TEAM E

SOMALIAN PIRATE SIMULATOR

SOFTWARE DEVELOPMENT DOCUMENT

DECEMBER 7, 2021

Nathan Moore
Luke Tomlinson
Aaron Mendez
Jack Bragg

Team E
Nathan Moore
Luke Tomlinson
Aaron Mendez
Jack Bragg

Table of Contents

Team E
Nathan Moore
Luke Tomlinson
Aaron Mendez
Jack Bragg

Introduction

Project Overview

The International Maritime Bureau (IMB) reported that over the period 2005–2012 nearly 1,000 ships were fired at, chased, boarded, or hijacked by pirates(Appendix A.1-3). According to the IMB these pirates often operate from bases in Somalia. Our team has been tasked with the creation of a featureless, gridded map to simulate interactions between naval patrols, merchant shipping, and pirate attacks. This simulation is intended to help the IMB study the likelihood of pirate attacks as a first step to eliminating future pirate attacks off of the coast of Somalia.

Scope

The Somalian Pirate Simulation(SPS) displays a featureless 100 by 400 gridded map populated by *cargo*, *patrol*, *captured cargo*, and *pirate ships*. These ships will interact with one another as defined in the Software Requirements Specification section in order to provide as realistic results as possible. Statistics of the number of each type of ship that has entered and exited the map and any action taken by a ship on another will be displayed to the user via an easy-to-navigate user interface. The user will be able to control the current state of the Somalian Pirate Simulation via a panel of control buttons which include starting, pausing, ending/exiting the simulation, as well as changing the speed at which the simulation updates. The simulation occurs within a specified time dilation where each single time step is equivalent to five minutes of real world time. Population probability of ships entering the map is also controlled by the user via the keyboard number-pad.

The Somalian Pirate Simulator will not contain geographically accurate information within the map area; eliminating this allows for the user to focus on numbers of ship interactions found during ideal conditions. Ships will not be displayed as “ship-like” entities—instead they will be represented as simple shapes, uniquely colored by type, with a distinct directionality in their movement in order to maintain performance during long term, highly populated simulations. The user will not be able to drag ships to specific locations within the map—ships must reach locations based on the predetermined factors and the interactions they have with other ships (See Appendix A.3 for more information.) User interaction with the simulation is limited to only the above-mentioned actions in order to preserve the integrity of the simulation.

Context Diagram

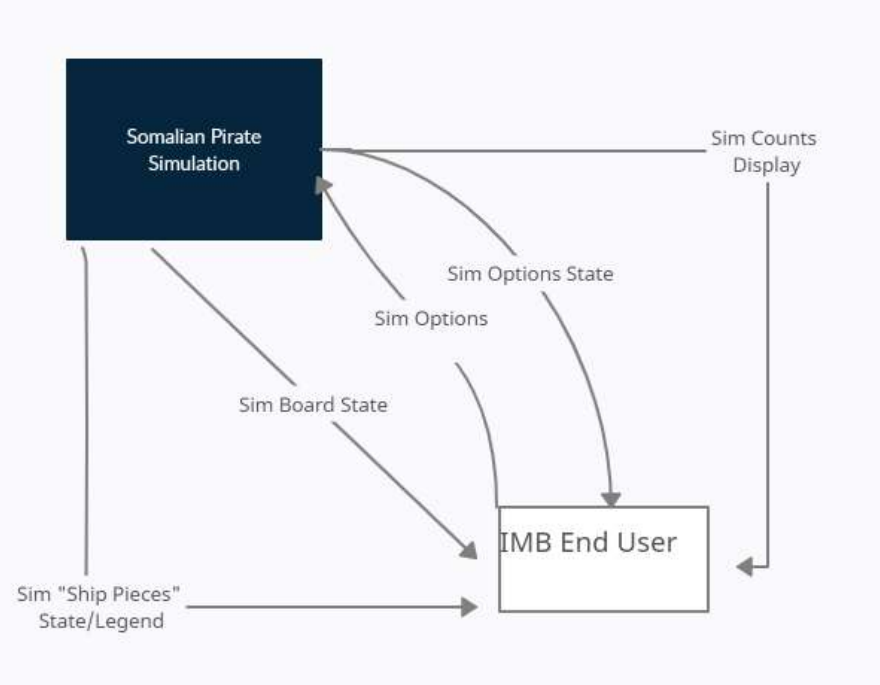


Figure 1: Context Diagram

Summary—Introduction

The introduction for SPS is made up of three sections:

1. Project Overview
2. Scope
3. Context Diagram

The Project Description gives a brief overview of the project assigned to Team E and the importance of developing The Somalian Pirate Simulator. This section is included to give the reader a brief understanding of the overall goal for implementation of the simulation. More detailed information relating to the project description in general terms can be found in the Statements Of Work found in Appendix A.

The Introduction section summarizes the capabilities of The Somalian Pirate Simulator. This section briefly describes all of the interactions the user can and cannot make with the simulation as well as

reasoning behind these design decisions. More detailed information relating to this section can be found in the Software Requirements Specification as well as Appendix A.
A visual Context Diagram has been included to give the reader a general understanding of exactly how the user will interact with The Somalian Pirate Simulation as well as how the simulation relays information to the user through the general user interface.

Team E
Nathan Moore
Luke Tomlinson
Aaron Mendez
Jack Bragg

Software Development Plan

The following section is intended to provide an introduction to the personnel working to develop The Somalian Pirate Simulator and showcase their qualifications. It will also provide a breakdown of the team's timeline to execute development, list milestones along the way, and outline the deliverables that the customer should expect and when they should expect them. This section also provides a list of team member risks that have been evaluated by Team E and their mitigation plans in the case that something happens to a team member.

Personnel

1. Nathan Moore—Team Lead

   Nathan is a senior Computer Science student at the University of Alabama in Huntsville; proficient in C++, C#, Java, Python, Haskell, and R languages. Nathan also has more than eight years of restaurant consulting and management experience.

2. Luke Tomlinson—Quality Lead

   Luke is a senior Computer Science student at the University of Alabama in Huntsville with a skill set mainly consisting of front-end design and development using Swift for iOS. Luke has produced and maintained two professionally released iOS applications.

3. Aaron Mendez—Technical Lead

   Aaron is a senior Computer Science student at the University of Alabama in Huntsville. His skills include object-oriented design in both C++ and Java. Aaron currently holds a professional position in the Computer Science field in Huntsville, Alabama.

4. Jack Bragg—Test Lead

   Jack is a senior Computer Science student at the University of Alabama in Huntsville. His skills include both academic and professional software development in: Python, C#, C++, Java, Arm, and Bash. Jack is also proficient in the creation of software design documentation such as UML and sequence diagrams for professional use.

Project Schedule

1. Deliverable Documentation Preparation—Sprints A-E
   1. Deliverables
      a. SDP—September 09, 2021
      b. SDP Presentation—September 07, 2021
      c. SRS/Backlog—October 1, 2021
      d. SRS/Backlog Presentation—September 23, 2021

e. Architectural Design—October 21, 2021
f. Architectural Design Presentation—October 19/21, 2021
g. UI Design—November 09, 2021
h. UI Design Presentation—November 04/09, 2021
i. Application Gold Build—December 07, 2021
j. Application Delivery Presentation—November 30/December 02, 2021

2. Milestones
   a. Delivery of SDP—September 09, 2021
   b. Delivery of SRS—October 01, 2021
   c. Delivery of Architectural Design—October 21, 2021
   d. Deliver of UI Design—November 09, 2021
   e. Deliver of Application—December 07, 2021

2. Conceptual Design Phase—Sprints B-C
   1. Deliverables
      a. SDP—September 09, 2021
      b. SDP Presentation—September 7, 2021
      c. SRS/Backlog—September 28, 2021
      d. SRS/Backlog Presentation—September 23/28, 2021
   2. Milestones
      a. Delivery of SDP—September 09, 2021
      b. Delivery of SRS—October 1, 2021
      c. Delivery of Backlog—October 1, 2021
3. Application Development—Sprints C-E
   1. Deliverables
      a. Delivery of Architectural Design—October 21, 2021
      b. Architectural Design Presentation—October 19/21, 2021
      c. UI Design—November 09, 2021
      d. Prelim GUI Presentation—November 04/09, 2021
      e. Application—December 07, 2021
   2. Milestones
      a. Begin Software Development—September 30, 2021
      b. Delivery of UI Design—November 09, 2021
      c. Delivery of Application—December 07, 2021
4. Organizing Testing Effort—Sprints D-E
   1. Deliverables
      a. UI Design—November 09, 2021
      b. Prelim GUI Presentation—November 04/09, 2021
      c. Application Beta Build Delivery—November 11, 2021
      d. Application Gold Build—December 07, 2021
   2. Milestones
      a. Begin Debugging—October 12, 2021
      b. Delivery of UI Design—November 09, 2021
      c. Application Beta Build Delivery—November 29, 2021

d. Delivery of Application—December 07, 2021

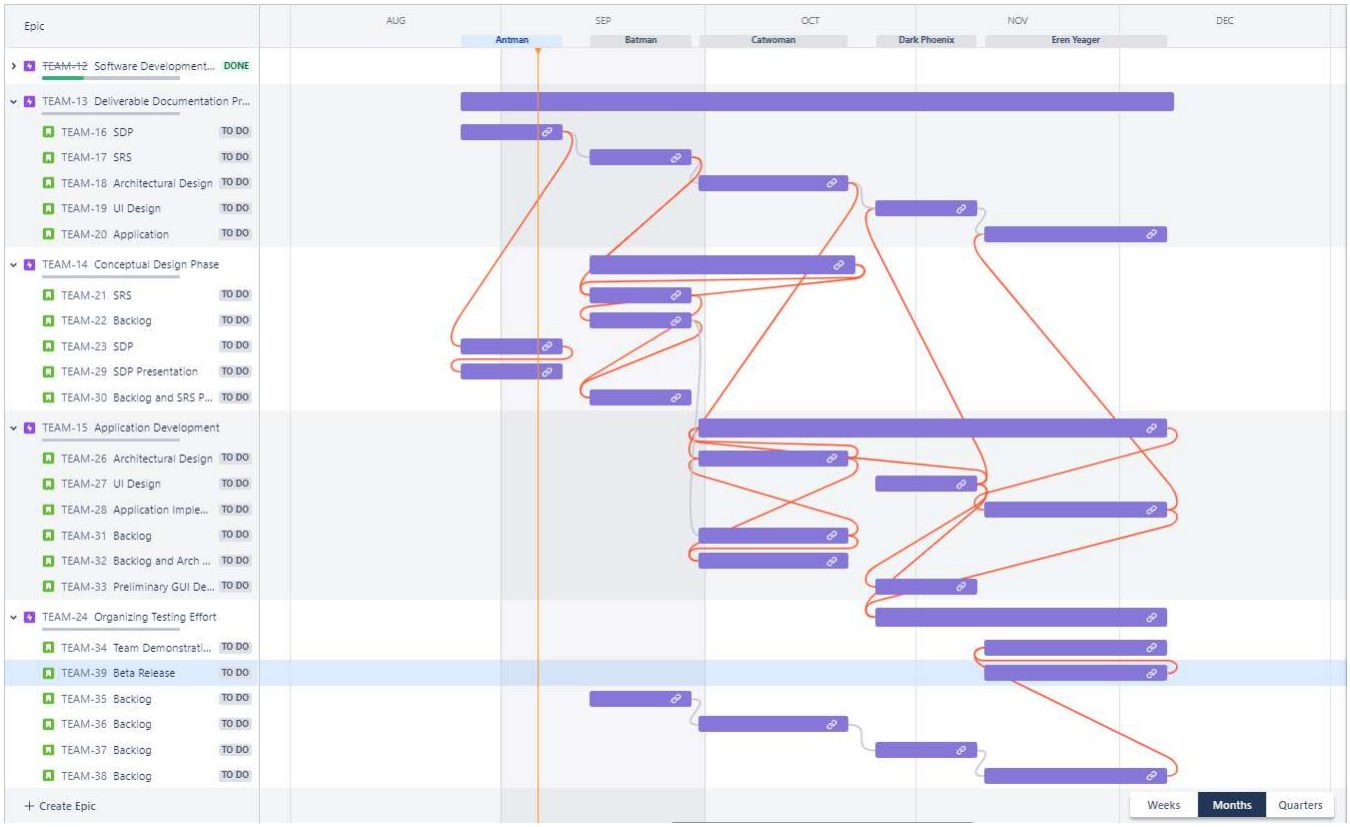


Figure 2: Weekly Software Development Plan

List of Deliverables

1. SDP Presentation—September 7, 2021
2. SDP—September 09, 2021
3. SRS—October 1, 2021
4. SRS/Backlog Presentation—September 23/28, 2021
5. Backlog—October 1, 2021
6. Architectural Design—October 21, 2021
7. Architectural Design/Backlog Presentation—October 19/21, 2021
8. UI Design—November 09, 2021
9. UI Design Presentation—November 04/09, 2021
10. Application Beta Build—November 11, 2021
11. Application Delivery Presentation—November 30/December 02, 2021
12. Application Gold Build—December 07, 2021

List of Milestones

1. Delivery of SDP—September 09, 2021
2. Delivery of SRS/Backlog—September 28, 2021
3. Begin Software Development—September 30, 2021
4. Begin Debugging—October 12, 2021
5. Delivery of Architectural Design—October 21, 2021
6. Delivery of UI Design—November 09, 2021
7. Application Beta Build Delivery—November 11, 2021
8. Delivery of Application Gold Build—December 07, 2021

Risks And Mitigation Plans

1. Multiple Team Members with Conflicting/Dynamic Work Schedules
    1. Likelihood Of Occurrence: 10—Team Cannot Avoid This Risk
    2. Impact On Team: (2-3)—Team Can Schedule Accordingly For Meetings
    3. Mitigation Plan: Planning for meetings must take place in advance, if a team member cannot make a meeting a reason must be submitted to the rest of team. A team group schedule has been implemented in an attempt to mitigate this risk as much as possible.
2. Differentiating OS: Mac/Windows For Development
    1. Likelihood Of Occurrence: 10 – Team Cannot Avoid This Risk
    2. Impact On Team: (1-2)—Team Understands How To Use VCS In Order To Develop Application Using Multiple Platforms
    3. Mitigation Plan: Team has been trained in use of GitHub to properly mitigate this issue which occurs by build targets within Unity Development Environment. All team members are aware of this platform switching issue understands how to switch the Unity build target based on their Operation System.
3. COVID-19 Exposure
    1. Likelihood Of Occurrence: 7—Team Can Somewhat Avoid This Risk
    2. Impact On Team: (5-6)—Team Will Be Down A Member For A Minimum Of One Week If Member Is Unvaccinated, 2-5 Days If They Are Vaccinated
    3. Mitigation Plan: Team members are required to wear well-fitting masks to all in person meetings; remote meetings will be held as often as possible; daily communication between team members via Discord has been implemented for daily stand up meetings/check-ins; tasks assigned to ill member will be redistributed among healthy members
4. Sick Family Members/Team Members Requiring Team Members To Step Back From Tasks
    1. Likelihood Of Occurrence: 10—Team Cannot Avoid This Risk – Risk Encountered 9/10/21
    2. Impact On Team: (5-7)—Team Will Be Down A Member For Undetermined Amount Of Time

3. Mitigation Plan: Tasks assigned to affected member will be redistributed to unaffected team members until affected member is able to return to work

5. Multiple Team Members Taking 4+ Courses At 400-level
    1. Likelihood Of Occurrence: 10—Team Cannot Avoid This Risk
    2. Impact On Team: (3-4)—Team Members Must Have Personal Schedules Implemented Correctly
    3. Mitigation Plan: Team members who have multiple 400-level courses must have their semester schedules planned in advance and know when exam periods occur, when course conflicts will occur, and when heavy work load periods are scheduled. These times must be reported to team to increase flexibility among members who have lighter schedules.
6. Team Mate Drops Course 499
    1. Likelihood Of Occurrence: 5—Team Can Potentially Avoid This
    2. Impact On Team: 8—Team Will Need To Restructure All Future Backlog Tasks
    3. Mitigation Plan: Daily team communication, course/outside work load should be reported to all members. In case that risk occurs, team will redistribute tasks among remaining members and request meeting with customer to discuss future plans/potential of scaling back scope.
7. Accidents Resulting In Bodily Injury
    1. Likelihood Of Occurrence: 3—Team Can Likely Avoid This Risk
    2. Impact On Team: (7-8)—Team Will Need To Restructure All Future Backlog Tasks Depending On Extent Of Injury
    3. Mitigation Plan: Team members are encouraged to always follow personal safety guidelines as well as all posted traffic laws. In the case this risk occurs and a bodily injury that requires a member to drop the semester or pull back from work for an undetermined amount of time, tasks will be redistributed to unaffected team members. A meeting with the customer will then be scheduled to discuss future planning.

Summary—Software Development Plan

The Software Development Plan is organized into five main sections:

1. Personnel Information
2. Project Schedule
3. List of Deliverables
4. List of Milestones
5. Risks And Mitigation Plans

The Personnel Information briefly introduces all of Team E's group members, their strengths, and their knowledge in the programming and development field.

The Project Schedule section gives a visual breakdown of Team E's software development plan by dividing the team's currently known epics into their respective deliverables and milestones. This section is cited for the next two portions of this chapter for lists of all milestones and

deliverables that will be present during the overall development period of The Somalian Pirate Simulation.

The List of Deliverables section is a brief list of all deliverables and their expected delivery dates based on the above project schedule.

The List of Milestones section is a brief list of all milestones the team expects to cross throughout development. The list will be used to determine if sprints, epics, and user stories have been completed and if the project development is happening in a timely manner.

The Risks and Mitigation Plans section details the team's currently known and reasonably foreseeable possible risks in the current environment outside of development. These risks are rated based on the likelihood that they occur and how much they will ultimately affect Team E's overall capability for completing The Somalian Pirate Simulation by the scheduled date of delivery.

Software Requirements Specifications

The following section gives a brief overview of all Statements of Work received by Team E, Use Cases derived from these Statements of Work, and requirements and corresponding tasks the team needs to fulfill to successfully develop the Somalian Pirate Simulator. The Use Cases section lists all possible actions the end user can enact upon the simulation and describes briefly the front- and back-end tasks that the simulation must accomplish when a user action is taken. The Functional Requirements section lays out, in detail, all integral software requirements needed to complete each use case properly. The Program Level Requirements section details all requirements that affect performance, reliability, and portability of The Somalian Pirate Simulator that cannot be directly tied to a Use Case. The Backlog section provides detailed breakdowns of specific tasks to be divided among team members based on the functional and program level requirements.

Sources Of Requirements

Team E received multiple Statements of Work from the customer. This section contains citations and brief descriptions of when the team received these documents; full documentation can be found in Appendix A.

The first two Statements of Work our team received were:

- Dr. Petty—CS 499, Project Description, Somalian Pirates Simulation (Petty—A.1.) Received On August 24, 2021
- Dr. Delugach—CS 499 Senior Project Assignment: Somalian Pirates Simulation (Delugach—A.2.) Received On August 24, 2021

Our team discovered discrepancies between these Statements of Work and through email correspondence with the customer our team was able to receive a list of desired attributes the simulation should exhibit (A.4). From this, Team E created a conjoined Statement of Work (Team E-A.3) which exemplified the desired attributes from both provided Statements of Work. This document was then approved by the customer and is the main source for all Use Cases and Requirements for The Somalian Pirate Simulation Requirements and Specifications.

Use Cases

Use Case 1: Open Simulation—Default State

1. User Opens Application Executable
2. Display A Featureless Grid Map
3. UI Buttons Display On Screen
4. UI Ship Information Counters Display With 0 As All Event Counter Variables
5. UI Time Step Counter Displays On Screen With 0 As Default Value
6. Simulation Speed Sets To 1X Speed
7. Simulation Is Set To Paused State
8. Simulation Waits For User Input

- Settings User May Alter In This State:
  - Simulation Speed Speed
  - Ship Spawn Rates

Use Case 2: Start Simulation, Variation 1: Default State Settings

1. User Clicks “Start” Button
2. Ships Begin To Spawn Based On User Selected Spawn Rates
   1. Cargos Begin Spawning On Left Most Column
   2. Pirates Begin Spawning On Bottom Most Row
   3. Patrols Begin Spawning On Right Most Column
3. Ship Movements Begin Updating At 1X Speed
4. Ships Begin To Move—All Ships Move At Same Time
   1. Cargos Begin Moving Rightward
   2. Pirates Begin Moving Upward
   3. Patrols Begin Moving Leftward
5. Ship Information Counters Begin Updating At 1X Speed
6. Determine If Following Actions Have Occurred
   1. Defeat—A Patrol Ship Has Defeated A Pirate
   2. Capture—A Pirate Has Captured A Cargo
   3. Rescue—A Pirate Has Rescued A Cargo From Pirate
   4. Evade—A Cargo Has Attempted To Evade Capture From Pirate
7. Time Step Counter Begins To Update At 1X Speed
8. Ship Spawn Rates Are Set To:
   1. Cargo Spawn %: 50
   2. Patrol Spawn %: 25
   3. Pirate Spawn %: 40

Use Case 2: Start Simulation, Variation 2: Paused State—User Speed Selected

1. User Clicks “Start” Button
2. Ships Begin To Move—All Ships Move At Same Time
   1. Captured Ships Begin Moving Downward
   2. Pirates Begin Moving Upward
   3. Cargos Begin Moving Rightward
   4. Patrols Begin Moving Leftward
3. Ships Begin To Spawn—Must Occur In Following Order
   1. Cargos Begin Spawning On Left Most Column
   2. Pirates Begin Spawning On Bottom Most Row
   3. Patrols Begin Spawning On Right Most Column
4. Ship Movements Begin Updating At Selected Speed
5. Ship Information Counters Begin Updating At Selected Speed
6. Determine If Following Actions Have Occurred – In Following Order
   1. Defeat—A Patrol Ship Has Defeated A Pirate
   2. Capture—A Pirate Has Captured A Cargo

3. Rescue—A Pirate Has Rescued A Cargo From Pirate
4. Evade—A Cargo Has Attempted To Evade Capture From Pirate

7. Time Step Counter Begins Updating At Selected Speed
8. Ship Spawn Rates Are Set To:
    1. Cargo Spawn %: 50
    2. Patrol Spawn %: 25
    3. Pirate Spawn %: 40

Use Case 3: Pause Simulation From Running State

1. User Clicks “Pause” Button
2. Simulation Stops Populating Ships
3. Ship Information Counters Stop Populating
4. Time Counter Stops Incrementing

Use Case 4: Select 1X Speed, Variation 1: Running State, 2X/10X/20X Active

1. User Clicks “1X Speed” Button
2. All Ship Movements Begin Updating At 1X Per Second
3. All Ship Information Counters Begin Updating At 1X Per Second
4. Time Counter Begins Updating At 1X Per Second

Use Case 4: Select 1X Speed, Variation 2: Paused State, 2X/10X/20X Active

1. User Clicks “1X Speed” Button
2. UX Variables For Updating Ship Movement Change to 1X Per Second
3. UX Variables For Updating Ship Information Change to 1X Per Second
4. UX Variables For Time Counter Information Change to 1X Per Second
5. Simulation Waits For User Input

Use Case 5: Select 2X Speed, Variation 1: Running State, 1X/10X/20X Active

1. User Clicks “2X Speed” Button
2. All Ship Movements Begin Updating At 2X Per Second
3. All Ship Information Counters Begin Updating At 2X Per Second
4. Time Counter Begins Updating At 2X Per Second

Use Case 5: Select 2X Speed, Variation 2: Paused State, 1X/10X/20X Active

1. User Clicks “2X Speed” Button
2. UX Variables for Updating Ship Movement Change To 2X Per Second
3. UX Variables for Updating Ship Information Change To 2X Per Second
4. UX Variables for Updating Time Counter To Change 2X Per Second
5. Simulation Waits For User Input

Use Case 6: Select 10X Speed, Variation 1: Running State, 1X/2X/20X Active

1. User Clicks “10X Speed” Button
2. All Ship Movements Begin Updating At 10X Per Second

3. All Ship Information Counters Begin Updating At 10X Per Second
4. Time Counter Begins Updating At 10X Per Second

Use Case 6: Select 10X Speed, Variation 2: Paused State, 1X/2X/20X Active

1. User Clicks “10X Speed” Button
2. UX Variables for Updating Ship Movement Change To 10X Per Second
3. UX Variables for Updating Ship Information Change To 10X Per Second
4. UX Variables for Updating Time Counter Change To 10X Per Second
5. Simulation Waits For User Input

Use Case 7: Select 20X Speed, Variation 1: Running State, 1X/2X/10X Active

1. User Clicks “20X Speed” Button
2. All Ship Movements Begin Updating At 20X Per Second
3. All Ship Information Counters Begin Updating At 20X Per Second
4. Time Counter Begins Updating At 20X Per Second

Use Case 7: Select 20X Speed, Variation 2: Paused State, 1X/2X/10X Active

1. User Clicks “20X Speed” Button
2. UX Variables for Updating Ship Movement Change To 20X Per Second
3. UX Variables for Updating Ship Information Change To 20X Per Second
4. UX Variables for Updating Time Counter Change To 20X Per Second
5. Simulation Waits For User Input

Use Case 8: Select Single Step, Variation 1: Running State

1. User Clicks “Single Step” Button
2. No Effect on UX or UI Elements of Simulation

Use Case 8: Select Single Step, Variation 2: Paused State

1. User Clicks “Single Step” Button
2. Running State Enables
3. A Single 1X Speed Time Step Occurs
4. Pause State Re-Enables
5. Simulation Waits For User Input

Use Case 9: Select End Simulation, Variation 1: Running State

1. User Clicks “End” Button
2. All Currently Updated UX Elements /Variables Are Cleared From Memory
3. All Currently Displayed UI Elements/Variables Are Cleared From Memory
4. Simulation Exits

Use Case 9: Select Reset Simulation, Variation 2: Paused State

1. User Clicks “End” Button
2. All Currently Updated UX Elements /Variables Are Cleared From Memory

3. All Currently Displayed UI Elements/Variables Are Cleared From Memory
4. Simulation Exits

Use Case 10: Zoom In During Running State

1. Simulation Is Currently Running
2. User Scrolls Mouse Wheel Away In the (+) Direction (Away From User)
3. Camera Orthographic Size Increments At A Speed Of 15 Towards The Max Zoom Size of 10
4. When Camera Reaches Max Zoom Size Camera Will Not Allow User To Continue Zooming In

Use Case 11: Zoom Out During Running State

1. Simulation Is Currently Running
2. User Scrolls Mouse Wheel Away In the (-) Direction (Toward User)
3. Camera Orthographic Size Increments At A Speed Of 15 Towards The Minimum Zoom Size of 50
4. When Camera Reaches Minimum Zoom Size Camera Will Not Allow User To Continue Zooming Out

Use Case 12, Variation 1: User Changes Spawn Rate Of Cargo Ships To 75%(Currently 50% Or 100%); Running State

1. Simulation Is Currently In Running State
2. User Presses The ‘2’ Key On The Keyboard
3. “Cargo Spawn %:” Display Text Changes to “Cargo Spawn %: 75”
4. Cargo Ships Begin Spawning If Any Number Between 0 and 75 Are Picked At Each Time Step

Use Case 12, Variation 2: User Changes Spawn Rate Of Cargo Ships To Default—50% (Currently 75% Or 100%); Running State

1. Simulation Is Currently In Running State
2. User Presses The ‘1’ Key On The Keyboard
3. “Cargo Spawn %:” Display Text Changes to “Cargo Spawn %: 50”
4. Cargo Ships Begin Spawning If Any Number Between 0 and 50 Are Picked At Each Time Step

Use Case 12, Variation 3: User Changes Spawn Rate Of Cargo Ships To 100% (Currently 50% Or 75%); Running State

