Ocean Simulator: Iteration 3

https://github.com/i12345/ocean-simulator/commit/c66a1a8c6b1573688c9a33a378faa0cb58b82449

Project Members: Shaina Ayer, Christian Blundell, Isaac Valdez

Vision:

The ocean simulator will be a video game that lets players virtually "swim" in an ocean with marine life using motion-control.

Target Users:

The target users will be an adolescent-focused group that enjoys things such as new technologies, games, science, animals, exercise, and exploration. A sample of this target audience is Jacob, a family member of group member Isaac Valdez, who falls into this target user category and is used for feedback. Jacob wants a simulator game that runs smoothly, has realistic graphics, and extensive gameplay.

Competitors:

Similar video games and interactive projects featuring marine life already exist, such as Ultimate Ocean Simulator, Ocean Mammals: Blue Whale Mari, Fish Abyss: Aquarium Simulator, and David Attenborough's Great Barrier Reef. They have many kinds of animals, and Ultimate Ocean Simulator lets the player "play" as one of the marine animals. David Attenborough's Great Barrier Reef has many educational videos and articles as well. Besides these, there are many non-interactive underwater experiences available on the Internet, such as regular and 3D videos. However, these games lack motion-controlled input.

Move! lets a person play a video game with motion-control, but it doesn't feature the ocean.

Requirements:

The major requirement for this project is to develop a motion control for the user to navigate through the simulator, with the user's webcam detecting movement and passing it out as movement in the simulation. The other major requirement is to design 3D models of marine life that can be discovered by the user while moving in the simulation. The marine life should have simple AI for generation and recognition of the type of marine life generated. Other requirements include generating AI patterns for certain types of marine life (schools of fish, shark behaviors, rarity of marine life) and dynamic generation of marine life. If possible, other requirements could include developing more in-depth patterns for certain types of marine life based on their natural behavior as well as fluidity of the water. Requirements will not include active interaction with marine life, or any simulation based outside of an ocean/sea environment.

Core Features:

- Navigation between main menu, "set up camera", and playing in the simulation
- 3D model / rendering for animals and diver
- Motion-control for the diver
- AI controls the animals' behavior

Project Plan:

The plan is to first develop motion control for the diver. This will be done with MediaPipe BlazePose for pose detection and PlayCanvas for 3D graphics and physics. The motion control foundation has been developed and tested during this iteration. We are developing this to work on a webcam, and we expect this simulator to work on any device that has access to a webcam. There may not be the option to develop a simulator that is connected via screencast, which is shown by the limited users in the target audience. This should take weeks to complete, finishing at iteration 3. The 3D rendering of the fish has also already begun, with 3D models of fish collected to be implemented into the simulator. We plan on getting more 3D models of various types of marine life to be generated (potentially dynamically) into the simulator. We anticipate that this will take a few weeks until iteration 3. The other requirements will be based on the completion of the previously mentioned steps and will take less time to implement. The simulator starting screen and user control setup will be developed during the third iteration. We anticipate 30 more hours of implementation of core features for each iteration, including testing.

Core Features I/O Table:

Input	**Output**
User: Enters Home Screen	**Program: Begins calibration for motion control**
Webcam: User moves using motion control	**Program: Defines movement to mirror movement of simulator**
User: Selects marine life found in simulator	**Simulator: Identifies fish based on value defined based on its 3D model**
Program: Generates marine life	**User: Sees marine life in simulated environment**

Domain Model:

The domain model shows the scope of each element of the Ocean Simulator. These elements include the motion control figure that guides through the simulator, the player pose that defines the motion control of the game, and the creatures that are generated using simple AI models into the simulator.

