Documentation: Project Bowling

Contents

Gameplay

This game can be played with only one button for starters you can only use space or the left mouse click.

You can see a bowling ball at the start which constantly moves left and right, when you click space or left click the ball will stop and a power indicator will be shown, after pressing space again the bowling ball will launch at your worst enemies the bowling pins.

While the ball is being launched at the pins you can also manipulate the position from left to right using ‘A’ or ‘D’.

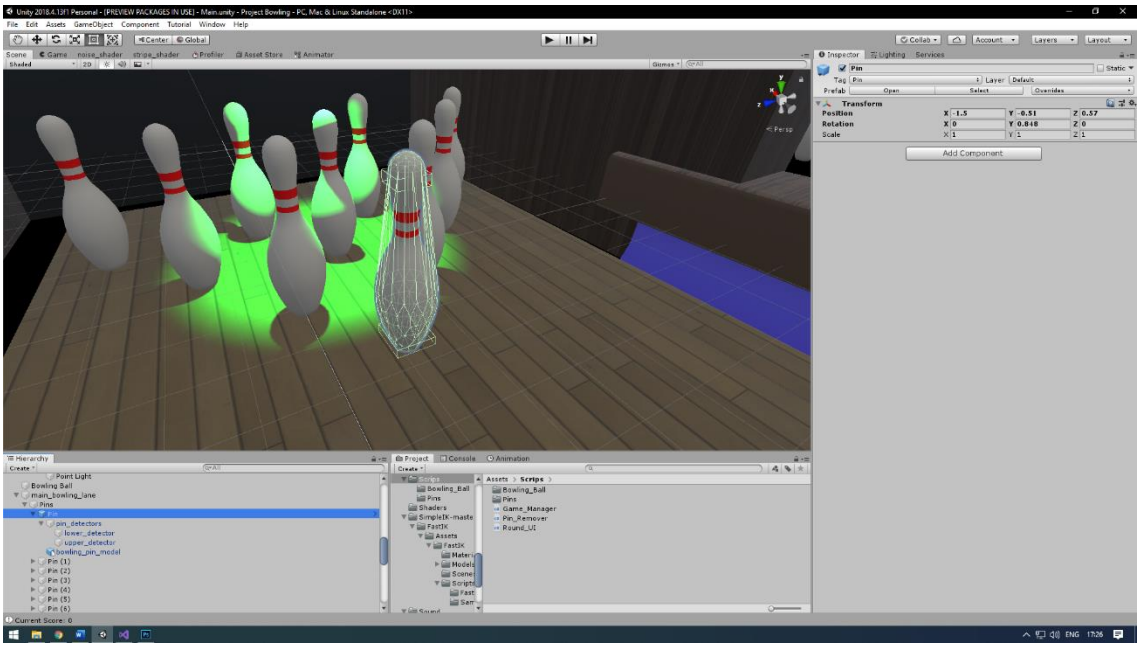
You can see your progress in the scoring table, happy bowling!

Scripting

Setting up the pins

As you can see on the picture bellow I have created a Bowling Pin GameObject that contains:

- Pin detectors
  - Lower detector
  - Upper detector
- Bowling pin model



Pin detectors are basically cubes which are invisible and they also have a collider which is set to be a trigger. To give this detectors some logic I wrote a script called “pin_detector.cs”. This script detects collision with the pin that these detectors are linked to, it compares if they are the same gameobject instance and if they are it detects that the pin is present on this detector(remember there are two detectors upper and lower).

