# Cat MANOR


Game Design & Game Art & Program : Olivia Iris Tian

Game Link:https://gujiufeng.itch.io/cat-manor-demo

**2D side-scrolling, RPG, puzzle**

## Introduction

A modern-day student travels to a world inhabited entirely by the souls of the deceased. There, a manor serves as a refuge for the cats that perished during the era of witch hunt, trapping the souls of those who were involved in the events. The protagonist's task is to uncover the mysteries of sins and free the trapped souls.

**Made With**

# Inspiration

## Witch hunt during Medieval period

Medieval Europeans, including colonists, saw witchcraft as a heathen practice. In an effor to purify Christianity, the hysteria about witchcraft began, leading to maniac witch hunt and symbolizing a dark period in European history marked by fanaticism.

### Witch Hunt

**16th century**

**17th century**

### Cats' role in Medieval Europe

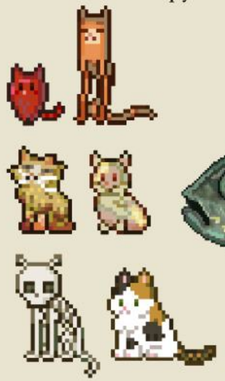
Animals, especially black cats, were seen as integral to witchcraft. Medieval Europeans believed that black cats were the transformations and disguises of witches, and they were often associated with the underworld. This lead to widespread discrimination against wild cats, causing public anxiety among both the nobility and the common people. When the first witch trial began in the 15th century, cats were also accused of aiding witches.

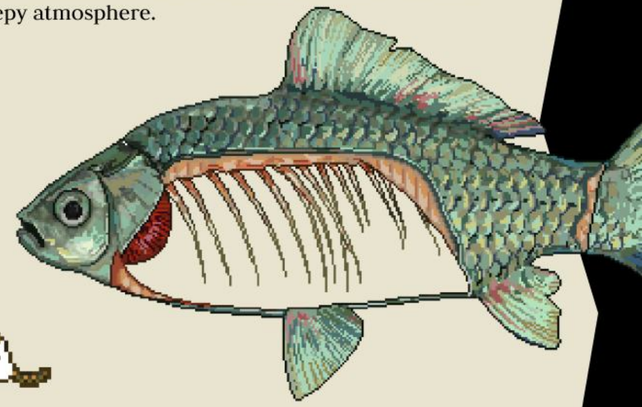
**Black Cat**

**18th century**

# Art Design

For the game, I chose retro pixel art as the primary visual style to highlight its grotesque elements. Most scenes are illustrated in a fairy-tale style, creating a striking contrast between the bright daytime and the eerie nighttime, which enhances the mysterious and creepy atmosphere.





### Character



# Game Flow

Game Start（Player wakes up）


Talk with the first NPC for info

Search for clues or items

Follow NPCS instruction(Spending time)

Day time used up?

no

Talk with other NPCs for info

Monster Chase: Starting during the night. Rocks that can decelerate both the monster and the player will emerge on the street.

Lose game Restart

yes

Enter night mode

Monster chase(Be cought, HP -1)

HP =0

no

yes

Back to shelteron time?

HP =0 or 15day passed

Solve avaliable puzzles

15day passed

Neither

Sleep Enter the next day

Complete the plot?

no

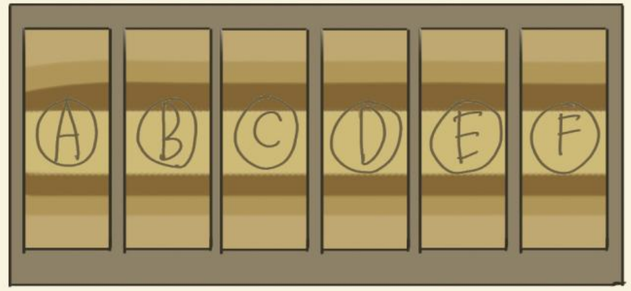
yes

End

The game world map is fairly large, encompassing locations such as the town, the manor, and various rooms, allowing players to explore and discover clues hidden throughout the environment.

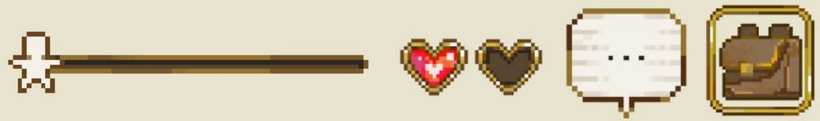
## Puzzle


**Puzzle 1** Feeding cats that guard the door with corresponding food. Each cat corresponds to a specific organ of the fish, and hints are shown in their appearance.




**Puzzle 2** Entering the password of a case. Players must arrange the patterns, which are the coats of arm of ancient British houses, in the correct time order.

