Game Programming assignment 2 report

Assignment #2 IS71030C^2023-24^1

IS71030C/B: GAMES PROGRAMMING 1 (2023-24)

Wenqu Tang

33811689

21 January 2024

This is life

itch.io Page

https://leotang.itch.io/this-is-life?password=goldsmiths

Playtesting Video

This is life playtesting video – YouTube

OneDrive

GameProgramming_CW2.zip

Overview

This game is a metaphor for life. **We live in a limited space,** the little fox stands for you and me. In this game, you can never turn back, so does life. Along the way, you will encounter many setbacks, moments of joy, and wonderful experiences. **The hardships you endure will quickly fade away, while the achievements you reach and the goodwill you accumulate will hand you the keys to a broader world.**

The game logic include:

- Blocks will disintegrate once the player steps on them, compelling players to move swiftly and make quick decisions.

- Mushrooms scattered throughout will cause a dizzying effect for five seconds, but players can counteract this by finding hearts.

- The trees would present a dimensionally condensed world—the fox becomes a 2D picture that moves around.

- After collecting five buttons, player can claim the key to next level.

Due to time constraints, some features are still in development. To enhance the game's appeal, I will continue to enrich the experiences within the trees, designing 5 different 2D mini games to provide a more entertaining for players. Additionally, I plan to incorporate more well-designed UI elements to improve user-friendliness.

Technical Overview

What I plan to do is something small but intrincate, since I have to do this in one month. I chose small maps to fit the limited space theme. Avoid too complicated functions but develop small features like audio, music, particle effects.

The code can be divided into five subclass, which controls scene object generation, 2D game implementation, game management, data management between scenes, and scene management.
It might not be the standard practice to organize code. I have written down my reflections on how to refactor the code for each section.

Scripts

Scene Object Manager

Scene Object Manager is responsible for generating all the scene object before game starts.

1. The objects include button, mushroom, trees, flowers, heart, and keys. It is all managed random generator script. I use a list to generate the object on the fly.

2. The button script, heartmovement, key script are specific script attached to the object prefabs to control special characteristics. For example, rotation, and particle effect playing. But it can be integrated into one Object Manager to make this less messy.
3. The player behaviour script is even messier... I know. It should be splited into Audio, Postprocessing, UI, and use a switch case instead. But in this way it can be more clear to see what different effect each object would trigger.

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

public class button : MonoBehaviour
{
    public GameObject knob;
    float newYposition = -0.05f;



    void Start()
    {
        Transform knobTransform = transform.Find("knob");
        if (knobTransform != null)
        {
            knob = knobTransform.gameObject;
        }
        else
        {
            Debug.LogError("Knob not found on " + gameObject.name);
        }
    }

    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            knob.transform.localPosition = new Vector3(0, newYposition, 0);
            StartCoroutine(Disappear());
        }
    }
    IEnumerator Disappear()
    {
        yield return new WaitForSeconds(3f);
        gameObject.SetActive(false);
    }

}
```

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

```
using UnityEngine;

public class Floor : MonoBehaviour
{
    public GameObject blockPrefabs;
    public int width = 20;
    public int length = 20;
    public RandomGenerator randomGenerator;
    public List<GameObject> blocks;
    public float acceraltion = -9.81f;

    void Start()
    {
        List<Vector3> blockPositions = FloorGenerator();
        InvokeRepeating("RandomBlockToDisappear", 1.0f, 2.0f);
        if (randomGenerator != null)
        {
            randomGenerator.RandomItem(blockPositions);
        }

    }

    List<Vector3> FloorGenerator()
    {
        List<Vector3> positions = new List<Vector3>();
        if (blockPrefabs == null)
        {
            Debug.Log("Block Prefab not assigned");
            return positions;
        }

        Vector3 currentPosition = transform.position;
        Vector3 newPosition = currentPosition;

        for (int z = 0; z < length; z++)
        {
            for (int x = 0; x < width; x++)
            {
                GameObject blockInstance = Instantiate(blockPrefabs, newPos
                blockInstance.transform.SetParent(transform);
                blocks.Add(blockInstance);

                // add a new object as trigger detector
                GameObject triggerObject = new GameObject("Trigger Detector"
                triggerObject.transform.SetParent(blockInstance.transform);
                triggerObject.transform.localPosition = Vector3.zero;

                // add the same layer as the parent object
                triggerObject.layer = blockInstance.layer;
```

```
52
53                  // add collidors to them
54                  BoxCollider triggerCollidor = triggerObject.AddComponent<Box
55                  triggerCollidor.isTrigger = true;
56
57                  // reshape the box collidor
58                  triggerCollidor.size = new Vector3(3f, 5f, 3f);
59
60                  // assign script to each block instance generated
61                  TriggerScript triggerScript = triggerObject.AddComponent<Tri
62                  triggerScript.blockInstance = blockInstance;
63
64                  // continue to spawn next block
65                  positions.Add(newPosition);
66                  newPosition.x += blockPrefabs.transform.localScale.x;
67              }
68              newPosition.x = currentPosition.x + 0.1f;
69              newPosition.z += blockPrefabs.transform.localScale.z;
70          }
71          return positions;
72      }
73      void RandomBlockToDisappear()
74      {
75          if (blocks.Count != 0)
76          {
77              int index = Random.Range(0, blocks.Count);
78              GameObject blockToDisappear = blocks[index];
79              Rigidbody rb = blockToDisappear.GetComponent<Rigidbody>();
80              rb.isKinematic = false;
81              rb.useGravity = true;
82              StartCoroutine(blockDisappear(blockToDisappear, 5f));
83              blocks.RemoveAt(index);
84          }
85          else
86          {
87              CancelInvoke("RandomBlockToDisappear");
88              Debug.Log("All block has disappeared");
89          }
90      }
91      IEnumerator blockDisappear(GameObject blockToDisappear, float delay)
92      {
93          yield return new WaitForSeconds(delay);
94          if (blockToDisappear.transform.position.y < -20)
95          {
96              blockToDisappear.SetActive(false);
97          }
98      }
99  }
100
```