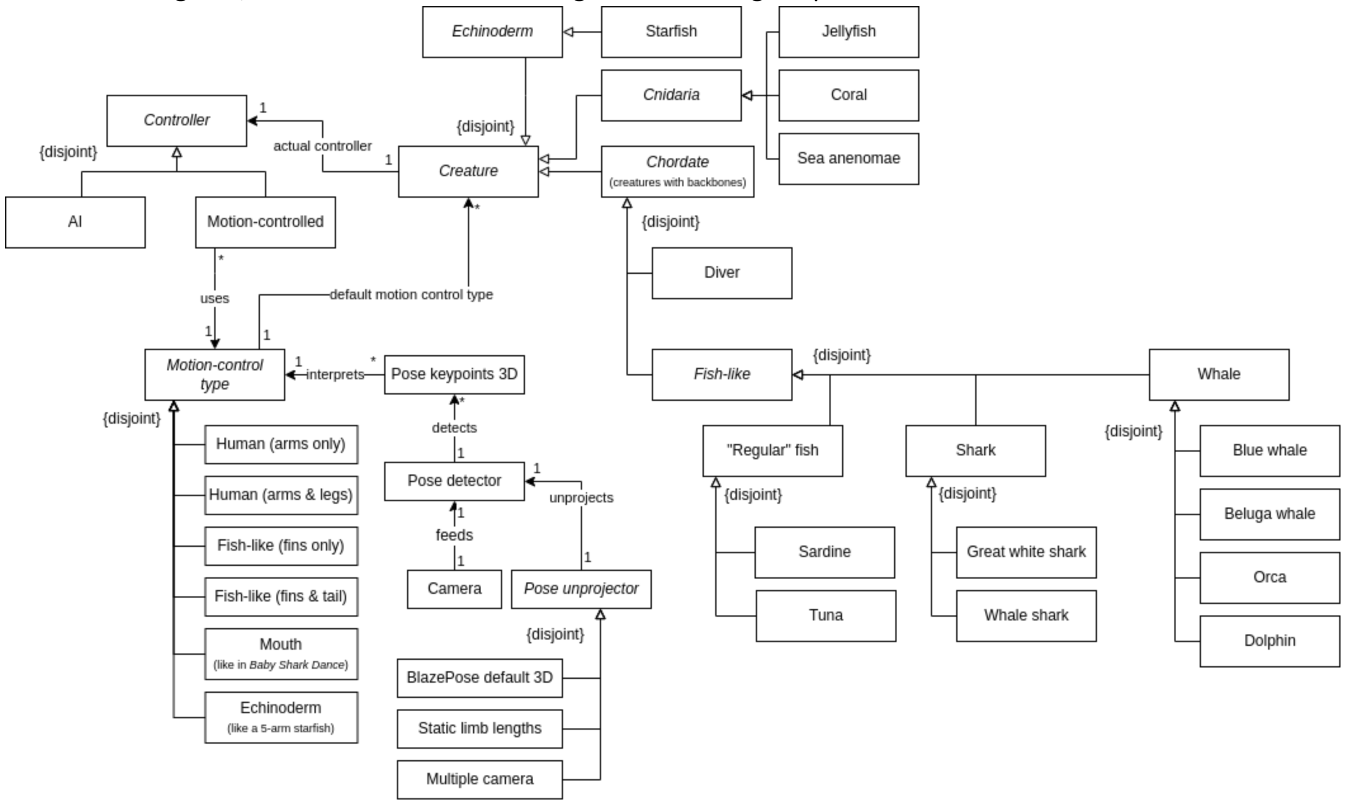

Figure 1: Domain Model for Ocean Simulator

Use Cases:

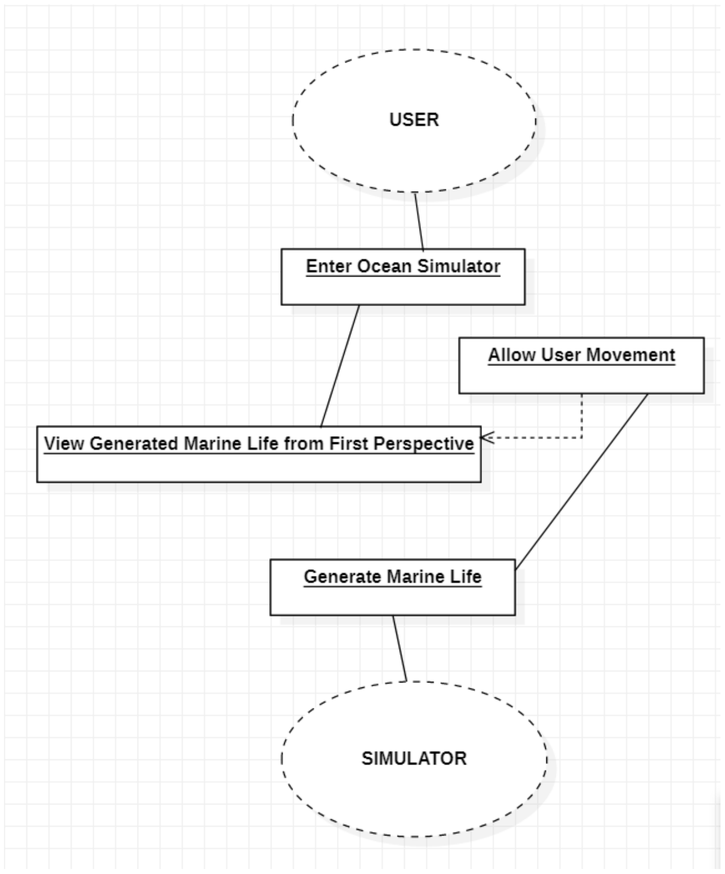

Figure 1: Use Case for General Ocean Simulator Project

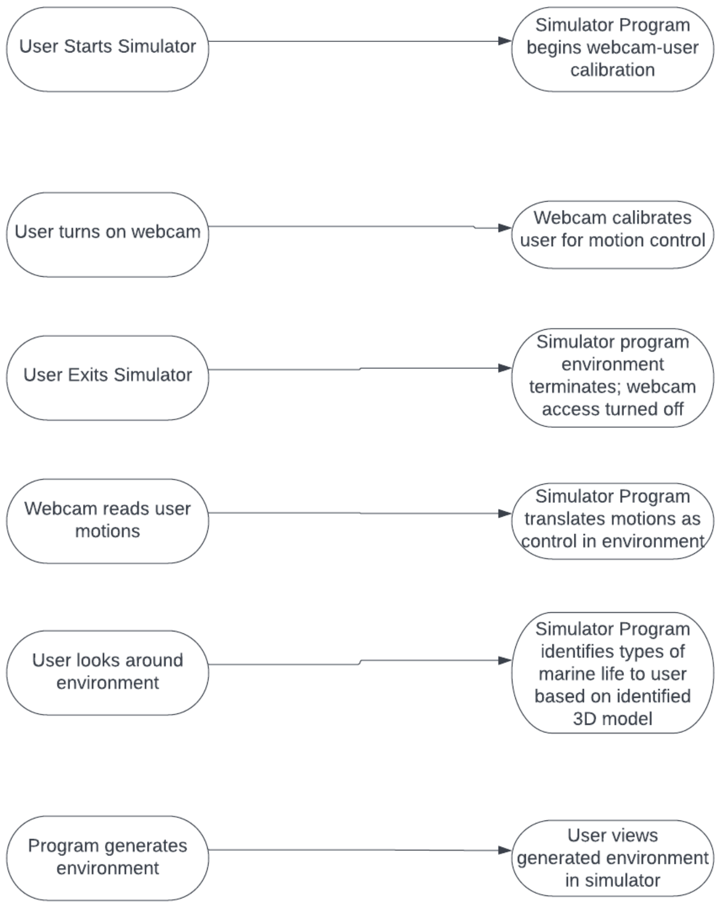
User Starts Simulator
Simulator Program begins webcam-user calibration
User turns on webcam
Webcam calibrates user for motion control
User Exits Simulator
Simulator program environment terminates; webcam access turned off
Webcam reads user motions
Simulator Program translates motions as control in environment
User looks around environment
Simulator Program identifies types of marine life to user based on identified 3D model
Program generates environment
User views generated environment in simulator

Figure 2: Use Cases for each aspect of simulator setup/gameplay

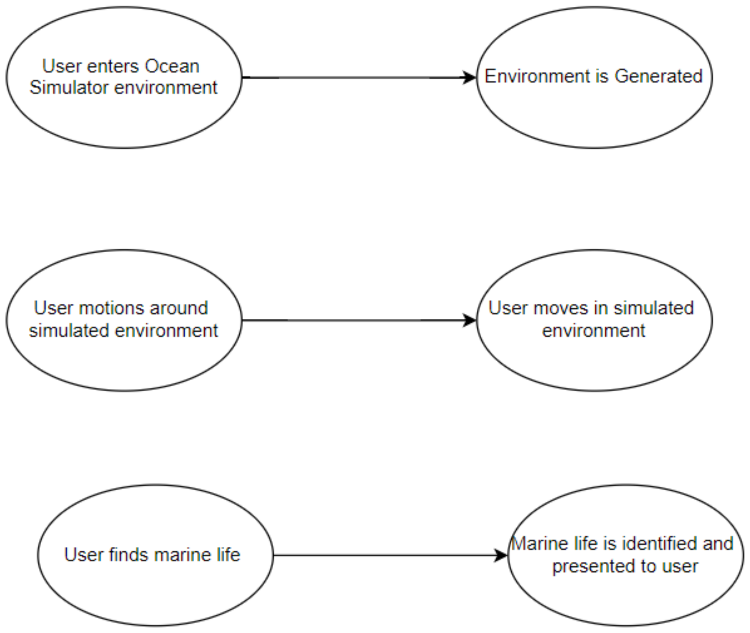
User enters Ocean Simulator environment
Environment is Generated
User motions around simulated environment
User moves in simulated environment
User finds marine life
Marine life is identified and presented to user

