CSCI 492 Final Project Report

Sitka Salmon

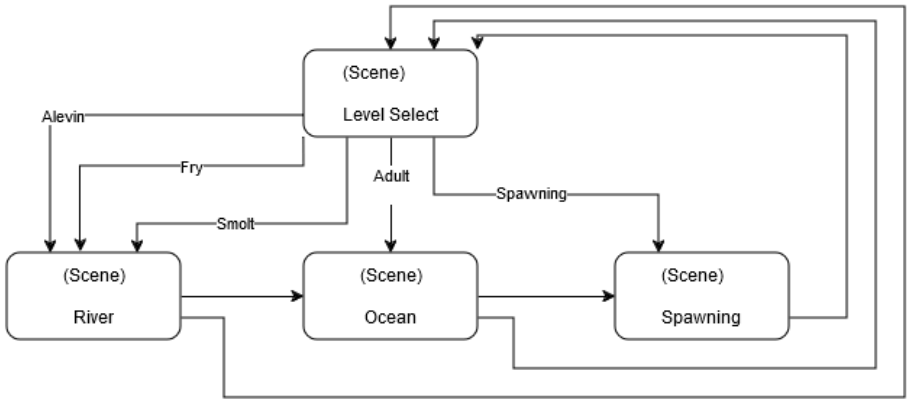
Mila Brooks
Steven Grubb
Wyatt Ayers

Introduction

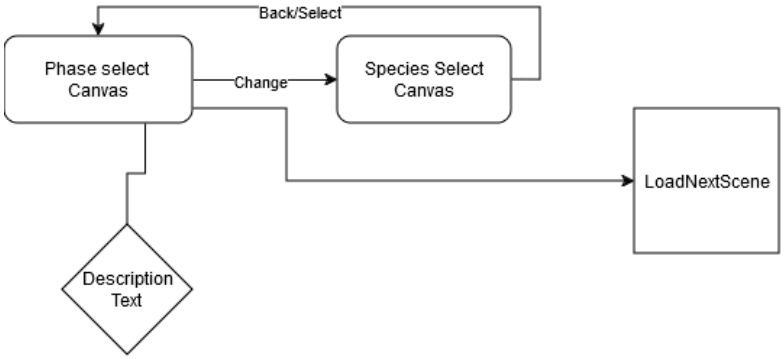
Our project involves designing and building a short computer game for the Sitka Sound Science Center. The motivation for creating this game is that the Science Center wishes to have an engaging way to teach people in the Sitka community and the diverse visitors that stop by the Science Center about the salmon life-cycle. Our contribution to this problem is an arcade-like game which features touch screen controls, three levels roughly corresponding with the three phases of salmon life, and game-play challenges modeled after those faced by salmon in reality.

Design

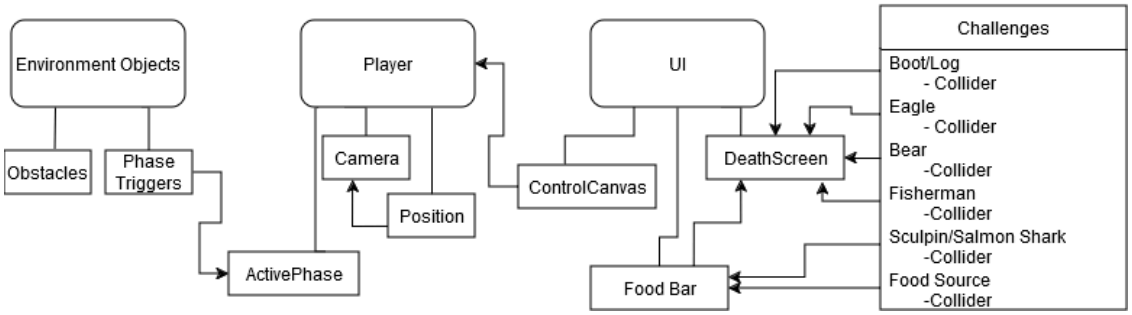
Scene
Navigation



Level Select Scene



Playable Scenes



By utilizing the Unity game engine and all the tools that come along with the Unity Editor, we were able to begin developing our game components without needing to implement the underlying technology. The main components of the engine that we used are GameObjects, a base class for all objects that we can expand upon, and the MonoBehavior, the base class for all scripts, that includes a framework for attaching the script to GameObjects, as well as hooks into core events such as Start and Update.

The unity manual describes all of the Functions and Events that are available to use from within the engine, as well as explanations as to how to effectively expand these objects. https://docs.unity3d.com/2022.3/Documentation/Manual/ScriptingImportantClasses.html

