Stranded
Desert Island

Team 13

CS4361.001

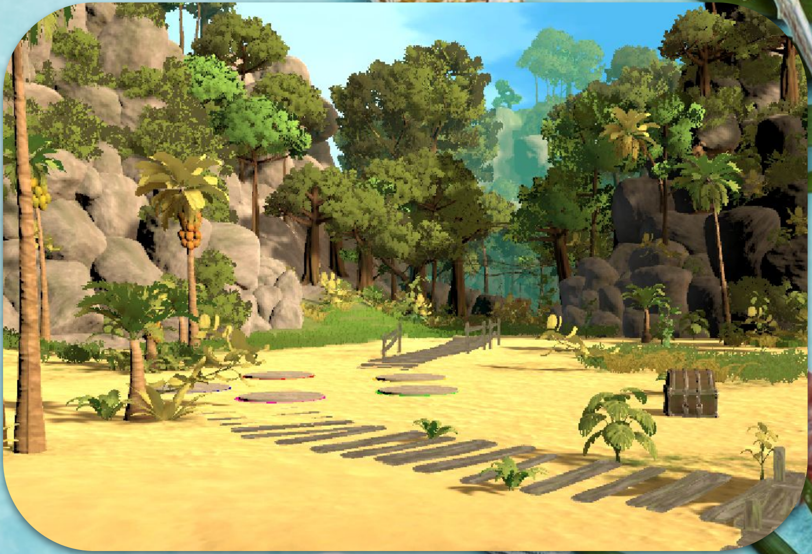
Dr. Pushpa Kumar

Kevin Nguyen
Nikhil Maraboyina
Ousama Batais
Noah Bowman
Nisai Sun

Project Overview

The game starts with a user stranded on an island, user must complete challenges and puzzles to collects planks of wood to rebuild their boat and escape the island

Inspired by adventure games like **MYST**, that requires exploration and world-relative interactivity.



Tools Used

Unity

Unity

blender®

Blender

GitHub

Discord

Discord

C#

C#

Womp

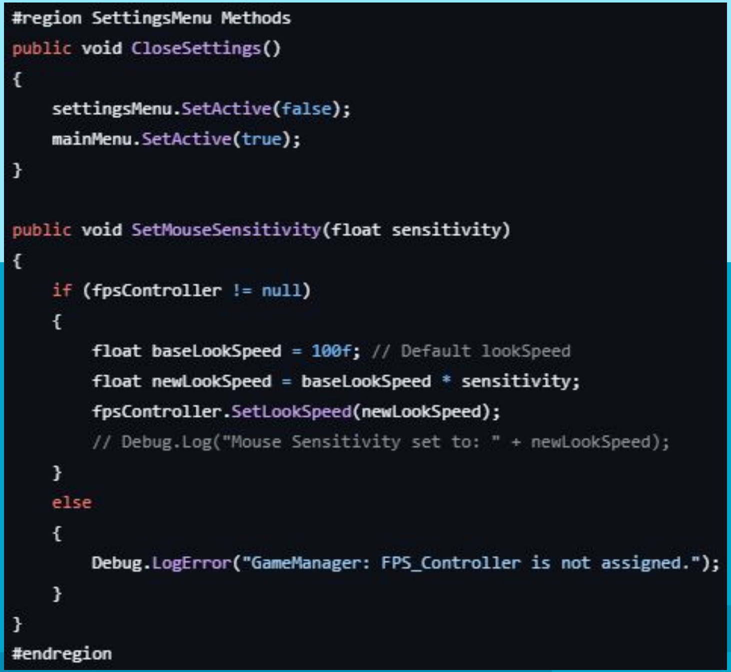
Map, Menu, & Player Movement

Player Movement & Camera Control (via FPS_Controller.cs):

- Handles walking, jumping, via CharacterController
- Movement is calculated along local X (side-to-side) & Z (forward-backward) axis relative to player orientation

Game Flow & UI Management (via GameManager.cs):

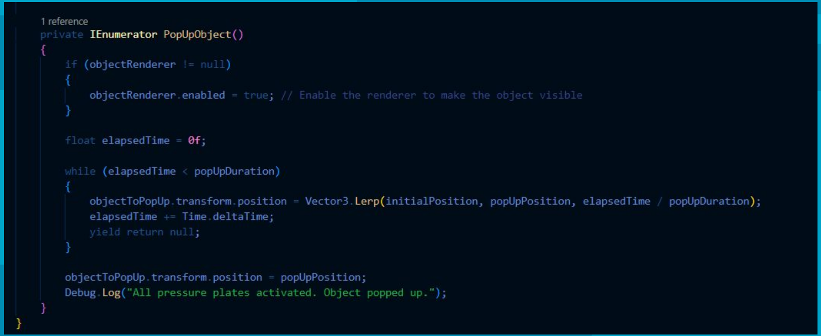
- Manages game states (like the main menu, pause menu, settings menu) & transition like Escape key
- Adjust settings like mouse sensitivity, updating player look speed in FPS_Controller.cs



