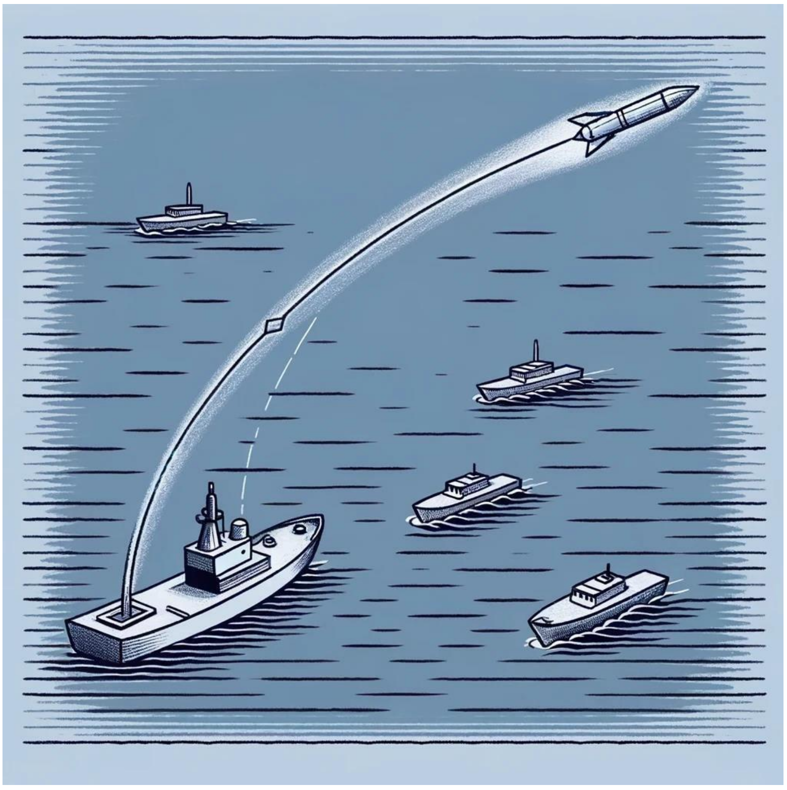
Project Title: Advanced Naval Battle Simulator

What's This Project About? We're building an advanced simulator for navy training. It's a special computer program that mimics real-life situations at sea. Unlike a video game, this is a serious tool that helps naval personnel practice and understand the strategies of naval warfare. The aspect of the simulator is that it's centered around a stationary warship engaging enemy ships on the sea surface.



Introduction

Imaging a situation where a battleship (B is used to denote) is attacking escort ships (E is used to denote) that are protecting cargo ships in the middle of the ocean. A simulator of war games in such a context can be developed by considering constrains and objectives of such war games. In this assignment, students are expected to develop such a simulator with given features.

Development of the simulator is broken into different parts such that simpler features are first added to the simulator and gradually, complex features are added. Objectives, requirements and constrains for each part is given as a standalone task. Students should do the tasks in a sequential manner and after completing each task, a simulation attempt should be done and the outputs should be saved accordingly.

This simulator is based on battleships and escort ships used in World War 2. There are 4 different battleships in the simulator and 5 different escort ship types. Battleship belongs to the Allies and escort ships belong to the Axis Powers. At any given time, there going to be only one battleship on the naval battlefield (called the canvas in simulator) and many number of escort ships. Each escort ship in the simulator represents both the escort ship and the passenger ship it is protecting. Important details of them are given in the Table 1 and 2.

Battleship is assumed to have one gun and each escort ship also assumed to have one gun. When the gun fires, the shell follows the typical parabolic projectile motion trajectory under the effect of gravity. If the attack is executed correctly, shell should explode when it hits the target enemy ship. This is called the impact. Throughout this projectile motion, air resistance and other such effects are considered have negligible effect. While some of the useful equations are given here, students should be able to derive other required equations from first principles as needed. (Important: projectile motion knowledge beyond Advanced Level (A/L) is not required to complete this assignment)

Objective

Battleship should try to destroy as many escort ships as possible. Battleship should also try to minimize the impact on itself from the attacks from escort ships.

Main features

- Guns in **escort ships (E) and battleship (B)** can move in any direction horizontally.
- When it comes to vertical angle, each different escort ship type has a working range (as shown in figure 1). o These are given in below tables.
- The battleship can launch the shell at any vertical angle (between 0-90 degrees).
- Initial speed of shells from E and B also has different ranges. The exact values of these should be given by the student either using random number generator in the code or as user inputs.