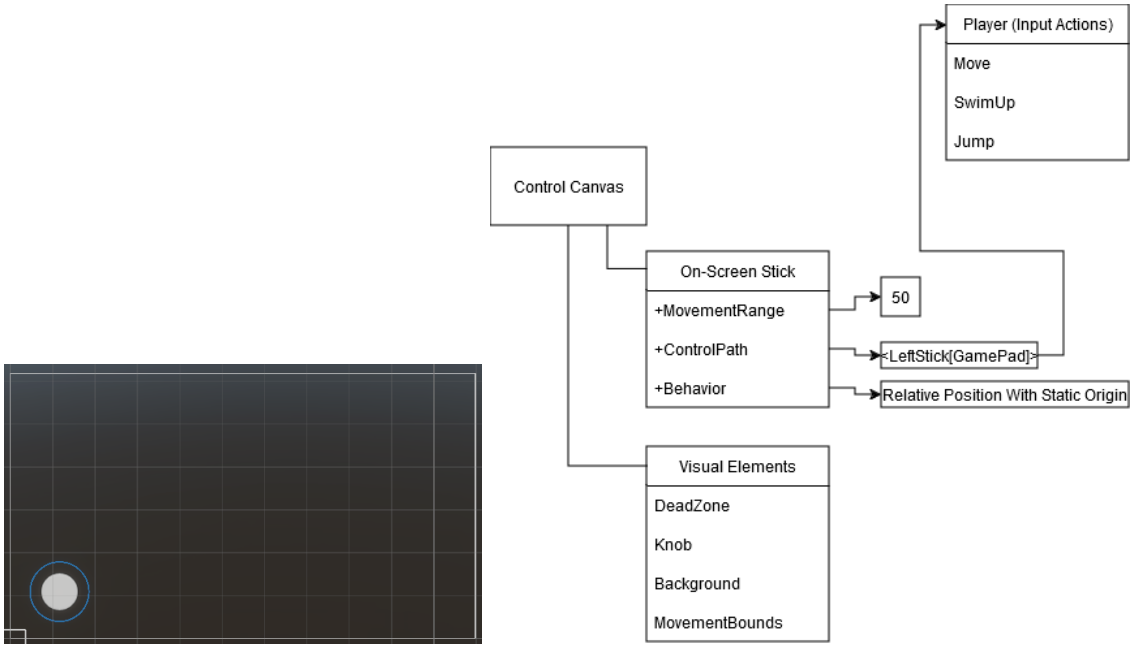
Our project makes great use of Inheritance; nearly every script is extending the MonoBehavior, and all of our objects extend GameObject.

User Stories

Sprint 1:

P0 – Player Controls

When the player is presented with the river game scene, there will be a ‘joystick’ in the bottom left of the screen that the player can drag around to control the fish sprite.

