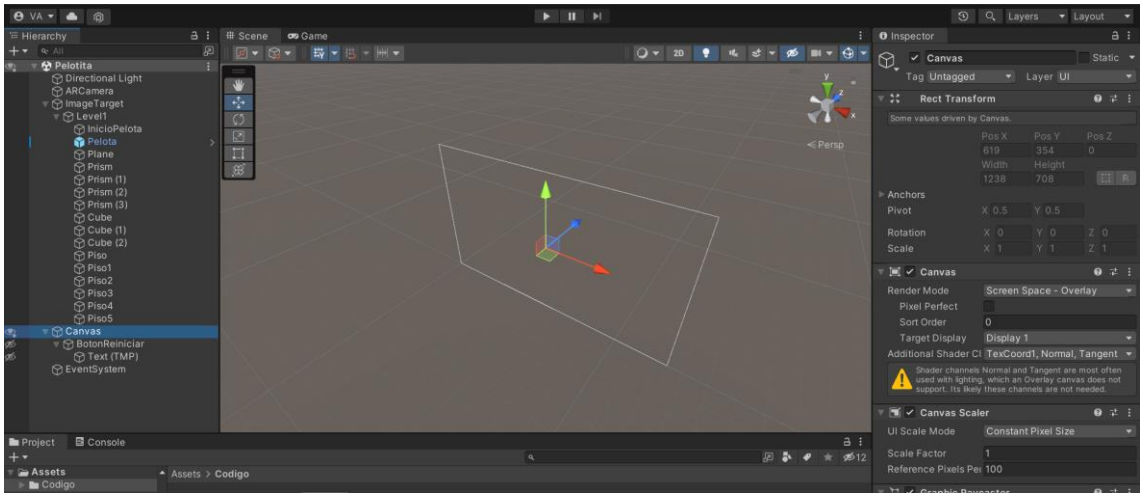
Como este va a ser un juego de varios niveles vamos hacer un objeto vacío nombrado “Levels” y hacemos todo nuestro tablero y pelota sus hijos para tenerlo más organizado, y también hacemos hijo a “Levels” del Target para que aparezca cuando encuentre el marcador. Finalmente hacemos un objeto vacio llamado “Level1” para guardar todo lo del nivel 1.

También vamos agregar a Levels un cubo hecho por planos transparentes que rodee el mapa, para facilidad vamos a poner los planos dentro de un objeto vacio como el “Levels” llamado “Cubo”. Para hacerlos transparentes vamos a quitar el componente “Mesh Renderer” de todos.



Antes de pasar al código vamos hacer el Canvas básico de la app.

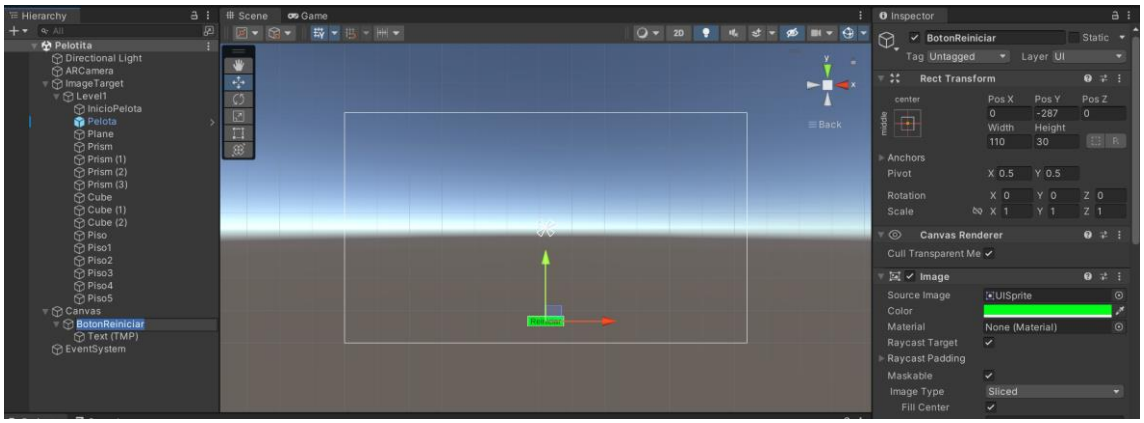
Para esto creamos un objeto de tipo Canvas.



