Polynesian Seas

Jeffrey Popek

EDG-220

Project 2 Team 1

Sprint 4

Table of Contents

Delivery Platform

Our game, GAME NAME, will be played on any device that can use a mouse and keyboard and run unity games. Our game is mainly aimed at PC players.

Development Environment

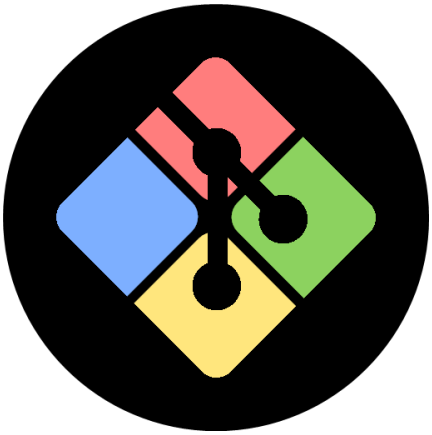
Unity

Our game was developed in the Unity Engine because of our team's experience with the engine and also the amount of documentation available online. Unity allows us to easily build our game and let a wide range of players play our game. Unity is also easily accessible for each role in our team. We are using Unity version **2021.3.17f**.

Git

Our team will use git for version control because it is supported by Redmine. Most of the team was somewhat experienced with using git from previous projects. Applications such as GitKraken or

GitBash will help simplify the version control process for less experienced git users.



Google Drive

Our team uses Google Drive to store and create documentation before finalizing it to be submitted to Redmine. Google Drive is easily accessible to everyone in the team and Champlain gives students a large amount of storage to use for projects. Google Drive also allows each team member to easily access and edit any existing documents for easy collaboration. We can also use redmine's macros to show our documents directly in the wiki.

Audacity

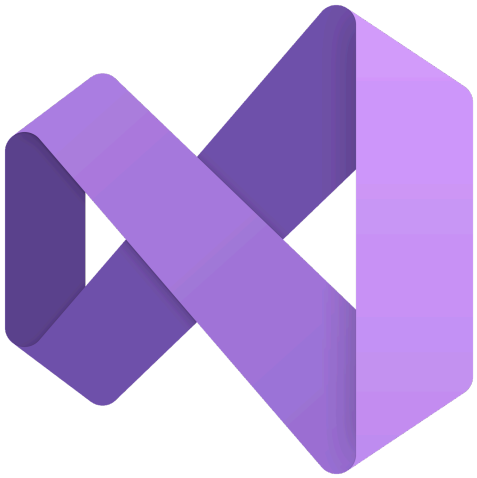
Since we do not have a sound designer for this project we are using Audacity to edit sounds. Audacity is the best option for our team because it is easily accessible and free.

