

Voices From The Corner



Voices from the Corner is an AI-driven narrative investigation game where players conduct open-ended interviews with characters who generate dynamic responses in real time. Players gather evidence and turn each investigation into an article shaped by what they choose to ask and uncover.

PROJECT OVERVIEW

ROLE

Solo Developer

ENGINE

Unity

PLATFORMS

Windows / macOS

DEVELOPMENT PERIOD

May 2026 - Present

LANGUAGES

English / Simplified Chinese



CREATIVE INSPIRATION

Inspired by my experience rescuing stray cats and imagining their side of the story.

My interest in this project grew from my experience rescuing stray cats and documenting the stories I encountered along the way. As I read more rescue stories, I began to wonder how the same events might be understood differently from the cats' perspective.

To explore this idea, I collected material from rescue posts and interviews with rescuers, focusing on recurring experiences, conflicts, and emotional details. I then translated these observations into the game's characters, narrative scenarios, and AI-driven conversations.

PRE-PRODUCTION



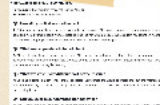
01 Social Media Research

Explore real rescue stories and discussion posts.



02 Field Observation

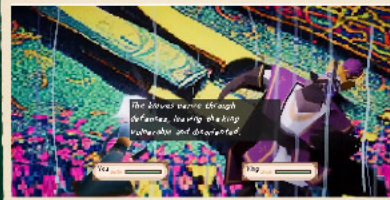
My own feeding and observation of stray cats.



03 Rescuer Interviews

Interview records about rescuers' perspectives and experiences

GAME REFERENCES & DESIGN INSIGHTS



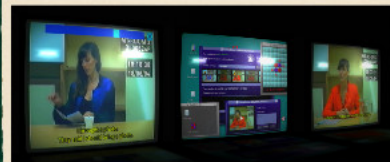
1001 Nights | Turning Language into Gameplay

1001 Nights turns weapons mentioned in AI dialogue into usable equipment. This inspired me to connect free-form interviews with discoveries players can use in their articles.



Storyteller | Building Narrative Through Choices

Storyteller lets players build narratives by arranging characters and scenes. This inspired my writing system, where players select and organize interview material to shape their reports.



Her Story | Following Curiosity Through Clues

Her Story lets players piece together events by searching interview footage. This inspired me to spread clues across characters, encouraging follow-up questions and comparisons between perspectives.

SYSTEM DESIGN

01 INTERVIEW STRATEGY

Questions affect both the answers and the interviewee's visible reactions.



TRUST

Willingness to share deeper memories



STRESS

Discomfort with the current topic



ATTENTION

Energy left to continue the interview

QUESTION EXAMPLES

Ask about food or familiar people

Trust ↑



Engaged

Keep probing a painful memory

Stress ↑



Uneasy

Repeat the same question

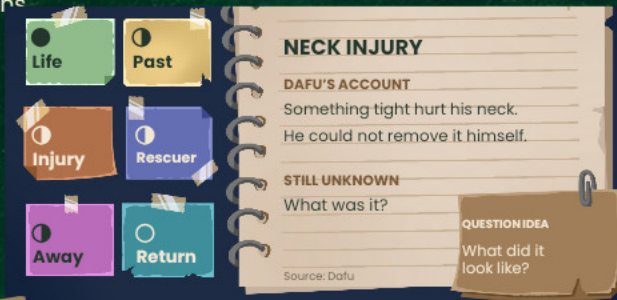
Attention ↓



Distracted

02 REPORTER'S NOTES

Review existing facts, find what's missing, and determine the next questions



Discovered Topics & Progress

Only discovered topics appear, keeping future clues hidden.

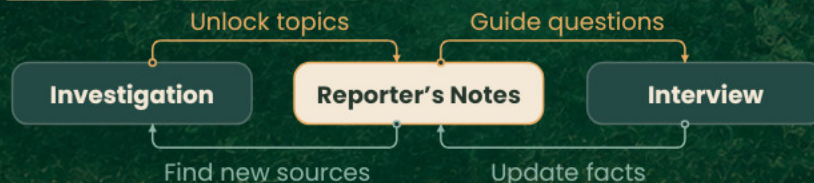
- New topic
- Open questions
- Ready to report

One Topic, Several Sources

New accounts clarify earlier clues within the same topic. For example, Dafu recalls neck pain, while Ms. Lin identifies the rope embedded in his neck.

Guidance Between Systems

For topics that are still unclear, players can edit follow-up questions or look for answers from other sources.



03 MATERIAL CARDS & WRITING

Shape the article with interview findings.



Unlock Material

Clue recorded in notes

Embedded Rope

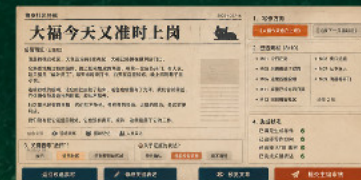
Source: Ms. Lin

Type: Fact

Clues unlock material cards labeled with their sources.



Article Assembly



Cards → Template → Article

Selected cards generate the draft, which players can edit.



Choose & Frame

16 preset cards

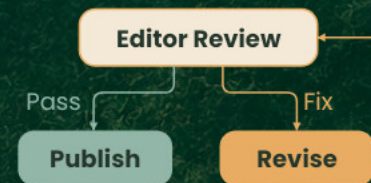


8-10 selected

Select a topic and matching material cards. Only unlocked cards are available.



Review & Revise

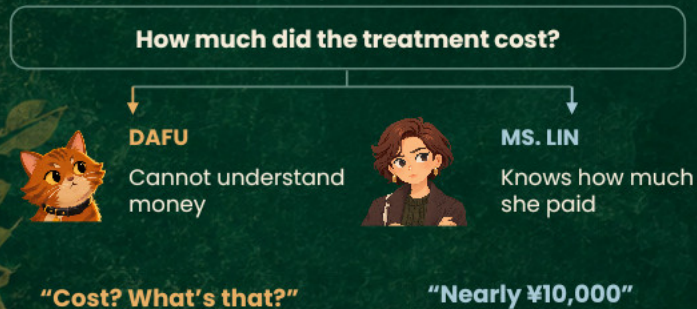


Draft is checked for missing facts or misleading wording. Player should revise before publishing.

DESIGNING BELIEVABLE VOICES

01 CHARACTER-AWARE INTERVIEWS

Dafu describes what he felt and remembers. Ms. Lin explains the rescue, treatment costs, and her own decisions. Players need both perspectives.



02 KEEP ANSWERS BELIEVABLE

EXAMPLE QUESTIONS / EXPECTED REPLIES

- "Was it a rope around your neck?" → **Dafu: "Don't know. Something tight. It hurt."**
- "What changed after the treatment?" → **Ms. Lin: "The rope was removed during surgery."**
- "What do you think of this neighborhood?" → **Dafu: "Lots of food here."**

TWO VOICES, ONE FULLER STORY



DAFU

Short, sensory, direct

What hurt your neck?

"Something tight.
I couldn't get it off."



MS. LIN

Detailed, reflective, personal

What happened to Dafu's neck?

"The vet found rope
embedded in his neck."



REPORTER'S NOTES

Dafu's neck pain

Something tight hurt Dafu's neck.

Embedded rope

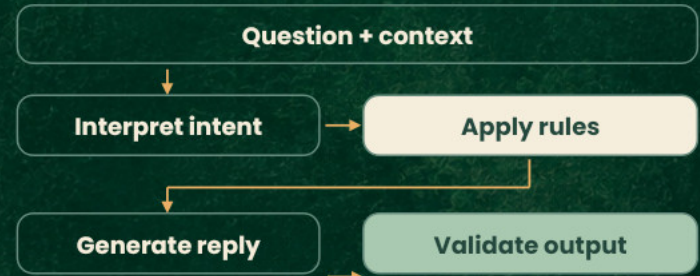
Ms. Lin brought Dafu to the vet and found rope embedded in his neck.