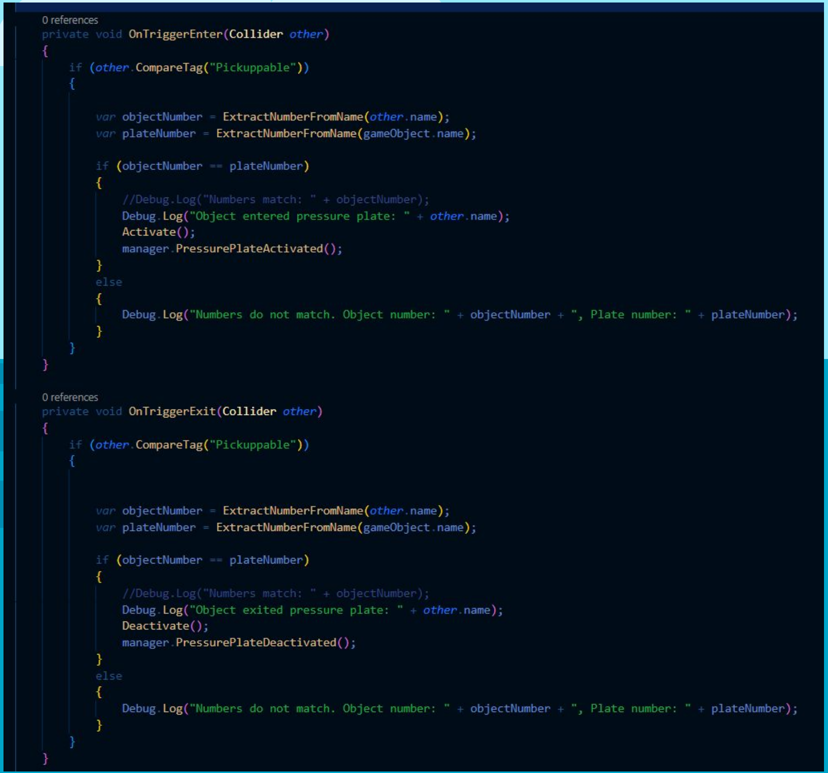
```
#region SettingsMenu Methods
public void CloseSettings()
{
    settingsMenu.SetActive(false);
    mainMenu.SetActive(true);
}

public void SetMouseSensitivity(float sensitivity)
{
    if (fpsController != null)
    {
        float baseLookSpeed = 100f; // Default lookSpeed
        float newLookSpeed = baseLookSpeed * sensitivity;
        fpsController.SetLookSpeed(newLookSpeed);
        // Debug.Log("Mouse Sensitivity set to: " + newLookSpeed);
    }
    else
    {
        Debug.LogError("GameManager: FPS_Controller is not assigned.");
    }
}
#endregion
```

Pressure Plate Functionality

- Each of the pressure plates has a collider.
- When a “pickuppable” object with the correct name lands on it, the plate activates.
- Plate Manager - waits for specific set of plates to activate and then raises chest.



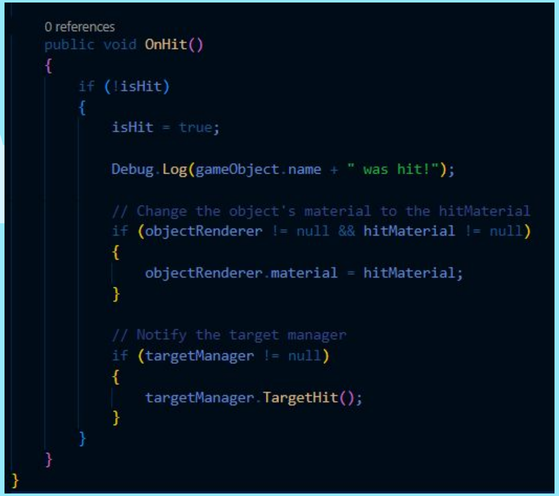
```
1 reference
private IEnumerator PopUpObject()
{
    if (objectRenderer != null)
    {
        objectRenderer.enabled = true; // Enable the renderer to make the object visible
    }

    float elapsedTime = 0f;

    while (elapsedTime < popUpDuration)
    {
        objectToPopUp.transform.position = Vector3.Lerp(initialPosition, popUpPosition, elapsedTime / popUpDuration);
        elapsedTime += Time.deltaTime;
        yield return null;
    }

    objectToPopUp.transform.position = popUpPosition;
    Debug.Log("All pressure plates activated. Object popped up.");
}
}
```



