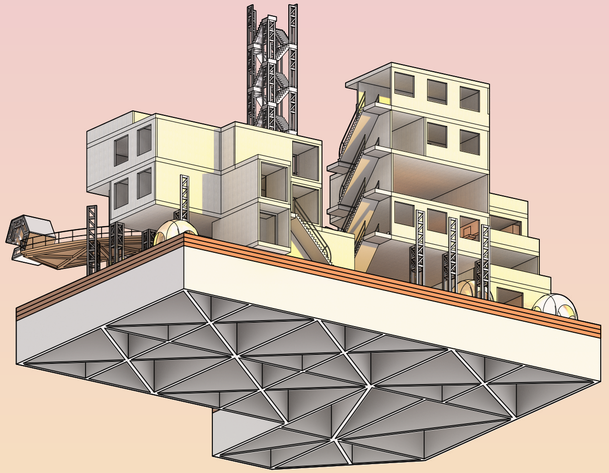
Just an Island

Game publication page:
https://mddddr.itch.io/just-an-island
Video:
https://youtu.be/YIrLdGIwMvQ
Individual project
Tools:

Fest Float

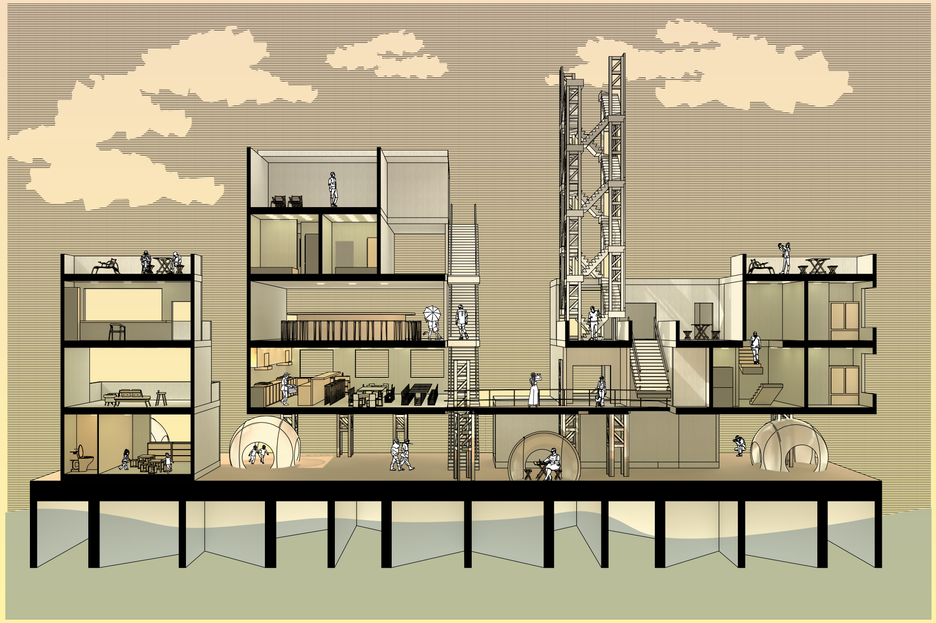
Architecture Design



Located in Pittsburgh's post-industrial Strip District, the project responds to the area's lack of engagement with the riverfront. In recent urban redevelopment, a wave of new housing has emerged: indivual apartment that were fully equipped yet socially hollow, often described as "Adult Dorms."

This design proposes a floating island community that restores connections between people and water. The islands dock along the riverbank to provide everyday housing and commerce, and once a year, they gather in the middle of the river to host a temporary festival.

Through sectional and spatial exploration, the architecture investigates how infrastructure, dwelling, and event can merge into a dynamic, participatory urban form.



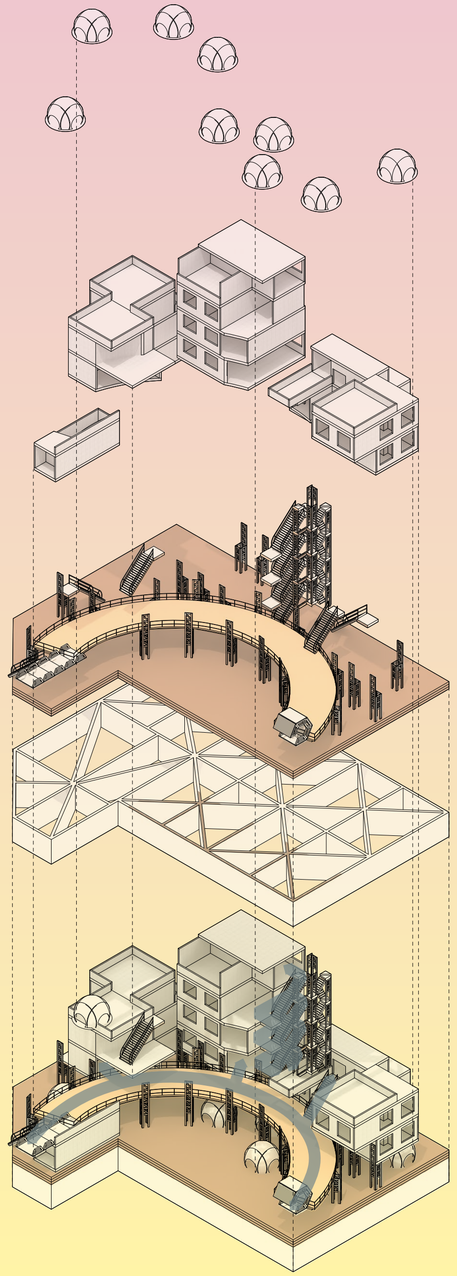
This project lends itself naturally to the format of a walking simulator because its central theme is the experience of place through movement, memory, and atmosphere rather than through action or competition. The architectural proposal already invites slow exploration,which contains the same qualities that define the most expressive walking games.