```
 public class TriggerScript : MonoBehaviour
{
    public GameObject blockInstance;
    public Rigidbody rb;

    private void OnTriggerEnter(Collider other)
    {
        Rigidbody rb = blockInstance.GetComponent<Rigidbody>();
        if (other.CompareTag("Player"))
        {
            // make the block disappear
            StartCoroutine(Delay(blockInstance, rb));
        }
    }

    public IEnumerator Delay(GameObject blockInstance, Rigidbody rb)
    {
        yield return new WaitForSeconds(0.5f);
        rb.isKinematic = false;
        rb.useGravity = true;

        yield return new WaitForSeconds(2f);
        blockInstance.SetActive(false);
    }
}
```

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

public class Key : MonoBehaviour
{
    private float rotationSpeed = 100f;
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        Rotation();
    }

    private void Rotation()
    {
        transform.Rotate(Vector3.right, rotationSpeed * Time.deltaTime);
```

```
    }
}
```

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

public class heartMovement : MonoBehaviour
{
    private Vector3 startPosition;
    private Vector3 endPosition;
    private float moveSpeed = 1f;
    GameObject heart;
    private float rotateSpeed = 100f;
    private bool heartMoving;

    // Start is called before the first frame update
    void Start()
    {
        startPosition = transform.position;
        endPosition = startPosition + new Vector3(0f, 2f, 0f);
    }

    private void Update()
    {
        heartMove();
        heartRotate();
    }

    void heartMove()
    {
        float step = moveSpeed * Time.deltaTime;
        if (heartMoving)
        {
            transform.position = Vector3.MoveTowards(startPosition, endPosit
            if (transform.position == endPosition)
            {
                heartMoving = false;
            }
        }
        else
        {
            transform.position = Vector3.MoveTowards(endPosition, startPosit
            if (transform.position == startPosition)
            {
                heartMoving = true;
            }
        }
```

```
    }

    void heartRotate()
    {

        transform.Rotate(Vector3.up, rotateSpeed * Time.deltaTime);
    }
}
```

```
using StarterAssets;
using UnityEngine;
using UnityEngine.Rendering.PostProcessing;
using TMPro;


public class playerBehavior : MonoBehaviour
{
    //postprocessing effect
    private PostProcessVolume postProcessVolume;
    public PostProcessProfile normalProfile;
    public PostProcessProfile blurredProfile;
    public sceneManager sceneManager;
    //count update
    public TextMeshProUGUI heartCountUI;
    public int heartCount = 0;
    public int buttonCount = 0;
    public int keyCount = 0;
    //object interaction
    private float timer = 0f;
    public bool mushroomEffectPlaying;
    //UI update
    public TextMeshProUGUI buttonCountUI;
    public TextMeshProUGUI keyCountUI;
    //audio controller
    public AudioSource bgmSource;
    public AudioSource buttonSource;
    public AudioSource mushroomSource;
    public AudioSource flowerSource;
    public AudioSource keySource;
    public AudioSource heartSource;
    // player manager
    public PlayerManager playerManager;

    private void Start()
    {
        playerManager.LoadInformation();
        bgmSource.Play();
        postProcessVolume = FindObjectOfType<PostProcessVolume>();
```

```
        if (postProcessVolume != null)
            postProcessVolume.profile = normalProfile;
        if (heartCountUI != null && buttonCountUI != null && keyCountUI != r
        {
            heartCountUI.text = heartCount.ToString();
            buttonCountUI.text = (buttonCount.ToString()+ "/5");
            keyCountUI.text = (keyCount.ToString()+ "/1");
        }
        mushroomEffectPlaying = false;

    }

    public void Update()

{
        if (mushroomEffectPlaying == true)
        {
            postProcessVolume.profile = blurredProfile;
            timer += Time.deltaTime;
            if (timer > 5)
            {
                postProcessVolume.profile = normalProfile;
                mushroomEffectPlaying = false;
                timer = 0;
                bgmSource.Play();
            }
        }
    }

    private void OnTriggerEnter(Collider other)
    {

        if (other.CompareTag("Heart"))
        {
            mushroomEffectPlaying = false;
            postProcessVolume.profile = normalProfile;
            other.gameObject.SetActive(false);
            heartCount++;
            heartCountUI.text = heartCount.ToString();
            heartSource.Play();
        }
        if (other.CompareTag("Mushroom"))
        {
            mushroomEffectPlaying = true;
            mushroomSource.Play();
            other.gameObject.SetActive(false);
            bgmSource.Stop();
        }
        if (other.CompareTag("Button"))
```

```
        {
            buttonCount++;
            buttonSource.Play();
            buttonCountUI.text = buttonCount.ToString();
        }
        if (other.CompareTag("Key"))
        {
            keyCount++;
            keyCountUI.text = keyCount.ToString();
            other.gameObject.SetActive(false);
            if (buttonCount >= 5 && keyCount >= 1)
            {
                keySource.Play();
                sceneManager.win();
            }
        }
        if (other.CompareTag("Tree"))
        {
            playerManager.SaveInformation();
            sceneManager.NextLevel();
        }
        if (other.CompareTag("Flower"))
        {
            flowerSource.Play();
        }

