The Silver Key

Rogue-like Deck-building Game
Cthulhu Style, 16+
Made by Kaiqin Kong

All of art resources are from the network or generated by AI painter - Midjourney

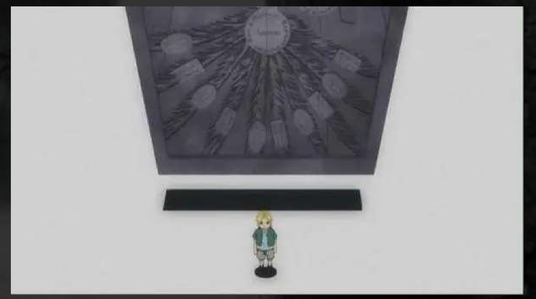
Beginning

Recently I was playing and addicted to "**Inscryption**". So when it came to designing a game with Cthulhu theme, I soon got an idea of a game sitting in a tiny and dark room, playing cards and TRPG with a Cthulhu God.

Inscryption

Therefore, I have searched for information about many Cthulhu God, and finally I decided to choose Yog-Sothoth, the God who knows everything. It is a very classic idea to paying something and gain knowledge from the Gate of Truth, so you can imagine, our hero wants to become stronger so he looks for the Wisdom God to gain power and knowledge. This leads to the following design about the story behind the game.

Dark Duel in Yu-Gi-Oh



The Gate of Truth in Fullmetal Alchemist

Fight against seemingly invincible enemies

Story Design

The Silver Key, a card game popular all over the world, and countless people flocked to it. Players call themselves duelers and compare playing cards to dueling. They gamble on their dignity, property and even life by playing cards.

You, a mad dueler decided to sacrifice everything, summoning the Outer God of Wisdom – Yog-Sothoth to obtain stronger power. Yog-Sothoth knows everything about the past, the present and the future. It gradually leads you to the door to the ultimate abyss, knowing the essence of cards and the nature of the world by showing you stories at different times.

Will you lose the duel and lose your life? Or fall madness in the whispers of the outer god? Or defeat all the enemies, and finally gain knowledge and power to become the strongest dueler? It all depends on you.

Card Design

Firstly, I have found many different card design in some famous game, like Hearthstone, Shadowverse and Yu-Gi-Oh.

Then I came up with two good designs, one is to use the style just like Hearthstone, the other is to use the hand-drawn style of Inscryption. The latter is pretty cool, but my AI doesn't seem very good at generating pictures of this style, so I chose the first style.

The Hand-drawn style card in Inscryption

Hand-drawn style card made by Midjourney

Cost
As summon card consumes **sanity**, I made the icon look like a brain.

Backgroung of card name is made by bloodstain. Strengthening the terror atmosphere.

Health
Many games use blood drop, a cross or a shield as health icon. I choose to use heart as health icon, as it is also shows that the card is “alive”.

Attack
I refer to some card games, and find that their attack icons are usually made of weapons.

I made the attack icon in the form of a **claw**. On the one hand, claws are animals’ offensive weapons. On the other hand, this is consistent with my design that the card is “alive”.

I also have tried to use weapons, such as sickle, or animal saber tooth as attack icon, but they are too thin to fit.

There are 3 levels of cards: Normal, Weird, and Unspeakable.
Normal stands for the most common animal.
Weird stands for creatures in Cthulhu Mythology, such as Rat-Thing.
Unspeakable stands for avatar or creation of Gods, such Nightgaunt.

Accordingly, I also designed three kinds of cards.

Normal

Weird

Unspeakable

Around the creature in the card are branches of Yggdrasilare. Because of the power of the Outer God, they became disgusting tentacles, and draining the life of the creature. The idea of Yggdrasilare comes from the story that Odin gained wisdom after hanging upside down from Yggdrasilare.

The frame of Normal card illustration is a style wrapped in many tentacles. These tentacles, which are changed from the branches of Yggdrasilare, are draining cards' life to Yggdrasil.

When the card level is Weird, they have the power to fight against tentacles.

When the card level is Unspeakable, their power is so strong that they can drain power drain life of Yggdrasil conversely, which makes tentacles become the original look of wizened branches.

