Simulation Game:
Emergency Management Training Online Simulator

Created By:
Marco de Freitas (19050739@sun.ac.za)

September
2017

Contents

1 Specifications Checklist

1.1 General

– Multiple users need to be able to experience the game separately ☑

– An instructor must be able to view, track and interact with the game players progress from a separate computer ☑

1.2 Player

– There should be sound responses when things happen like a new message/warnings. ☑

– A column to the side to act as a "Whatsapp chat" where the user can send messages to the 'ship crew' (instructor) and receive problem questions ☑

– The ship should be a stationary object on the bottom section of the screen and the (zoomed in) map should move as the ship progresses ☑

– A steering wheel should be on the side of the map (possible bottom right) which will allow the ship's course to be changed. The ship's direction should display in degrees. ☐

– A small map should be on the side (possibly top left) which can be clicked and will show the full map with the ship's position and the game end point. ☑

– Warning messages should pop up on the top section of the game map. ☑

– The game should end if the player crashes into a land mass. So there needs to be a button to 'dock' the ship. ☑

1.3 Instructor

– The instructor must be able to talk to talk to different users at once. So for each section of the instructors screen, there needs to be a way to switch between users ☑

– The instructor chat works the same as the users. ☑

– To the side of the base menu, there should be a permanent command section. ☑

– The instructor should be able to select multiple warnings (check boxes). And then there should be a button saying "Send to all ships". ☑

– If the instructor is on the whatsapp homepage type screen, the command will be sent to all ships. ☑

– If the instructor is on a whatsapp chat, the command section will still be to the side, but will instead say "Send to Player". So that the warning will only go to a specific user. ☑

– When the instructor clicks a user's message, to the side of that message a map of their progress pops up. They can't do anything to the map except view where the user is and track the user's progress. ☐

1.4 Additional

– Instructor can set vertical and horizontal velocities for all players (to simulate engine failures/ counter active winds, etc). ☑

– Instructor can set vertical and horizontal velocities for a selected player ☐

– A compass to show direction ☑

– A ship can move a maximum of 45 degrees. ☑

– When the instructor selects a player the map section mimics the players, indicating direction and visible land masses the player has as well as providing a button to view the entire map that also displays a line of the players course that is updated live. ☑

– A players status is shown to the instructor (Crashed, Docked, Out of Bounds, and Preparing to Dock). ☑

– The dock button slows the player down by half. ☑

2 How to Run the Simulation

The server hosting this game needs to have Node.js installed. Download the project files that consists of:

1. node_modules (directory)
2. www (directory)
3. app.json
4. package.json
5. index.js
6. README.md
7. Report.pdf

Place the files on a server with "SERVER-IP". In the directory execute the command node index.js. The game should now be hosted and available to clients on "http://SERVER-IP:8005". It is important that the Instructor joins first, as any player joining before the instructor might break the current session as well as not being visible to the Instructor. In your browser (does not need to be on the same machine as the server) connect to "http://SERVER-IP:8005", a black screen will load with a dialog box prompting your name, type "Instructor" (note capital "I"). The server will then print out "The Instructor has started the Simulation", at this point clients may connect to the server using the same url ("http://SERVER-IP:8005") and with any name except "Instructor". The player will load the map and immediately move 5 pixels forward every half second (default).

3 How to Change the Map

3.1 A Few Things to Note

You can download or create any image as the map, just be sure that the resolution of the image is larger than the resolution of the devices that will be using the game. Be sure that the map does not have a large amount of the colour white. Be sure that the colour of land masses is not similar to that of water, and that the color of the land masses is relatively uniform.

3.2 Creating a Map

You can create an ideal map to work for this game with MSPaint, set the canvas size to a large amount of pixels (larger than 3000), import island PNGs, position them appropriately ensure they are all the same colour, and that you know the rgb value of this colour. Paint the rest of the map a

shade of blue or with a water texture and save as jpg or png. You might also want to add a border to indicate the end of the map.

3.3 Changing Map

Place the image in www/img/. In the file www/messaging.js change the top three variables to suit that of your new map. set canvasHeight (line 8) to the images height in pixels, canvasWidth (line 9) to the images width in pixels, and img_path to the path to your image from the current file, ideally "../img/YOUR_MAP.png".

3.4 Setting the Collision Detection

In the file client.js locate the function detectCollison(x,y) on line 107 you will see an if statement comparing p[0], p[1], and p[2] with some values, these are the respective RGB values of the land masses, you need to replace the current values with the RGB value of your land masses. Multiple shades of the same color can be used by finding the RGB range of your land mass and applying it to the if statement appropriately.

If you are struggling to find the RGB colour of your land mass un-comment the console.log(..) on line l06, and put the rest of the function in comments (make no dock emits or set END to true), start the game as a client and sail on top of a land mass, it will then display the colour detected by my program in your browser console.

4 Possible Issues and Solutions

Scaling is a possible issue, placing the avatar off center, this is due to the necessity of a strict pixel off set to place the image in the center, and the different resolutions and scaling of devices, a fix for this would be to change the offset variable in the file messaging.js on line 12 to a value that centers the avatar on your device.

Clients immediately crash after connecting, the clients are spawning on a landmass, be sure that the lower left hand corner of your map if entirely water, if the problem persists or clients run out of bounds immediately or soon after starting, it means that the vertical offset is incorrect for your device, in the file client.js on line 9 change the variable initY to the vertical size in pixels of the clients device.

Other issues are unknown at this time, but as the majority of testing was done on desktop, there are bound to be compatibility issues with the Microsoft surface, the developer is willing to sort these issues out, as well as adding minor tweaks and preferences before the game is used officially.