    }
}
```

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


public class RandomGenerator : MonoBehaviour
{
    public Floor floorScript;

    [System.Serializable]
    public class ItemData
    {
        public GameObject prefab;
        public int count;
        public float yRange;
    }
    public ItemData[] itemDataArray;
    public Vector3 playerStartPosition;
    public float radius = 20f;

```

```
    private void Start()
    {
        GameObject player = GameObject.FindGameObjectWithTag("Player");
        if (player != null)
        {
            playerStartPosition = player.transform.position;
        }
    }

    public void RandomItem(List<Vector3> positions)
    {
        foreach (ItemData itemdata in itemDataArray)
        {
            if (itemdata.prefab != null && itemdata.count > 0)
            {
                positions = ShuffleList(positions);
                int maxitems = Mathf.Min(positions.Count, itemdata.count);
                for (int i = 0; i < maxitems; i++)
                {
                    if (Vector3.Distance(positions[i],playerStartPosition) <
                    {
                        continue;
                    }
                    Vector3 adjustedPosition = new Vector3(positions[i].x, p
                    Instantiate(itemdata.prefab, adjustedPosition, itemdata.
                    positions.RemoveAt(i);
                }
            }
        }
    }

    private List<T> ShuffleList<T>(List<T> inputList)
    {
        for (int i = 0; i < inputList.Count; i++)
        {
            T temp = inputList[i];
            int randomIndex = UnityEngine.Random.Range(i, inputList.Count);
            inputList[i] = inputList[randomIndex];
            inputList[randomIndex] = temp;
        }
        return inputList;
    }
}
```

Small 2D Game

Here is everything I code for the mini 2D game. A simple character controller for dealing with horizontal movement. Game Manager for loading scenes, item Spawner to instantiate the item.

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

public class characterController : MonoBehaviour
{
    public float movingSpeed = 0.5f;
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        float horizontalInput = Input.GetAxis("Horizontal");
        transform.Translate(Vector2.right * horizontalInput * movingSpeed *
    }
}
```

```
using System.Collections;
using System.Collections.Generic;
using System.Threading;
using UnityEditor.Rendering;
using UnityEngine;

public class itemSpawner : MonoBehaviour
{
    public GameObject[] itemPrefabs;
    public float spawnInterval = 3f;
    public float speed = 2f;
    public float timer;
    public List<GameObject> itemArray;

    void Start()
    {

    }

    void Update()
    {
        timer += Time.deltaTime;
        if (timer > spawnInterval)
        {
            timer = 0f;
            Spawner();
        }
```

```
        ItemMover();
    }

    public void Spawner()
    {
        int index = Random.Range(0, itemPrefabs.Length);
        float screenTop = Camera.main.ScreenToWorldPoint(new Vector3(0, Scre
        float randomX = Random.Range(Camera.main.ScreenToWorldPoint(new Vect
        Vector2 spawnerPosition = new Vector2(randomX, screenTop);
        GameObject ItemInstance = Instantiate(itemPrefabs[index], spawnerPos
        itemArray.Add(ItemInstance);
    }
    public void ItemMover()
    {
        float screenBottom = Camera.main.ScreenToWorldPoint(new Vector3(0, 0
        for (int i = itemArray.Count - 1; i >= 0; i--)
        {
            GameObject itemToFall = itemArray[i];
            if (itemToFall != null)
            {
                itemToFall.transform.position = new Vector3(
                itemToFall.transform.position.x,
                itemToFall.transform.position.y - speed * Time.deltaTime,
                itemToFall.transform.position.z
                );
                if (itemToFall.transform.position.y < screenBottom)
                {
                    Destroy(itemToFall);
                    itemArray.RemoveAt(i);
                }
            }
        }
    }
}
```

```
using TMPro;
using UnityEngine;
using UnityEngine.SceneManagement;

public class gameManager2 : MonoBehaviour
{
    public float roundTime = 60.0f;
    private float timer;
    public TextMeshProUGUI timerUI;
    public eatItem eatItem;
    public GameObject winUI;
    public PlayerManager playerManager;

```

```
    private void Start()
    {
        timer += Time.deltaTime;
        timerUI.text = timer.ToString();
        winUI.SetActive(false);
    }

    void Update()
    {
        if (eatItem.score >= 30)
        {
            winUI.SetActive(true);
        }
    }

    public void EndGame()
    {
        SceneManager.LoadScene("SampleScene");
        playerManager.LoadInformation();

    }
}
```

```
using TMPro;
using UnityEngine;

public class eatItem : MonoBehaviour
{
    public int score = 0;
    public TextMeshProUGUI scoreUI;
    public TextMeshProUGUI timerUI;
    public float time = 0;
    public AudioSource dingSource;

    private void Update()
    {
        scoreUI.text = ("Score: " + score.ToString() + "/30");
        time += Time.deltaTime;
        timerUI.text = ("Time: " + score.ToString());

    }

    public void OnCollisionEnter2D(Collision2D collision)
    {
        if (collision.gameObject.CompareTag("Item"))
        {
            Destroy(collision.gameObject);
            Score(1);
```

```
            dingSource.Play();
        }
    }

    public void Score(int Value)
    {
        score += Value;
    }
}
```

Game Manager

Game manager controls the game logic.

In fact, game manager can be used to control audio play, UI activity. In my code, I do this the other way around. So I can try to refactor the code by moving the code in player behaviour to here.