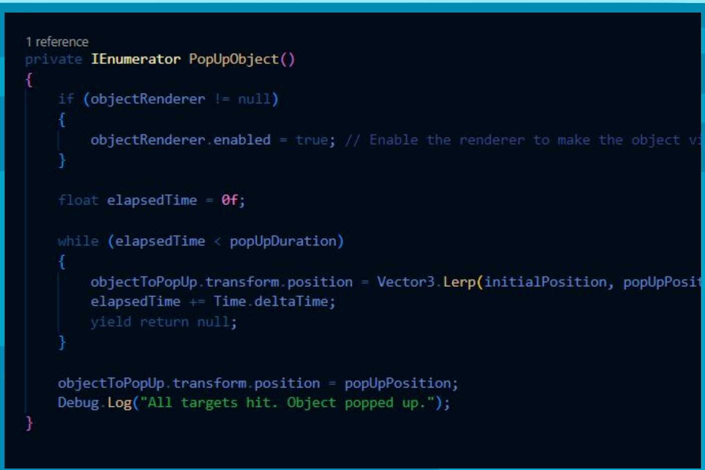
```
0 references
private void OnTriggerEnter(Collider other)
{
    if (other.CompareTag("Pickuppable"))
    {

        var objectNumber = ExtractNumberFromName(other.name);
        var plateNumber = ExtractNumberFromName(gameObject.name);

        if (objectNumber == plateNumber)
        {
            //Debug.Log("Numbers match: " + objectNumber);
            Debug.Log("Object entered pressure plate: " + other.name);
            Activate();
            manager.PressurePlateActivated();
        }
        else
        {
            Debug.Log("Numbers do not match. Object number: " + objectNumber + ", Plate number: " + plateNumber);
        }
    }
}

0 references
private void OnTriggerExit(Collider other)
{
    if (other.CompareTag("Pickuppable"))
    {


        var objectNumber = ExtractNumberFromName(other.name);
        var plateNumber = ExtractNumberFromName(gameObject.name);

        if (objectNumber == plateNumber)
        {
            //Debug.Log("Numbers match: " + objectNumber);
            Debug.Log("Object exited pressure plate: " + other.name);
            Deactivate();
            manager.PressurePlateDeactivated();
        }
        else
        {
            Debug.Log("Numbers do not match. Object number: " + objectNumber + ", Plate number: " + plateNumber);
        }
    }
}
```

Gun Functionality

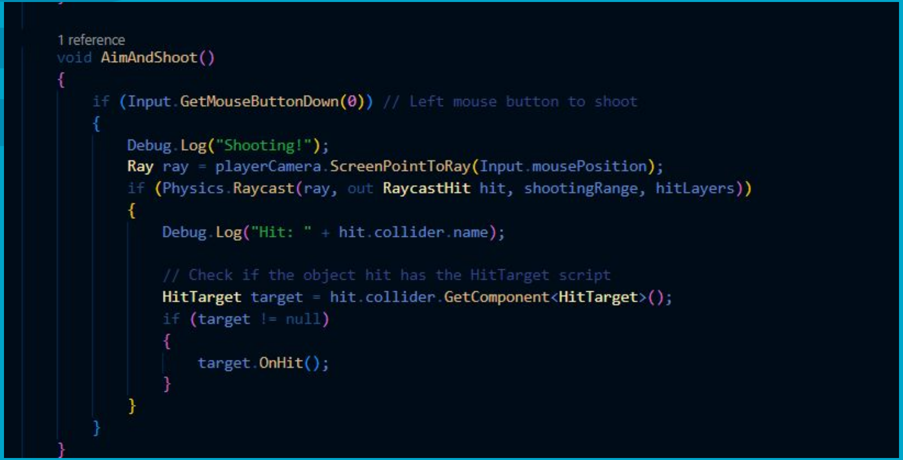
- Builds off of the same functionality of the pressure plates
- Gun sends out a ray, if target lies within the ray, it activates
- Lerp Vector Transformation for Chest to raise from Ground



```
0 references
public void OnHit()
{
    if (!isHit)
    {
        isHit = true;

        Debug.Log(gameObject.name + " was hit!");

        // Change the object's material to the hitMaterial
        if (objectRenderer != null && hitMaterial != null)
        {
            objectRenderer.material = hitMaterial;
        }

        // Notify the target manager
        if (targetManager != null)
        {
            targetManager.TargetHit();
        }
    }
}
}
```

