Universidad Nacional Autónoma de México

Facultad de Ingeniería

Temas Selectos III: Realidad Virtual y Realidad Aumentada

Nombre del Profesor: Ing. Arturo Pérez de la Cruz

Semestre 2024-2

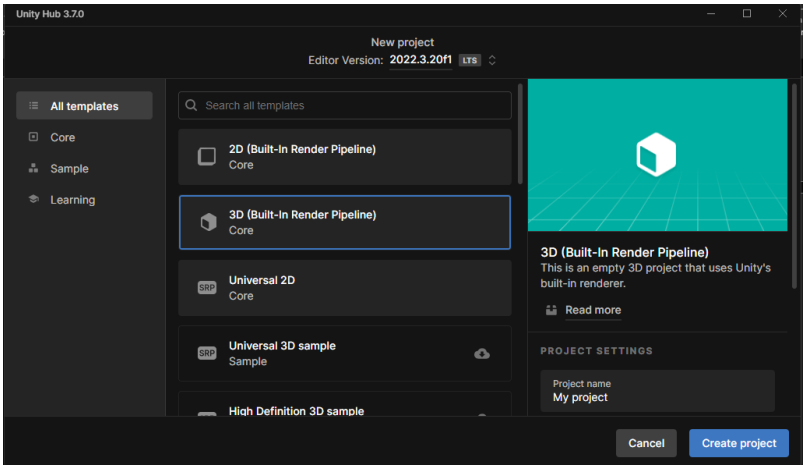
Grupo 01

Proyecto Final: Tiro con Arco.

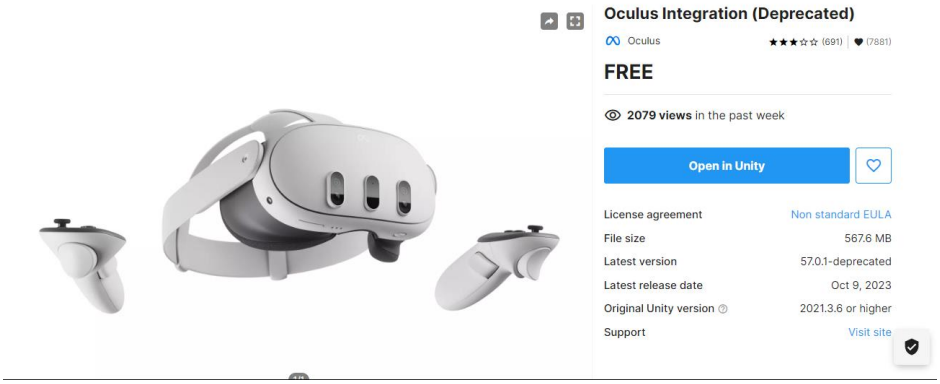
Fecha de entrega: 18/05/2024

Pérez Duarte Brenda Elizabeth 114004268

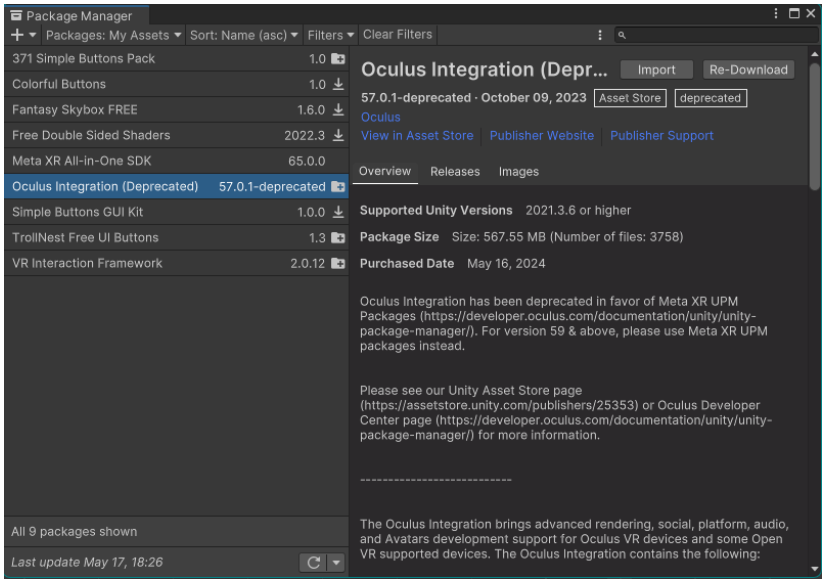
Para la configuración del proyecto primero tenemos que crear un proyecto nuevo 3D:



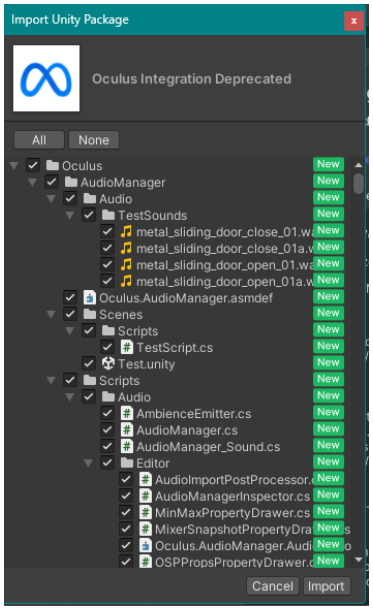
Descargamos el paquete de oculus en la Asset Store llamado Ouculus Integration (deprecated)