# Animation & UI

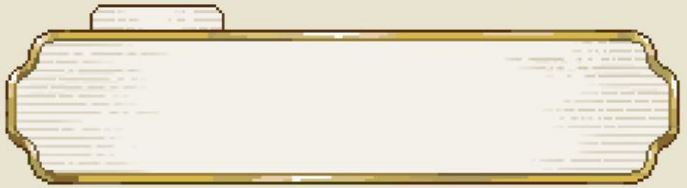


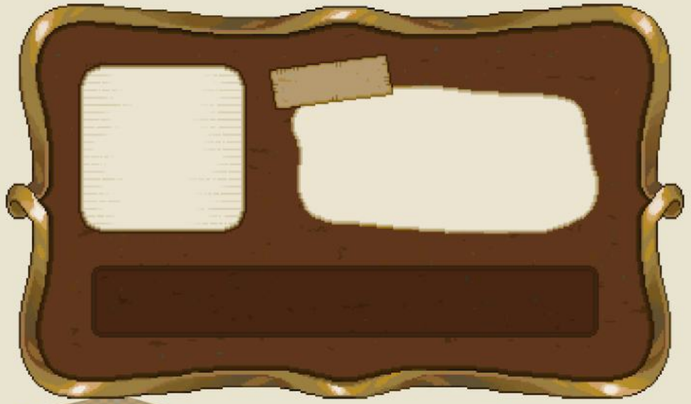


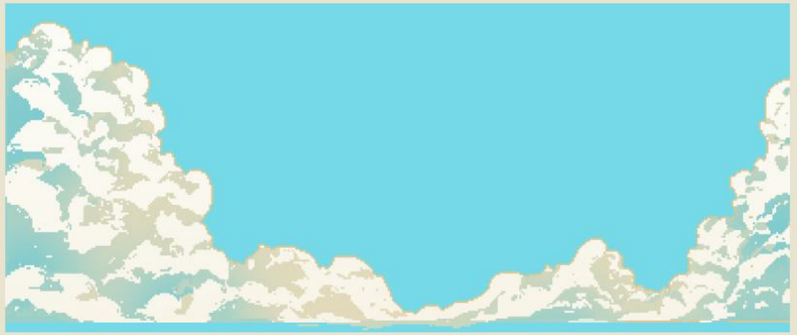




Walk Animation







# Player Move

A/D Horizontal & W/S Vertical Move

```
if (Input.GetKey(KeyCode.A))
{

    rig.velocity = new Vector3(-speed, 0, 0);
    transform.localScale = new Vector3(1, 1, 1);
    Qenter.GetComponent<Transform>().localScale = new Vector3(1, 1, 1);
}
else if (Input.GetKey(KeyCode.D))
{

    rig.velocity = new Vector3(speed, 0, 0);
    transform.localScale = new Vector3(-1, 1, 1);
    Qenter.GetComponent<Transform>().localScale = new Vector3(-1, 1, 1);
}
else if (Input.GetKey(KeyCode.W))
{

    rig.velocity = new Vector3(0, speed, 0);
    transform.localScale = new Vector3(1, 1, 1);
    Qenter.GetComponent<Transform>().localScale = new Vector3(1, 1, 1);

}
else if (Input.GetKey(KeyCode.S))
{
    rig.velocity = new Vector3(0, -speed, 0);
    transform.localScale = new Vector3(1, 1, 1);
    Qenter.GetComponent<Transform>().localScale = new Vector3(1, 1, 1);
}
else
{
    rig.velocity = new Vector3(0, 0, 0);
    this.speed = 0;
    this.timeRunning = 0;
    ani.Play("Player_Idle");
}
```

# Code

## Collecting items into inventory

```
public Item thisItem;
public Inventory playerInventory;

public Inventory DetailPanel;
public bool isPlayerBeside = false;
public GameObject Etake;

Unity 消息|0 个引用
void Start()
{
    playerInventory.itemList.Clear();
    DetailPanel.itemList.Clear();
    Etake.SetActive(false);
}
Unity 消息|0 个引用
void Update()
{
    if (isPlayerBeside&&Input.GetKeyDown(KeyCode.E))
    {
        AddItemIntoBag();

    }
}
public void AddItemIntoBag()
{
    playerInventory.itemList.Add(thisItem);
    DetailPanel.itemList.Add(thisItem);
    InventoryManager.CreateNewItem(thisItem);
    Destroy(gameObject);
}

//识别物品是否被点击
Unity 消息|0 个引用
private void OnMouseDown()
{
    AddItemIntoBag();
}
Unity 消息|0 个引用
private void OnTriggerEnter2D(Collider2D collision)
{
    if (collision.gameObject.CompareTag("Player"))
    {
        isPlayerBeside = true;
        Etake.SetActive(true);
    }

}
Unity 消息|0 个引用
private void OnTriggerExit2D(Collider2D collision)
{
    if (collision.gameObject.CompareTag("Player"))
    {
        isPlayerBeside = false;
        Etake.SetActive(false);
    }
}
```

## Inputting & Checking Password

```
public int[] password;

public int selectingSlot;
public List<Image> slots;

public GameObject highlight;

public int[] selectingSprite;
public List<Sprite> sprites;

public Animator redWoodenCaseAnim;
// Start is called before the first frame update
Unity 消息|0 个引用
void Start()
{
    Close2();
}

// Update is called once per frame
Unity 消息|0 个引用
void Update()
{
    if (Input.GetKeyDown(KeyCode.A))
    {
        SelectLeft();
    }

    if (Input.GetKeyDown(KeyCode.D))
    {
        SelectRight();
    }

    if (Input.GetKeyDown(KeyCode.W))
    {
        SwitchUp();
    }

    if (Input.GetKeyDown(KeyCode.S))
    {
        SwitchDown();
    }

    if (CheckPassWord())
    {
        gameObject.SetActive(false);
        redWoodenCaseAnim.SetBool("Open", true);
    }
}

1 个引用
public void SelectLeft()
{
    if (selectingSlot != 0) return;

    selectingSlot -= 1;
    highlight.transform.position = slots[selectingSlot].transform.position;
}

1 个引用
public void SelectRight()
{
    if (selectingSlot == slots.Count - 1) return;

    selectingSlot += 1;
    highlight.transform.position = slots[selectingSlot].transform.position;
}


1 个引用
public void SwitchUp()
{
    selectingSprite[selectingSlot] += 1;

    if (selectingSprite[selectingSlot] > sprites.Count - 1) selectingSprite[selectingSlot] = 0;
    slots[selectingSlot].GetComponent<Image>().sprite = sprites[selectingSprite[selectingSlot]];
}

1 个引用
public void SwitchDown()
{
    selectingSprite[selectingSlot] -= 1;
    if (selectingSprite[selectingSlot] < 0) selectingSprite[selectingSlot] = sprites.Count - 1;
    slots[selectingSlot].GetComponent<Image>().sprite = sprites[selectingSprite[selectingSlot]];
}

1 个引用
public bool CheckPassWord()
{
    for (int i = 0; i < password.Length; i++)
    {
        if (password[i] != selectingSprite[i])
        {
            return false;
        }
    }

    return true;
}
1 个引用
public void Close2()
{
    gameObject.SetActive(false);
}
```