1. Simulation Is Currently In Running State
2. User Presses The ‘3’ Key On The Keyboard
3. “Cargo Spawn %:” Display Text Changes to “Cargo Spawn %: 100”
4. Cargo Ships Begin Spawning If Any Number Between 0 and 100 Are Picked At Each Time Step – Cargo Ships Will Spawn Every Time Step Update

Use Case 13, Variation 1: User Changes Spawn Rate Of Patrol Ships To 50%(Currently 25% Or 100%); Running State

1. Simulation Is Currently In Running State
2. User Presses The ‘5’ Key On The Keyboard

3. “Patrol Spawn %:” Display Text Changes to “Patrol Spawn %: 50”
4. Patrol Ships Begin Spawning If Any Number Between 0 and 50 Are Picked At Each Time Step

Use Case 13, Variation 2: User Changes Spawn Rate Of Patrol Ships To Default—25% (Currently 50% Or 100%); Running State

1. Simulation Is Currently In Running State
2. User Presses The ‘4’ Key On The Keyboard
3. “Patrol Spawn %:” Display Text Changes to “Patrol Spawn %: 25”
4. Patrol Ships Begin Spawning If Any Number Between 0 and 25 Are Picked At Each Time Step

Use Case 13, Variation 3: User Changes Spawn Rate Of Patrol Ships To 100% (Currently 25% Or 50%); Running State

1. Simulation Is Currently In Running State
2. User Presses The ‘5’ Key On The Keyboard
3. “Patrol Spawn %:” Display Text Changes to “Patrol Spawn %: 100”
4. Patrol Ships Begin Spawning If Any Number Between 0 and 100 Are Picked At Each Time Step –Patrol Ships Will Spawn Every Time Step Update

Use Case 14, Variation 1: User Changes Spawn Rate Of Pirate Ships To 75%(Currently 40% Or 100%); Running State

1. Simulation Is Currently In Running State
2. User Presses The ‘7’ Key On The Keyboard
3. “Pirate Spawn %:” Display Text Changes to “Pirate Spawn %: 75”
4. Pirate Ships Begin Spawning If Any Number Between 0 and 75 Are Picked At Each Time Step

Use Case 14, Variation 2: User Changes Spawn Rate Of Pirate Ships To Default—40% (Currently 75% Or 100%); Running State

1. Simulation Is Currently In Running State
2. User Presses The ‘8’ Key On The Keyboard
3. “Pirate Spawn %:” Display Text Changes to “Pirate Spawn %: 40”
4. Pirate Ships Begin Spawning If Any Number Between 0 and 40 Are Picked At Each Time Step

Use Case 14, Variation 3: User Changes Spawn Rate Of Pirate Ships To 100% (Currently 40% Or 75%); Running State

1. Simulation Is Currently In Running State
2. User Presses The ‘9’ Key On The Keyboard
3. “Pirate Spawn %:” Display Text Changes to “Pirate Spawn %: 100”
4. Pirate Ships Begin Spawning If Any Number Between 0 and 100 Are Picked At Each Time Step –Pirate Ships Will Spawn Every Time Step Update

Functional Requirements

- FR1: UI—Application Must Display a 400(Horizontal) x 100(Vertical) Featureless Map
  a. Use Case 1
- FR2a: UI—Application Must Display A *Cargo Ship Game Object*
  a. Use Case 2
- FR2b: UI—Application Must Display A *Patrol Ship Game Object*
  a. Use Case 2
- FR2c: UI—Application Must Display A *Pirate Ship Game Object*
  a. Use Case 2
- FR2d: UI—Application Must Display A *Captured Ship Game Object*
  a. Use Case 2
- FR3a: UI—Application Must Display a “Start” Button
  a. Use Cases 1, 2
- FR3b: UI—Application Must Display a “Pause” Button
  a. Use Cases 1, 3
- FR3c: UI—Application Must Display a “1X Speed” Button
  a. Use Cases 1, 4
- FR3d: UI—Application Must Display a “2X Speed” Button
  a. Use Case 5
- FR3e: UI—Application Must Display a “10X Speed” Button
  a. Use Case 6
- FR3f: UI—Application Must Display a “20X Speed” Button
  a. Use Case 7
- FR3g: UI—Application Must Display a “Single Step” Button
  a. Use Case 8
- FR3h: UI—Application Must Display an “End Simulation” Button
  a. Use Case 9
- FR4a: UI—Application Must Display “Cargos Entered” Counter
  a. Use Case 1
- FR4b: UI—Application Must Display “Cargos Exited” Counter
  a. Use Case 1
- FR4c: UI—Application Must Display “Cargos Captured” Counter
  a. Use Case 1
- FR4d: UI—Application Must Display “Captured Cargos Rescued” Counter
  a. Use Case 1
- FR4e: UI—Application Must Display “Patrols Entered” Counter
  a. Use Case 1
- FR4f: UI—Application Must Display “Patrols Exited” Counter
  a. Use Case 1
- FR4g: UI—Application Must Display “Pirates Entered” Counter
  a. Use Case 1

- FR4h: UI—Application Must Display “Pirates Exited” Counter
  a. Use Case 1
- FR4i: UI—Application Must Display “Pirates Defeated” Counter
  a. Use Case 1
- FR4j: UI—Application Must Display “Evades Not Captured” Counter
  a. Use Case 1
- FR4k: UI—Application Must Display “Evades Captured” Counter
  a. Use Case 1
- FR4l: UI—Application Must Display “Time Step” Counter
  a. Use Case 1
- FR5a: UX—Selecting “Start” Button Must Toggle From Paused State to Running State
  a. Use Case 2
- FR5b: UX—Selecting “Pause” Button Must Toggle From Running State to Paused State
  a. Use Case 3
- FR5c: UX—Selecting “1X Speed” Button Must Toggle Updates of Ship Movement and Counter Updates to 1 Times Per Second
  a. Use Case 4
- FR5d: UX—Selecting “2X Speed” Button Must Toggle Updates of Ship Movement and Counter Updates to 2 Times Per Second
  a. Use Case 5
- FR5e: UX—Selecting “10X Speed” Button Must Toggle Updates of Ship Movement and Counter Updates to 10 Times Per Second
  a. Use Case 6
- FR5f: UX—Selecting “20X Speed” Button Must Toggle Updates of Ship Movement and Counter Updates to 20 Times Per Second
  a. Use Case 7
- FR5g: UX—Selecting “Single Step” Button Must Toggle To Running State 1X Speed, Update Once, Return to Pause State. Do Nothing If In Running State
  a. Use Case 8
- FR5h: UX—Selecting “End Simulation” Button Must Clear All Objects From And Exits Simulation
  a. Use Case 9
- FR6a: UX—*Cargo Ships* Must Spawn On the Left Most Column And Move Rightwards
  a. Use Cases 2, 4, 5, 6, 7
- FR6b: UX—*Captured Ships* Must Spawn On Grid Where *Cargo Ship* Was Captured By Pirate And Move Downwards
  a. Use Cases 2, 4, 5, 6, 7
- FR6c: UX—*Pirate Ships* Must Spawn On the Bottom Most Row And Move Upwards
  a. Use Cases 2, 4, 5, 6, 7
- FR6d: UX—*Patrol Ships* Must Spawn On the Right Most Column And Move Leftwards
  a. Use Cases 2, 4, 5, 6, 7

- FR7a: UX—“Cargos Entered” Counter Update Must Reflect Amount Of *Cargo Ships* That Have Ever Been Displayed
  a. Use Cases 2, 4, 5, 6, 7
- FR7b: UX—“Cargos Exited” Counter Update Must Reflect Amount Of *Cargo Ships* That Have Successfully Crossed The Map And Exited
  a. Use Cases 2, 4, 5, 6, 7
- FR7c: UX—“Cargos Captured” Must Reflect Amount Of *Captured Ships* That Have Ever Been Displayed
  a. Use Cases 2, 4, 5, 6, 7
- FR7d: UX—“Captured Cargos Rescued” Counter Must Reflect The Amount Of *Captured Ships* That Have Been “Rescued” By *Patrol Ships*
  a. Use Cases 2, 4, 5, 6, 7
- FR7e: UX—“Patrols Entered” Counter Must Reflect The Amount Of *Patrol Ships* That Have Ever Been Displayed
  a. Use Cases 2, 4, 5, 6, 7
- FR7f: UX—“Patrols Exited” Counter Must Reflect The Amount Of *Patrol Ships* That Have Successfully Crossed The Map And Exited
  a. Use Cases 2, 4, 5, 6, 7
- FR7g: UX—“Pirates Entered” Counter Update Must Reflect The Amount Of *Pirate Ships* That Have Ever Been Displayed
  a. Use Cases 2, 4, 5, 6, 7
- FR7h: UX—“Pirates Exited” Counter Must Reflect The Amount Of *Pirate Ships* That Have Successfully Cross The Map And Exited
  a. Use Cases 2, 4, 5, 6, 7
- FR7i: UX—“Pirates Defeated” Counter Must Reflect The Amount Of Instances Where A *Pirate Ship* Was Defeated By A *Patrol Ship*
  a. Use Cases 2, 4, 5, 6, 7
- FR7j: UX—“Evades Not Captured” Counter Must Reflect The Amount Of Instances Where A *Cargo Ship* Successfully Avoided Capture By A *Pirate Ship*
  a. Use Cases 2, 4, 5, 6, 7
- FR7k: UX—“Evades Captured” Counter Must Reflect The Amount Of Instances Where A *Cargo Ship* Attempted An Evasion Maneuver But Was Still Captured By A *Pirate Ship*
  a. Use Cases 2, 4, 5, 6, 7
- FR7l: UX—“Time Step” Counter Must Reflect The Amount Of Time Steps That Have Passed Since The Simulation Initially Began
  a. Use Cases 2, 4, 5, 6, 7

Product-Level Requirements (See A.3 For More Detailed Information)

Reliability:

- FRR1: *Cargo Ships* Have A 50% Probability Of Spawning On Left Most Column
- FRR2: *Patrol Ships* Have A 25% Probability Of Spawning On Any Grid Space In Right Most Column
- FRR3: *Pirate Ships* Have A 40% Probability Of Spawning On Any Grid Space In Bottom Most Row
- FRR4: *Cargo Ships* Must Have A 4x4 Grid Size Sensor For Detecting Incoming *Pirate Ships*
- FRR5*:* Cargo Ships Must Be Situated In The Bottom Left Grid Of The Innermost Portion Of The Surrounding Sensor Grid
- FRR6: *Cargo Ships* Must Be Able To Attempt An Evasion Maneuver If A *Pirate Ship* Enters The 4x4 Sensor Area
- FRR7: *Cargo Ships* Must Move 1 Grid Space Rightwards Every Time Step
- FRR8: *Pirate Ships* Must Move 1 Grid Space Upwards Every Time Step
- FRR9: *Patrol Ships* Must Move 2 Grid Spaces Leftwards Every Time Step
- FRR10: All Ship Counters Must Initialize To 0 In Default State
- FRR11: Time Step Counter Must Initialize To 0 In Default State
- FRR12: Simulation Must Offer A Way To Change All Ship Spawning Probabilities
- FRR13: All Ships Must Have An Obvious Front Of Ship Pointed In Direction Of Movement
- FRR14: *Captured Ships* Must Be Accompanied By The Pirate Ship That Captured Them In Same Grid Space
- FRR15: *Cargo Ships* Cannot Be Captured By The Same Pirate Twice
- FRR16: *Patrol Ships* Must Have A 3x3 Sensor To Detect *Pirate Ships—Patrol Ship* Is Located In Center Of Sensor
- FRR17: *Patrol Ships* Perform An Attack On Pirate Ships If Adjacent (Including Diagonal) To *Pirate Ship*
- FRR18*: Patrol Ships* Perform A Rescue On Captured Cargo Ships If Adjacent (Including Diagonal) To *Pirate Ship/Cargo Ship* Pair—If Successful *Pirate* Is Removed From Simulation
- FRR19: Defeated *Pirate Ships* Must Disappear From Simulation In Next Time Step
- FRR20: *Pirate Ships* Must Include A Box Collider To Allow *Cargo Ships* To Detect Them
- FRR21: Pirates Can Only Capture Cargos If After Both Ships Have Moved They Occupy The Same Grid Space

Portability:

- PTR1*:* Simulation Must Execute Properly On A Windows System
- PTR2: Simulation Must Execute Properly On A Macintosh System

Performance:

- PER1: Simulation Must Use Simple Objects For All Ships
- PER2: Buttons Must Be Responsive And Timely
- PER3: Simulation Must Display At A Minimum 1FPS For Correct Motion To Display On Screen
- PER4: Simulation Must Display At A Maximum 20FPS For Correct Motion To Display On Screen
- PER5: Ships Must Spawn In Following Order: Cargo, Pirate, Patrol At The Beginning Of Each Time Step
- PER6: When Any Type Of Ship Spawns, Spawning Is Considered That Ships Turn For The Current Time Step
- PER7: If (Other Than The Described Behaviors For Attack, Rescue, And Capture) Two Ships Are Within The Same Grid Or Adjacent Grids There Should Be No Action Taken
- PER8: Simulation Can Adjust Ship Spawning Probabilities For Calibration

Backlog

- **TSK-01**: Setup Google Drive—Create A Shared Google Drive For Team Documentation
  a. Requirement ID: NA
  b. Assigned To: Nathan Moore
  c. Open Date: 8/19/2021
  d. Closed Date: 8/21/2021
- **TSK-02**: Notes On Statement(s) of Work—Read Statement(s) of Work and Take Notes
  a. Requirement ID: NA
  b. Assigned To: All Team Members
  c. Open Date: 8/19/2021
  d. Closed Date: 8/26/2021
- **TSK-03**: Setup Jira Repository—Create A Jira Repository To Host Project Timeline
  a. Assigned To: Luke Tomlinson
  b. Requirement ID: NA
  c. Open Date: 8/19/2021
  d. Closed Date: 8/20/2021
- **TSK-04**: C#/Unity Training—Watch Antarsoft Youtube Unity Training Videos
  a. Requirement ID: NA
  b. Assigned To: All Team Members
  c. Open Date: 8/19/2021
  d. Closed Date: 8/26/2021

- **TSK-05**: Google Group Calendar—Setup Google Group Calendar For Personnel Meetings
  a. Requirement ID: NA
  b. Assigned To: All Team Members
  c. Open Date: 8/26/2021
  d. Closed Date: 8/31/2021
- **TSK-06**: Project Schedule—Review Project and Import Epics, User Stories, Begin Backlog on Jira
  a. Requirement ID: NA
  b. Assigned To: Nathan Moore
  c. Open Date: 8/26/2021
  d. Closed Date: 9/06/2021
- **TSK-07**: Individual Risk Assessment—Team Members Should Turn In Individual Risks To Nathan Moore for Import to Weekly Reports
  a. Requirement ID: NA
  b. Assigned To: All Team Members
  c. Open Date: 8/26/2021
  d. Closed Date: 8/30/2021
- **TSK-08**: Team Personnel Information—Team Members Should Update Their Information In Weekly Report Documentation and Jira
  a. Requirement ID: NA
  b. Assigned To: All Team Members
  c. Open Date: 8/26/2021
  d. Closed Date: 8/29/2021
- **TSK-09**: Jira Training—Team Members Should Read Resources Supplied By Dr. Delugach For Jira Basic Information
  a. Requirement ID: NA
  b. Assigned To: All Team Members
  c. Open Date: 8/26/2021
  d. Closed Date: 8/29/2021
- **TSK-10**: Software Development Plan Review—Team Members Need To Review SDP Before Submission to Customer
  a. Requirement ID: NA
  b. Assigned To: All Team Members
  c. Open Date: 8/31/2021
  d. Closed Date: 9/9/2021
- **TSK-11**: Software Development Plan Final Draft—Team Needs A Final Deliverable Of SDP Created
  a. Requirement ID: NA
  b. Assigned To: Nathan Moore
  c. Open Date: 8/31/2021
  d. Closed Date: 9/08/2021
- **TSK-12**: GitHub Repo—Team Needs Version Control Software Implemented
  a. Requirement ID: NA

   b. Assigned To: Nathan Moore
   c. Open Date: 8/31/2021
   d. Closed Date: 9/06/2021
- **TSK-13**: Level Of Effort Documentation—Team Needs Personnel Biographies for SDP
   a. Requirement ID: NA
   b. Assigned To: All Team Members
   c. Open Date: 8/31/2021
   d. Closed Date: 9/8/2021
- **TSK-14**: Formal Requirements Draft—Create A Draft Of All Application Formal Requirements
   a. Requirement ID: NA
   b. Assigned To: Jack Bragg/Nathan Moore
   c. Open Date: 9/7/2021
   d. Closed Date: **TBA**
- **TSK-15**: Update Statement Of Work—Combine Accepted Changes From Discrepancies Found In Original Statements Of Work Supplied By Customer
   a. Requirement ID: NA
   b. Assigned To: Jack Bragg
   c. Open Date: 9/08/2021
   d. Closed Date: 9/14/2021
- **TSK-16**: SRS Documentation Final Draft—Team Needs Statement of Work, Functional/Non-functional Requirements, Backlog Added To SRS Documentation and Formatted Correctly
   a. Requirement ID: NA
   b. Assigned To: Nathan Moore
   c. Open Date: 9/10/2021
   d. Closed Date: **TBA**
- **TSK-17**: Backlog Population—Team Needs A Backlog Of All Tasks Required To Complete Development On Time
   a. Requirement ID: NA
   b. Assigned To: Luke Tomlinson/Nathan Moore
   c. Open Date: 9/10/2021
   d. Closed Date: **TBA**
- **TSK-18**: Product (Non-Functional) Level Requirements—Team Needs All Product Level Requirements Found in Statement Of Work Need To Be Logged In Software Requirements Specifications Documentation
   a. Requirement ID: NA
   b. Assigned To: Aaron Mendez/Nathan Moore
   c. Open Date: 9/10/2021
   d. Closed Date: **TBA**
- **TSK-19**: Create Patrol Ship Object and Movement Function—Create A Patrol Ship That Instantiates and Moves Correctly In The Simulation
   a. Requirement ID: FR2b, FR6d
   b. Assigned To: Nathan Moore
   c. Open Date: 9/10/2021

   d. Closed Date: 9/16/2021
- **TSK-20**: Create GUI Parent—Simulation Needs A GUI Interface To Interact With Users
   a. Requirement ID: FR1
   b. Assigned To: Nathan Moore
   c. Open Date: 9/10/2021
   d. Closed Date: 9/26/2021
- **TSK-21**: Create Pirate Ship and Movement Function—Create A Pirate Ship That Instantiates and Moves Correctly In The Simulation
   a. Requirement ID: FR2c, FR6c
   b. Assigned To: Jack Bragg
   c. Open Date: 9/10/2021
   d. Closed Date: 9/16/2021
- **TSK-22**: Create Cargo Ship and Movement Function—Create A Cargo Ship That Instantiates and Moves Correctly In The Simulation
   a. Requirement ID: FR2b, FR6a
   b. Assigned To: Jack Bragg
   c. Open Date:9/10/2021
   d. Closed Date: 9/16/2021
- **TSK-23**: Create Pause Button—Create A UI Button Titled “Pause”
   a. Requirement ID: FR3b
   b. Assigned To: Nathan Moore
   c. Open Date: 9/10/2021
   d. Closed Date: 9/18/2021
- **TSK-24**: Create Pirates Entered Counter—Create A UI Counter Image Object For Amount of Pirates Ever Displayed in Simulation
   a. Requirement ID: FR4g
   b. Assigned To: Aaron Mendez
   c. Open Date: 9/10/2021
   d. Closed Date: **TBA**
- **TSK-25**: Create Single Step Button—Create A UI Button Titled “Single Step”
   a. Requirement ID: FR3g
   b. Assigned To: Nathan Moore
   c. Open Date: 9/10/2021
   d. Closed Date: 9/18/2021
- **TSK-26**: Create Captured Counter—Create A UI Counter Image Object For Total Amount Of Captured Ships Ever In Simulation
   a. Requirement ID: FR4c
   b. Assigned To: Luke Tomlinson
   c. Open Date: 9/10/2021
   d. Closed Date: **TBA**
- **TSK-27**: Create Simulation Speed Buttons—Create UI Buttons Titled “1X,2X,10X,20X Speed” Respectively
   a. Requirement ID: FR3c, FR3d, FR3e, FR3f

    b. Assigned To: Nathan Moore
    c. Open Date:9/10/2021
    d. Closed Date: 9/16/2021
- **TSK-28:** Create Cargo Entered Counter—Create UI Counter Image Object For Total Amount of Cargo Ships Ever In Simulation
    a. Requirement ID: FR4a
    b. Assigned To: Nathan Moore
    c. Open Date: 9/10/2021
    d. Closed Date: **TBA**
- **TSK-29:** Create Start Button—Create UI Button Titled “Start”
    a. Requirement ID: FR3a
    b. Assigned To: Nathan Moore
    c. Open Date: 9/10/2021
    d. Closed Date: 9/18/2021
- **TSK-30:** Resize Viewport—Viewport For Simulation Needs to Be Letterbox 16x9 in 1080P
    a. Requirement ID: FR1
    b. Assigned To: Nathan Moore
    c. Open Date: 9/18/2021
    d. Closed Date: 9/18/2021
- **TSK-31:** Resize Gridded Map—Grids Of Map and Ships Need To Be Same Size
    a. Requirement ID: FR1
    b. Assigned To: Nathan Moore
    c. Open Date: 9/23/2021
    d. Closed Date: **TBA**
- **TSK-32:** Camera Zoom Function—User Needs To Be Able To Focus FOV To Specific Areas Of Map
    a. Requirement ID: NA
    b. Assigned To: Nathan Moore
    c. Open Date: 9/23/2021
    d. Closed Date: **TBA**
- **TSK-33:** Sources Of Requirements—SRS Needs To Cite Statements Of Work And Correspondence Between Development Team And Customer
    a. Requirement ID: NA
    b. Assigned To: Nathan Moore
    c. Open Date: 9/18/2021
    d. Closed Date: **TBA**
- **TSK-34:** Create End Simulation Button—Create A UI Button Titled “End”
    a. Requirement ID: FR3h
    b. Assigned To: Nathan Moore
    c. Open Date: 09/10/2021
    d. Closed Date: 09/18/2021
- **TSK-35:** Create Cargos Exited Counter—Create A UI Counter Image To Reflect Amount Of Cargos That Have Successfully Exited Simulation

    a. Requirement ID: FR4b
    b. Assigned To: **TBA**
    c. Open Date: **TBA**
    d. Closed Date: **TBA**
- **TSK-36:** Create Cargos Captured Counter—Create A UI Counter Image To Reflect Amount Of Cargos That Have Been Captured By Pirates
    a. Requirement ID: FR4c
    b. Assigned To: **TBA**
    c. Open Date: **TBA**
    d. Closed Date: **TBA**
- **TSK-37:** Create Captured Cargos Rescued Counter—Create A UI Counter Image To Reflect Amount Of Captured Ships That Have Been Rescued By Patrols
    a. Requirement ID: FR4d
    b. Assigned To: **TBA**
    c. Open Date: **TBA**
    d. Closed Date: **TBA**
- **TSK-38:** Create Patrols Entered Counter—Create A UI Counter Image To Reflect Amount Of Patrols That Have Ever Been Displayed In Simulation
    a. Requirement ID: FR4e
    b. Assigned To: **TBA**
    c. Open Date: **TBA**
    d. Closed Date: **TBA**
- **TSK-39:** Create Patrols Exited Counter—Create A UI Counter Image To Reflect Amount Of Patrols That Have Successfully Exited The Simulation
    a. Requirement ID: FR4f
    b. Assigned To: **TBA**
    c. Open Date: **TBA**
    d. Closed Date: **TBA**
- **TSK-40:** Create Pirates Entered Counter—Create A UI Counter Image To Reflect Amount Of Pirates That Have Ever Been Displayed In Simulation
    a. Requirement ID: FR4g
    b. Assigned To: **TBA**
    c. Open Date: **TBA**
    d. Closed Date: **TBA**
- **TSK-41:** Create Pirates Exited Counter—Create A UI Counter Image To Reflect Amount Of Pirates That Have Successfully Exited The Simulation
    a. Requirement ID: FR4h
    b. Assigned To: **TBA**
    c. Open Date: **TBA**
    d. Closed Date: **TBA**
- **TSK-42:** Create Patrols Defeated Counter—Create A UI Counter Image To Reflect Amount Of Patrols That Have Successfully Defeated A Pirate Ship
    a. Requirement ID: FR4i

   b. Assigned To: **TBA**
   c. Open Date: **TBA**
   d. Closed Date: **TBA**
- **TSK-43:** Create “Evades Not Captured:” Counter—Create A UI Counter Image To Reflect Amount Of Cargo Ships That Successfully Evaded A Pirate Attack
   a. Requirement ID: FR4j
   b. Assigned To: **TBA**
   c. Open Date: **TBA**
   d. Closed Date: **TBA**
- **TSK-44:** Create “Evades Captured:” Counter—Create A UI Counter Image To Reflect Amount Of Cargo Ships That Captured In A Pirate Attack After Attempting To Evade
   a. Requirement ID: FR4k
   b. Assigned To: **TBA**
   c. Open Date: **TBA**
   d. Closed Date: **TBA**
- **TSK-45:** Create Time Step Counter—Create A UI Counter Image To Reflect Amount Time Steps Since The Start Of The Simulation
   a. Requirement ID: FR4l
   b. Assigned To: **TBA**
   c. Open Date: **TBA**
   d. Closed Date: **TBA**
- **TSK-46:** Implement Start Button Function—Create A UX Script That Allows The Start Button To Toggle From a *Paused State* to a *Running State*
   a. Requirement ID: FR5a
   b. Assigned To: **TBA**
   c. Open Date: **TBA**
   d. Closed Date: **TBA**
- **TSK-47:** Implement Pause Button Function—Create A UX Script That Allows The Start Button To Toggle From *Running State* to a *Paused State*
   a. Requirement ID: FR5b
   b. Assigned To: **TBA**
   c. Open Date: **TBA**
   d. Closed Date: **TBA**
- **TSK-48:** Implement 1X Speed Button Function—Create A UX Script That Allows The 1X Speed Button To Toggle From Any Other Time Step Speed To 1X Per Second
   a. Requirement ID: FR5c
   b. Assigned To: **TBA**
   c. Open Date: **TBA**
   d. Closed Date: **TBA**
- **TSK-49:** Implement 2X Speed Button Function—Create A UX Script That Allows The 2X Speed Button To Toggle From Any Other Time Step Speed To 2X Per Second
   a. Requirement ID: FR5d
   b. Assigned To: **TBA**

    c. Open Date: **TBA**
    d. Closed Date: **TBA**
- **TSK-50:** Implement 10X Speed Button Function—Create A UX Script That Allows The 10X Speed Button To Toggle From Any Other Time Step Speed TO 10X Per Second
    a. Requirement ID: FR5e
    b. Assigned To: **TBA**
    c. Open Date: **TBA**
    d. Closed Date: **TBA**
- **TSK-51:** Implement 20X Speed Button Function—Create A UX Script That Allows The 20X Speed Button To Toggle From Any Other Time Step Speed To 20X Per Second
    a. Requirement ID: FR5f
    b. Assigned To: **TBA**
    c. Open Date: **TBA**
    d. Closed Date: **TBA**
- **TSK-52:** Implement Single Step Button Function—Create A UX Script That Allows The Single Step Button To Toggle From *Paused State* To *Running State* For A Single(1) 1X Speed Time Step No Matter The Selected Speed; Does Nothing If Simulation Is In *Running State*
    a. Requirement ID: FR5g
    b. Assigned To: **TBA**
    c. Open Date: **TBA**
    d. Closed Date: **TBA**
- **TSK-53:** Implement End Simulation Button Function—Create A UX Script That Allows The End Simulation To Clear The Simulation And Reset To *Default State* From Either *Running State* Or *Paused State*
    a. Requirement ID: FR5h
    b. Assigned To: **TBA**
    c. Open Date: **TBA**
    d. Closed Date: **TBA**
- **TSK-54:** Captured Ship Spawn—Create A UX Script That Allows The Cargo Ships To Change From A *Captured Ship Object* To A *Captured Ship Type* And Update Movement Function To Move Downwards
    a. Requirement ID: FR6b
    b. Assigned To: **TBA**
    c. Open Date: **TBA**
    d. Closed Date: **TBA**
- **TSK-55:** Implement Cargos Entered Function—Create A UX Script That Updates The Cargos Entered UI Element To Reflect The Total Number Of Cargo Ships That Have Ever Entered The Simulation
    a. Requirement ID: FR7a
    b. Assigned To: **TBA**
    c. Open Date: **TBA**
    d. Closed Date: **TBA**