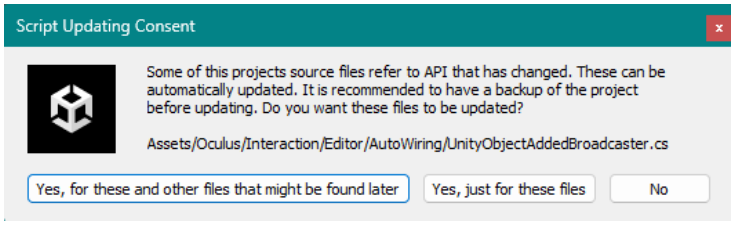
Y lo instalamos en nuestro proyecto:



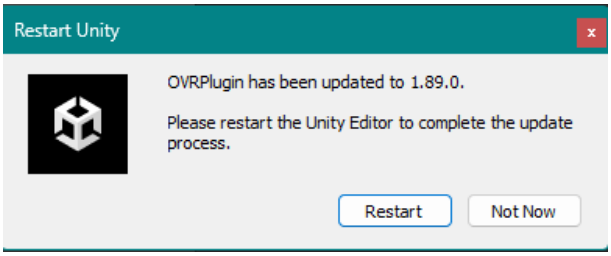
Importamos el paquete



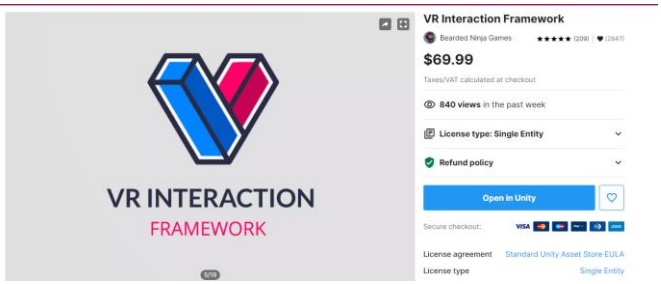
Damos en si a todos los archivos existentes y por existir:



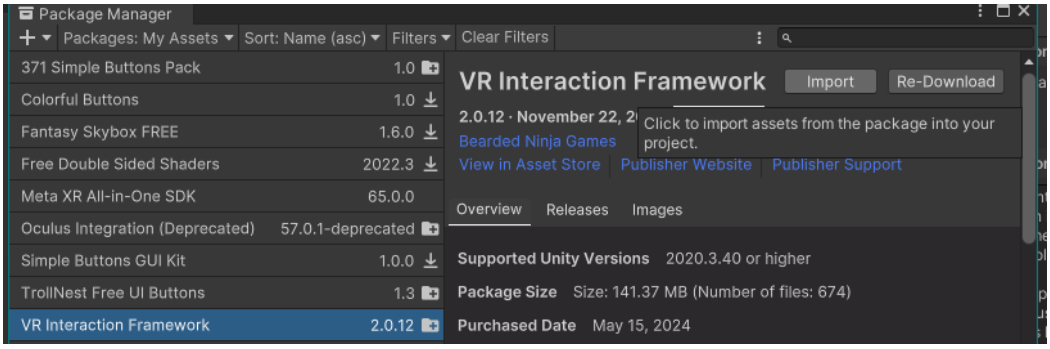
Damos a que reinicie



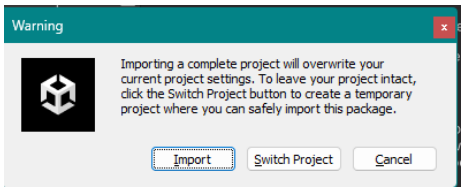
Ya que reinicio el proyecto, descargamos de igual forma el asset VR interaction framework de la tienda de assets de Unity.