Translating it into a game transforms architectural space into a narrative of remembrance. The player's movement becomes storytelling: each step recalls a fragment of collective memory, each photograph reconstructs a vanished moment of togetherness. Instead of solving puzzles or achieving goals, the player's reward is emotional discovery through understanding the world by inhabiting it.

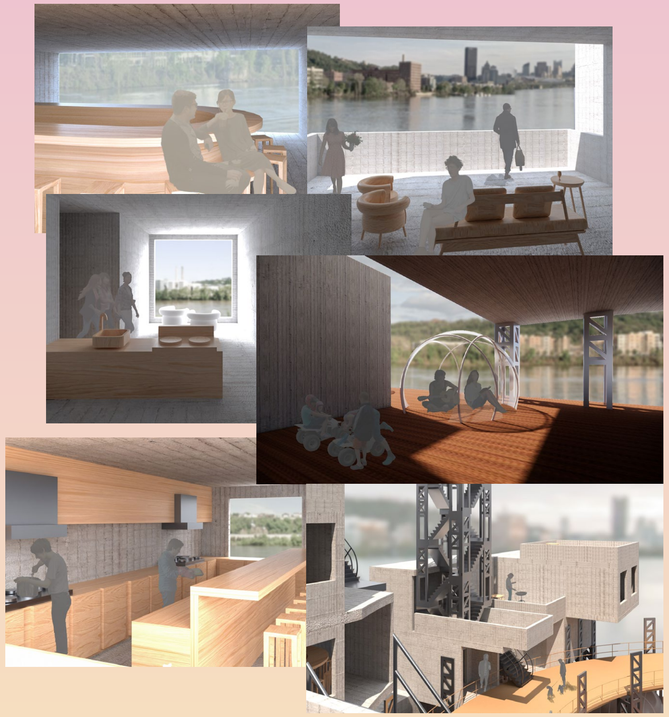
In this way, the project shifts from designing physical environments to designing experiential rhythm. The walking simulator becomes not a replica of the architecture, but its continuation in another medium and creating a way to let people feel what the space was always meant to express.

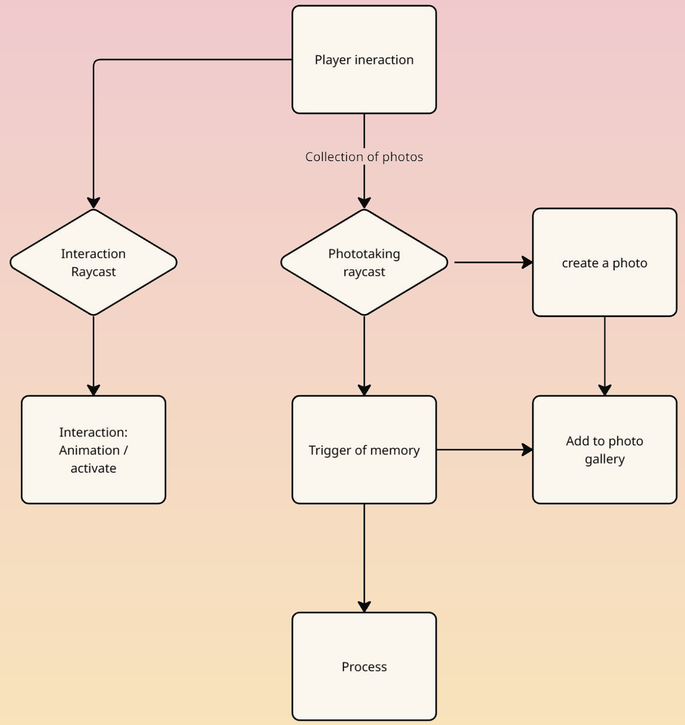
Bubble Light Structures
small, translucent pavilions that host informal gatherings, short pauses, and spontaneous exchanges. They blur the boundary between public and private, creating nodes of encounter.