Mario Alberto Vásquez Cancino
315021963

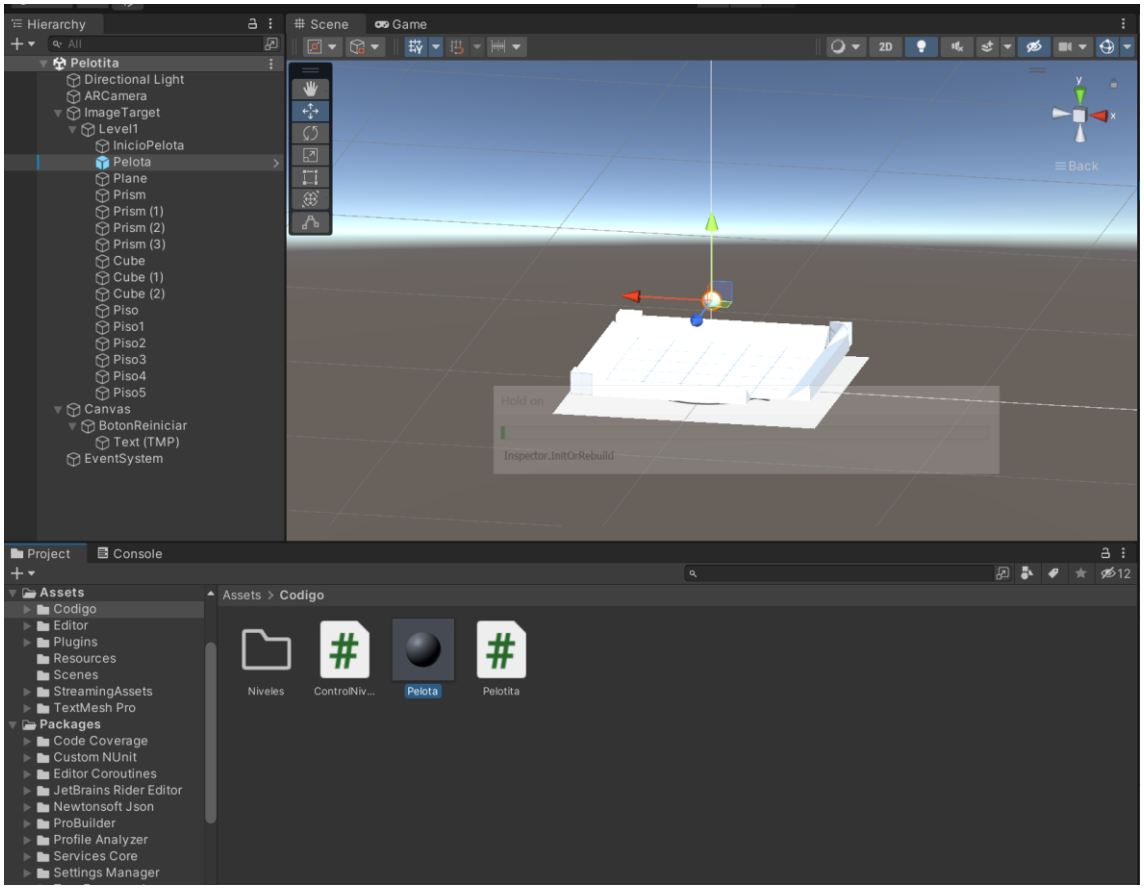
Este no lo modificamos ya que, así como esta se va a poner en la aplicación, solo tratar de poner todo en el centro.

Vamos a crear un botón “Reiniciar” que aparecerá cuando no esté la pelota para que vuelva a generar otra pelota y te reinicie desde el nivel 1 si es el caso.



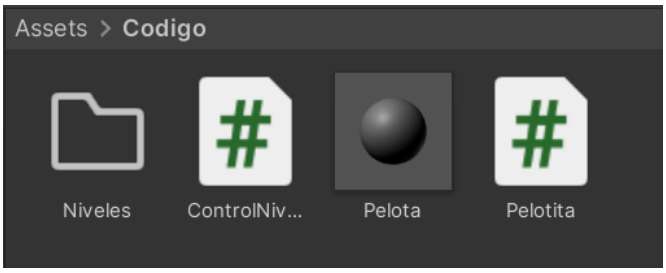
Para los códigos vamos a crear una carpeta llamada “Codigo” para poner todos los códigos que ocuparemos, y también haremos otra para poner los prefabs a ocupar.

Los prefabs son prototipos no hechos para que Unity pueda copiarlos cuando lo ocupemos. Para hacer uno debemos arrastrar nuestro modelo de la escena a Assets, para iniciar lo haremos con nuestra pelota, ya que vamos a generar varias durante el juego.



