a. Graphical Assets

Designing the player

I first started with designing the player character before anything. It was easiest for me to build a world around a simple character at the time of making these sketches. I envisioned something zooming through the land and I went from there. I knew I wanted something that is alien to the player, something quick but not dangerous or threatening. I worked by drawing up silhouettes of figures with defining shapes that can communicate to the viewer efficiently. I explored many possibilities and concepts but I liked the second to last one best. They looked alien but friendly. They have ears that resemble that of a rabbit and big eyes. They have no defining limbs as limbs can cause drag and I wanted this spirit to care about one thing and one thing alone- going really fast.

Before finalizing, I wanted to explore this design further. What's the most interesting aspects of this design? How would they function? How would they emote? I drew up concepts of splash art, game over and a concept for a possible "bump" animation. At this point I was confident and pleased with this design so I moved onto what was the most difficult part I faced in designing: Colouring.

Colouring the player character was a challenge. At this point I didn't have a world to take the colours out of as I was working in reverse. I knew I wanted the setting to be a forest of sorts but I didn't know which kind of forest. As a result I took pallets from all of the spectrum, making sure that I could make a good decision moving forward. I made the conscious decision of making keeping the design very simple. I limited myself to only using 3 colours at most onto this character for maximum readability. I remember I had a very vague idea if what I wanted out of the player character but was unhappy that I wasn't getting close to what I was envisioning (The first three designs are me trying to pinpoint what I was picturing actually!). Out of the six concepts art- I chose the second to last colour palette. I felt like it works the best and compliments the fast nature spirit well.

Designing the Enemy

My teammate approached me and asked me to design a character for a moving obstacle. This threw me for a loop. I had no clue what to do as a moving obstacle. Woodland animals? A Bird? Fairies? A rolling log? After attempting to ponder on it more, I booted up my program and went in on sketching with no plan whatsoever. Perhaps I could come up with something as I went. And I did.

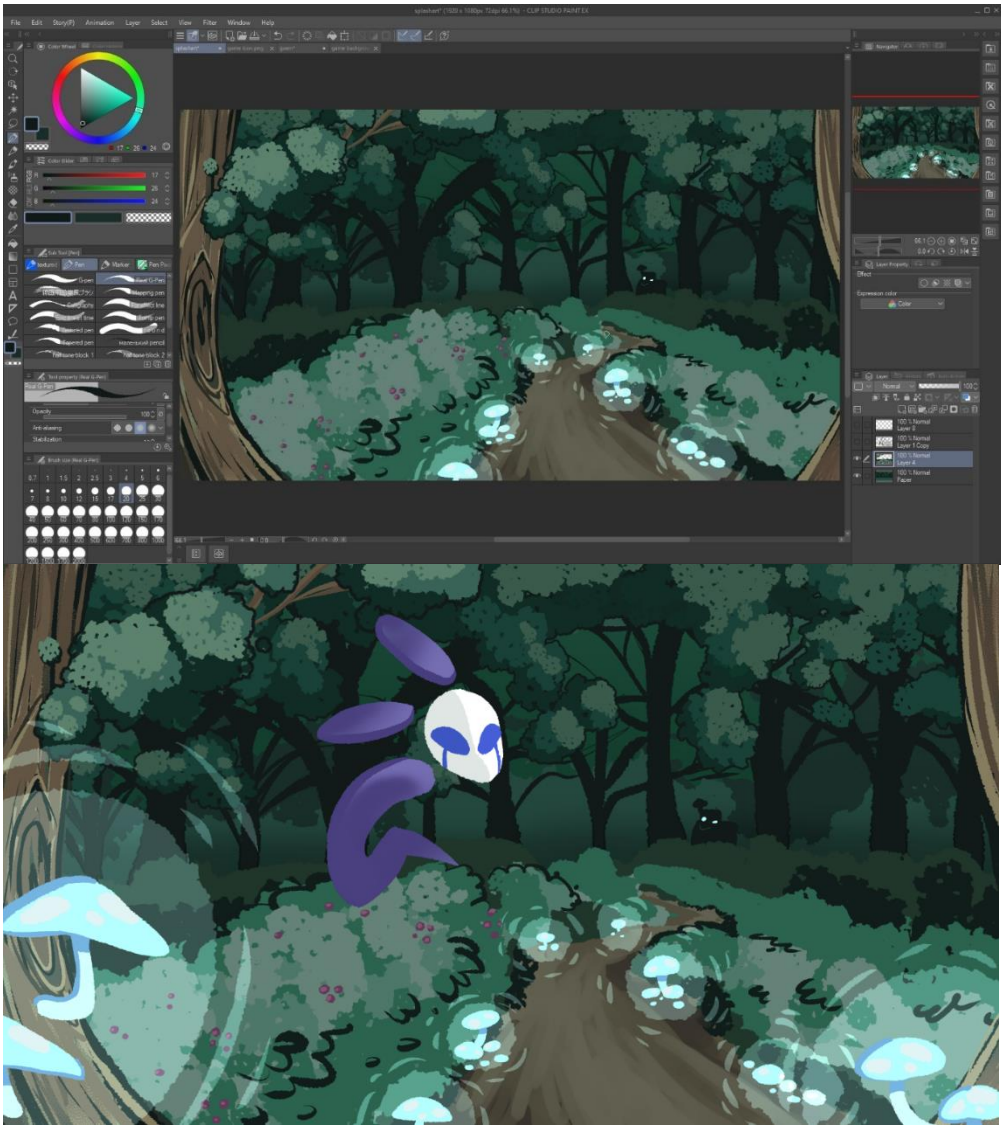
Using the same technique that I used in designing the player character, I drew with silhouettes and clear shapes. I tested out anything that everything I could and even went a little too abstract and creepy with my designs. I kept a close eye on the player character's defining shapes so that this character has traits that opposes them. I went with slower, heavyset frames. I played with the idea that maybe the enemies weren't directly attacking the player but are more in their way, blocking their path. I also kept in mind how wide and squarish shapes can give off a different vibe from the main character.