Frame is made of Cutout & Liqufy in Photoshop

illustration Design

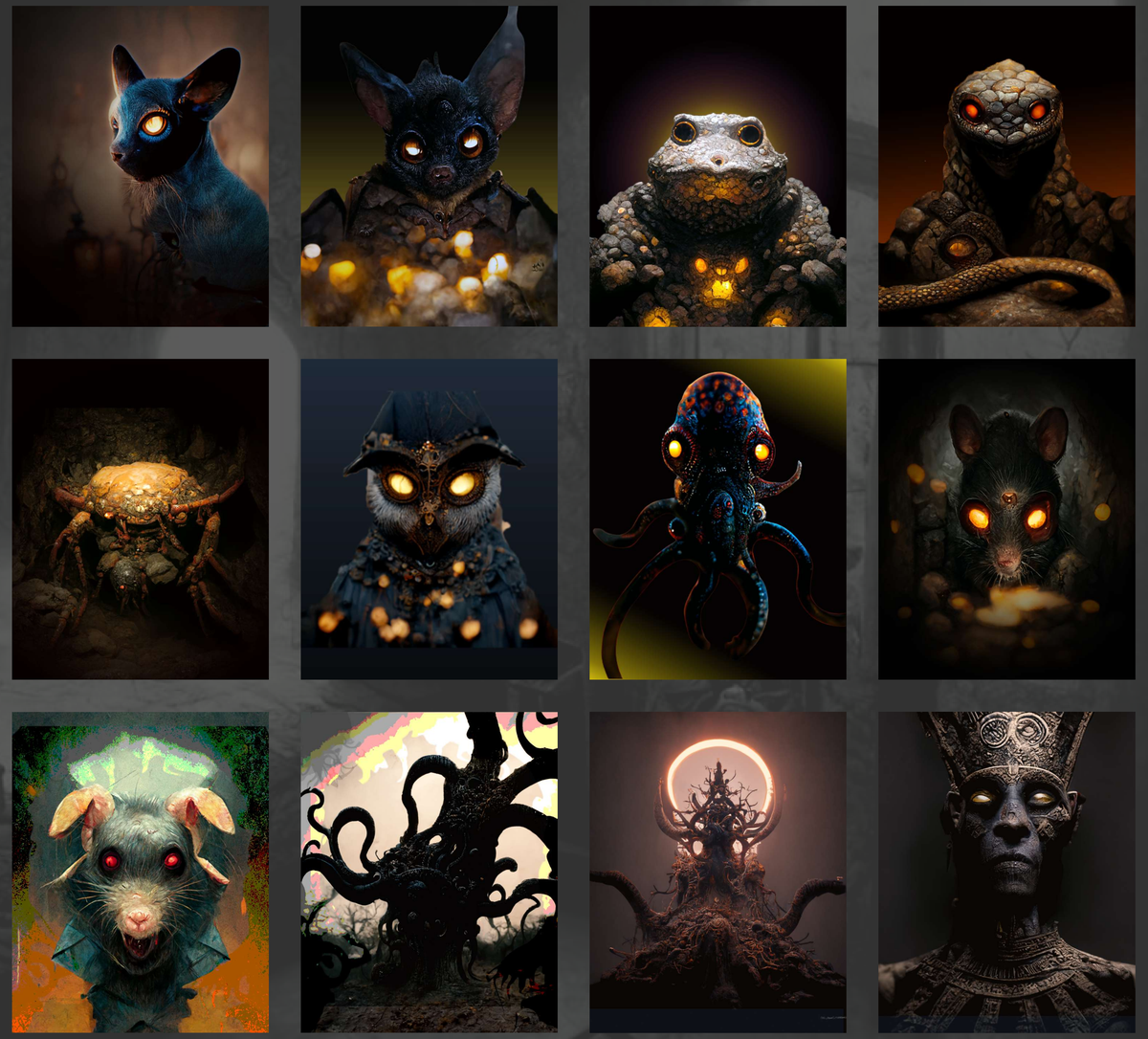
First edition

They look good, but ... do not fit with my card design very well, so I have to give them up.

I think this part is pretty simple, but it is also difficult.

I know it sounds contradictory, but the truth is I can use AI to make an enormous number of amazing pictures in a short time, and I like them all.

However, those amazing pictures may not fit with my card, so I could only try them one by one, abandoning the inappropriate ones with pain.



I have made 8 normal cards, 2 weird cards and 2 unspeakable cards.

Battle Scene Design

First edition is very simple, I was going to make it as a canvas on a 3d table. But in the end, I decided to make full 2d, because it was unnecessary.

So I made this battle scene, which has a top view of the card table. It looks good I think.

You may notice that I put things about player information into a separate board, because they look strange from top view, and player won't have any interaction with them in game.

Health

There is also a red picture behind it. Its transparency and the size of Heart will change while health changes, simulating the "heart beat".

When you are going to summon a card, available grids will "open its mouth".

Grid

Summon UI

Rule Book

Turn-End Bell

Deck

Relics

Sanity

Bone

Props

Hand Cards

Gameplay

Player have Health, Sanity and Bone in each fight.