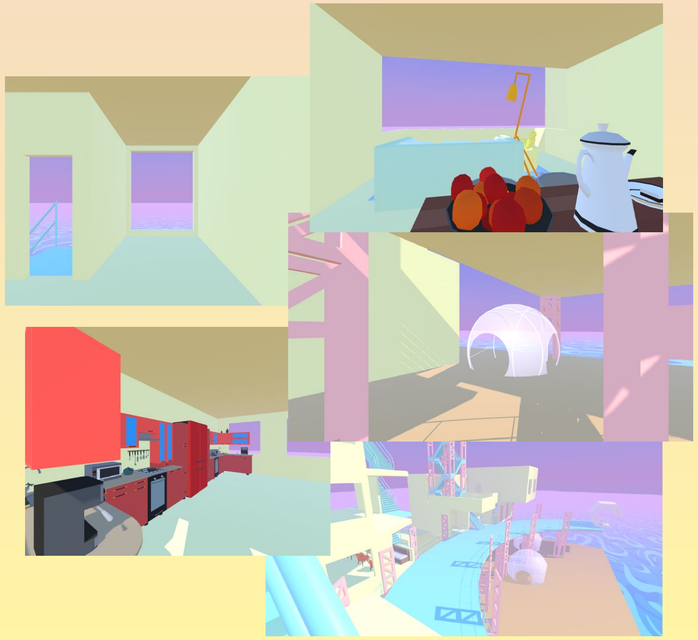
Skybridge
wooden decks close to the water's surface, allowing walking, resting, and direct contact with the river.



Core Gameloop





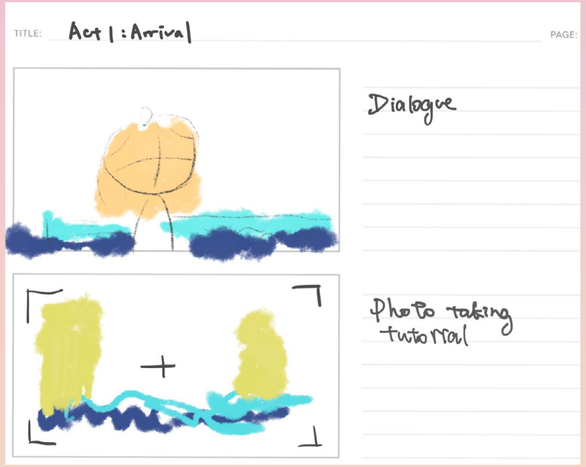


The core mechanic of the game is the retro film camera player carry everywhere. At first glance it behaves like any old analog camera as it enable frame, shot, press the shutter, and a physical photo prints instantly. Every picture player take, no matter when or what you shoot, produces a tangible print that can kept, examine, or give to NPCs.

But the camera has one unusual flaw(or gift).

Every so often, instead of capturing the present moment, the camera will flash back and reveal an image from the island's past. These photos show people, places, or objects as they once were: characters years younger, recipies that no longer remembered, or the good old time when people still hang out. These unexpected glimpses into history become essential clues. They can spark memories in the island's residents, unlock new dialogue, and guide the player toward solutions for current problems.

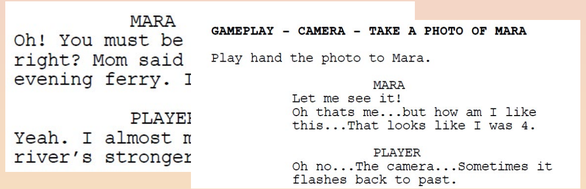
As the story progresses, all flashback photos are automatically added to the gallery wall, a communal archive space inside the living quarters. The wall slowly fills with fragments of the island's forgotten days, letting players physically walk through the past and see how every memory connects.



Act 1: Anchored Arrival

Arrive at the bridge entrance, photo basics, deck floor remain unavailable

NPC:
Mara

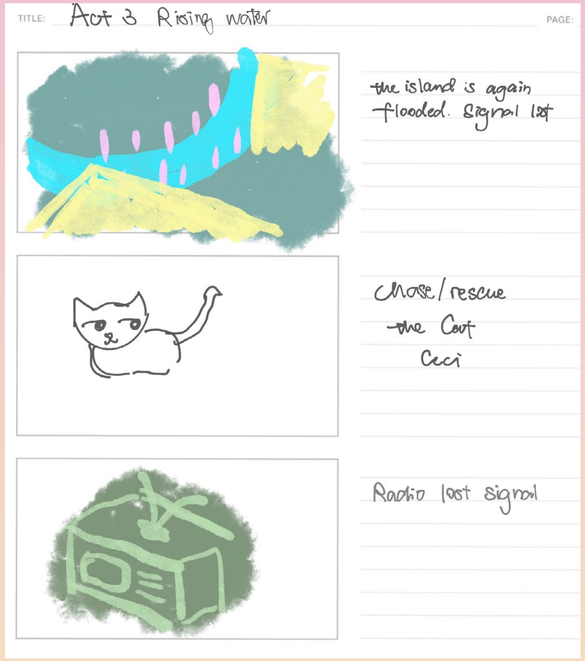


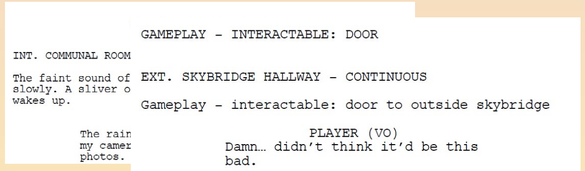
Dialogues:
Trigger to mechanism, intro to phototaking