With this idea in mind, I designed three last characters.

I went with the middle design, it was the polar opposite of what the player character looks like. I also did some experimental sketches about the enemy character before finalizing. And I must admit I fell in love with this design. During this process, I lovingly named them Frog and gave them frog friends.

Splash art and Icon

The icon was very straight forward. Easily the most iconic of the player character is the mask. On a canvas that's roughly 250x250 pixels drew the face and ears symmetrically for a pleasing, minimalistic look. I briefly considered giving it a solid purple background but it looks best as a standalone icon.



Lastly is the splash art. With all of the working files open I was able to compose an illustration of what I envision the game's world of looking like. I used a more painterly look for the splash art as it reminds me of old videogames having higher quality renders in their loading screens. I used a colour blocking technique throughout the whole composition.

C: > Users > gabri > Documents > GitHub > AssignmentProject > Assignment > Assets > Scripts > Wick.cs

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

public class Wick : MonoBehaviour
{
    [SerializeField]
  //creating a container(variable) to contain our player speed
    private float _speed = 3.5f;
    private float _force = 1000f;
    // Start is called before the first frame update

    void /// <summary>
    /// Start is called on the frame when a script is enabled just before
    /// any of the Update methods is called the first time.
    /// </summary>
        Start()
        {
        transform.position = new Vector3(0,0,0);
        }
    void Update()
    // Update is called once per frame
        {
        //if we press the right arrow move the player to the right
        if(Input.GetKeyDown(KeyCode.RightArrow))
        {
            //move the player to the right
            transform.position += Vector3.right;
            //informs software what player is doing
            Debug.Log("Right arrow key was pressed");
        }
        else if(Input.GetKeyDown(KeyCode.LeftArrow))
        {
            //move the player to the left
            transform.position += Vector3.left;
            Debug.Log("Left arrow key was pressed");
        }
        //move the player upwards
        if(Input.GetKeyDown(KeyCode.UpArrow))
        {
            transform.position += Vector3.up;
            Debug.Log("Up arrow key was pressed");
        }
        //move the player downwards
        else if(Input.GetKeyDown(KeyCode.DownArrow))
        {
            transform.position += Vector3.down;
            Debug.Log("Down arrow key was pressed");
```

```
        }
        //if position on the y axis is smaller than -4 reset the scene
        //if position on the y axis is larger than 6 reset the scene
        //if position on the y axis is < than -4f OR > than 1f
        if(transform.position.y < -4f)
        {
            //restarts scene from beginning
            SceneManager.LoadScene(0);
        }
        if(transform.position.y > 6f)
            //restarts scene from beginning
            SceneManager.LoadScene(0);
        //if the position of the player on the y axis is larger than 9 reset the scene
        //if the position of the player  on the y axis is < than -9f OR > than 9f
        if(transform.position.x < -9f || transform.position.x > 9f)
        {
            SceneManager.LoadScene(0);
        }
    }
}
```

b. Scripts Overview

This code is for the main character. It has basic mechanics like moving up, down, left and right. I used “GetKeyDown” so the player does not move continuously when the buttons are kept pressed down.

In the second part of the code I made it so when the player hits an obstacle the scene restarts from ‘0’. I then made another code for when the player goes over the frames, it will also restart from ‘0’.

This code is the general code for the obstacle, the one where it makes the game restart when the player hits the obstacle. Scene Management was used to achieve the reloading of the scene. The player tag was set on the player on Unity, with this there was less coding needed. Since it was labeled on the software itself, it is supposedly always detected as that.

C: > Users > gabri > Documents > GitHub > AssignmentProject > Assignment > Assets > Scripts > ObstacleCollision.cs

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

public class ObstacleCollision : MonoBehaviour
{
    void OnTriggerEnter2D(Collider2D other)
    {
        if(other.tag == "Player")
        {
            Debug.Log("The player has entered into the obstacle");
            //reload the scene when the player hits the obstacle
            SceneManager.LoadScene(0);
        }

    }
}
```