- **TSK-56:** Implement Cargos Exited Function—Create A UX Script That Updates The Cargos Exited UI Element To Reflect The Total Number Of Cargo Ships That Have Successfully Exited The Simulation
  a. Requirement ID: FR7b
  b. Assigned To: **TBA**
  c. Open Date: **TBA**
  d. Closed Date: **TBA**
- **TSK-57:** Implement Cargos Captured Function—Create A UX Script That Updates The Cargos Captured UI Element To Reflect The Total Number Of Cargo Ships That Have Ever Been Captured By A Pirate Ship
  a. Requirement ID: FR7c
  b. Assigned To: **TBA**
  c. Open Date: **TBA**
  d. Closed Date: **TBA**
- **TSK-58:** Implement Captured Cargos Rescued Function—Create A UX Script That Updates The Cargos Rescued UI Element To Reflect The Total Number Of Captured Cargo Ships That Have Been Successfully Rescued By A Patrol Ship
  a. Requirement ID: FR7d
  b. Assigned To: **TBA**
  c. Open Date: **TBA**
  d. Closed Date: **TBA**
- **TSK-59:** Implement Patrols Entered Function—Create A UX Script That Updates The Patrols Entered UI Element To Reflect The Total Number Of Patrol Ships That Have Ever Entered The Simulation
  a. Requirement ID: FR7e
  b. Assigned To: **TBA**
  c. Open Date: **TBA**
  d. Closed Date: **TBA**
- **TSK-60:** Implement Patrols Exited Function—Create A UX Script That Updates The Patrols Exited UI Element To Reflect The Total Number Of Patrol Ships That Have Successfully Exited The Simulation
  a. Requirement ID: FR7f
  b. Assigned To: **TBA**
  c. Open Date: **TBA**
  d. Closed Date: **TBA**
- **TSK-61:** Implement Pirates Entered Function—Create A UX Script That Updates The Pirates Entered UI Element To Reflect The Total Number Of Pirate Ships That Have Ever Entered The Simulation
  a. Requirement ID: FR7g
  b. Assigned To: **TBA**
  c. Open Date: **TBA**
  d. Closed Date: **TBA**

- **TSK-62:** Implement Pirates Exited Function—Create A UX Script That Updates The Pirates Exited UI Element To Reflect The Total Number Of Pirate Ships That Have Successfully Exited The Simulation
  a. Requirement ID: FR7h
  b. Assigned To: **TBA**
  c. Open Date: **TBA**
  d. Closed Date: **TBA**
- **TSK-63:** Implement Pirates Defeated Function—Create A UX Script That Updates The Pirates Defeated UI Element To Reflect The Total Number Of Defeated Pirate Ships That Have Ever Displayed In The Simulation
  a. Requirement ID: FR7i
  b. Assigned To: **TBA**
  c. Open Date: **TBA**
  d. Closed Date: **TBA**
- **TSK-64:** Implement Evades Not Captured Function—Create A UX Script That Updates The Evades Not Captured UI Element To Reflect The Total Number Of Cargo Ships That Have Successfully Evaded A Pirate Ship Attack
  a. Requirement ID: FR7j
  b. Assigned To: **TBA**
  c. Open Date: **TBA**
  d. Closed Date: **TBA**
- **TSK-65:** Implement Evades Captured Function—Create A UX Script That Updates The Evades Not Captured UI Element To Reflect The Total Number Of Cargo Ships That Have Failed An Attempt To Evade A Pirate Ship Attack And Have Been Captured
  a. Requirement ID: FR7k
  b. Assigned To: **TBA**
  c. Open Date: **TBA**
  d. Closed Date: **TBA**
- **TSK-66:** Implement Time Step Function—Create A UX Script That Updates The Time Step UI Element To Reflect The Total Number Of Time Steps That Have Occurred Since The Simulation Began
  a. Requirement ID: FR7l
  b. Assigned To: **TBA**
  c. Open Date: **TBA**
  d. Closed Date: **TBA**
- **TSK-67:** Cargo Spawn Probability—Edit The Cargo Ship Movement Script To Include A Switch To Spawn On Any Grid On The Left-most Column At Different Probabilities
  a. Requirement ID: FRR1
  b. Assigned To: **TBD**
  c. Open Date: **TBD**
  d. Closed Date: **TBD**
- **TSK-68:** Patrol Spawn Probability—Edit The Patrol Ship Movement Script To Include A Switch To Spawn On Any Grid On The Right-most Column At Different Probabilities

a. Requirement ID: FRR2
b. Assigned To: **TBA**
c. Open Date: **TBA**
d. Closed Date: **TBA**

- **TSK-69:** Pirate Spawn Probability—Edit The Pirate Ship Movement Script To Include A Switch To Spawn On Any Grid On The Bottom-most Row At Different Probabilities
  a. Requirement ID: FRR3
  b. Assigned To: **TBA**
  c. Open Date: **TBA**
  d. Closed Date: **TBA**
- **TSK-70:** Cargo Ship Collider—Edit The Cargo Ship UI Element To Include A Collider The Size of 4 Grids Horizontal and 4 Grids Vertical, Ship Sits In Bottom Left Grid Of Inner Area

	HERE		

Figure 3: Cargo Collider

  a. Requirement ID: FRR4, FRR5
  b. Assigned To: **TBA**
  c. Open Date: **TBA**
  d. Closed Date: **TBA**
- **TSK-71:** Cargo Evade Ability—Edit The Cargo Ship Collider Within The Cargo Movement Script To Include Pirate Ship Collision Detection And Attempt An Evade Movement To The Next Up-ward and Right-ward Grid Space
  a. Requirement ID: FRR6
  b. Assigned To: **TBA**
  c. Open Date: **TBA**
  d. Closed Date: **TBA**
- **TSK-72:** Cargo Movement Speed—Edit The Cargo Ship Movement Script To Have The Cargo Ship Move Exactly One Grid Space Per Time Step
  a. Requirement ID: FRR7
  b. Assigned To: **TBA**
  c. Open Date: **TBA**
  d. Closed Date: **TBA**
- **TSK-73:** Patrol Movement Speed—Edit The Patrol Ship Movement Script To Have The Patrol Ship Move Exactly Two Grid Spaces Per Time Step
  a. Requirement ID: FRR9
  b. Assigned To: **TBA**
  c. Open Date: **TBA**
  d. Closed Date: **TBA**

- **TSK-74:** Pirate Movement Speed—Edit The Pirate Ship Movement Script To Have The Pirate Ship Move Exactly One Grid Space Per Time Step
  a. Requirement ID: FRR8
  b. Assigned To: **TBA**
  c. Open Date: **TBA**
  d. Closed Date: **TBA**
- **TSK-75:** Time Counter Initialization—Edit Time Counter Script To Display ‘0’ As Their Default Value in the *Default State*
  a. Requirement ID: FRR11
  b. Assigned To: **TBA**
  c. Open Date: **TBA**
  d. Closed Date: **TBA**
- **TSK-76:** Ship Counter Incrementation—Edit Each Ship Counter Script To Display ‘0’ As Their Default Value In The *Default State*
  a. Requirement ID: FRR10
  b. Assigned To: **TBA**
  c. Open Date: **TBA**
  d. Closed Date: **TBA**
- **TSK-77:** Ship Directionality—Edit Each Ship UI Object To “Face” The Direction They Are Currently Moving
  a. Requirement ID: FRR13
  b. Assigned To: **TBA**
  c. Open Date: **TBA**
  d. Closed Date: **TBA**
- **TSK-78:** Captured Ship/Pirate Ship Accompaniment—Edit The Pirate Ship Script To Attach Its Location To The Captured Cargo Ship It Has Successfully Captured
  a. Requirement ID: FRR14
  b. Assigned To: **TBA**
  c. Open Date: **TBA**
  d. Closed Date: **TBA**
- **TSK-79:** Cargo Re-Capture Attempt—Edit The Cargo Ship Script To Include A List Of Ships That It Has Previously Been Captured By And Not Allow These Ships To Attempt To Capture This Ship Again
  a. Requirement ID: FRR15
  b. Assigned To: **TBA**
  c. Open Date: **TBA**
  d. Closed Date: **TBA**
- **TSK-80:** Patrol Ship Collider—Edit The Patrol Ship UI To Include A 3x3 Grid Size Box Collider
  a. Requirement ID: FRR16
  b. Assigned To: **TBA**
  c. Open Date: **TBA**
  d. Closed Date: **TBA**

- **TSK-81:** Patrol Attack Maneuver—Edit The Patrol Ship Script To Include An Action To Attack A Pirate Ship If The Pirate Ship Enters The Collider Space
  a. Requirement ID: FRR16
  b. Assigned To: **TBA**
  c. Open Date: **TBA**
  d. Closed Date: **TBA**
- **TSK-82:** Patrol Capture Rescue Maneuver—Edit The Patrol Ship Script To Include An Action To Rescue A Captured Cargo Ship If The Ship Enters The Collider Space
  a. Requirement ID: FRR17
  b. Assigned To: **TBA**
  c. Open Date: **TBA**
  d. Closed Date: **TBA**
- **TSK-83:** Pirate Defeat—Edit The Pirate Ship Script To Have A Check If A Successful Attack From A Patrol Has Occurred, If So Remove The Object From The Scene
  a. Requirement ID: FFR18
  b. Assigned To: **TBA**
  c. Open Date: **TBA**
  d. Closed Date: **TBA**
- **TSK-84:** Test On Windows Platform—Application Should Be Thoroughly Tested On A Windows Device And Run Properly
  a. Requirement ID: PTR1
  b. Assigned To: **TBA**
  c. Open Date: **TBA**
  d. Closed Date: **TBA**
- **TSK-85:** Test On Macintosh Platform—Application Should Be Thoroughly Tested On A Macintosh Device And Run Properly
  a. Requirement ID: PTR2
  b. Assigned To: **TBA**
  c. Open Date: **TBA**
  d. Closed Date: **TBA**

Summary—Software Requirement Specifications

The Software Requirements Specifications consists of five main sections:

1. Sources Of Requirements
2. Use Cases
3. Functional Requirements
4. Product Level Requirements
5. Task Backlog

The Sources Of Requirements portion gives a brief detail of all Statements of Work delivered and created by Team E as well as correspondence with the customer for specification of desired

product attributes. This section also creates citations for all Statements Of Work found in Appendix A.

The Use Cases describes all possible interactions that the end user can have with the Somalian Pirate Simulator in detail. These Use Case descriptions are broken down into all actions that must be taken by the system if/when a specific action, or sequence of actions, is performed by the user.

The Functional Requirements lists and briefly describes all requirements that can be tied to a specific use case and are integral to system. These requirements cite specific use cases from which they were derived for reference.

The Product Level Requirements lists and briefly describes all requirements that cannot easily be tied to a use case but can affect the reliability, performance, or portability of the Somalian Pirate Simulation. This section is directly tied to the approved, combined Statement of Work found in Appendix A Section 3.

The Task Backlog section lists in detail all foreseeable UI and UX tasks that are required for both functional requirements and product level requirements to be fulfilled. The backlog also contains a number of tasks assigned to team members which were integral for team performance and product development; they have no functional or product level requirement. The Backlog also includes logs of all completion dates for previously accomplished tasks.

Architectural Design

The following section details the architectural design for Somalian Pirate Simulation. The architectural design portion provides a breakdown of the systems main components into their respective sub-components, how information is stored and shared throughout the application, and lists constraints imposed upon the system to provide an optimal runtime experience. This section also provides details on how the user interface is provided and managed by the system.

Main System Components

The Somalian Pirate Simulator is made up of three main components:

- The Grid
  - Type: GameObject
  - Functionality:
    - Displays the 100x400 featureless map
    - Controls Spawning of the three initial ship types (Cargo, Patrol, Pirate)
    - Increments all counters for all ships and their respective interactions
- The UI Canvas
  - Type: UIObject
  - Functionality:
    - Displays the eight user command keys
    - Displays the # counters for all ship interactions and ships
    - Allows the User to interact with the simulation
    - Allows display of controls to always be displayed in the viewport
- The Camera
  - Type: CameraObject
  - Functionality:
    - Allows the user to view the map in an orthographic setting
    - Allows the user to zoom and pan the span of the map for closely observing interactions

Main Sub-Components

- The Grid is divided into eight main sub-components
  - CargoShip
    - Type: GameObject
    - Functionality:
      - The ship object is comprised of a simple shape with directionality just smaller than one grid block size, two movement target location transforms, and seven sensor transforms for detecting interactions with Pirate type ships
      - Ship moves one grid space rightward per single time step

        - When a pirate ship that has not been detected previously is detected by a sensor transform the Cargo ship has a #% chance of being captured by the Pirate and spawning a Captured type ship
    - CapturedShip
        - Type: GameObject
        - Functionality:
            - The ship object is comprised of a simple shape with directionality just smaller than one grid block size, and one movement target location transform
            - Ship moves one grid space downward per time step
    - PirateShip
        - Type: GameObject
        - Functionality:
            - The ship object is comprised of a simple shape with directionality just smaller than one grid block size, one movement target location transform
            - Ship moves one grid space upward per single time step
    - PatrolShip
        - Type: GameObject
        - Functionality:
            - The ship object is comprised of a simple shape with directionality just smaller than one grid block size, one movement target location transform, and nine sensor transforms for detecting interactions with Pirate and Captured type ships
            - Default movement is one grid space leftward per single time step
    - CargoSpawnTransforms
        - Type: List<Transforms>
        - Functionality:
            - Display possible spawning locations of all incoming Cargo ships on the leftmost column of the grid area
            - 100 possible grid space locations exist on map
    - PirateSpawnTransforms
        - Type: List<Transforms>
        - Functionality:
            - Display possible spawning locations of all incoming Pirate ships on the bottom-most row of the grid area
            - 400 possible grid space locations exist on map
    - PatrolSpawnTransforms
        - Type: List<Transforms>
        - Functionality:
            - Display possible spawning locations of all incoming Patrol ships on the rightmost column of the grid area
            - 100 possible grid space locations exist on map
- The UI Canvas is divided into nine main components in three panels
    - The Speed Control Button Panel contains four main components

- Speed 1x
  - Type: UI Button
  - Functionality:
    - When pressed from a running state this button will set the simulation speed to 1 time step per second
    - When pressed from a paused state this button will set the simulation speed to 1 time step per second
- Speed 2x Button
  - Type: UI Button
  - Functionality:
    - When pressed from a running state this button will set the simulation speed to 2 time steps per second
    - When pressed from a paused state this button will set the simulation speed to 2 time steps per second
- Speed 10x Button
  - Type: UI Button
  - Functionality:
    - When pressed from a running state this button will set the simulation speed to 10 time steps per second
    - When pressed from a paused state this button will set the simulation speed to 10 time steps per second
- Speed 20x Button
  - Type: UI Button
  - Functionality:
    - When pressed from a running state this button will set the simulation speed to 20 time steps per second
    - When pressed from a paused state this button will set the simulation speed to 20 time steps per second

- Runtime Manipulation Button Panel contains four main components
  - Start Button
    - Type: UI Button
    - Functionality:
      - When pressed from a paused state this button will begin/resume the simulation
      - When pressed from a running state this button will have no upon the simulation
  - Pause Button
    - Type: UI Button
    - Functionality:
      - When pressed from a running state this button with pause the simulation
      - When pressed from a paused state this button will have no effect upon the simulation
  - Single Step Button
    - Type: UI Button

- Functionality:
  - When pressed from a paused state this button will cause the simulation to resume for one 1x speed time step then re-enter a paused state
  - When pressed from a running state this button will have no effect upon the simulation
- End Simulation Button
  - Type: UI Button
  - Functionality:
    - When pressed during either the Running or Paused state this button will called the Application.Quit() command and close the current window running The Somalian Pirate Simulation

- Ship/Interaction Counter Text Pane Panel contains nine main components
  - Cargo Entered Counter
    - Type: UI Text
    - Functionality: When a Cargo Ship enters the simulation this counter will increment by 1
  - Patrol Entered Counter
    - Type: UI Text
    - Functionality: When a Patrol Ship enters the simulation this counter will increment by 1
  - Pirate Entered Counter
    - Type: UI Text
    - Functionality: When a Pirate Ship enters the simulation this counter will increment by 1
  - Cargo Exited Counter
    - Type: UI Text
    - Functionality: When a Cargo Ship exits the simulation this counter will increment by 1
  - Patrol Exited Counter
    - Type: UI Text
    - Functionality: When a Patrol Ship exits the simulation this counter will increment by 1
  - Pirate Exited Counter
    - Type: UI Text
    - Functionality: When a Pirate Ship exits the simulation this counter will increment by 1
  - Cargo Captured Counter
    - Type: UI Text
    - Functionality: When a Pirate Ship successfully captures a Cargo Ship this counter will increment by 1
  - Captured Cargos Rescued Counter
    - Type: UI Text

        - Functionality: When a Patrol Ship successfully rescues a Captured Cargo Ship this counter will increment by 1
    - Pirates Defeated Counter
        - Type: UI Text
        - Functionality: When a Patrol Ship defeats a Pirate Ship this counter will increment by 1
    - Evades Not Captured Counter
        - Type: UI Text
        - Functionality:
            - When a Cargo Ship performs an evade maneuver for the first time and is not captured this counter will increment by 1
            - If the same Cargo Ship performs an evade on a separate ship this counter will not increment
            - If the Cargo Ship previously performed an evade maneuver and is subsequently captured by a Pirate Ship this counter will decrement by 1
    - Evades Captured Counter
        - Type: UI Text
        - Functionality: When a Cargo Ship performs an evade maneuver at any point during simulation runtime where it is captured by a Pirate Ship this counter will increment by one
    - Time Step Counter
        - Type: UI Text
        - Functionality:
            - When the Time Step Speed is set to 1x this counter will increment by 1 every second
            - When the Time Step Speed is set to 2x this counter will increment 2 distinct times per second
            - When the Time Step Speed is set to 10x this counter will increment 10 distinct times per second
            - When the Time Step Speed is set to 20x this counter will increment 20 distinct times per second
- The Camera contains two main script components that control user interaction with the simulation
    - Click and Drag
        - Type: Script
        - Functionality: This script provides the user with an intuitive movement style by allowing for clicking the mouse in one location and dragging to move the viewport to a different location inside of the simulation
    - Zoom With Mouse Wheel:
        - Type: Script
        - Functionality: This script provides the user with the ability to scroll the mouse wheel forward and backward to zoom in and out of interesting areas within the simulation

System Information Storage and Management

- The Somalian Pirate Simulation manages information storage in four different ways:
- Ship Instantiation and Destruction:
  - All Ship Type's instantiation is controlled by a single script attached to The Grid game object – only one script is activated during any time during simulation for controlled determination how many ships can instantiate over a given amount of time – this allows for user control over how many game objects are currently active inside of the simulation and in main memory
  - Ship destruction is controlled by a script that is attached to each ship type prefabricated objects – once ships leave a specified bound they are removed from both the simulation display and main memory.
- Ship and Interaction counters:
  - Ship and Interaction counters are cleared from main memory when the "End" button has been selected
  - Ship and Interaction counters will decrement and increment in sync with the user selected speed
- Map and spawn point transforms:
  - The gridded map and all ship spawn points are static items which always exist in memory as long as the application remains open
- Camera and UI:
  - The camera remains in main memory during the duration of application runtime
  - UI elements(buttons, text fields) remain in main memory during the duration of application runtime

Component Information Exchange and Storage

- The components of the Somalian Pirate Simulation exchange and store information in the following ways:
- Ship Interactions
  - Cargo and Patrol ship prefabricated objects contain sensors which detect assigned Layers to Pirate ships and Captured ships – when a ship is detected it is stored inside of a dynamically sized list inside of the script that controls the ships movement and behavior
  - When Cargo sensors detect a Pirate object they check if the Pirate object already exists within a list of ships they have previously interacted with and if not then add the Pirate ship ID to a list
  - When Patrol sensors detect a Pirate ship they call upon the Pirate ship to destroy itself from the simulation as well as main memory
  - When Patrol sensors detect a Captured ship they have a #% chance of freeing the Cargo ship from the Pirate
- User Interactions
  - Buttons:

- Start and Pause – When selected these buttons change a boolean value within The Grid control structure
- End – When selected this button will remove all currently displaying objects from memory as well as any background data collect and close the “Somalian Pirate Simulation” window
- Speed 1x,2x,10x,20x – When selected these buttons will change an integer value for a switch inside of The Grid control structure causes an alteration in simulation step speed

User Interface Management

User Interface with the Somalian Pirate Simulation is control by the both the Camera and it’s sub-components as well as the UI Canvas and it’s sub-components. The end user will only be able to manipulate the Somalian Pirate Simulation through use of the provided scripts for camera zoom/movement housed within the Camera main components and the buttons provided by the UI Canvas. This allows for the simulation to be manipulated in a controlled environment and stops the user from being able to accidentally manipulating the simulation in an unwanted way thus preserving data gathering integrity. Interactions with different parts of the user interface will change specific back-end variables which will allow the system to perform in the selected manner. These methods are described above in the sections titled System Information Storage and Management and Component Information Exchange and Store.

System Communication Constraints

Somalian Pirate Simulation has very specific constraints on how system interactions can occur. Following is a list of the ways each component can exchange information with other components found within the simulation.

- Cargo Ship, Pirate Ship, and Event Counter Information Exchange Protocol:
  - Cargo Ship’s have seven sensors situated one grid space from the main game object’s center – these sensors can detect when a Pirate Ship has entered the spaces surrounding the Cargo Ship triggering a boolean value switch named pirateEncounter within the Cargo Ship control script to report true.
  - If a Pirate Ship has not previously been detected and has triggered the boolean value within the Cargo Ship control script it will then be added to a list of encountered Pirate Ships within the Cargo Ship control script.
  - If the pirateEncounter value is true the Cargo Ship will attempt to perform an evade maneuver
  - If evade maneuver is successful the isEvading boolean variable triggers true
    - If evade is successful the Evades Not Captured counter will be incremented by 1
    - If at any time after the Cargo Ship encounters another pirate ship this step will be repeated
    - If at any point the Cargo Ship becomes captured by a pirate the Evades Captured Counter will increment by 1 and the Evades Not Captured will decrement by 1

    - If the evade maneuver is unsuccessful the isEvading boolean variable remains false
        - Pirate Ship will be passed the Cargo Ship transform position and move to that position
        - The Cargos Captured Counter will increment by 1
        - The Cargo Ship will call the changeToCaptured method and alter appearance and movement target position to point downwards
        - Pirate Ship will trigger a haveCaptured boolean to true and will begin to move downward
        - If the Captured Ship passes the bottom of the grid and is removed from the simulation the Pirates Exited Counter will be incremented
- Patrol Ship, Pirate Ship, and Event Counter Information Exchange Protocol:
    - Patrol Ships have nine sensors situated one grid space from the main game object's center—these sensors can detect when a Pirate Ship has entered the spaces surrounding the Patrol Ship triggering a boolean value switch named pirateEncounter within the Patrol Ship control script to report true.
    - If the pirateEncounter boolean value is triggered true that information is relayed to the Pirate Ship and the Pirate Ship then calls the destroy command upon itself.
    - The Pirates Defeated Counter will then increment by 1
- Patrol Ship, Captured Ship, and Event Counter Information Exchange Protocol:
    - Patrol Ships have nine sensors situated one grid space from the main game object's center—these sensors can detect when a Captured Ship accompanied by a Pirate Ship has entered the spaces surrounding the Patrol Ship triggering a boolean value switch named capturedEncounter within the Patrol Ship control script to report true.
    - If the capturedEncounter boolean value is triggered true that information is relayed to the Pirate Ship and the Pirate Ship then calls the destroy command upon itself and the Captured Ship to be released
        - Captured Ship calls changeToCargo method and movement target is changed back to the righthand target
        - Captured Cargos Rescued Counter is incremented by 1
        - Cargos Captured Counter is decremented by 1
        - Pirates Defeated Counter is incremented by 1
- Main Ship Type Instantiation and Counter Information Exchange Protocol:
    - Cargo Ship Instantiation:
        - The Grid contains a list of 100 static transforms that are aligned along the left-most column which are stored within The Grid Game Object's parent object.
        - The Grid control structure allows for a user determined chance of one Cargo Ship to instantiate at one grid space per time step.
        - If chance of a Cargo Ship instantiating falls within the acceptable limit the The Grid control structure then generates a random number between 1 and 100 and instantiates a new Cargo Ship Prefab.
        - The Cargos Entered Counter will increment by 1.
    - Patrol Ship Instantiation:
        - The Grid contains a list of 100 static transforms that are aligned along the right-most column which are stored within The Grid Game Object's parent object.

        - The Grid control structure allows for a user determined chance of one Patrol Ship to instantiate at one grid space per time step.
        - If chance of a Patrol Ship instantiating falls within the acceptable limit the The Grid control structure then generates a random number between 1 and 100 and instantiates a new Patrol Ship Prefab.
        - The Patrols Entered Counter will increment by 1.
    - Pirate Ship Instantiation:
        - The Grid contains a list of 400 static transforms that are aligned along the bottom-most row which are stored within The Grid Game Object's parent object.
        - The Grid control structure allows for a user determined chance of one Pirate Ship to instantiate at one grid space per time step.
        - If chance of a Pirate Ship instantiating falls within the acceptable limit the The Grid control structure then generates a random number between 1 and 100 and instantiates a new Pirate Ship Prefab.
        - The Pirate Entered Counter will increment by 1.
- Main Ship Type Exiting and Counter Information Exchange Protocol:
    - Cargo Ship Exiting:
        - The Grid contains a list of 100 static transforms that are aligned along the right-most column which are stored within The Grid Game Object's parent object.
        - If a Cargo Ship's position is equal or greater than one of these transforms x-axis positions the ship then triggers an exiting boolean value true and calls the destroy command on itself.
        - The Cargos Exited Counter will increment by 1.
    - Patrol Ship Exiting:
        - The Grid contains a list of 100 static transforms that are aligned along the right-most column which are stored within The Grid Game Object's parent object.
        - If a Patrol Ship's position is equal or less than one of these transforms x-axis positions the ship then triggers an exiting boolean value true and calls the destroy command on itself.
        - The Patrols Exited Counter will increment by 1.
    - Pirate Ship Exiting:
        - The Grid contains a list of 400 static transforms that are aligned along the bottom-most and top most rows which are stored within The Grid Game Object's parent object.
        - If a Pirate Ship's position is equal or less than one of the transforms on the bottom most row's y-axis position and it is accompanied by a Captured Ship, the ship then triggers an exiting boolean value true and calls the destroy command on itself.
            - The Pirate Exited Counter will increment by 1
        - If a Pirate Ship's position is equal or greater than one of the transforms on the top most row's y-axis position and it is not accompanied by a Captured Ship, the ship then triggers an exiting boolean value true and calls the destroy command on itself.
            - The Pirate Exited Counter will increment by 1.

Team E
Nathan Moore
Luke Tomlinson
Aaron Mendez
Jack Bragg

Testing Plan

The testing plan section provides the layout and descriptions of how the team will execute unit testing, how components will be integration tested, which team members will be responsible for these tests, descriptions of any special considerations Somalian Pirate Simulator has poses restrictions on the tests provided, and how the test data will be developed and constructed by the assigned team member as well as how the data will be shared with the reviewing team member and team as a whole.