## Feeding the right cat

```
public class CMCat : MonoBehaviour
{
    public Item NeededItemCat;
    public bool isPlayerBesideCat = false;
    public bool isSelecting = false;
    public static CMCat instance;
    public GameObject slotGrid;
    public GameObject IronRailing;
    public GameObject Efeed;
    // Start is called before the first frame update
    Unity 消息|0 个引用
    void Start()
    {
        Efeed.SetActive(false);
    }
    Unity 消息|0 个引用
    void Awake()
    {
        instance = this;
    }

    // Update is called once per frame
    Unity 消息|0 个引用
    void Update()
    {
        if (isPlayerBesideCat)
        {
            for (int i = 0; i < slotGrid.transform.childCount; i++)
            {
                //print("working");
                var child = slotGrid.transform.GetChild(i);
                var slot = child.gameObject.GetComponent<Slot>();

                if (slot != null)
                {
                    slot.CatBeside = gameObject;
                }
            }
            if (Input.GetKeyDown(KeyCode.E))
            {
                CMBagpackSystem.instance.OpenUI();
                isSelecting = true;
            }


        }
    }
    Unity 消息|0 个引用
    private void OnTriggerEnter2D(Collider2D collision)
    {
        isPlayerBesideCat = true;
        Efeed.SetActive(true);

    }
    Unity 消息|0 个引用
    private void OnTriggerExit2D(Collider2D collision)
    {
        isPlayerBesideCat = false;
        Efeed.SetActive(false);

    }

    1 个引用
    public void RemoveCat()
    {
        InventoryManager.DestroyItem(NeededItemCat.itemName);
        InventoryManager.instance.OnEnable();
        CMBagpackSystem.instance.CloseUI();
        IronRailing.GetComponent<CMIronRailing>().MoveUp();
        Destroy(gameObject);
    }
}
```

# Test Feedback

## Pros

The puzzles, especially the one involving feeding cats according to their bizarre appearance, are interesting. The game art is attractive as well.

## Cons

**1.** Since there is no punishment for providing the wrong food, players can try each option continuously until they pick the correct one.

**2.** The hint is not obvious enough for player to find the clues, which are the letters and the dates on the letters.

**Screenshot**

## Optimisation

1. If players feed the cat with the wrong food, they have to wait until the next day for a second try.
2. Hints for the letters' location, bookshelf, are added to conversations with NPCs.
3. To make the dates more obvious, they are isolated from the texts on letters by empty lines.


## Summary

Video Link : https://youtu.be/JAy7TTVRttQ

"Let's Harvest" is a strategic and educational board game in which players act as farmers, growing crops through the four seasons while competing for weather and manufactured resources. Players must carefully consider and make prudent decisions in managing their crops and resources to win the game. Their goal is to successfully harvest three different crops while navigating seasonal changes and natural disasters.

**Game Design & Game Art : Olivia Iris Tian**

# Art Work

## Research

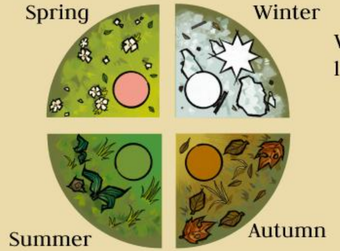
**Don't Starve**

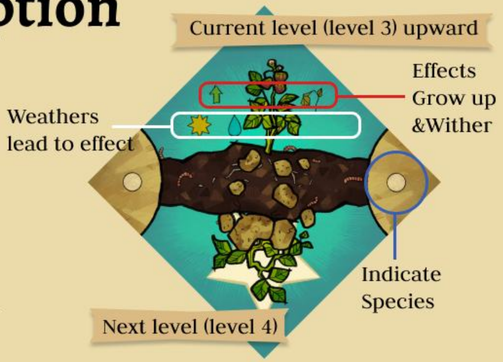
The game's popularity stems not only from its well-designed mechanics but also from its unique art style, characterized by "dark and whimsical visuals: 2D characters and odd creatures inhabiting a distinctive 3D world."


For the art style of the game, I have created image designs with **abundant textures and effects** of rag and dirt. The **irregular patterns** contribute to a sense of raw nature. The 3D installation of the map is designed to provide a natural background and atmosphere, allowing players to fully immerse themselves in their farmer identity within the game.

## Design Iteration


## Cards Description




- With reference to the appearance of real crops and scenery, I have designed **stylized icons** for the cards to fulfill both educational and artistic goals. I conducted research on the growing conditions for all four species in my board game and established corresponding mechanics for each one. Most functions of the cards are expressed through icons instead of words, enabling players to quickly associate the cards with their functions and effects.

- Since potatoes can withstand cold weather and have strong adaptability in reality, potatoes in the board game are minimally affected by cold conditions such as snow. **Unlike other crops, level 2 potatoes cannot be withered by snow.**



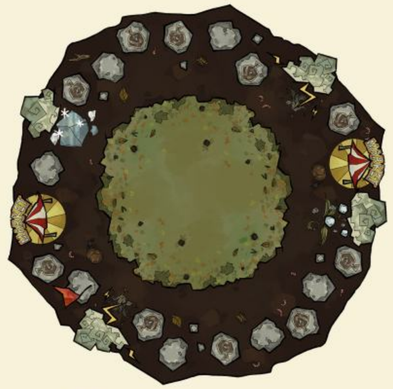









- **Rule Book**

- **1 Map**

- **1 Season Indicator**

- **3 Types of Resource Markers**

Rainwater 20+ Sunlight 20+ Snow 10+

- **1 Four-sided Dice**

- **4 Scarecrow Pieces**

- **4 Types of Weather Cards**

Hail 2+ Rain 10+ Sun 10+ Snow 4+

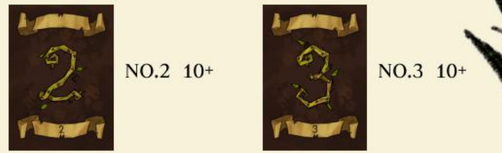
- **4 Types of Seed Cards**

Wheat 10+ Potato 10+ Mooli 10+ Corn 10+

- **2 Types of Function Cards**

Fertilizer 4+ Rotate 4+

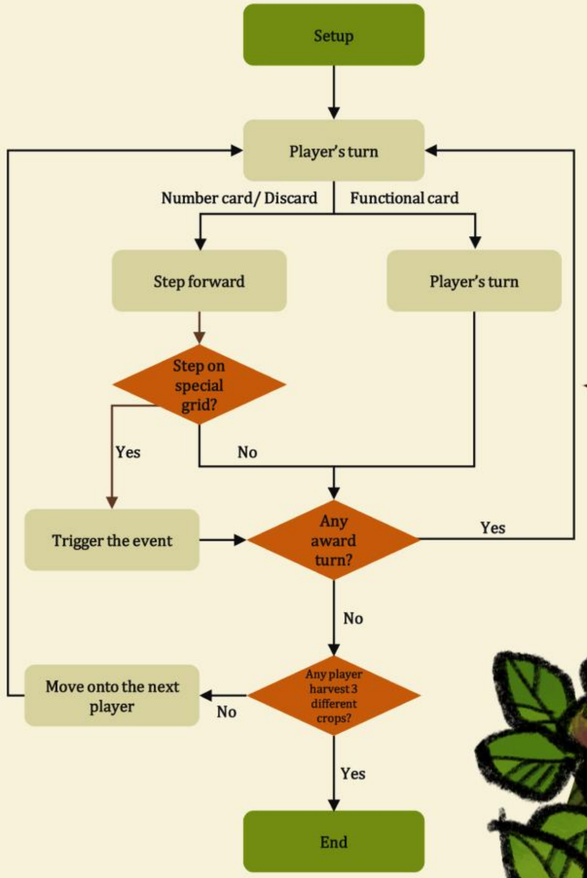
- **2 Types of Number Cards**



NO.2 10+ NO.3 10+

# COMPONENT

# Game Flow




- **Win!**
  The first player who harvests **3 different types out of 4 crops** (by collecting 3 different 4-level crops) wins.