- Maximum speed of the battleship's shell should be given by user or else should be randomly generated.
    - Minimum speed of the B's shell (V_{max^B}) is 0.
- The 2D canvas area (i.e., naval battlefield) should be determined by the student but it should be a square. This can be determined by, ○ Lower left corner coordinates = (0, 0) ○ Upper right corner coordinates = (D, D) where D should be given by student.

    (This too can be a user input or a generated random number within the code.)
- Number of escort ships (N) should be decided by the student as a user input.
- Canvas should be populated by the N number of escort ships by randomly generating the (x,y) coordinates of each escort ship.
- Type of each escort ship should also be randomly selected from the 5 types given in the below table.
- Each escort ship should have a unique identifier (i.e., index number).
- Type of the battleship should be given as a user input at the start of the program.
- All randomly generated numbers should be determined by appropriate maximum value that should be decided by the student.
- Starting position coordinates of the battleship should be either given by the user or should be randomly generated.
- All E ships and B are not moving in this simulator (i.e., ship velocities for E and B can be considered as 0).

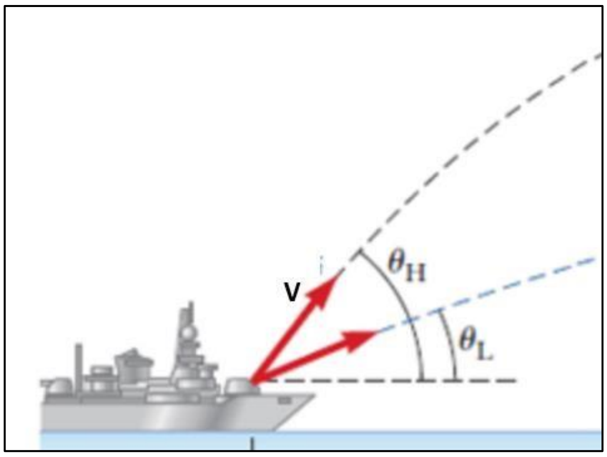


Figure 1 Vertical angle range in escort ships

Type Notation	Type Name	Gun Name	Impact Power	Angle range	Minimum angle (θ_L)	Minimum velocity	Maximum velocity
EA	**1936Aclass Destroyer**	SK C/34 naval gun	0.08	20	Randomly generated	Randomly generated	1.2 $* V_{maxB}$
EB	**Gabbianoclass Corvette**	L/47 dualpurpose gun	0.06	30	Randomly generated	Randomly generated	Randomly generated

E_C	**Matsuclass Destroyer**	Type 89 dualpurpose gun	0.07	25	Randomly generated	Randomly generated	Randomly generated
E_D	**F-class Escort Ships**	SK C/32 naval gun	0.05	50	Randomly generated	Randomly generated	Randomly generated
E_E	**Japanese Kaibōkan**	(4.7 inch) naval guns	0.04	70	Randomly generated	Randomly generated	Randomly generated

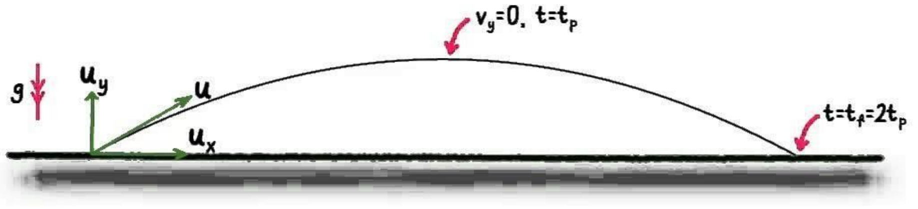
$V_{max}{}^{B} = maximum\ velocity\ of\ Battleship'$s shell

Angle range = $\theta_H - \theta_L$

All randomly generated maximum velocities of escort ships except E_A should be less than V_{max}^{B}
.

Battleship Name	Notation	Gun name
USS Iowa (BB-61)	U	50-caliber Mark 7 gun
MS King George V	M	(356 mm) Mark VII gun
Richelieu	R	(15 inch) Mle 1935 gun
Sovetsky Soyuz-class	S	(16 inch) B-37 gun

