KILL KISS

Production Date: July 10, 2026 – Present

Team Members:

wren — Technical Artist

Leyao Ni — UI, Narrative & Game Designer

Chenyang Xu (Me) — Programmer

Sheng Liu — Artist

Lin Shui — Game Designer & Project Manager



KILLKISS

Coffeehouse

Game Summary:

KILLKISS is a game that combines coffee shop management with strategic deck-building card combat. By day, players manage a mysterious café and craft a variety of coffee drinks for customers. By night, they become a demon slayer, exploring the city, hunting monsters, and collecting materials. Coffee serves as the bridge between the two gameplay systems, creating a seamless gameplay loop that blends relaxing management with exciting, strategic combat.

#CoffeeShopSimulation
#DeckBuilding
#Narrative

Video Link: https://youtu.be/LDg3MVn0F14

Made With

KILLKISS Coffeehouse

Inspiration

Core Concept: **Dual Identity Cycle (Day / Night)**
The game draws inspiration from the film American Psycho, particularly its portrayal of a double life. I adapted this contrast into a day–night gameplay cycle: the protagonist runs a café by day and becomes a demon hunter after dark.



Card Design





We drew inspiration from the card designs of Yu-Gi-Oh!, Hearthstone, and Pokémon. Following Pokémon's approach, we created three card types that **share a consistent layout but use distinct color palettes and visual details for easy recognition.**

References



Slay the Spire 2 – draw–discard cycle, card exhaustion, reshuffling mechanics, status effects, and debuffs.

Hearthstone – deck-building mechanics: custom deck construction, card limits, and card synergies for strategic play.





VA-11 Hall-A: Cyberpunk Bartender Action – revealing the game's world and narrative through management gameplay.

KILL KISS

Combat

- Show the enemy's planned attacks for the next turn.
- Drag the magic hand onto a card to play a Magic Card or activate an Action Card's hidden ability.
- Show player and enemy HP and Poise.
- HP depleted: battle ends.
- Poise depleted: stunned for one turn.
- Show or hide the settings menu.
- Use the menu to quit the game.



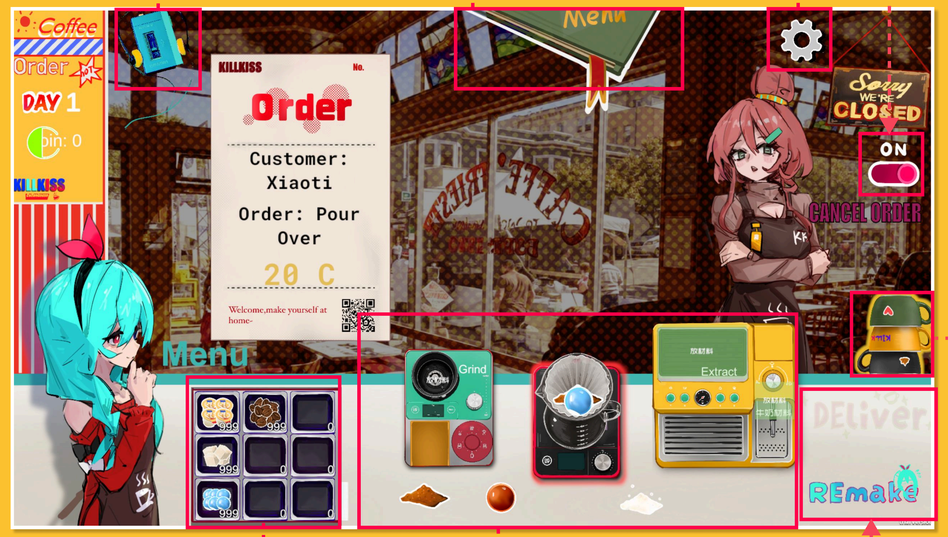
- Use coffee to deal damage, restore health, or apply buffs.
- End your turn.
- Show cards left in the deck.
- Discard pile count.
- Show the resources available this turn.
- Playing Action Cards costs AP; 3 AP are restored each turn.
- Playing Magic Cards costs MP; 1 MP is restored each turn.
- Click a card or drag it onto an enemy to play it.
- Action Cards: discarded after use.
- Magic Cards: removed until the battle ends.

Coffee Making

- Show the coffee-brewing guide.



- Play BGM
- Show or hide the settings menu.
- Use the menu to quit the game.

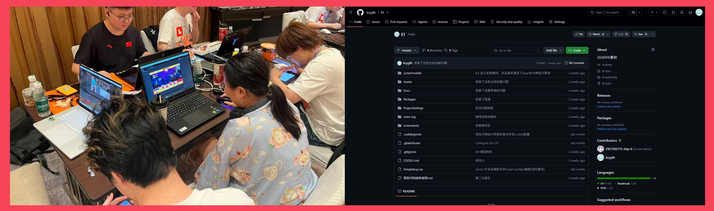


- Ingredients selected when opening the café can be dragged to the workspace on the right for processing.
- Remake or submit the order.
- Pick up a cup
- Drag ingredients to the corresponding coffee machine.
- Click the machine to process them.