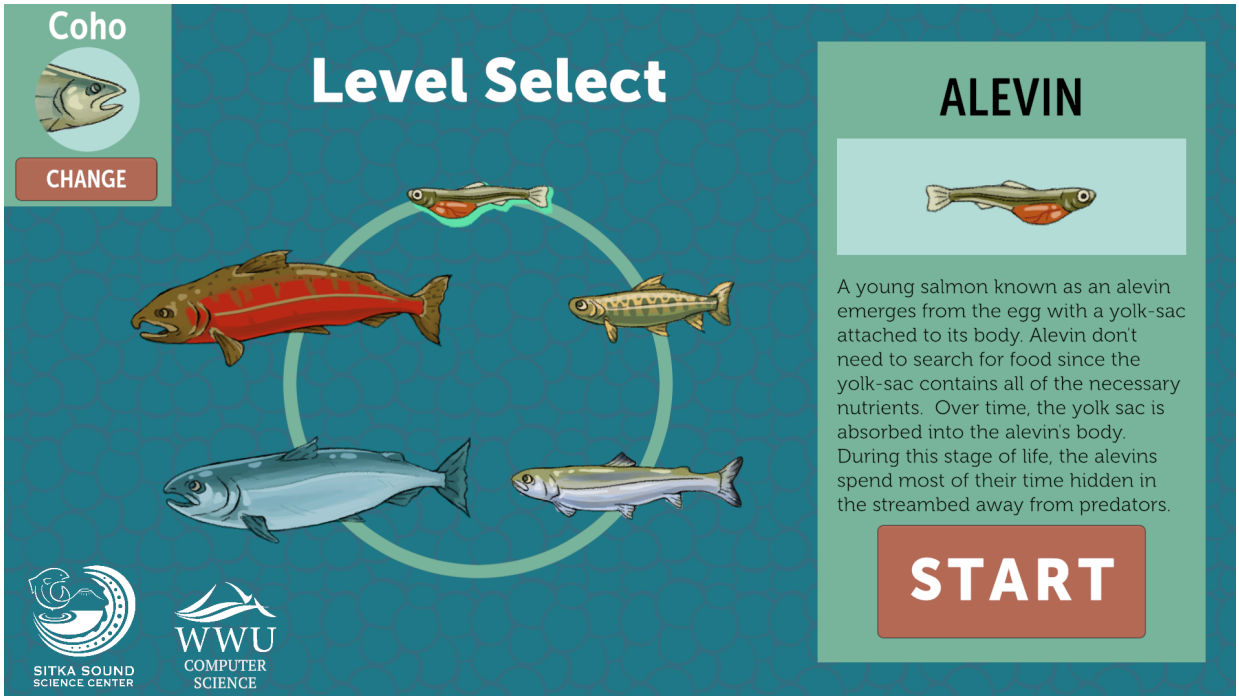


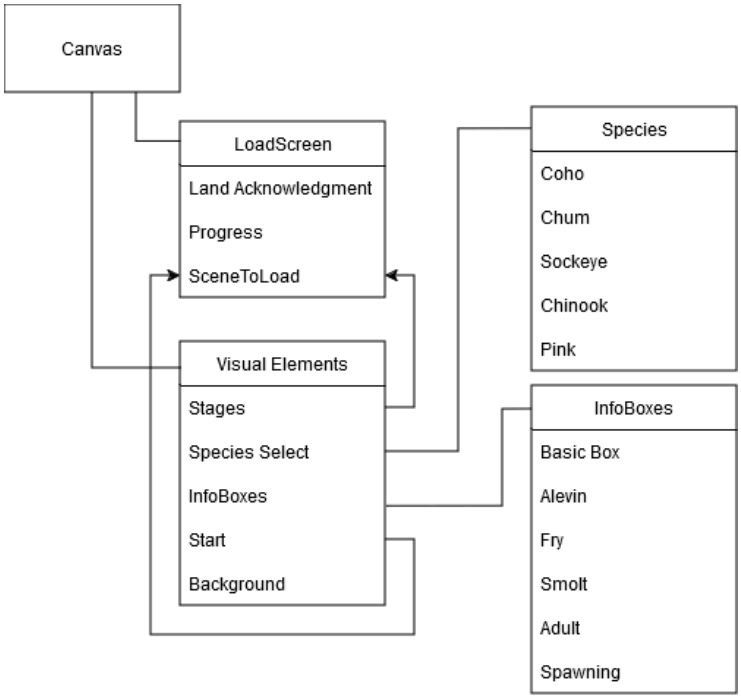
P0 – Landing Screen

When a player approaches the device they will be presented with a page to select the species and lifecycle phase that they wish to play, and then press a start button to begin the game

P1 – Species Select Screen

The initial requirement for the Species select screen was just the three Salmon relevant to SE Alaska, However we were able to implement two additional species, Sockeye and Chinook





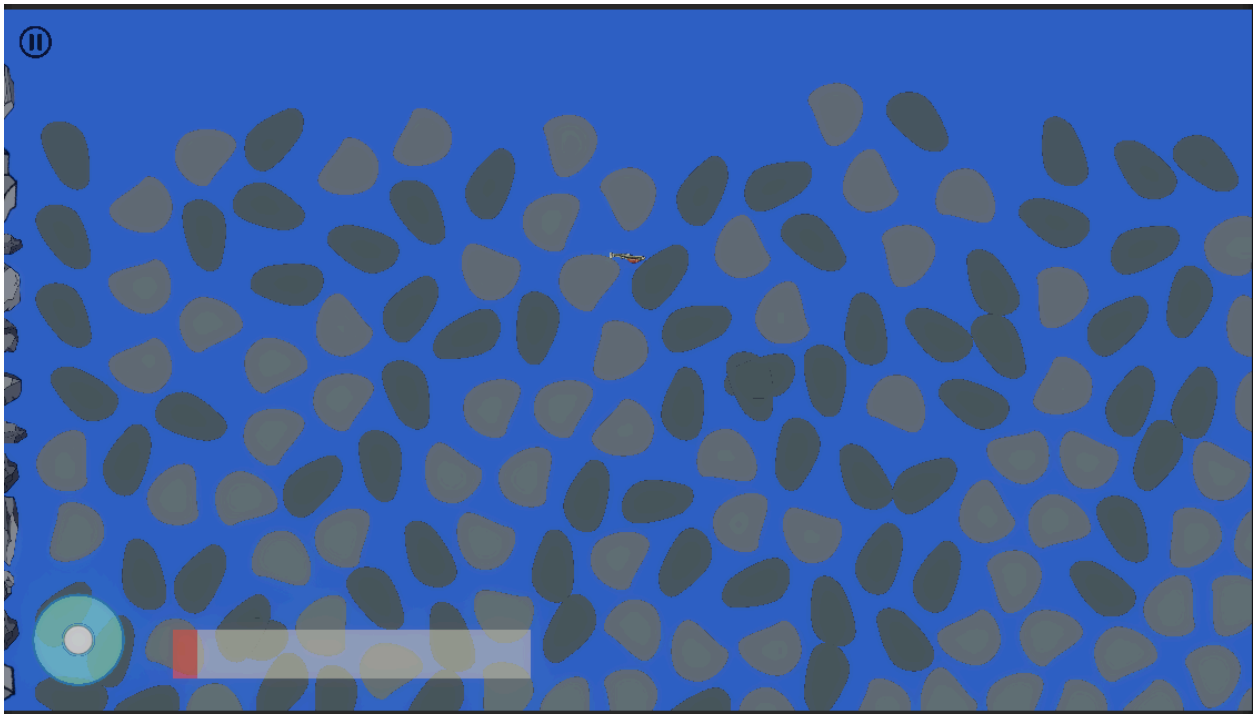
P0 – Tutorial

Upon starting a game level, the player will be presented with a pop up text box instructing the player how to use the joystick and pause menu.

Sprint 2:

P0 – River Level

Upon selection of the River level, the player will be placed at the beginning of a river environment where they will control a fish sprite with the goal of making it to the ocean. The player will need to consume an amount of food to progress to the next lifecycle phase. The player will also be tasked with avoiding predators such as larger fish and eagles that desire to eat the player.