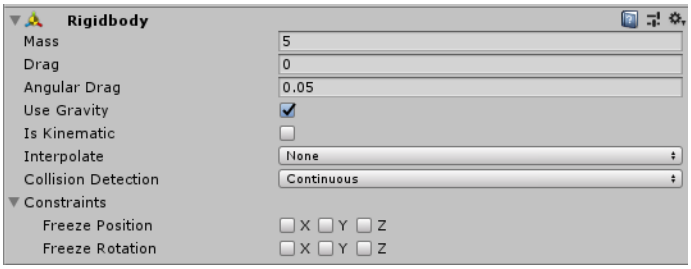
```
private void OnTriggerStay(Collider other)
{
    if (other.gameObject.CompareTag("Pin"))
    {
        if (other.gameObject == targetPin)
        {
            isPinPresent = true;
        }
        else
        {
            isPinPresent = false;
        }
    }
}

private void OnTriggerExit(Collider other)
{
    if (other.gameObject.CompareTag("Pin"))
    {
        if (other.gameObject == targetPin)
        {
            isPinPresent = false;
        }
    }
}

public bool getIsPinPresent()
{
    return isPinPresent;
}
```

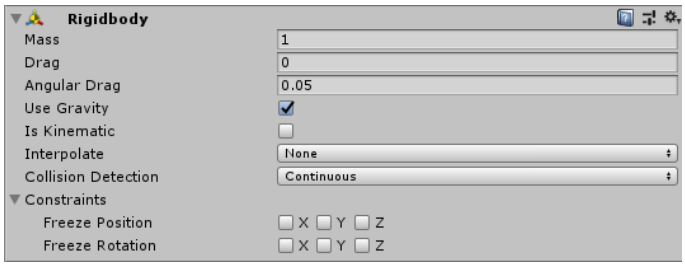
```
public class pin_behavior : MonoBehaviour
{
    [SerializeField]
    private bool isKnocked = false;
    public pin_detector[] detectors;
    public Image pinStatusImage;

    void Update()
    {
        bool isPinPresent = true;
        foreach(pin_detector el in detectors)
        {
            if (el.getIsPinPresent()==false)
            {
                isPinPresent = false;
            }
        }

        isKnocked = isPinPresent ? false : true;

        if (isKnocked)
        {
            pinStatusImage.color = Color.red;
        }
        else
        {
            pinStatusImage.color = Color.green;
        }

    }
}
```

Afterwards we move to the “pin_behavior.cs” script where will look at collecting the previous information.

This script is attached to the bowling pin model, it reads the information given from the detectors and it decides if it is knocked or not. For a pin to be knocked it needs to be absent from one of the two detectors that are following it. If the pin is knocked it changed the given pinStatusImage color to red to indicate that it has been knocked, otherwise if a pin is present the color will be set to green.

Let’s not forget about Physics, in order to use the physics engine in Unity we need to attach rigidbodies to our Pins and Bowling Ball, in this case the only thing that needed setup was that we needed continuous collision detection and the difference in mass. Our bowling ball in this case is 5 units heavy while our pins are 1 unit, there is a x5 difference in the weight which gives close to reality results.





Our player the Bowling Ball

Next up is our main character in this game and that's the bowling ball, except the sphere collider and a rigidbody there is a "Player_Behavior.cs" script which is a mess but I will get over some of the most important functions.

Here we are handling our input, I am using a flag to know if an action should be executed.

```
void Update()
{
    if (Input.GetKeyUp(KeyCode.Space) || Input.GetMouseButtonUp(0))
    {
        actionPressed = true;
    }
}
```

The bowling ball before we start bowling has 3 states as we mentioned above first is choosing our path, second is choosing our power and third is launching the ball at the pins.

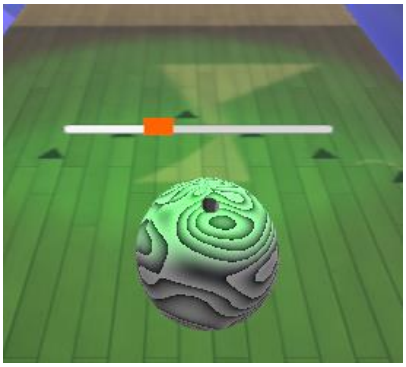
```
if (!selectPosition && !selectForce)
{
    if (transform.localPosition.z < -5.5f)
    {
        side = (1);
    }else if(transform.localPosition.z > 3.1f)
    {
        side = (-1);
    }

    transform.localPosition += new Vector3(0, 0, 1f * side * Time.deltaTime*speed);

    if (actionPressed)
    {
        selectPosition = true;
        actionPressed = false;
    }
}
```

Therefore we have 2 flags selectForce and selectPosition.

