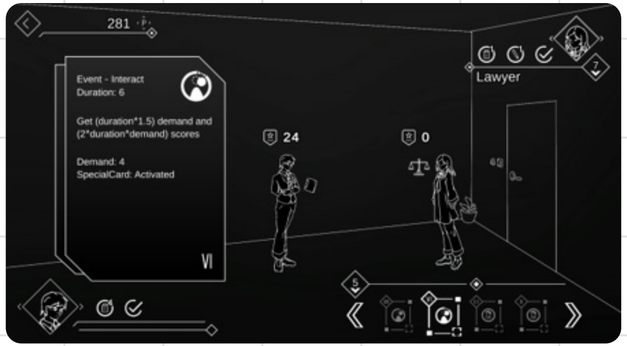
Life At Work

Summary

▷ LIFE AT WORK is a turn-based card game with roguelike elements. The target of the game is to defeat the opponent by using different EVENT CARDS that have different DURATIONS to earn more SCORES than the enemy.



#Roguelike #Turn-based #Card Game #2D

Made With

Procreate Unity Visual Studio

IDEA

After playing some Roguelike games, I found them really appealing and interesting. Players must be careful taking every step in the game (or they’ ll lose), but they can deal a great amount of damage to enemies if operated properly after a while. Therefore, I decided to design a game with Roguelike elements.

I didn’ t play turn-based games or card games a lot, but I think it would be a good idea to challenge myself and design a turn-based card game.

Character Design

Accountant

Lawyer

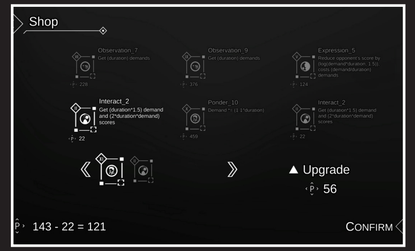
Programmer

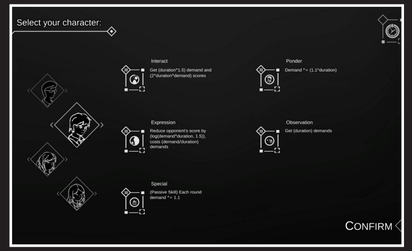
Salesperson

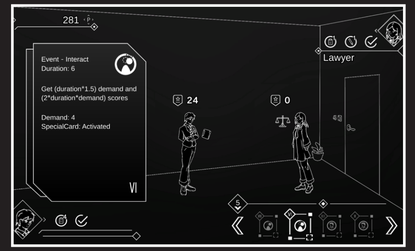
Character Passive Effect Icons

Comment Logic Demand SpecialSkill

Iteration - UI







Final UI

UI Design

Button to switch to full screen

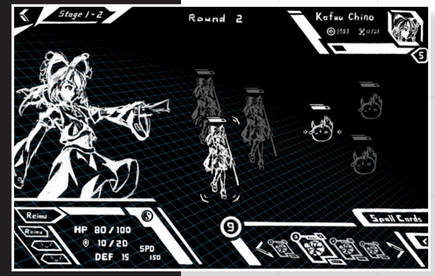
A box showing the number of cards remaining

Buttons

Character Score Icons

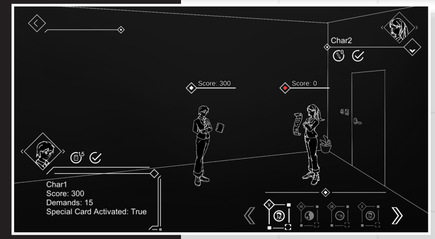
Currency icons

Left/Right



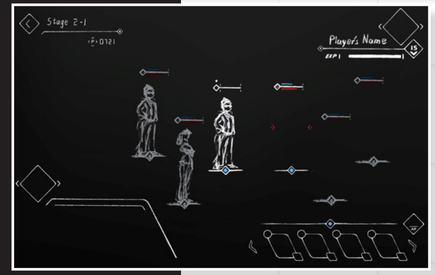
Draft: UI_V1

Reason for Abandoned:Doesn’ t fit the theme of Work (Too Energetic)



Draft: UI_V2

Reason for Abandoned: No enough place for Card Details.



Draft: UI_V3

Reason for Abandoned:Unattractive, Too Simple

Background box for pop-up tips

Mechanism – Part 1

- Before each battle, players can choose a character from the 4 occupations listed: Programmer, Salesperson, Accountant, Lawyer.

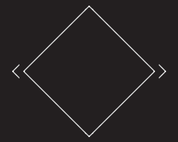
  The EVENT CARDS owned by a player have five different types: Interaction, Observation, Ponder, Expression, Special. Special card is given by the system automatically when starting a game, and a player can have only one special card at a time. Each card have a Duration from 1 to 12 hours. By selecting character of different occupations, the effect of a card differs.

