**CS4344**

**Assignment 3 Report:**

**Team Members:**

Nguyen Trong Son (A0088441B)

Pham Minh Viet (A0088454U)

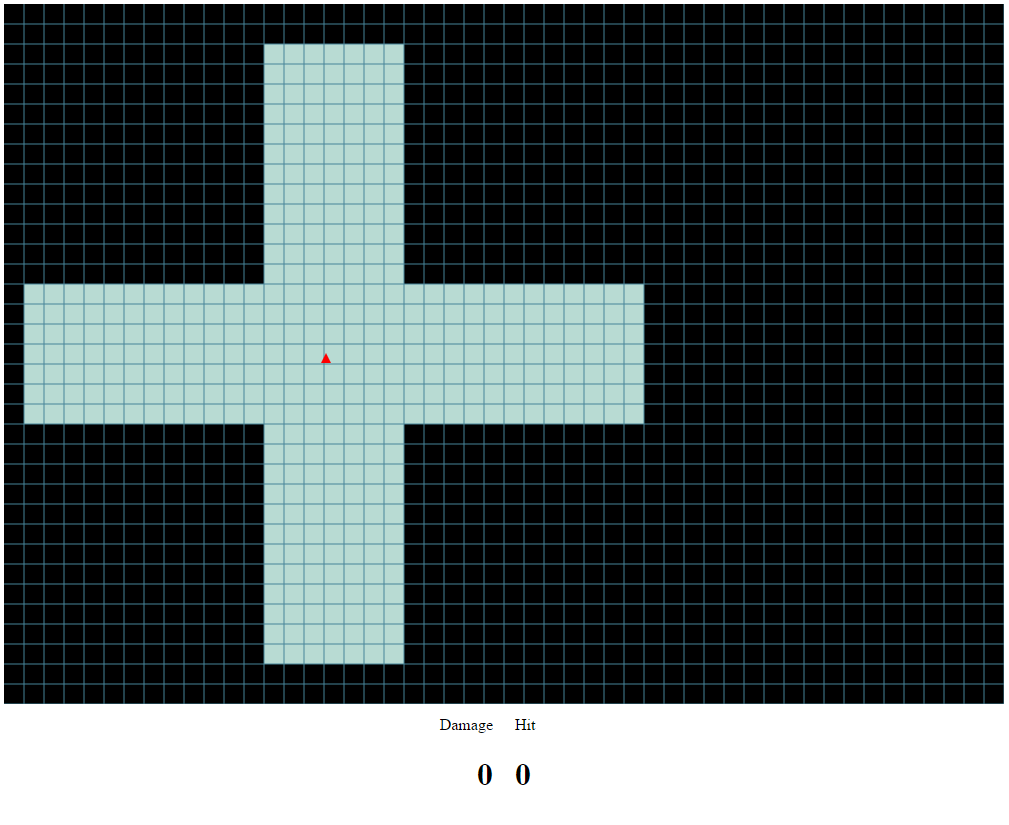
Project Repository:

[github.com/SeanNguyen/nus-cs4344.git](https://github.com/SeanNguyen/nus-cs4344.git)

1. **STANDARD REQUIREMENT**
2. **Approach strategy:**

The game world is divided into small individual cells. Each cells is a group channel by itself and a player can subscribe to a group (cell) to receive information sent to that group.

In this particular game we divide the canvas into 50x35 cells. Therefore the number of cell is not too many that cost overhead calculation and also the cell is small enough to have a reasonable accuracy.



1. **Area of Interest:**

Width

Height

Ship of player A Rocket of Player A Rocket of Player A

Area of Interest of Player A

AOI of a player is a combination of AOI of the ship and AOI of its rockets.

Ship AOI:

We call the AOI of the ship above is the cross area and its dimension is legend as above. The width of the cross area does not have to be so long because the rocket is slow and cannot have any impact on the ship from a far distance. On the other hand the height of the cross area need to be big so the play have enough time the estimate the coming ships and rockets flying on it route.

In this project we set the width spans 30 cells and the height spans 7 cells

Rockets AOI:

Player only need to see what their rocket is about to hit. Therefore, rocket AOI will be sidelong to the same direction with the direction of the Rocket.

Ship AOI have the save behaviour as Ship:

A ship can go off at an edge and appear on the opposite edge so player will interest in what happen on the other edge when they nearly go off an edge as well. Therefore, when a Ship AOI near an edge it will not span to the othe side of the canvas as well.

Figure show how a ship AOI behaves

Figure show how a ship AOI behaves

Rocket AOI, on the other hand, will not have this behavior, it will just go off when reaching the edge of the game world.

1. **Event Handling:**

A new player join the room:

This player still need to be introduced and need to know all the others in the room so this event will be kept as original.

A ship moves to a new cell:

1. Update the AOI of the ship:

* Remove the ship subscription on the old AOI’s cells
* Subscribe to new AOI’s cells

2. From the subscribers of the cell, it can tell who is the others guys listening to this cell other than the old subscribers from the old cell. By this it know there will be a need for an update message. Then the ship will send an update message to the necessary player (a “turn” message in the actual implementation because it has the same effect).

**CELL 2**

Subscribers:

A, B, C

**CELL 1**

Subscribers:

A, C

For example, a ship X move from cell 1 to cell 2. Cell 2 has 3 subscribers is A, B, and C but A, B are the old subscribers from cell 1 so only C need to be update about X.

Code implementation: MMOServer.js -> changeShipCell() function. This function is used in game loop.

A ship turns:

Only need to notify the subscribers of the cell where this event happens.

Code implementation: MMOServer.js -> start() function -> on case “turn” and “fire” of the socket events.

A ship fires:

Similar to the “turn” event above, we only need to notify the subscribers of the cell where this event happens.

Code implementation: MMOServer.js -> start() function -> on case “fire” of the socket events.

A rocket move:

Because a “fire” event is not broadcast so some play will not see a rocket. So the rocket will be notify to this player when the rocket move to the AOI of the ship.

1. Similar to the Ship, first, the rocket update its AOI:

* Remove the ship subscription on the old AOI’s cells
* Subscribe to new AOI’s cells

1. Also similar to the ship case, from the subscribers of the cell, it can tell who is the others ships listening to this cell other than the old subscribers from the old cell. By this it know there will be a need for an update message. Then the ship will send an update message to the necessary player (a “fire” message in the actual implementation because it has the same effect).

Code implementation: MMOServer.js -> changeRocketCell () function. This function is used in game loop.

On Hit event:

We decide this event will be broadcasted because all client need to know when a rocket be removed. Otherwise, the rocket on some client will be just visual and have no impact on anything and this can cause some confusion for player.

1. **Checking if a rocket hits a ship:**

Base on the rocket current position, we can get the cell and the subscribers of that cell. We will loop through only these subscribers instead of all ships.

The logic is that if a ship can be hit by that rocket, the ship must be on the same cell with the rocket. If the ship in the same cell with the rocket, the ship must be one of the subscribers of that cell.

Code Implementation: MMOServer -> gameLoop() function.

1. **BONUS – MULTIPLE SERVERS**

We created 4 servers. Each server is in charge of 1 region. Following is the server ids and the area that they are in charge of correspondingly.

|  |  |
| --- | --- |
| 0 | 1 |
| 3 | 2 |

Code to change server in client when a ship move from one region to another.