```
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class gameManager : MonoBehaviour
{
    public Transform playerTransform;
    public sceneManager sceneManager;

    public void Start()
    {
        if (playerTransform == null)
        {
            GameObject playerObj = GameObject.FindGameObjectWithTag("Player'
            if (playerObj != null)
            {
                playerTransform = playerObj.transform;
            }
            else
            {
                Debug.LogError("Player object not found. ");
            }
        }
    }
    public void Update()
    {
        if (playerTransform != null && playerTransform.position.y < -3)
        {
            sceneManager.lose();
        }
    }
}
```

Data Manager

Data Manager is used to manage the data when loading and exiting different scenes.

1. Because in my scene, player would enter the 2D game and come back, with all the previous data lost. It would be such a disappointing experience.
2. And I have to delete the data after game overs, so that it won't be loaded next time.
3. I don't know how to do this, and follow some tutorial to run it. It took me ages to get this done. Unfortunately, I didn't manage to store the number of the blocks and the missing block position due to its complexity.

```
using Newtonsoft.Json;
using System;
using System.IO;
using UnityEngine;

public class JsonDataService
{
    public bool SaveData<T>(string relativePath, T data)
    {
        string path = Application.persistentDataPath + relativePath;

        try
        {
            string jsonData = JsonConvert.SerializeObject(data, Formatting.I
            File.WriteAllText(path, jsonData);
            Debug.Log("Data saved successfully.");
            return true;
        }
        catch (Exception e)
        {
            Debug.LogError($"Unable to save data: {e.Message}");
            return false;
        }
    }

    public T LoadData<T>(string relativePath)
    {
        string path = Application.persistentDataPath + relativePath;

        try
        {
            if (File.Exists(path))
            {
                string jsonData = File.ReadAllText(path);
                return JsonConvert.DeserializeObject<T>(jsonData);
            }
            else
            {
                Debug.LogWarning($"File not found: {path}");
```

```
                return default(T);
            }
        }
        catch (Exception e)
        {
            Debug.LogError($"Failed to load data: {e.Message}");
            throw;
        }
    }

    public void DeleteData(string relativePath)
    {
        string path = Application.persistentDataPath + relativePath;
        if (File.Exists(path))
        {
            File.Delete(path);
            Debug.Log("Data file deleted successfully.");
        }
        else
        {
            Debug.LogWarning("Data file not found.");
        }
    }

}
```

```
using UnityEngine;
using System;

[System.Serializable]
public class SerializableVector3
{
    public float x, y, z;
    public SerializableVector3(float rX, float rY, float rZ)
    {
        this.x = rX;
        this.y = rY;
        this.z = rZ;
    }
    public Vector3 ToVector()
    {
        return new Vector3(x, y, z);
    }
    public static SerializableVector3 FromVector(Vector3 vector)
    {
        return new SerializableVector3(vector.x, vector.y, vector.z);
    }
}
```

```
23
24  [System.Serializable]
25  public class PlayerData
26  {
27      public SerializableVector3 PlayerPosition;
28      public int ButtonCount;
29      public int HeartCount;
30  }
31
32  public class PlayerManager : MonoBehaviour
33  {
34      public static PlayerManager Instance;
35      public Vector3 playerPosition;
36      public playerBehavior playerBehavior;
37      public int buttonCount;
38      public int heartCount;
39      public GameObject player;
40
41      private JsonDataService jsonDataService = new JsonDataService();
42      private const string DataFile = "playerSave.json";
43
44      private void Awake()
45      {
46          if (Instance == null)
47          {
48              Instance = this;
49              DontDestroyOnLoad(gameObject);
50          }
51          else if (Instance != this)
52          {
53              Destroy(gameObject);
54          }
55      }
56      public void Start()
57      {
58          player = GameObject.FindGameObjectWithTag("Player");
59      }
60
61      public void SaveInformation()
62      {
63          if (playerBehavior != null)
64              try
65              {
66                  Debug.Log("Saving player information...");
67                  PlayerData data = new PlayerData
68                  {
69                      PlayerPosition = SerializableVector3.FromVector(player.t
70                      ButtonCount = playerBehavior.buttonCount,
71                      HeartCount = playerBehavior.heartCount
```

```
                };
                jsonDataService.SaveData(DataFile, data);
            }
            catch (Exception e)
            {
                Debug.LogError("Error when saving data:" + e);
            }
    }

    public void LoadInformation()
    {
        if (playerBehavior != null)
            try
            {
                PlayerData data = jsonDataService.LoadData<PlayerData>(DataF

                if (data != null)
                {
                    Debug.Log("Loading player information...");
                    player.transform.position = data.PlayerPosition.ToVector
                    playerBehavior.buttonCount = data.ButtonCount;
                    playerBehavior.heartCount = data.HeartCount;
                }
            }
            catch (Exception e)
            {
                Debug.LogError("Error when loading data:" + e);
            }
    }

    private void OnApplicationQuit()
    {
        DeletePlayerData();
    }

    private void DeletePlayerData()
    {
        if (jsonDataService != null)
        {
            jsonDataService.DeleteData(DataFile);
        }
    }
}
```

Scene Manager

The Scene Manager takes control of the loading and exiting logic of every scenes.