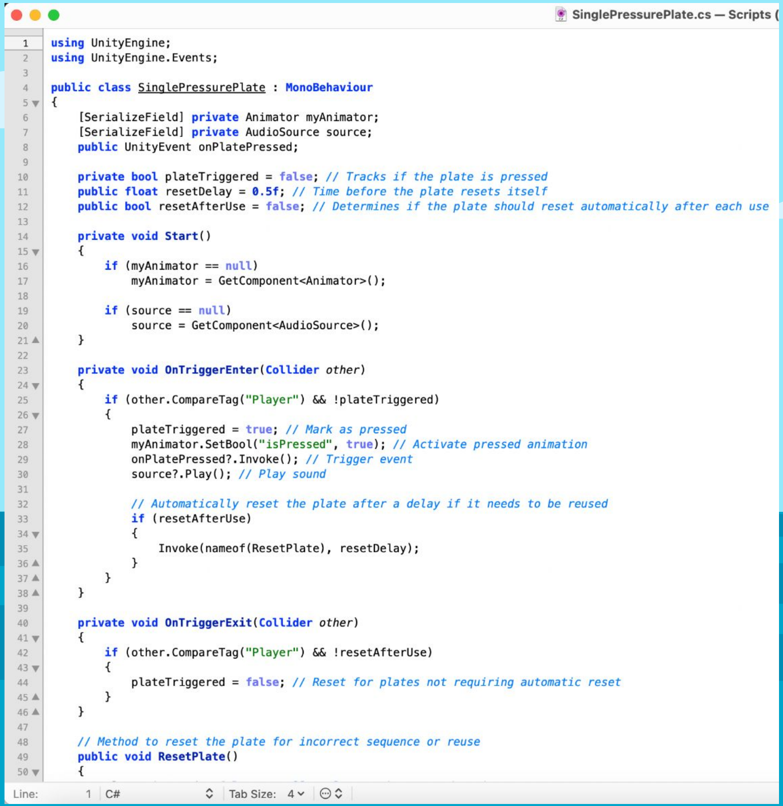


```
1 reference
private IEnumerator PopUpObject()
{
    if (objectRenderer != null)
    {
        objectRenderer.enabled = true; // Enable the renderer to make the object vi
    }

    float elapsedTime = 0f;

    while (elapsedTime < popUpDuration)
    {
        objectToPopUp.transform.position = Vector3.Lerp(initialPosition, popUpPosit
        elapsedTime += Time.deltaTime;
        yield return null;
    }

    objectToPopUp.transform.position = popUpPosition;
    Debug.Log("All targets hit. Object popped up.");
}
```



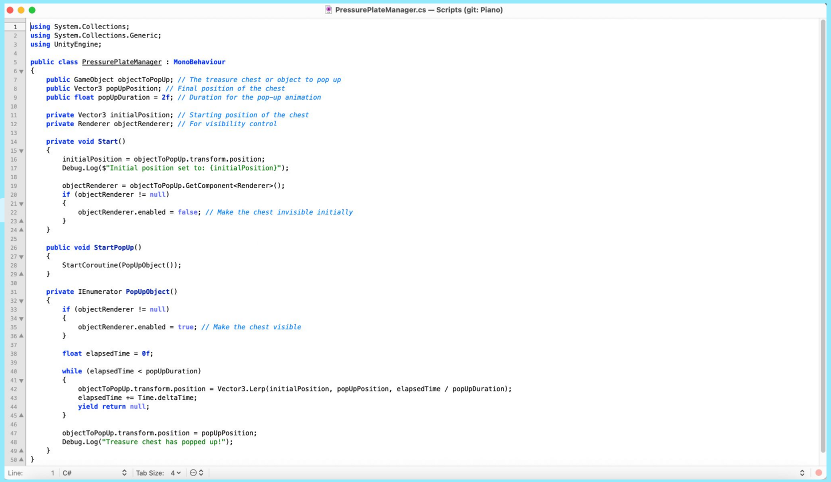
```
1 reference
void AimAndShoot()
{
    if (Input.GetMouseButtonDown(0)) // Left mouse button to shoot
    {
        Debug.Log("Shooting!");
        Ray ray = playerCamera.ScreenPointToRay(Input.mousePosition);
        if (Physics.Raycast(ray, out RaycastHit hit, shootingRange, hitLayers))
        {
            Debug.Log("Hit: " + hit.collider.name);

            // Check if the object hit has the HitTarget script
            HitTarget target = hit.collider.GetComponent<HitTarget>();
            if (target != null)
            {
                target.OnHit();
            }
        }
    }
}
```