# Game Play

**1**

Each player starts with **6** plots of land. Choose any player to be the starting player.

**2**

Each player draws **5** cards, and at the end of every turn, players should take one new card and always maintain a hand of 5 cards.

**3**

Players can move **1** step counterclockwise in the order of seasons by discarding a functional card. Number cards cannot be discarded.

**4**

Only **1** number card can be played per turn, allowing movement based on the number shown on the card.

**5**

Once a player has planted on all **6 slots** (having 6 seed cards), if they land on the seed module again, they gain 1 extra turn.

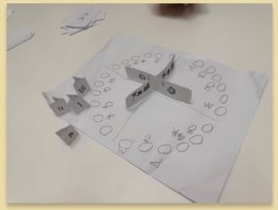
**6**

However, a player can only take up to 4 extra consecutive turns. The extra turn from passing the start point does not count as a consecutive turn.

# Version Iteration

## Version 1.0






## Final Version



# Test Feedback

### Pros

- During gameplay, players learn about the growth habits of various plants, such as their harvesting seasons.
- Players not only need to take care of their own crops but can also enjoy the process of interfering with other players' progress.
- The 3D installation is unique and interesting.

### Cons

- Some players may remain on the same grid by continuously playing resource cards to avoid the weather grid, which undermines the function of the weather grids.
- Once a crop reaches level three, there are limited options-only marketing and a few hail cards-to prevent it from being harvested, while level two crops are prone to withering easily.
- The season indicator cannot be precisely adjusted in 90-degree increments.

### Suggested Improvements for Each Problem:

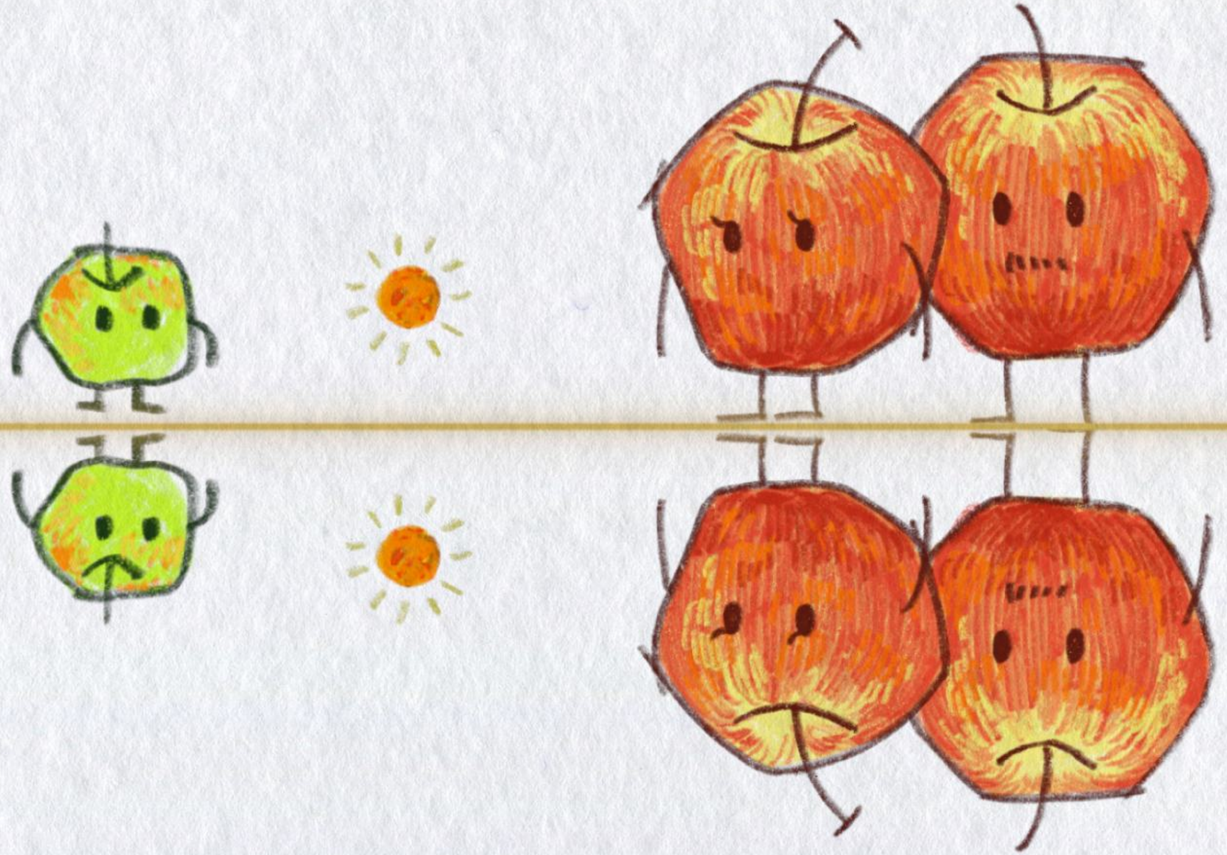
- After staying on the same grid for one turn, players must move in the next round. If they do not, they will be forced to move one step forward and cannot play any cards during that turn.
- Decrease the number of snow cards and increase the number of hail cards.
- Drawing guidelines for the location of the season indicator would be helpful.

# Brain Astronaut

## Summary

Brain Astronaut is a 2D platform adventure game that tells the story of a child with autism exploring a divided fantasy world. The game revolves around the core concepts of dual-world perspective and mental projection. Players will constantly shuttle between the upper and lower fantasy worlds, collecting light to obtain shadows, which allows them to switch positions and overcome insurmountable obstacles, ultimately completing the challenges of each level.