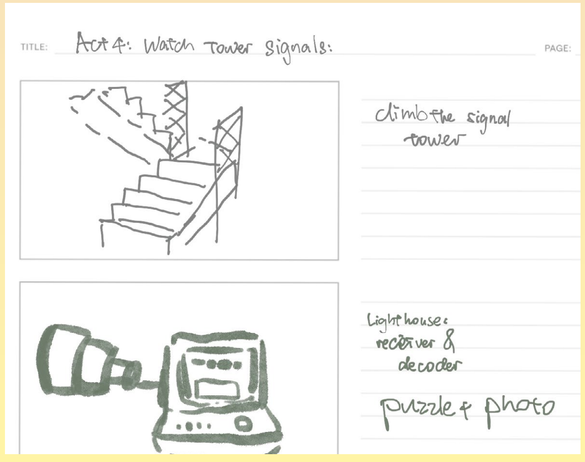
Act 2: Daily Life

Deck open, learn fishing, make cat food, rest for the night.

NPC:
Mara at the pet center
Kai fishing







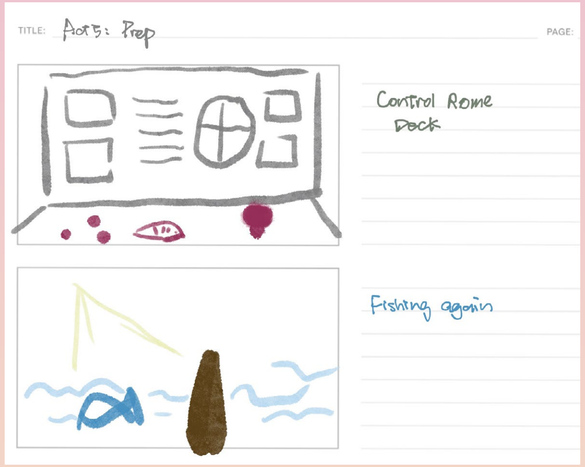
Act 3: Rising Water

Player wake up found the dock level is flooded again. Rain isolates island; help NPCs protect pets and supplies.

NPC:
Mara-caling for cat
Ceci: the cat

Act 4: Watchtower Signals

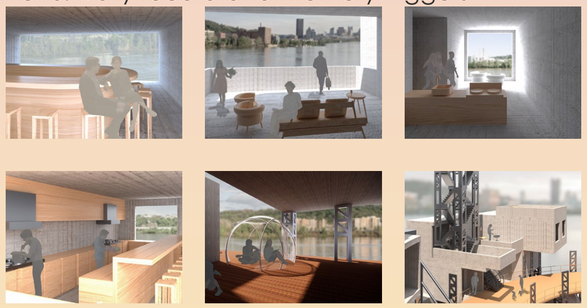
Now need to rely on the ancient watch tower for comm. watchtower to decode and send light messages.



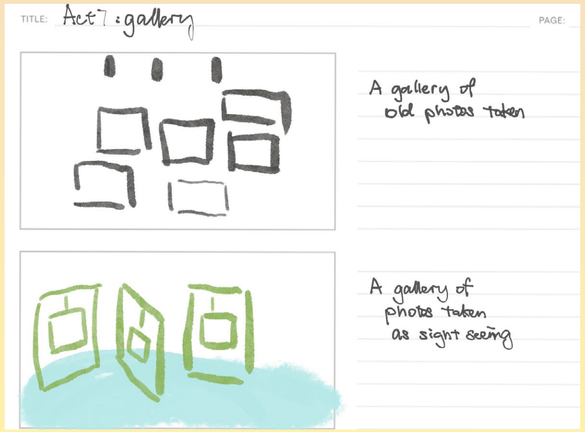
Act 5: Adaption and Preparation

Activate floating mechanism to move the island. More fishing for preperation

Act 6: Daily record and memory triggers



Explore Island and trigger memories with camera.
community kitchen - old time when they bake together
Bar- old time when they date
The bubble: kids used to whisper in the bubbles

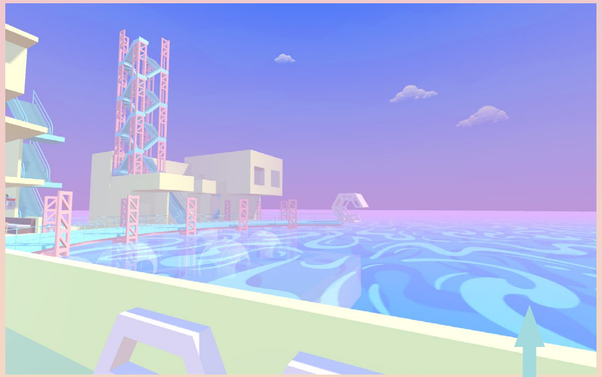


Act 7: Resolution

Calm morning celebration of resilience.
View final photo gallery of memories.