Gameplay Systems

Division

I was responsible for the combat system and shared technical infrastructure, while my teammate wren implemented the café gameplay systems.



Our five-person team covered project management, game design, art, technical art, and programming. I focused on the combat system and shared technical infrastructure, then handled integration and debugging in the final development stage. We used GitHub to coordinate development and track code changes.

Division of Labor

AI-ASSISTED ENGINEERING



I used AI as an engineering copilot for design exploration, implementation, debugging and documentation. Architecture decisions, code review, integration and final verification remained my responsibility.

1. CONTEXT
Provide requirements, existing code and technical constraints.
2. DESIGN
Compare solutions and define responsibilities, data flow and edge cases.
3. IMPLEMENT
Generate focused code changes, tests and debugging hypotheses.
4. VERIFY
Review every change, run tests, integrate it into Unity and validate it through playtesting.

CONTEXTUAL TUTORIAL CALLOUTS

Tutorial information is attached directly to the relevant interface element instead of being placed in a separate instruction panel. Tooltips can follow the cursor, adapt to different text lengths and remain inside the visible canvas.



Context-sensitive trigger Target attachment
Adaptive layout Screen-boundary protection

```
        /// <summary>
        /// 把提示框放到屏幕坐标附近（用于悬停跟随鼠标）。offset 为相对该坐标的偏移。
        /// 只移动位置，不重建布局——布局只在 SetMessage/OnValidate 时计算，
        /// 避免每帧 RebuildLayout + ForceUpdateCanvases 造成布局抖动（频闪）。
        /// </summary>
        public void AttachToScreenPosition(Vector2 screenPosition, Vector2 offset)
        {
            EnsureReferences();
            if (_rectTransform == null)
                return;

            if (_rectTransform.parent is not RectTransform parent)
                return;

            var canvas = GetComponentInParent<Canvas>();
            var camera = canvas != null && canvas.renderMode !=
RenderMode.ScreenSpaceOverlay
                ? canvas.worldCamera
                : null;

            if (!RectTransformUtility.ScreenPointToLocalPointInRectangle(
                    parent, screenPosition, camera, out var localPoint))
                return;

            _rectTransform.anchoredPosition = localPoint + offset;
            ClampToParent();
        }
```

OpenAI

System Architecture

Engineering Architecture

STORAGE SYSTEM ARCHITECTURE

Game Repository Pattern & Separation of Concerns

Gameplay Systems & UI Events

JSON Data Configurations

RuntimeGameRepository.cs
DYNAMIC STORAGE (VOLATILE/USER STATE)

Tracked State Models:

- Selected Card IDs
- Selected Deck IDs
- Owned Card Counts
- Coins & Materials

ID Index / Bridge

Query ID › Fetch

StaticGameRepository.cs
STATIC STORAGE (FIXED DEFINITIONS)

Tracked State Models:

- Card Definitions
- Player Card Metadata
- Card Import Structure
- Card Categories

Lifecycle & Data Integration Flow

Step 1	Step 2	Step 3	Step 4
1	2	3	4
Load Static Data	Record Selection	Resolve Details	Validate

H-Type Battle Manager

H-Type Battle Manager & Unified Effect Generator

Player Side

Player Controller & State

Hand / State Context

- HP
- Attributes
- Buffs & Debuffs
- AP & MP

Player Turn Loop

- Capability Check
- Action Validation
- Card Draw
- Resource Costs

Effect Resolution Core

Battle Flow & Resolution Core

- 01. Flow Control
- 02. Phase Checks
- 03. Action Validation
- 04. Effect Resolution
- 05. Event Dispatch
- 06. Victory Check

Enemy Controller Side

AI Behavior & Intent

Enemy State Context

- HP
- Resistances
- Status Effects
- Passive Traits

Enemy AI & Intent

- Card Selection
- Target Selection
- Turn Scheduling
- Intent Submission

Data Flow & Lifecycle Timeline

Player Input / AI Intent → Battle Manager Registry → Effect Generator Resolution → State & Attribute Mutation → Victory/ Defeat Evaluation → UI Dispatch & Animation FX