**Game Design & Game Art & Program: Olivia Iris Tian**



## Genre

**#2D Platform** **#Story**

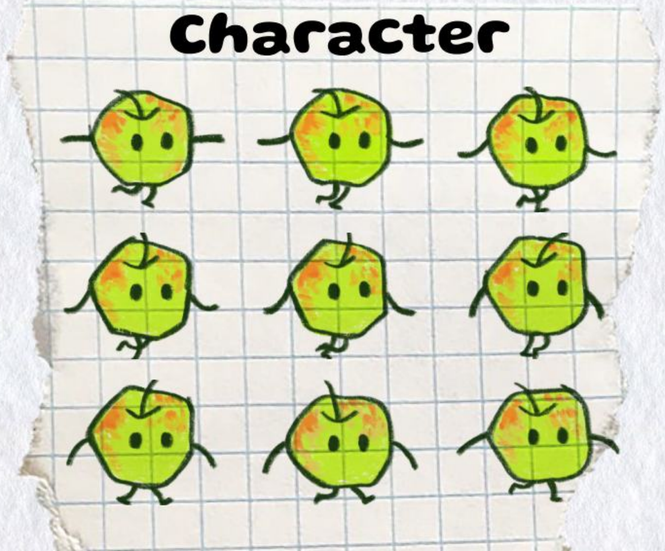
**Game Link: https://gujiufeng.itch.io/brain-astronaunt**

# Research



According to estimates from the CDC's Autism and Developmental Disabilities Monitoring (ADDM) Network, approximately 1 in 36 children is identified with autism spectrum disorder (ASD).

# Character



# Concept

A child with autism wanders in their own world. In the dual-world perspective, the upper side represents the child's imagination (spiritual world), while the lower side reflects reality (indicating that their actions in the imaginary world are mirrored in real life). The true spirit exists only in the imagination, whereas the false shadow is the body in reality. When a child with autism acts differently, as the light illuminates their spirit, players can see what is happening in the child's mind.

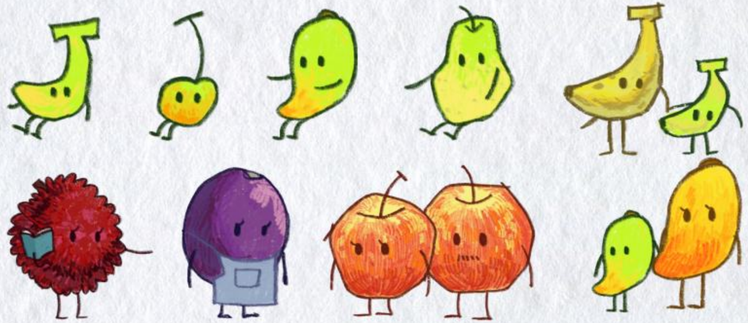
# Game Play

By taking the stars to generate a reflection in the lower side, players can switch positions with the reflection, which can go through any obstacles except for stars

**E: Interact/Get stars**
**Left Shift: Switch position**
**Space: Jump**
**A: Move left**
**D: Move right**

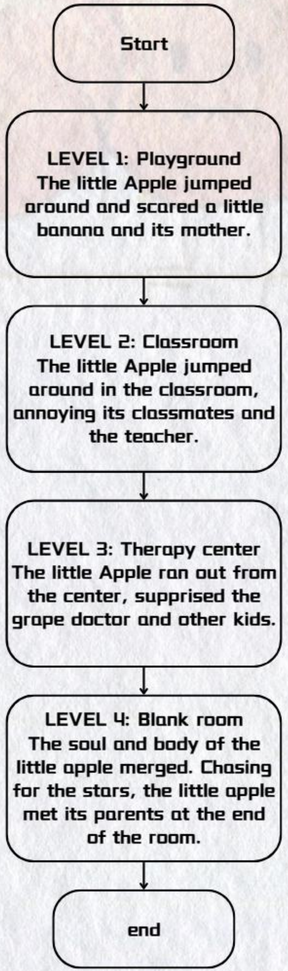
Collections: 0/3


Collections: 1/3

Collections: 2/3


# Game Flow



# Level Design

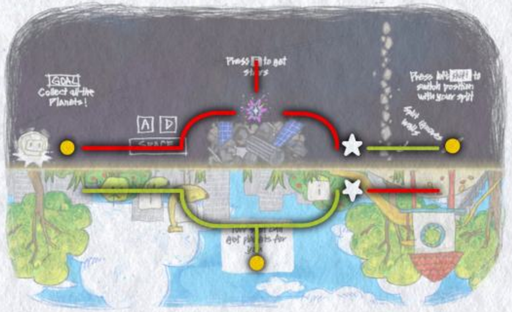
All three levels are designed with **increasing difficulty and complexity of mechanics**. Level 1 serves as the tutorial level. In Level 2, the mechanics of deadly lava is introduced to the player.