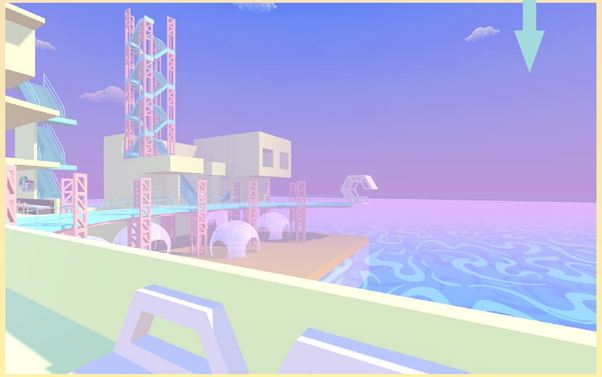
Narrative and Plot

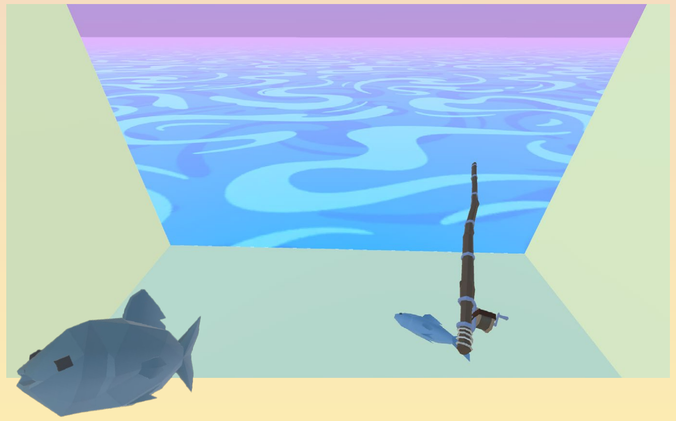
Interactable objects



The water level raise and falls according to the wheather information along the gameplay plot. It simulate the flood system and constrain the movable area for the player. As the player experience the island, the water level might continuously goes up and down, leaving different areas for exploration.

Dialogue trigger: taking to the NPCs, pushing and processing the storyline to beyond.



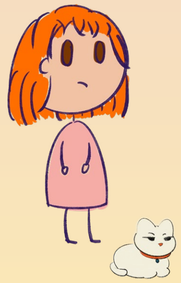


Fishing on the deck, enabling the player character to get prepared to the island life sytle. This mechanism only exists when the lower dock is available for exploration.

KAI
Age: 22
Role: Architecture student / island repairman and fisher
Location: Communal living and lower docks

Grounded, dry sense of humor, quiet but thoughtful. He treats the island like a living system as something he both maintains and mourns. He's the one who sees the island's architecture as fragile "lungs," always leaking or patching.

Gameplay presence:
-Introduces crafting/fishing mechanics and teaches the player how the island functions physically.
-Serves as an emotional midpoint character, bridges the island's human and structural conditions.
-Later becomes the player's ally when investigating the flooding or mechanical failures.choly.



MARA
Age: 17
Role: Pet daycare worker / daughter of the island's organizer
Location: Pet center, cafe area

Mara acts as the player's first guide and anchor. Her dialogue introduces the tone of island life. Through her eyes, players glimpse what the community used to be: lively, tight-knit, with old rituals like the festival.

Gameplay presence:
-Introduces core mechanics: camera tutorial, navigation between spaces.
-Her cat Ceci creates the first chain of events leading to the island's leak mystery.
-Her later scenes oscillate between light humor and subtle melancholy.

CECI
Age: 5(cat years)
Role: Mara's pet / symbolic "messenger" between spaces
Location: Pet center, cafe area

Independent but attached to Mara, often seen watching people from elevated spots. She tends to vanish before storms. Locals say animals sense changes before humans do.

Gameplay presence:
-Introduces chase/interactable mechanics.
-Acts as a soft environmental guide, where Ceci runs, something significant usually lies ahead.
-Later, she might appear in old photos the camera takes, even in places she shouldn't have been.

Chracter Design

Playtests and Feedback

Experience

“The camera really ties the narrative together.”
“Flashbacks were surprising and cool, felt meaningful.”
“I love getting physical photos.”
The camera mechanic created excitement and surprise.
Characters (especially Mara and Ceci) created warmth and connection.

Observation

All players quickly experimented with the camera.
Players like the multi-floor village concept, but environmental readability is too low.
They are struggling with vertical transitions, visual hierarchy, and interactable visibility.

Hypothesis

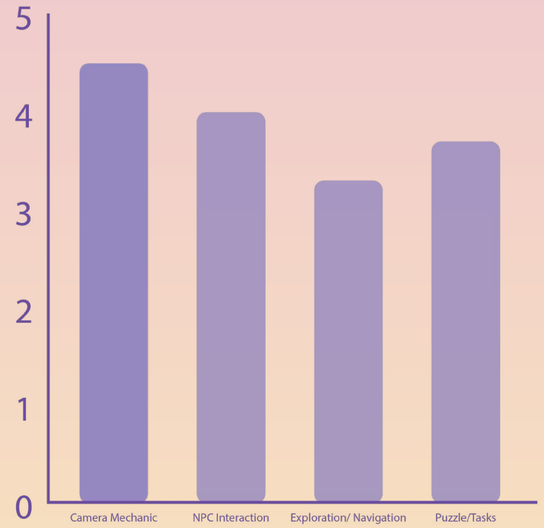
Enhancing environmental cues and adding subtle guidance will reduce disorientation without diminishing exploration.
Early-game objectives and backstory context can be reinforced to ground players in the island’s story while keeping mystery intact.