Code It Juicy

JUICE

LAYERED IMPACT FEEDBACK



```
                _image.color = flashWhite;

                // 瞬间放大（Squash）
                _rect.localScale = _originScale * hitScalePunch;

                // 击退
                _seq.Append(_rect.DOLocalMoveX(_originPos.x + knockbackDistance,
knockbackDuration)
                    .SetEase(Ease.OutQuad));

                // 震动
                _seq.Join(_rect.DOShakePosition(shakeVibrato * 0.01f, shakeStrength,
shakeVibrato, 90f, false)
                    .SetEase(Ease.InOutQuad));

                // 缩放弹回（Stretch）
                _seq.Join(_rect.DOScale(_originScale,
hitScaleDuration).SetEase(Ease.OutBack));

                // 闪白+闪红+恢复
                if (_image != null)
                {
                    _seq.Join(DOTween.To(() => _image.color, c => _image.color = c, hitColor,
flashDuration * 0.3f)
                        .OnComplete(() =>
                        {
                            if (_image != null)
                                DOTween.To(() => _image.color, c => _image.color = c,
_originColor, flashDuration);
                        }));
                }

                // 返回原位
                _seq.Append(_rect.DOLocalMoveX(_originPos.x, returnDuration)
```

Hit feedback combines movement, scaling, screen shake, color flashes, and number animations to make combat outcomes immediately clear.

The code on the left implements most combat effects.

ANIMATED CARD DEALING & HAND REFLOW

Players need to understand:
Where are cards drawn from?
Where do new cards enter the hand?
How does the hand rearrange after a card is played?
Where does a card go if a drag action fails?



When players use the Smoke Bomb card, new cards are added without disrupting the hand layout.



Cards are drawn from the bottom-left corner and fly to the bottom-right corner when discarded.

CLEAR INVALID-ACTION FEEDBACK

When an action cannot be performed, the game explains the reason immediately instead of silently ignoring the input.



You can choose up to two common cards



Activate the Magic Card first.



Your deck is full. The maximum deck size is 15 cards.



No AP remaining.

RESPONSIVE CARD INTERACTION

Hover, selection and card-category feedback help players understand which card is being inspected and whether it can be interacted with.



NORMAL

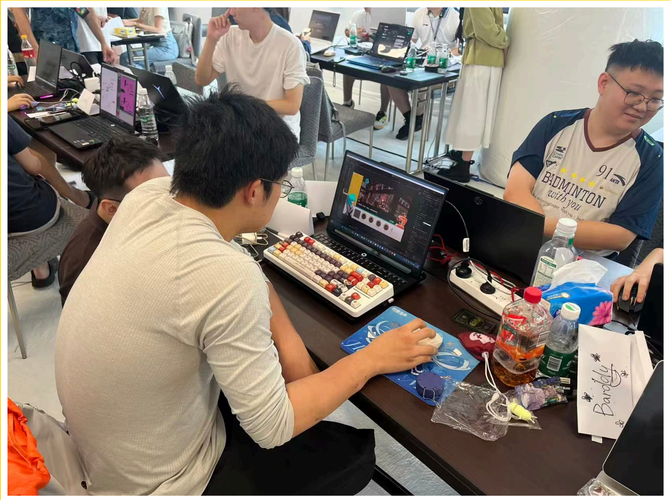


HOVERED / SELECTED



Cards lift and tilt slightly when hovered over. An outline highlights the selected card, while different card types use distinct highlight colors. Clicking and playing cards also trigger animations.

Playtesting





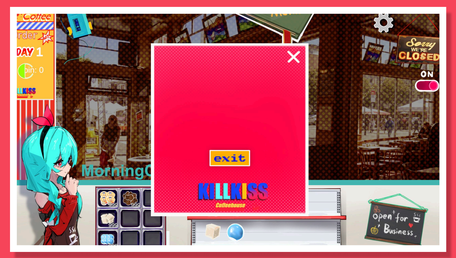


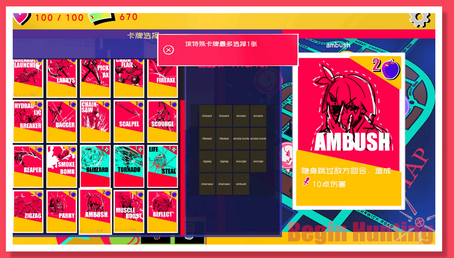


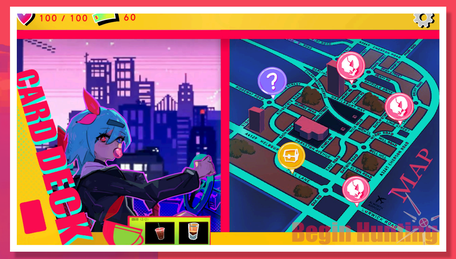
Outcome

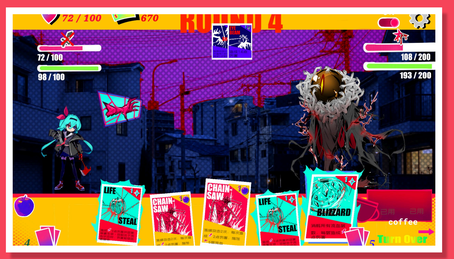












SHOP

FIGHT