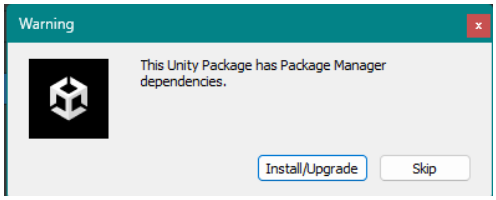
Importamos el Asset en el proyecto



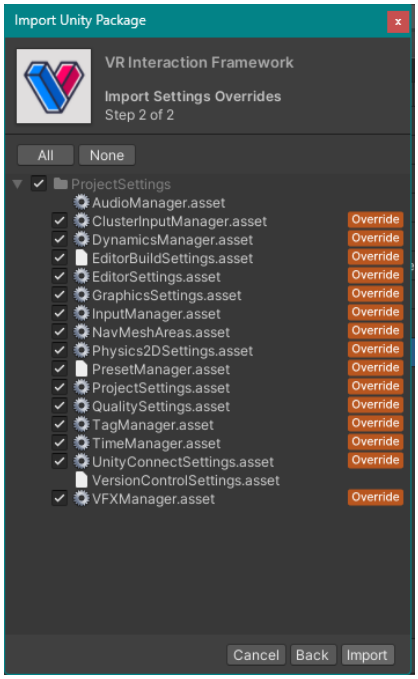
Y le damos import



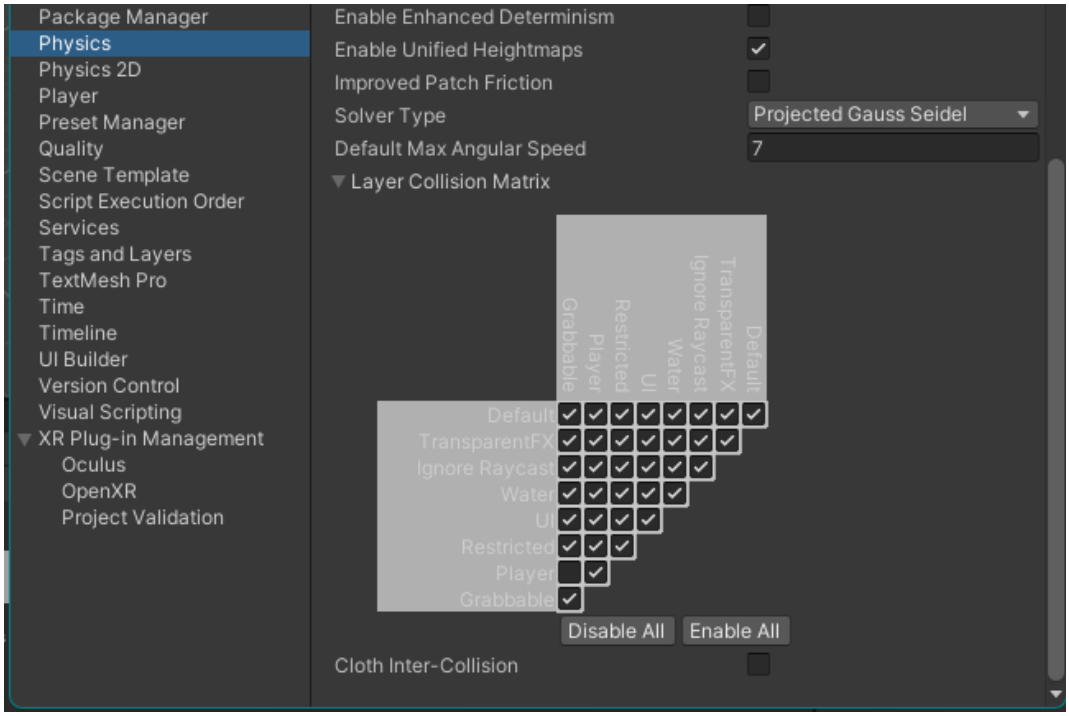
Damos instalar



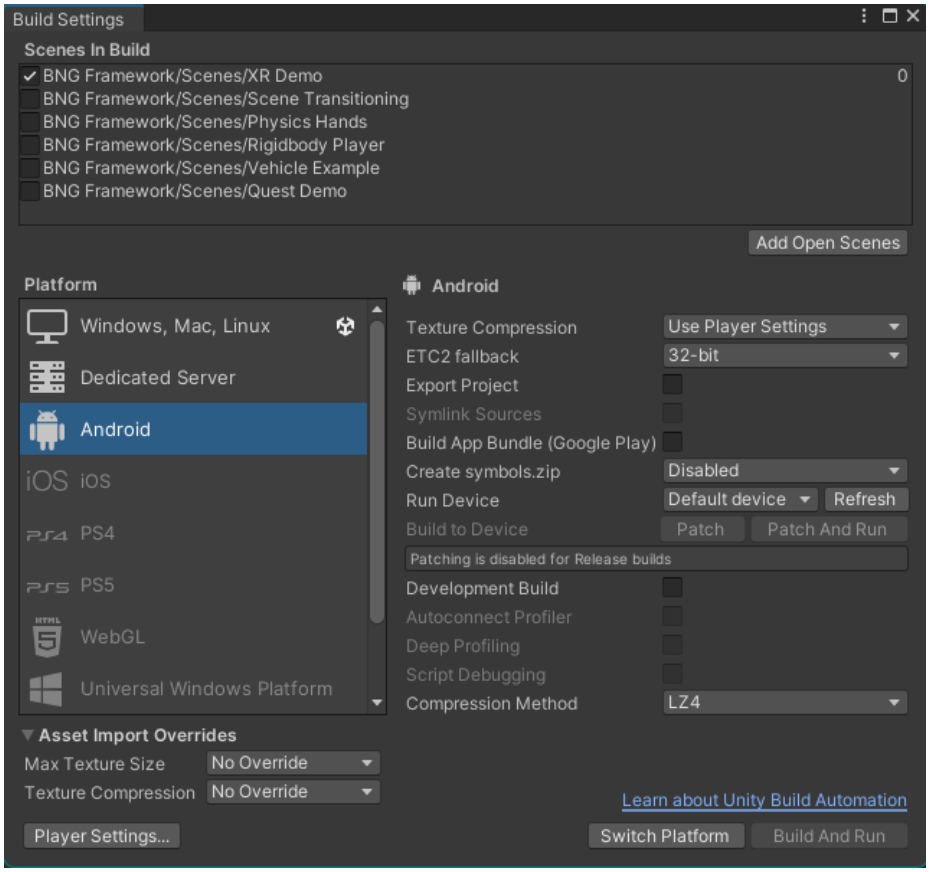
E importamos el paquete



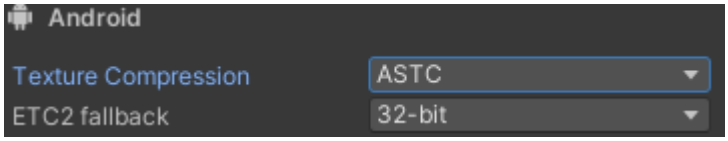
Checar que la parte de físicas se encuentre así:



Cambiamos la plataforma



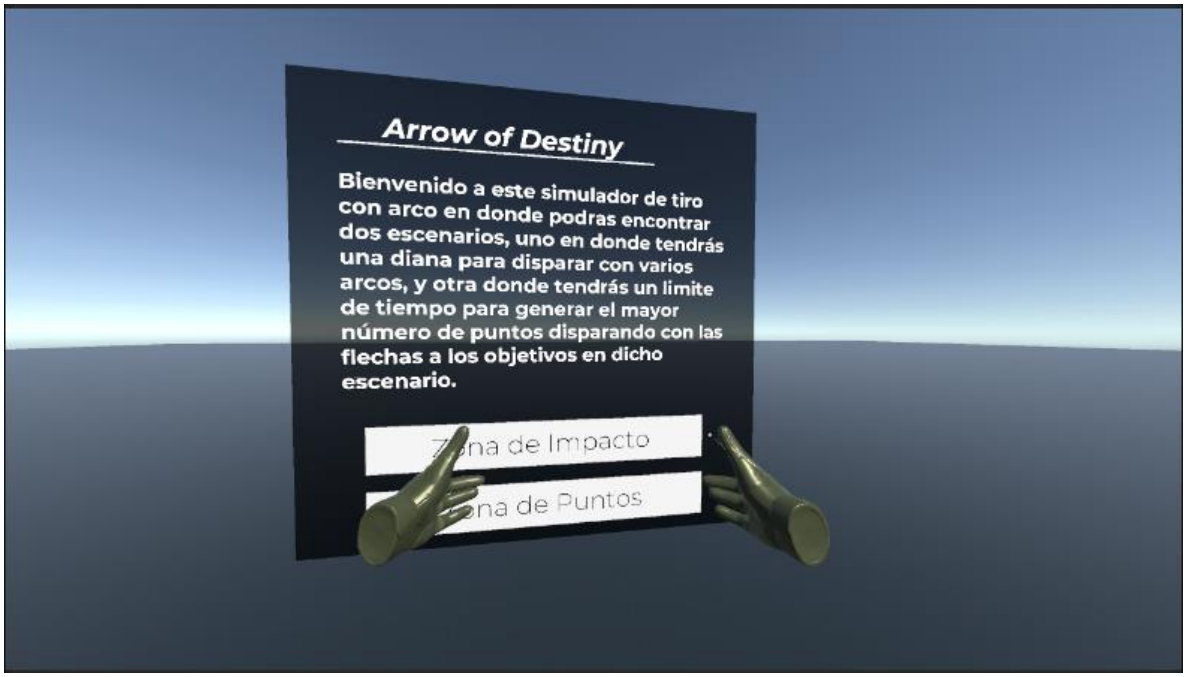
Cambiamos la compresión de texturas:



Y eso seria todo, estamos listo para comenzar a desarrollar.

Gracias a este asset, vienen varias cosas útiles para el proyecto, la parte que vamos a usar son lo del arco, entonces, haremos uso de los arcos que ya vienen, la diana y algunos objetivos que encontramos ahí.

La primera escena, que es la de inicio, será una escena que ya trae el asset para seleccionar el escenario que queramos probar.

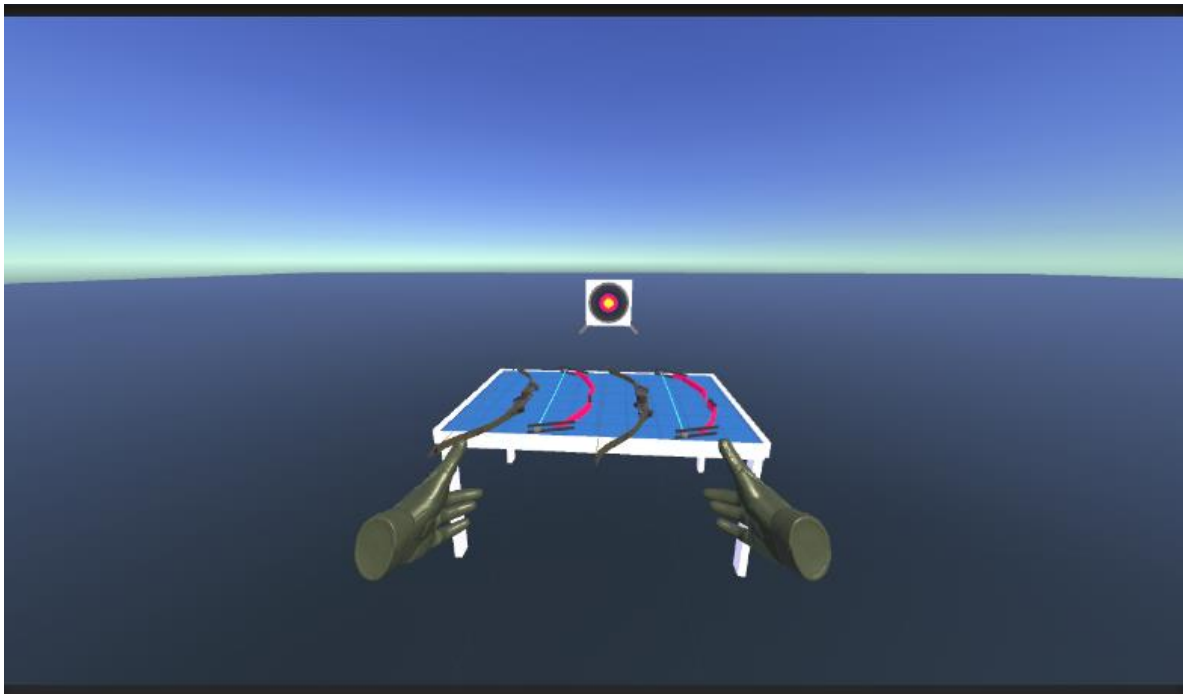


Podemos observar como tiene un gaze pointer para seleccionar la escena a la que queremos teletransportarnos.