Projectile motion equations



$+\uparrow$ $(v_y = u_y + a_y t)$

$$0 = u_y + (-g)t_p$$

$$t_p = \frac{u_y}{g}$$

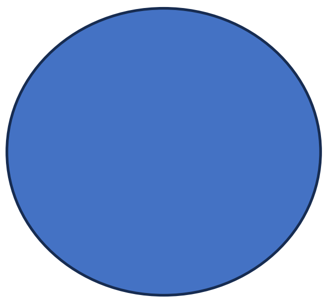
$(s_x = u_x t)$

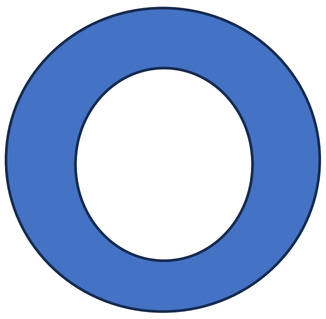
$$\longrightarrow + \quad R = u_x t_f = u_x \frac{2u_y}{g}$$

$$= \frac{2(u\cos\theta)(u\sin\theta)}{g}$$

$$= \frac{u^2 \sin 2\theta}{g}$$

Attack range


Attack range of Battleship [B is at the center]


Attack range of a typical escort ship [E is at the center; Blue coloured area is the attack range]

Tasks

Simulator features are added gradually in two parts. Each part contains sub parts A, B and C. While the below mentioned features and simulation outputs are necessary, students can add any other features they consider useful to have in a simulator.

Part 1 - A

In this part we add the very basic features including initial conditions of the naval battlefield. Our objective here is to simulate the battle at any given time by drastically simplifying the battlefield. To achieve that we assume,

- The gun on P can be reloaded and fired in 0 seconds.
    - In other words, we consider there is no time difference between two consecutive gun firings.
- We also assume that one E ship can only fire once.
- We assume a single shell impact on B can destroy it.
- We assume a single shell impact on any E can destroy it as well.

Battleship can hit all ships in its attack range. Any escort ship can also hit B if B is in escort ship's attack range.

Steps

- Initial conditions of the battlefield should be saved as text file.
    - o Includes positions, max V, min V, max angle, min angle, type of E.
    - o Initial position, type and V max of battleship.
    - o Any other values you have in your implementation.
- Determine if B is going to sink in this scenario or not.
- If B is going to sink, o display the, index number of the E that sank B o save the final conditions of battlefield as a text file
- If B is no going to sink, o Display how many E ships get hit by B. o Display how long it takes to end the battle. o Save to a text file the detail of all Es that got hit by B
    - Index of E
    - Time to hit
    - Any other values o Save the final conditions of the battlefield as a text file.

<u>Part 1 – B</u>

In this sub task, there are two different simulations.

Simulation 1

In this, we simulate battles when battleship is movement in a given path. To simplify this process, we do not use any ship velocities but instead generated k number of points within the canvas. Student can use a random number generator to generate these points or any other approach. At each point, simulation is done as in part 1-A thus we repeat simulations for all k points *or* until the B sinks due to a shell attack from an E ship.

<u>Steps</u>

- Students can use their own method to generate the k number of points and the order of the points (which corresponds to a path) *or* randomly select positions and order of points.
- If a E ship gets destroyed in ith iteration, then it should not be present when simulating the rest of the iterations.
- Get the simulation results for k iterations or until the p (p<k) iteration when B is destroyed.
    - o Save results [same details as in part 1-A] of each simulation step as a text file.

Simulation 2

In this, we assume after t iterations (t<k), gun on the B get jammed such that it can now only work in a particular vertical angle range. It still can attack in any direction in the 2D plane. With this new feature, we do the simulation in the same way we did in *simulation 1.*

<u>Steps</u>

- Vertical angle is only between $\theta_{min} - 90$ degrees.
    - $0 < \theta_{min} < 30$
- Do the simulation again with the same initial conditions as in *simulation 1* and observe the differences between this and the previous one. • Save the outputs to text files in the same way we did in previous one.

<u>Part 1-C</u>

In this sub task we no longer consider that a E ship can destroy the B with a single shell attack since that is not what happens generally. Generally, battleship can withstand multiple attacks from small destroyers etc. (i.e., escort ships).
In order to add this consideration, we assume that a particular type of E ships can only impact the B by a certain percentage. This is given by the impact power in the table 1.

For instance, if when it is 0.08 for E_A s, that means one shell attack from that type of E ship can only destroy 8% of the B. Thus, at minimum, 13 such attacks from 13 such ships are needed to destroy the B assuming there is no other type of E ships which have B in their attack range.

Thus, with this added feature we are making the simulation more realistic.

- Still one E ship can attack only once.
- B can destroy any E ship with a single attack.

<u>Steps</u>

- Taking this new feature into consideration, redo the simulations in part 1-A and B.
- Save the results into text files as detailed part 1-A and B.
    - Add the cumulative impact on the B ship if it wasn't destroyed by the end of the simulation.

Part 2-A (Optional)

In this task, we take into consideration of the time it takes to go from one gun firing to another gun firing. There is still only one gun in the B as well as in any E ship.

This time between two firing is important since getting a gun to the right configuration (angles etc.) in real battleships take a non-negligible time.