Photoshop and Procreate

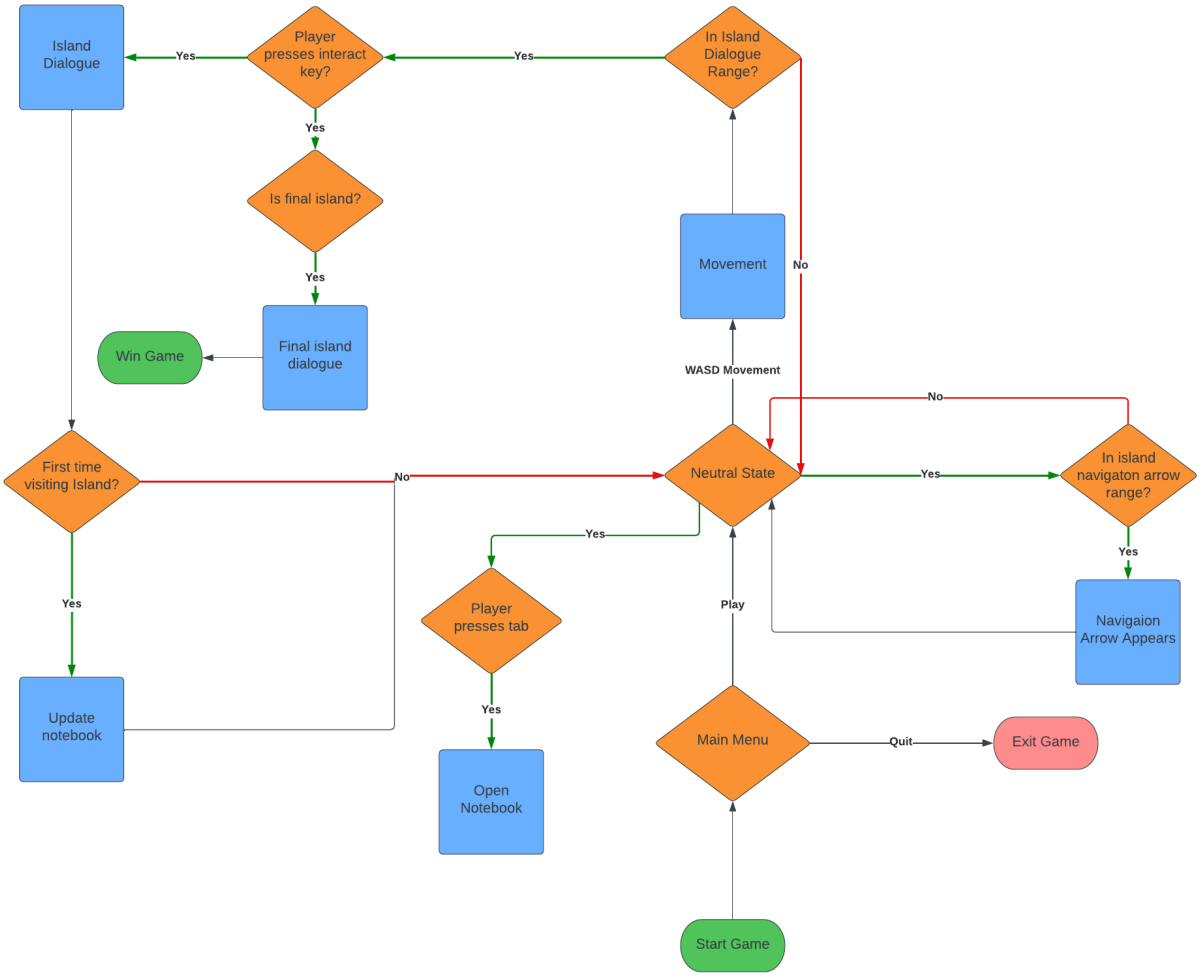
Our artists use Photoshop and Procreate to create and edit their art. Both of these programs help the artist make creating and implementing art as easy as possible.

Visual Studio

Our Programmer used visual studio 2022 to create scripts for the game. Visual studio is our best choice because it is easily accessible and worked well with unity.



Logical Flow Diagram



Game Mechanics and Systems

Menus

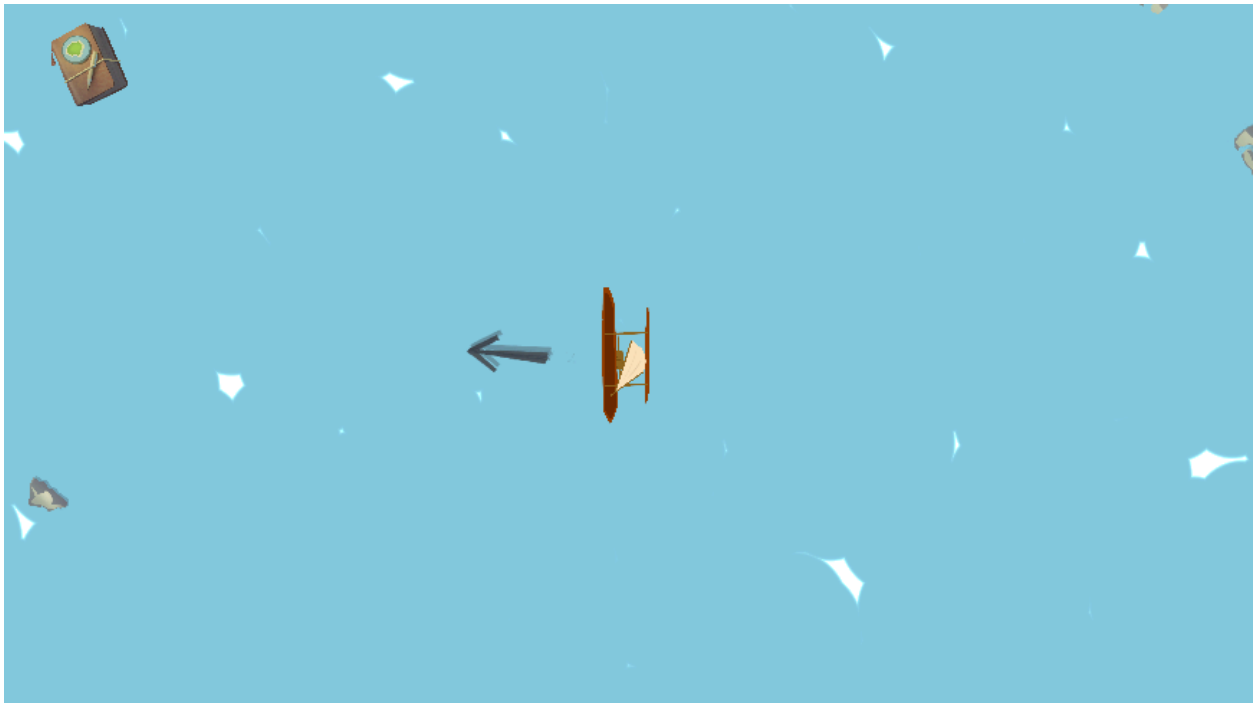
Our game does not center around menus so it would just be to make the game feel more cohesive. The main menu would be relatively simple and not have any mechanics. There would be the title of our games, a play button and a quit button. Each button will have a sound effect when clicking. The most difficult part of this would be creating the art assets for the menus.

