Javier Cayetano Gualda Núñez

VIDEOGAMES AND VIRTUAL REALITY

Throwable Component Tutorial

Minigame: Bowling Texas

25/04/2019

INDEX

1 STEAMVR THROWABLE COMPONENT

In this tutorial we will learn how to create an object and be able to pick it up and throw it around using the Vive's controllers.

Set Up your Development Enviroment

You'll need a HTC Vive VR headset with the motion controllers. You may also need to install and run **Steam** and **SteamVR**.

Launch Unity and Create a New 3D Project

1. Find the **Unity** editor in the **Windows start** menu, click on the icon to launch it.
2. Create a new **3D** Project named *<your first name>_SteamVRT_Project*, e.g. *Javier_SteamVRT_Project*.

Import the Assests for this Project

First, we import the SteamVR Plugin.

1. Open up the Asset Store inside Unity by accessing the menu **Window > Asset Store**.
2. Look for the **SteamVR Plugin** asset, **Download** and **Import** it.

You'll notice that that plugin includes the Throwable.cs script that we'll be using in this tutorial.

Creating the Scene

First of all, we will create a simple scene so we can test the behaviour of the component we are dealing with. To do so:

1. We have to create the floor for our scene. In the Hierarchy, **Right Click > 3D Object > Plane.** In the Inspector change it's **Position** to **x=0 y=0 z=0**. Name it Floor.
2. Delete the **Main Camera** from the Hierarchy and add the **Player** prefab. You can find it in the folder **SteamVR/InteractionSystem/Core/Prefabs.** Make sure the Player position is within the floor's boundaries. This Player Prefab have all the necessary components to start interacting with the world around you.
3. Now we have to create the object we want to interact with. In the Hierarchy, **Right Click > 3D Object > Cube**. In the Inspector change it's **Position** to **x=1 y=0.1 z=0** and the **Scale** to **0.2** in all axes.

Now hit the **Play** Button on top and see what happens. As you can see, you still can't interact with the cube. Let's fix that.

Add Throwable Component

We have to add all the necessary components to the cube so that we can throw it around. The fastest way to do it is the following:

1. Click on the **Cube** object in the Hierarchy.
2. At the bottom of the Inspector click on **Add Component**.
3. In the Search Bar write Throwable and select the **Throwable Script**.

   As you can see the Throwable Script is not the only component that was added. Now we also have and **Interactable Script**, **RigidBody** and **Velocity Estimator Script**. These were added automatically by Unity because these components are mandatory if we want the Throwable Script to work, as we want the object to also interact with physics and with our player.

   With all these components in mind we can tweak some of the parameters, such as the mass of the object or the velocity assigned to the object when we realease it. For now, let's use the default ones.

Play the Scene

Hit **Play** to test the Scene. As you can see now, you can now pick up the cube and throw it around. For better testing we can also add a teleporting area so we can pick up the object once we have thrown it.

1. In the Project browser find the folder **SteamVR/InteractionSystem/Teleport/Prefabs**.
2. Drag the **Teleporting** prefab into the Scene.
3. In the Hierarchy, **Right Click > 3D Object > Plane.** In the Inspector change it's **Position** to **x=0 y=0 z=0**, click the Add Component button at the bottom and add the **Teleporting Area** script.