When selectPosition is false and selectForce is false we start with selecting our position. The ball moves left and right and we are waiting for actionPressed to become true so we can finish with choosing our position.

```
if (!selectForce && selectPosition)
{
    powerUI.SetActive(true);

    if (interpolationFlag)
        powerValue += 0.05f;
    else
        powerValue -= 0.05f;

    if (powerValue <= 0.1f)
        interpolationFlag = true;
    else if (powerValue >= 1.4f)
        interpolationFlag = false;

    powerSlider.value = powerValue;

    if (actionPressed)
    {
        powerUI.SetActive(false);
        selectForce = true;
        Invoke("LaunchTheBall", 1f);
        actionPressed = false;
    }
}
```

Now we have selected our position and we can move to choosing our power, the powerUI pops up and we can get ready to hit space and get our desired force.

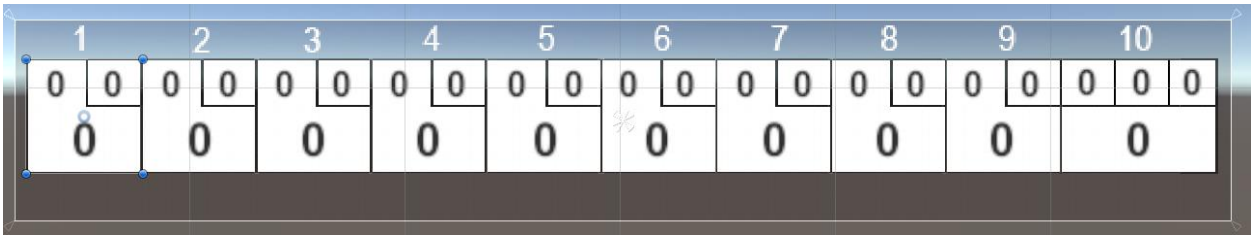


Okay now we everything is set and we can launch the bowling ball at the pins using a function from the rigidbody called addForce(), it takes a Vector3 parameter as force.

```
private void LaunchTheBall()
{
    rb.AddForce(new Vector3(1, 0, 0) * force * powerValue);
}
```

Scoring Table

The scoring table is made out of 10 rounds, you can strike twice per round(I decided to call them steps in my script). If you manage to get a strike(knocking out ten bowling pins) you will be moving to the next round. The 10th round is specific and is considered a bonus round in this game which means you'll get to 3 by 10 fresh sets of pins in order to increase your score.

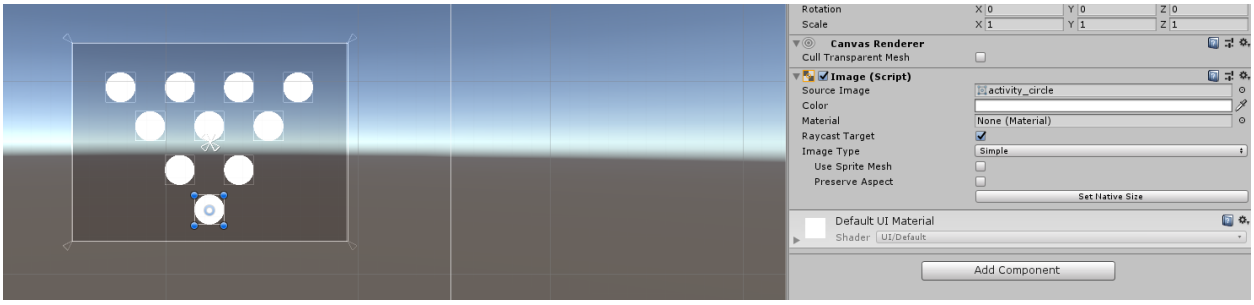


Every round GameObject is consisted of the text fields you can see in the picture, one text field for the first throw, another for the second throw and below you can see a bigger number. That bigger number is the accumulated score of the current and previous rounds.