Unit and Integration Test and Result Recording Guidelines

Team members will be required to follow specified testing rules for each individual game object or back end variable—these guidelines are divided into the following six sections based on similarity in test steps:

- Ship Counter Information Exchange Interactions
- Ship Interactions
- Ship Interaction Counter Information Exchange Interactions
- Ship Movement
- Button Interactions
- Camera Movement Interactions

These guidelines are further separated into Unit Tests and Integration Tests depending on if the tests involve a transfer of data between two or more objects within the game or if the test can be run by checking singular objects for values during the performance of the test case.

The Test Assignee will follow a strict set of steps and document results through screenshots and raw data. All Test Results from the Assignee will be uploaded to the task number associated with the test in the teams Jira Backlog and mark that task as “In Review” allowing for the Reviewee assigned to the task to receive a push notification alerting them that the test is ready for review. The Test Reviewee will then rerun the same test and compare their results to the original Assignee’s results and post their results on the associated task and mark that task as “Done”.

Unit Test Descriptions

- **Test Title:** Cargo Ship Spawn Probability Test
- **Test ID:** TSTFR1
- **Requirement ID:** FRR1
- **Assignee:** Nathan Moore
- **Reviewee:** Luke Tomlinson
- **Test Steps:**
  1. Open shipScript script and check for a switch based on key input in Cargo Ship spawn method
  2. Switch should contain the keys ‘1’,’2’,’3’ and default as cases

3. Check script for generation of a random float value between 0 and 1 being generated and saved as a variable inside the Cargo Ship spawn method
4. Default case should be set to 0.5 chance of spawning, case 1 should be equal to the default case, case 2 should be 0.75, case 3 should be 0.90
5. Create a dummy variable to set to 1 if a ship has spawned after a random number is generated
6. Have both these numbers print to the console in “Cargo: Roll: #, Spawn: #” format
7. Press Start Button in Unity
8. Press Start Button in simulation
9. Run for 30 seconds minimum in default state
10. Press the ‘2’ key on the keyboard
11. Run for 30 seconds minimum in this state
12. Press the ‘3’ key on the keyboard
13. Run for 30 seconds minimum in this state
14. Press the ‘1’ key on the keyboard
15. Run for 30 seconds minimum in this state
16. Create a copy of all information printed to the console as a .txt file
17. Report Findings

- **Date Completed:** 11/08/2021

- **Test Title:** Patrol Ship Spawn Probability Test
- **Test ID:** TSTFR2
- **Requirement ID:** FRR2
- **Assignee:** Nathan Moore
- **Reviewee:** Luke Tomlinson
- **Test Steps:**
  1. Open shipScript script and check for a switch based on key input in Cargo Ship spawn method
  2. Switch should contain the keys ‘4’,’5’,’6’ and default as cases
  3. Check script for generation of a random float value between 0 and 1 being generated and saved as a variable inside the Patrol Ship spawn method
  4. Default case should be set to 0.25 chance of spawning, case 4 should be equal to the default case, case 5 should be 0.50, case 6 should be 0.70
  5. Create a dummy variable to set to 1 if a ship has spawned after a random number is generated
  6. Have both these numbers print to the console in “Patrol: Roll: #, Spawn: #” format
  7. Press Start Button in Unity
  8. Press Start Button in simulation
  9. Run for 30 seconds minimum in default state
  10. Press the ‘2’ key on the keyboard
  11. Run for 30 seconds minimum in this state
  12. Press the ‘3’ key on the keyboard
  13. Run for 30 seconds minimum in this state

14. Press the ‘1’ key on the keyboard
15. Run for 30 seconds minimum in this state
16. Create a copy of all information printed to the console as a .txt file
17. Report Findings

- Date Completed: 11/08/2021

- **Test Title:** Pirate Ship Spawn Probability Test
- **Test ID:** TSTFR3
- **Requirement ID:** FRR3
- **Assignee:** Nathan Moore
- **Reviewee:** Luke Tomlinson
- **Test Steps:**
  1. Open shipScript script and check for a switch based on key input in Pirate Ship spawn method
  2. Switch should contain the keys ‘7’,’8’,’9’ and default as cases
  3. Check script for generation of a random float value between 0 and 1 being generated and saved as a variable inside the Pirate Ship spawn method
  4. Default case should be set to 0.40 chance of spawning, case 7 should be equal to the default case, case 8 should be 0.60, case 9 should be 0.80
  5. Create a dummy variable to set to 1 if a ship has spawned after a random number is generated
  6. Have both these numbers print to the console in “Pirate: Roll: #, Spawn: #” format
  7. Press Start Button in Unity
  8. Press Start Button in simulation
  9. Run for 30 seconds minimum in default state
  10. Press the ‘2’ key on the keyboard
  11. Run for 30 seconds minimum in this state
  12. Press the ‘3’ key on the keyboard
  13. Run for 30 seconds minimum in this state
  14. Press the ‘1’ key on the keyboard
  15. Run for 30 seconds minimum in this state
  16. Create a copy of all information printed to the console as a .txt file
  17. Report Findings
- **Date Completed:** 11/08/2021

- **Test Title:** Cargo Ship Sensor Test
- **Test ID:** TSTFR4
- **Requirement ID:** FRR4, FRR5
- **Assignee:** Nathan Moore
- **Reviewee:** Jack Bragg
- **Test Steps:**
  1. Open The Cargo Ship Prefab

2. Check for 7 Sensor Transforms
3. These Sensors should all match specified distances (see FRR5) from the center of the cargo ship object in individual grid spaces
4. Check the Cargo Ship script that it colors all of these transforms
5. Save the layout and drop a Cargo Ship Prefab on the map in the center of any grid
6. If the sensors do not match the center of the surrounding grids adjust values of Sensor Transforms in the Cargo Ship Prefab
7. Take a screenshot of the cargo ship prefab inside of the project build viewer
8. Report Findings

- **Date Completed:** 11/10/2021

- **Test Title:** Cargo Ship Move Target Test
- **Test ID:** TSTFR5
- **Requirement ID:** FRR7
- **Assignee:** Nathan Moore
- **Reviewee:** Jack Bragg
- **Test Steps:**
  1. Open The Cargo Ship Prefab
  2. Check for 1 Movement Target Transform
  3. This Sensor should be placed 1 grid space length to the right from the center of the Cargo Ship game object
  4. Check the Cargo Ship script that it colors this transform
  5. Save the layout and drop a Cargo Ship Prefab on the map in the center of any grid
  6. If the sensor does not match the center of the grid space on to the right of the Cargo Ship adjust values of Movement Target Transform in the Cargo Ship Prefab
  7. Take a screenshot of the Cargo Ship prefab inside of the project build viewer
  8. Report Findings
- **Date Completed:** 11/10/2021

- **Test Title:** Pirate Ship Move Target Test
- **Test ID:** TSTFR6
- **Requirement ID:** FRR7
- **Assignee:** Nathan Moore
- **Reviewee:** Jack Bragg
- **Test Steps:**
  1. Open The Pirate Ship Prefab
  2. Check for 1 Movement Target Transform
  3. This Sensor should be placed 1 grid space length to the top from the center of the Pirate Ship game object
  4. Check the Pirate Ship script that it colors this transform
  5. Save the layout and drop a Pirate Ship Prefab on the map in the center of any grid

6. If the sensor does not match the center of the grid space on to the right of the Pirate Ship adjust values of Movement Target Transform in the Pirate Ship Prefab
7. Take a screenshot of the Pirate Ship prefab inside of the project build viewer
8. Report Findings

- **Date Completed:** 11/10/2021

- **Test Title:** Patrol Ship Move Target Test
- **Test ID:** TSTFR8
- **Requirement ID:** FRR8
- **Assignee:** Nathan Moore
- **Reviewee:** Aaron Mendez
- **Test Steps:**
  - Open The Patrol Ship Prefab
  - Check for 1 Movement Target Transform
  - This Sensor should be placed 2 grid space length to the left from the center of the Patrol Ship game object
  - Check the Patrol Ship script that it colors this transform
  - Save the layout and drop a Patrol Ship Prefab on the map in the center of any grid
  - If the sensor does not match the center of the grid space on to the right of the Patrol Ship adjust values of Movement Target Transform in the Patrol Ship Prefab
  - Take a screenshot of the Patrol Ship prefab inside of the project build viewer
  - Report Findings
- **Date Completed:** 11/09/2021

- **Test Title:** Patrol Ship Sensor Test
- **Test ID:** TSTFR10
- **Requirement ID:** FRR16
- **Assignee:** Nathan Moore
- **Reviewee:** Aaron Mendez
- **Test Steps:**
  - Open The Patrol Ship Prefab
  - Check for 9 Sensor Transforms
  - These Sensors should be placed 1 grid space length surrounding the Patrol Ship game object
  - Check the Patrol Ship script that it colors these transforms
  - Save the layout and drop a Patrol Ship Prefab on the map in the center of any grid
  - If the sensor does not match the center of the grid space on to the right of the Patrol Ship adjust values of Sensor Target Transforms in the Patrol Ship Prefab
  - Take a screenshot of the Patrol Ship prefab inside of the project build viewer
  - Report Findings
- **Date Completed:** 11/20/2021

Integration Test Descriptions

- **Test Title:** Cargo Entered Counter Implementation Test
- **Test ID:** TSTSC-1
- **Requirement ID:** FR4a
- **Assignee:** Luke Tomlinson
- **Reviewee:** Nathan Moore
- **Test Steps:**
  1. Press the play button in the Unity Dashboard.
  2. Count 10 seconds.
  3. Press the pause button in the Unity Dashboard.
  4. Count the number of cargo ships displayed on the screen and compare with the counter label at the bottom of the GUI.
  5. Count the number of cargo ship prefabs that were built on the left hand side of the Unity window and compare with the cargo counter label at the bottom of the GUI.
- **Date Completed:** 11/07/2021

- **Test Title:** Cargo Exited Counter Implementation Test
- **Test ID:** TSTSC-2
- **Requirement ID:** FR4b
- **Assignee:** Luke Tomlinson
- **Reviewee:** Nathan Moore
- **Test Steps:**
  1. Press the play button in the Unity Dashboard.
  2. Run the program until a cargo ship has gone off the screen.
  3. As the cargos go off the screen count the number of ships removed from the screen to 5.
  4. Press the pause button in the Unity Dashboard.
  5. Compare with the cargo exited label at the bottom of the GUI.
- **Date Completed:** 11/07/2021

- **Test Title:** Cargo Capture Counter Implementation Test
- **Test ID:** TSTSC-3
- **Requirement ID:** FR4c
- **Assignee:** Luke Tomlinson
- **Reviewee:** Nathan Moore
- **Test Steps:**
  1. Inside of the function that captures the cargo, log to the console everytime the function is called.
  2. Press the play button in the Unity Dashboard.
  3. Run the program until the simulator has allowed enough pirates to capture cargo ships.
  4. Press the pause button in the Unity Dashboard.

5. Compare the console output with the captured label.

- **Date Completed:** 11/07/2021

- **Test Title:** Cargo Rescued Counter Implementation Test
- **Test ID:** TSTSC-4
- **Requirement ID:** FR4d
- **Assignee:** Luke Tomlinson
- **Reviewee:** Nathan Moore
- **Test Steps:**
  1. Create a log to the console everytime the function is called to rescue a cargo.
  2. Press the play button in the Unity Dashboard.
  3. Run the program for enough time to rescue cargo ships.
  4. Press the pause button in the Unity Dashboard.
  5. Compare the console output with the cargo rescued label.
- **Date Completed:** 11/07/2021

- **Test Title:** Patrols Entered Counter Implementation Test
- **Test ID:** TSTSC-5
- **Requirement ID:** FR4e
- **Assignee:** Luke Tomlinson
- **Reviewee:** Nathan Moore
- **Test Steps:**
  1. Press the play button in the Unity Dashboard.
  2. Count 10 seconds.
  3. Press the pause button in the Unity Dashboard.
  4. Count the number of patrol ships displayed on the screen and compare with the counter label at the bottom of the GUI.
  5. Count the number of patrol ship prefabs that were built on the left hand side of the Unity window and compare with the patrol counter label at the bottom of the GUI.
- **Date Completed:** 11/07/2021

- **Test Title:** Patrol Ship Exited Counter Implementation Test
- **Test ID:** TSTSC-6
- **Requirement ID:** FR4f
- **Assignee:** Luke Tomlinson
- **Reviewee:** Nathan Moore
- **Test Steps:**
  1. Press the play button in the Unity Dashboard.
  2. Run the program until a patrol ship has gone off the screen.

3. As the patrol ships go off the screen count the number of ships removed from the screen to 5.
4. Press the pause button in the Unity Dashboard.
5. Compare with the patrol exited label at the bottom of the GUI.

- **Date Completed:** 11/07/2021

- **Test Title:** Pirate Ship Entered Counter Implementation Test
- **Test ID:** TSTSC-7
- **Requirement ID:** FR4g
- **Assignee:** Luke Tomlinson
- **Reviewee:** Nathan Moore
- **Test Steps:**
  1. Press the play button in the Unity Dashboard.
  2. Count 10 seconds.
  3. Press the pause button in the Unity Dashboard.
  4. Count the number of pirate ships displayed on the screen and compare with the counter label at the bottom of the GUI.
  5. Press the pause button in the Unity Dashboard.
  6. Count the number of pirate ship prefabs that were built on the left hand side of the Unity window and compare with the pirate counter label at the bottom of the GUI.
- **Date Completed:** 11/07/2021

- **Test Name:** Pirate Ship Exited Counter Implementation Test
- **Test ID:** TSTSC-8
- **Requirement ID:** FR4h
- **Assignee:** Luke Tomlinson
- **Reviewee:** Nathan Moore
- **Test Steps:**
  1. Press the play button in the Unity Dashboard.
  2. Run the program until a pirate ship has gone off the screen.
  3. As the pirates go off the screen count the number of ships removed from the screen to 5.
  4. Press the pause button in the Unity Dashboard.
  5. Compare with the pirate exited label at the bottom of the GUI.
- **Date Completed:** 11/10/2021

- **Test Title:** Pirate Defeated Counter Implementation Test
- **Test ID:** TSTSC-9
- **Requirement ID:** FR4i
- **Assignee:** Luke Tomlinson
- **Reviewee:** Nathan Moore
- **Test Steps:**

1. Create a log to the console everytime the function is called to defeat a pirate.
2. Press the play button in the Unity Dashboard.
3. Run the program for enough time to defeat pirate ships.
4. Press the pause button in the Unity Dashboard.
5. Compare the console output with the pirate defeated label.

- **Date Completed:** 11/10/2021

- **Test Title:** Evades Not Captured Counter Implementation Test
- **Test ID:** TSTSC-10
- **Requirement ID:** FR4j
- **Assignee:** Luke Tomlinson
- **Reviewee:** Jack Bragg
- **Test Steps:**
  1. Drag pirates on to screen to collide with cargo.
  2. Press the play button.
  3. Wait until 3 ships have collided/evaded a pirate and check if the counter has changed.
  4. If the counter changed, count the number of cargos that evaded and compare with the label.
- **Date Completed:** 12/02/2021

- **Test Title:** Evades Captured Counter Implementation Test
- **Test ID:** TSTSC-11
- **Requirement ID:** FR4k
- **Assignee:** Luke Tomlinson
- **Reviewee:** Jack Bragg
- **Test Steps:**
  1. Drag pirates on to screen to collide with cargo.
  2. Press the play button.
  3. Wait until one of the cargos changing to a captured ship judging by the shape of the ship.
  4. Count the number of captured ships.
  5. Compare with the evade captured label.
- **Date Completed:** 12/02/2021

- **Test Title:** Time Step Counter Implementation Test
- **Test ID:** TSTSC-12
- **Requirement ID:** FR7l
- **Assignee:** Luke Tomlinson
- **Reviewee:** Nathan Moore/Jack Bragg
- **Test Steps:**
  1. Create a counter that updates an integer variable for each time step and print to console
  2. Press Play button in Unity
  3. Wait 30 seconds

4. Press Pause button in Unity
5. Compare the output of both the counter on screen in simulation and the output in the console

- **Date Completed:** 12/02/2021

- **Test Title:** Start Button Implementation Test
- **Test ID:** TSTBT1
- **Requirement ID:** FR5a
- **Assignee:** Jack Bragg
- **Reviewee:** Nathan Moore
- **Test Steps:**
  1. Check to make sure that the script for the Start Button includes a method that sets Time.timescale equal to 1 and set a dummy private variable equal to Time.timescale for debugging
  2. Click Play Button to start game in Unity
  3. Game should begin in Default (Paused) State
  4. Check for dummy variable value is equal to 0
  5. Click Start Button in simulation
  6. Check dummy variable value is equal to 1
  7. Record findings
- **Date Completed:** 12/04/2021

- **Test Title:** Pause Button Implementation Test
- **Test ID:** TSTBT2
- **Requirement ID:** FR5a, FR5b
- **Assignee:** Jack Bragg
- **Reviewee:** Aaron Mendez
- **Test Steps:**
  1. Check to make sure that the scripts for the Start Button and Pause Button include a method that sets Time.timescale equal to 1 and set a dummy private variable equal to Time.timescale for debugging
  2. Start the simulation in Unity
  3. Press the Start Button in simulation
  4. Check dummy variable for value of 1 and that simulation begins to run
  5. Press the Pause Button in simulation
  6. Check dummy variable for value of 0 and that simulation enters a paused state
  7. Record findings
- **Date Completed:** 12/04/2021

- **Test Title:** Speed 1x Button Implementation Test
- **Test ID:** TSTBT3
- **Requirement ID:** FR5a, FR5b, FR5c

- **Assignee:** Jack Bragg
- **Reviewee:** Nathan Moore
- **Test Steps:**
  1. Check to make sure that the scripts for the Start Button and Pause Button include a method that sets Time.timescale equal to 1 and set a dummy private variable equal to Time.timescale for debugging
  2. Check to make sure that the script for the Start Button includes a switch that can allow for changes to be made to the time scale and that switch case 1 sets the timescale to 1
  3. Check that 1x Speed Button has a public integer switchCase and is passed as an object to Start Button
  4. Start the simulation in Unity
  5. Click the 1x Speed Button in simulation
  6. Check integer value passed to Start Button for switch case is set to 1
  7. Check timescale speed set value
  8. Press the Start Button in simulation
  9. Check dummy variable for value of 1 and that simulation begins to run
  10. Press the Pause Button in simulation
  11. Check dummy variable for value of 0 and that simulation enters a paused state
  12. Record findings
- **Date Completed:** 12/04/2021

- **Test Title:** Speed 2x Button Implementation Test
- **Test ID:** TSTBT4
- **Requirement ID:** FR5a, FR5b, FR5c
- **Assignee:** Jack Bragg
- **Reviewee:** Nathan Moore
- **Test Steps:**
  1. Check to make sure that the scripts for the Start Button and Pause Button include a method that sets Time.timescale equal to 1 and set a dummy private variable equal to Time.timescale for debugging
  2. Check to make sure that the script for the Start Button includes a switch that can allow for changes to be made to the time scale and that switch case 2 sets the timescale to 2
  3. Check that 1x Speed Button has a public integer switchCase and is passed as an object to Start Button
  4. Start the simulation in Unity
  5. Click the 2x Speed Button in simulation
  6. Check dummy boolean value
  7. Check integer value passed to the Start Button for switch case is set to 2
  8. Check timescale speed set value
  9. Press the Start Button in simulation
  10. Check dummy variable for value of 2 and that simulation begins to run
  11. Press the Pause Button in simulation

12. Check dummy variable for value of 0 and that simulation enters a paused state
13. Record findings

- **Date Completed:** 12/04/2021

- **Test Title:** Speed 10x Button Implementation Test
- **Test ID:** TSTBT5
- Requirement ID: FR5a, FR5b, FR5c
- **Assignee:** Jack Bragg
- **Reviewee:** Nathan Moore
- **Test Steps:**
  1. Check to make sure that the scripts for the Start Button and Pause Button include a method that sets Time.timescale equal to 1 and set a dummy private variable equal to Time.timescale for debugging
  2. Create a dummy boolean value to set true when 10x Speed Button is pressed
  3. Check to make sure that the script for the Start Button includes a switch that can allow for changes to be made to the time scale and that switch case 3 sets the timescale to 10
  4. Start the simulation in Unity
  5. Click the 10x Speed Button in simulation
  6. Check dummy boolean value
  7. Check integer value for Start Button for switch case is set to 3
  8. Check timescale speed set value
  9. Press the Start Button in simulation
  10. Check dummy variable for value of 10 and that simulation begins to run
  11. Press the Pause Button in simulation
  12. Check dummy variable for value of 0 and that simulation enters a paused state
  13. Record findings
- **Date Completed:** 12/04/2021

- **Test Title:** Speed 20x Button Implementation Test
- **Test ID:** TSTBT6
- **Requirement ID:** FR5a, FR5b, FR5f
- **Assignee:** Jack Bragg
- **Reviewee:** Nathan Moore
- **Test Steps:**
  1. Check to make sure that the scripts for the Start Button and Pause Button include a method that sets Time.timescale equal to 1 and set a dummy private variable equal to Time.timescale for debugging
  2. Create a dummy boolean value to set true when 20x Speed Button is pressed
  3. Check to make sure that the script for the Start Button includes a switch that can allow for changes to be made to the time scale and that switch case 4 sets the timescale to 20
  4. Start the simulation in Unity
  5. Click the 1x Speed Button in simulation

6. Check dummy boolean value
7. Check integer value for Start Button for switch case is set to 4
8. Check timescale speed set value
9. Press the Start Button in simulation
10. Check dummy variable for value of 1 and that simulation begins to run
11. Press the Pause Button in simulation
12. Check dummy variable for value of 0 and that simulation enters a paused state
13. Record findings

- **Date Completed:** 12/04/2021

- **Test Title:** Single Step Button Implementation Test
- **Test ID:** TSTBT7
- **Requirement ID:** FR5a, FR5b, FR5f
- **Assignee:** Jack Bragg
- **Reviewee:** Nathan Moore
- **Test Steps:**
  1. Check to make sure that the scripts for the Start Button and Pause Button include a method that sets Time.timescale equal to 1 and set a dummy private variable equal to Time.timescale for debugging
  2. Create a dummy boolean value to set true when Single Step Button is pressed
  3. Check to make sure that script for the Start Button includes a check for isRunning and if this is true the Single Step function is disabled
  4. Check to make sure that the script for the Start Button includes a switch that can allow for changes to be made to the time scale and that switch case 1 sets the timescale to 1
  5. Start the simulation in Unity
  6. Click the Single Speed Button in simulation
  7. Check dummy boolean value
  8. Check integer value for Start Button for switch case is set to 1
  9. Check timescale speed set value
  10. A short script should run setting the timescale to 1, increment a counter to 1 at the end of the time step set isRunning to true, then set the timescale to 0 and set isRunning to be false
  11. Check to insure that both the Start and Pause Button can view the isRunning boolean value
  12. Record findings
- **Date Completed:** 12/04/2021

- **Test Title:** End Simulation Button Implementation Test
- **Test ID:** TSTBT8
- **Requirement ID:** FR5a, FR5b, FR5h
- **Assignee:** Jack Bragg
- **Reviewee:** Luke Tomlinson
- **Test Steps:**

1. Check to make sure that End Simulation Button can change the boolean value isRunning found in the start and pause buttons to 0
2. Check that if isRunning equals false that Time.timescale equals 0
3. Set a boolean value to report true when the End Simulation Button is pressed if not already existent for viewing privately
4. Check for function to print all Counter information to console beside name in “Counter Type: #\n” format if End Simulation Button is pressed
5. Check for function that clears all Counter information inside Stop Button script
6. Check that lists of all current in play ship game objects are passed to Stop Button script
7. Check that Start Button is passed to End Simulation Button
8. Check for function that allows for time scale switch is set to 1 in Start Button script
9. Press Start Button in Unity to begin Simulation
10. Press Start Button in Simulation
11. Let simulation run for 5 minutes to gather information available
12. Press End Simulation Button from Running State
13. Simulation should return to Default State in Paused State
14. Press Start Simulation Button
15. Let simulation run for 5 minutes
16. Press Pause Simulation Button
17. Press End Simulation Button
18. Simulation should return to Default State in Paused State
19. Record findings

- **Date Completed:** 12/03/2021

- **Test Title:** Cargo Ship Spawn Implementation Test
- **Test ID:** TSTSS1
- **Requirement ID:** FR6a
- **Assignee:** Aaron Mendez
- **Reviewee:** Luke Tomlinson
- **Test Steps:**
  1. Create a dummy integer counter inside the cargo ship spawn method inside shipScript tied to The Grid game object
  2. Check that Cargo Ship Prefab is passed to shipScript for spawning
  3. Check that the list of Cargo Spawn Points is passed to shipScript
  4. Press Start Button in Unity
  5. Press Start Simulation Button in simulation
  6. View the left-most column of the simulation to view if ships are spawning inside the grid spaces correctly – take screen shots if needed
  7. Run Simulation for 30 seconds minimum
  8. Press Pause Button in Unity
  9. Check to make sure that the size of the list of Cargo Ships in the End Simulation Button script is the same size as the dummy counter

10. Check to make sure that the size of the list of Cargo Ships in the Cargos Entered Counter is the same size as the dummy counter
11. Record findings

- **Date Completed:** 12/03/2021

- **Test Title:** Patrol Ship Spawn Implementation Test
- **Test ID:** TSTSS2
- **Requirement ID:** FR6d
- **Assignee:** Aaron Mendez
- **Reviewee:** Luke Tomlinson
- **Test Steps:**
  1. Create a dummy integer counter inside the patrol ship spawn method inside shipScript tied to The Grid game object
  2. Check that Patrol Ship Prefab is passed to shipScript for spawning
  3. Check that the list of Patrol Spawn Points is passed to shipScript
  4. Press Start Button in Unity
  5. Press Start Simulation Button in simulation
  6. View the right-most column of the simulation to view if ships are spawning inside the grid spaces correctly – take screen shots if needed
  7. Run Simulation for 30 seconds minimum
  8. Press Pause Button in Unity
  9. Check to make sure that the size of the list of Patrol Ships in the End Simulation Button script is the same size as the dummy counter
  10. Check to make sure that the size of the list of Patrol Ships in the Patrols Entered Counter is the same size as the dummy counter
  11. Record findings
- **Date Completed:** 12/03/2021

- **Test Title:** Pirate Ship Spawn Implementation Test
- **Test ID:** TSTSS3
- **Requirement ID:** FR6c
- **Assignee:** Aaron Mendez
- **Reviewee:** Luke Tomlinson
- **Test Steps:**
  1. Create a dummy integer counter inside the pirate ship spawn method inside shipScript tied to The Grid game object
  2. Check that Pirate Ship Prefab is passed to shipScript for spawning
  3. Check that the list of Pirate Spawn Points is passed to shipScript
  4. Press Start Button in Unity
  5. Press Start Simulation Button in simulation
  6. View the bottom-most row of the simulation to view if ships are spawning inside the grid spaces correctly – take screen shots if needed

7. Run Simulation for 30 seconds minimum
8. Press Pause Button in Unity
9. Check to make sure that the size of the list of Pirate Ships in the End Simulation Button script is the same size as the dummy counter
10. Check to make sure that the size of the list of Pirate Ships in the Pirate Entered Counter is the same size as the dummy counter
11. Record findings