Piano Puzzle



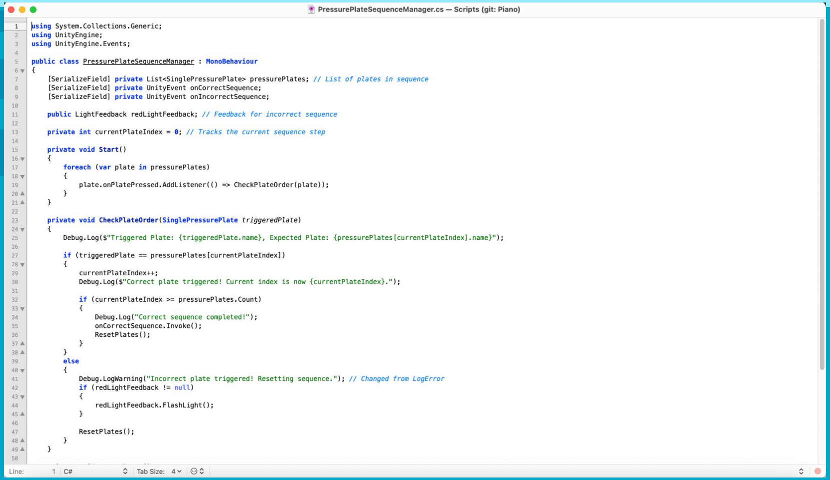
SinglePressurePlate.cs — Scripts (

```csharp
using UnityEngine;
using UnityEngine.Events;

public class SinglePressurePlate : MonoBehaviour
{
    [SerializeField] private Animator myAnimator;
    [SerializeField] private AudioSource source;
    public UnityEvent onPlatePressed;

    private bool plateTriggered = false; // Tracks if the plate is pressed
    public float resetDelay = 0.5f; // Time before the plate resets itself
    public bool resetAfterUse = false; // Determines if the plate should reset automatically after each use

    private void Start()
    {
        if (myAnimator == null)
            myAnimator = GetComponent<Animator>();

        if (source == null)
            source = GetComponent<AudioSource>();
    }

    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player") && !plateTriggered)
        {
            plateTriggered = true; // Mark as pressed
            myAnimator.SetBool("isPressed", true); // Activate pressed animation
            onPlatePressed?.Invoke(); // Trigger event
            source?.Play(); // Play sound

            // Automatically reset the plate after a delay if it needs to be reused
            if (resetAfterUse)
            {
                Invoke(nameof(ResetPlate), resetDelay);
            }
        }
    }

    private void OnTriggerExit(Collider other)
    {
        if (other.CompareTag("Player") && !resetAfterUse)
        {
            plateTriggered = false; // Reset for plates not requiring automatic reset
        }
    }

    // Method to reset the plate for incorrect sequence or reuse
    public void ResetPlate()
    {
```



PressurePlateManager.cs — Scripts (git: Piano)

```csharp
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PressurePlateManager : MonoBehaviour
{
    public GameObject objectToPopUp; // The treasure chest or object to pop up
    public Vector3 popUpPosition; // Final position of the chest
    public float popUpDuration = 2f; // Duration for the pop-up animation

    private Vector3 initialPosition; // Starting position of the chest
    private Renderer objectRenderer; // For visibility control

    private void Start()
    {
        initialPosition = objectToPopUp.transform.position;
        Debug.Log($"Initial position set to: {initialPosition}");

        objectRenderer = objectToPopUp.GetComponent<Renderer>();
        if (objectRenderer != null)
        {
            objectRenderer.enabled = false; // Make the chest invisible initially
        }
    }

    public void StartPopUp()
    {
        StartCoroutine(PopUpObject());
    }

    private IEnumerator PopUpObject()
    {
        if (objectRenderer != null)
        {
            objectRenderer.enabled = true; // Make the chest visible
        }

        float elapsedTime = 0f;

        while (elapsedTime < popUpDuration)
        {
            objectToPopUp.transform.position = Vector3.Lerp(initialPosition, popUpPosition, elapsedTime / popUpDuration);
            elapsedTime += Time.deltaTime;
            yield return null;
        }

        objectToPopUp.transform.position = popUpPosition;
        Debug.Log("Treasure chest has popped up!");
    }
}
```