There is a "Round_UI.cs" script attached to the Round type GameObjects. I won't get in to too much details but it helps with managing all of those text fields you see on the screen.

Pins Activity

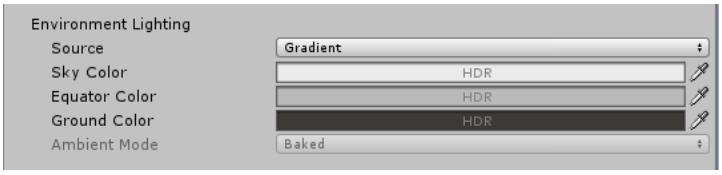
There is also an UI element on the left side of the screen that shows you your progress at the moment. Those 10 white circles are just basic circle sprites which change their color based on the status received in the "pin_behavior.cs". Red color means that the pin is knocked and Green means that it is in play.



Lighting

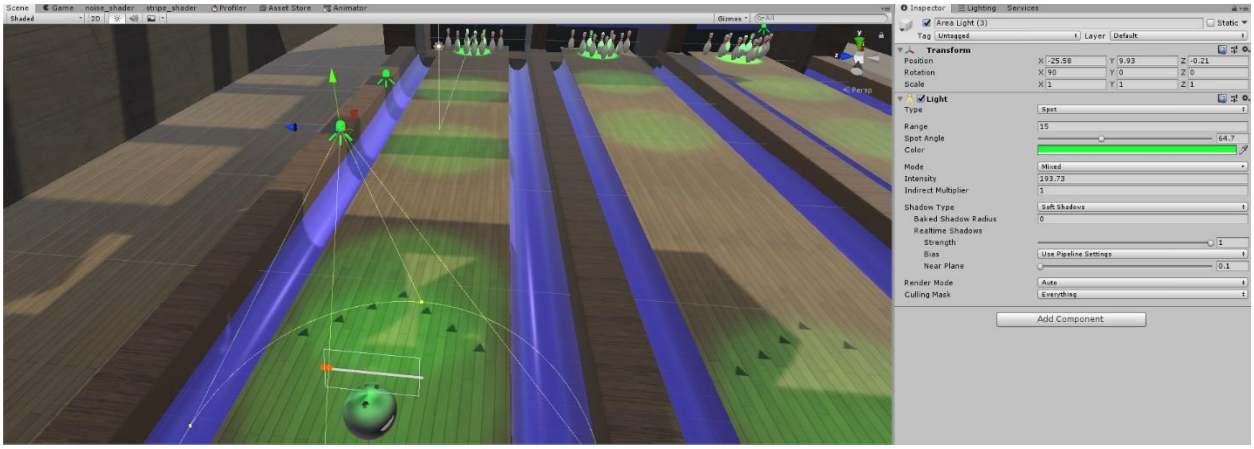
For lighting in these scenes I used Spot lights, Directional light, Point light and Environment Lighting.

I changed the Environment Lighting from the bluish skybox to gradient shades of gray.



Point light just above the bowling lanes.

Most of the atmosphere in the game is made by the green spot lights that are set along the bowling lane. Every lane consists of 4 spot green spot lights.