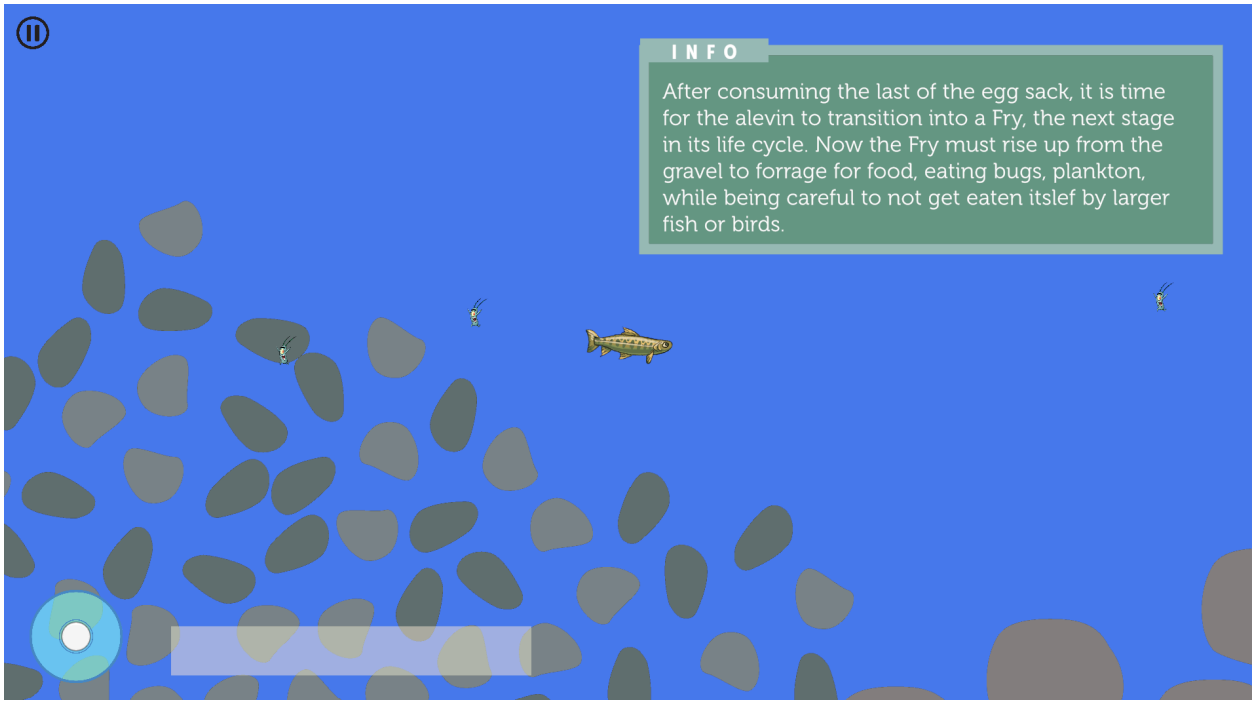
The Diagram ‘playable scenes’ in the Design Section describes the design of this component well.

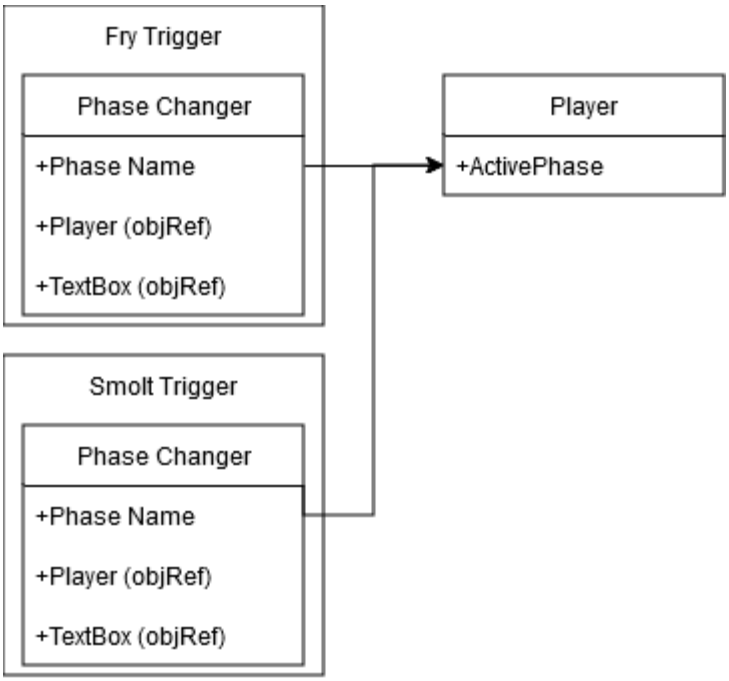
P0 – Unit Testing Framework

As developers, whenever we have created a new component, and its associated tests, we will be able to run those tests from within the unity editor to check if our code is behaving as expected.
Design Doc: Implemented the Unity Test runner

P0 – Phase Transitions in River level

As the player is consuming food in the river level, once they have reached some threshold to be determined by playtesting, the player sprite will update to the next phase, and the player will be presented with a text box describing the relevance to the salmon life cycle.





P1 – Target Machine Deployment

While we are onsite on Nov.8th we will test the game on the deployment hardware, installing a demo build onto the target machine for a demo during our presentation. This did not go as planned. The plan was to install Ubuntu LTS onto the target machine, however, after installing Ubuntu and testing the game the touch screen components did not work. This is due to a driver issue, where touch inputs are handled differently on windows vs linux, so we had to change plans to deploy to windows instead, however due to windows licensing we need the Science centers tech dept, AlasConnect to install the

OS for us. So instead of leaving AK with the target machine configured, we are still waiting on that machine to get set up.

Sprint 3:

P1 – Egg Phase

For this section of the game the player will be presented with a prerecorded or choreographed scene that conveys the fragility of the salmon Redd, and the connection between the spawning salmon swimming up river to lay their eggs, and the next generation of Alevin.