03 FROM QUESTIONS TO GAME EVENTS

AI interprets. Rules decide facts and states.

Context: facts, sources, state, exclusions



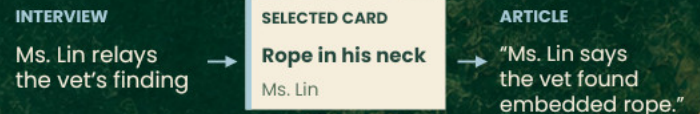
reply
Dialogue

behavior
Reactions

validated IDs
Notes + cards

04 HOW INTERVIEWS SHAPE THE ARTICLE

Selected material cards add known facts to the article and retain who provided them.



VISUAL DESIGN



Dafu

A round silhouette and relaxed posture reflect Dafu's life in the community today.



02

ENVIRONMENT DESIGN

Familiar urban spaces anchor the investigation. Feeding bowls, improvised shelters and worn surfaces reveal how people and cats share the neighborhood.



Community

A readable neighborhood layout connects everyday activity with places worth investigating.



Guard Booth

The meeting point of the guard's working day and Dafu's daily routine.



Café

Clear furniture shapes and muted colors bring the shared visual style into an interior space.



Lin's Desk

Desks, screens and evening light ground the reporter's workflow in an ordinary workplace.

01

CHARACTER DESIGN

Clothing, posture, and personal belongings hint at each character's age, work, and daily habits before they speak.



Ling

Reporter
Tousled hair, a red jacket, and expressive eyes convey a young reporter's curiosity.



Lin Min

Rescuer
An everyday coat and tote ground Lin Min in her life as a community resident and rescuer.



Security Guard

Community guard
A worn uniform, radio and relaxed stance suggest years of familiar routines at the community gate.



Shen He

Editor-in-chief
Dark tailoring and a composed posture reflect Shen He's direct manner and editorial authority.

03

VISUAL LANGUAGE

Simplified shapes, broad color-block shadows and muted hues give the world a storybook quality grounded in everyday life.

01

COLOR



Muted greens, warm ochres and softened contrasts.

02

SHAPE



Clear silhouettes, simple forms and restrained shadows.

03

TEXTURE



Fine grain over broad, layered color shapes.

04

STORIES IN THE DETAILS

Convey character and atmosphere through everyday objects and visual details.



INTERFACE DESIGN

The UI combines the warmth of a scrapbook with the familiar tools of reporting, giving each screen a personal, lived-in feel. Changing expressions bring conversations to life, while paper material cards invite players to piece their findings into a story.

01

REPORTER-INSPIRED ELEMENTS

MAIN MENU

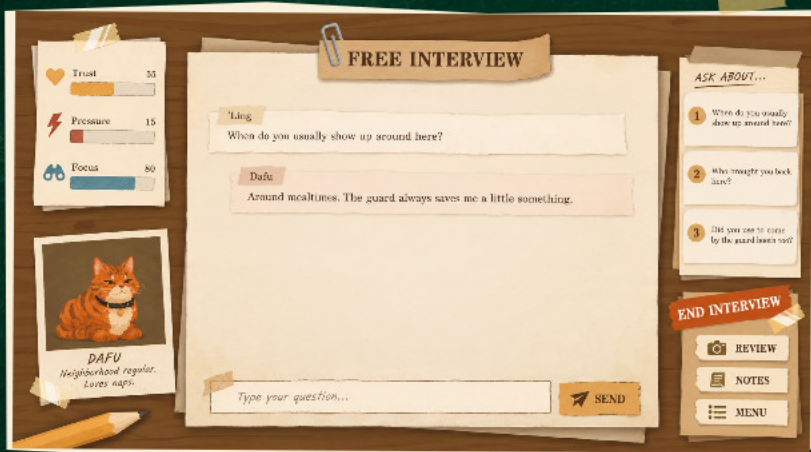


Polaroid photos, camera icons and the cat translator connect the interface to the player's role as a reporter.

02

INTERVIEW FEEDBACK

FREE INTERVIEW



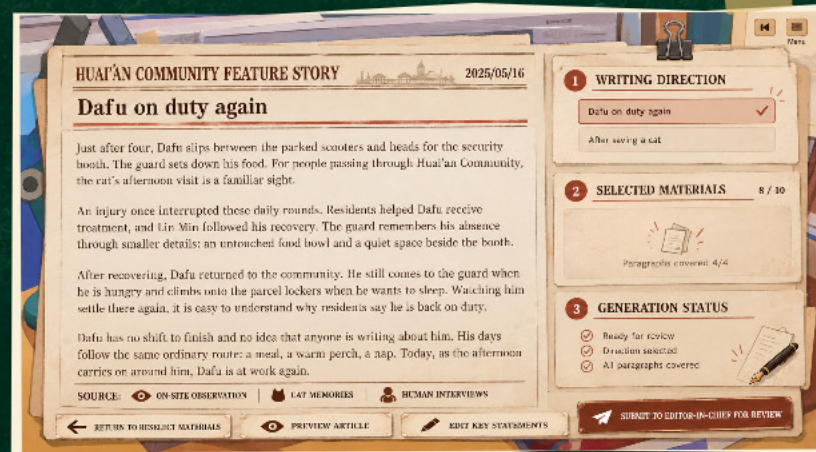
1 When do you usually show up around here?

Character portraits change in response to what the player asks. Questions affect Trust, Pressure and Focus, while suggested questions update as the conversation develops.

03

FROM MATERIAL TO ARTICLE

ARTICLE WRITING



2 SELECTED MATERIALS 8/10



MATERIAL CARDS

SELECTION COUNT

PARAGRAPH COVERAGE

Interview notes become paper cards for drafting. Selection counts and paragraph coverage help players check whether the article has enough supporting material.

CODE BEHIND THE CONVERSATION

01

IMPORTANT SYSTEM LOGIC

Rules, state, and authorial control shape believable replies.



Code-Built Interview UI

Columns and portrait masking are created in C#.

```
GameUI.Interview.cs

var col = new GameObject(
    "LeftColumn", typeof(RectTransform));
col.transform.SetParent(parent, false);
Stretch(col.GetComponent<RectTransform>(),
    new Vector2(0.020f, 0.04f),
    new Vector2(0.210f, 0.97f),
    Vector2.zero, Vector2.zero);
BuildInterviewStatusPad(col.transform);
BuildInterviewPortraitPad(col.transform);

var window = new GameObject("PhotoWindow",
    typeof(RectTransform), typeof(RectMask2D));
interviewPortraitImage.preserveAspect = true;
Anchored columns + a masked, aspect-preserving portrait.
```

Interview Runtime Flow

Session setup selects the character's rule engine.

```
InterviewController.cs

subject = who;
returnToWritingAfterEnd =
    returnToWritingAfter;
log.Clear();
gainedThisInterview.Clear();
LastExtractedMaterials =
    Array.Empty<string>();
boundaryHit = false;
lastReply = null;
lastPlayerQuestion = null;
recentDafuFoodMentions.Clear();
// ...
if (who == InterviewSubject.Dafu)
    engine = new DafuRuleEngine();
else
    engine = new LinRuleEngine();
InterviewDebugLog.SessionStart(who);
Reset session data; choose Dafu's or Lin's rule engine.
```

02

CODE SHOWCASE & ANALYSIS

From design rules to working systems.

```
LinRuleEngine.cs

var reply = engine.Process(question);
var matsBefore = SnapshotUnlockedMaterials();
foreach (var id in reply.unlockedIntel)
{
    if (GameState.Instance.GrantIntel(id))
    {
        gainedThisInterview.Add(id);
        TryCrossCheck(id);
    }
}
LastExtractedMaterials =
    DiffNewMaterials(matsBefore);
InterviewMaterialExtractor.ApplyExtraction(
    subject, reply, LastExtractedMaterials, log);

if (reply.cognitiveBoundary)
{
    boundaryHit = true;
    GameState.Instance.Data
        .dafuCognitiveBoundaryHit = true;
}

if (!deferSpeakerLines)
{
    if (subject == InterviewSubject.Dafu
        && reply.replyLines != null)
        reply.replyLines = EnforceDafuFoodQuota(
            reply.replyLines, reply, question);
    AppendSpeakerReply(reply);
}
```

GrantIntel() records new facts, then checks related clues.