Health is shared by both sides of Battle Scene. When one side has 15 more HP than the other, the game ends.

The Sanity controls the number of cards that can be played in the current turn. The sanity can't be less than 0.

The Bone will plus 1 when your card is destroyed, and you can use it to summon some special cards.

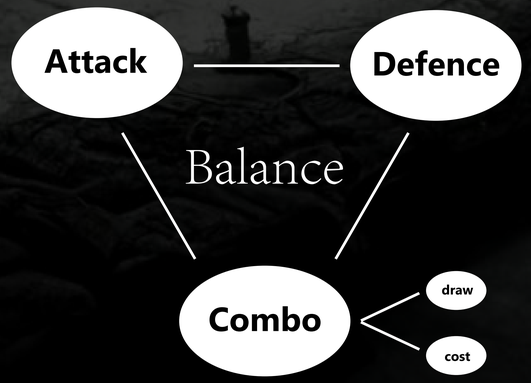
Cards have attack, health, cost, and effect.
At the end of the turn, the card will automatically attack the card in front of it, causing damage equal to the attack of this card to the health of the card in front. If the damage exceeds the health of the card in front, or there is no card in front, it will do damage to the player.

Card will only attack the card in front of it.

Health Point is shared in the game, so when it larger than 15 points or less than -15 points, the game is over.

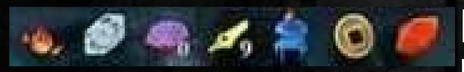
Sacrifice
Player can choose to sacrifice their card in block, which will increase 1 sanity for players and destroy this card, it will also increase your bones of course. This will diversify the players' decisions.

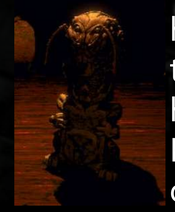
refer to Slay the Spire



At first, I tried to let my game emphasize combos. Silent in Slay the Spire and Rogue in Hearthstone are good samples, I tried to design cards and decks based on them. You know, low-cost, strong ability to draw cards as well as magnificent combos.

Maybe I have to design hundreds of cards in final... A big work!





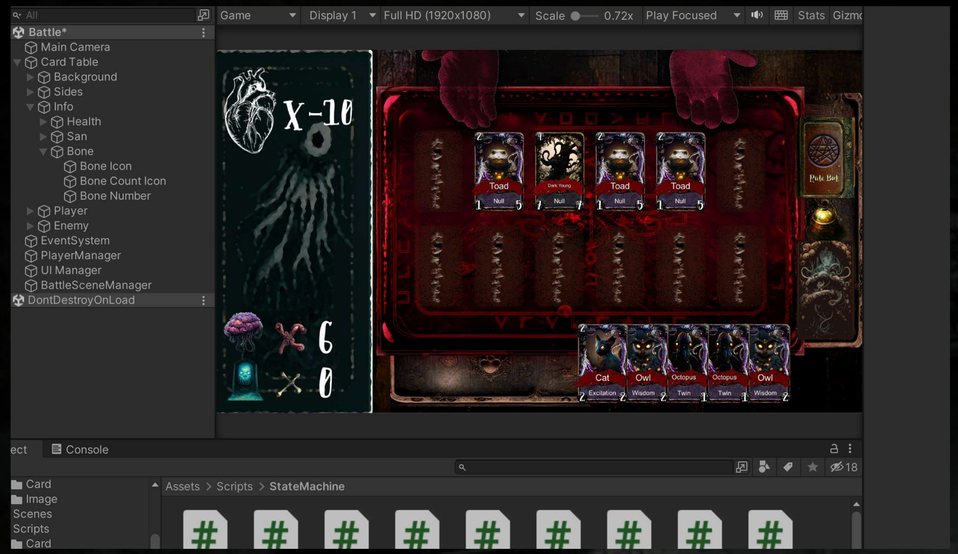
Relics System

Many Rogue-like games have such a system, the effect of relics will throughout the game. Even in Inscryption we have wood carvings.
Relics can significantly enhance the strength of the player, and make our game fun.
Different relics will affect the player's strategy of building their deck in this game.

Prop System

Props will affect the game instantly.

Development



I have experienced several code refactorings, which was painful.