- **Date Completed:** 12/03/2021

- **Test Title:** Captured Ship Spawn Implementation Test
- **Test ID:** TSTSS4
- **Requirement ID:** FR6b
- **Assignee:** Aaron Mendez
- **Reviewee:** Luke Tomlinson
- **Test Steps:**
  1. Create a dummy integer counter inside the captured ship spawn method inside shipScript tied to The Grid game object
  2. Check that Captured Ship Prefab is passed to shipScript for spawning
  3. Drop 3-5 Cargo Ship Prefabs into grid spaces in simulation
  4. Drop 3-5 Pirate Ship Prefabs into grid spaces in simulation
  5. These ships must be positioned where they will interact
  6. Press Start Button in Unity
  7. Press Start Simulation Button in simulation
  8. View the area where the prefabs were dropped onto the map
  9. Run Simulation for 30 seconds minimum
  10. Press Pause Button in Unity
  11. Record the Evades Evades Captured Counter
  12. Record the Evades Captured
  13. Record the Cargos Captured Counter
  14. Check to make sure that the size of the list of Patrol ships in the End Simulation Button script is the same size as the dummy counter
  15. Check to make sure that the size of the list of Patrol ships in the Patrols Entered Counter is the same size as the dummy counter
  16. Record findings
- **Date Completed:** 12/03/2021

- **Test Title:** Cargo Ship Captured Pirates List Test
- **Test ID:** TSTFR9
- **Requirement ID:** FRR15
- **Assignee:** Nathan Moore
- **Reviewee:** Jack Bragg
- **Test Steps:**

    - Check Cargo Ship script for a numbered list that can hold Pirate Ship game object names
    - Check shipScript to ensure that each Pirate Ship is instantiated with a unique tag in name
    - Drag a Cargo Ship into the simulation space
    - Drag a Pirate Ship into the simulation space
    - Press Play in Unity
    - Press Start in simulation
    - Force interaction between these two prefabs until Cargo Ship isCaptured boolean reports true and the Cargo Ship transforms into a Captured Ship
    - Pause Simulation through Unity
    - Move the Pirate Ship that Captured the Cargo Ship to a new area simulation where the Cargo Ship will interact with it once more
    - Change Variables for the Captured type Cargo Ship back to Cargo Variables for movement
    - Resume the Simulation through Unity
    - Force interaction between these ships
    - Report Findings
- **Date Completed:** 11/27/2021

- **Test Title:** Patrol Ship Attack Pirate Test
- **Test ID:** TSTFR9
- **Requirement ID:** FRR17
- **Assignee:** Nathan Moore
- **Reviewee:** Jack Bragg
- **Test Steps:**
    - Check Cargo Ship script for a numbered list that can hold Pirate Ship game object names
    - Check shipScript to ensure that each Pirate Ship is instantiated with a unique tag in name
    - Drag a Patrol Ship into the simulation space
    - Drag a Pirate Ship into the simulation space
    - Press Play in Unity
    - Press Start in simulation
    - Force interaction between these two prefabs until Patrol Ship PirateInRange boolean reports true
    - Pause Simulation through Unity after one time step
    - Check that the Pirate Ship no longer exists in the simulation and all counters report correctly
    - Report Findings
- **Date Completed:** 11/30/2021

- **Test Title:** Patrol Ship Rescue Captured Cargo Test
- **Test ID:** TSTFR11
- **Requirement ID:** FRR18
- **Assignee:** Nathan Moore
- **Reviewee:** Aaron Mendez/Jack Bragg
- **Test Steps:**

    - Drag a Patrol Ship into the simulation space
    - Drag a Captured Ship into the simulation space
    - Press Play in Unity
    - Press Start in simulation
    - Force interaction between these two prefabs until Patrol Ship CapturedInRange boolean reports true
    - Pause Simulation through Unity after one time step
    - Check that the Pirate Ship that was attached to the Captured Ship no longer exists in the simulation and all counters report correctly
    - Check that the Captured Ship has returned to its default Cargo Ship state
    - Report Findings
- **Date Completed:** 12/02/2021

- **Test Title:** Pirate Ship Capture Cargo Test
- **Test ID:** TSTFR12
- **Requirement ID:** FRR21
- **Assignee:** Nathan Moore
- **Reviewee:** Luke Tomlinson
- **Test Steps:**
    - Check Cargo Ship script for a numbered list that can hold Pirate Ship game object names
    - Check shipScript to ensure that each Pirate Ship is instantiated with a unique tag in name
    - Drag a Cargo Ship into the simulation space
    - Drag a Pirate Ship into the simulation space
    - Press Play in Unity
    - Press Start in simulation
    - Force interaction between these two prefabs until Pirate Ship CargoInRange boolean reports true
    - Pause Simulation through Unity after one time step
    - Check values for Cargo Ship have been altered to Captured Ship and are now pointing to move downward
    - Check values for Pirate Ship have been altered and movement is now pointing to move downward
    - Report Findings
- **Date Completed:** 12/03/2021

Special Considerations

The Somalian Pirate Simulator has very few environmental, front-end, or back-end requirements that require special consideration. However one important special consideration to consider is that the

simulation must be tested on both Macintosh and Windows 10/11 PCs in a DirectX11 environment. This application must be optimized to run on both of these environments at gold release time.

Team E
Nathan Moore Luke Tomlinson
Aaron Mendez Jack Bragg

Detailed Designs

The Detailed Designs section contains screenshots exemplifying how all Use Cases interactions listed in the Software Requirements Specifications section, beginning on page 12, are reported to the user during runtime. These photos are introduced to the reader with a header citing which Use Case the reader should reference for a full description of the interactions needed to achieve the following runtime frame. The screenshots are then followed by the figure number associated with them and a brief description of what stage within the cited Use Case the current screenshot represents.

General User Interface and Workflow Description

The Somalian Pirate Simulator has a very straightforward approach for the general user interface(GUI). The GUI consists of one single screen which consists of three main sections: The featureless gridded map, the button-control panel, and the simulation information panel. The button-control panel and simulation information panel are housed together in the top 1/5 of the user's monitor. Somalian Pirate Simulator has been optimized of a 1920x1080p HD resolution but can handle a multitude of display ratios. The user is able to interact with any of the 8 buttons found in the control panel in the top left corner of their screen as well as the 1–9 number keys at the top of any standard 60%+ keyboard in order to alter the spawn change of any of the three initial ship types. The user may also change the view volume of the simulation from up-close (a grid size about 3x5) to far away (a grid size about 15x20) using the mouse scroll wheel. The user may also pan to new locations within the grid by holding down the left mouse button and "dragging and dropping" to a new location on the map. The Somalian Pirate Simulator begins in a paused empty state and the user must click the "Start" button to begin the simulation.

There are many variations in the workflow for The Somalian Pirate Simulator. The user has the option to change the spawn rates of any initial ship type (Cargo, Pirate, Patrol) before the simulation begins, during the simulations running state, and during a paused state. The user may also alter the speed at which the simulation runs between 1X, 2X, 10X, and 20X speed, in both in the running and the paused states.

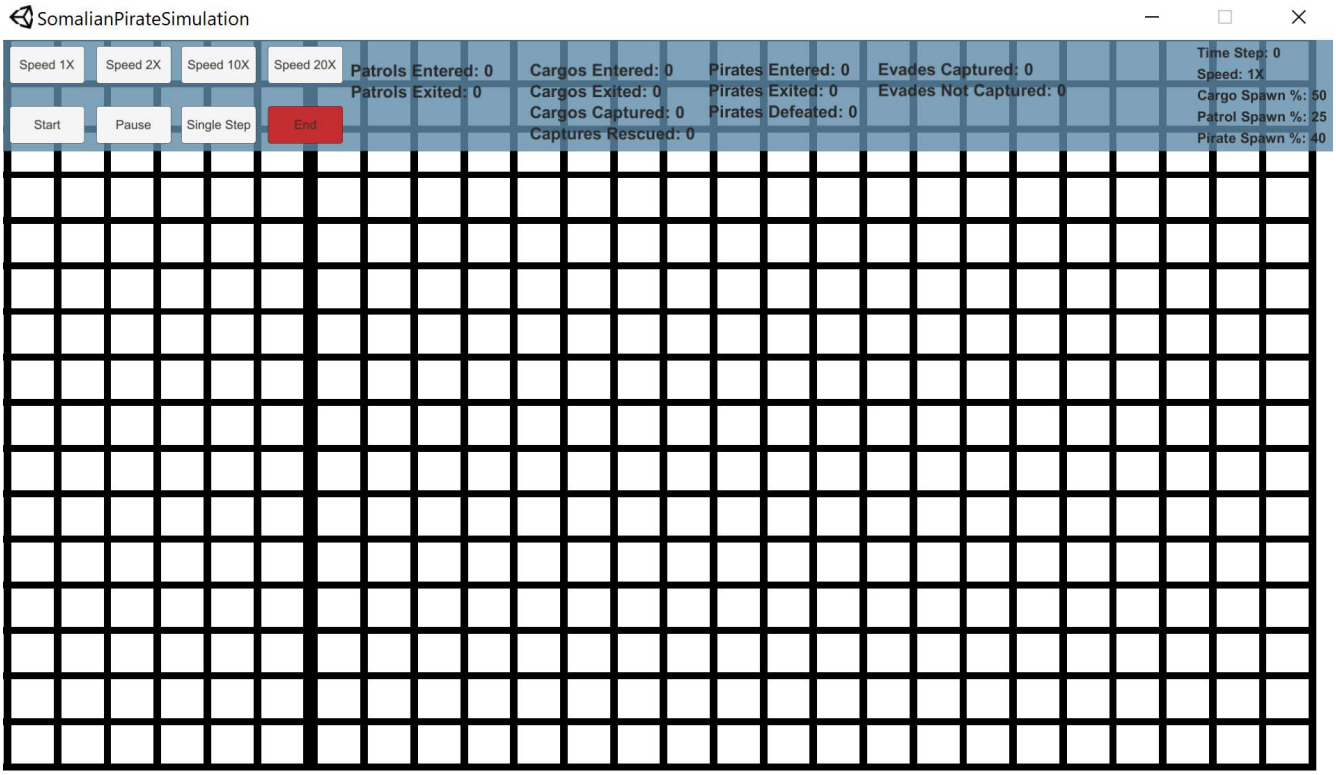
There are also strict limitations on two interactions that the user can choose: choosing the single step button must occur during a paused state or no interaction will occur and no update other than the current data update will be displayed to the user. The reset/end simulation button can be selected during the running or the paused state, but selection of this button will immediately end the simulation and print a .csv file containing all final information displayed in the information panel. Due to this, the reset/end simulation button should only be chosen when the user desires to truly end the current simulation run.

The following section includes screenshots of all of the actions a user can enact upon the Somalian Pirate Simulation system. A completed walk-through of each interaction the user is able to accomplish within Somalian Pirate Simulator can be found beginning on page 12 and ending on page 17 of the Software Requirements Specifications portion of this document.

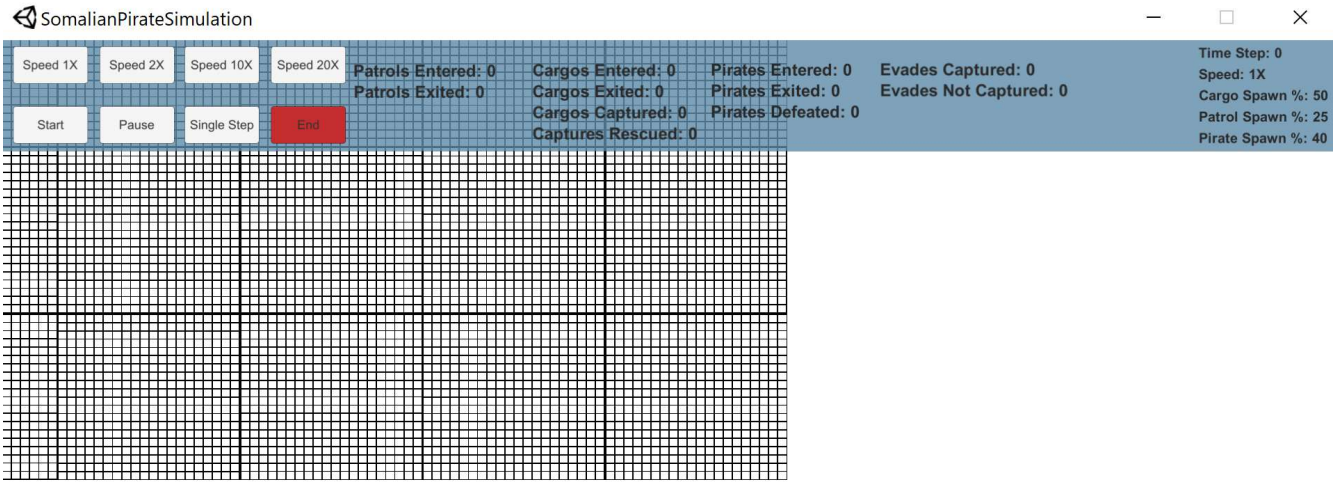
Team E
Nathan Moore
Aaron Mendez
Luke Tomlinson
Jack Bragg

Screenshots Of Use Case Activities

Use Case 1: Open Simulation—Default State and Use Case 2: Start Simulation, Variation 1: Default State Settings

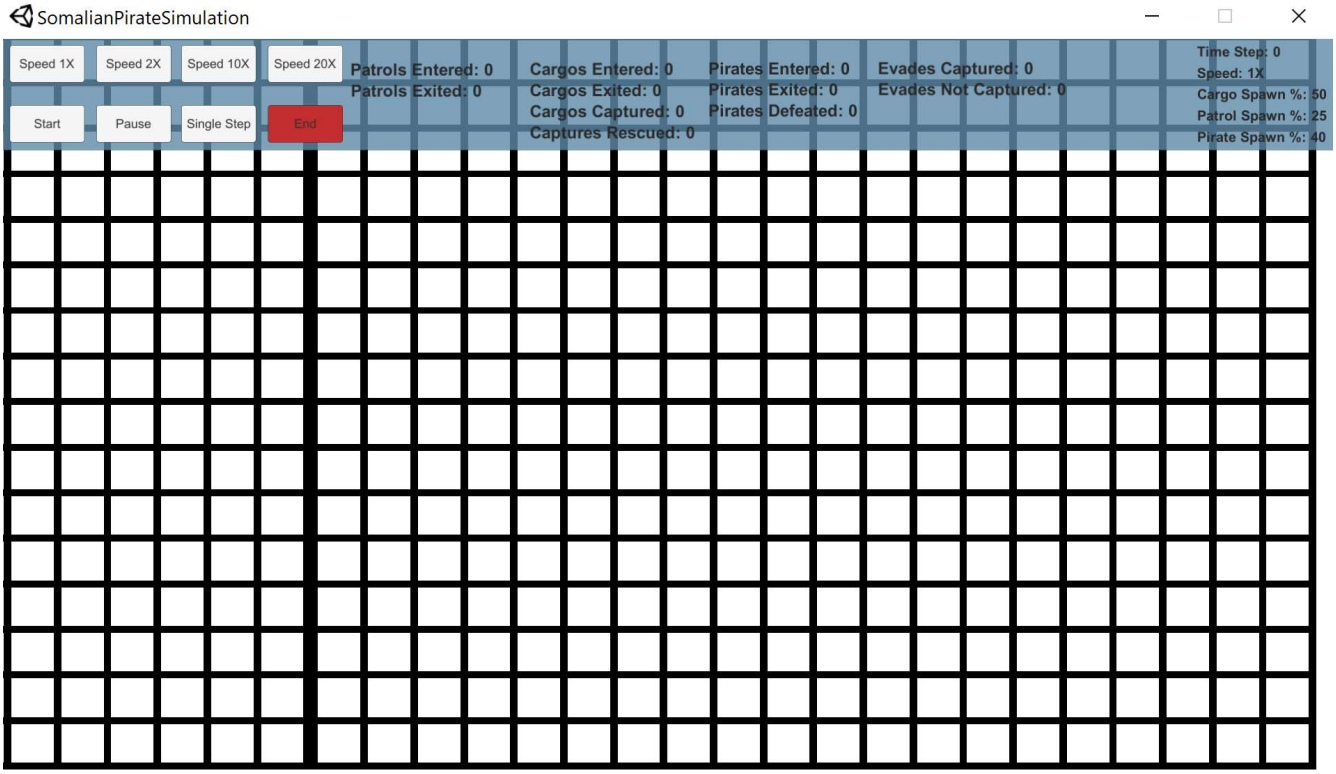


Default state – default zoom

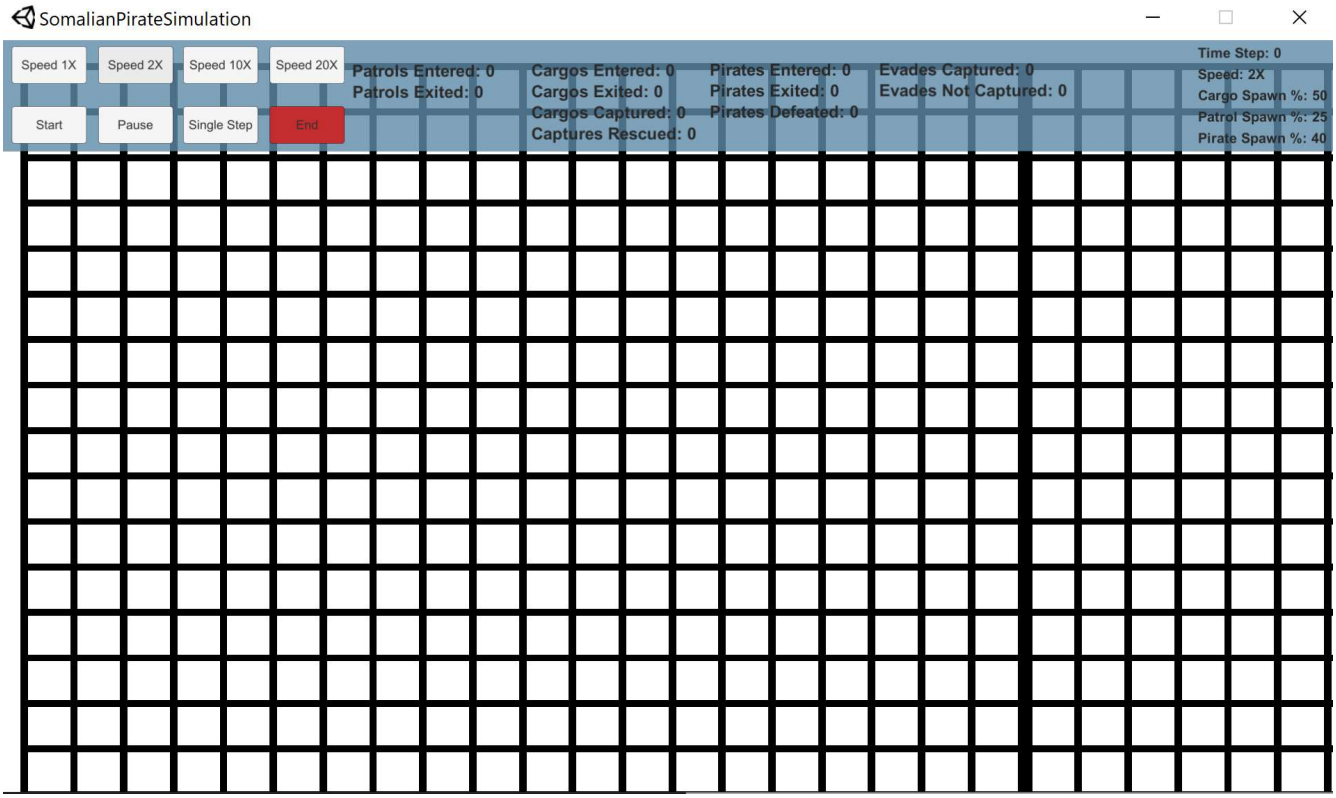


Default State – max zoom out

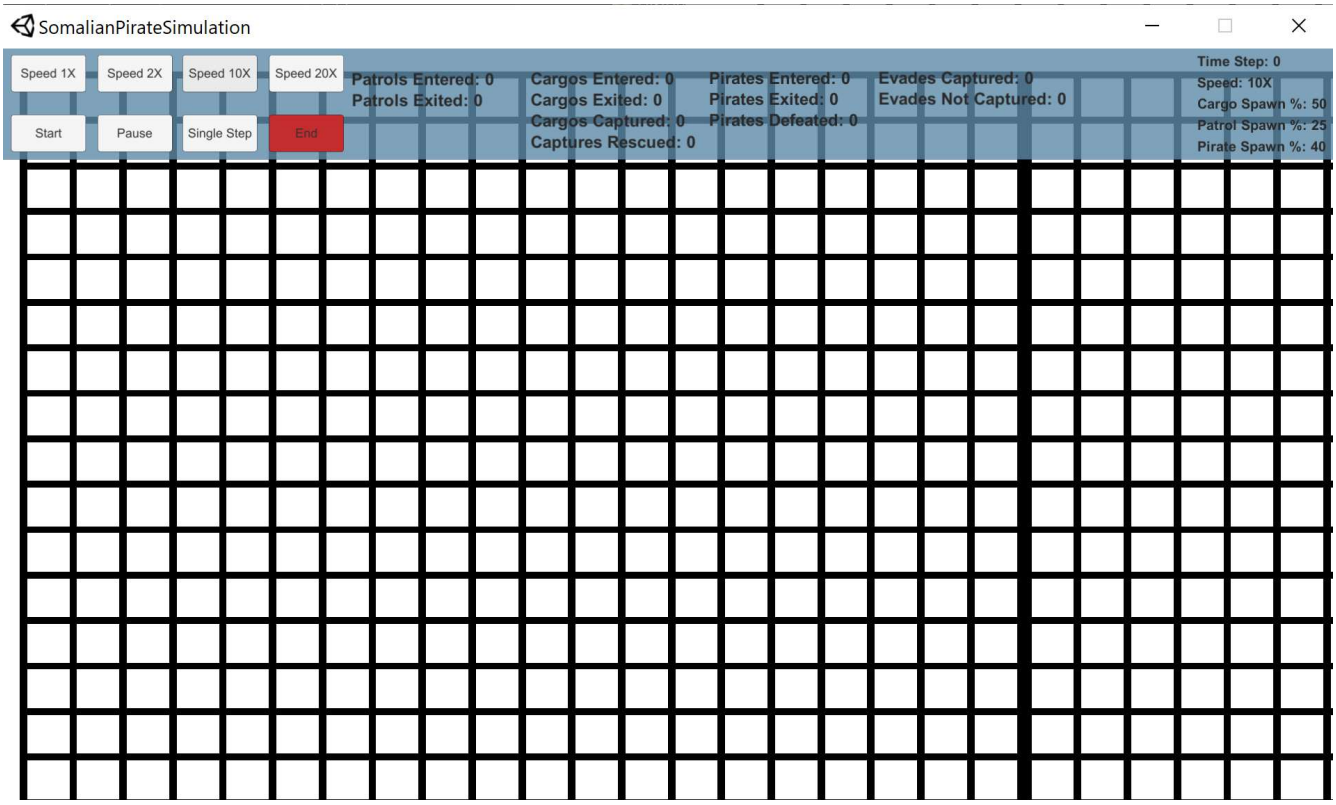
Use Case 2: Start Simulation, Variation 2: Paused State—User Speed Selected
The following frames are shown after the user selects any other speed than the default speed(1X) when the Simulation is in a paused state