Compare material states to extract newly unlocked cards.

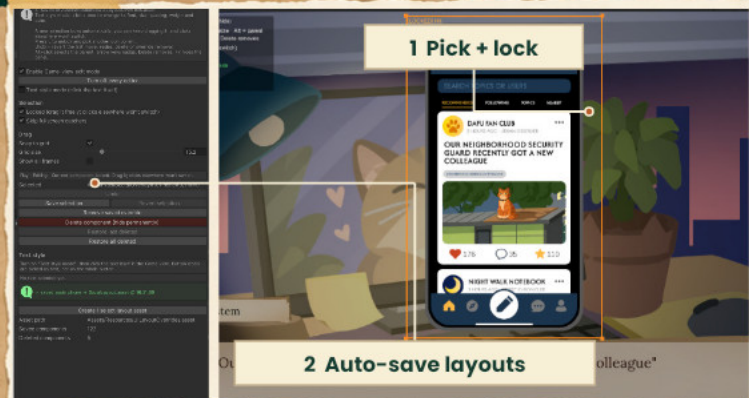
Record when the character reaches a knowledge boundary.

Apply character-specific rules to keep Dafu's replies natural.

UNITY DEVELOPMENT TOOLS

01

RUNTIME UI LAYOUT



1

Resolve the editable target

Pick deeper controls, skip fullscreen catchers, and lock the selection before dragging.

2

Restore layouts after restarting

Save each control's position and size by its place in the UI hierarchy. When the interface is rebuilt, find the same control and restore those settings.

02

INVESTIGATION HOTSPOTS



1

Store normalized bounds

Convert local drag deltas to 0-1 anchors so hit areas scale with the map.

2

Edit hotspots directly on the map

Drag and resize highlighted areas in Game view. Visible outlines and labels make each hotspot easy to identify and adjust.

03

PORTRAIT EDITOR



1

Fit every expression to one slot

Recompute size from shared slot bounds and each sprite's aspect ratio.

2

Preview different characters instantly

Switch between characters to check their size and placement in the scene. Adjust the shared layout while comparing how each portrait fits.

DAWN AWAITS



Dawn Awaits is a 2D narrative puzzle-platformer that turns mirrored movement into metaphor, asking players to guide a girl and her inner self through memories of trauma toward escape and self-acceptance.

PROJECT OVERVIEW

ROLE

Director · Game Designer · Lead Programmer ·
Narrative Designer

ENGINE

Unity

PLATFORMS

Windows / macOS

DEVELOPMENT PERIOD

Sep 2024 – Mar 2025

RELEASE DATE

Mar 11 2025

LANGUAGES

English / Simplified Chinese

600+
PLAYERS

42
REVIEWS

100%
POSITIVE

● DESIGN QUESTION

Could a single player feel the mutual dependence of a cooperative game?

Dawn Awaits began as a USC Unity course prototype exploring a personal question: as an introverted co-op fan, could I recreate the joy of cooperation in a single-player game? I designed a dual-character system where each input affects both characters, requiring players to divide attention, anticipate consequences, and coordinate two spaces. To create this sense of cooperation, I drew on *Binary Land*'s mirrored controls and *It Takes Two*'s cooperative puzzle design and environmental storytelling.

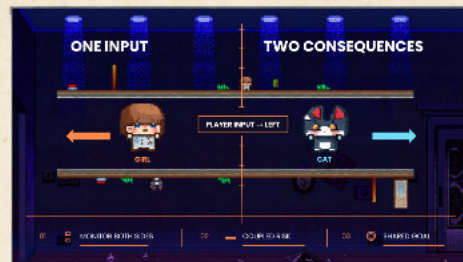


Binary Land (1983) is a reference for simultaneous control and mirrored movement, where every action must account for two characters at once.



It Takes Two (2021) inspired my approach to cooperative puzzle design and environmental storytelling, showing how gameplay mechanics and spaces can work together to express character relationships.

● CORE MECHANIC



ONE INPUT, TWO OUTCOMES

The girl and Domino move in opposite directions under the same input, so progress for one can create danger for the other. Every move becomes a trade-off rather than a simple directional choice.



CO-OP FEELING, SOLO PLAY

I designed the dual-character system to recreate the interdependence of co-op games within a single-player experience, asking players to think for and take care of two characters at once.



THINK BEFORE YOU MOVE

The mechanic challenges divided attention, spatial reasoning, and risk prediction. The fun comes from resolving two conflicting situations with one action—and finding the moment when both sides finally work together.

● FROM SINGLE RULES TO COMBINED PUZZLES

● SINGLE RULES

CORE MECHANICS BUILD THE VOCABULARY

A. CHARACTER STATE

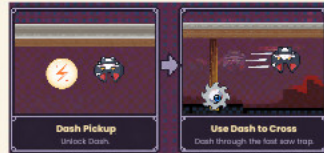
SIZE SHIFT

Changes the character's collider, mobility, and weight.



DASH

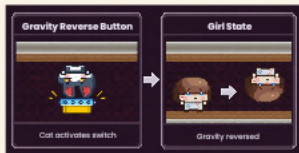
Adds speed and timing pressure to cross traps and brief openings.



B. SPATIAL RULES

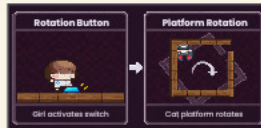
GRAVITY FLIP

Reverses the characters' vertical orientation and falling direction.



INTERACTIVE PLATFORM

One character repositions platforms for the other, making the environment part of their coordination.



C. CONTROL & ATTENTION

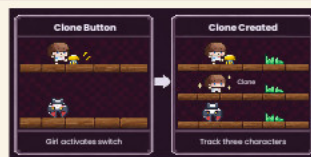
DIRECTION SWITCH

Changes how characters move relative to each other, switching between mirrored and shared movement.



CLONE

Introduces a third clone, requiring players to manage three characters and risks at once.



COMBINE
MECHANICS

NEW
CHALLENGES
EMERGE

● COMBINED PUZZLES

FAMILIAR RULES. NEW INTERACTIONS.

01

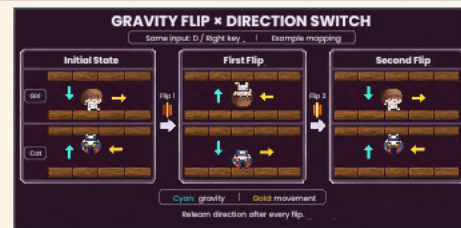
GRAVITY FLIP × DIRECTION SWITCH

MECHANIC DESCRIPTION

Reverses gravity and movement mapping, forcing players to relearn how each input affects both characters.

PLAYER CHALLENGE

Adapt quickly when familiar controls no longer behave as expected.



02

DASH × FIRE TRAP

MECHANIC DESCRIPTION

Combines Dash with timed flame bursts, requiring players to cross during brief safe windows.

PLAYER CHALLENGE

Read the pattern and time each dash between bursts.



03

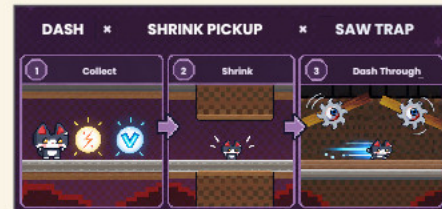
DASH × SHRINK × SAW TRAP

MECHANIC DESCRIPTION

Combines shrinking, dashing, and moving saws, requiring precise timing through narrow gaps.