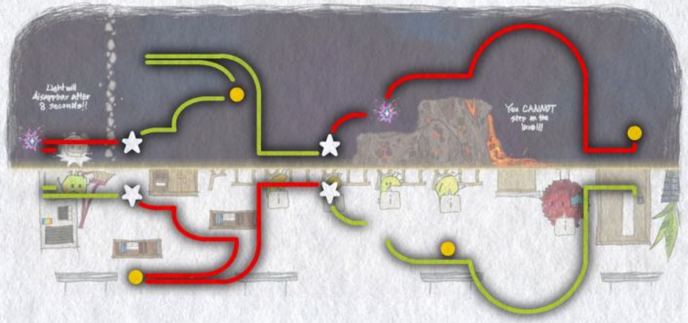
Route If The Slit

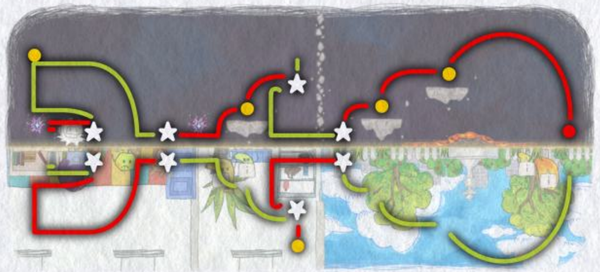
Route If The Player

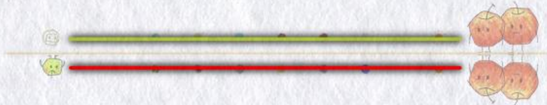
Switch Position

Position Of Planets





# Code

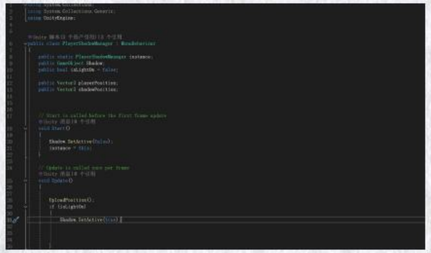
**#Getting the star & star fading out after a certain duration**

```
void Start()
    GL = this;
    rend = GetComponent<Renderer>();
    originalColor = rend.material.color;

// Update is called once per frame
每Unity 调用1次 每帧
void Update()
    if (isPlayerBeside && Input.GetKeyDown(KeyCode.E))
        Player.GetComponent<PlayerShadowManager>().isLightOn = true;
        isFollow = true;
        //fadeIn();
        gameObject.transform.parent = GameObject.Find("Player").transform;
    if (isFollow)
        if (Player.transform.position.y < 0)
            transform.position = new Vector2(Player.transform.position.x, Player.transform.position.y);
        if (Player.transform.position.y > 0)
            transform.position = new Vector2(Player.transform.position.x, Player.transform.position.y);
        Invoke("fadeOut", duration);
        //Invoke("shadowFadeOut", duration);
```

```
public void LightOff()
    Destroy(gameObject);
    Player.GetComponent<PlayerShadowManager>().isLightOn = false;
每Unity 调用1次 每帧用
private void OnTriggerEnter2D(Collider2D collision)
    if (collision.gameObject.layer == LayerMask.NameToLayer("Player"))
        isPlayerBeside = true;
每Unity 调用1次 每帧用
private void OnTriggerExit2D(Collider2D collision)
    if (collision.gameObject.layer == LayerMask.NameToLayer("Player"))
        isPlayerBeside = false;
每 0 次调用
private void fadeOut()
    timer += Time.deltaTime;
    float alphaOut = Mathf.Clamp01(1.0f - (timer / fadeOutDuration));
    rend.material.color = new Color(originalColor.r, originalColor.g, originalColor.b, alphaOut);
    if (timer >= fadeOutDuration)
        LightOff();
每 0 次调用
private void shadowFadeOut()
    ShadowManager.instance.shadowFadeOut();
```

**#Split generation &Position Switching with the split when a star is got**



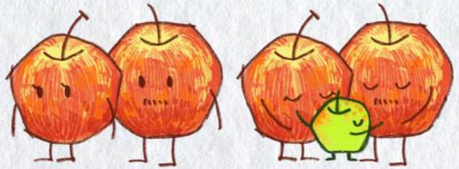
# Scene Design

## Scene 1

## Scene 2

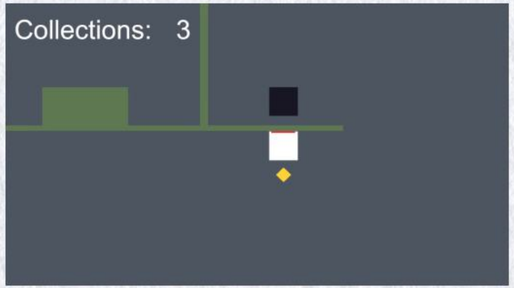
## Scene 3

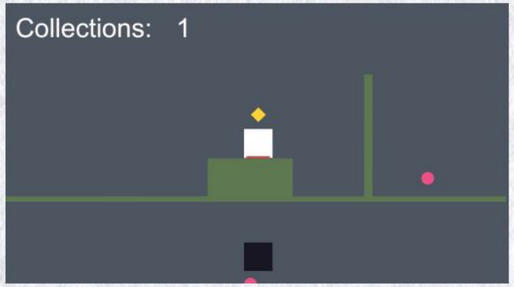
## Scene 4

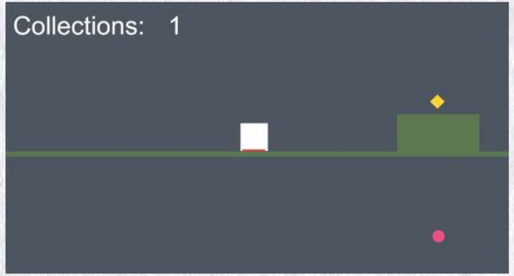


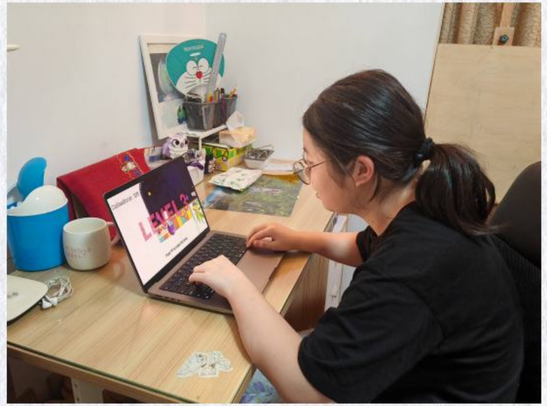
# Iterative Process

The initial version establishes the design for both game mechanics and levels, ensuring that level design aligns well with the mechanics. I identified issues and areas of ineffective design and revised them accordingly.



# Test Feedback

**Pros:**

1. The game mechanics is innovative.
2. The plot, subtly woven into the background and through interactions with non-player characters, is well-designed, with the final scene where the player hugs their parents being especially impactful.

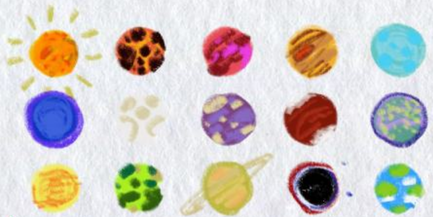
**Cons:**

1. Planets are difficult to spot as they blend into the static background.
2. Instructions lack clarity, making it challenging for players to quickly learn the mechanics.