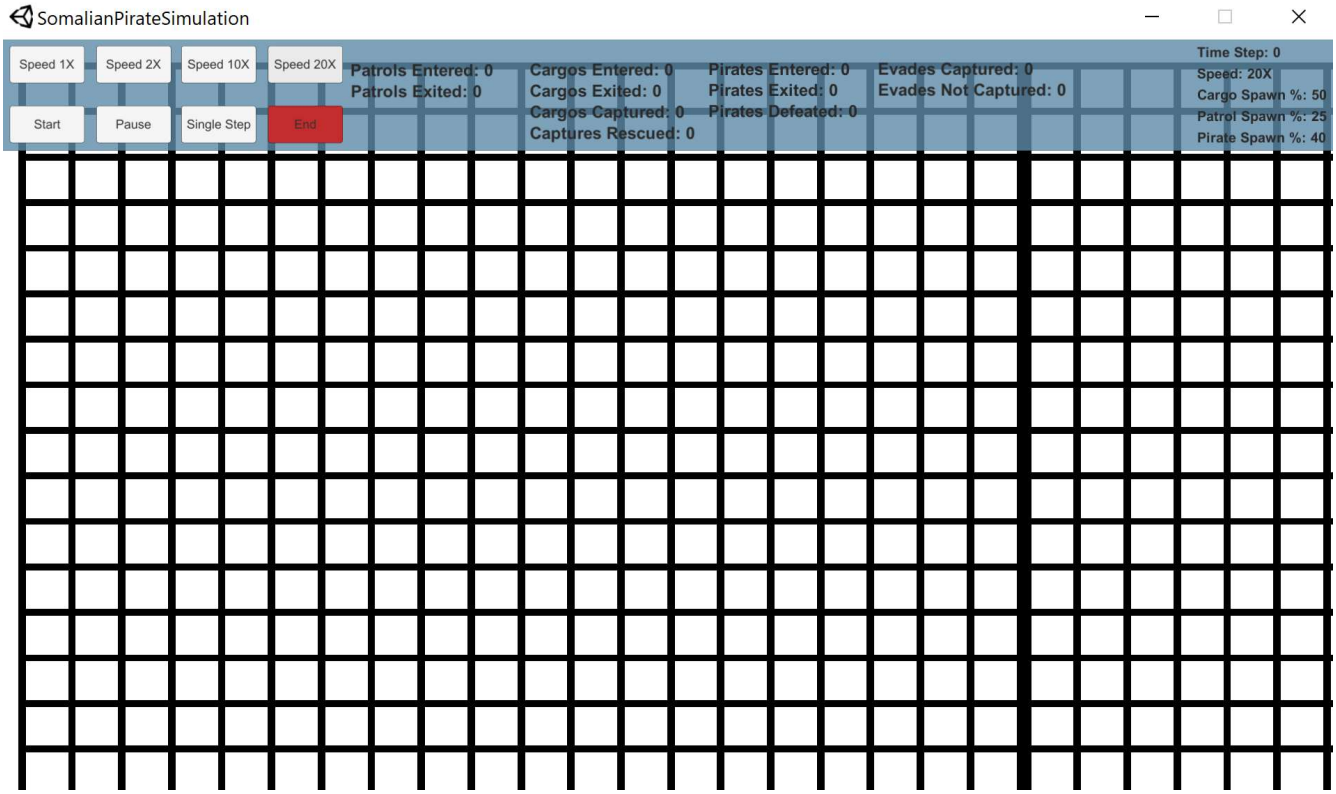
Paused State – Speed 1X



Paused State – Speed 2X



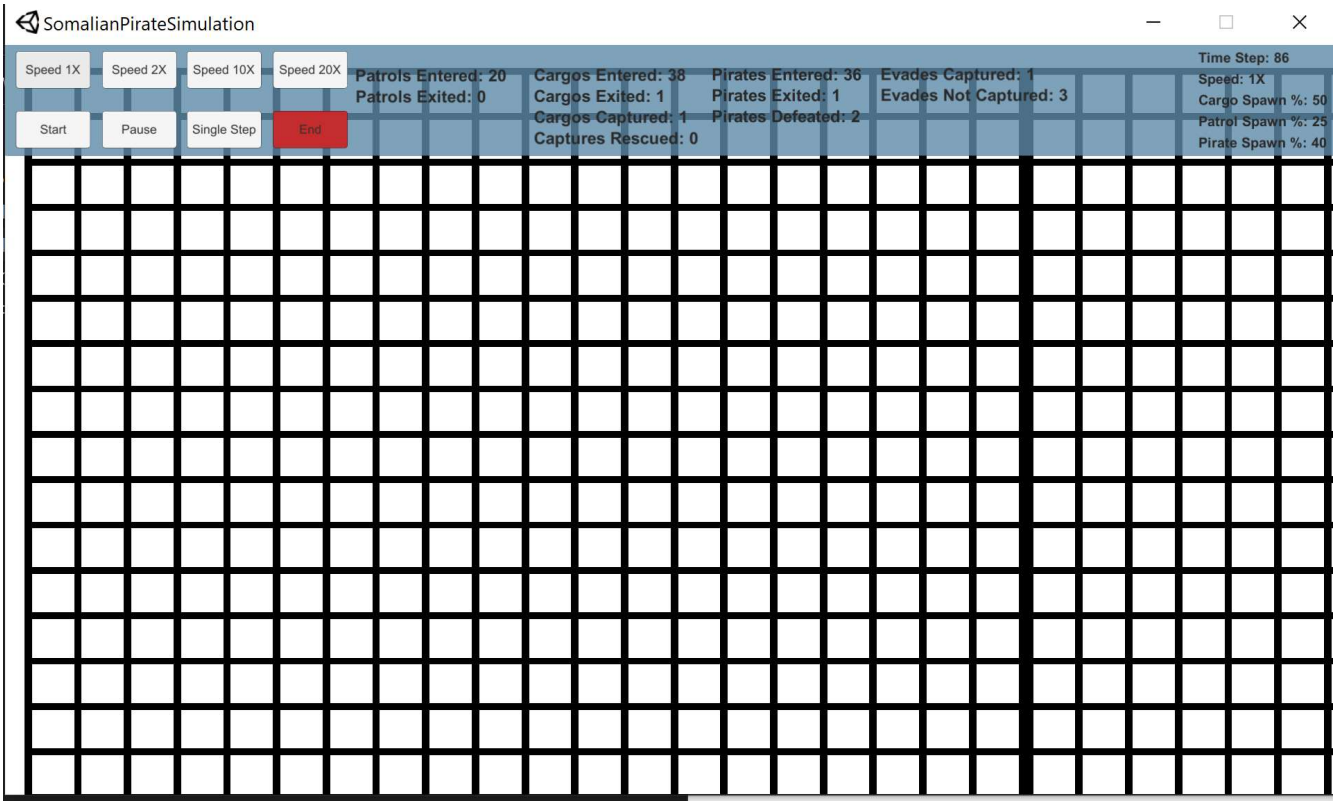
Paused State – Speed 10X



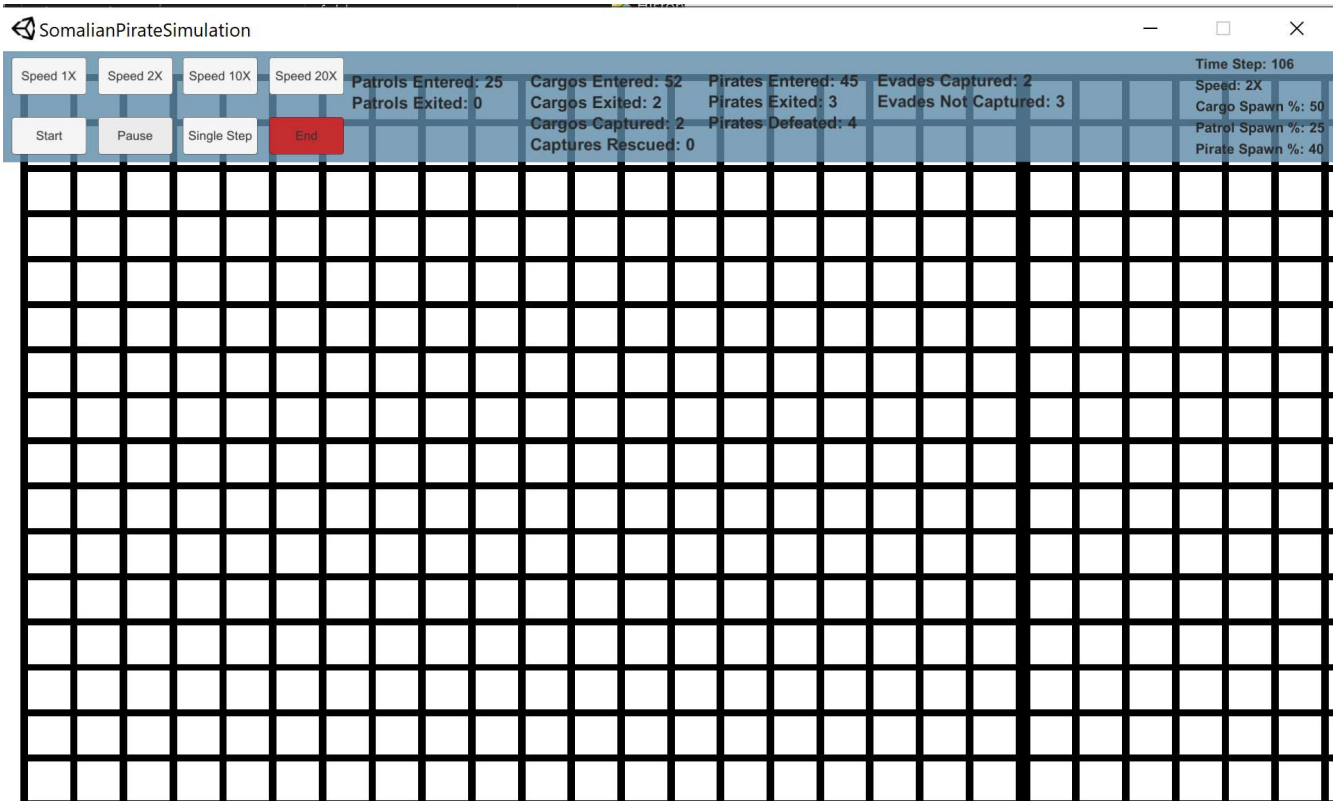
Paused State – 20X

Use Case 3: Pause Simulation From Running State

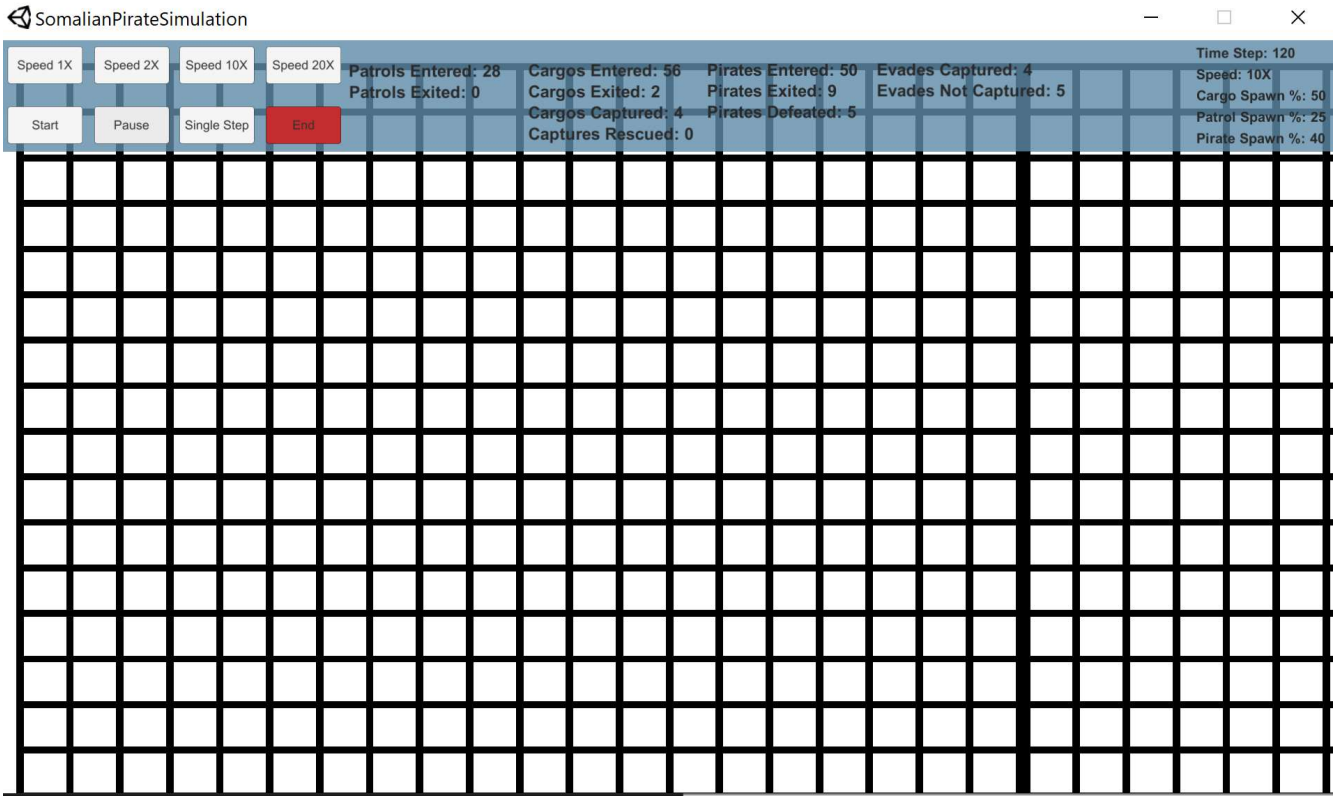
The following frames show that when in a paused state, there should be no updates to any counter. Time is set to 0 when “Pause” button is selected



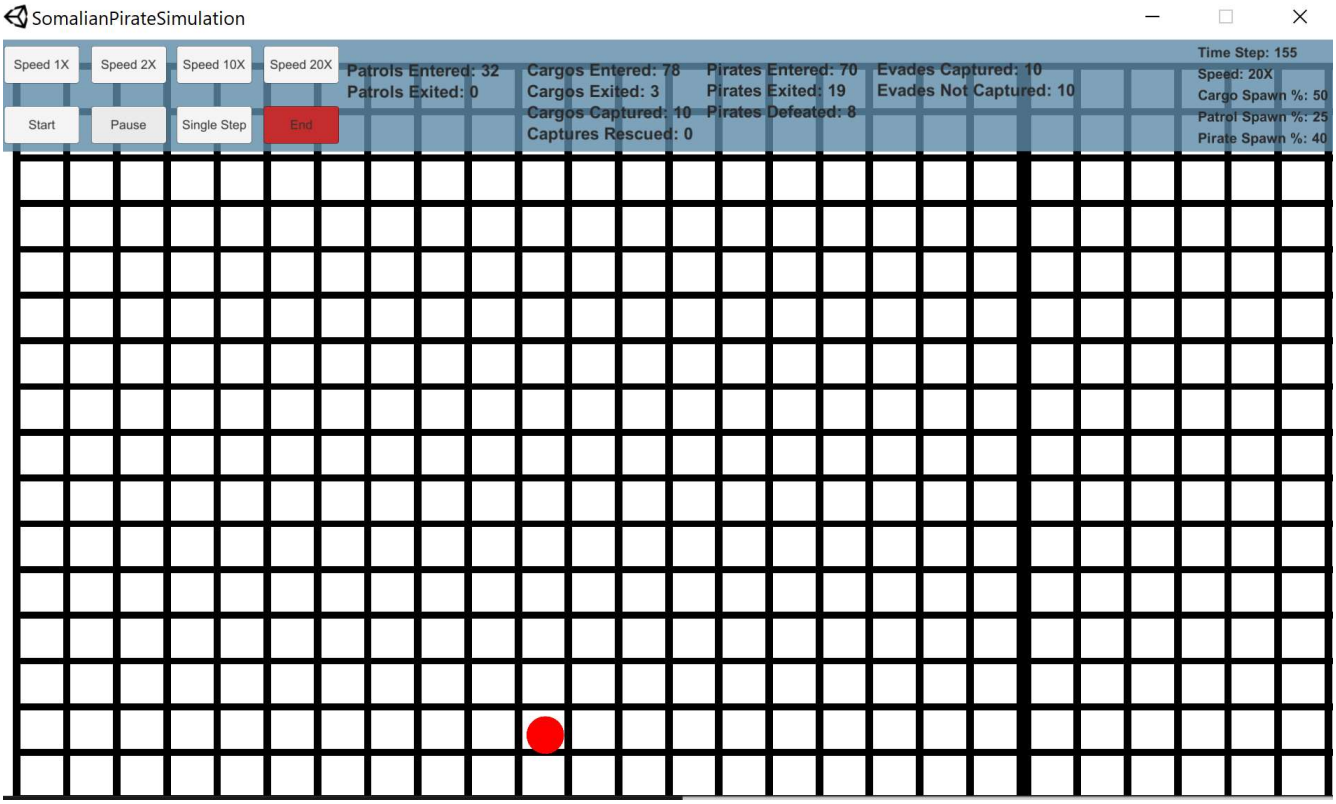
Running State – Speed 1X



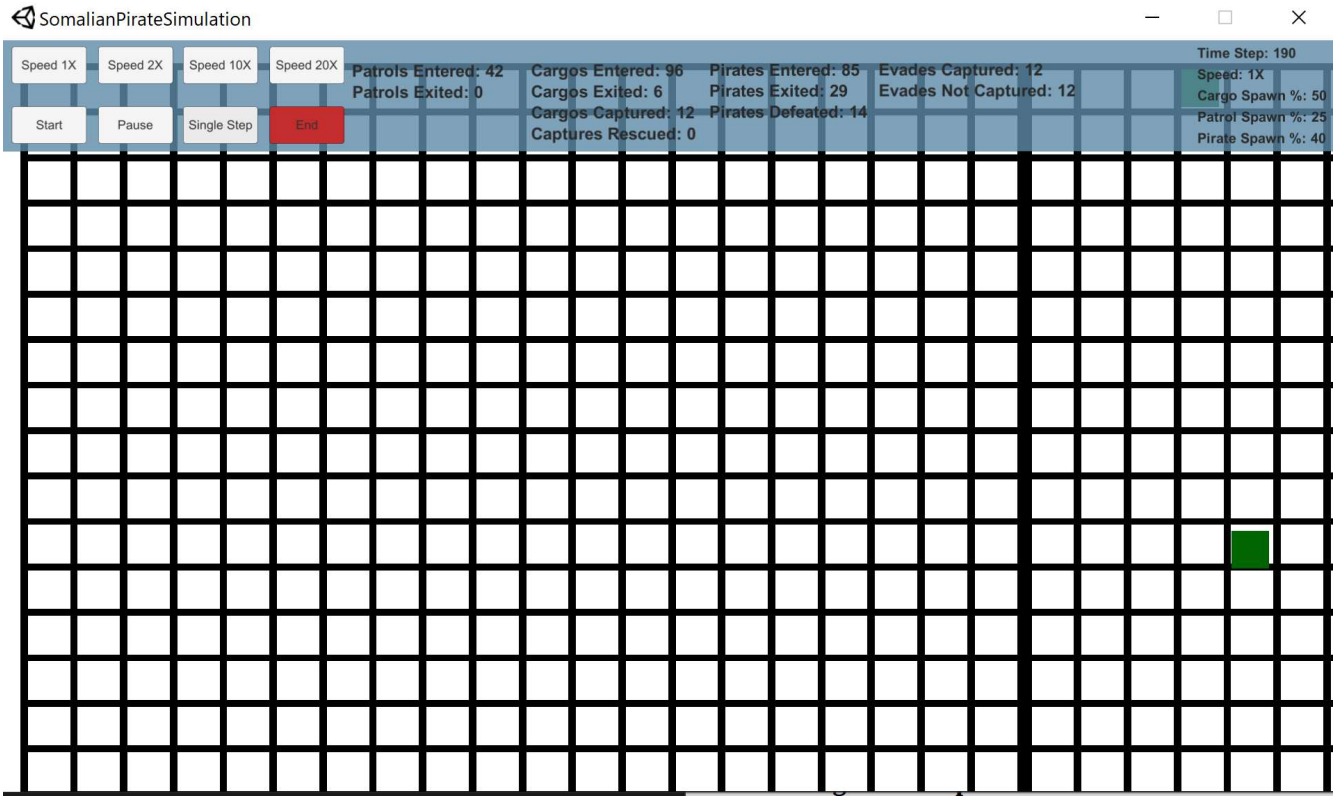
Running State – Speed 2X



Running State – 10X



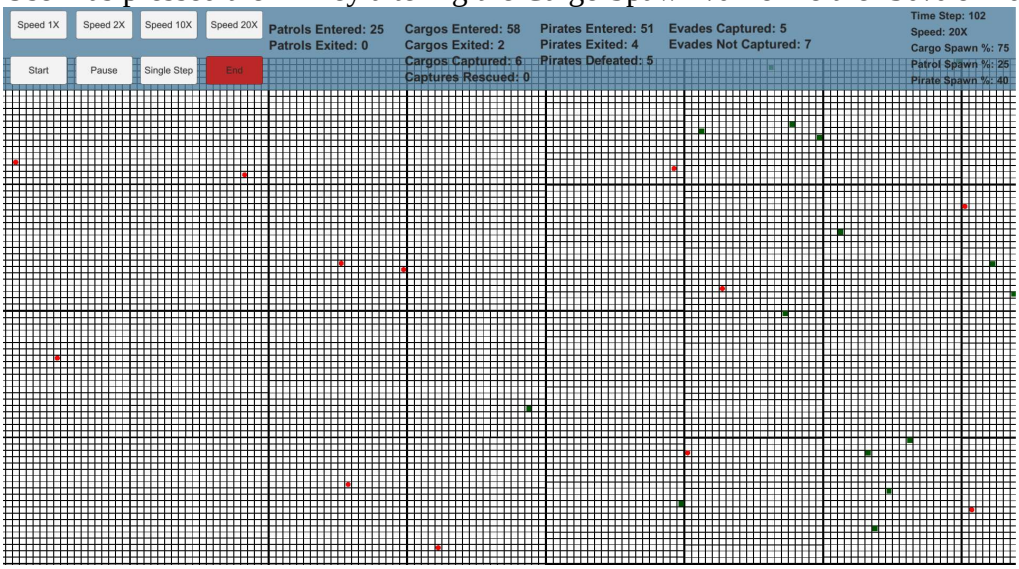
Running State – Speed 20X



Single Step – Before Selection in Paused State

Use Case 12, Variation 1: User Changes Spawn Rate Of Cargo Ships To 75%(Currently 50% Or 100%); Running State

User has pressed the ‘2’ key altering the Cargo Spawn % from either 50% or 100% to 75%



Use Case 12, Variation 2: User Changes Spawn Rate Of Cargo Ships To Default—50% (Currently 75% Or 100%); Running State

User has pressed the ‘1’ key altering the Cargo Spawn % from either 75% or 100% to 50%

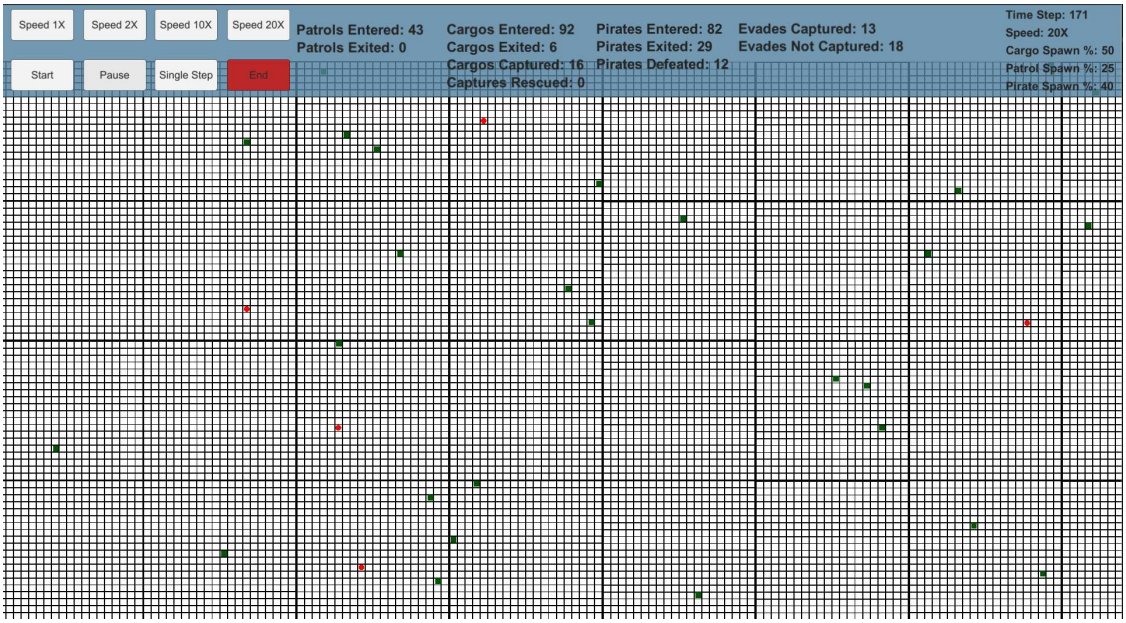
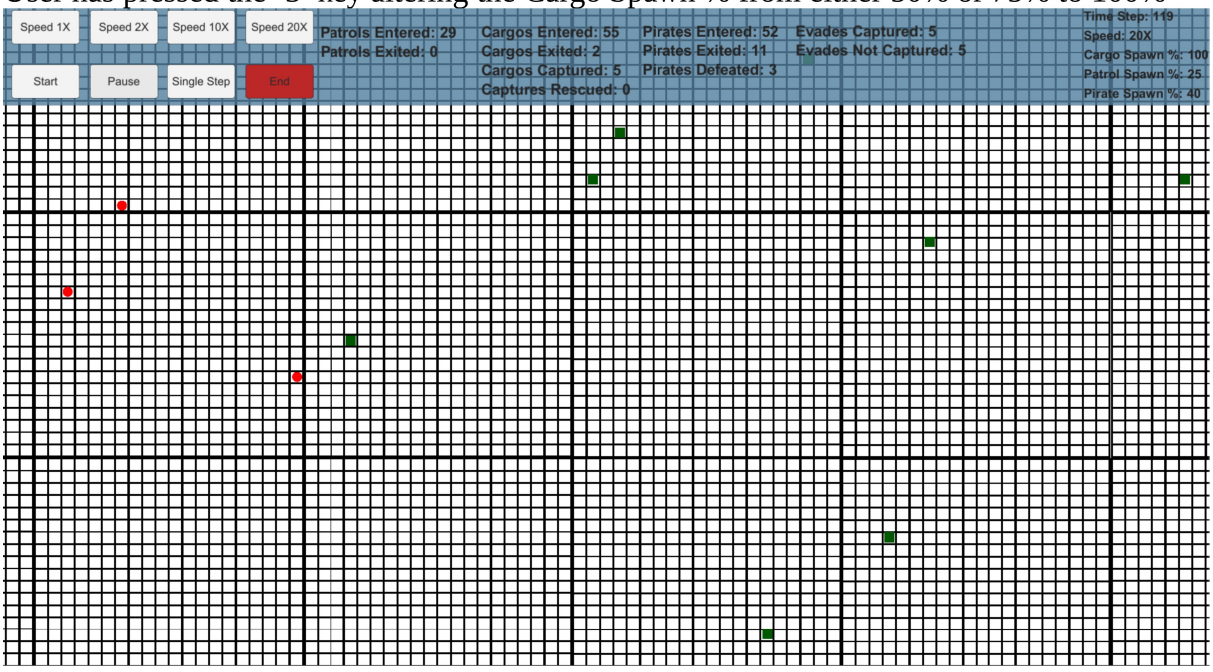


Figure 26 – Cargo Spawn Rate Set To 50%, Running State

Use Case 12, Variation 3: User Changes Spawn Rate Of Cargo Ships To 100% (Currently 50% Or 75%); Running State

User has pressed the ‘3’ key altering the Cargo Spawn % from either 50% or 75% to 100%



Use Case 13, Variation 1: User Changes Spawn Rate Of Patrol Ships To 50%(Currently 25% Or 100%); Running State

User has pressed the ‘5’ key altering the Patrol Spawn % from either 25% or 100% to 50%

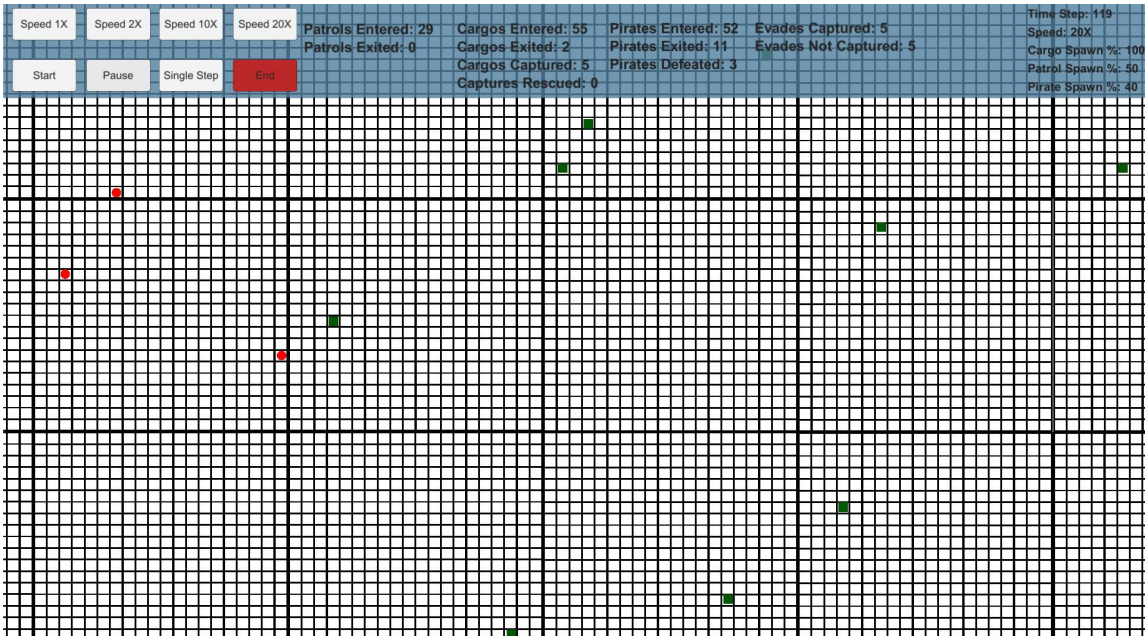


Figure 28 – Patrol Spawn Rate Set To 50%, Running State

Use Case 13, Variation 2: User Changes Spawn Rate Of Patrol Ships To Default—25% (Currently 50% Or 100%); Running State

User has pressed the ‘4’ key altering the Patrol Spawn % from either 50% or 100% to 25%

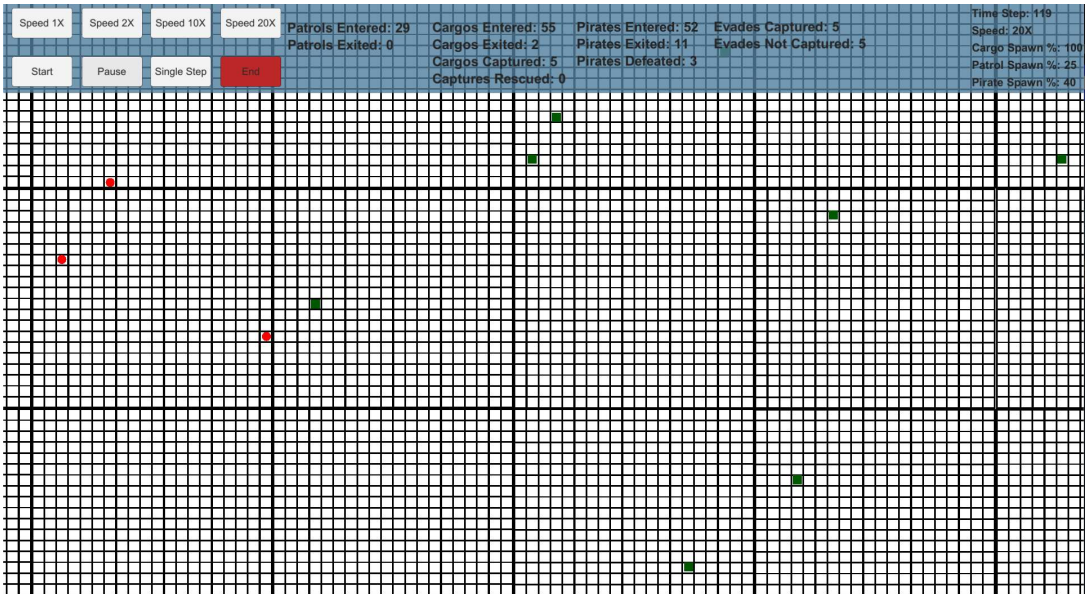


Figure 29 – Patrol Spawn Rate Set To 25%, Running State

Use Case 13, Variation 3: User Changes Spawn Rate Of Patrol Ships To 100% (Currently 25% Or 50%); Running State