C: > Users > gabri > Documents > GitHub > AssignmentProject > Assignment > Assets > Scripts > Move.cs

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

public class Move : MonoBehaviour
{
    [SerializeField]
    //total speed
    private float _speed = 3f;
    private bool _randomizeHeight = true;
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        //Vector3.left = Vector3(-1,0,0)
        //Vector3.left (-1) * 1 * _speed(3) = -3
        transform.Translate(Vector3.left * Time.deltaTime * _speed);

        //if the position is smaller than -15
        if(transform.position.x < -15)
        {
            if(_randomizeHeight)
            {
                //specific spawn points so the obstacles can be in range with the game
                float randomYPosition = Random.Range(-3f,5f);
                //comment to advize what is happening and gives out the specific spawn point
                Debug.Log("The random position is: " + randomYPosition);
                //the x position is randomized along with the y position
                float randomXPosition = Random.Range(4f, 8f);
                Debug.Log("The random position is: " + randomXPosition);
                //only z is not randomized since everything is surface level
                transform.position = new Vector3(randomXPosition,randomYPosition,0);
            }
        }
    }
}
```

This is the spawner. With this code, the obstacles can randomly respawn after a certain amount of time. I gave the respawner a range so the obstacle does not generate beyond the frame of the game screen. I set a general speed of the obstacle and tweaked it on Unity itself.

This is the main code for the scoring. The code shows that the scoring is going to +1 point every second instead of going past the obstacle.

C: > Users > gabri > Documents > GitHub > AssignmentProject > Assignment > Assets > Scripts > Score.cs

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

public class Score : MonoBehaviour
{
    [SerializeField] private int  startingValue = 0;
    [SerializeField] private int  targetValue = 0;
    [SerializeField] private float incrementTime = 0;
    [SerializeField] private GameObject enableWhenDone;

    private Text label
    {
        get
        {
            if ( mLabel == null )
                mLabel = GetComponent<Text>();
            return mLabel;
        }
    }
    private Text mLabel;

    private void OnEnable ()
    {
        StartCoroutine(IncrementCoroutine());
    }

    private IEnumerator IncrementCoroutine ()
    {
        float time = 0;
        //tells the text the score so it can go up
        label.text = startingValue.ToString();
        while ( time < incrementTime )
        {
            yield return null;
            time += Time.deltaTime;
            float factor = time / incrementTime;
            label.text = ((int) Mathf.Lerp(startingValue, targetValue, factor)).ToString();
        }
        if ( enableWhenDone != null )
            enableWhenDone.SetActive(true);
        yield break;
    }
}
```

C: > Users > gabri > Documents > GitHub > AssignmentProject > Assignment > Assets > Scripts > ScoreKeeper.cs

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

public class ScoreKeeper : MonoBehaviour
{
    [SerializeField]
    private int _currentScore = 0;
    // Start is called before the first frame update
    void Start()
    {

    }

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

    }

public void IncrementScore()
    {
        //_currentScore + 1
        //_currentScore += 1
        _currentScore++;
        Text scoreText = GetComponent<Text>();
        scoreText.text = "Score: " + _currentScore.ToString();
    }
}

```

This is another code for the scoring but for a different use. This code is the one where it actually increments the scoring system onto Unity. The scoring code above is quoted and code relating to the text is added along with it so the points in the game can go up.

c. Process

I first started with the player. I added a 2-Dimentional cube to the workspace and added a script. A box collider was added at the time since it was simple a cube at first. The only script I added for the player was the 'Wick' script. The colour of the background was shortly adjusted to a reasonable colour. Afterwards, the obstacles were added with specific placing so they would spawn at particular times. For the player to actually collide with the obstacle I had to add in physics which is why rigidbody was added and I ticked 'is trigger'. I added both the 'Move' and the 'ObstacleCollision' scripts. After that I created a canvas and added text so the scoring system could be added. I gave the text box an a generous amount of space so the scoring can go up without getting cut off then I made the scripts. I added both the 'Score' and the 'ScoreKeeper' scripts. Finally, I switched out all the cubes with their respective art assets and added a polygon collider since they were irregular shapes.

d. Testing

Test 1: The obstacles kept on overlapping which causes a mess up with the positioning and could easily kill the player off since some obstacles have different moving speed. The controls were going smoothly and the player died whenever it collided with the enemy.

Test 2: It was noticed how the ends of the player hit the enemy but the game still did not restart and kept on going. When the middle section of the player hits the obstacle, the game restarts. The rest of the controls work perfectly, the collider was just glitching.

Test 3: It was pointed out that the player is larger than the enemies but the player still takes two steps to fully dodge the enemy.

Test 4: The overlapping of the obstacles is causing the player to get killed off without it being the users fault since sometimes they all spawn in as a big chunk.

Art Assets & task 3 part a: Shanice Fenech

Coding, Unity & task 3 part b, c, d and excel sheet: Gabriel Noah Vella