– Not Done

P1 – Set Dressing

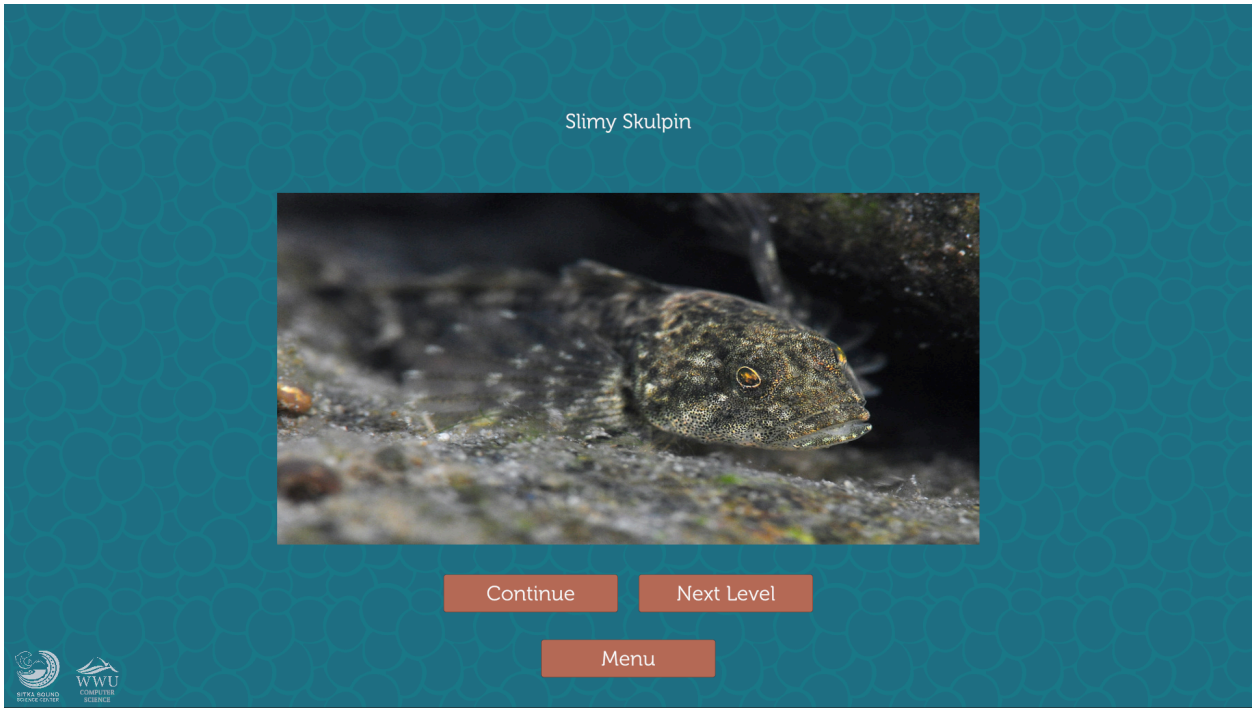
Add artwork and assets to the River level so that the user is more immersed in the ecosystem portrayed by the game environment. These assets will mirror the Sitka Sound river systems.

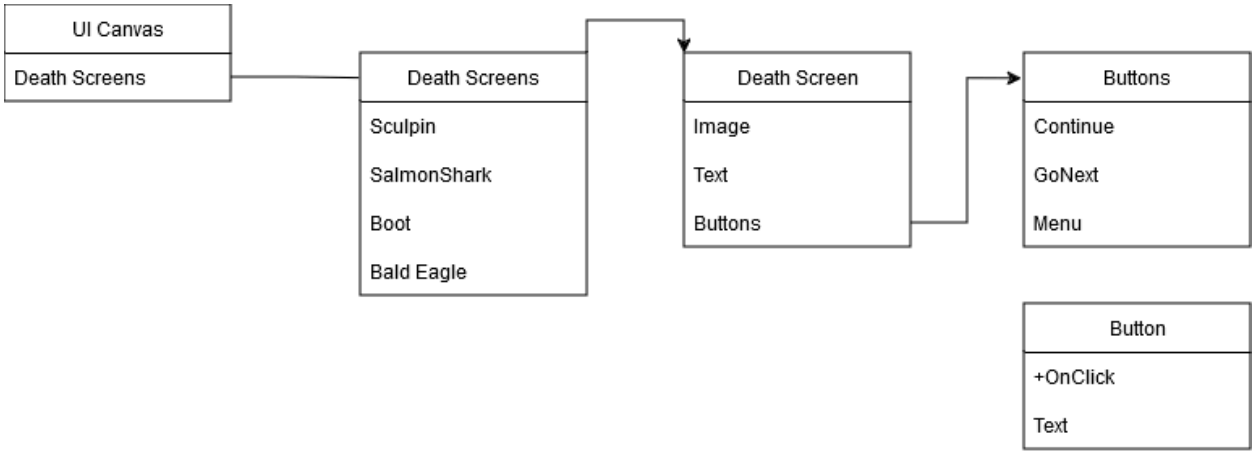
– Not done, Waiting on assets from our client.

P0 – Navigation and Death Screen

Upon a death from any cause, or a successful completion of a level, the player will be presented with a screen that offers the choice of either continuing to the next lifecycle phase, or restarting the current phase, or returning to the phase select scene.