Mario Alberto Vásquez Cancino
315021963

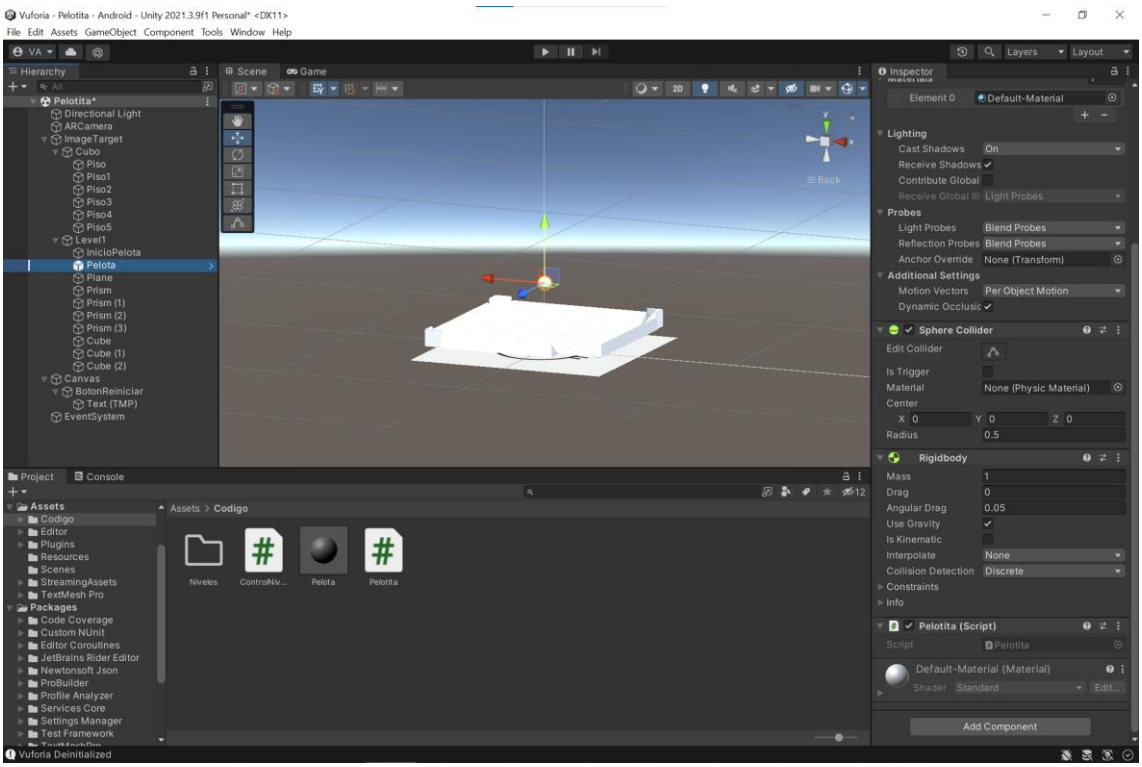
Ahora si desde Assets y nuestra carpeta para los códigos creamos dos scripts de C#, uno será para el control de la pelota (Pelotita.cs) y otro para controlar los niveles (ControlNivel.cs).



Abrimos el de Pelotita.cs y vamos a indicar como se muestra en la imagen que en cuanto colisione con cualquiera de los planos que rodean nuestro mapa se va a destruir.

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

Script de Unity (1 referencia de recurso) | 0 referencias
public class Pelotita : MonoBehaviour
{
    Mensaje de Unity | 0 referencias
    private void OnCollisionEnter(Collision collision)
    {
        if (collision.collider.name == "Piso" | collision.collider.name == "Piso1" | collision.collider.name == "Piso2" | collision.collider.name == "Piso3"
            | collision.collider.name == "Piso4" | collision.collider.name == "Piso5")
        {
            Object.Destroy(this.gameObject);
            //print("Se cayo");
        }
        // print("Choco");
    }
    // Start is called before the first frame update
    Mensaje de Unity | 0 referencias
    void Start()
    {

    }

    // Update is called once per frame
    Mensaje de Unity | 0 referencias
    void Update()
    {

    }
}
```

Para indicar a Unity que es código es de la Pelota, seleccionamos la Pelota y arrastramos el código desde Assets hasta los componentes de la pelota.



Ahora abrimos el script “ControlNivel.cs” y escribimos el siguiente código.

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

Script de Unity (1 referencia de recurso) | 0 referencias
public class ControlNivel : MonoBehaviour
{
    public GameObject pelota;
    public GameObject botonReinicio;
    public GameObject pelotaActiva;
    public Transform inicioPelota;

    // Start is called before the first frame update
    Mensaje de Unity | 0 referencias
    void Start()
    {
        botonReinicio.SetActive(false);
    }

    // Update is called once per frame
    Mensaje de Unity | 0 referencias
    void Update()
    {
        if(pelotaActiva == null)
        {
            perdio();
            //print("No hay pelota");
        }
    }

    0 referencias
    public void reinicio()
    {
        pelotaActiva = Instantiate(pelota, inicioPelota);
        botonReinicio.SetActive(false);

    }

    1 referencia
    public void perdio()
    {
        botonReinicio.SetActive(true);
    }
}
```

Tenemos 4 variables:

- Pelota: Guarda el prefab de la Pelota.
- botonReinicio: Guarda la referencia al botón.
- pelotaActiva: Referencia a que pelota esta activa en el momento del juego.
- inicioPelota: Guarda las coordenadas de inicio de la Pelota.