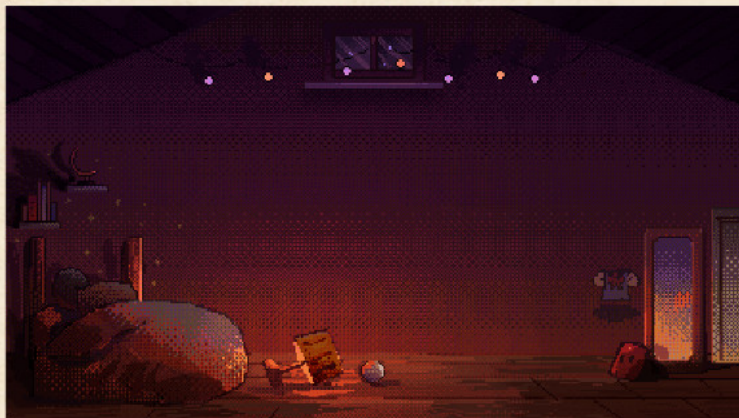
PLAYER CHALLENGE

Carefully consider size, spacing, and dash timing to pass safely.



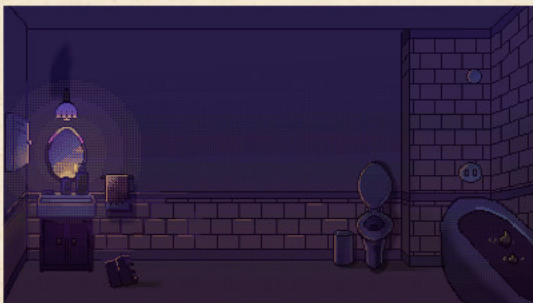
● ENVIRONMENT DESIGN

Home As Confinement



BEDROOM / ATTIC

Introduces the girl's isolated home life and her bond with Domino, turning a familiar bedroom into the starting point of an imagined escape.



BATHROOM

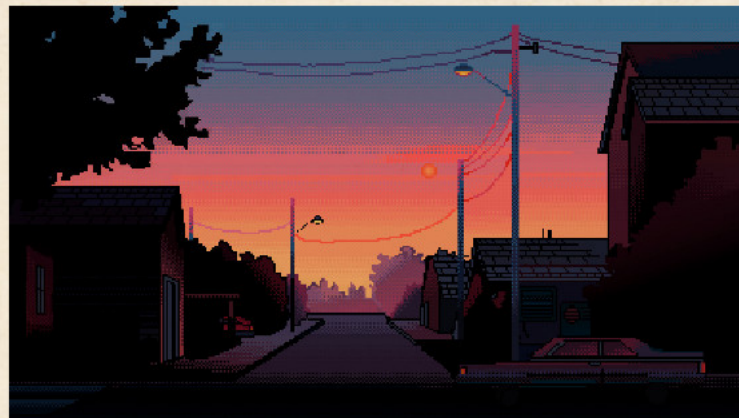
Uses cracked mirrors, steam, and the feeling of being watched to transform the bathroom into an uncomfortable space of exposure.



PARENTS' BEDROOM

Turns signs of marriage and domestic life into symbols of instability and danger, pushing the girl to question why love can hurt.

Escape From Trauma



COMMUNITY STREET

Moves the conflict decisively outside the home as the girl stops imagining escape and actually runs away after Domino is destroyed.



CAR / MOVING SPACE

Turns a family road trip into a "moving sad house," contrasting confinement with the girl's growing desire for freedom and escape.



GRANDPARENTS' HOUSE

Reveals the father as part of the same cycle of violence, offering understanding without forced forgiveness while preserving the girl's anger, boundaries, and agency.

● LEVEL DESIGN EXAMPLE

Bigger, Smaller, Together

This level uses **size-changing abilities to create asymmetric cooperation**: the girl grows to move heavy objects, while Domino shrinks to access spaces she cannot. Together, their different abilities turn one environment into a shared puzzle.

DESIGN GOALS



TEACH THROUGH CONSEQUENCES

Players learn switch-hazard relationships by triggering them and observing the result, rather than through explicit instructions.



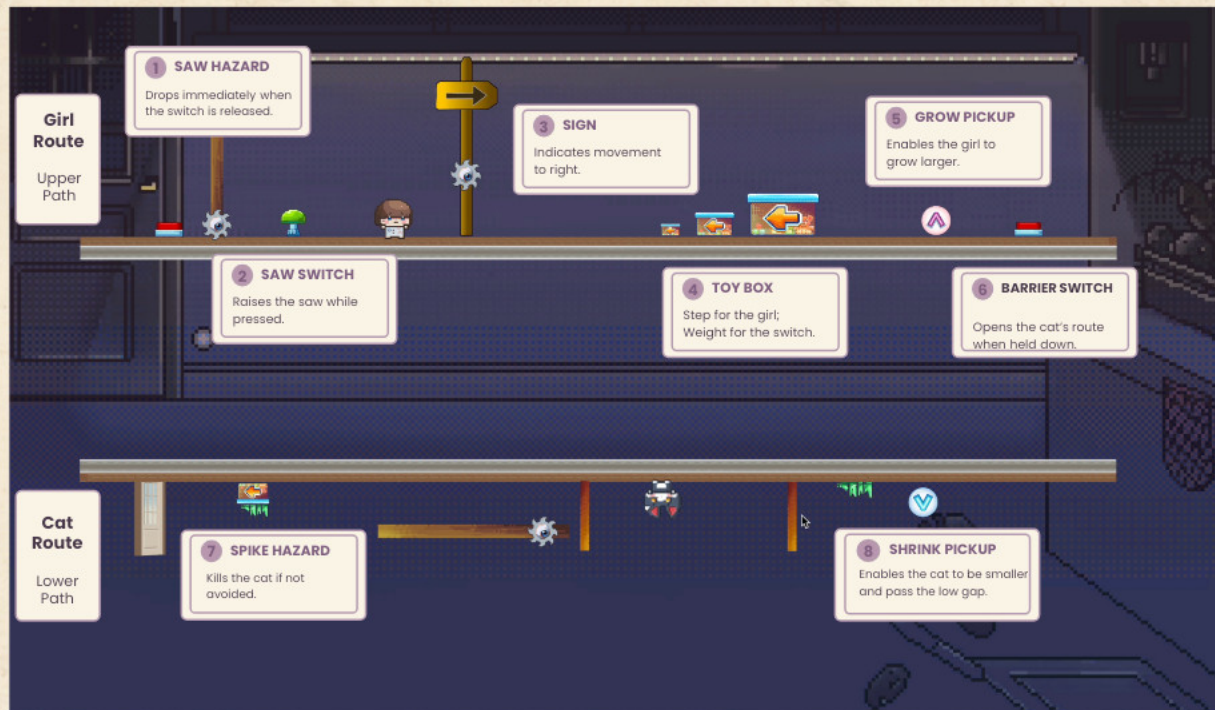
MAKE SIZE MATTER

Use growing and shrinking as meaningful puzzle affordances—changing what each character can move, reach, or pass through.



CREATE ASYMMETRIC COOPERATION

Design the two routes so each character solves a different part of the same puzzle, with one character's actions creating opportunities for the other.



PUZZLE FLOW

1 TRIGGER, OBSERVE, LEARN



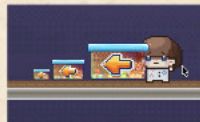
The player is tempted to step on the green switch. Releasing it drops the saw immediately, teaching the switch-saw relationship.

2 TRY ANOTHER DIRECTION



A sign nudges the player right. The toy box acts as a stair so the girl can reach the far side and collect the grow pickup.

3 TURN SIZE INTO STRENGTH



Once enlarged, the girl can push the toy box. The arrow on the box hints that it should be moved left.

4 ONE ACTION, SHARED BENEFITS



Pushing the box fully left both holds down the red switch to free the cat's route and blocks the saw so the girl is safe.

5 ASYMMETRIC PROGRESSION



While the girl solves the upper path, the cat moves through the lower route, avoiding the spike-like hazard and reaching the shrink pickup.