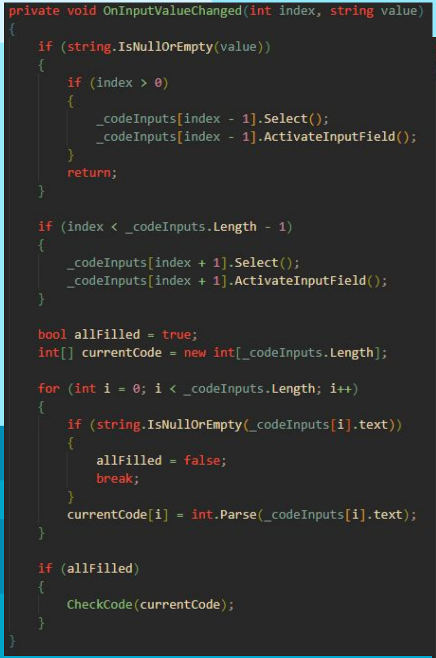


PressurePlateSequenceManager.cs — Scripts (git: Piano)

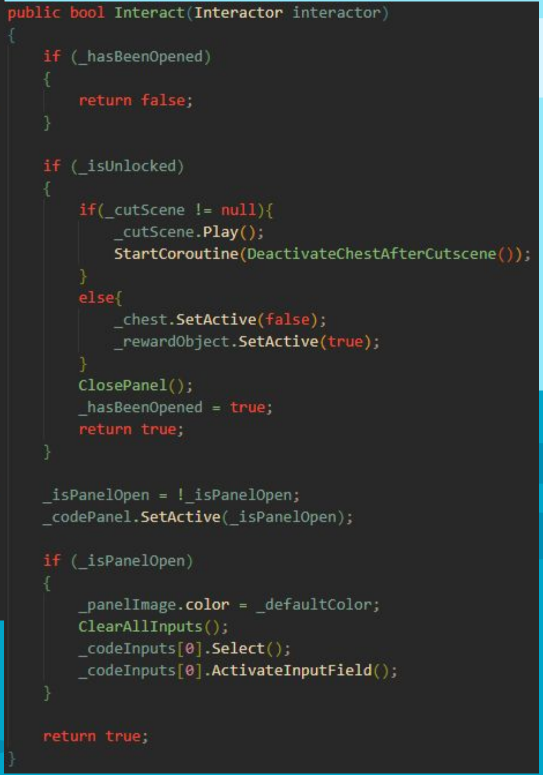
```csharp
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Events;

public class PressurePlateSequenceManager : MonoBehaviour
{
    [SerializeField] private List<SinglePressurePlate> pressurePlates; // List of plates in sequence
    [SerializeField] private UnityEvent onCorrectSequence;
    [SerializeField] private UnityEvent onIncorrectSequence;

    public LightFeedback redLightFeedback; // Feedback for incorrect sequence

    private int currentPlateIndex = 0; // Tracks the current sequence step

    private void Start()
    {
        foreach (var plate in pressurePlates)
        {
            plate.onPlatePressed.AddListener(() => CheckPlateOrder(plate));
        }
    }

    private void CheckPlateOrder(SinglePressurePlate triggeredPlate)
    {
        Debug.Log($"Triggered Plate: {triggeredPlate.name}, Expected Plate: {pressurePlates[currentPlateIndex].name}");

        if (triggeredPlate == pressurePlates[currentPlateIndex])
        {
            currentPlateIndex++;
            Debug.Log($"Correct plate triggered! Current index is now {currentPlateIndex}.");

            if (currentPlateIndex >= pressurePlates.Count)
            {
                Debug.Log("Correct sequence completed!");
                onCorrectSequence.Invoke();
                ResetPlates();
            }
        }
        else
        {
            Debug.LogWarning("Incorrect plate triggered! Resetting sequence."); // Changed from LogError
            if (redLightFeedback != null)
            {
                redLightFeedback.FlashLight();
            }

            ResetPlates();
        }
    }

```

- Sequence Checker: Make sure player presses the correct sequence of piano notes.
- Flashes red if by the final (5th) note, it is incorrect.
- Unlocks chest if all piano note indexes are correct.