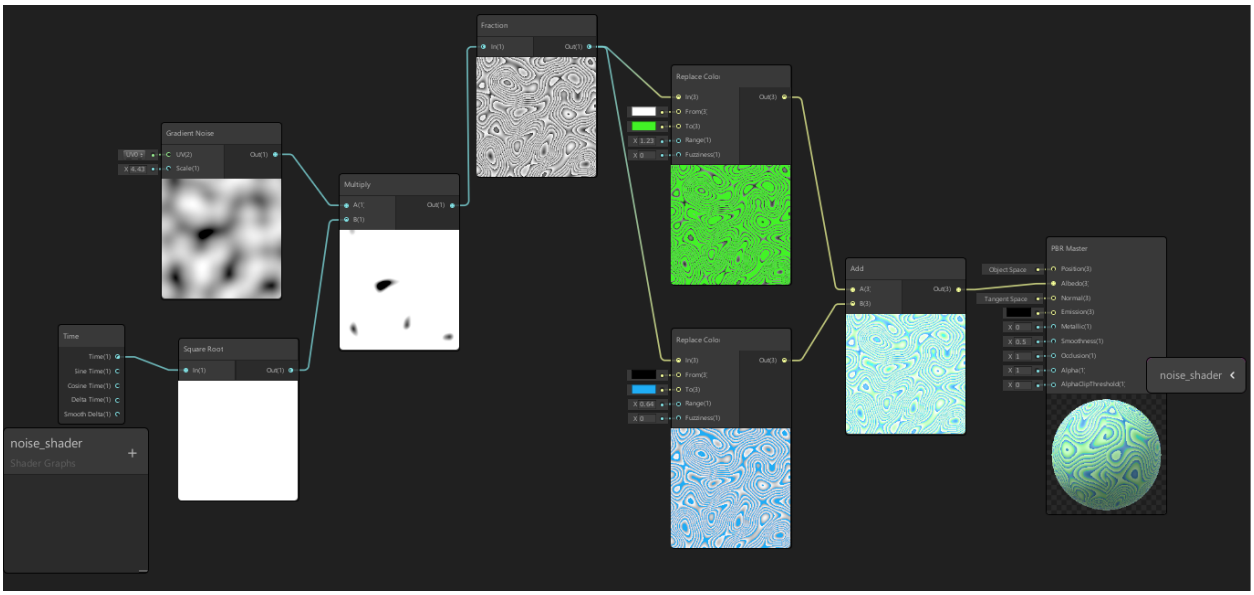


Shaders using the Unity Shader Graph

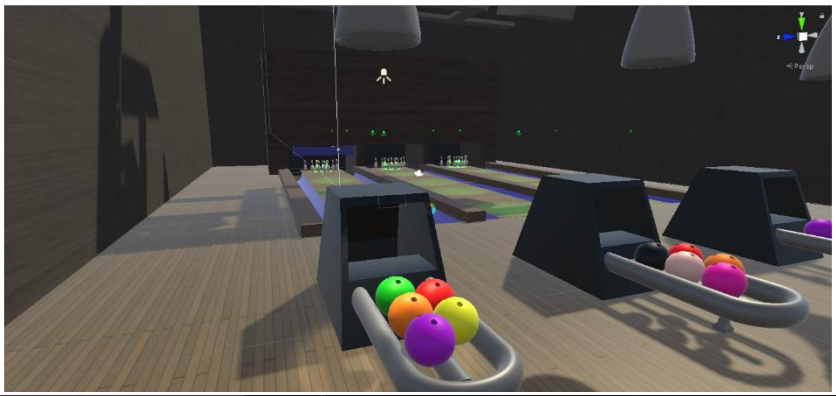
I am pretty new to shaders, so I wanted to tryout the new Shader Graph editor that has been added to Unity since version 2018.1.

I wasn't getting around pretty well but I managed to make a simple noise texture on the bowling ball that changes as time passes.

Link to a tutorial that explains all the components in the Unity Shader Graph editor.



Previews of the scene



References to Models and Textures

References to the models and textures that were used in the project

- Old Wood Siding
- Rough Concrete
- Rough Old Wood
- Bunker Concrete
- Bowling Pin
- Bowling Alley (only textures were used)