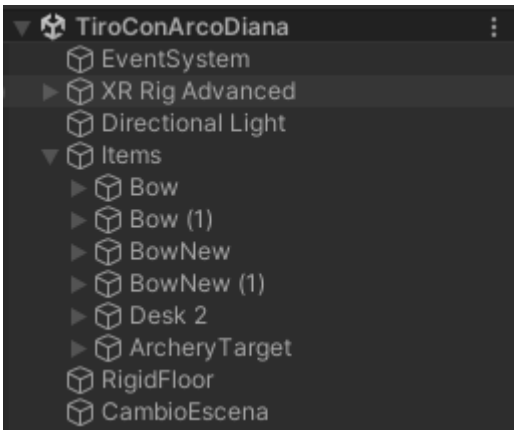
Esta escena esta configurada con los siguientes elementos

Tenemos el piso, una interfaz de usuario donde contienen cuadros de texto y botones para seleccionar la siguiente escena a mostrar y la cámara.

La siguiente escena es el tiro con arco a una diana, en esta escena tiene varios elementos



Tenemos como podemos ver, una mesa para poder detener los arcos, varios arcos que podamos usar que se encuentran sobre la mesa, y una diana, que se encuentra a una distancia de 15 metros.

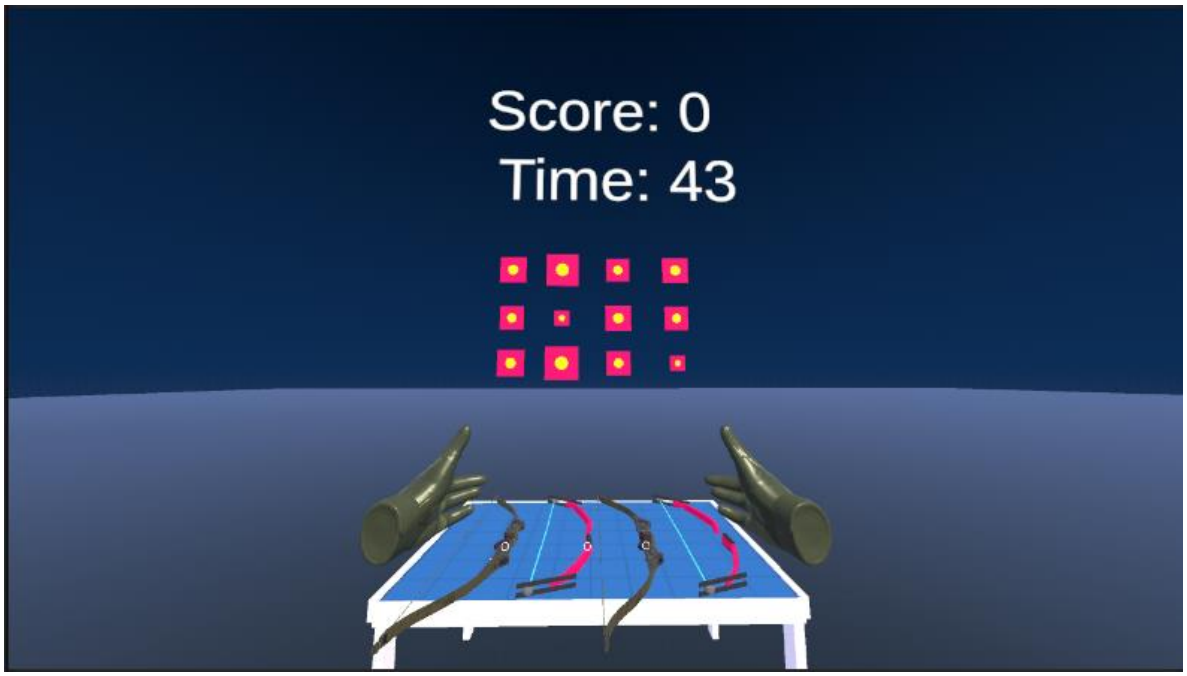


Estos, son los elementos, tenemos de igual forma la cámara, una luz, un objeto de ítems donde se encuentran todos los elementos de la escena, el piso, y un cambio de escena, el código de este cambio de escena lo mostrare a continuación.

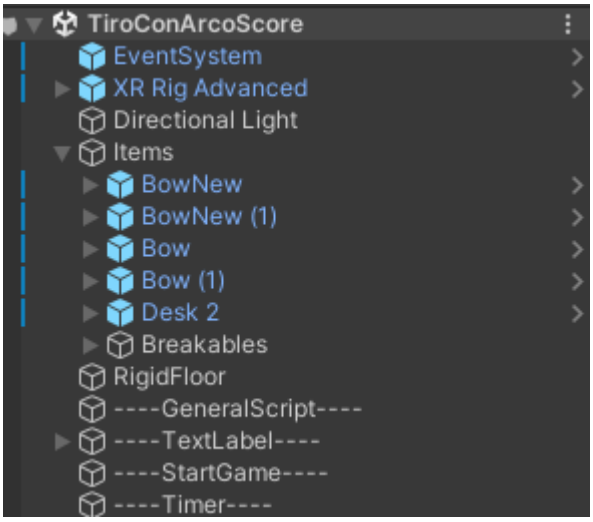
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class CambiarEscena : MonoBehaviour
{