Advise

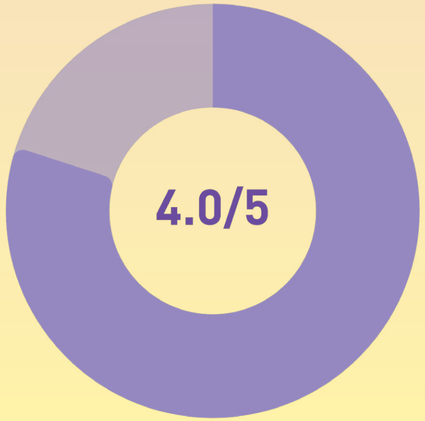
Implement a guided first flashback and add subtle visual/audio hints to indicate upcoming flashbacks. Highlight the gallery wall early to reinforce the connection between physical photos and the narrative.
Add minor animated elements such as rain

Ratings of different gameplays

Participants rated Just an Island on key gameplay elements from 1 (poor) to 5 (excellent). The camera mechanic scored highest (4.5/5), praised for flashbacks and physical photos. NPC interactions were strong (4.0/5), with Mara, Kai, and Ceci engaging players effectively. Exploration/navigation was weaker (3.3/5), with some players struggling to find key areas. Puzzle/tasks scored moderately (3.7/5), enjoyable but sometimes unclear.
Overall enjoyment was 4.3/5, showing the game’s narrative and atmosphere resonated well.
The playtest demonstrates that Just an Island has a solid core of gameplay enjoyment, with particularly strong narrative and camera-driven elements. Improvements in navigation clarity, interaction affordances, and task guidance will enhance overall player experience and accessibility.

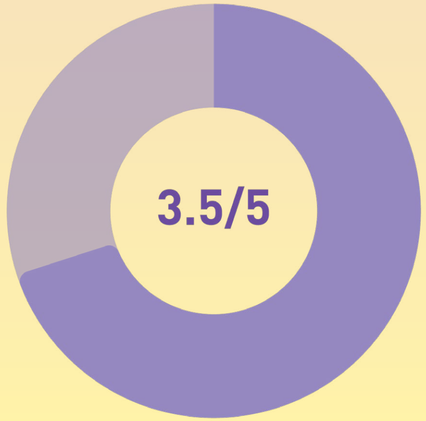


Art & World Atmosphere



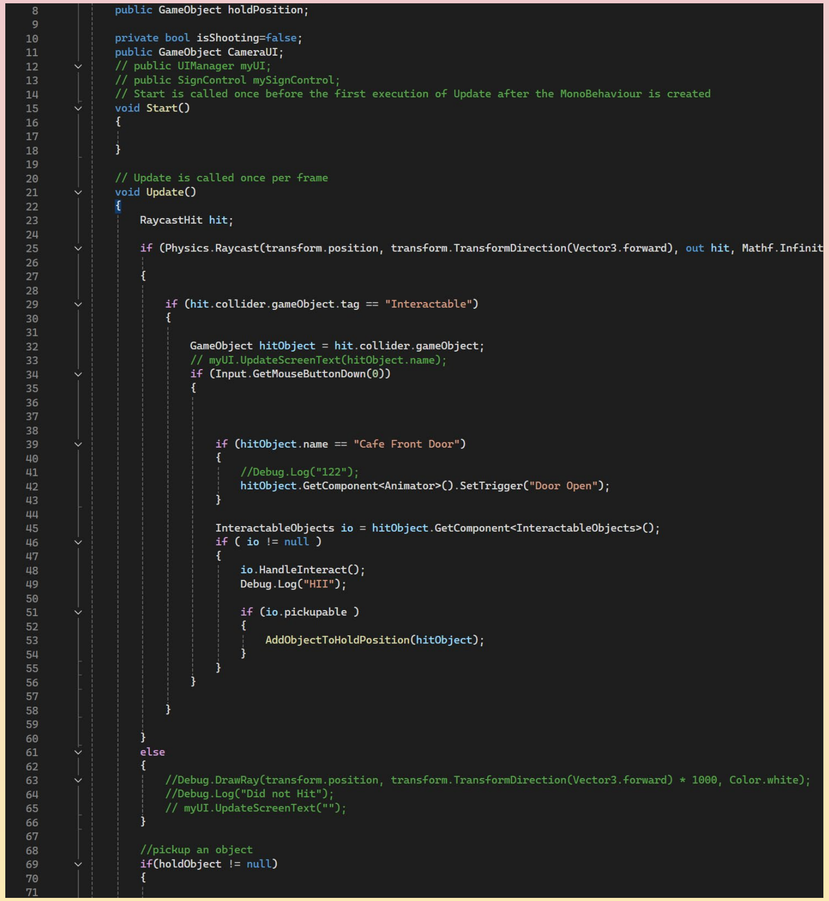
“Feels like a slightly magical fishing village.”