6 NEW SIZE, NEW PATH

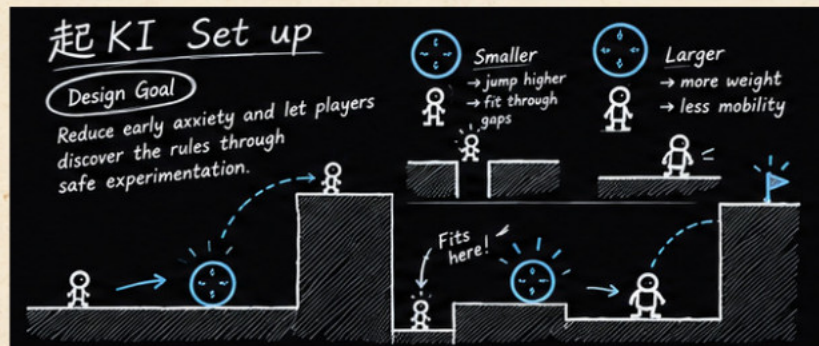


After shrinking, the cat can jump higher and slip through the low gap beneath the trap, a route impossible or lethal at normal size.

LEVEL DESIGN FRAMEWORK

Adapting Nintendo's Kishōtenketsu to Level Design

01 KI / SET UP



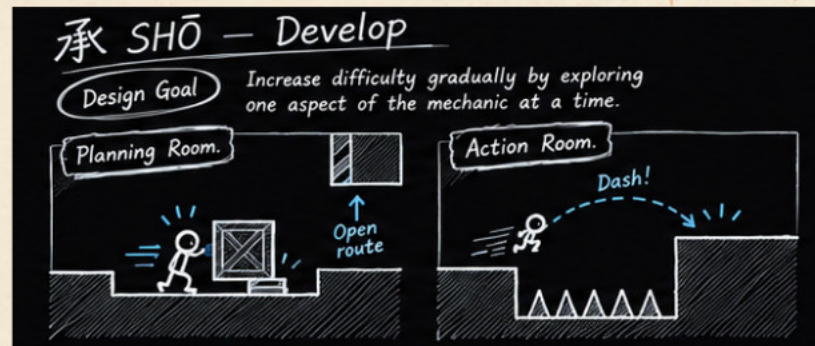
GOAL

Discover the rules through safe experimentation.

EXAMPLE

Size Orbs teach higher jumps and narrow gaps when small, and greater weight with reduced mobility when large.

02 SHŌ / DEVELOP



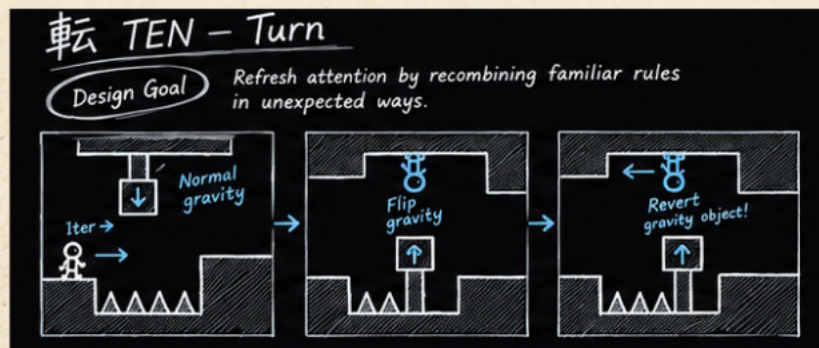
GOAL

Explore one aspect of a mechanic at a time.

EXAMPLE

Use size to push objects and open routes, then combine jumps and dashes with hazards.

03 TEN / TURN



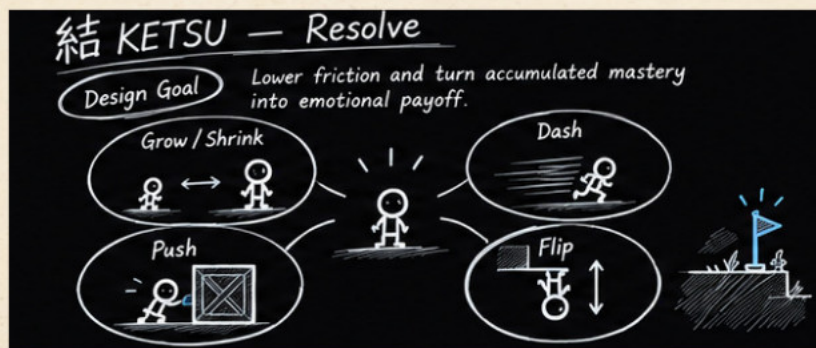
GOAL

Recombine familiar rules to refresh attention.

EXAMPLE

Gravity shifts reverse orientation and movement direction, asking players to rebuild their mental model.

04 KETSU / RESOLVE



GOAL

Turn mastery into emotional payoff.

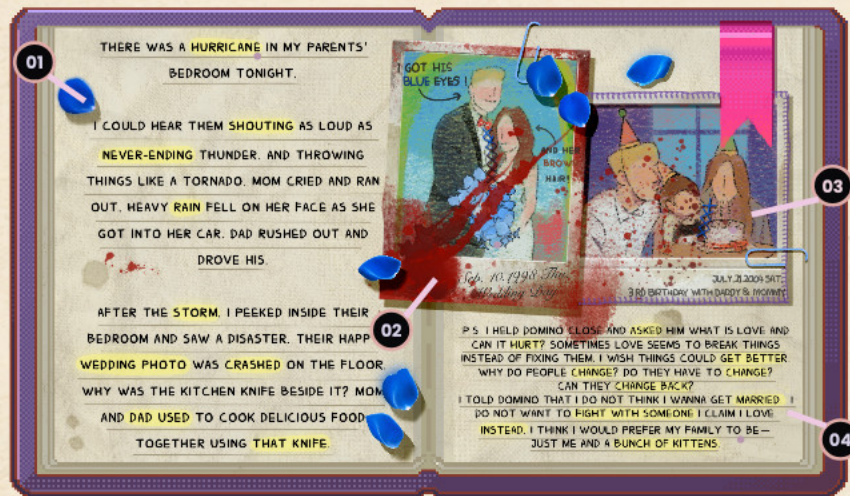
EXAMPLE

Chain familiar actions into a fluid final run, ending with a diary fragment.

DIARY DESIGN

EXAMPLE: PARENTS' BEDROOM

The diary uses a child's innocent voice to describe things she cannot fully understand, allowing the damage of domestic violence to emerge through her confusion and everyday observations—which makes it feel all the more painful.



01 / FALLEN PETALS

The scattered blue petals echo the wedding bouquet, turning a symbol of love into fragments left behind by the "hurricane" of the fight.

03 / FAMILY PHOTOS

Warm wedding and family photos contrast with the violence described, exposing the gap between a happy-family image and reality.

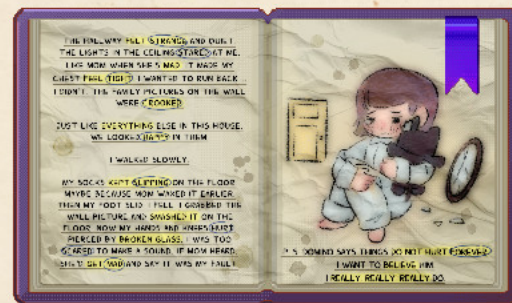
02 / BLOOD-LIKE STAINS

Red stains echo the knife in the diary, suggesting how violence has stained even the girl's warmest memories.

04 / CHILDLIKE HANDWRITING

Uneven handwriting and fluorescent highlights make the diary feel genuinely written by a child and emphasize the details she remembers most.

Page Design & Level Connections



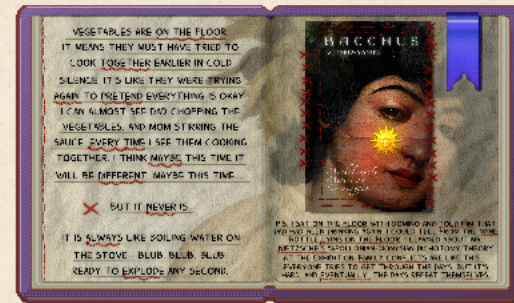
02 / HALLWAY

The injured girl and fallen frame turn physical damage into a broken family portrait.