```
using UnityEngine;
```

```
using UnityEngine.SceneManagement;

public class sceneManager : MonoBehaviour
{

    public void win()
    {
        Debug.Log("win");
        SceneManager.LoadScene("win");
    }

    public void lose()
    {
        Debug.Log("lose");
        SceneManager.LoadScene("gameOverScene");
    }

    public void FirstLevel()
    {
        SceneManager.LoadScene("SampleScene");
        Debug.Log("replay game");
    }

    public void NextLevel()
    {
        SceneManager.LoadScene("2D");
    }

    public void QuitGame()
    {
        Application.Quit();
        Debug.Log("quit game");
#if UNITY_EDITOR
        UnityEditor.EditorApplication.isPlaying = false;
#endif
    }
}
```

Reflection

Playtesting

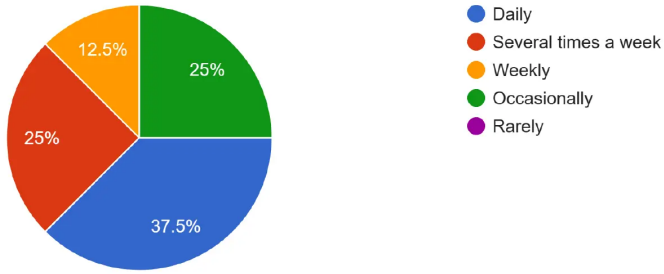
Phase 1: Questionnaire for general opinion and critical bugs

https://docs.google.com/forms/d/1bflBch20rJnnxbRsmtKIMkpUAwc7oCGQ5rma0QaITMY/edit

Results: I received 11 reponses and listed their suggestions below to fix bugs. My reflections is attached to each step.

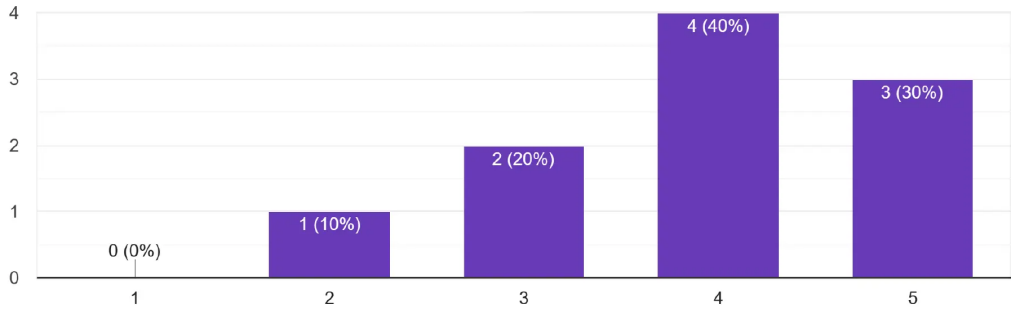
How often do you play video games?

8 responses



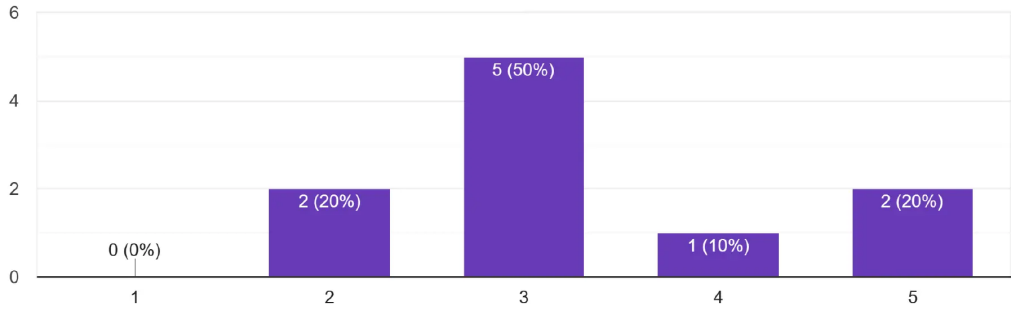
What was your initial impression of the game?

10 responses



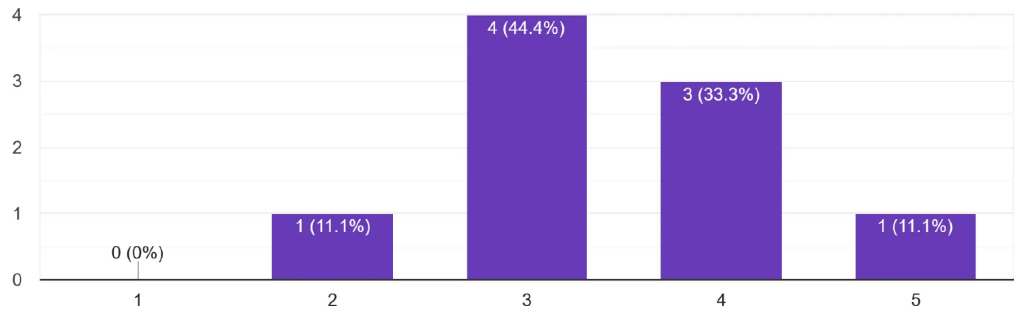
How would you rate the visual appeal of the game?

10 responses



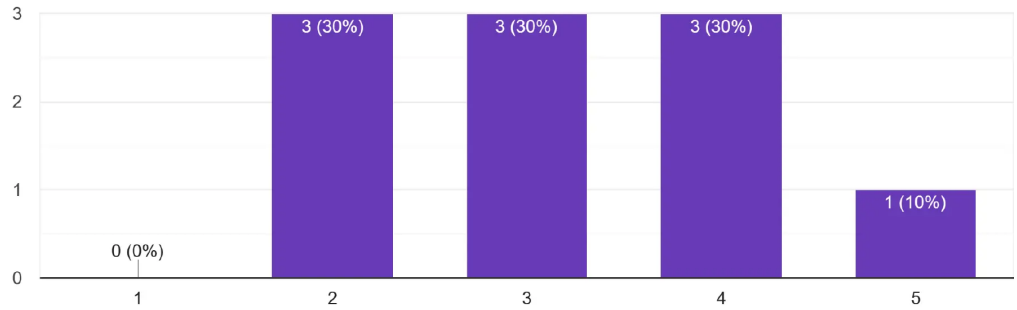
How intuitive did you find the game controls?

9 responses



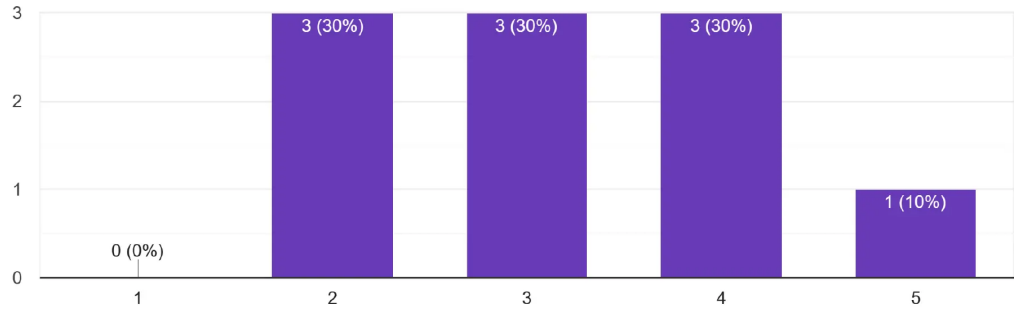
How would you rate the difficulty lev of the game?

10 responses



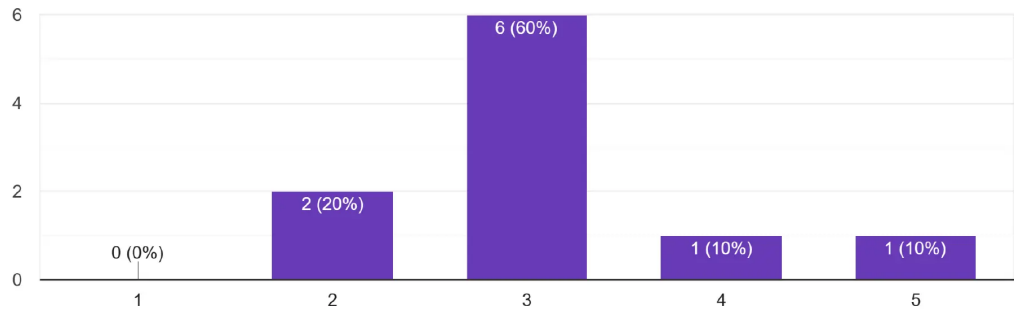
How engaging did you find the gameplay?

10 responses



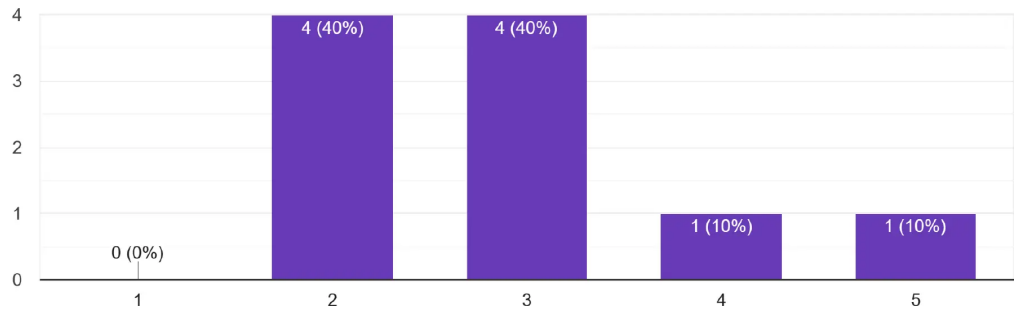
How compelling did you find the game's story?

10 responses



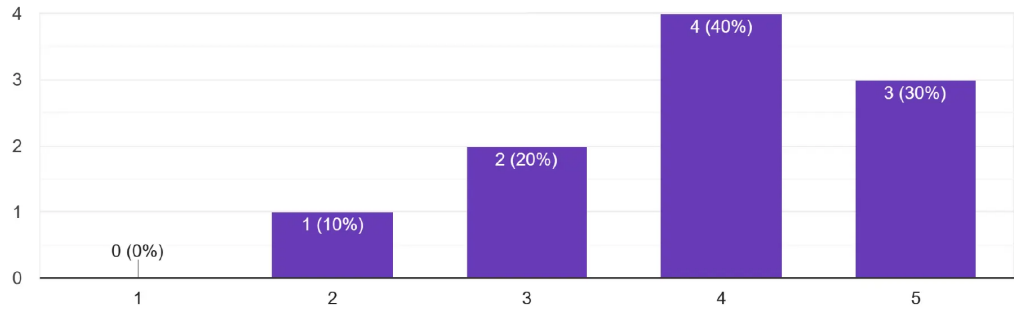
How would you rate the development and depth of the character in the game?

10 responses



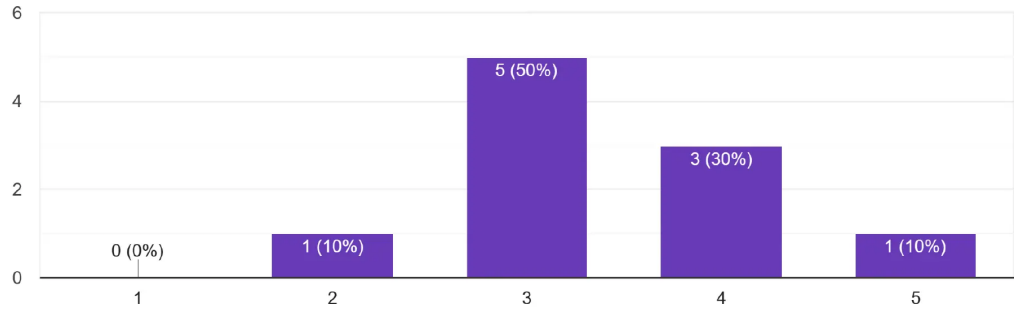
How would you rate the game's music and sound effects?

10 responses



Overall, how satisfied are you with the game?

10 responses



Do you have any suggestions for improvements or features you would like to see in the final version of the game?

10 responses

The number of mushrooms is a bit too much...

Sometimes keys are generated in trees and are impossible to get.

You can change the color scheme, such as a sweeter style

I found that the fox's feet did not touch the ground, which affected my sense of immersion.

The map is too big, there are too many open spaces

Sometimes there is an error in the counting UI and my score is missed.

I was quite delighted to find the 2D game you set up here, plz add more hidden games!