Clarity of Goals

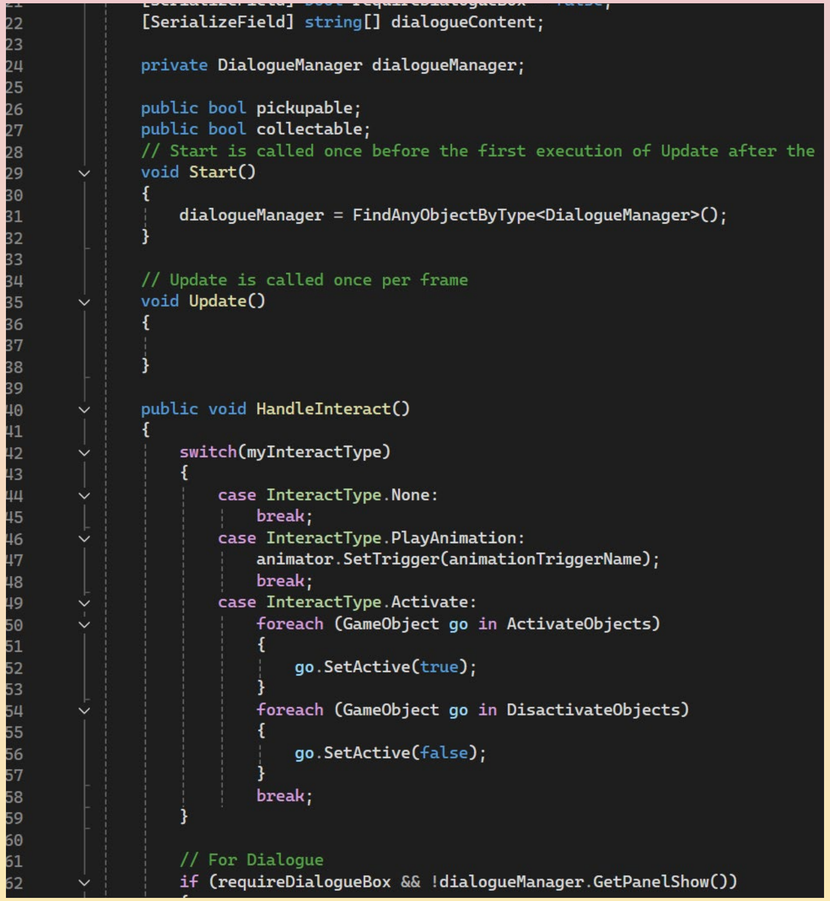


Early-game goals were not always clear. Players wanted a little more hand-holding at the start.