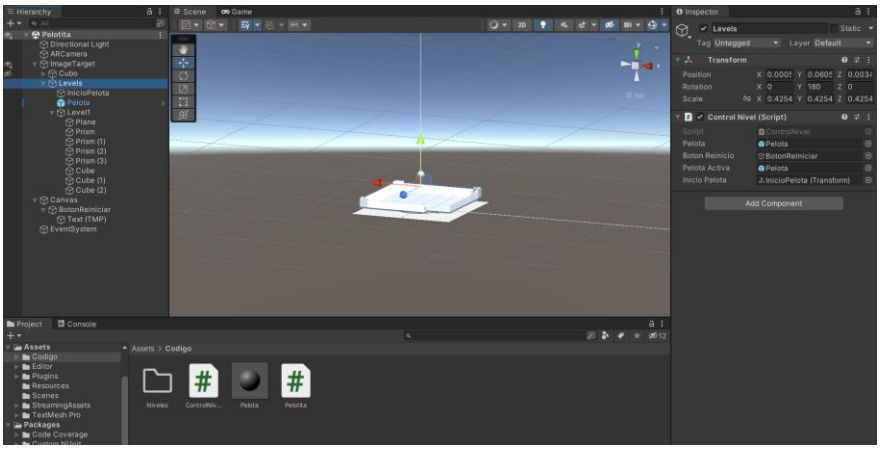
Start(): Al inicio del juego se desactiva el botón.

Update(): Si la pelota activa se destruye va a llamar a la función perdio().

reinicio(): Esta función crea una nueva pelota y oculta el botón de reinicio, esta diseñada para activarse cuando se de click en el botón.

perdio(): Esta función activa el botón de reinicio.

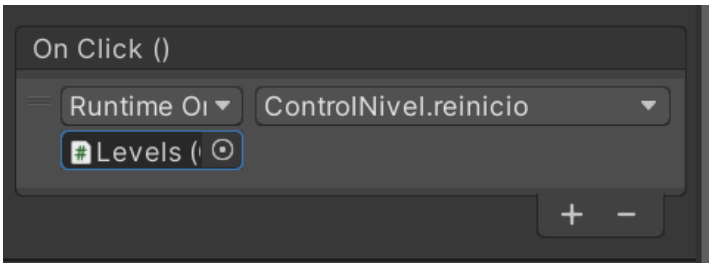
Ya con el código terminado lo vamos a relacionar con el objeto “Levels” ya que controlara los niveles.



Mario Alberto Vásquez Cancino
315021963

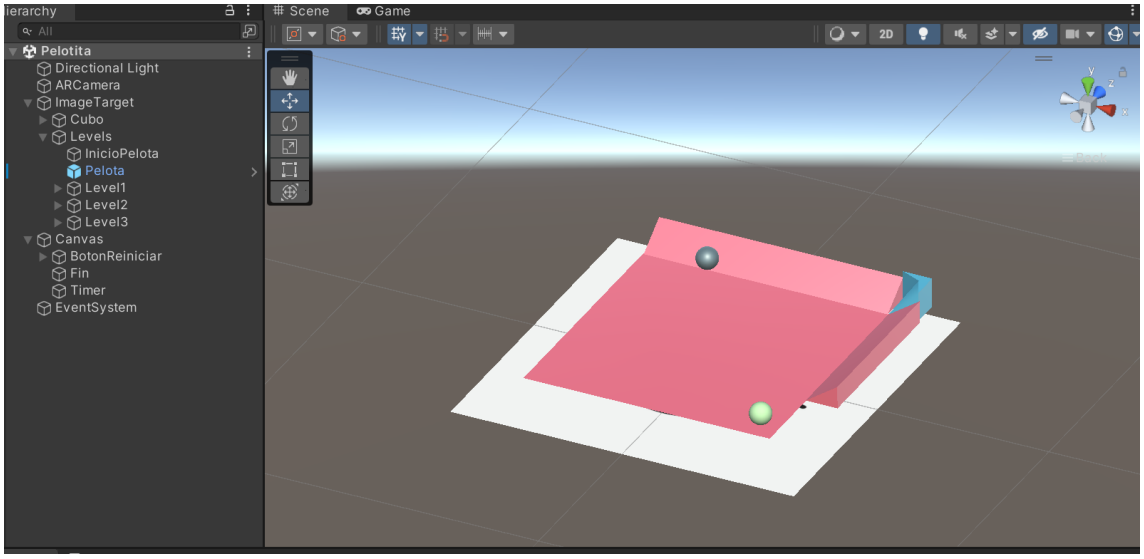
También vamos a darle la funcionalidad al botón con la función que creamos en el script.

En los componentes del botón buscamos la sección “On Click” donde agregaremos una función, y le indicamos que del objeto Levels agarre la función de reinicio que se encuentra en el archivo ControlNivel.cs que está relacionado con Levels.