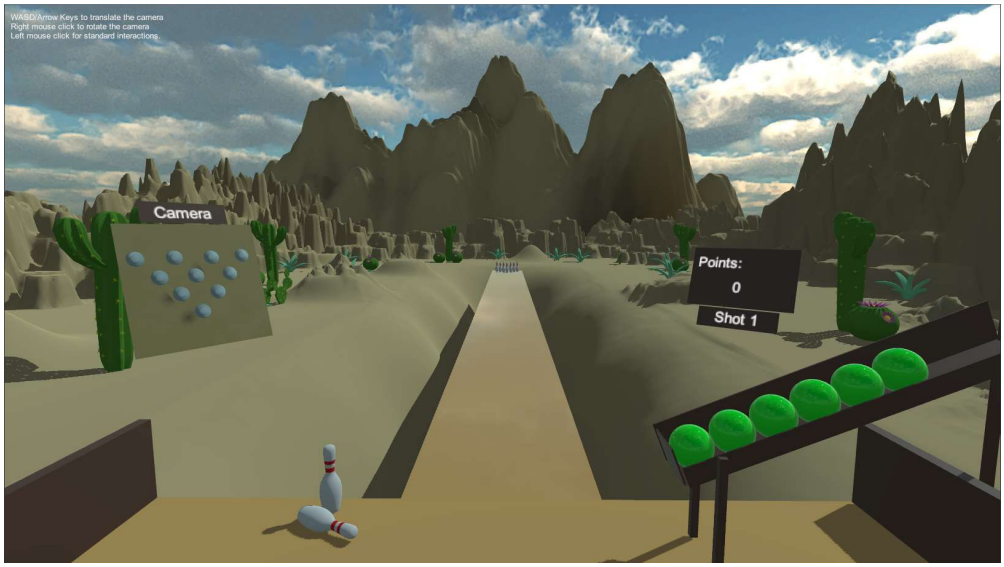
   Save (**File > Save**) and hit **Play** again.

2 MINIGAME: BOWLING TEXAS

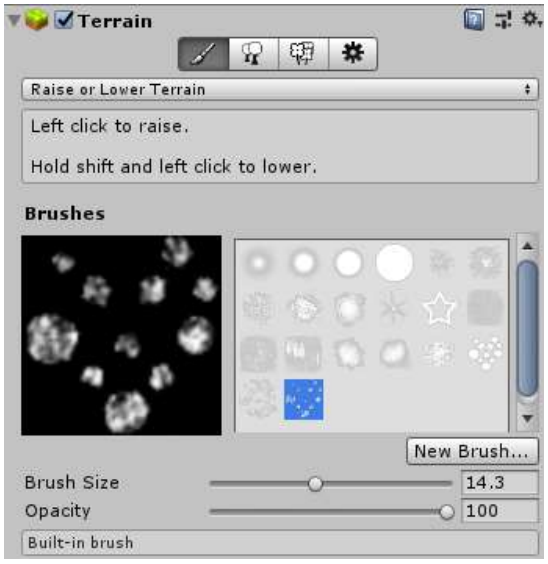
Let's take a look at the minigame that I've created using the Throwable component as the core of it's design. The minigame is called **Bowling Texas** and, as you can guess, it's about bowling!

2.1 VISUALS

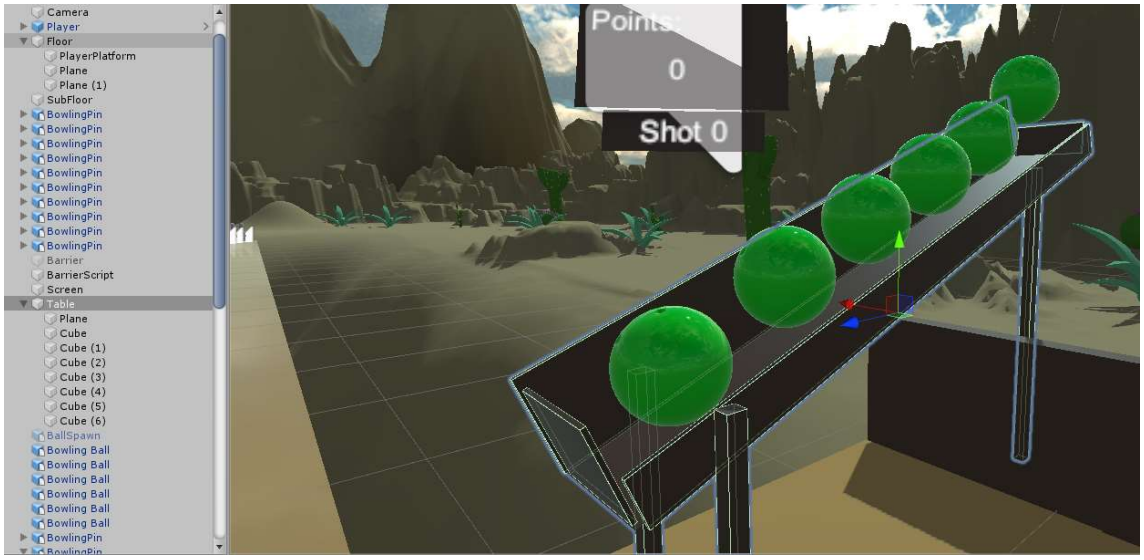
First of all let's talk about the visuals. I decided to create a theme for the scenario instead of going for the typical bowling alley, and that's why I went for a desert look.