  Each card can only be used once in one battle, and one player can only use one card every round. When both sides use all of their cards, the side with more scores wins.

UI Design

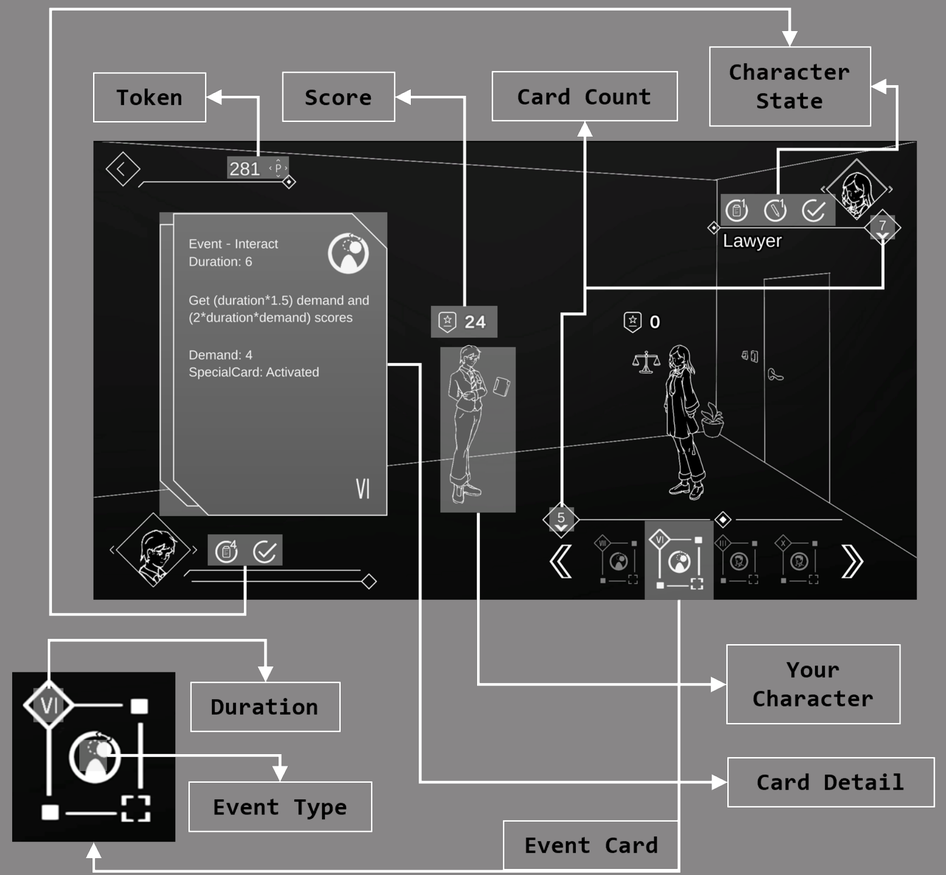
Background frame showing used cards

Character avatar frame



Score Icons

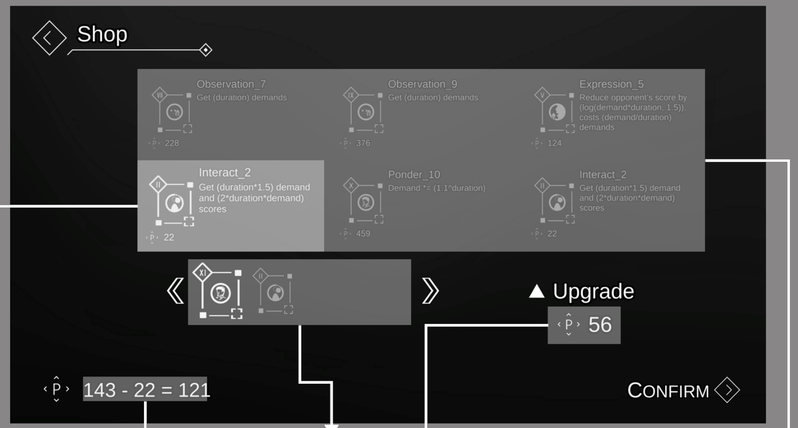
Illustration for Battle Scene



Mechanism – Part 2

▷ After each battle with the enemy, players would get a certain amount of tokens. Tokens can be used to upgrade or purchase new EVENT CARDS. Players would also have a chance to change their character’ s occupation again. The game have only Endless Mode currently.