    // Update is called once per frame

    void Update()
    {
        if (Input.GetButtonDown("Fire1") || OVRInput.GetDown(OVRInput.Button.One))
        {
            SceneManager.LoadScene("Inicio");
        }
    }
}

```

Como podemos ver, es un código bastante sencillo, simplemente si detecta que en el caso del simulador el click izquierdo del mouse, o en el caso del oculus, un click en el botón A, hará el cambio de escena a la denominada inicio, que es la interfaz para seleccionar el escenario.

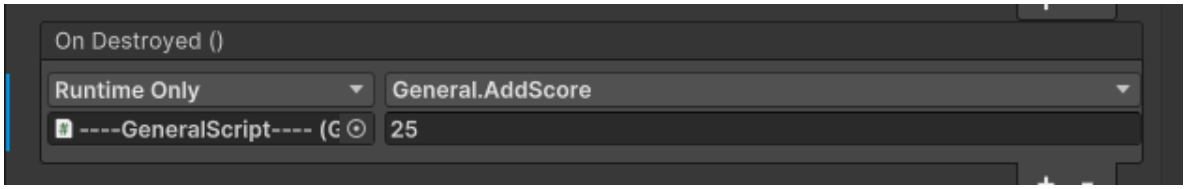
Y por último la escena de zona de puntos, en esta el objetivo es que el jugador impacte con los objetos que tenemos ahí para poder sumando puntos, cada uno de los objetivos tiene un valor diferente dependiendo de su tamaño y volverán a surgir después de 2 segundos de haber sido destruidos



Los elementos de esta escena son:



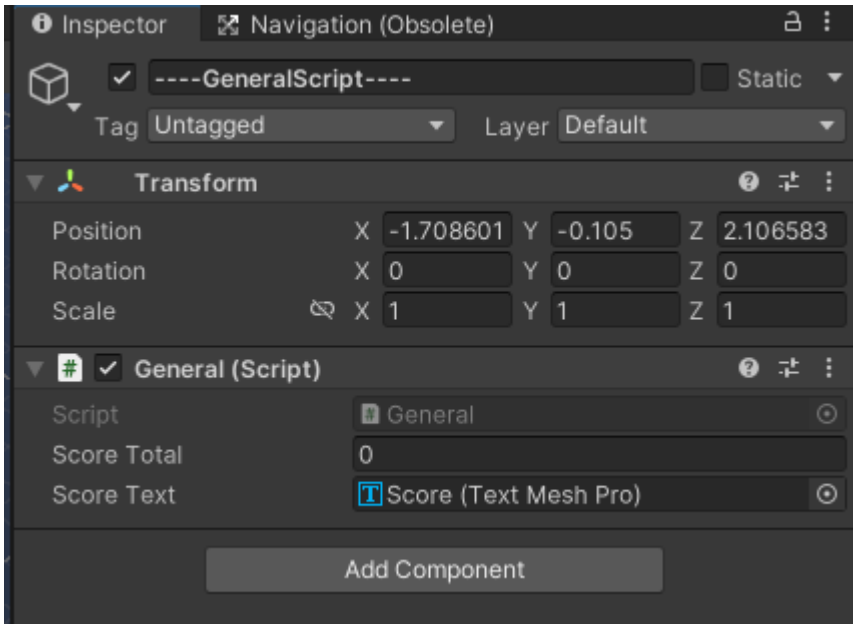
Como podemos ver contamos con la cámara, una luz, el conjunto de ítems el piso, y algunos objetos que contienen los scripts de la dinámica, de igual forma los breakables tienen un script para poder hacer uso de ellos. Procederemos a explicar cada uno de los scripts que contiene esta escena.



El primer elemento a explicar será el tarjet, este tiene una función en on Destroyed que es que cuando se destruya sume en un contador de puntos el score correspondiente a ese objetivo. El código es el siguiente:

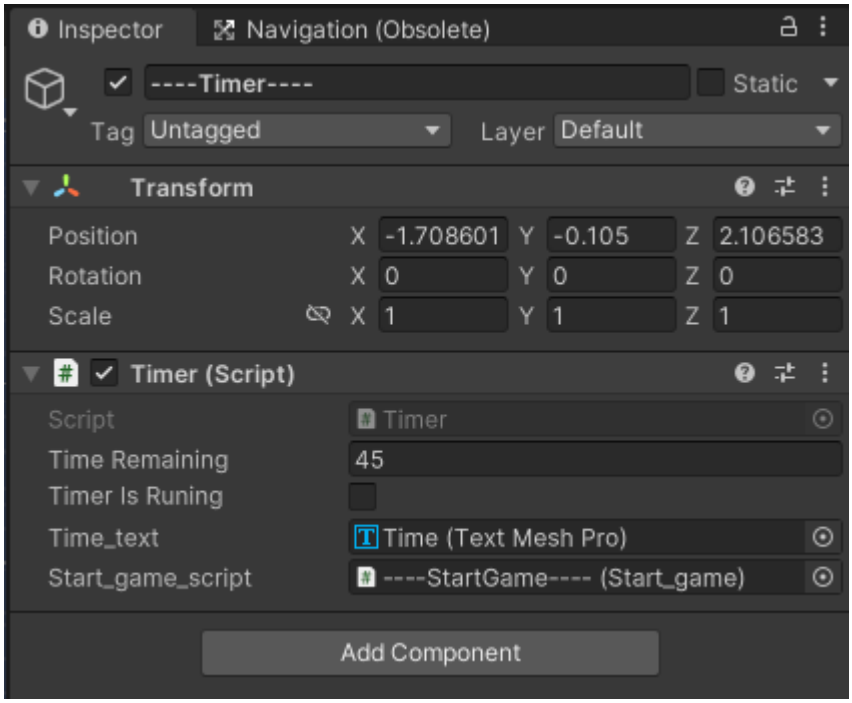
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;
using UnityEngine.SceneManagement;

Script de Unity (1 referencia de recurso) | 0 referencias
public class General : MonoBehaviour
{
    public int scoreTotal = 0;
    public TextMeshPro scoreText;

    Mensaje de Unity | 0 referencias
    void Update()
    {
        if (Input.GetButtonDown("Fire1") || OVRInput.GetDown(OVRInput.Button.One))
        {
            SceneManager.LoadScene("Inicio");
        }
    }
    0 referencias
    public void AddScore(int score)
    {
        scoreTotal += score;
        scoreText.text = "Score: " + scoreTotal;
    }
}
```