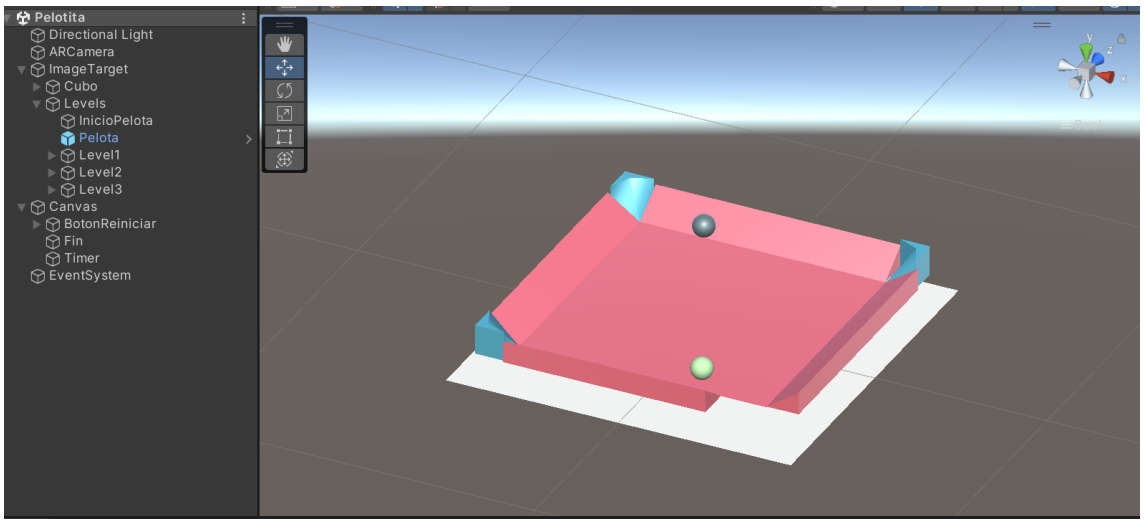


Con esto tenemos la base del juego, ya que cada vez que la pelota se caiga nos va aparecer el botón para volver a generar la pelota y volver a jugar, ahora vamos a crear los demás niveles.