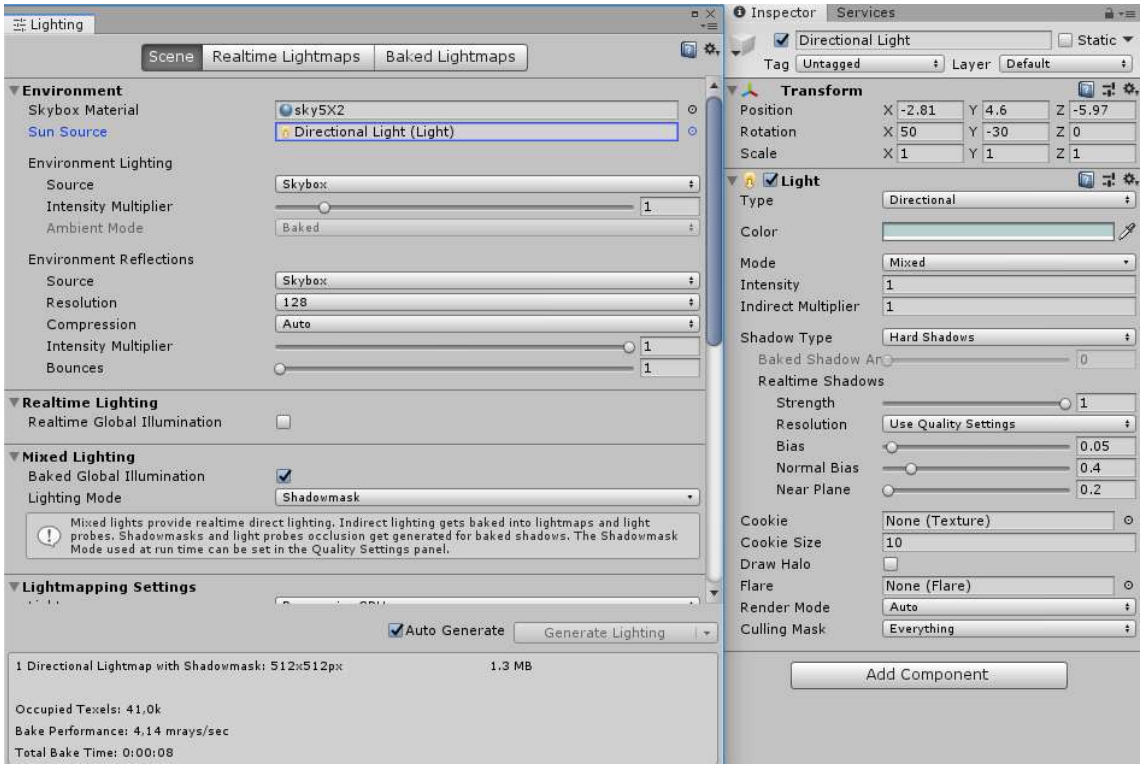
I used the Terrain object to create the ground and the mountains using the editor tool that comes with it and added some cacti from the Asset Store.



I also added the bowling pins and the balls from the Asset Store. The player platform, the ball table and the alley itself are just a bunch of planes and cubes placed in a certain way.



I changed the Skybox to fit the scenario along with the environmental light. As I wanted to go for a **Low Poly style**, I changed the Shadow Type to **Hard Shadows,** giving more of a cartoony look.