Declaramos un entero para guardar los puntos acumulados, y un texto para poder imprimirlo durante la ejecución.En la función update simplemente preguntamos si hay algún click en el botón A del Ouculus o click izquierdo en el mouse del simuladorY la función AddScore que recibe como parámetro un entero y este lo guarda en una variable y lo imprime en el texto que creamos.



Asignamos el script al objeto vacío y asignamos el texto creado para imprimir el score.

Aquí podemos ver estos textos generados para mostrar estos valores.



Tenemos este siguiente objeto que es el contador, como podemos ver es un script que recibe un objeto de tipo texto y objeto del script game_script. El código del timer es:

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

Script de Unity (1 referencia de recurso) | 1 referencia
public class Timer : MonoBehaviour
{
    public float timeRemaining;
    public bool timerIsRuning = false;
    public TextMeshPro time_text;
    public start_game start_game_script;

    Mensaje de Unity | 0 referencias
    void Start()
    {
        timerIsRuning = false;
    }
```

```
19      void Update()
20      {
21          if (timerIsRuning)
22          {
23              if (timeRemaining > 0)
24              {
25                  timeRemaining -= Time.deltaTime;
26                  DisplayTime(timeRemaining);
27              }
28              else
29              {
30                  timeRemaining = 0;
31                  timerIsRuning = false;
32                  start_game_script.Hide_targets();
33              }
34          }
35      }
36
```

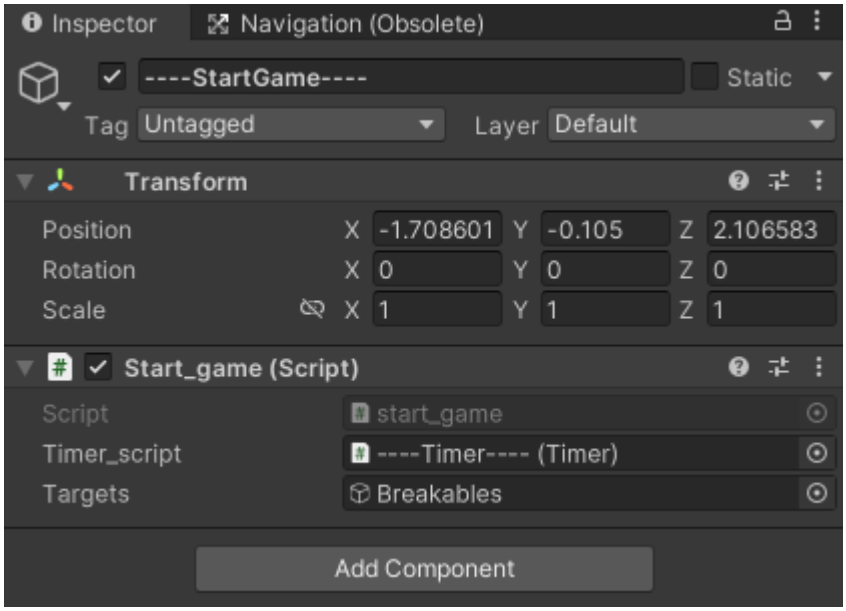
```
37      public void SetRunning()
38      {
39          timerIsRuning = true;
40      }
41
        0 referencias
42      public void StopTimer()
43      {
44          timerIsRuning = false;
45      }
46
        1 referencia
47      void DisplayTime(float timeToDisplay)
48      {
49          timeToDisplay += 1;
50          float minutes = Mathf.FloorToInt(timeToDisplay / 60);
51          float seconds = Mathf.FloorToInt(timeToDisplay % 60);
52          time_text.text = "Time: " + string.Format("{1:00}", minutes, seconds);
53      }
54  }
```

Como primera instancia declaramos un flotante, un booleano, un objeto de texto y un objeto de tipo start_game, que es un script, en la función start, asignamos el booleano como falso, en el update tenemos varios acciones, preguntamos por la variable booleana, después si el tiempo restante es mayor a 0, si lo es, le restamos time.deltatime y mandamos a llamar a la función DisplayTime, en caso de que no sea mayor a 0, el tiempo restante es igual a 0, la variable booleana la ponemos falsa y además escondemos los objetivos llamando a la función de Hide_targets que esta definida en el script de start_game.