Steps

- $T_B^q$is the time between two consecutive gun firings of B ship of type q.
- T_B^q should be decided by the user *or* should be randomly generated.
- Student should come up with a custom strategy that can be used by B to determine the attacking order of E ships in its attack range.
    - Students are encouraged to develop the strategy such that it maximizes the damage on the E ships while minimizing the impact on B ship itself (from E ships).
    - This strategy doesn't have to be the optimal strategy.
        - This is a NP-hard combinatorial optimization problem.
    - Simulate part 1-A, B and C and output the results as a text file as before.
        - Also add the attack order of E ships within B ship's attacking range in each simulation. Part 2-B (Optional)

In this task, going a step further from part 2-A, E ships also have the ability to fire guns continuously instead of firing just once. And the time difference between two consecutive gun firing is a constant value similar to part 2-A. This time difference is also unique to the escort ship type.

Steps

- T_E^p is the time between two consecutive gun firings for the type p escort ship category.
    - For all 4 escort ship types there are 4 different values.
- T_E^p can be a user input *or* a randomly generated value.
- Students are encouraged to develop the strategy such that it maximizes the damage on the E ships while minimizing the impact on B ship itself (from E ships).
- This strategy also doesn't have to be the optimal strategy.
    - This is a NP-hard combinatorial optimization problem.
- Simulate part 1-A, B and C and output the results as a text file as before.
    - Also add the attack order of E ships within B ship's attacking range in each simulation.

Part 2-C (Optional)

To make the simulator more realistic, we assume that number of firing of guns on B and E ships result in impact power degradation of each ship. We also assume that this degradation can be modeled as a power law.

- When this happens to B, that means B can no longer destroy an E ship in a single attack. It's going to take 2 or more attacks to destroy an E ship.
- In all previous simulations, impact power was 1 (i.e., one attack resulted in 100% damage on target E ship).
- Equation for impact power (IP_n) degradation is given below. Gamma (γ) value determine the rate of degradation, and this is a unique value for each ship type of B and E.
  - Number of gun firings is denoted by n.
  - IP_0 denotes the impact power at the very beginning.
  - Gamma is close to zero for B and relatively higher for E ships.

$$IP_n = IP_0 \cdot e_{-\gamma \cdot n}$$ Steps

- Gamma values for each ship type (B and E) can be either user inputs or randomly generated values.
- Simulate the part 1-C and save the results as previously in a text file for all simulations.
  - Save the current impact factor of each ship also in the text file.

The menus of the simulator can be as follows *(but not limited to. You can customize everything as you wish):*

Main Menu Options

1. **Start Simulation**: Accesses a submenu for adjusting various simulation parameters and start simulation.
   1. **Setup**: Set variables required.
   2. **Show simulation**: display the statistics.
2. **View Instructions**: Displays instructions or help about how to use the simulator.
3. **Simulation Statistics**: Shows statistics from previous simulation. 4. **Exit**: Closes the simulator.

Setup Submenu Options

1. **Battleship Properties**:
   - Notation used (Type).
   - Gamma values.
2. **Escort ships Settings**:

    - Notation used (Type).
    - V max and V min values.
    - Angle max and Angle min values.
    - Impact power.
    - Gamma values.
3. **Seed value**: setup the random number generator seed value
4. **Return to Main Menu**: Goes back to the main menu.

Instructions Menu

- Provide detailed guidance on how the simulation works, controls, and objectives.

Simulation Statistics

- Display past simulation results by accessing saved text files.

Exit
- Confirm if the user wants to exit and then close the application.

Simulation Statistics:

Your program should save the user input variables and the simulated variables from the previous sessions.
Therefore, save all parts of each simulation in a text file (unless they are already saved as text files) and load them whenever user choose the simulation statistics from the menu.

Submission Guidelines:

- Ensure your code is well-commented
- Write a report including the following
    - If your system does not completely fulfill the above specification, describing what has not been implemented and the challenges you had not implementing them.
    - If your system works as in the specification, write about the challenges you faced and how you overcome those challenges with examples.
    - If you get another opportunity to implement the same system, what are the improvements you are planning to make
- Submit:
    - Your C programs
    - The report (minimum 500 words)

The rubric will evaluate, but is not limited to the following criteria:

- Functionality: Does the program perform all the specified tasks correctly?
- Code Quality: Is the code well-organized, readable, and free of errors?
- Error Handling: How effectively does the program handle potential errors in user input?
- Report: Is the report well-written and does it provide a clear understanding of what was not implemented and why it was not implemented, or challenges faced and how those were overcome and new features to be added.