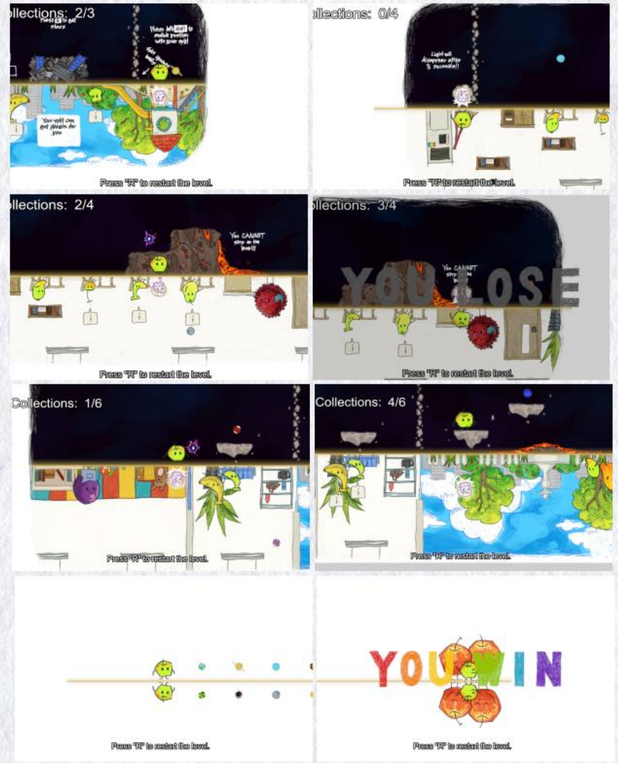
**Improvements:**

1. Ungathered planets now rotate continuously to make them more noticeable.
2. The tutorial level layout has been adjusted to provide space for clearer instructions.

# Art & Animation



# Screenshot




# UI

YOU LOSE

YOU WIN

# The Ruined

"The Ruined" is a 3D adventure game where players control three distinct characters from an archaeological team in the 22nd century, a time when humanity has survived a mass extinction. By overcoming multiple challenges, solving ancient mechanisms, and investigating city ruins and organic remains from the 21st century, players will uncover what happened to humanity.

**Game Design & Game Art: Olivia Iris Tian**

## Character avatar & expressions

### 1. The Biologist:

- Collects and analyzes samples, gathers and interprets information.
- Low HP, low energy

### 2. The Military:

- Skilled in fighting, proficient with weapons, can jump high.
- Moderate HP, high energy

## Monsters Design Draft:

Mutated animals are a major contributor to the disaster. These creatures have been grotesquely transformed into visually striking, crustacean-like forms.

From left to right:
spider, fish, mouse, cat, deer, tiger, squid, whale

### 3. The Geologist:

- Directs and steers the team, responsible for maps, runs fast
- High HP, high energy

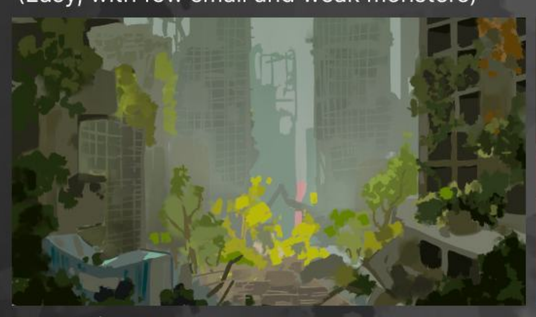
## Gameplay

Players control one character at a time but can switch between characters to decide whose skills to use in different situations.
- Each character has individual hit points (HP) and energy levels.
- Using skills depletes energy, while being attacked by monsters reduces HP.

## Scene Concept Draft:

In the adventure, players will traverse three main terrains: Yellow Wasteland, Eroded City, and Monster Birthplace. The **difficulty increases as players progress toward the center of Monster Birthplace**, which is a meteorite harboring mutating virus that generate monsters.

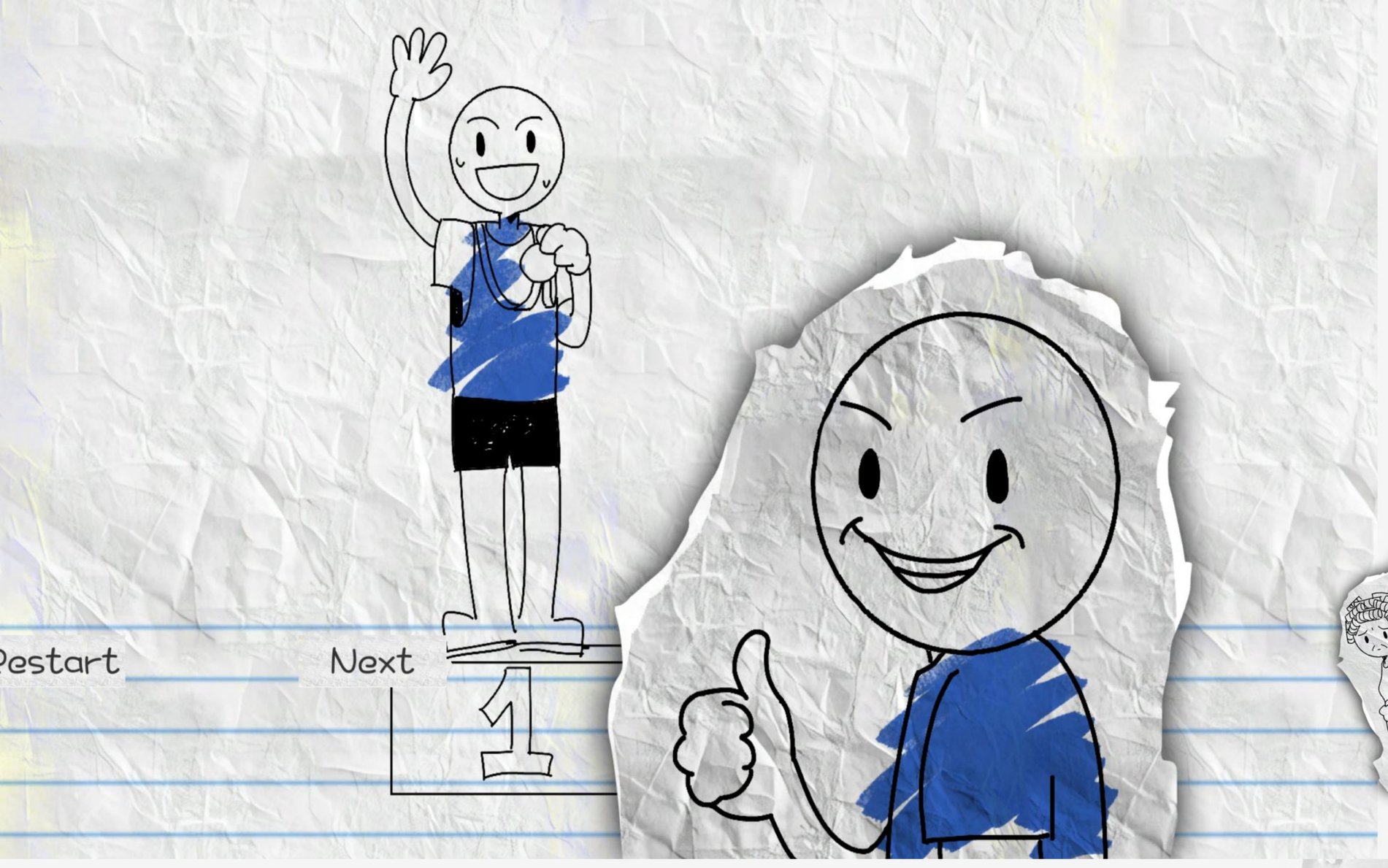
**Yellow Wasteland**
(Easy, with few small and weak monsters)