Risk: Low

Movement

The movement of our game is a simple top down styled movement. The player can use the WASD keys to maneuver around the scene. The player's movement speed increases as they move until they reach their max speed. The player can increase their max speed through talking to a specific NPC (Fetu on Samoa). The player will rotate in the direction they are moving. The player will have an arrow to point in the direction of the next island. This arrow was implemented because of feedback regarding the difficulty of navigation. The arrow will appear only when the player is far enough away from any island. The most difficult part of movement would be getting the navigation arrow.

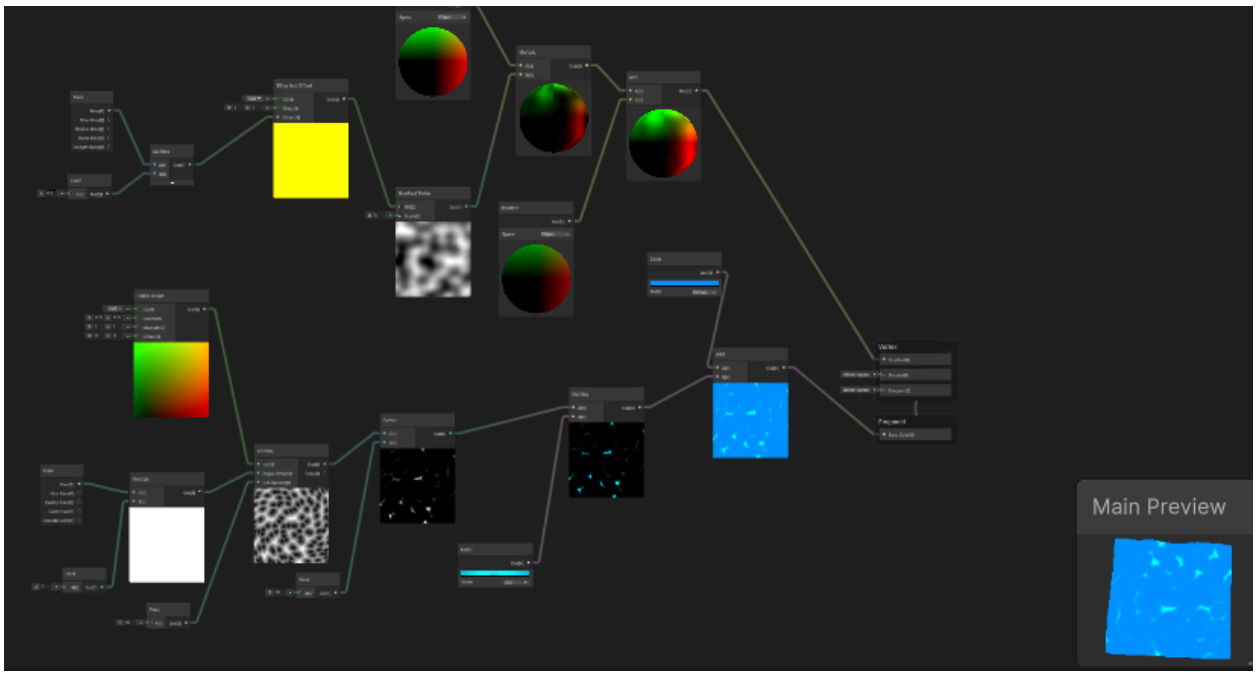
Risk: Low



Shaders

The setting of our game is on the ocean, so we naturally want to have an ocean background. We will be using unity's built in shader support such as URP to create a shader of the water. The most difficult part of this would be learning how to use it and implementing it into our game.

Risk: High



Dialogue

Our game is centered around navigating so getting directions from dialogue is essential. Dialogue is one of the core mechanics of our game so we wanted it to be flexible and easy to implement. You can respond with buttons in the UI. The player will be able to ask predetermined questions and give predetermined answers to the

NPC’s dialogue. The most difficult part of this would be programming it into unity.