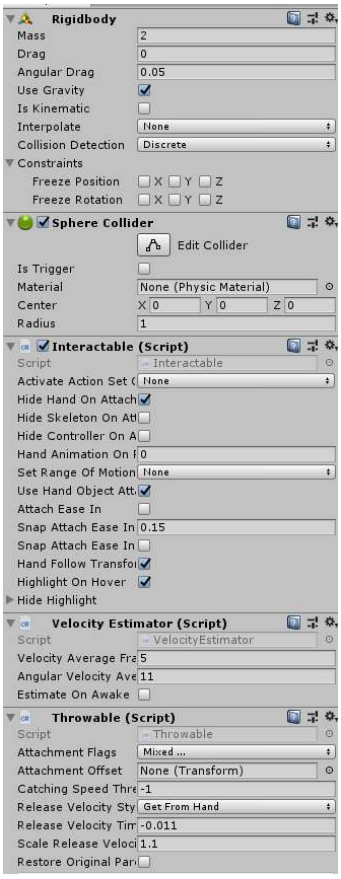
Finally, the textures and materials I used for the environment are just plane colours with nothing more to it. As the alley is supposed to be oily, I added some Smoothness to the material.

2.2 THROWABLE COMPONENT

The game revolves around throwing a bowling ball to take the maximum number of pins out in 2 throws. So we need to add a Throwable component to the bowling balls.

By tweaking some of the properties like the mass or the velocity estimator parameters, I can make the balls feel more realistic.

I also had to change some of the Rigidbody parameters and the colliders on the bowling pins so that the physics resemble those in real life.



2.3 CODE

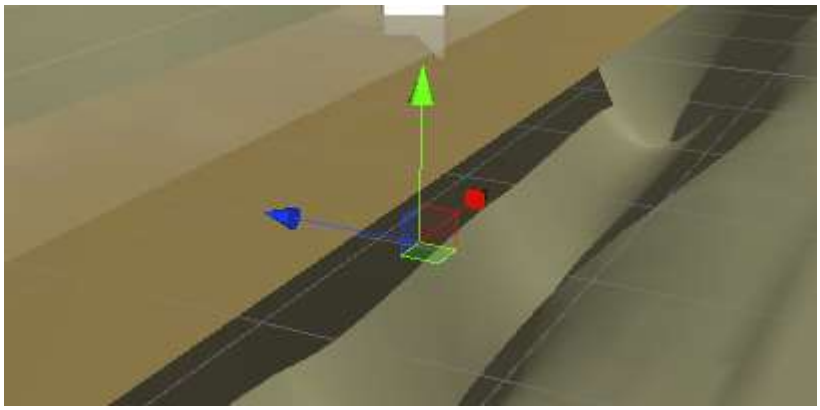
To implement the different features of a bowling game I had to add several scripts to the different objects of the game. Note that the code is not as efficient as it should be because I had to learn how to do certain things along the way.

2.3.1 OutCollision.cs

Script that spawns a ball whenever a ball falls into the pit.

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

public class OutCollision : MonoBehaviour
{

    public GameObject collision;
    public GameObject ballspawn;
    private Vector3 spawnPosition;

    // Start is called before the first frame update
    void Start()
    {
        spawnPosition = ballspawn.transform.position;
    }

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

    }

    void OnCollisionEnter(Collision col)
    {
        if (col.gameObject.tag == "BowlingBall")
        {
            Debug.Log("Colisión detectada");
            Instantiate(col.gameObject, spawnPosition, transform.rotation);
            Destroy(col.gameObject);