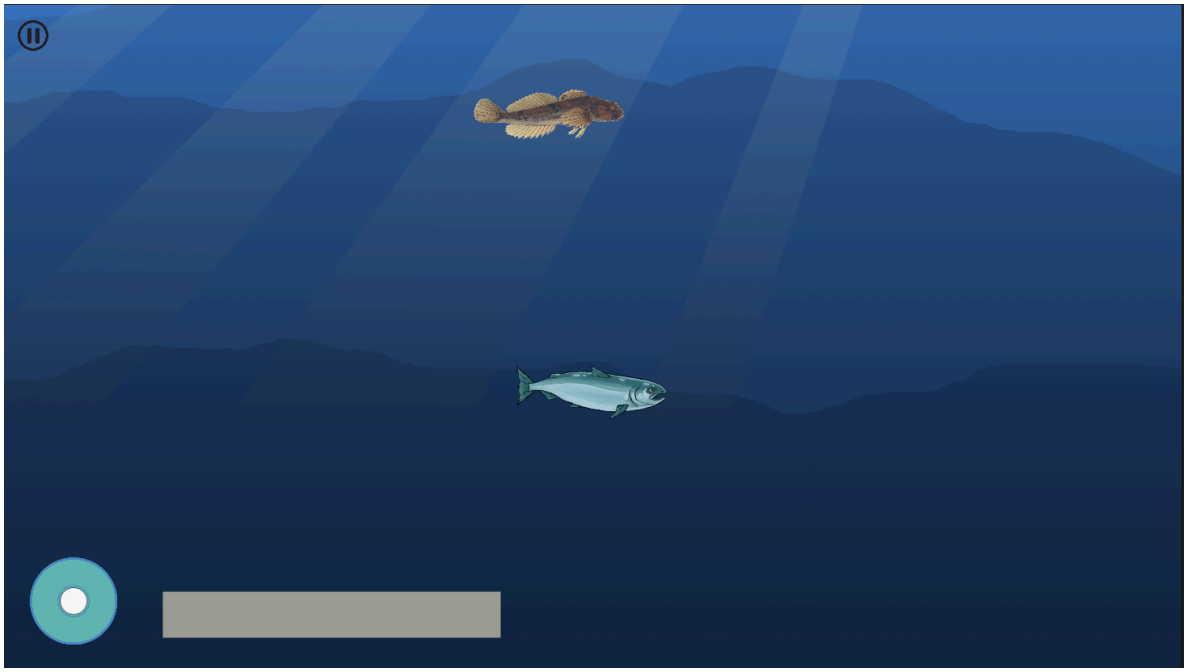
The various ways salmon die should be described more completely and prominently shown to visitors so we can communicate the importance of salmon to a broader ecosystem.





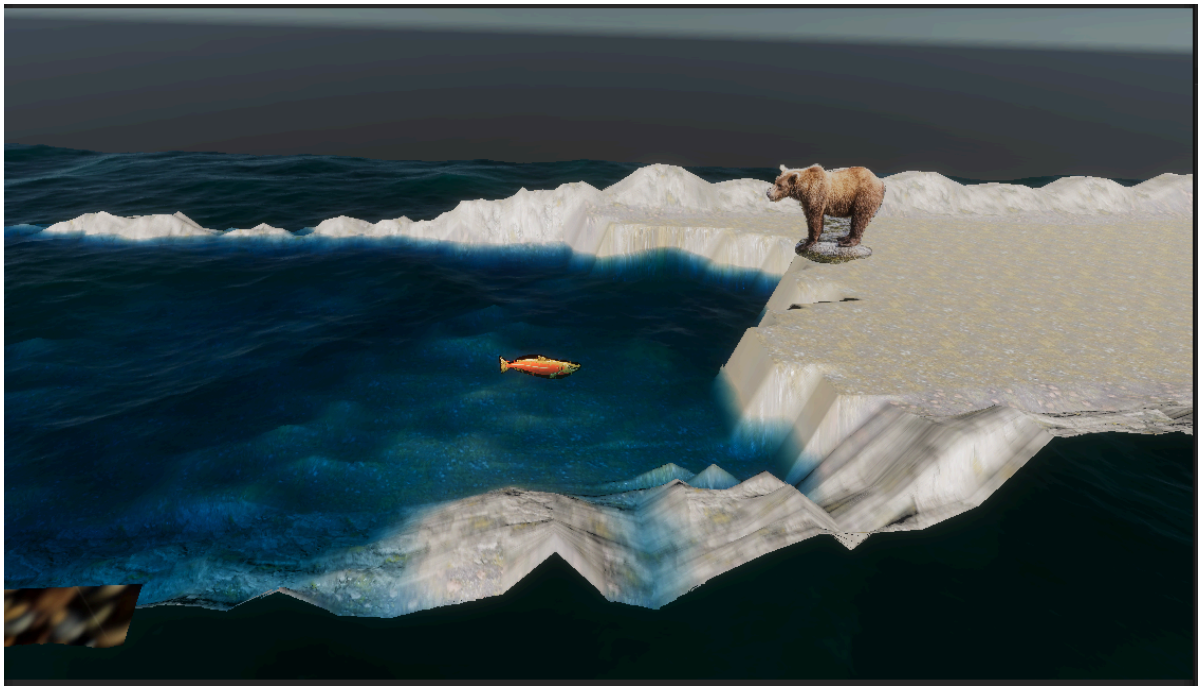
Prototype Ocean Level

Create a prototype of the ocean level that has basic movement and some white boxed environmental challenges.