Nivel 1

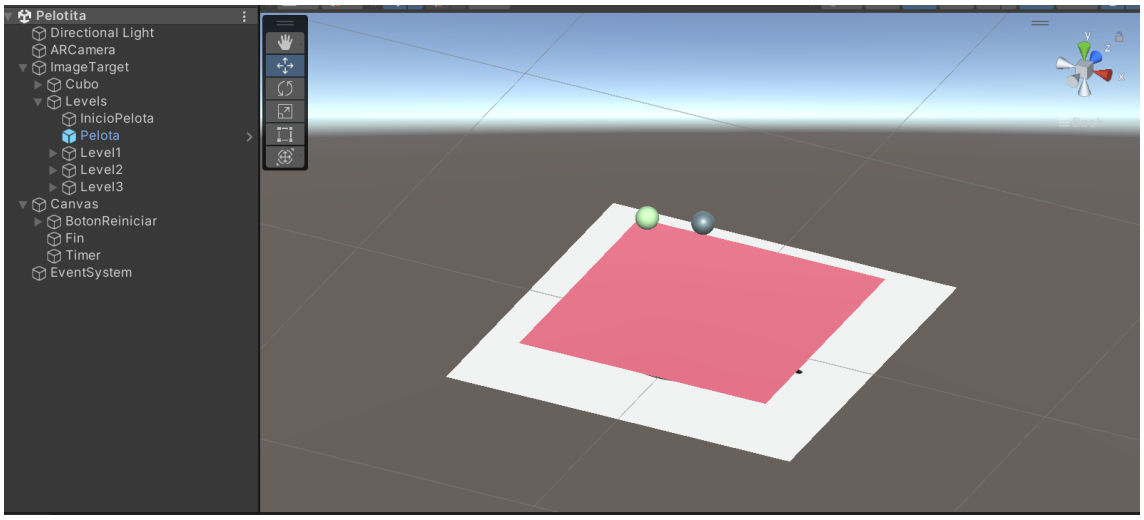


Nivel 2



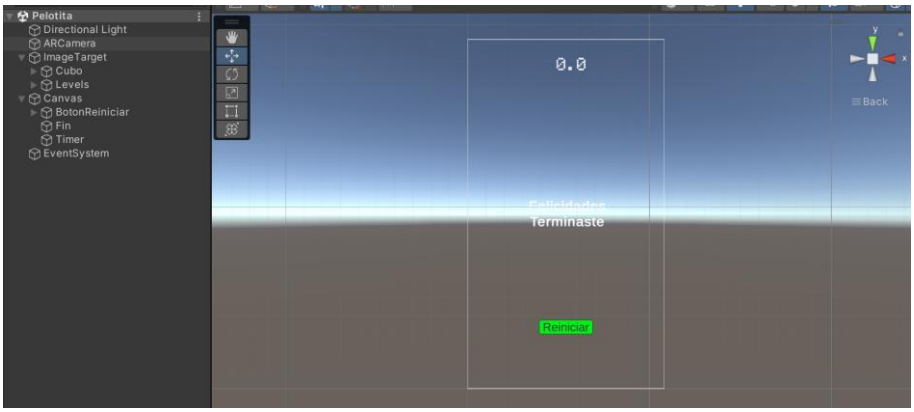
Mario Alberto Vásquez Cancino
315021963

Nivel 3

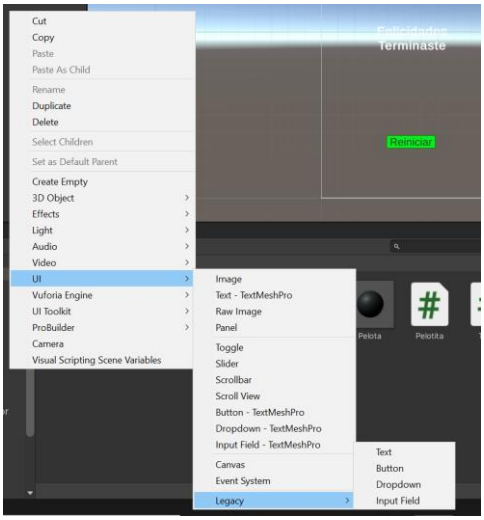


Para facilitar el trabajo se ocupo la misma base para todos los niveles, modificando los colores y elementos, también se agregaron otras pelotas verdes que indican la meta del nivel.

Antes de ir al código modificamos el Canvas agregando un mensaje de concluido que aparecerá cuando acabe el juego y el texto donde estará el contador de tiempo.

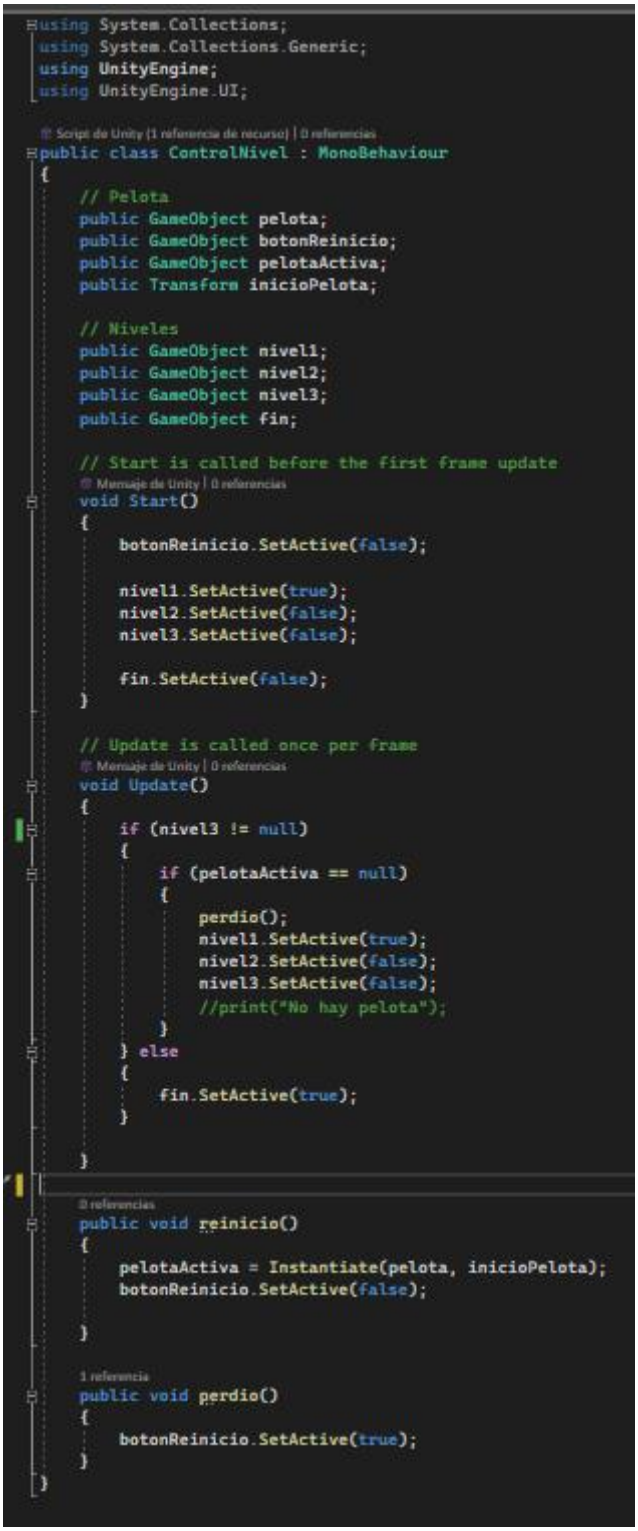


Hay que tener en cuenta que para el contador es necesario ocupar un objeto Text Legacy, no del TextMeshPro.