I don't know how to trigger the 2D game? it just happened....

I won the mini game but without any award, I am little confused about why am I doing this? And I died immediately after I come back.

- [x] ~~The number of mushrooms is a bit too much...~~

I cut down the number of mushrooms.

- [x] ~~Sometimes keys are generated in trees and are impossible to get.~~

Add RemovedAt to the list to avoid the repeated generation on same spot

- [x] ~~You can change the color scheme, such as a sweeter style~~

At first, my game is a lavatory theme, I aimed to make realistic lavas, trying to make the falling floor more sense and real. My character looks like this in my first version:

I downloaded the model from Kenny's website. It is really ugly and makes my game weird. To make real lava erupation theme is too challenging. I opted for the Fall Guys style after a bit of research. This kind of game theme is cheap, quick and makes you happy.

The most difficult part is the skybox. I browsed every free skybox website to find a suitable one for my game. They are either too realistic or too simple. Overall, they are crap. Then I came across the website that allows users to generate Skybox using Generative AI. I prompted "Clean blue sky, cute clouds Super Mario style". And I select "clay" style. At first I was just expecting some clouds, but it came out to be more delightful details. I added postprocessing to it, making the colour more dreamy.

- [x] ~~I found that the fox's feet did not touch the ground, which affected my sense of immersion.~~

Bugs fixed by reimport third person starter assets.

- [x] ~~The map is too big, there are too many open spaces~~

  Shrink the map size from 50*50 to 20*20. It is more crowded and adds more fun.

- [x] ~~Sometimes there is an error in the counting UI and my score is missed.~~

  The UI is really a pain in developing. The screen size of WebGL and Unity doesn't match. It looks different on WebGL resulting completely different experience. The UI would be covered by other elements and cannot trigger its functions. Really took me ages to fix these.....

- [ ] I was quite delighted to find the 2D game you set up here, plz add more hidden games!

  In the future.

- [x] ~~I don't know how to trigger the 2D game? it just happened....~~

  Player might be confused about how they enter the 2D game. They thought it is a random thing, while in fact, it is triggered by entering the trees.

- [ ] I won the mini game but without any award, I am little confused about why am I doing this? And I died immediately after I come back.

  Bug fixed by adding another delete saved data function.

- [ ] so hard to rotate my view in web, I was always dying