I don't have much experience of making a big project, or building a huge system, and I also didn't write documents before writing code!!! I have done some projects but mostly on Graphics(a Ray-Tracer and a co-op project on OpenGL). For those reasons, I don't know how to manage my code, and I paid for it in this project.

```
public class BattleSceneFSM : MonoSingleton<BattleSceneFSM>
{
    public  Parameter   parameter;
    public  MyState     current_state;                          // 当前状态，只能有一个状态
    public  StateType   current_type;
    private Dictionary<StateType, MyState>
                        states = new Dictionary<StateType, MyState>();

    #region FSM
    void Start() {
        states.Add(StateType.GameStart,     new GameStart(this));

        states.Add(StateType.PlayerDraw,    new PlayerDraw(this));
        states.Add(StateType.PlayerAction,  new PlayerAction(this));
        states.Add(StateType.PlayerAttack,  new PlayerAttack(this));

        states.Add(StateType.EnemyDraw,     new EnemyDraw(this));
        states.Add(StateType.EnemyAction,   new EnemyAction(this));
        states.Add(StateType.EnemyAttack,   new EnemyAttack(this));

        TransitionState(StateType.GameStart);                   // 游戏开始
    }

    void Update() {
        current_state.OnUpdate();                               // 调用当前状态的update
    }

    // Change current state
    public void TransitionState(StateType type) {
        if(current_state != null)                               // 退出当前状态
        {
            current_state.OnExit();
        }

        current_type = type;                                    // 进入下一个状态
        current_state = states[type];
        current_state.OnEnter();
    }
    #endregion
```

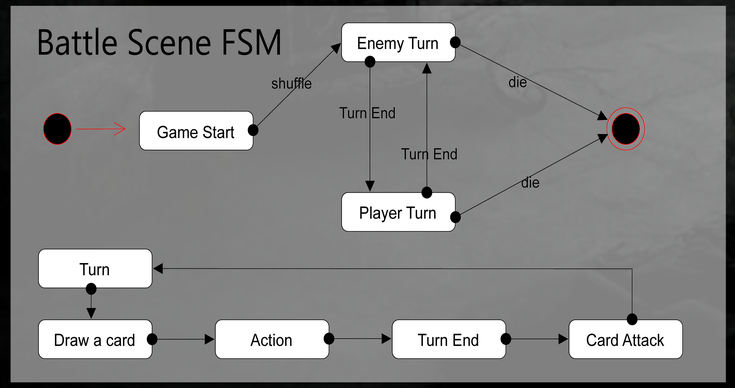
State-Machine Code and Some Basic Functions

```
public interface MyState
{
    void OnEnter();

    void OnUpdate();

    void OnExit();
}
```

The current code logic is relatively clear. but the FSM class is too large and the control parameter is too long. I will continue to optimize code.




```
public void LoadEffect(CardPlayer _player, BattleSceneFSM _manager, Parameter _parameter)
{
    card = this.GetComponent<CardDisplay>().card;

    string  raw_name    = card.effect;
    // var     effect_name = raw_name.Substring(0, raw_name.Length - 1);
    var     effect_name = raw_name.Substring(0, raw_name.Length);
    Type    type        = Type.GetType(effect_name);

    if(type != null)
    {
        effect = (Effect)Activator.CreateInstance(type);
    }
    else
    {
        effect = new EmptyEffect();
    }

    effect.OnTrigger(_player, _manager, this, _parameter);

    if(_parameter.twin_count == 0 || type == Type.GetType("Twin"))
    {
        summon_event.Invoke();
    }
    else
    {
        for(int i = 0; i < 2 * _parameter.twin_count; ++i)
        {
            summon_event.Invoke();
        }
        _parameter.twin_count = 0;
    }
}

public void OnDestroy()
{

}
```

Code of Card Effect Logic

```
public void Display()
{
    this.text_name.text         = card.name;
    this.text_attack.text       = card.attack.ToString();
    this.text_health.text       = card.health.ToString();
    this.text_cost.text         = card.cost.ToString();

    rarity                      = card.rarity;

    this.text_effect.text       = card.effect;
    this.text_descriptin.text   = card.description;

    LoadCardFace();
}

public void UpdateState()
{
    this.text_attack.text   = card.attack.ToString();
    this.text_health.text   = card.health.ToString();
    this.text_effect.text   = card.effect;
}

public void Hit()
{
    this.text_health.text   = card.health.ToString();
}

public void LoadCardFace()
{
    Texture2D texture = Resources.Load<Texture2D>("Card/" + card.name);
    this.card_face.sprite = Sprite.Create(texture,
        new Rect(0, 0, texture.width, texture.height), new Vector2(0.5f, 0.5f));
}
```

Card Display Code

```
public class MonoSingleton<T> : MonoBehaviour where T : MonoBehaviour
{
    private static T instance;
    public  static T Instance
    {
        get
        {
            if(instance == null)
            {
                instance = FindObjectOfType<T>();
            }
            return instance;
        }
    }

    private void Awake()
    {
        if(instance != null)
        {
            Destroy(gameObject);
        }
    }
}
```

MonoSingleton

https://youtu.be/QSqaVKCIZJE