MECH 415

Advanced Programming for Mechanical and Industrial Engineers

Final Project

Mechanical System Modelling & Simulation

Jalen Charrette-Wells (40058375)

Vinura Paranagama (40027111)

Noah Reid (40024495)

Presented to: Dr. Brandon W. Gordon

Date of submission: December 24th, 2020

Table of Contents

Introduction

The following report will serve as a user manual whereby the functionality and key components (objects, equations) of the simulation will be presented. Additionally, the different abilities of the simulation will be correlated with the classes that allow for such an ability to be performed. On a final note, detailed descriptions of the more involved components of the program have been thoroughly described within the program.

How to play:

Press the 'I' key to throttle up and increase the speed, press the 'K' key to throttle down and slow the ship. Press 'T' to fire the torpedo, only one torpedo may be fired at a time. In multiplayer, the objective is to either shoot down the enemy ship or collect more coins than the other player before the timer runs out.

In the game menu screen, you can choose between multiplayer and single player, decide who is player 1/ player 2, choose the type of ship, and the number of islands in the game. For multiplayer, The IP address of the local and client computers are set in the main at line 72 and 75 respectively.

Project Structure & Implementation

Object Oriented Implementation

An object-oriented approach has been used to develop this simulation. One of the virtues of C++ is in its ability to provide excellent encapsulation capabilities. This approach provides a structured software with each member variable stored within its class, we can notice here that as our program has scaled in size and consequently functionality, we avoid any accidental interferences between the different components of the program. A project this size makes it evident why an object-oriented approach with encapsulation is required and useful. Ultimately, from an industry point of view, this approach may reduce debugging time and also allows for multiple engineers to collaborate on a piece of software with each working on different objects that can at a later point be assimilated.

Testing & Debugging

The approach here was simple. We meticulously assembled our program one module at a time and verified for any issues prior to adding more features. However, despite our best efforts, as networking was added during the later stages of the project, some issues were spread out the program. At that point we restarted the process and established the networking framework and added one feature at a time until the software was restructured.

Mechanical System Modelling & Simulation

Equations used

$$Fx = Wd + ma$$

The governing equation of motion is shown above. It is used to find the velocity in the vel() function. Fx is the forward force, Wd is the drag from the water, and Ma is the mass multiplied by the acceleration. Wd is given as: wd=0.5*rho*s*cf*(v^2). rho=density of water (1000), s= wetted surface area (60), cf=0.075/(log(r)-2)^2,
r=reynolds number=rho*u*L/mu, L=length of hull, U=flow speed=0.5 m/s, and Mu=dynamic viscosity 0.003.
Deriving the equation gives: Fx=(0.5*rho*s*cf)v^2+(m/dt)v. This can be rearranged as a quadratic equation to solve for the velocity Av^2+Bv+C=0. The negative values are neglected.

2D Graphics using DirectX Toolkit library

The designed software emulates the motion of a boat. The display of the simulation has been created using the DirectX Toolkit Library. The various Sprites include a background, a boat, an island, a torpedo and coins. Finally, an information box displays the system velocity, the engine RPM, the boat hit points and finally the number of chambered torpedoes.

Keyboard Input

The system input is governed by the input (double dt) member function of the “World” class. In terms of operations, the system can be controlled with the following control keys. Press key “I” to speed up and key “K” to slow down.

[KEY I]: Throttle up

[KEY J]: Control Angle CCW

[KEY K]: Throttle down

[KEY L]: Control Angle CW

[KEY T]: Launch Torpedo

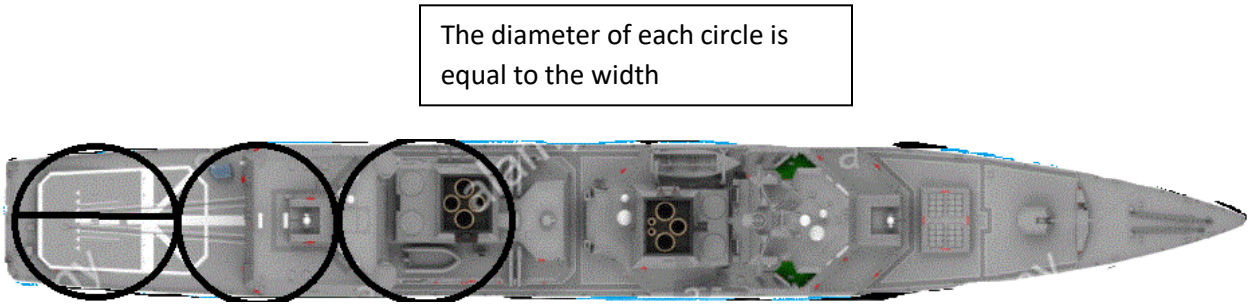
Sound Effects

The software contains some simple audio playouts. The user can expect audio playback occurrence during the following circumstances:

1) The boat collects coins
2) The boat is struck by an opponent torpedo
3) The boat launches a torpedo
4) The boat collides with an island
5) The boats collide with one another

Collider

The collider function determines the edges of the ship to determine if the ship has collided with an object. The function draws circles of diameter equal to the width of the ship along the length of the ship, from the front to the back, the number of circles is therefore equal to the length/width.



The xb and yb coordinates for the rear of the ship are `xb = (cos(q + pi) * d) + (x), yb = (sin(q + pi) * d) + (y).` Where x and y are the centroids of the ship. And d is half the length of the ship. So xb and yb gives the position of the rear relative to the origin.

Collision (Boat-Island)

The Boat and Island collisions functionality is governed by the member function collider() of the "ship" class and the member function collision() of the "World" class. Briefly, the collider() function initializes the position based on the rear of the ship and places collision circles along the length of the boat. Afterwards, the collision() functions computes the distances between the boat and island radii and using an if-statement, the program will establish whether the two systems are in contact.

Torpedo & Coins

The boat has also been interactable with the coins and torpedoes of the system. Firstly, the coins are removed from the system using the member function collision_coin() from the "World" class. Similarly, to the Boat-island collision, the distance between the boat and the coin is computed, afterwards the sum of the radii is compared with the center distances. The collision will be registered if the center distance is less than or equal to the sum of the radii. Now, when a collision occurs, the member variable coin_counter is incremented and appropriately displayed to the screen. Furthermore, the member function remove_coins(int i) from the "World" class is used to delete the coin from memory and consequently the coin sprite. Finally, this member function will trigger an audio playback occurrence (Coins_Collected.wav).

The Torpedo is essentially governed by the exact same approach. Briefly, the torpedo verifies for collisions using the following members functions from the "World" class. The collision_torpedo () and collision_torpedo_ship() verify for collisions with the borders of the map and with other ships respectively. After a collision is detected, the remove_torpedo (int i) member function also part of the "World" class deletes the torpedo and will remove hit points from the targeted boat.

Display (Coin, torpedo and health counters)

The ship_speed () member function of the "World" class uses the text() function from the DirectX library to display the number of coins collected, the remaining chambered torpedoes as well as the hit points of the boats.

Networking

Two Users network communication

This game can be played in either single or multiplayer. The opening_screen function allows for the player to select which option they would like to play by entering either 1 for single player or 2 for multiplayer. If multi player is chosen, then the player must next select to be player 1 and set the game parameter for both players i.e., Boat selection for the local player, Boat selection for the client player, and number of islands using the check_initial_value function. They can choose to be player 2 and receives its initial values from the client player also using the check_initial_value function. If both players select to be the same player default parameters are used in the set_default function and a player 1 and 2 is set based on the 10^{th} value in their respective IP address in the set_default_player function. Both players will be stuck in a while loop waiting until completion of the opening screen by both players and a message is sent to the opposite computer once completed. This allows both players to synchronize their games so they begin at the same time assuming similar operating speeds while running the Sleep(1000) command. Once a message is received on both computers the initial values are set and a 5 second countdown begins to allow both players time to move to the DirectX window from the command prompt. While the game is running the World object begins to be updated in a infinite while loop. In each loop information is sent to the other computer to update ship location, health, torpedo parameters. The while loop is finally broken either once 60 seconds has gone by or when a players ship has been shot down by the opposing player

UDP/IP and socket API

This game is set up to use IPv4 IP address' thus both players must be connected to the same wireless local area network. Code for Ipv6 functions have also been included but the main was not set for the ipv6 sockets. The ipv4 socket programming allows for both players to play together in real time, both players can view each other's ships and shoot torpedo's at each other by pressing the 'T' button. If a torpedo is shot both screens show the torpedo allowing the two players to battle to gather coins that appear on the linked map and avoid being hit by the enemy torpedo's.

Contributions

Networking: Jalen

Velocity model: Vinura

Objects: Vinura, Jalen

Collisions: Jalen, Vinura, Noah

Sounds: Noah

Testing/Debugging: Jalen, Vinura, Noah