Chests



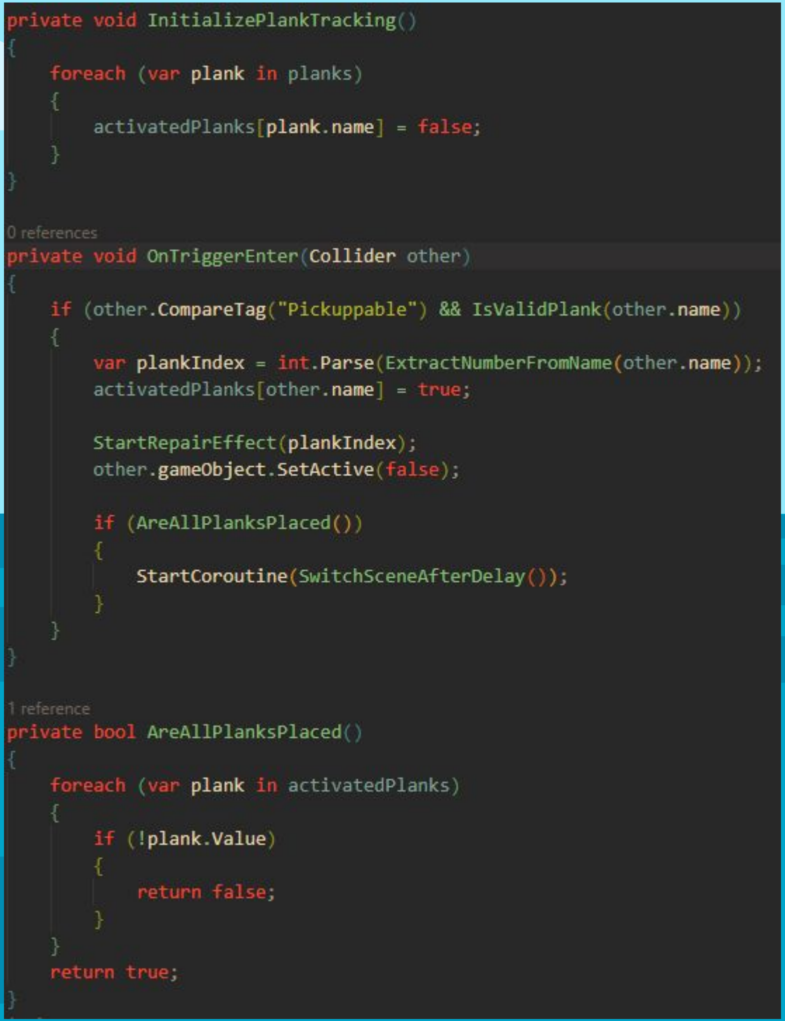
```csharp
private void OnInputValueChanged(int index, string value)
{
    if (string.IsNullOrEmpty(value))
    {
        if (index > 0)
        {
            _codeInputs[index - 1].Select();
            _codeInputs[index - 1].ActivateInputField();
        }
        return;
    }

    if (index < _codeInputs.Length - 1)
    {
        _codeInputs[index + 1].Select();
        _codeInputs[index + 1].ActivateInputField();
    }

    bool allFilled = true;
    int[] currentCode = new int[_codeInputs.Length];

    for (int i = 0; i < _codeInputs.Length; i++)
    {
        if (string.IsNullOrEmpty(_codeInputs[i].text))
        {
            allFilled = false;
            break;
        }
        currentCode[i] = int.Parse(_codeInputs[i].text);
    }

    if (allFilled)
    {
        CheckCode(currentCode);
    }
}
```



```csharp
public bool Interact(Interactor interactor)
{
    if (_hasBeenOpened)
    {
        return false;
    }

    if (_isUnlocked)
    {
        if(_cutScene != null){
            _cutScene.Play();
            StartCoroutine(DeactivateChestAfterCutscene());
        }
        else{
            _chest.SetActive(false);
            _rewardObject.SetActive(true);
        }
        ClosePanel();
        _hasBeenOpened = true;
        return true;
    }

    _isPanelOpen = !_isPanelOpen;
    _codePanel.SetActive(_isPanelOpen);

    if (_isPanelOpen)
    {
        _panelImage.color = _defaultColor;
        ClearAllInputs();
        _codeInputs[0].Select();
        _codeInputs[0].ActivateInputField();
    }

    return true;
}
```