**Eroded City**
(Medium, with some big and strong monsters)

**Monster Birthplace**
(Hard, with iinfinite various monsters)

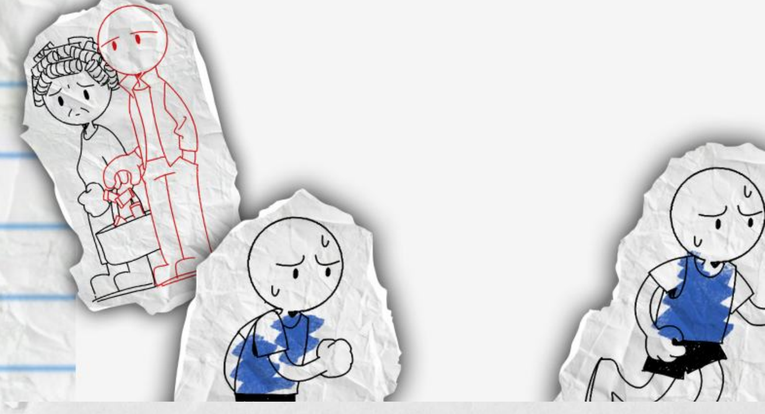
## Narrative Computer Graphics

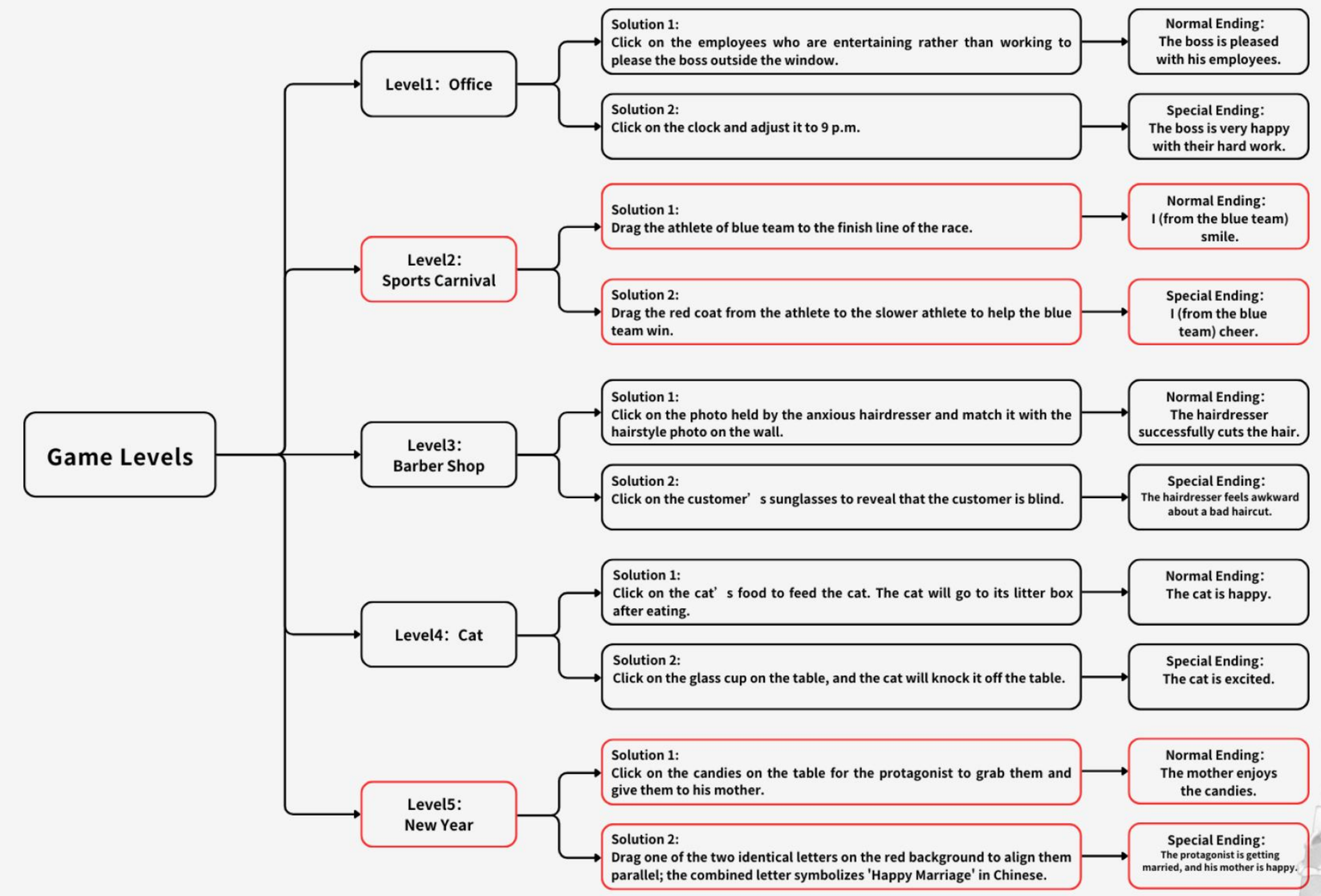



# Why Don't You Laugh

"Why Don't You Laugh" is a 2D puzzle game controlled by mouse that blends whimsy and intrigue to create a uniquely captivating experience and amuse the players. Set in a world that mirrors everyday life with a playful twist, this game invites players to navigate imaginative levels filled with hidden details, quirky interactions, and clever puzzles. Players will face distinct challenges and can achieve both normal and special outcomes based on how they solve each puzzle.

**Game Design & Game Art: Olivia Iris Tian**






## Art

## Screenshot