        }
    }
}
```

The script is attached to a collider right below the alley. When a ball touches this collider, the ball will be destroyed and another ball will spawn on a fixed spot on the ball table.



2.3.2 WhileBallOnTrack.cs

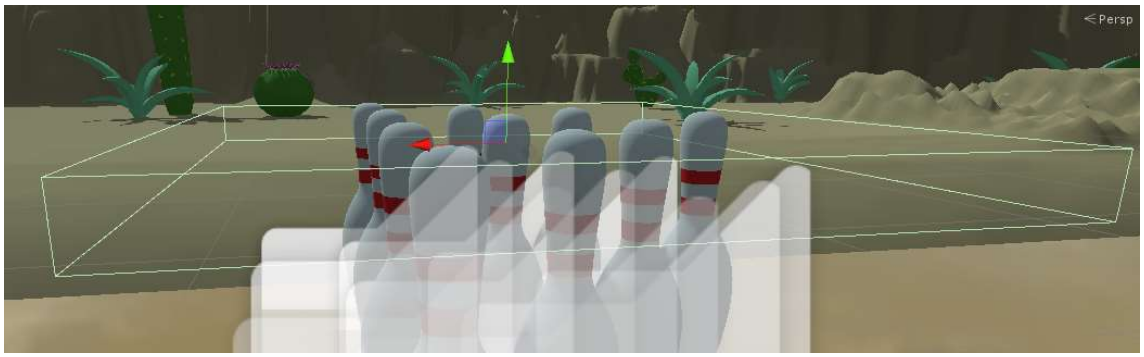
Script that spawns a collider while there's a ball on the track. This collider prevents the player from throwing a ball while there's already one on the alley. I also added a sound when the ball touches the track.

I had to add a couple of if statements on the OnCollisionStay to prevent the ball from getting stuck on the track. On those conditions I just added a force to get the ball out of the track.

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

public class WhileBallonTrack : MonoBehaviour
{

    public GameObject colliderThrow;
    public bool collision;
    public GameObject score;
    private AudioSource source;
    private float timer;

    // Start is called before the first frame update
    void Start()
    {
        colliderThrow.SetActive(false);
        collision = false;
        source = GetComponent<AudioSource>();
        timer = 2.00f;
    }

    // Update is called once per frame
    void Update()
    {
        if(collision == false)
            colliderThrow.SetActive(false);
    }

    void OnCollisionStay(Collision col)
    {
        if (col.gameObject.tag == "BowlingBall")
        {
            //Debug.Log("Colisión bola track");
            collision = true;
            colliderThrow.SetActive(true);
            if (score.GetComponent<Score>().score > 0)
                timer -= Time.deltaTime;
            if(timer < 0)
            {
                col.rigidbody.AddForce(100, 0, 0);
                timer = 2.00f;
            }
        }
    }
    private void OnCollisionExit(Collision col)
    {
        if (col.gameObject.tag == "BowlingBall")
        {
            //Debug.Log("Colisión detectada");
            collision = false;
        }
    }

    private void OnCollisionEnter(Collision collision)
    {
        if (collision.gameObject.tag == "BowlingBall")
        {
            source.time = 0.15f;
            source.PlayDelayed(0);
        }

    }
}
```

2.3.3 PinDespawn.cs

Script that removes those pins that have been taken out. To do this, I added a collider that is always touching the pins.



Whenever a pin is taken out, it will stop touching that collider and the OnTriggerExit function will be executed. I added a timer of 2 seconds for the pins to be removed. I also added some sounds. The score will be updated.

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

public class PinDespawn : MonoBehaviour
{
    public GameObject spawn;
    public GameObject score;

    private float timer;
    private AudioSource source;
    public AudioClip clip;
    public bool collision;

    // Start is called before the first frame update
    void Start()
    {
        collision = false;
        source = GetComponent<AudioSource>();
    }

    // Update is called once per frame
    void Update()
    {
        if (collision)
        {
            timer -= Time.deltaTime;
            if (timer < 0)
            {
                this.transform.position = spawn.transform.position;
                Debug.Log("Despawn pin");
                collision = false;
            }
        }
    }

    private void OnTriggerExit(Collider other)
    {
        if(other.name == "TriggerPin")
        {
            timer = 2.00f;
            collision = true;
            score.GetComponent<Score>().score++;
            source.time = 0.2f;
            source.PlayDelayed(0);
        }
    }