Risk: Moderate

NPCs

Our game will have NPCs which are represented as islands around the map. When the player is within their radius an indicator will appear telling them they can talk to the NPC by pressing the interact key (E). When pressing the interact button the player will be frozen in place and start talking to an NPC. If the player has their notebook open then it will automatically close when entering dialogue with an NPC. The player cannot open their notebook while in dialogue. You can give responses to the NPC by clicking on response buttons. The most difficult part of implementing this feature would be the creation of art assets to fit the theme.

Risk: Moderate

Pipelines

Art Pipeline

Creating the Art

Our two artists will create their art in a default photoshop or Procreate template, it can be scaled down later if necessary.

Implementation

1. Have the art you want to add to the repo ready in the correct format. (.png)
2. Pull the git repo to make sure your project is up to date. If you have any git issues stop and contact the programmer unless you know exactly what the issue is.

3. Copy your file(s) into the project folder Assets/Sprites. There are a few different folders so pick one that fits the type of art you are adding (there is Environment, UI, and NPC).
4. Once the art is in you can commit and push to the repo.

Audio Pipeline

Finding Audio

Since we have no sound designer for this project we will find all our audio online. Anyone can look for audio and put it into the shared drive folder.

Editing Audio

Once a sound is found we will determine whether it needs to be edited or not. If so we will bring it into Audacity where we can edit it.

Implementation

1. Have the sound(s) you want to add to the repo ready in the correct format. (.wav)
2. Pull the git repo to make sure your project is up to date. If you have any git issues stop and contact the programmer unless you know exactly what the issue is.
3. Copy your file(s) into the project folder Assets/Sounds. There are a few different folders so pick one that fits the type of art you are adding (there is Environment, UI, and NPC).
4. Once the sound(s) are in you can commit and push to the repo.

Design Pipeline

Design Stage

First our designers will come up with an idea and get it written down. Next they will discuss with the team if the idea is in scope or not. If it is then they can begin to polish the idea and ask the artists or the programmer for any additional tools.

In Engine

We want the designers to have an easy time creating our map so we will be using prefabs. Prefabs allow our designers to drag and drop premade assets into the scene and still be able to change small aspects. The prefabs will be created by the programmers with the input from the designers and the artists.

Implementation

1. When you have an idea you want implemented you must first run it through the team first to see if it is possibly in scope for this project.
2. If in scope then the programmers and artists will begin to create assets. If the idea is out of scope you can scrap it or try and rework it to be in scope, repeat from step one when reworking your ideas.
3. Pull the git repo to make sure your project is up to date. If you have any git issues stop and contact the programmer unless you know exactly what the issue is.
4. Find out from your programmers and artists what tools and assets were created to implement your idea.
5. Create your design in the scene and save it as a prefab.
6. Commit and push to repo.

Sprint Updates

Sprint 1 - Game Concepts

Deliverables

- Game Build
    - A small prototype of our game.

- Visual Design Document
    - A Guide for our game's mechanics.

- Game Loop Flow Chart
    - Explains the flow of our game and how each action would lead to the next.

- Game Concept Document
    - Explains the concept of our game in detail.

- Art Concept Document
    - Six Possible art directions for our game to go in.

- Sound Concept Document
    - Explains what the sound of our game would be.

- Audience Development Document
    - Explains who our game will be targeted towards.

Goals for next Sprint

- Maintain documentation
- Have a working digital prototype to show the game's core mechanics.

Sprint 2 - Digital Prototype

Deliverables

- Game Build
    - A working prototype of our game showing off the core mechanics.

- Updated Documentation
    - Maintained each of our documents from the previous sprint.

- Finalized Documentation
    - Finalized documents that do not need to be iterated anymore.

- Visual Design Guide
    - Explains how our game will play out and what it takes to get something implemented.

- Art Document
    - Decided on one art direction to take our game in and put it into a document.

- Testing Plan
    - Team prepares to have our game tested in the GTL with documentation and a prototype.

Goals for next Sprint

- Maintain documentation.
- Continue working on digital prototype.
- Have our game tested at the GTL.

Sprint 3 - Continue Digital Prototype

Deliverables

- Game Build
    - A working prototype of our game showing off the core mechanics.

- Updated Documentation
    - Maintained each of our documents from the previous sprint.

- Finalized Documentation
    - Finalized documents that do not need to be iterated anymore.
- Implementing Art
    - Artists implement art into game.

Goals for next Sprint

- Maintain documentation.
- Have a completed digital prototype.
- Have our game tested at the GTL.

Sprint 4 - Complete Digital Prototype

Deliverables

- Game Build
    - A final version of our game with all mechanics completed and playable.

- Finalized Documentation
    - Maintained and completed each of our documents from the previous sprint.