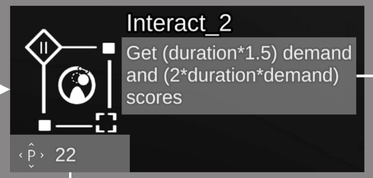
▽ Illustration for Shop Scene



Token Info

Player’s Card List

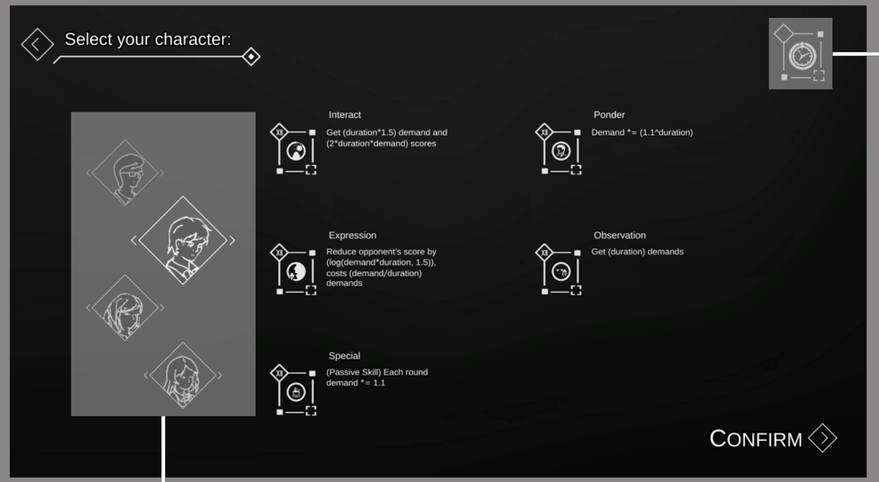
Cards in the Shop



Card Effect

Price

▽ Illustration for Character Selection Scene



Check Current Cards

Select a character

```
IEnumerator SetLocation(Vector2 original, Vector2 newPos, float time, bool hideAfterAni) {
    SetLocation(original);
    float elapsedTime = 0;
    while (elapsedTime < time) {
        float t = elapsedTime / time;
        t = (float)(-Math.Cos(t * Math.PI) + 1) / 2;
        Vector2 currentPosition = Vector2.Lerp(this.GetLocation(), newPos, t);
        SetLocation(currentPosition);
        elapsedTime += Time.deltaTime;
        yield return new WaitForEndOfFrame();
    }
    SetLocation(newPos);
    if (hideAfterAni) SetActive(false);
    locationC = null;
}
```

```
private Coroutine locationC, colorC, scaleC;
public void SetLocationAni(Vector3 orig, Vector3 v, float t, bool hide = false) {
    if (active) {
        if (locationC != null)
            StopCoroutine(locationC);
        locationC = StartCoroutine(SetLocation(orig, v, t, hide));
    }
    else SetLocation(v);
}
```

Using IEnumerator for Animations.

//Uses "Lerp" and "Cos" function to create a slow-in and slow-out effect.
//This IEnumerator is in the super class "LWElements" for most classes in this project.
//LWElements inherits MonoBehaviour.

Saves the coroutine of each attribute's animation in a private variable, so when an animation is called when the former animation is not completed, the former one can be stopped.

Uses a format similar to XML to save games.

// "GetTag(string str, string tag)" is a method that returns a string surrounded by a certain tag in the first string parameter.
//Int(string str) converts a string into int.

//There are also methods not shown in this page like "SaveCard", "LoadCharacter", "LoadPLayer" etc.

```
private static string AddTag(object content, string tag) {
    return "<" + tag + ">" + content + "</" + tag + ">";
}
1 个引用
public static string SaveCharacter(LWCharacter c) {
    string str = "";
    str += AddTag((int)c.occupation, "occupation");
    str = AddTag(str, c.Name);
    return str;
}
public static LWEventCard LoadCard(int id) {
    string str = GetTag(fileContent, "CardId_" + id.ToString());
    int length = Int(GetTag(str, "duration"));
    int type = Int(GetTag(str, "type"));
    string owner = GetTag(str, "owner");
    LWEventCard c = LWEventCard.Create(type, owner, length);
    c.id = id;
    return c;
```

Coding

- Animations/save game

//All codes are based on C#.

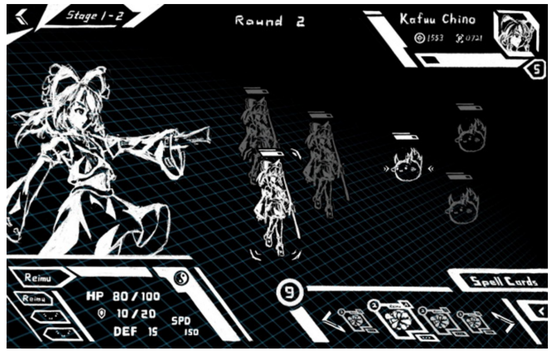
```
public static string ReadFile(string fileName) {
    try {
        StreamReader sr = new(Application.persistentDataPath + "/" + fileName);
        string s = sr.ReadToEnd();
        sr.Close();
        return s;
    } catch (Exception e) {
        Debug.Log("Exception: " + e.Message);
        return "";
    }
}
3 个引用
public static void WriteInFile(string fileName, string str) {
    if (!File.Exists(Application.persistentDataPath + "/" + fileName)) {
        File.Create(Application.persistentDataPath + "/" + fileName).Dispose();
    }
    StreamWriter sw = new(Application.persistentDataPath + "/" + fileName);
    sw.Write(str);
    sw.Close();
}
```