    public bool GetCollision()
    {
        return collision;
    }
}
```

2.3.4 Score.cs

Script that will track the number of pins that have been taken out and shows that in the score panel.

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

public class Score : MonoBehaviour
{
    public int score;
    public GameObject trigger;
    private int shot;
    //private AudioSource source;

    // Start is called before the first frame update
    void Start()
    {
        score = 0;
        gameObject.GetComponent<TextMesh>().text = score.ToString();
        //source = GetComponent<AudioSource>();
    }

    // Update is called once per frame
    void Update()
    {
        shot = trigger.GetComponent<BallCount>().counter;
        gameObject.GetComponent<TextMesh>().text = score.ToString();
        if (score == 10 && (shot == 1 || shot == 0))
        {
            gameObject.GetComponent<TextMesh>().text = "Strike !";
            trigger.GetComponent<BallCount>().strike = true;
            //source.PlayDelayed(0);
        }
        else if (score == 10 && shot == 2 && !gameObject.GetComponent<TextMesh>().text.Equals("Strike !"))
        {
            gameObject.GetComponent<TextMesh>().text = "Spare !";
            //source.PlayDelayed(0);
        }
    }

    public void ResetScore()
    {
        score = 0;
    }
}
```

2.3.5 BarrierSpawn.cs

Script that will spawn a barrier in front of the pins, preventing the player from shooting another ball. This will happen whenever a pin is taken down but still hasn't been removed from the track or after the second shot of the player. This was done to prevent some bugs that I encountered after throwing that second shot.

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

public class BarrierSpawn : MonoBehaviour
{

    public GameObject[] pin;
    public GameObject barrier;
    public GameObject trigger;

    public bool isActive;


    // Start is called before the first frame update
    void Start()
    {
        barrier.SetActive(false);
        isActive = false;
        //Debug.Log("OK");
    }

    // Update is called once per frame
    void Update()
    {
        if ((pin[0].GetComponent<PinDespawn>().GetCollision() || pin[1].GetComponent<PinDespawn>().GetCollision() || pin[2].GetComponent<PinDespawn>().GetCollision() ||
            pin[3].GetComponent<PinDespawn>().GetCollision() || pin[4].GetComponent<PinDespawn>().GetCollision() || pin[5].GetComponent<PinDespawn>().GetCollision() ||
            pin[6].GetComponent<PinDespawn>().GetCollision() || pin[7].GetComponent<PinDespawn>().GetCollision() || pin[8].GetComponent<PinDespawn>().GetCollision() ||
            pin[9].GetComponent<PinDespawn>().GetCollision()) || trigger.GetComponent<BallCount>().barrera)
        {
            //Debug.Log("Barra On");
            barrier.SetActive(true);
            isActive = true;
        }
        else
        {
            //Debug.Log("Barra Off");
            barrier.SetActive(false);
            isActive = false;
        }

    }
}
```

The barrier also places a collider right in front of the player, preventing him from shooting another ball.

2.3.6 ShotTracker.cs

Script that keeps track of shots within a turn and displays it on the text the scripts is attached to.

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

public class ShotTracker : MonoBehaviour
{
    public GameObject trigger;

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

    }

    // Update is called once per frame
    void Update()
    {
        if (trigger.GetComponent<BallCount>().counter == 0)
            gameObject.GetComponent<TextMesh>().text = "Shot 1";
        else if (trigger.GetComponent<BallCount>().counter == 1)
            gameObject.GetComponent<TextMesh>().text = "Shot 2";
    }
}
```

2.3.7 BallCount.cs

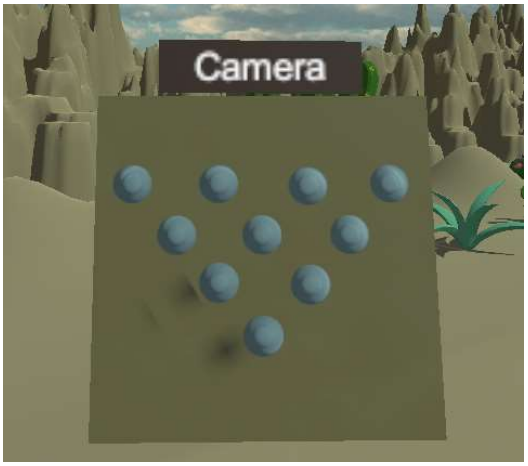
Script that will detect the ball falling out of the track, increasing a counter that will determine the shot the player is within a turn (e.g 1 or 2). When the counter is 2 (after shot 2) it will tell the barrier to spawn to the SpawnBarrier script and will reset the pin's positions by spawning them into a fixed spot after 8 seconds (this was necessary to synchronize it with the pins being removed after the second shot). While the pins are spawning, their rotation and position constraints will be fixed for 1 second to prevent them from falling.