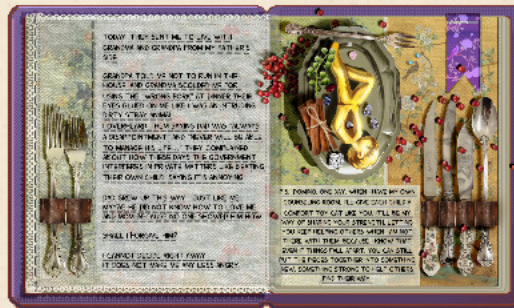
07 / GARAGE

A clockwork butterfly transforms her father's abandoned craft into a symbol of self-repair.



06 / KITCHEN

The Bacchus collage links the kitchen's bottles and ruined meal to drinking and conflicts.



10 / GRANDPARENTS' HOME

Repaired collage and formal cutlery reveal inherited violence without forcing forgiveness.

FROM DESIGN TO UNITY

DUAL-CHARACTER CONTROL

Domino reads the main character's move and jump states. One input is reinterpreted through *moveReversed* to match or mirror movement.

```
private void HandleMovement()
{
    // Read input (apply reverse when needed)
    float inputX = heroController.GetMoveInput();
    if (moveReversed)
        inputX = -inputX;

    Vector2 velocity = rb.velocity;
    velocity.x = inputX * moveSpeed;
    rb.velocity = velocity;

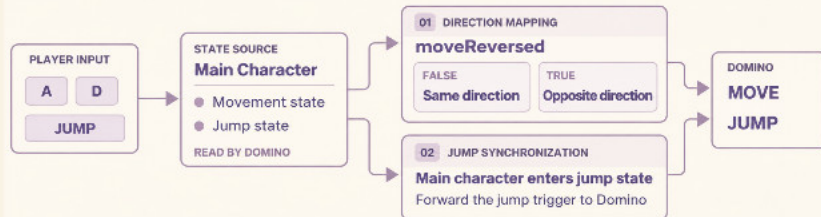
    // Update animation only when state changes
    bool isWalking = Mathf.Approximately(inputX, 0f);
    if (animator.GetBool("isWalking") != isWalking) {
        animator.SetBool("isWalking", isWalking);
    }

    // Flip sprite based on movement direction
    UpdateFacingDirection(inputX);
}

private void HandleJumping()
{
    // Trigger jump on rising edge of input
    bool isJumpPressed = heroController.IsJumping();
    if (isJumpPressed && !heroMacJumping && !isJumping) {
        Jump();
    }
    heroWasJumping = isJumpPressed;
}

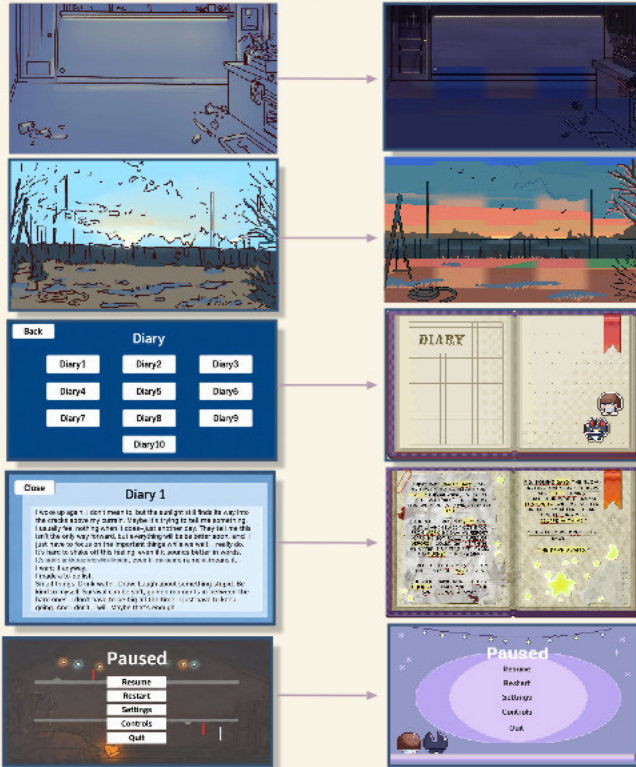
private void Jump()
{
    rb.AddForce(Vector2.up * jumpForce, ForceMode2D.Impulse);
    isJumping = true;
}

private void UpdateFacingDirection(float moveInput)
{
    // Only flip when moving (moveInput, 0f) returns
    if (Mathf.Abs(moveInput) > 0.01f) {
        transform.localScale = new Vector3(Mathf.Sign(moveInput), 1f, 1f);
    }
}
```



DEVELOPMENT PROGRESS

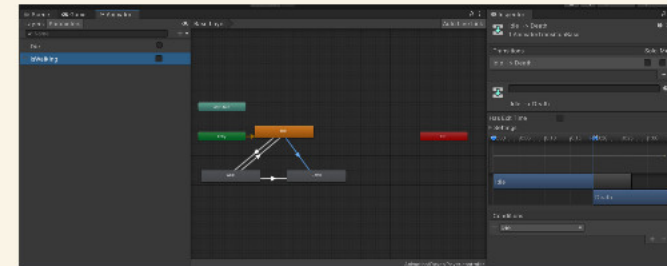
From concepts and level mockups to in-engine implementation.



CHARACTER DESIGN / ANIMATION



Sprite Sheets — Girl / Cat



Animator switches **Idle**, **Walk**, and **Death** based on gameplay states.

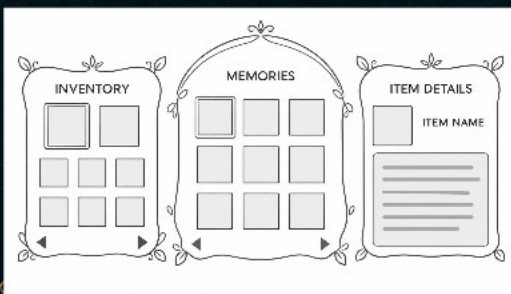
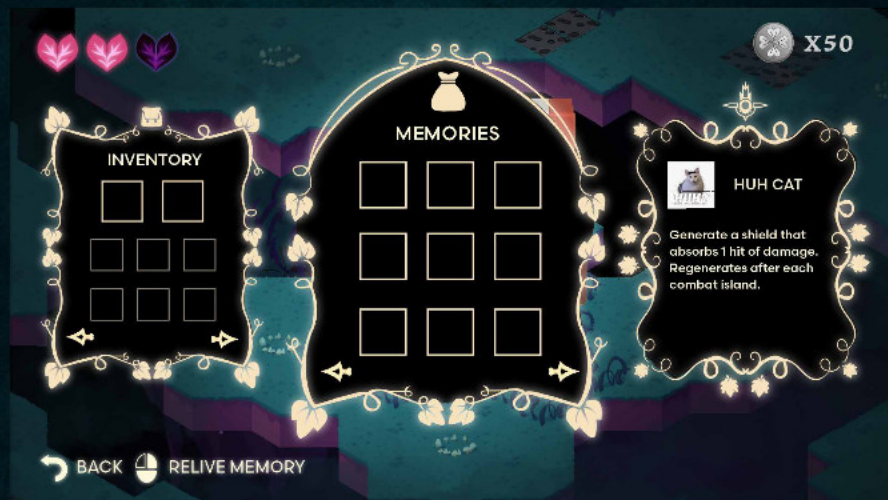
```
1 // Horizontal input updates velocity, isWalking, and sprite direction.
2 rb.velocity = new Vector2(moveInput * moveSpeed, rb.velocity.y);
3 animator.SetBool("isWalking", moveInput != 0);
4 if (moveInput > 0) transform.localScale = Vector3.one;
5 else if (moveInput < 0) transform.localScale = new Vector3(-1, 1, 1);
```

Horizontal input updates velocity, *isWalking*, and sprite direction.