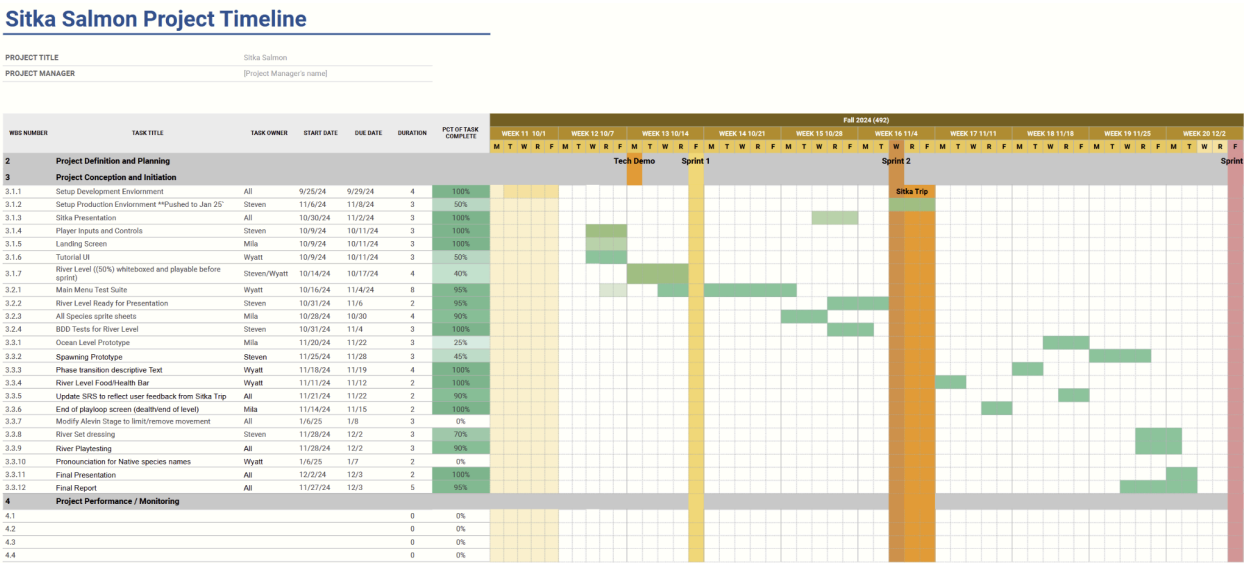
Prototype Spawning Level

Create a prototype of the Spawning level that has basic movement and some white boxed environmental challenges.



The Diagram ‘playable scenes’ in the Design Section describes the design of this component well for both of the above stories.

Project Management



Sitka Salmon Project Timeline

PROJECT TITLE	Sitka Salmon
PROJECT MANAGER	[Project Manager's name]

WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
2	**Project Definition and Planning**					
3	**Project Conception and Initiation**					
3.1.1	Setup Development Enviornment	All	9/25/24	9/29/24	4	100%
3.1.2	Setup Production Enviornment **Pushed to Jan 25`	Steven	11/6/24	11/8/24	3	50%
3.1.3	Sitka Presentation	All	10/30/24	11/2/24	3	100%
3.1.4	Player Inputs and Controls	Steven	10/9/24	10/11/24	3	100%
3.1.5	Landing Screen	Mila	10/9/24	10/11/24	3	100%
3.1.6	Tutorial UI	Wyatt	10/9/24	10/11/24	3	50%
3.1.7	River Level ((50%) whiteboxed and playable before sprint)	Steven/Wyatt	10/14/24	10/17/24	4	40%
3.2.1	Main Menu Test Suite	Wyatt	10/16/24	11/4/24	8	95%
3.2.2	River Level Ready for Presentation	Steven	10/31/24	11/6	2	95%
3.2.3	All Species sprite sheets	Mila	10/28/24	10/30	4	90%
3.2.4	BDD Tests for River Level	Steven	10/31/24	11/4	3	100%
3.3.1	Ocean Level Prototype	Mila	11/20/24	11/22	3	25%
3.3.2	Spawning Prototype	Steven	11/25/24	11/28	3	45%
3.3.3	Phase transition descriptive Text	Wyatt	11/18/24	11/19	4	100%
3.3.4	River Level Food/Health Bar	Wyatt	11/11/24	11/12	2	100%
3.3.5	Update SRS to reflect user feedback from Sitka Trip	All	11/21/24	11/22	2	90%
3.3.6	End of playloop screen (death/end of level)	Mila	11/14/24	11/15	2	100%
3.3.7	Modify Alevin Stage to limit/remove movement	All	1/6/25	1/8	3	0%
3.3.8	River Set dressing	Steven	11/28/24	12/2	3	70%
3.3.9	River Playtesting	All	11/28/24	12/2	3	90%
3.3.10	Pronounciation for Native species names	Wyatt	1/6/25	1/7	2	0%
3.3.11	Final Presentation	All	12/2/24	12/3	2	100%
3.3.12	Final Report	All	11/27/24	12/3	5	95%
4	**Project Performance / Monitoring**					
4.1					0	0%
4.2					0	0%
4.3					0	0%
4.4					0	0%

We used a spreadsheet within our shared google workspace to track project goals. Modeled after gantt chart, with task names, assignments, schedule, and completion tracking.

We tracked all of our todo items using the Gantt chart above and using GitHub issues. We began sprints by selecting a list of things to build and which stories we would prioritize. We then add these tasks and an approximate timeline for their completion to the Gantt chart. A corresponding GitHub issue was added describing the user story and what we would build to complete the story. For example, when we were working on adding the death screens to our game, we started with an “epic” issue for the user story. This issue referred generally to our idea of a death screen. There were, of course, many subproblems that needed to be solved within this “epic”, which we tracked in their own separate issues.