  Bug fixed by adjusting the screen size of itch.io. But it is still really confusing when displayed on different screens, they all have different sizes that leads to bugs.

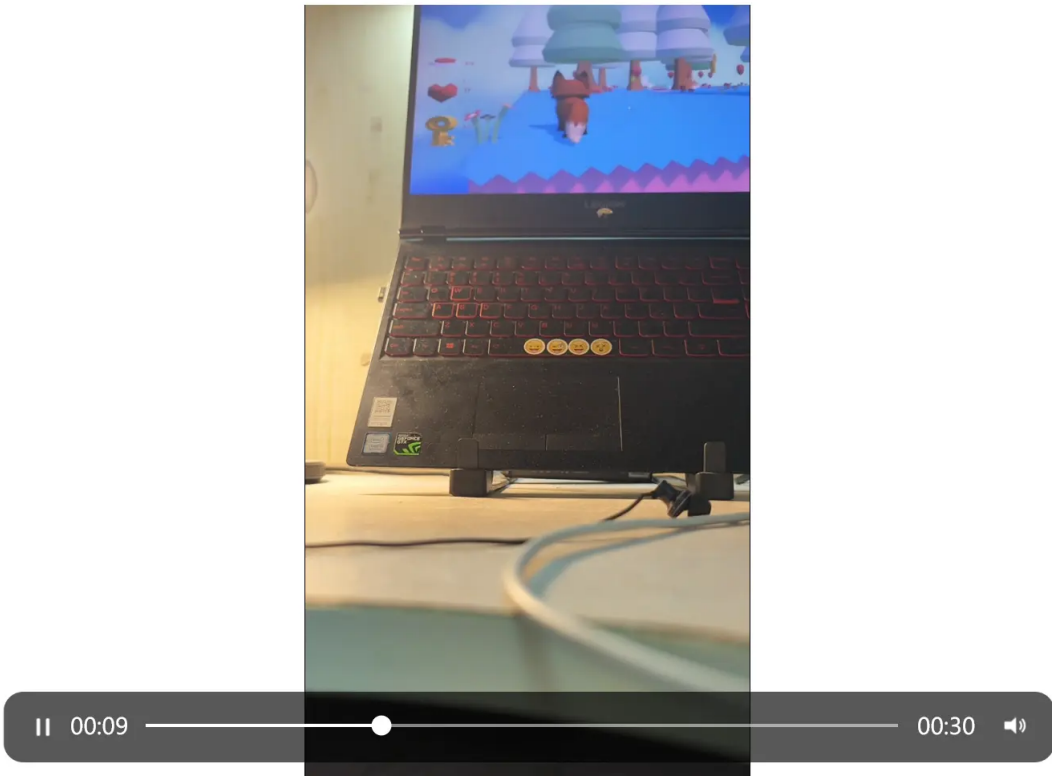
- [ ] make a multiple layer, even player fall down the first layer, they can continue their journey on the lower deck.

  Future, will do.

Phase 2: Interview for Improvement Suggestions

I asked them to record their screen when they are playing. I watched this recordings to see their game experience and intuitive actions towards certain things.
I interviewed two people, 10 mins each. One is via video, another is text on WhatsApp.
Takeaways:

1. They both mentioned that the camera rotation is not smooth, which I didn't manage to solve. Because I already adjust embed the game as the Building Setting size in unity. Couldn't work.
2. They both describe the colors and music I choose as "precise" and like the character.
3. They both says it is too difficult to win. So I make some improvements to lower the difficulties.



Overall, this is a good little game. Although it is not clear what the story is about, the mushroom and fox elements in it can let gamers know that this game takes place in a magical world. The sound effects are great and the level objectives are clear enough. There are also opportunities to trigger other mini-games, which add interest to the game. However, it would have been nice if the game had a few more controls. I would have been happy to see a fox that could sprint through the air, or double jump to save himself when he lost his footing.

Yup 16:23

That looks cute impressed u could put the animation in i could never figure that one lol 16:24

16:31

Also cool effect when blocks fall and the power ups and that r goof 16:32

Known Issues

1. Some data lost after loading into the 2D game. Following the online tutorial, I managed to create a data array that can instore players' position, the count of objects, including loading and saving in different scenes. It can be foreseened that controlling the block information is even more challenging and complex. As a result, when player comes back from the 2D game scene, they will find the falling floor is intact now.
2. Users might repeatly load into the same 2D game if they walk close to them. I meant to design five or more mini 2D game to make the game more entertaining. However, due to time limitation, and the management of different games and data management would be rather difficult for me(And a lot of time and crying!). Therefore, I have programmed one 2D game as a prototype to present my game design.

Accessibility and Impact

Not suitable for people without PC and people with color blindness.