Para dar introducción a los códigos se explicará el flujo del juego, donde se iniciará con el contador de tiempo activo, la pelota tendrá que ir a la meta que es otra pelota verde estática que al chocar con ella cambiara el escenario que representa el siguiente nivel. En total son tres niveles con tres metas por lo que al cruzar la ultima meta sin caerse de los escenarios se mostrará un mensaje de concluido y se detendrá el tiempo. Cabe destacar que si la pelota se sale del escenario se reiniciaría desde el nivel 1.

Código final de ControlNivel.cs



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

Script de Unity (1 referencia de recurso) | 0 referencias
public class ControlNivel : MonoBehaviour
{
    // Pelota
    public GameObject pelota;
    public GameObject botonReinicio;
    public GameObject pelotaActiva;
    public Transform inicioPelota;

    // Niveles
    public GameObject nivel1;
    public GameObject nivel2;
    public GameObject nivel3;
    public GameObject fin;

    // Start is called before the first frame update
    Mensaje de Unity | 0 referencias
    void Start()
    {
        botonReinicio.SetActive(false);

        nivel1.SetActive(true);
        nivel2.SetActive(false);
        nivel3.SetActive(false);

        fin.SetActive(false);
    }

    // Update is called once per frame
    Mensaje de Unity | 0 referencias
    void Update()
    {
        if (nivel3 != null)
        {
            if (pelotaActiva == null)
            {
                perdio();
                nivel1.SetActive(true);
                nivel2.SetActive(false);
                nivel3.SetActive(false);
                //print("No hay pelota");
            }
        } else
        {
            fin.SetActive(true);
        }

    }

    0 referencias
    public void reinicio()
    {
        pelotaActiva = Instantiate(pelota, inicioPelota);
        botonReinicio.SetActive(false);

    }

    1 referencia
    public void perdio()
    {
        botonReinicio.SetActive(true);
    }
}
```

Lo que se agrego fue de que ahora el código maneja los niveles y mensaje final, done al iniciar solo muestra el nivel 1 y oculta el mensaje final. Ya durante la ejecución va verificando si el nivel 3 no esta destruido (ya que si lo esta es señal que el jugador paso los tres niveles), si no esta destruido verifica que la pelota no se haya caído del escenario, y en caso de que, si reinicia los niveles, pone el botón de reiniciar para volver a generar la pelota. En caso de que el nivel 3 este destruido el mensaje final se desplegara.

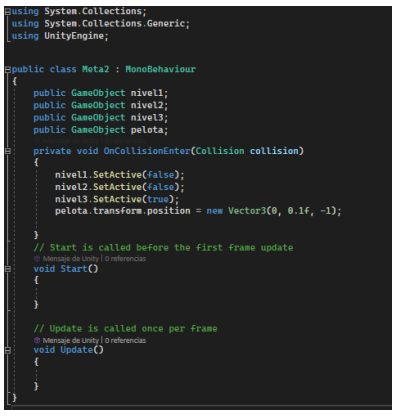
El script de Pelotita.cs no se modificó.

Se crearon tres archivos para cada una de las metas de los niveles.

Meta.cs

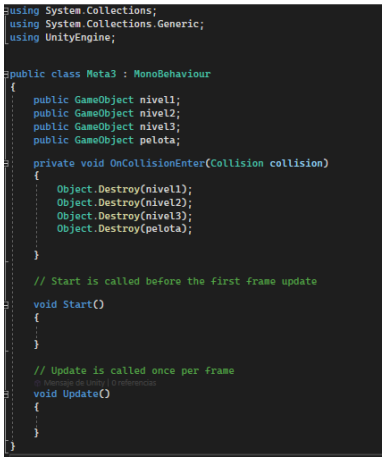
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Meta : MonoBehaviour
{
    public GameObject nivel1;
    public GameObject nivel2;
    public GameObject nivel3;
    public GameObject pelota;

    private void OnCollisionEnter(Collision collision)
    {
        nivel1.SetActive(false);
        nivel2.SetActive(true);
        nivel3.SetActive(false);
        pelota.transform.position = new Vector3(0, 0.1f, -1);

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

    }

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

    }
}
```