Source Code

https://github.com/Milaeris/Sitka-Salmon

Testing

Conveniently, there is a Unity specific test framework and system inside of the Unity Editor. Named the Unity Test Framework, this operates in the test runner component of Unity. This system gives us the ability to do two modes of testing. The first being Edit Mode tests which are static tests that run in edit mode. These can test static values in a scene such as whether or not the joystick object is active in the scene and being displayed. The second is Play Mode tests

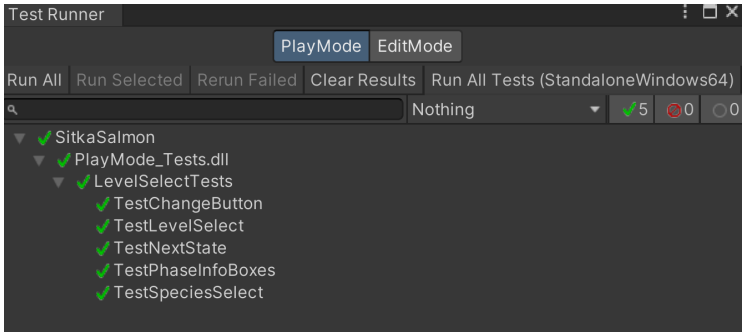
which run as if the game were being played. With these kinds of tests, we can test if a UI element updates game state correctly, or if player collision with an enemy collider has the appropriate action performed depending on the game state at the moment of collision. For example, enemy interaction has a different set of actions performed depending on whether or not the player's health bar is empty. The Unity Test Runner provides us a nice UI to view our Edit or Play mode test suites and an easy way to run all tests of a subset of the tests and quickly view the results.

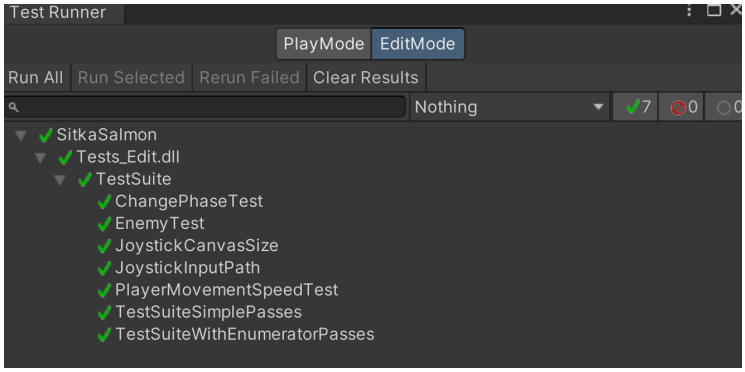
TDD

One of our Play Mode test suites performs UI testing for our main menu scene. This is done by having the test code generate an instance of the main menu as if it were being played. From here we are able to generate test button clicks and touch inputs to test if the UI is properly handling these inputs and making the correct game state modifications. One test in particular is for the fish buttons in the circle at the center. When touched or clicked, these generate OnClick events. These events fire a handler which will update the variable determining which of the three levels to load into. In this case, the test operated by generating one OnClick event for each of the 5 phases currently on the menu then comparing the value in the scene to load variable with the expectation.
https://github.com/Milaeris/Sitka-Salmon/blob/main/Assets/PlayMode_Tests/LevelSelectTests.cs

One of our Edit Mode Tests ensures that our player GameObject is configured correctly. Similarly to the menu testing, it firsts generates an instance of our player object. It ensures that the movement speed is set and correct, and the prefab is able to be found and initialized. It also tests our ChangePhase code, and makes sure that both sets the correct fields and disables the correct child GameObject.

https://github.com/Milaeris/Sitka-Salmon/blob/main/Assets/Tests_Edit/TestSuite.cs





Test Coverage

Describe how you determined test coverage. What is your coverage percentage?
Our test coverage is not the greatest, Definitely the weakest aspect of our project at this time. We do have runtime tests for the entirety of the Level Select scene, and static tests for the player movement and control systems, so two of our key components have automated testing to ~%90 coverage, however other main components are lacking in code testing. We did develop much of the game based on our Gherkin tests, however that testing is not automated, it is all done by us at playtime. Another struggle with testing this project is that a decent chunk of testing cannot be automated, as much of the gameplay is judged by ‘feel’ when trying out how the mechanics work, we implement a component and then play the game to test the parameters and then tweak, play, repeat, until the component feels good to us.

BDD

Below is a link to our Gherkin tests. We would write a test before implementing some behavior, and then use that test to make sure that the action that the player would do matched the result defined by the test.
https://github.com/Milaeris/Sitka-Salmon/blob/main/GherkinTests.md

Performance Considerations

Because our target machine is an i5 Nuc there is not a ton of computation available, so to minimise the volume of physics calculations at runtime there is a bounding box just larger than the camera size that enables and disables the physics properties of the dynamic rocks in the river level so that only the rocks on screen, and a little bit ahead of where the player is going are actively having calculations computed.

Summary

The most significant challenge was by far learning to use the Unity Game Engine. When you first open Unity, the default layout presents a lot of information immediately, which is overwhelming at first glance. Then, to begin doing things during gameplay, we had to become familiar with the Unity Scripting API. The API is robust and sufficiently documented. This made our development process challenging at first, but as we improved our workflow over the first few weeks of the quarter, we were able to quickly make larger systems for the game. The top-down approach of our Agile sprints was the perfect environment for learning how to use an industry standard tool for a project like this. Beginning at the highest level of “as a user, I want to do X”, then proceeding down to the technical level of what components would combine to achieve satisfaction of the story. Even lower was identifying within each component, what tools provided by the Unity API are necessary to fulfill the functionality that this feature requires.

Aside from learning Unity and the various technical challenges that are associated with that, it was also tricky to balance learning about salmon, their lifecycle, and their significant role in the ecosystem along with learning how to design a video game while using a new development tool. We learned throughout this process that it is not trivial to design an educational game. The question we have asked the most is how do we present the necessary educational content in a way that is both engaging and informative.


PINE - LIFE CYCLE
MOSS - LIFE CYCLE
Game Screens - Level/Phase Select
Level Select
ALEVIN
Level Select
ADULT
START