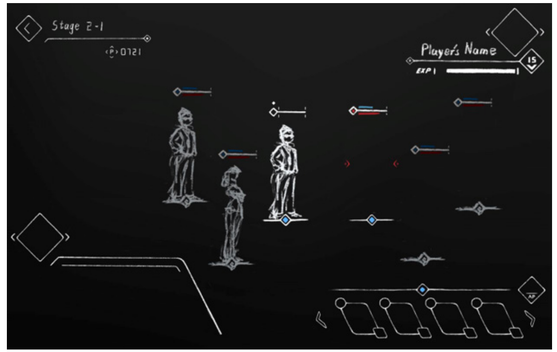
Uses StreamReader and StreamWriter class to read & write files.

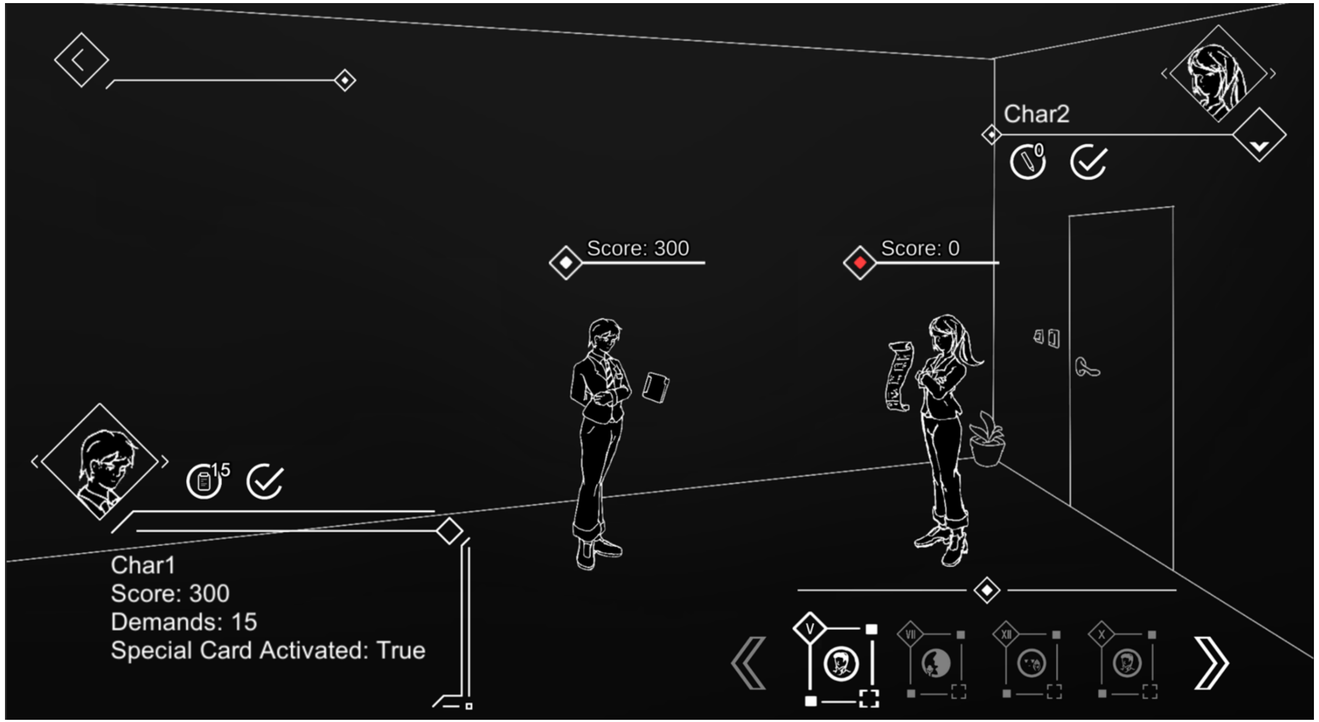
Iteration - Mechanism

▷ Version 1 -2

These two versions are too similar to other turn-based card games, and wasn’ t attracting at all.
Besides, in these versions, characters attack each others, which doesn’ t fit the theme “working” as well.







The final version included the “score” system and the “occupation” system, making the game more innovative and attracting.