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

public class BallCount : MonoBehaviour
{

    public GameObject[] pin;
    public GameObject[] spawnPin;
    public Rigidbody[] rigidPin;
    public GameObject score;

    public float timer = 600.00f;
    private float timer2 = 8.00f;
    public int counter;
    public bool barrera = false;
    public bool strike = false;

    // Start is called before the first frame update
    void Start()
    {
        counter = 0;
    }

    // Update is called once per frame
    void Update()
    {
        if (counter == 2 || strike)
        {
            barrera = true;
            timer2 -= Time.deltaTime;
            if (timer2 < 0)
            {
                ResetPins();
                counter = 0;
                strike = false;
                score.GetComponent<Score>().ResetScore();
                timer2 = 8.00f;
            }
        }
        timer -= Time.deltaTime;        //Timer
        if (timer < 0)
        {
            for (int i = 0; i < 10; i++)
            {
                rigidPin[i].constraints = RigidbodyConstraints.None;
            }
            barrera = false;
            Debug.Log("Constraints clear");
            timer = 600.00f;
        }
    }
```

```
54      private void OnTriggerExit(Collider other)
55      {
56          if(other.tag == "BowlingBall")
57          {
58              counter++;
59              Debug.Log(counter);
60          }
61
62      }
63
64      public void ResetPins()
65      {
66          for(int i=0; i<10; i++)
67          {
68              rigidPin[i].constraints = RigidbodyConstraints.FreezeRotation;
69              pin[i].transform.rotation = spawnPin[i].transform.rotation;
70              pin[i].transform.position = spawnPin[i].transform.position;
71          }
72          timer = 1.00f;
73      }
74  }
```

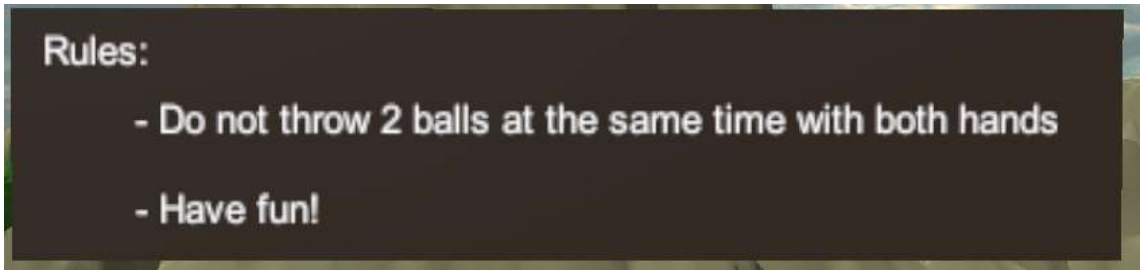
2.4 Misc

The game has a camera that shows the position of the pins so that the player can actually see the pins that are left on track.



I also added background music to fit the game and environment.

If the player turns back, he will be able to see a panel with things to take in mind while playing the game.



I also added colliders so that the player can't throw the ball to places he isn't supposed to.

2.5 How to play the game

I tried to make the game as similar as possible to a real bowling game, so you should be able to play it right away. Pick up a ball and throw it into the track trying to take down as much pins as possible in a single turn (2 shots). As you are not playing against anyone, the scores don't add up: every turn your score will reset.

Notice that whenever you throw a ball, you won't be able to throw another one until the first one is out of the track! Whenever you see the barrier spawning in front of the pins you won't be able to shoot a ball either, so please be patient as the pins respawn.

That's everything. Have fun!