Figure 3: Use Cases for Ocean Simulator Gameplay

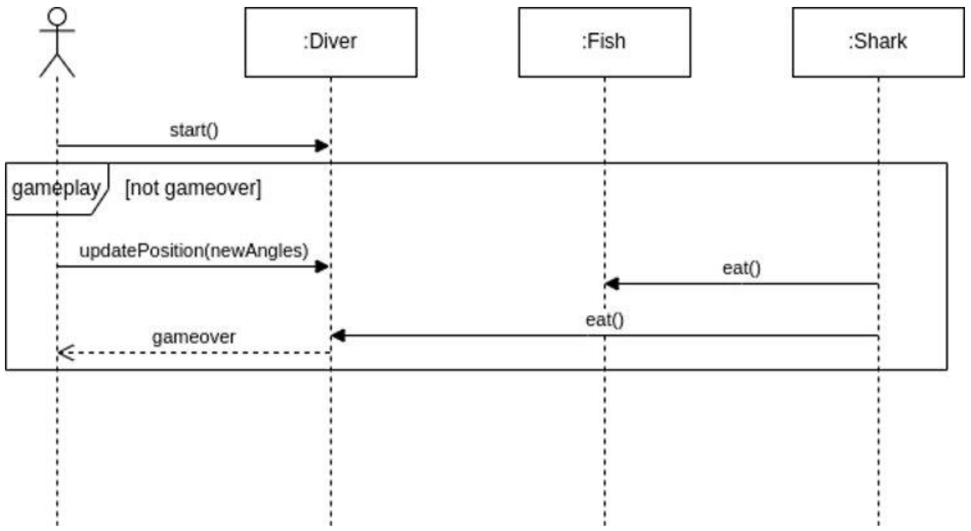

Figure 4: System Sequence Diagram for Ocean Simulator Gameplay

Risks and Risk Mitigation:

- **TV/Large monitor implementation: 100% x remaining project hours = Effect on total project (15 hr)**
    - TV/Monitor implementation difficult to achieve, only able to test and develop on single monitor setups.

    - Market Study: 20 people that fit target audience demographic took market study; results confirmed that they do not have a setup that justifies the implementation of large monitor device for motion control
    - Mitigation: We will implement and test this project using a single webcam setup on a computer as it is the most available for our target audience. We will potentially develop a simple prototype for multi-device setups that work on all types of large monitors/TV that support screen casting. (We will not make large monitors a must-have requirement for our project's final delivery)
- **Unrealistic Schedule: 80% x remaining project hours (15 hr) = 12 hours**
    - The time frame for each iteration may not be enough to complete each feature to the highest quality anticipated. This will result in a potentially incomplete product for the final delivery.
    - Mitigation: Schedule each feature for each iteration properly and set realistic goals for each feature. In the project plan, we have anticipated that we will complete the most necessary features by iteration 3 so that the product will have the minimum functionalities we anticipate demonstrating.
- **Failure in Pose Detection Functionality: 60% x remaining project hours (15 hr) = 9 hours**
    - If pose detection technology is too technical and abstract to implement, core features of the product will not be available, leaving an unpolished product as the resulting effect.
    - Mitigation: Analyze pose detection and implementation in earlier iterations to ensure that pose detection remains a key part of the product. (Currently working on pose detection for this iteration)

Current Progress:

We have a foundation for the motion control diver, as we have set up the code and test cases for the math to decompose a set of key points (e.g., shoulder, elbow, etc.) into the corresponding rotations of limbs. The code uses MediaPipe BlazePose to detect poses.

However, this code is currently not running in the demo. To make the diver able to swim and interact with the rest of the world, the code is currently being modified to use the Ammo/Bullet physics engine. The hinge joints work for the diver's hand, but not yet for his arms or legs. This code will be debugged and worked on more.

Also, now there is a fluid simulator. It is not very realistic, but it seemed to exert an opposite force on rigid bodies that move through it. It is slow sometimes, so it may be replaced by an entirely different algorithm.