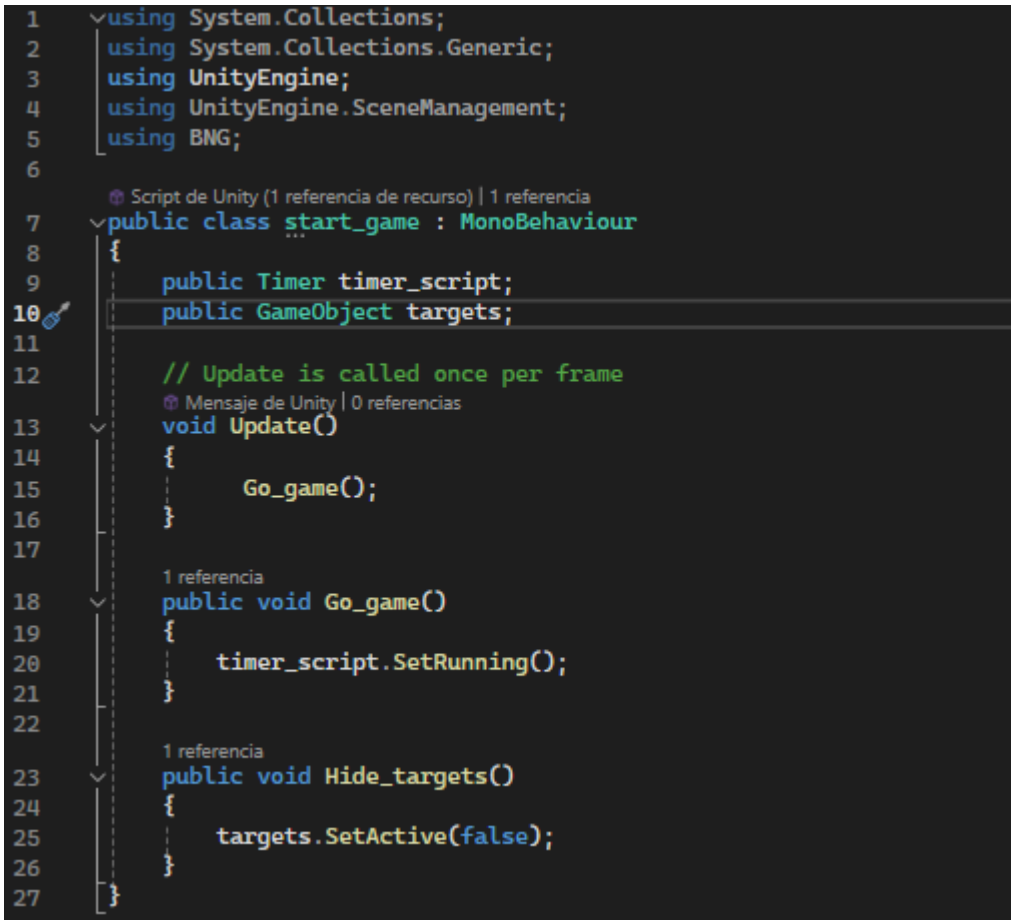
Ahora tenemos una función llamada setRunning en tonde asignamos al booleano falso, al igual que la función stoptime.

La función displaytime, que es la que muestra en ejecución el tiempo restante recibe un flotante, a este flotante le suma 1, después dividimos en 2 esa variable, minutos y segundos, los minutos lo dividimos entre 60 y a los segundos le sacamos el módulo, ahora sí, desplegamos este tiempo en el objeto tipo texto que creamos con un formato especificado.

Y por último el siguiente objeto:



Tenemos asignado en el un script, que recibe un script de tiempo, que es mostrado anteriormente y los objetivos a los que va a afectar, el código es el siguiente:



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

Script de Unity (1 referencia de recurso) | 1 referencia
public class start_game : MonoBehaviour
{
    public Timer timer_script;
    public GameObject targets;

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

    1 referencia
    public void Go_game()
    {
        timer_script.SetRunning();
    }

    1 referencia
    public void Hide_targets()
    {
        targets.SetActive(false);
    }
}
```

De igual forma, es bastante sencillo, generamos un objeto de tipo Timer del script, y un gameObject, targets, en el update mandamos a llamada a la función Go_game(), y la definición de esta es una llamada a SetRunning del script de tiempo y generamos la función Hide_targets() que simplemente desactiva estos gameobjects, que de igual forma la mandamos llamar en el Timer.

Cronograma de actividades realizadas

Actividad	22/03-08/04	09/04–16/04	16/04-27/04	27/04-04/05	04/05-13/05
Modelados					
Escenarios					
Configuración					
Dinámica Tiro					
Interfaz					
Menú					
Pruebas					

Análisis de costo del proyecto

Descripción	Cantidad	Precio	Total
Licencias (Mensual)	1	$2620	$7,860
Equipo Computo (Pago Único)	1	$11,999	$11,999
Servicios de luz e Internet (Mensual)	1	$1,000	$3,000
Pagos de Servicio al desarrollador (Mensual)	1	$2,000	$18,000
Compra de Assets (Pago Único)	1	$1,360	$1,360
Cascos para prueba (Pago Único)	1	$3321.55	$3321.55
Repuestos (Pago Único)	1	$5978.27	$5978.27
	Base imponible		$51,518.82
	21% IVA		$10,818.95
	Total		$62,337.77

Se cobraría por este proyecto un total de $62, 337.77

Ahora, para poder recuperar la inversión de esta aplicación a través de la venta de copias de este mismo, considerando que el producto lo vendemos a un precio de $250^{00}, tendríamos que vender un total de 249 copias para recuperar la inversión, la venta de la copia 250 ya empezaría a generar ganancias por la creación del proyecto.