766 Final Project Report: Fishing Simulation

Xubin Wang

Abstract

This project aims to create a fishing simulation environment using Unity. The simulation features a fishing rod, line, and hook, as well as sharks swimming within a buoyant water system. By implementing Unity's physics system, including forces such as gravity, buoyancy, and water drag, the project demonstrates an engaging and visually accurate simulation.

1 Introduction

1.1 Objective

The primary goal of this project is to create an fishing simulation that incorporates real-world physics, enabling realistic interactions between the rod, line, hook, water, and fish.

1.2 Motivation

This project was inspired by assignments and lab work, exploring the combination of rigid body dynamics and kinematics. Initially, a YouTube tutorial was used as a reference, but due to the unavailability of its materials and textures, the system was built entirely from scratch.

2 Methodology

2.1 Features and Components

2.1.1 Fishing Rod

The rod consists of three segments: the handle, mid, and tip, connected using hinge joints to allow them to move as one. Animations add to simulate waving.

2.1.2 Fishing Line

The line is segmented and controlled using a LineRenderer. Spring joints between segments enable dynamic movement. A controller script manages line tension and updates its position.

2.1.3 Hook

The hook is attached to the last segment of the line. Physics forces, such as buoyancy and gravity, affect its movement. The hook interacts with the water area.

2.1.4 Water System

Water Area: Simulates buoyancy and drag forces, enabling interaction with the hook.
Water Mesh: Provides a visual representation of the water surface.

2.1.5 Sharks

Move within the water area. They interact with the water and can be programmed to respond to the hook.

2.2 Physics Simulation

The simulation incorporates the following forces:

- **Gravity:** Applies downward force to all objects.
- **Buoyancy:** Upward force proportional to the submerged volume.
- **Water Drag:** Resists motion within the water.

3 Implementation

3.1 Key Scripts

3.1.1 FishingLineController

- Dynamically updates the `LineRenderer` to reflect the positions of the rod tip, hook, and all segments in between.
- Provides a smooth visual representation of the fishing line as it interacts with the rod and hook.

3.1.2 WaterArea

- Applies directional flow forces to objects with rigidbodies when they stay in the water area.
- Triggers a splash particle effect upon entry of specified objects (e.g., a fishing hook).
- Allows customization of water flow direction and strength through public variables.

3.1.3 RodController

- Enables animation control for the fishing rod.
- Links the `Animator` component to the rod object for seamless animation playback.

3.1.4 Fish Movement

- Implements random movement with periodic direction changes.
- Ensures fish remain within the water area by clamping their position and turning them away from boundaries.
- Smoothly transitions fish orientation using `Quaternion.Slerp` for natural turning.
- Dynamically adjusts movement speed using a randomized multiplier.
- Visualizes the water area boundaries with debug gizmos in the editor.

3.1.5 Buoyancy

- Calculates buoyant force proportional to the submerged depth using:

$$F_b = \min(\rho \cdot g \cdot d, F_{\max})$$

  where F_b is the buoyant force, ρ is the fluid density (assumed constant), g is the gravitational acceleration, d is the submerged depth, and $F_{\max}$ is the maximum buoyant force (clamped value).

- Applies drag force and angular drag to reduce movement and stabilize the object in water:

$$\text{Drag Force} = -C_d \cdot \mathbf{v}, \quad \text{Angular Drag Torque} = -C_a \cdot \boldsymbol{\omega}$$

  where C_d and C_a are the drag coefficients, $\mathbf{v}$ is the velocity, and $\boldsymbol{\omega}$ is the angular velocity.

- Automatically resets drag values when the object exits the water.

3.2 Challenges

- **Fine-Tuning Parameters:**
  - Balancing forces like buoyancy and drag was tricky to ensure objects moved naturally in water.
  - Adjusting fish speed, turning, and direction changes required extensive testing to make movements realistic.
  - Tuning the fishing line's components and responsiveness to rod and hook movements took multiple iterations to prevent unnatural behavior.
- **Avoiding Shark-Wall Collisions:**
  - Sharks needed to avoid walls without abrupt direction changes.
  - A buffer zone was added near boundaries to detect walls early, and sharks were programmed to turn smoothly away from them.
  - Position clamping was used to ensure sharks stayed within the water area.
- **Rendering the Fishing Line:**

  - The fishing line had to dynamically follow the rod, hook, and segments while appearing flexible.
  - Spring joints were used for a natural curve, and the `LineRenderer` was adjusted to ensure smooth and realistic visuals.

4 Results

4.1 Features Demonstrated

You can find the demo with this Link

5 Future Improvements

- Add functions to make sharks react to the hook.
- Enhance water visuals with ripples, reflections, and splashes.
- Introduce gameplay mechanics, such as a scoring system and fishing success rates.

6 Conclusion

The project demonstrates a physics-based fishing simulation with interactive components and realistic animations. Future enhancements can further improve visual fidelity and gameplay depth.