User has pressed the ‘6’ key altering the Patrol Spawn % from either 25% or 50% to 100%

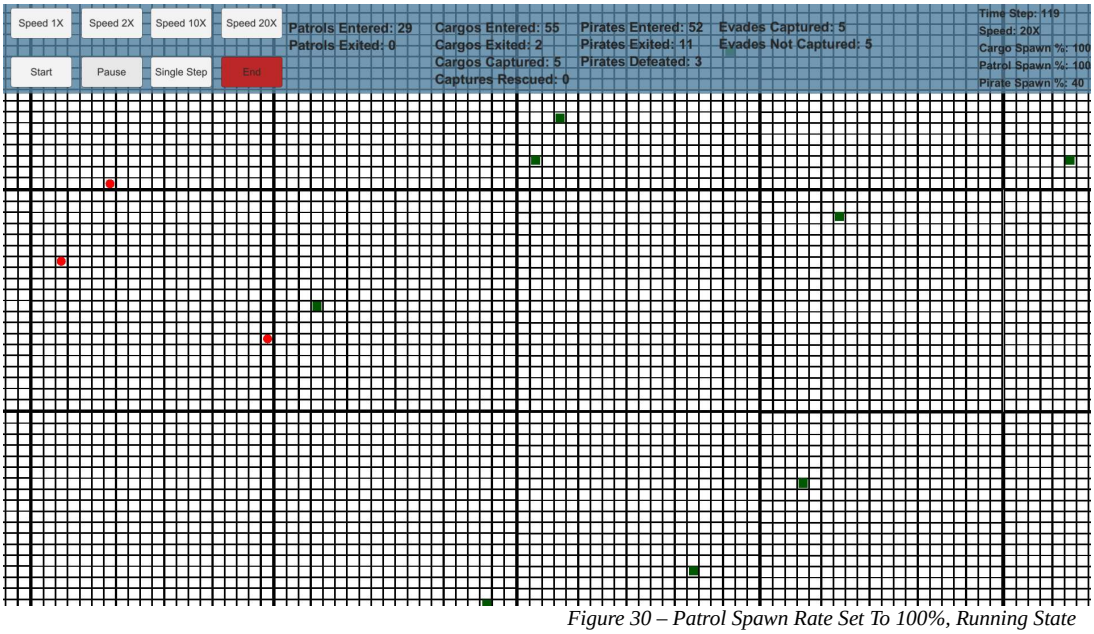
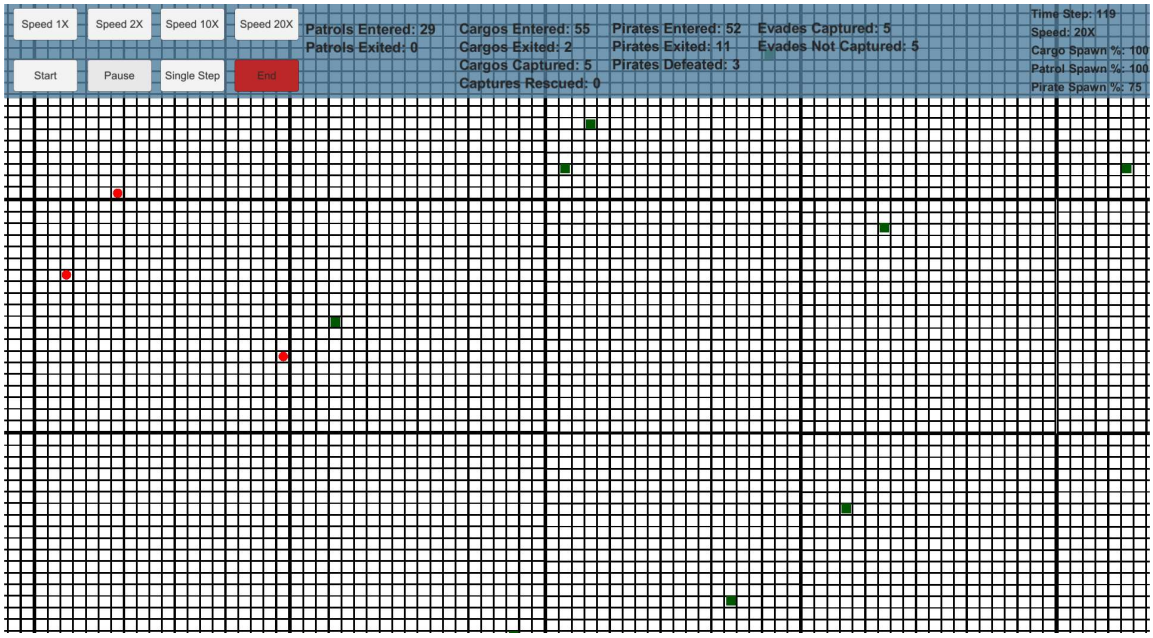


Figure 30 – Patrol Spawn Rate Set To 100%, Running State

Use Case 14, Variation 1: User Changes Spawn Rate Of Pirate Ships To 75%(Currently 40% Or 100%); Running State

User has pressed the ‘8’ key altering the Pirate Spawn % from either 40% or 100% to 75%



Use Case 14, Variation 2: User Changes Spawn Rate Of Pirate Ships To Default—40% (Currently 75% Or 100%); Running State

User has pressed the ‘7’ key altering the Pirate Spawn % from either 75% or 100% to 40%

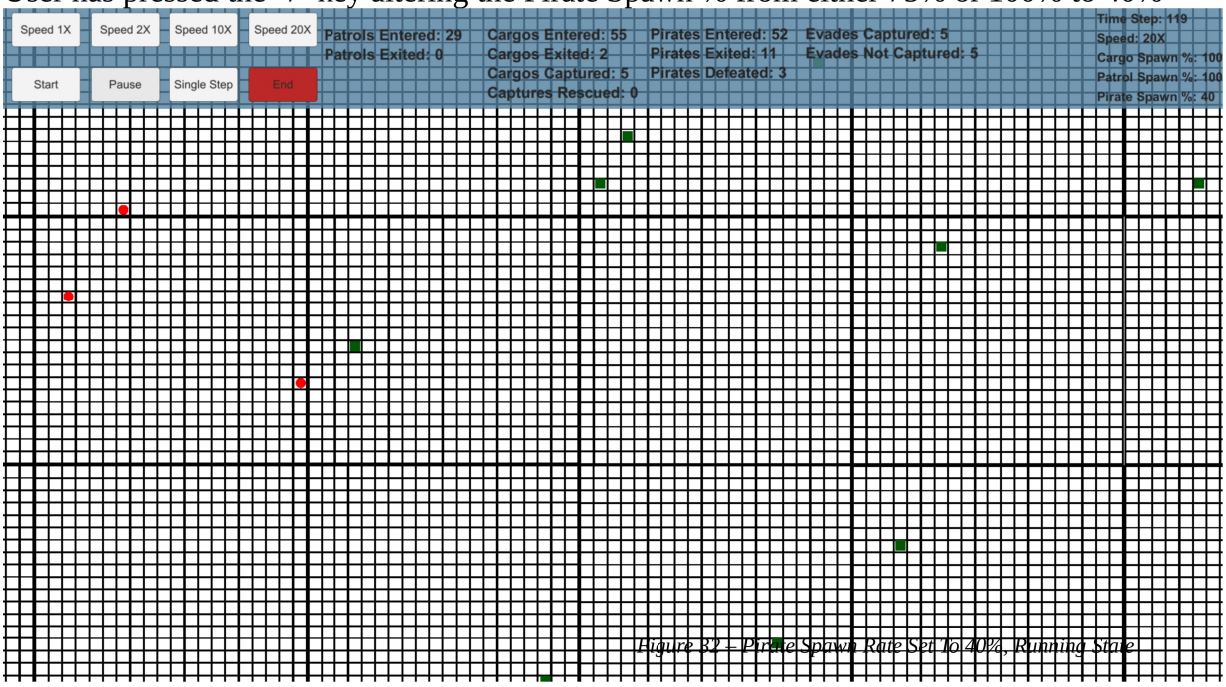
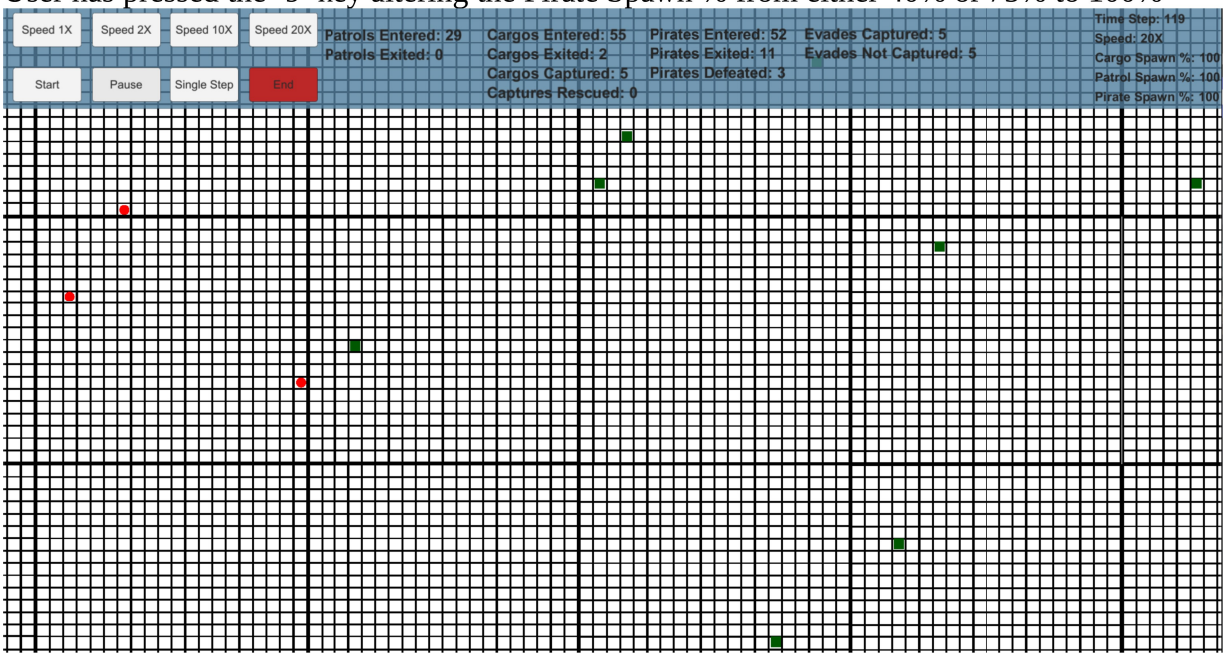


Figure 82 – Pirate Spawn Rate Set To 40%, Running State

Use Case 14, Variation 3: User Changes Spawn Rate Of Pirate Ships To 100% (Currently 40% Or 75%); Running State

User has pressed the ‘9’ key altering the Pirate Spawn % from either 40% or 75% to 100%



The Detailed Design section provides a brief description of how the user can interact with the Somalian Pirate Simulation general user interface and the bounds in which the user can achieve these actions, as well as a list of screenshots associated with the these Use Cases. A more detailed analysis of the steps the user must take to reach all of these screens can be found in the Software Requirements Specifications section on pages 12 through 17.

Conclusion

Team E learned many things throughout the development of The Somalian Pirate Simulation. All members of Team E learned new skills whether they pertained to Unity, C#, Jira, the Agile development, project management, or general teamwork communication. Team E was able to achieve all tasks and abilities for The Somalian Pirate Simulation that were described in the Approved Combined Statement of Work found in Appendix A.3. There are currently two small bugs that were found on December 06, 2021 after development was halted on December 05, 2021.

These bugs include:

When the user zooms in all of the way if the camera is allowed to push back against the grid it will begin to zoom out on the opposing corner of the map

When the single step button is clicked the simulation does not default to 1X speed every time and sometimes will step in the currently user selected speed

These bugs should be able to be quickly resolved in future development and will allow for all statement of work abilities to be implemented properly and completely.

Team E was able to deploy software for Mac OS X running on both Intel x86 and Apple Silicon as well as Windows 10/11 for both 32 and 64 but architectures. All of these programs have been tested on a variety of monitor sizes and architectures. After testing completed Team E opted to deploy Somalian Pirate Simulation in: a forced windowed, 1080P resolution, 16×9 aspect ratio; which cannot be altered in order to maintain continuity over all platforms.

Future development for The Somalian Pirate Simulation will include:

1. Current identified bug fixes
2. Testing to find more bugs in development
3. Adding a print option for data collected during runtime
4. Adding an “Are You Sure?” dialogue/selection box upon selecting the “End” button
5. Creation of an interactive user guide scene to introduce first time users to simulation layout
6. Adding animations to ship interactions
7. Adding sounds to simulation

References

1. *IMB Piracy Reporting Centre*. ICC Commercial Crime Services. (2017, July 7). https://www.icc-ccs.org/piracy-reporting-centre.

Glossary

- Counter – Reflects the number of times an event occurred
- Enter – When a ship spawns
- Evade Space – The spaces in this range, relative to a cargo, where row is the row on the Map the cargo is and col is the column on the Map the cargo is:
- Map(row – 1 to row + 2, col + 2) OR Map(row + 2, col – 1 to col + 2)
- Exit – When a ship leaves the defined map area through any type of movement
- Grid – A single square within the Map. A grid can be referenced by the number of rows and columns: gridName(rowNumber, columnNumber).
- Map – The 100(row) x 400 (column) featureless grid area that is displayed in the background during the simulation specifying each ships exact location.
- Ship – Four types of ships exist within the Somalian Pirate Simulation:
  a. Capture – A Cargo Ship that has been captured by a Pirate Ship
  b. Cargo – A Merchant Vessel that cannot resist Pirate Attacks
  c. Patrol – Naval Vessel that can defeat Pirate Ships and Rescue Captured Cargo Ships
  d. Pirate – An Armed Vessel that can capture a Cargo Ship but cannot resist an attack from a Naval Patrol Vessel
- Simulation Speed – Measured in Time-Steps-per-second; this speed may be adjusted by the user to: 1X, 2X, 10X, or 20X.
- Simulation State – The Simulation may exist in three states:
  a. Running – The simulation advances at the user defined simulation speed.
  b. Pauses – The simulation speed is 0 – no updates will occur.
  c. Default – The simulation speed is placed in the Paused State and each counter is set to 0.
- Time Step: A user defined block of time – One Time Step is equivalent to five minutes in real time.
- UI – Acronym for User Interface or Front End
- UX – Acronym for Back End

Appendix

Appendix A – Statements of Work

A.1: Dr. Petty—CS 499, Project Description, Somalian Pirates Simulation

CS 499 Senior Project
Project Assignment
Somali Pirates Simulation

Rationale

The International Maritime Bureau reported that over the period 2005-2012 nearly 1,000 ships were fired at, chased, boarded, or hijacked by pirates. The pirates, often operating from bases in Somalia, usually attacked ships in the northwest portion of the Indian Ocean, especially in the Gulf of Aden. The attacks were carried out nearly daily during some periods of the year, and often occurred despite the presence of naval patrol vessels in the area. Vessels hijacked by pirates were usually ransomed by their owners and insurers, sometimes for as much as tens of millions of dollars, which is usually a small percentage of the value of the vessel and its cargo.

As a first step towards reducing or eliminating future pirate attacks, you have been asked to develop a highly simplified simulation of merchant shipping, naval patrols, and pirate attacks.

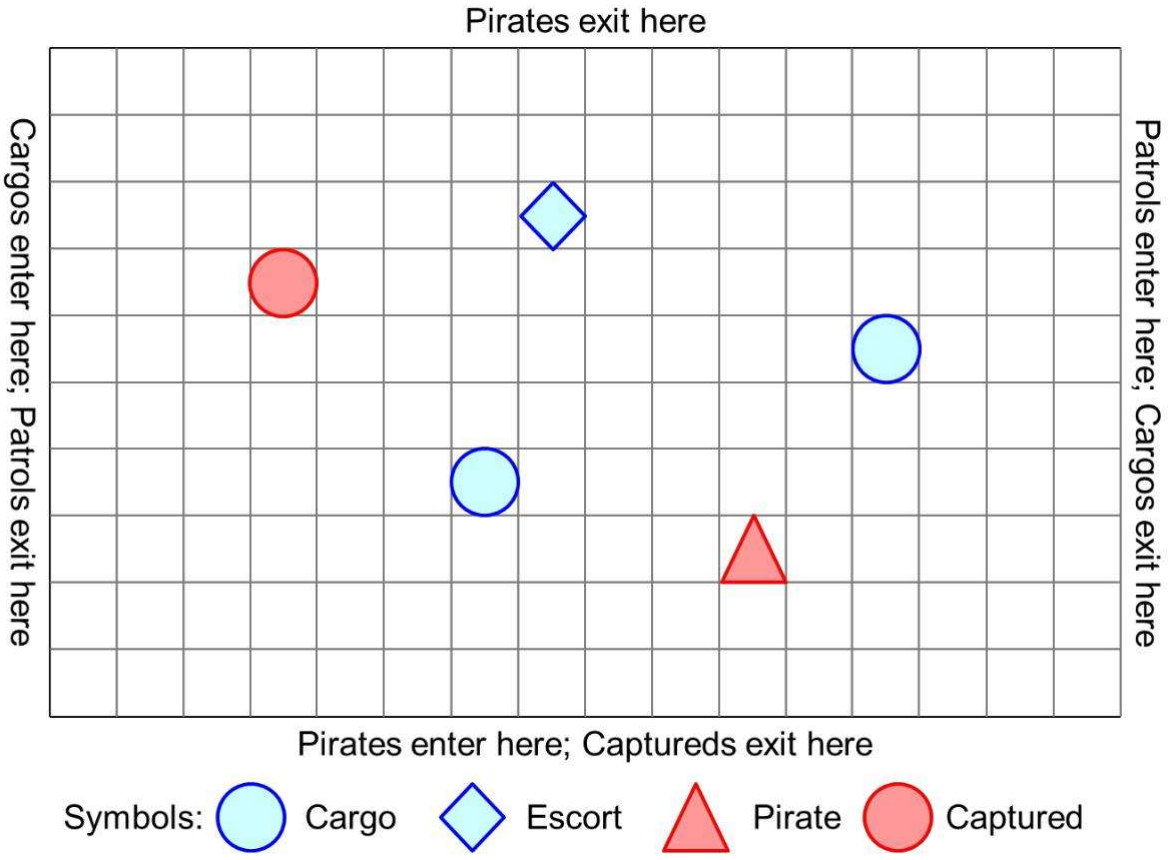


Figure 1. An example of the simulation map display.

Features and requirements

- Movement. The ocean area of operations for the simulation will be a rectangular grid of square cells, 35 cells (horizontal) x 20 cells (vertical). This area is featureless ocean, i.e., there are no continents or islands. This area will be referred to as the *map*. The individual square cells within the map will be referred to as *grids*. Each grid can be identified to by its (x, y) coordinates, with $0 \le x \le 34$ and $0 \le y \le 19$. Grid (0, 0) is in the lower left (southwest) corner and grid (34, 19) is in the upper right (northeast) corner.
- Entities. There are four types of entities in the simulation. At any given time in the simulation there may be 0 or more of each of these types of entities located in any of the grids.
  - Cargo ships. Merchant vessels carrying materials or oil. They are unarmed and cannot resist pirate attacks.
  - Naval patrol vessels. Armed warships that can defeat pirate boats or rescue cargo ships that have been captured by pirates.
  - Pirate boats. Small boats with armed pirate crews. They can capture unarmed cargo ships, but they cannot resist a naval patrol vessel.
  - Captured cargo ships. Cargo ships that have been captured by pirates.

  For brevity, in the remainder of this document these entity types will be referred to as *Cargos*, *Patrols*, *Pirates*, and *Captureds*.

.Time steps. The simulation proceeds in discrete time steps, each representing one hour of real time. In each time step, each of the entities currently active in the simulation will move, following rules detailed later for each entity type.

.Cargos.

- In each time step, Cargos already on the map move 1 grid directly to the east. A Cargo that starts a time step in a grid on the east edge of the map exits the map and is removed from the simulation.
- In each time step, the probability of a new Cargo entering the map is 0.50. (See the technical notes for an explanation of probabilistic events.) If a new Cargo does enter, it is placed in a randomly selected grid on the west edge of the map, with all grids on the west edge equally likely. (See the technical notes for an explanation of randomly selecting a grid. In the time step they enter the map, new Cargos do not move,
  i.e., entering the map is considered to be their movement in that time step.

5. Patrols.
   - In each time step, Patrols already on the map move 2 grids directly to the west. A Patrol that starts a time step in a grid on the west edge of the map exits the map and is
   - removed from the simulation.

     In each time step, the probability of a new Patrol entering the map is 0.25. If a new Patrol does enter, it is placed in a randomly selected grid on the east edge of the map, with all grids on the east edge equally likely. In the time step they enter the map, new Patrols do not move, i.e., entering the map is considered to be their movement in that time step.
6. Pirates.

   In each time step, Pirates already on the map move 1 grid directly to the north.

- A Pirate that starts a time step in a grid on the north edge of the map exits the map and
- is removed from the simulation.

In each time step, the probability of a new Pirate entering the map is 0.40. If a new Pirate does enter, it is placed in a randomly selected grid on the south edge of the map, with all grids on the south edge equally likely. In the time step they enter the map, new Pirates do not move, i.e., entering the map is considered to be their movement in that time step.

7. Captured's.
   - In each time step, Captured's already on the map move 1 grid directly to the south. A Captured that starts a time step in a grid on the south edge of the map exits the map and is removed from the simulation.
   - Captured's do not enter the map in the same way that Cargos, Patrols, and Pirates do. They are instead created in a manner described below.

1. Actions. There are three types of possible actions that may occur during each time step: *Captures*, *Defeats*, and *Rescues*. Each has a pre-condition that must be met for the action to occur.
2. Defeat. If, after both have moved, a Pirate and a Patrol are either in the same grid or adjacent grids (including diagonally adjacent), the Pirate is defeated by the Patrol. The Pirate entity is removed from the map and the simulation.
3. Capture. If, after both have moved, a Pirate and a Cargo are either in the same grid or adjacent grids (including diagonally adjacent), the Cargo is captured by the Pirate. The Cargo entity is changed to a Captured entity, and the Pirate entity is removed from the map and the simulation. Beginning with the next time step the new Captured will follow the rules of movement and actions for Captureds.
4. Rescue. If, after both have moved, a Captured and a Patrol are either in the same grid or adjacent grids (including diagonally adjacent), the Captured is rescued by the Patrol. The Captured entity is changed to a Cargo entity. Beginning with the next time step the new Cargo will follow the rules of movement and actions for Cargos.
5. No action. Other than the specific actions listed earlier, if two entities end a time step in the same grid or adjacent grids, there is no action, and both entities procedure normally in the next time step.
6. Sequence. The sequence of execution in each time step shall be as follows:
   a. Move all entities already on the map.
   b. Determine if any new Cargo, Pirate, and Patrol entities enter the map, in that order, placing the new entities on the map as explained earlier.
   c. Determine if any Defeat, Capture, or Rescue actions occur, in that order, changing entity types and/or removing entities as explained earlier.
7. Counts. The system shall count all of these simulation events: Cargos entered, Cargos exited, Patrols entered, Patrols exited, Pirates entered, Pirates exited, Pirates defeated, Cargos Captured, Cargos exited, Captureds rescued.
8. Map display. The system shall have a graphical user interface. The interface shall display the map and all entities on it, in a manner similar to Figure 1. Note that Figure 1 does not show all the required features of the interface; the placement and format of the additional

required features is up to the implementers. Note also that Figure 1 shows a 16 x 10 map; this is for illustrative purposes only; the map should have the dimensions specified earlier.

9. Counts display. The interface shall also display the time step number and the current values of the counts listed earlier. The display of map, time step, counts shall be updated after each time step.
10. Controls. The interface shall have *start*, *pause*, *single step*, and *end* buttons that control the simulation execution. The system shall start the simulation at the beginning of time step 1 in a paused state. Clicking the start button from a paused state will cause the simulation to execute one time step after another until either the pause button or the end button is clicked. Clicking the single step button from a paused state shall cause the simulation to execute one time step and then re-enter a paused state. Clicking the single step button while the simulation is executing (because the start button was clicked earlier) shall have no effect. Clicking the end button ends the simulation, whether it was executing or paused.
11. Calibration. Once the system has been implemented and tested according to these instructions, it should be calibrated. To calibrate it, adjust the probabilities of Cargos, Patrols, and Pirates entering the map on a given time step either up or down as needed until executing a simulation usually produces an interesting sequence of events that include all three of the possible actions.

Constraints

1. The system must be portable. The system must run on either a PC or a Macintosh.
2. The system should be robust; i.e., no sequence of user actions should cause the system to crash.

Technical notes

- *Probabilistic events*. For events which may occur with some probability, e.g., a new Cargo entering the simulation, use the following procedure. Generate a random number R uniformly distributed such that $0 \le R < 1$; most programming languages have a function or method that will do this. Compare R to the probability p of the event occurring. If $R < p$, the event occurs, otherwise the event does not occur. For example, in any given time step a new Cargo will enter the simulation with probability $p = 0.50$. For each time step, generate R and test $R < 0.4$; if true, a new Cargo enters the simulation.
- *Random selection*. To choose one of n grid cells randomly, use the following procedure. Generate a random number R uniformly distributed such that $0 \le R < 1$. Then calculate $k = \text{floor}(n \cdot R)$; k will be an integer in the range $0 \le k < n$. Use k as the grid's x or y coordinate as needed.

Questions, clarifications, and more information

For more information on this project, contact Dr. Mikel D. Petty. He can be reached at pettym@uah.edu or 256-824-6140. His office hours are Monday and Wednesday, 2:00-5:30; his office is OKT N300G.

Team E
Nathan Moore
Aaron Mendez
Luke Tomlinson
Jack Bragg

A.2: Dr. Delugach—CS 499 Senior Project Assignment: Somalian Pirates Simulation

Rationale

The International Maritime Bureau reported that over the period 2005-2012 nearly 1,000 ships were fired at, chased, boarded, or hijacked by pirates. The pirates, often operating from bases in Somalia, usually attacked ships in the northwest portion of the Indian Ocean, especially in the Gulf of Aden. The attacks were carried out nearly daily during some periods of the year, and often occurred despite the presence of naval patrol vessels in the area. Vessels hijacked by pirates were usually ransomed by their owners and insurers, sometimes for as much as tens of millions of dollars, which is usually a small percentage of the value of the vessel and its cargo.

As a first step towards reducing or eliminating future pirate attacks, you have been asked to develop a simulation of merchant shipping, naval patrols, and pirate attacks.

This is a three (3) person project.

Features and Requirements

The features listed below shall be included in the software.

1. **Movement**. The ocean area of operations for the simulation will be a rectangular grid of square cells, 400 miles x 100 miles (vertical). This area is on the northern border of Somalia in the Gulf of Arden (see image below) and stretches from Saylac in the west to Caluula in the east and 100 miles out to sea. This area will be referred to as the ***map***. The individual square cells within the map will be referred to as ***grids***. Each grid can be identified by its (x, y) coordinates, with $0 \le x \le 400$ and $0 \le y \le 100$. Grid (0, 0) is in the upper left (northwest) corner and grid (400, 100) is in the lower right (southeast) corner.