This game use keyboard and mouse as input, not accessible for mobile and console palyers.
This game features dynamic colors, people with color blindness won't find this joyful.

Resource Used

Making fireworks:
Fireworks VFX Effect Particle System| How to make Fireworks using unity Particle System VFX (youtube.com)
Unity Starter Asset:
Starter Assets - ThirdPerson | Updates in new CharacterController package | 必备工具 | Unity Asset Store
Postprocessing:
How we use 100% of profits for the planet (youtube.com)
How To Add Post Processing! | Unity Tutorial (youtube.com)

Skybox:
How to Create a CUSTOM SKYBOX in Unity! (Step by Step Tutorial) (youtube.com)
https://skybox.blockadelabs.com/
I used the website to customize a skybox for my game using prompts. It is so cool!

Particle System:
Particle System Trails | Unity Particle Effects | Visual FX (youtube.com)
I follow this tutorial to make fireworks.

For the small 2D Game:
How to make a 2D Game in Unity (youtube.com)

For Data Storation:
Json.NET - Newtonsoft

Releases · JamesNK/Newtonsoft.Json (github.com)

Data Persistence - Save & load your game state while avoiding common mistakes | Unity Tutorial (youtube.com)

I used the toolkit and revised some code from the github repo.

Unity Asset Store:

Starter Asset for third person controller

Sunny Land | 2D 角色 | Unity Asset Store

I only used the sprite of the fox, the spawn items.

Bubble Font (Free Version) | 2D 字体 | Unity Asset Store

I used the free font to make my user interface more pretty.

Toon Fox | 3D 动物 | Unity Asset Store

I used the free fox model from the asset store as my character.

Platformer Kit · Kenney

I used the 3D models from the Kenny's website for most of my scene object.

Appendix A: Questionnaire

https://docs.google.com/forms/d/1bflBch20rJnnxbRsmtKIMkpUAwc7oCGQ5rma0QaITMY/edit

This is Life Playtesting Survey

Thank you for participating in the testing of our game. Your feedback is invaluable to us. Please take a few minutes to fill out this survey. Your responses are anonymous, and we appreciate your honest opinions.

1. How often do you play video games?
2. What was your initial impression of the game?
3. How would you rate the visual appeal of the game?
4. How intuitive did you find the game controls?
5. How would you rate the difficulty lev of the game?
6. How engaging did you find the gameplay?
7. How compelling did you find the game's story?
8. How would you rate the development and depth of the character in the game?
9. How would you rate the game's music and sound effects?
10. Overall, how satisfied are you with the game?
11. Do you have any suggestions for improvements or features you would like to see in the final version of the game?