A static environment is also generated using the PlayCanvas engine, which has an underwater setting which is where the user begins to move and interact.

The test can be run by opening https://localhost:8080/ after running the commands "`npm i && npm run dev`" in the project directory. Currently, the test shows a "bird" flapping its wings in the fluid environment.

3D models have been found and will be implemented in the next iteration. Using 3D objects extracted from zip files we can use these models and identify them before implementing them into the simulator.

Each 3D model will have a form of identification that can be used to identify the type of marine species for the user. Various coral and marine plant 3D models will be accessed from the collection of coral 3D models developed by the Smithsonian Institution (Public domain access).

Different 3D environments have been found for the implementation of the simulator, which are found on open-source sites such as CadNav.

Current Use Case for Webcam Calibration:

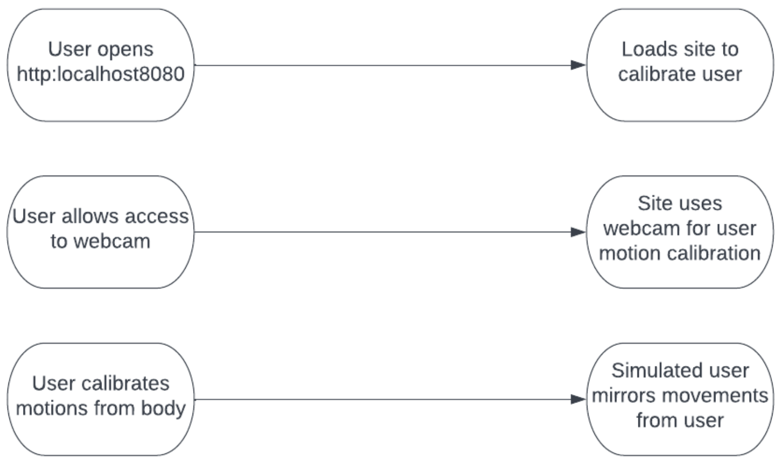
User opens http:localhost8080
Loads site to calibrate user
User allows access to webcam
Site uses webcam for user motion calibration
User calibrates motions from body
Simulated user mirrors movements from user

Figure 1: Use Cases for Webcam Calibration

Joel test:

1. Source control: yes
2. Build in one step: yes
3. Daily builds: no
4. Bug database: no
5. Fix bugs before writing code: sometimes
6. Up-to-date schedule: (I don't know)
7. Have a spec? No
8. Quiet working conditions: yes
9. Best tools money can buy: no
10. Dedicated testers: no
11. New candidates write code: N/A
12. Hallway usability testing: not yet

Test Cases:

There are currently two kinds of tests: automated tests for math functions and observational tests for entity functions. The math tests just run functions and test whether the answers are close enough to expected results; the observational tests are run by commenting/uncommenting code that calls “playground” functions that make entities, and the tester observes whether the resulting 3D rendered scene and physics seem to match what the playground functions are supposed to do.

There are 36 automated math tests (33 currently passing). They test the functions dealing with Euler angle rotations (rotating a vector by Euler angles and decomposing one or two rotated vectors into their transforming Euler angles), functions dealing with (geometric) transformations (specifically, transforming a translation-rotation-scale transformation with another translation-rotation-scale transformation), and functions that help to decompose pose key points into corresponding pose angles. These tests are run with a testing framework like JUnit where inputs and expected outputs are given for each test-runner function; also, these tests use a custom function to compare whether the given output is close enough to the expected output.

There are 19 observational tests in the “playground”, and they are tested by uncommenting lines from the game entry point function and re-running the project. (For this iteration, the “ground” and “bird” playground functions are shown.) The observational tests currently deal primarily with hinges. They also test code for capsule, sphere, and box entity generators, moving rigid bodies, the diver, and components of him. These tests are helpful for quickly observing the consequences of implementing a function one way or another when the function is difficult to understand, and they also make unintended consequences more quickly visible.

Instructions to Run Project:

1. Install nodejs.
2. Clone ocean simulator repository.
3. Run "npm install" in there.
4. Run "npm run dev" in there.
5. Navigate to https://localhost:8080 from the development computer and wait for it to load.
6. Observe the “bird” flapping its “wings”; notice the hinges and fluid simulation.

Feedback:

A representative of the target audience viewed the current environment of the ocean simulator and believed it had characteristics of an ocean environment. The fluidity of the motion control felt more like swimming, according to the user, and is anticipating the combination of the motion control into the actual environment of the game.