Figure 1. Area for the simulation map display.

2. **Entities**. There are four types of entities in the simulation. At any given time in the simulation there may be 0 or more of each of these types of entities located in any of the grids: (1) Cargo ships. Merchant vessels are unarmed and cannot resist pirate attacks. (2) Naval patrol vessels. Armed warships that can defeat pirate boats and rescue cargo ships that have been captured by pirates. (3) Pirate boats. Small boats with armed pirate crews. They can capture cargo ships, but they cannot resist a naval patrol vessel. (4) Captured cargo ships. Cargo ships that have been captured by pirates. For brevity, in the remainder of this document these entity types will be referred to as *Cargos*, *Patrols*, *Pirates*, and *Captures*.
3. **Time steps**. The simulation display shall be updated at a rate of once each second, each second represents 5 minutes of real time. In each time step, each of the entities currently active in the simulation will move, following rules given below for each entity type. The simulation may also be run in 2X, 10X, or 20X speed changeable while the simulation is running.
   a. **Cargos**. In each time step, Cargos already on the map move 1 grid directly to the east. A Cargo that starts a time step in a grid on the east edge of the map exits the map and is removed from the simulation. In each time step, the probability of a new Cargo entering the map is 0.50. (See the technical notes below for an explanation of probabilistic events.) If a new Cargo does enter, it is placed in a randomly selected grid on the west edge of the map, with all grids on the west edge equally likely. (See the technical notes for an explanation of randomly selecting a grid.) In the time step they enter the map, new Cargos do not move, i.e., entering the map is considered to be their movement in that time step.
   b. **Patrols**. In each time step, Patrols already on the map move 2 grids directly to the west. A Patrol that starts a time step in a grid on the west edge of the map exits the map and is removed from the simulation. In each time step, the probability of a new Patrol entering the map is 0.25. If a new Patrol does enter, it is placed in a randomly selected grid on the east edge of the map, with all grids on the east edge equally likely. In the time step they enter the map, new Patrols do not move, i.e., entering the map is considered to be their movement in that time step.
   c. **Pirates**. In each time step, Pirates already on the map move 1 grid directly to the north. A Pirate that starts a time step in a grid on the north edge of the map exits the map and is removed from the simulation. In each time step, the probability of a new Pirate entering the map is 0.40. If a new Pirate does enter, it is placed in a randomly selected grid on the coast on the south of the map, with all grids on the south edge equally likely. In the time step they enter the map, new Pirates do not move, i.e., entering the map is considered to be their movement in that time step.
   d. **Captures**. In each time step, captures already on the map move 1 grid directly to the south accompanied by the capturing pirate ship. A Captured that starts a time step in a grid on the south edge (coast) of the map exits the map and is removed from the simulation. Captures do not enter the map in the same way that Cargos, Patrols, and Pirates do. They are instead created in a manner described below.
4. **Actions**. Following are the possible actions that may occur during each time step. Each has a pre-condition that must be met for the action to occur.

a. **Defeat**. If, after both have moved, a Pirate and a Patrol are in the same 3 x 3 grid square, the Pirate is defeated by the Patrol. The Pirate entity is removed from the map and the simulation.

b. **Capture**. If, after both have moved, a Pirate and a Cargo are in the same 3 x 3 grid square, the Cargo is captured by the Pirate. The Cargo entity is marked as a Captured entity, and the Pirate entity is moved to the same grid square. Beginning with the next time step the new Captured and its accompanying Pirate will follow the rules of movement and actions for Captures.

c. **Rescue**. If, after both have moved, a Captured and a Patrol are in the same 3 x 3 grid square, the Captured is rescued by the Patrol. The Captured entity is changed to a Cargo entity and the Pirate is removed from the map and the simulation. Beginning with the next time step the new Cargo will follow the rules of movement and actions for Cargos.

d. **Evade**. A Cargo shall be able to evade an individual Pirate one-time by moving 1 diagonal grid north and east. The Cargo can evade if a Pirate is in the same 4 x 4 grid, not in the 3 x 3 grid, can remain on the map and has not evaded this same Pirate already.

d. **No action**. Other than the specific actions listed above, if two entities end a time step in the same grid or adjacent grids, there is no action, and both entities proceed normally in the next time step.

5. **Sequence**. The sequence of execution in each time step shall be as follows: (1) Move all entities already on the map. (2) Determine if any new Cargo, Pirate, and Patrol entities enter the map, in that order, placing the new entities on the map as explained above. (3) Determine if any Defeat, Capture, Rescue or Evade actions occur, in that order, changing entity types and/or removing entities as explained above.
6. **Counts**. The system shall count all of these simulation events: Cargos entered, Cargos exited, Patrols entered, Patrols exited, Pirates entered, Pirates exited, Pirates defeated, Cargos Captured, Cargos exited, Captures rescued, evades not captured, Evades captured.
7. **Map display**. The system shall have a graphical user interface. The interface shall display the map and all entities on it, in a manner similar to Figure 1 but with the coast line closer to the bottom of the display. It is expected that the GUI created will resemble an elevated view looking down on the ocean with each ship represented by an appropriate image. Images should be oriented to indicated the direction of travel of the ship.
8. **Counts display**. The interface shall also display the time step number and the current values of the counts listed earlier. The display of map, time step, counts shall be updated after each time step.
9. **Controls**. The interface shall have ***start, pause, single step, simulation speed (1X, 5X, 10X, 20X)*** and ***end*** buttons that control the simulation execution. The system shall start the simulation at the beginning of time step 1 in a paused state. Clicking the start button from a paused state will cause the simulation to execute one-time step after another until either the pause button or the end button is clicked. Clicking the single step button from a paused state shall cause the simulation to execute one-time step and then re-enter a paused state. Clicking the single step button while the simulation is executing (because the start button was clicked earlier) shall have no effect. Clicking the end button ends the simulation, whether it was executing or paused.

10. **Calibration**. Once the system has been implemented and tested according to these instructions, it should be calibrated. To calibrate it, adjust the probabilities of Cargos, Patrols, and Pirates entering the map on a given time step either up or down as needed until executing a simulation usually produces an interesting sequence of events that include all three of the possible actions.

Technical notes

Probabilistic events. For events which may occur with some probability, e.g., a new Cargo entering the simulation, use the following procedure. Generate a random number R uniformly distributed such that $0 \leq R < 1$; most programming languages have a function or method that will do this. Compare R to the probability p of the event occurring. If $R < p$, the event occurs, otherwise the event does not occur. For example, in any given time step a new Cargo will enter the simulation with probability $p = 0.50$. For each time step, generate R and test $R < 0.5$; if true, a new Cargo enters the simulation.

Random selection. To choose one of n grid cells randomly, use the following procedure. Generate a random number R uniformly distributed such that $0 \leq R < 1$. Then calculate $k = \text{floor}(n * R)$; k will be an integer in the range $0 \leq k < n$. Use k as the grid's x or y coordinate as needed.

Team E
Nathan Moore
Luke Tomlinson
Aaron Mendez
Jack Bragg

A.3: Team E – Approved Combined Statement Of Work

CS 499 Senior Project

Project Assignment

Somali Pirates Simulation

Rationale

The International Maritime Bureau reported that over the period 2005-2012 nearly 1,000 ships were fired at, chased, boarded, or hijacked by pirates. The pirates, often operating from bases in Somalia, usually attacked ships in the northwest portion of the Indian Ocean, especially in the Gulf of Aden. The attacks were carried out nearly daily during some periods of the year, and often occurred despite the presence of naval patrol vessels in the area. Vessels hijacked by pirates were usually ransomed by their owners and insurers, sometimes for as much as tens of millions of dollars, which is usually a small percentage of the value of the vessel and its cargo.

As a first step towards reducing or eliminating future pirate attacks, you have been asked to develop a highly simplified simulation of merchant shipping, naval patrols, and pirate attacks.

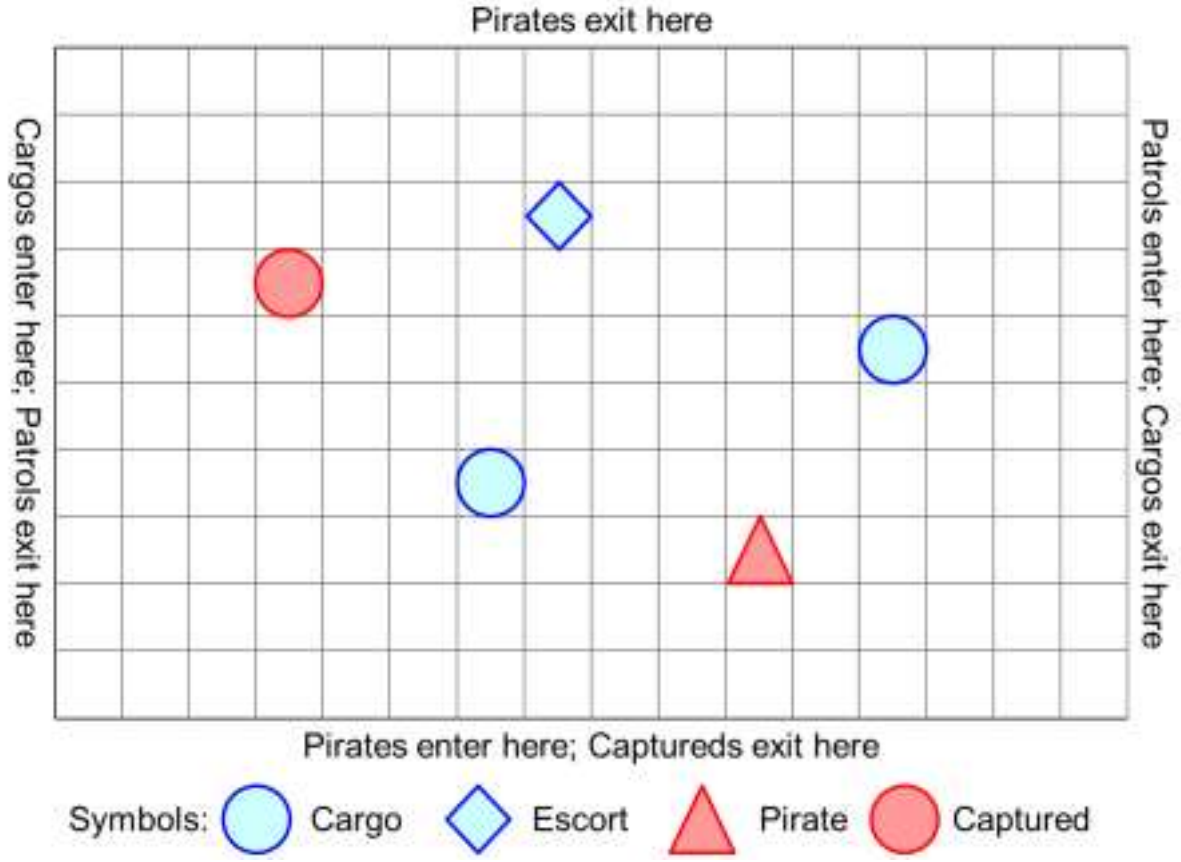


Figure 1. An example of the simulation map display.

Features and requirements

The features listed below shall be included in the software.

1. *Movement.* The ocean area of operations for the simulation will be a rectangular grid of square cells, 400 miles x 100 miles (vertical). This area is featureless ocean, i.e., there are no continents or islands. This area will be referred to as the *map*. The individual square cells within the map will be referred to as *grids*. Each grid can be identified by its (*x*, *y*) coordinates. Grid (0,0) is in upper left (northwest) corner and grid (400, 100) is in lower right (southeast) corner.
2. *Entities.* There are four types of entities in the simulation. At any given time in the simulation there may be zero or more of each of these types of entities located in any of the grids.
    1. *Cargo ships.* Merchant vessels are unarmed and cannot resist pirate attacks.
    2. *Naval patrol vessels.* Armed warships that can defeat pirate boats and rescue cargo ships that have been captured by pirates.
    3. *Pirate boats.* Small boats with armed pirate crews. They can capture cargo ships, but they cannot resist a naval patrol vessel.
    4. *Captured cargo ships.* Cargo ships that have been captured by pirates.
    5. For brevity, in the remainder of this document these entity types will be referred to as *Cargos*, *Patrols*, *Pirates*, and *Captures*.
3. *Time steps.* The simulation display shall be updated at a rate of once each second, each second represents 5 minutes of real time. In each time step, each of the entities currently active in the simulation will move, following rules given below for each entity type. The simulation may also be ran in 2x, 10x, or 20x speed changeable while the simulation is running.
    1. *Cargos.*
        1. In each time step, Cargos already on the map move 1 grid directly to the east. A Cargo that starts a time step in a grid on the east edge of the map exits the map and is removed from the simulation.
        2. In each time step, the probability of a new Cargo entering the map is 0.50. (See the technical notes for an explanation of probabilistic events.) If a new Cargo does enter, it is placed in a randomly selected grid on the west edge of the map, with all grids on the west edge equally likely. (See the technical notes for an explanation of randomly selecting a grid. In the time step they enter the map, new Cargos do not move, i.e., entering the map is considered to be their movement in that time step.
    2. *Patrols*.

1. In each time step, Patrols already on the map move 2 grids directly to the west. A Patrol that starts a time step in a grid on the west edge of the map exits the map and is removed from the simulation.
2. In each time step, the probability of a new Patrol entering the map is 0.25. If a new Patrol does enter, it is placed in a randomly selected grid on the east edge of the map, with all grids on the east edge equally likely. In the time step they enter the map, new Patrols do not move, i.e., entering the map is considered to be their movement in that time step.

3. *Pirates*.
    1. In each time step, Pirates already on the map move 1 grid directly to the north. A Pirate that starts a time step in a grid on the north edge of the map exits the map and is removed from the simulation.
    2. In each time step, the probability of a new Pirate entering the map is 0.40. If a new Pirate does enter, it is placed in a randomly selected grid on the coast on the south of the map, with all grids on the south edge equally likely. In the time step they enter the map, new Pirates do not move, i.e., entering the map is considered to be their movement in that time step.
4. *Captures*.
    1. In each time step, captures already on the map move 1 grid directly to the south. A Captured that starts a time step in a grid on the south edge (coast) of the map exits the map and is removed from the simulation.
    2. Captures do not enter the map in the same way that Cargos, Patrols, and Pirates do. They are instead created in a manner described below.

4. *Actions*. Following are the possible actions that may occur during each time step. Each has a pre-condition that must be met for the action to occur.
    1. *Defeat*.
        1. Pre-condition: after each entity moves, a Patrol and Pirate occupy the same grid or contiguous grids.
        2. Action: The Pirate is defeated by the Patrol. The Pirate entity is removed from the map and the simulation.
    2. *Capture*.
        1. Pre-condition: after each entity moves, a Cargo and Pirate occupy the same grid or contiguous grids.
        2. Action: The Cargo is captured by the Pirate and becomes a Capture entity. The Pirate will accompany the Capture by occupying the same grid.
    3. *Rescue*.

1. Pre-condition: after each entity moves, a Capture and Patrol occupy the same grid or contiguous grids.
2. Action: The Capture is rescued by the Patrol and becomes a Cargo entity, and the Pirate is removed from the map and the simulation.

4. *Evade.*
    1. A Cargo shall be able to evade an individual Pirate one-time by moving 1 diagonal grid north and east. The Cargo can evade if a Pirate is in the same 4 x 4 grid, not in the 3 x 3 grid, can remain on the map and has not evaded this same Pirate already.

5. *No action.* Other than the specific actions listed above, if two entities end a time step in the same grid or adjacent grids, there is no action, and both entities proceed normally in the next time step.

5. *Sequence.* The sequence of execution in each time step shall be as follows:
    1. Move all entities already on the map.
    2. Determine if any new Cargo, Pirate, and Patrol entities enter the map, in that order, placing the new entities on the map as explained above.
    3. Determine if any Defeat, Capture, Rescue or Evade actions occur, in that order, changing entity types and/or removing entities as explained above.
6. *Counts.* The system shall count all of these simulation events: Cargos entered, Cargos exited, Patrols entered, Patrols exited, Pirates entered, Pirates exited, Pirates defeated, Cargos captured, Cargos exited, Captures rescued, successful evade attempts, and unsuccessful evade attempts.
7. *Map display.* The system shall have a graphical user interface. The interface shall display the map and all entities on it, in a manner similar to Figure 1. Note that Figure 1 does not show all the required features of the interface; the placement and format of the additional required features is up to the implements. Note also that Figure 1 shows a 16 x 10 map; this is for illustrative purposes only; the map should have the dimensions specified earlier.
8. *Counts display.* The interface shall also display the time step number and the current values of the counts listed above. The display of the map, time step number, counts shall be updated after each time step.
9. *Controls.* The interface shall have **start**, **pause**, **single step**, **simulation speed** (2x, 10x, or 20x) and **end** buttons that control the simulation execution. The system shall start the simulation at the beginning of time step 1 in a paused state. Clicking the start button from a paused state will cause the simulation to execute one time step after another until either the pause button or the end button is clicked. Clicking the single step button from a paused state shall cause the simulation to execute one time step and then re-enter a paused state. Clicking the single step

button while the simulation is executing (because the start button was clicked earlier) shall have no effect. Clicking the end button ends the simulation, whether it was executing or paused.

10. *Calibration.* Once the system has been implemented and tested according to these instructions, it should be calibrated. To calibrate it, adjust the probabilities of Cargos, Patrols, and Pirates entering the map on a given time step either up or down as needed until executing a simulation usually produces an interesting sequence of events that include all three of the possible actions.

Technical notes

1. *Probabilistic events.* For events which may occur with some probability, e.g., a new Cargo entering the simulation, use the following procedure. Generate a random number R uniformly distributed such that $0 \leq R < 1$; most programming languages have a function or method that will do this. Compare R to the probability p of the event occurring. If $R < p$, the event occurs, otherwise the event does not occur. For example, in any given time step a new Cargo will enter the simulation with probability $p = 0.50$. For each time step, generate R and test $R < 0.5$; if true, a new Cargo enters the simulation.
2. *Random selection.* To choose one of n grid cells randomly, use the following procedure. Generate a random number R uniformly distributed such that $0 \leq R < 1$. Then calculate k = floor(n * R); k will be an integer in the range $0 \leq k < n$. Use k as the grid's x or y coordinate as needed.

A.4: Email Correspondence – Customer Desired Project Attributes

A.1—Petty	**A.2—Delugach**
35 cell (horizontal) by 25 cell (vertical) area	400 x 100 mile (vertical) area – unclear instruction please elaborate which value is horizontal and vertical if this is the preferred choice
Featureless area	Map using provided Somaliland Map
Grid(0,0) is bottom left, (34,19) is top right	Grid(0,0) is in upper left corner and (400,100) is in lower right corner
No time step user controls exist	Time steps (1x, 2x, 10x, 20x) user speed control – can change while simulation is running
Pirates enter randomly on the bottom [(0,0)-(0,19)] row	Pirates start near coast at south of map (there is a curve in the map in the south-west and no explicit grid size is defined) – requires more to execute the coast line and some grids will not ever spawn ships
“Captures” (Captured Cargo Ships) absorb Pirate ship and begin their new trajectory	“Captures” (Captured Cargo Ships) are accompanied by the capturing pirate ship within same grid (occupy same grid square)
No Evade ability Specified	Cargo Ship – Evade ability if this is the choice to exist simulation also requires counting evades that are captured as well as evades that escape (if ship escapes in one turn does this increment the counter or; if ship escapes must it complete its journey across the map before either the “evades captured” or “evades not captured” counter increment) Ship can evade if possible in one turn and then continues until it reaches the edge of the map.
Calibration matches the amount of ship abilities can be left unchanged if Evade from Outline B is	Calibration references 3 possible actions for any ship to execute yet there are 4 (5 being no action)

not wanted	actions available for ships to enact We'll have to discuss.

Appendix C – Presentation Slides

C.1: Software Development Plan Presentation

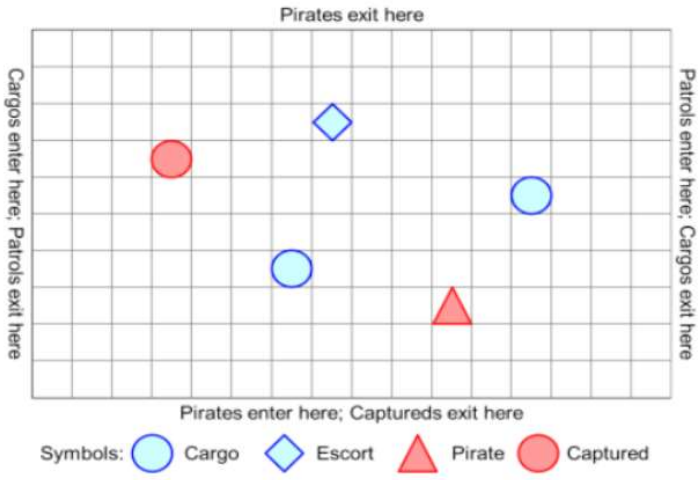
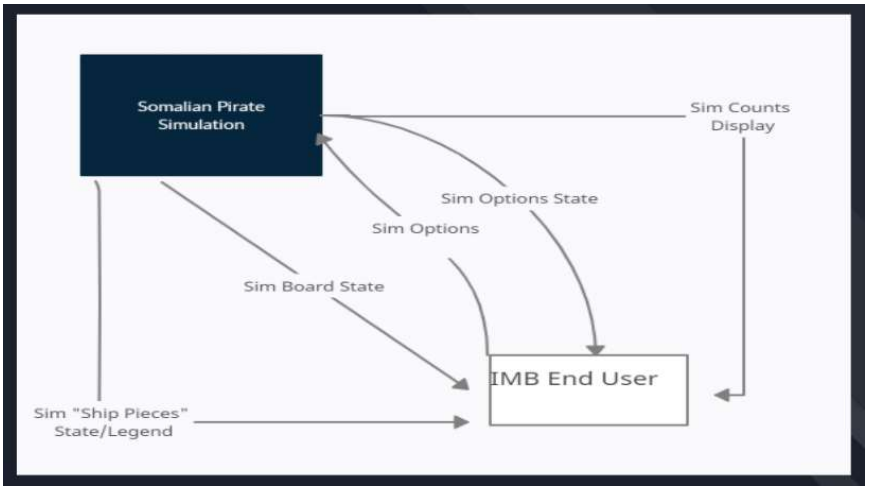
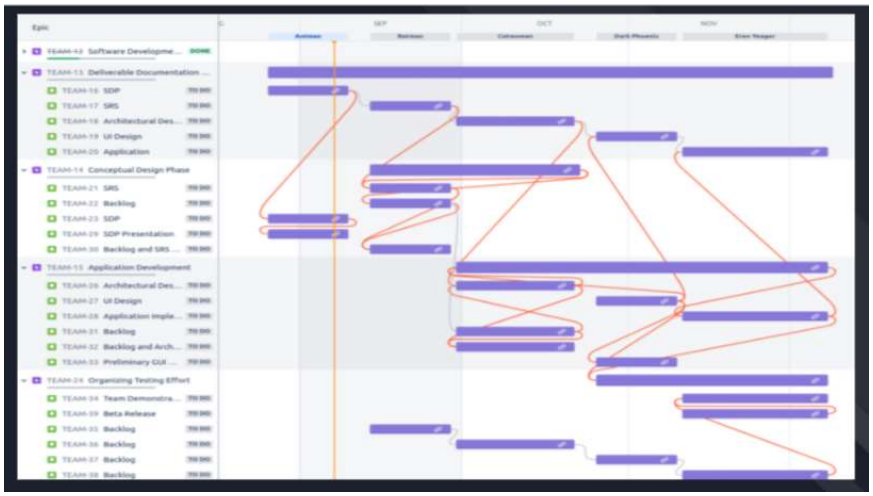


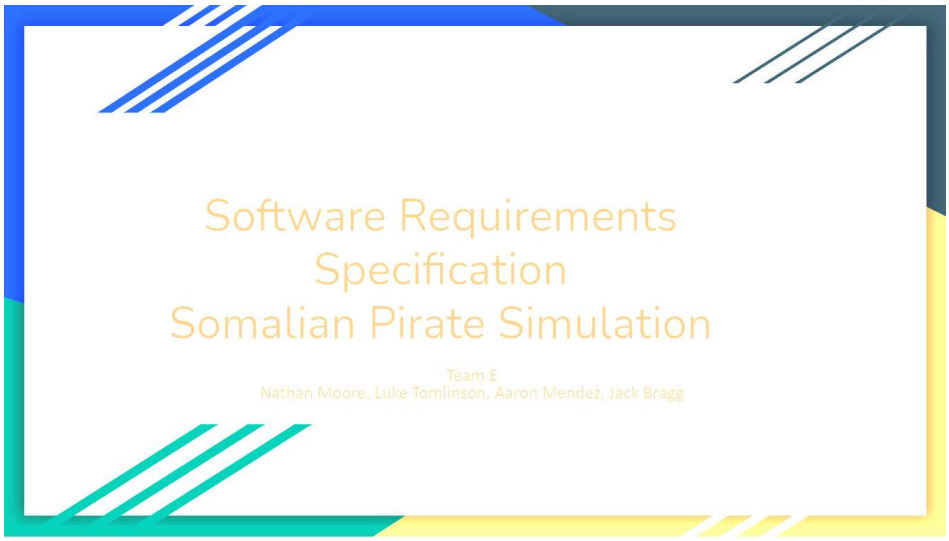
Figure 1. An example of the simulation map display.

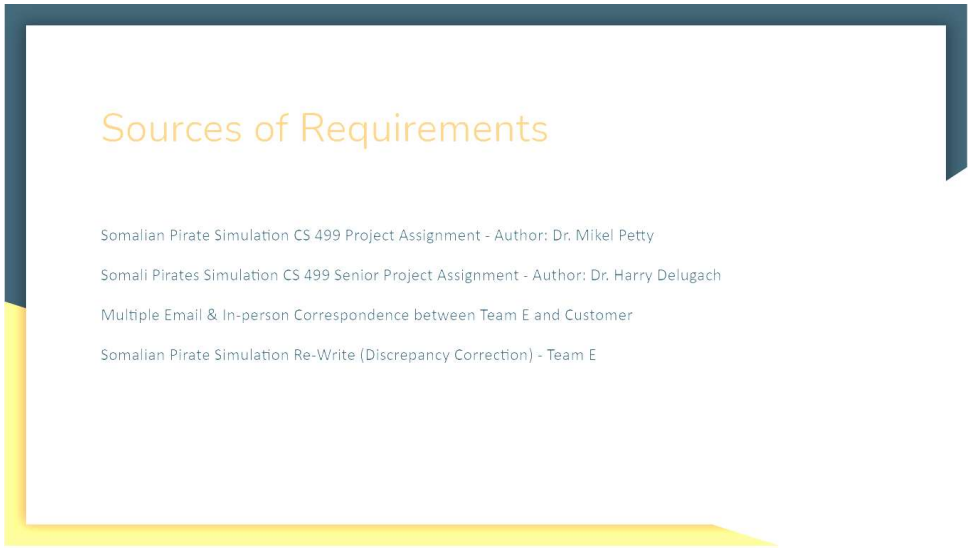

Somalian Pirate Simulation
Sim Counts Display
Sim Options State
Sim Options
Sim Board State
IMB End User
Sim "Ship Pieces" State/Legend




Thank you!

C.2: SRS Presentation Slides





Product Level Requirements

- Reliability
  - Application Runs Properly, Display Current Information Correctly, User Interface is Responsive
- Portability
  - Ability to Distribute on WIndows and Mac
- Functionality/Performance
  - Sprite Frame-by-Frame Animation
  - Memory Allocation

Current Backlog

- 54 issues
- 5 sprints
- 4 epics
- 4 team members
- Backlog
- Ships
  - Movements
  - Collision Interactions

Pre-Alpha Application Development Demo