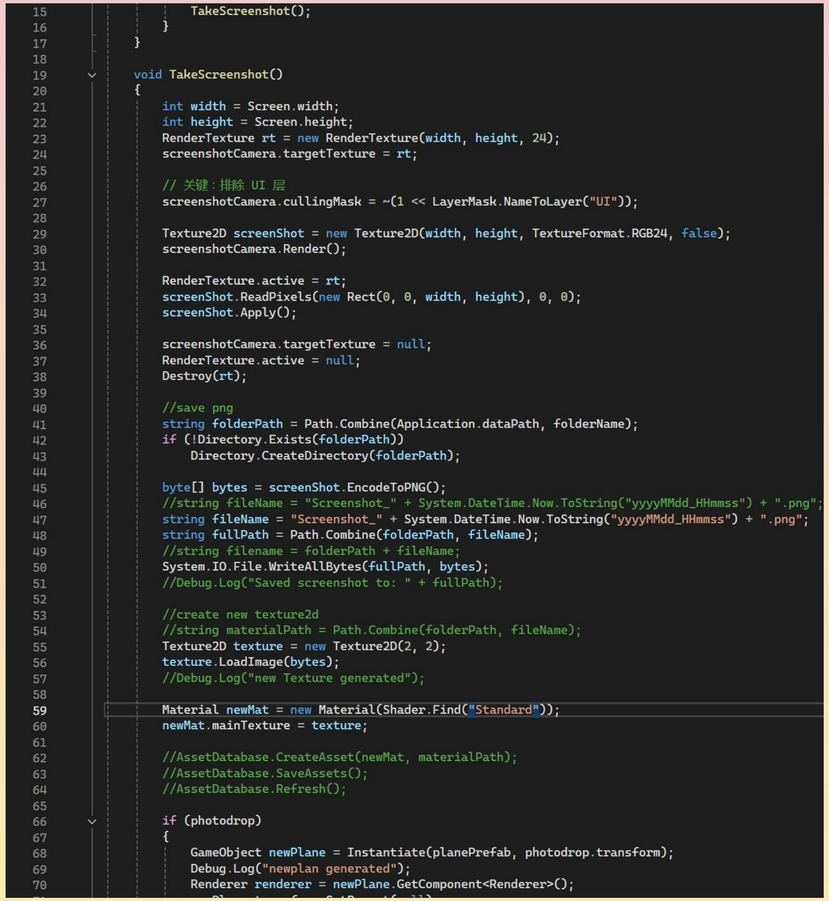
Unity Programming



```
    public GameObject holdPosition;

    private bool isShooting=false;
    public GameObject CameraUI;
    // public UIManager myUI;
    // public SignControl mySignControl;
    // Start is called once before the first execution of Update after the MonoBehaviour is created
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        RaycastHit hit;

        if (Physics.Raycast(transform.position, transform.TransformDirection(Vector3.forward), out hit, Mathf.Infinit

        {

            if (hit.collider.gameObject.tag == "Interactable")
            {

                GameObject hitObject = hit.collider.gameObject;
                // myUI.UpdateScreenText(hitObject.name);
                if (Input.GetMouseButtonDown(0))
                {



                    if (hitObject.name == "Cafe Front Door")
                    {
                        //Debug.Log("122");
                        hitObject.GetComponent<Animator>().SetTrigger("Door Open");
                    }

                    InteractableObjects io = hitObject.GetComponent<InteractableObjects>();
                    if ( io != null )
                    {
                        io.HandleInteract();
                        Debug.Log("HII");

                        if (io.pickupable )
                        {
                            AddObjectToHoldPosition(hitObject);
                        }
                    }
                }

            }
        }
        else
        {
            //Debug.DrawRay(transform.position, transform.TransformDirection(Vector3.forward) * 1000, Color.white);
            //Debug.Log("Did not Hit");
            // myUI.UpdateScreenText("");
        }

        //pickup an object
        if(holdObject != null)
        {
```



```
    [SerializeField] string[] dialogueContent;

    private DialogueManager dialogueManager;

    public bool pickupable;
    public bool collectable;
    // Start is called once before the first execution of Update after the
    void Start()
    {
        dialogueManager = FindAnyObjectByType<DialogueManager>();
    }

    // Update is called once per frame
    void Update()
    {

    }

    public void HandleInteract()
    {
        switch(myInteractType)
        {
            case InteractType.None:
                break;
            case InteractType.PlayAnimation:
                animator.SetTrigger(animationTriggerName);
                break;
            case InteractType.Activate:
                foreach (GameObject go in ActivateObjects)
                {
                    go.SetActive(true);
                }
                foreach (GameObject go in DisactivateObjects)
                {
                    go.SetActive(false);
                }
                break;
        }

        // For Dialogue
        if (requireDialogueBox && !dialogueManager.GetPanelShow())
```



```
            TakeScreenshot();
        }
    }

    void TakeScreenshot()
    {
        int width = Screen.width;
        int height = Screen.height;
        RenderTexture rt = new RenderTexture(width, height, 24);
        screenshotCamera.targetTexture = rt;

        // 关键：排除 UI 层
        screenshotCamera.cullingMask = ~(1 << LayerMask.NameToLayer("UI"));

        Texture2D screenShot = new Texture2D(width, height, TextureFormat.RGB24, false);
        screenshotCamera.Render();

        RenderTexture.active = rt;
        screenShot.ReadPixels(new Rect(0, 0, width, height), 0, 0);
        screenShot.Apply();

        screenshotCamera.targetTexture = null;
        RenderTexture.active = null;
        Destroy(rt);

        //save png
        string folderPath = Path.Combine(Application.dataPath, folderName);
        if (!Directory.Exists(folderPath))
            Directory.CreateDirectory(folderPath);

        byte[] bytes = screenShot.EncodeToPNG();
        //string fileName = "Screenshot_" + System.DateTime.Now.ToString("yyyyMMdd_HHmmss") + ".png";
        string fileName = "Screenshot_" + System.DateTime.Now.ToString("yyyyMMdd_HHmmss") + ".png";
        string fullPath = Path.Combine(folderPath, fileName);
        //string filename = folderPath + fileName;
        System.IO.File.WriteAllBytes(fullPath, bytes);
        //Debug.Log("Saved screenshot to: " + fullPath);

        //create new texture2d
        //string materialPath = Path.Combine(folderPath, fileName);
        Texture2D texture = new Texture2D(2, 2);
        texture.LoadImage(bytes);
        //Debug.Log("new Texture generated");

        Material newMat = new Material(Shader.Find("Standard"));
        newMat.mainTexture = texture;

        //AssetDatabase.CreateAsset(newMat, materialPath);
        //AssetDatabase.SaveAssets();
        //AssetDatabase.Refresh();

        if (photodrop)
        {
            GameObject newPlane = Instantiate(planePrefab, photodrop.transform);
            Debug.Log("newplan generated");
            Renderer renderer = newPlane.GetComponent<Renderer>();
```

PlayerRaycast

The Player Raycasting script allows the player to interact with objects and NPCs by detecting what they are looking at. It casts an invisible ray from the camera, highlights interactable objects, and triggers actions like dialogue, object activation, item pickup, or animation. This system enables smooth exploration, ensures key gameplay events occur reliably, and keeps interactions intuitive throughout the game.

Interactable Obejct

Handles all objects the player can interact with, such as NPCs, items, or environmental triggers. It defines how objects respond to player input (Click), including initiating dialogue, animations, activating or disactivating objects or pickuping the object, and can display visual cues like highlights or prompts. It is a reaction towards the player raycast.

Screenshot

Manages the in-game camera functionality, allowing the player to bring up the camera UI for aiming with P key and take photos with the Z key. Each shot generates a physical material in assset folder and create a physical in-game photo out of that material. While shooting, it can trigger flashbacks, and stores images for display in the gallery wall, integrating narrative and gameplay.