Meta2.cs



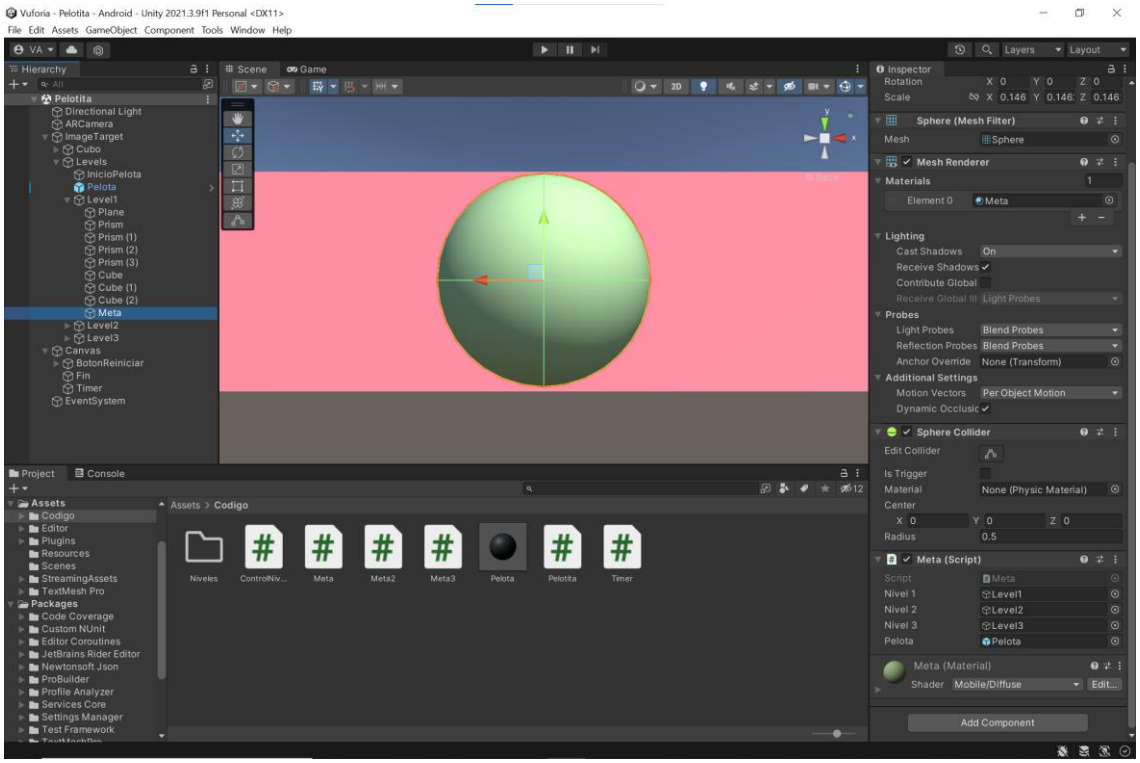
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Meta2 : MonoBehaviour
{
    public GameObject nivel1;
    public GameObject nivel2;
    public GameObject nivel3;
    public GameObject pelota;

    private void OnCollisionEnter(Collision collision)
    {
        nivel1.SetActive(false);
        nivel2.SetActive(false);
        nivel3.SetActive(true);
        pelota.transform.position = new Vector3(0, 0.1f, -1);

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

    }

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

    }
}
```

Meta3.cs



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

public class Meta3 : MonoBehaviour
{
    public GameObject nivel1;
    public GameObject nivel2;
    public GameObject nivel3;
    public GameObject pelota;

    private void OnCollisionEnter(Collision collision)
    {
        Object.Destroy(nivel1);
        Object.Destroy(nivel2);
        Object.Destroy(nivel3);
        Object.Destroy(pelota);

    }

    // Start is called before the first frame update

    void Start()
    {

    }

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

    }
}
```

Los tres son similares lo que diferencia que cuando la pelota de meta detecte que la pelota del jugador colisione con ella se active el siguiente nivel (también se intentó reacomodar la pelota pero no hubo efecto alguno), la única que cambia es la meta del último nivel, ya que al llegar a ella elimina todos los niveles y la pelota. Cabe destacar que cada código se relaciono con su meta correspondiente.



Por ultimo se hizo otro código para el contador del tiempo llamado Timer.c.

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

Script de Unity (1 referencia de recurso) | 0 referencias
public class Timer : MonoBehaviour
{
    public float timeStart;
    public Text textBox;
    public GameObject nivel3;

    // Start is called before the first frame update
    Mensaje de Unity | 0 referencias
    void Start()
    {
        textBox.text = timeStart.ToString("F2");
    }

    // Update is called once per frame
    Mensaje de Unity | 0 referencias
    void Update()
    {
        if(nivel3 != null)
        {
            timeStart += Time.deltaTime;
            textBox.text = timeStart.ToString("F2");
        }

    }
}
```

Este último ocupa las funciones por defecto de Unity para mostrar el tiempo transcurrido desde que se inició la aplicación hasta que detecte que el jugador acabo los niveles (El nivel 3 este destruido).

Con esto se tiene concluida el juego, aun se puede agregar un botón de inicio del juego y un botón al final para volver a jugar.