ECHOES: CORE SCREENS & VISUAL FEEDBACK

Lead UI/UX Designer. Hand-drawn, skeuomorphic interfaces for an isometric action roguelite.

INVENTORY & MEMORIES



Memories define the hierarchy. The largest central frame gives lasting progression more visual weight than collected items.

Browse and inspect together. The wireframe places effects beside the grid, keeping the selected object and its function in view.

MAIN MENU



Character art establishes the world. A separate action column keeps Continue and New Game readable against the illustration.

PAUSE MENU



SHARED VISUAL CUES



A shared visual language. Floral curves and butterfly silhouettes bring the game's fae world into the interface. A shared cream palette connects intricate ornament with simpler icons.



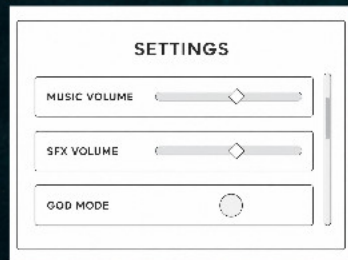
Health uses shape and color. Full, partial and empty leaves show the same resource at different states. The changing fill gives players another cue alongside the pink glow.

Color has a specific role. Pink highlights health and pale cream defines actions. Dark teal keeps the brightest accents tied to useful information.

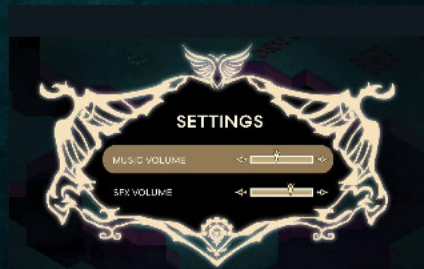
ECHOES: ITERATION & A SHARED VISUAL LANGUAGE

SETTINGS FRAME DEVELOPMENT

01 STRUCTURE



02 FRAME DRAFT



Control layout comes first.

Shared rows pair labels with sliders. The wireframe establishes spacing before decoration enters the panel.

The draft feels too fancy and crowded.

Dense branches compete with labels and slider handles. Their visual weight makes the controls harder to locate.

03 FINAL FRAME



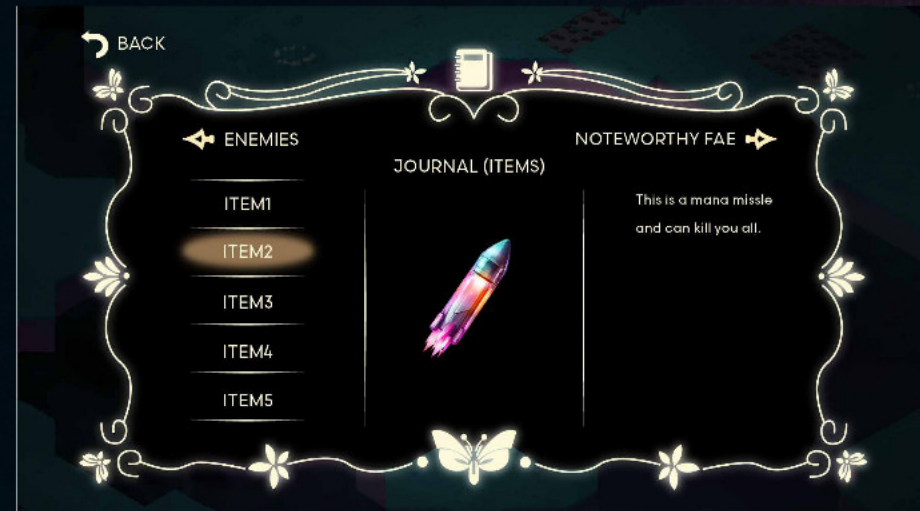
The final frame protects the controls. Ornament moves to the perimeter, leaving a quiet center. Shared spacing separates each row and brighter handles identify adjustable values.

MOTIFS & ICON STUDIES



Silhouettes carry meaning before glow. Books and flames suggest memory and magic, while hand studies explore interaction. Small highlights emphasize the defining feature of each symbol and connect the icons to the cream menu frames.

JOURNAL UI ANATOMY



CATEGORIES

Category tabs sit above the list, giving players a clear place to switch the type of entry they browse.

SELECTION

A warm glow marks the active row. Local highlighting distinguishes selection feedback from the decorative frame.

DETAILS

The illustration and description stay beside the list, so players can read details without losing their browsing position.

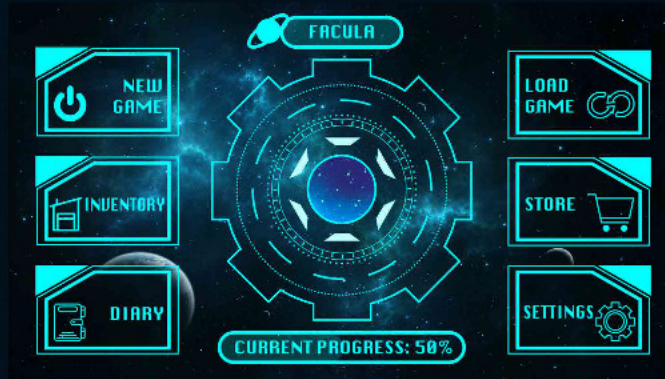
SPACE TRAVELER: NAVIGATION & PLAYER FLOW

A sci-fi adventure about exploring planets and gathering stories for a novel, featuring a cyan holographic UI with crisp outlines and beveled panels.

PLANET-CENTERED NAVIGATION

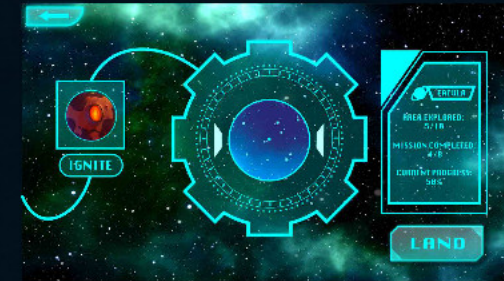


One clear entry action. The Enter prompt starts the journey with a single input beneath the title.



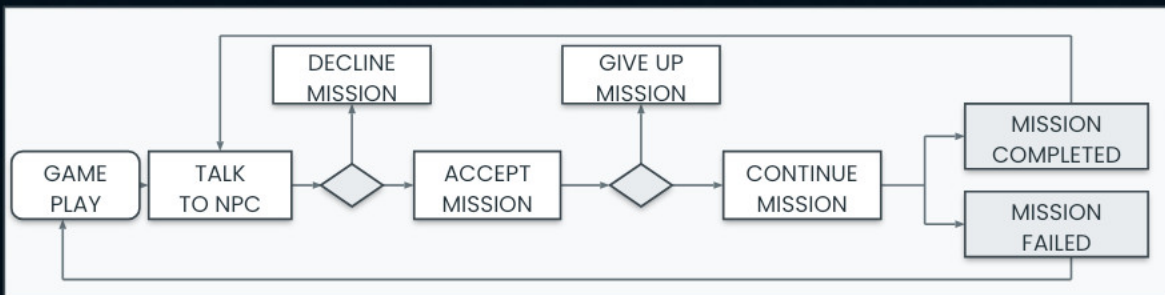
The planet anchors navigation. Six labeled destinations surround the current world. Stable positions keep the planet visible while players choose a function.

Missions remain optional. Players can decline a request or give it up later. The flow below maps these choices back to play, including success and failure.

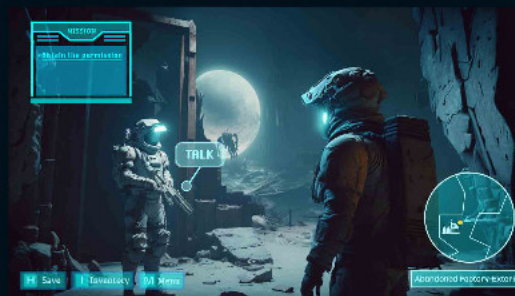


Progress sits beside the next action. Exploration and mission counts stay near LAND for a quick review before entering.