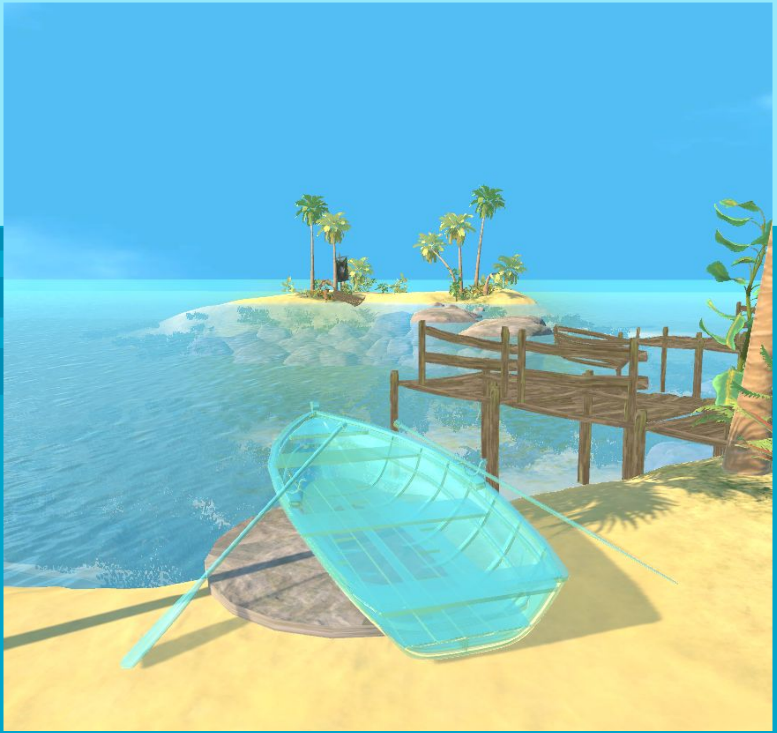
- Used Unity's UI canvas with TMP_Input to create chests which require a variable length code
- Panel is triggered by an interactor which is placed at the front of the character. Once you enter a code if it is correct, the chest will unlock. Otherwise it exits.

Boat Repair



```
private void InitializePlankTracking()
{
    foreach (var plank in planks)
    {
        activatedPlanks[plank.name] = false;
    }
}

0 references
private void OnTriggerEnter(Collider other)
{
    if (other.CompareTag("Pickuppable") && IsValidPlank(other.name))
    {
        var plankIndex = int.Parse(ExtractNumberFromName(other.name));
        activatedPlanks[other.name] = true;

        StartRepairEffect(plankIndex);
        other.gameObject.SetActive(false);

        if (AreAllPlanksPlaced())
        {
            StartCoroutine(SwitchSceneAfterDelay());
        }
    }
}

1 reference
private bool AreAllPlanksPlaced()
{
    foreach (var plank in activatedPlanks)
    {
        if (!plank.Value)
        {
            return false;
        }
    }
    return true;
}
```

- Repairing is triggered by previously mentioned pressure plates.
- Script has a set of planks, and when all plans are placed, the boat is fully repaired and cutscene is initiated.



Demo

Possible Future Improvements

User Engagement

Sound Effects, Point System, Time System to improve user engagement and make experience more enjoyable

WebGL Build

Have a WebGL build available for team member's portfolio

Appearance and aesthetics more visually appealing for players. Additionally adding custom assets to give more personalization

UI and Assets

Collision and general bugs need to be ironed out. Additionally detecting loopholes and and cheats will need to be more tested and prevented

Bug Fixes

Problems Encountered

Experience

Many of us did not have the proper technical experience. Testing and thorough research should be implemented for better results

Large Scope

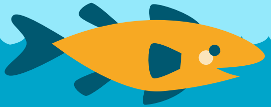
The ambitions of the team were too large for the given time frame.

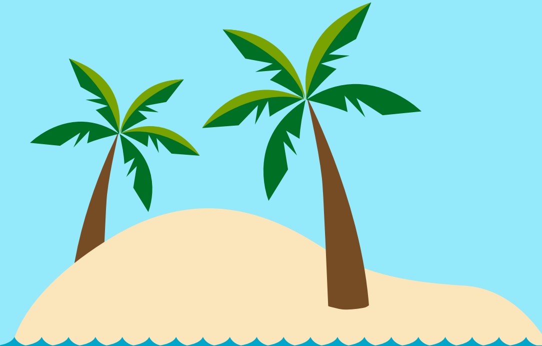
Merging Code

Numerous conflicts with the code base due to software version conflicts and a properly configured .gitignore.

Communication

Effective Communication and Better Time Management would help in producing more effective results.





Questions + Answers

Thanks!

Team Member Contributions

- **Noah Bowman** - Chest & Boat Repair implementation; conducted the gameplay demonstration.
- **Nisai Sun** - Project Overview & Tools Used
- **Kevin Nguyen** - Game Map, UI (Main Menu, Settings Menu, Pause Menu), Player Movement Mechanics, & Floral Cipher Level
- **Ousama Batais** - Piano Puzzle Implementation
- **Nikhil Maraboyina** - Pressure Plate and Gun Based Puzzle Development, Asset creation in Womp, Possible Future Improvements, and Problems Encountered.