CHOICES WITHIN A MISSION



ROOM FOR EXPLORATION



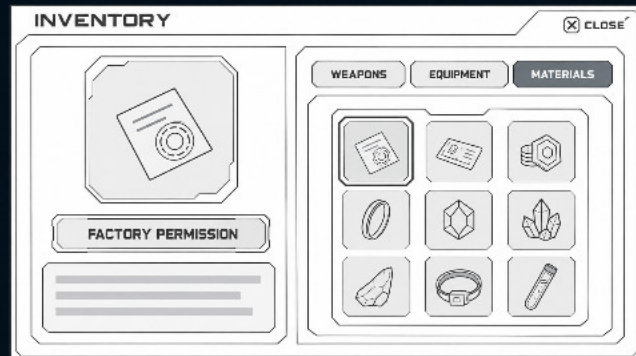
Space for the encounter. TALK stays beside the NPC. Missions and shortcuts occupy the edges, keeping the interaction visible.

SPACE TRAVELER: INVENTORY, DIALOGUE & UI SYSTEM

INVENTORY: BROWSE & INSPECT

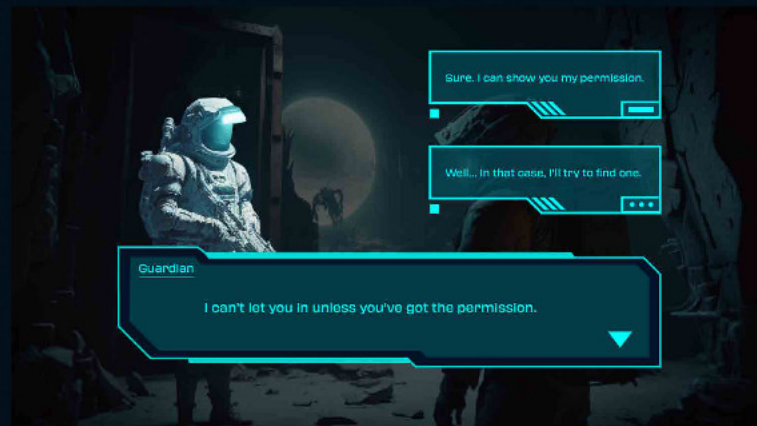


Selection and details share one screen. Category tabs organize the grid, while the left panel shows the active item's illustration and description. Players can inspect an object without opening another view.



The wireframe establishes stable regions. Categories, item slots and the detail panel keep their positions in the final UI. Color and bevels add hierarchy while preserving the original browsing structure.

DIALOGUE WITH A PURPOSE



A dialogue choice leads to a concrete goal. The lower panel holds the NPC's statement, while replies sit beside the character. The permission exchange points players toward an object in the inventory.

FACTORY PERMISSION

Dialogue connects to inventory. The matching Factory Permission label makes the required object identifiable across both screens.

HOLOGRAPHIC UI KIT



Bright edges identify actions. Cyan outlines distinguish buttons from the dark scene. Familiar icons support their labels, so each destination has both a visual cue and a readable name.



Shared geometry. Beveled corners unify the UI. Dark fills support text and cyan outlines define panel boundaries.

DAILY WISHES

DAILY TASKS ACROSS GAME VERSIONS

Daily Wishes rewards players with activity points for completing everyday goals in Infinity Nikki. For v2.0, I redesigned **task selection around story progress** and adapted the interface for content from both versions, making **objectives and reward progress easier to read**.

1 TASK OBJECTIVES & VERSION VISUALS



Task objectives

Goals and required quantities appear more prominently.

Version visuals

1.0 uses wishing bottles. 2.0 introduces shell visuals.



1.0 BOTTLE



2.0 SHELL

Tasks from both versions contribute activity points to the same reward track.

2 UI CHANGES IN 2.0

1.0 ORIGINAL



Task tabs

Separate tabs for Daily Wishes and Starlit Moments.

2.0 REDESIGNED



Reward layout

Rewards grouped in a dedicated right-hand column.

Version-specific Visuals

Familiar bottle motifs retained, with new shell motifs introduced.

3 TASK SELECTION BY PLAYER PROGRESS

Daily tasks are randomly drawn from a pool filtered by each player's gameplay progress and unlocked regions.



STORY PROGRESS

Check completed story milestones.

ONLY 1.0 UNLOCKED

Common + 1.0 tasks

ONLY 2.0 UNLOCKED

Common + 2.0 tasks

BOTH UNLOCKED

Common + 1.0 + 2.0 tasks



RANDOM TASK SELECTION

Draw eligible tasks regardless of current region.

4 TASK APPEARANCE BY VERSION

Shared tasks use bottle or shell visuals based on unlocked regions.

1.0-ONLY TASKS

Bottle visual

2.0-ONLY TASKS

Shell visual

SHARED TASKS / 1.0

Bottle visual

SHARED TASKS / 2.0

Shell visual

SHARED TASK EXAMPLE: SAME OBJECTIVE, EITHER APPEARANCE

1.0 bottle visual



Collect 5 plants

Progress 0 / 5

2.0 shell visual



Collect 5 plants

Progress 0 / 5

Same objective and progress

EXPEDITIONS

REGIONAL ALBUMS & PHOTO CHECK-INS

Regional albums group photo entries and show collection progress.

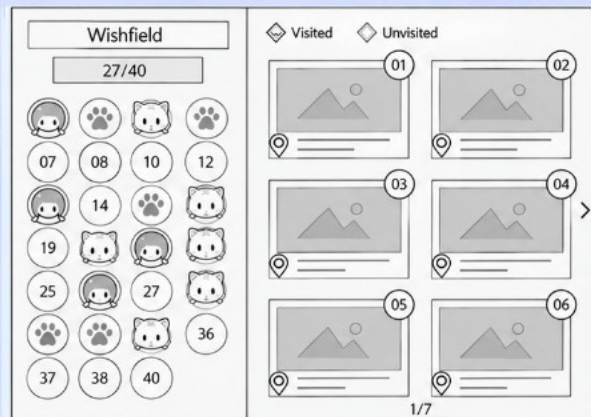
Photo clues guide players to each location, while unlock notifications let them **choose when to view** new content.

1

REGIONAL ALBUM

Regional progress and photo clues in one album.

WIREFRAME



FINAL UI

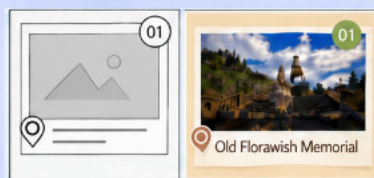


Region overview



Browse a region's locations.

Photo clues



Use photos to find each location.

Collection progress



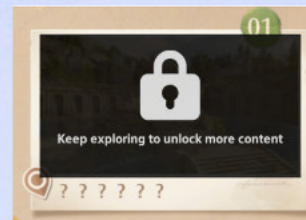
See which spots you have visited.

2

PHOTO CHECK-IN FLOW

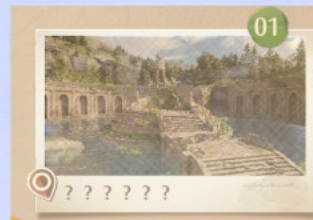
Entries progress from locked locations to photo clues and recorded player photos.

LOCKED



Location details remain hidden until the player meets the unlock conditions.

CLUE AVAILABLE



A reference photo helps the player identify the location.

PHOTO RECORDED



The player's check-in photo replaces the reference image in the album.



Photo Check-in

Use the in-game camera to photograph the location and complete the entry.



Future Content

Coming Soon marks entries reserved for future updates.

3

NEW ENTRY NOTIFICATION HUD

Receive a notice when a new location is unlocked, and choose when to view it.



View now

Opens the newly unlocked entry directly in the album.



Later

Keeps the notification available until the next game session, allowing players to continue exploring.