Star Search Simulation System Proposal

CS6310 - Spring 2020

Group 25 : (Sungin Baek sbaek33@gatech.edu, Leonardo Gabriel Puy lpuy3@gatech.edu, Nathaniel Yohannes nyohannes3@gatech.edu, Ye Shen yshen61@gatech.edu, Mahesh Chandra Sehalli Jagadish mjagadish3@gatech.edu)

Introduction

This document contains our group's three (3) proposed project modifications to the Star Search Simulation System. These changes aim to improve the overall effectiveness of the current system. The first three sections discuss each proposed change based on questions from Heilmeier's Catechism. The fourth section addresses those remaining questions with answers common to all three modifications. We also include images of our mockup screenshot and the software development technical stack we plan to use.

Graphical User Interface (GUI)

For our first modification, we are proposing to develop a Graphical User Interface that will enable interactive abilities to load a scenario file, control the pace of the simulation run, display the state of the simulation, display the simulation status, manual control of the drone, and export the final report. Most of these abilities either don't exist or are done today as console application in the existing iteration of the product. This current practice is limiting because most of the time users are not sure what's happening. This forces users to be familiar with Java to use the application, which is not very user friendly. They also cannot control the pace of the simulation run. For manual control, users have to type actions in the terminal, which is cumbersome. Users also cannot save the final report.

Our approach to adding a GUI gives users/space explorers a more streamlined and natural interaction. Users could see the simulation's visual and status updates in real time, and can control the pace of the simulation run. They could also do manual control by pushing buttons, and have the option to export the final report. If successful, the GUI will provide a more advanced space exploration simulation experience to users and will broaden the potential user pool. Since Java or command line knowledge is not needed to run the program, the user experience is enhanced by avoiding the need to type in terminal, and those who are not used to consoles and Java can use this application easily. The visualization will also help users make a wiser decision and better strategies in manual control. Children can also gain familiarity if they use this application; the fancy visualization will increase their interest in astronomy or computer science.

The mockup graphical user interface is shown in Figure 1, which is made of the simulation visualization panel (Left), status panel (Top right) and manual control panel (bottom right).

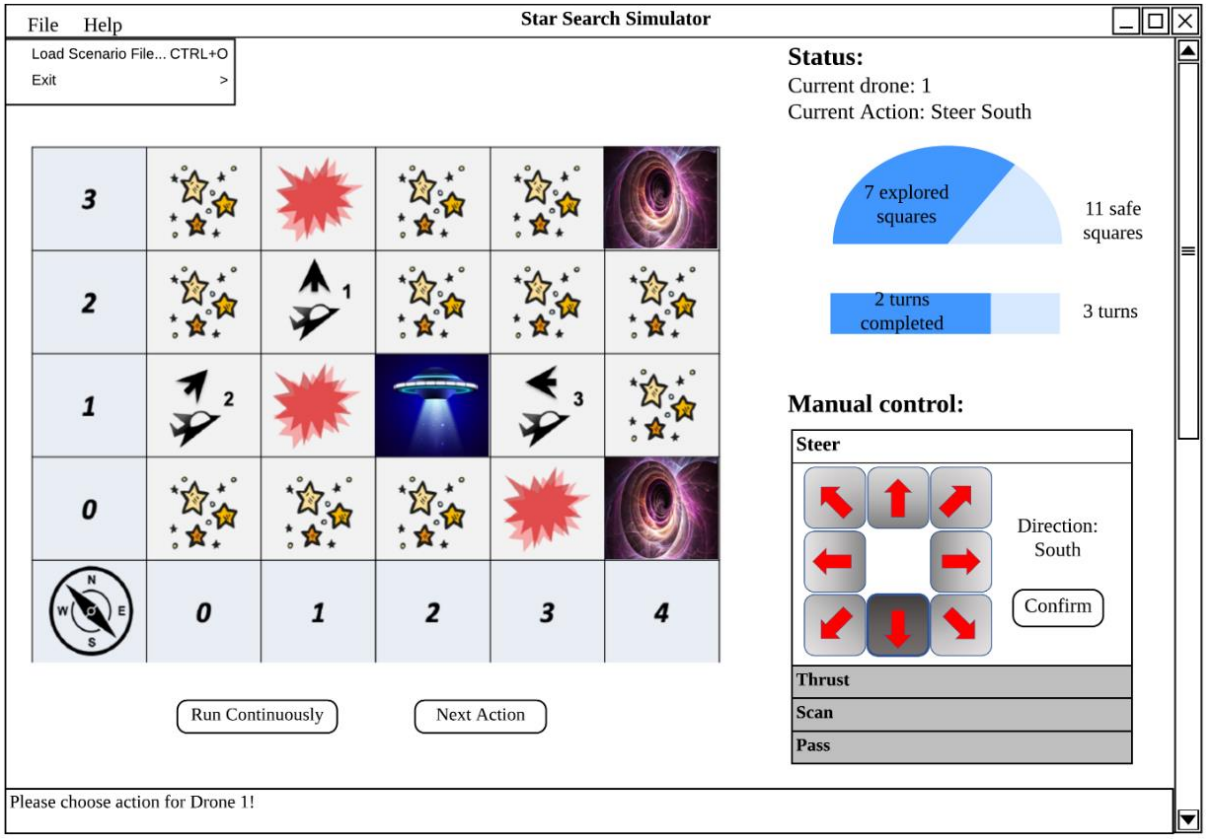

Figure 1. Mockup GUI

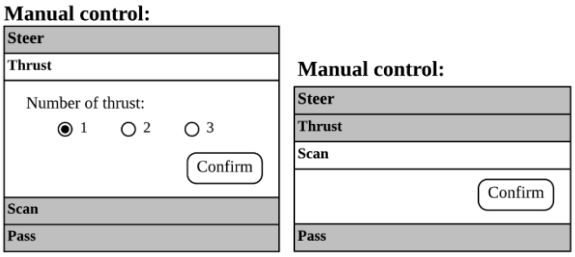

Figure 2. Manual control panel when user chooses

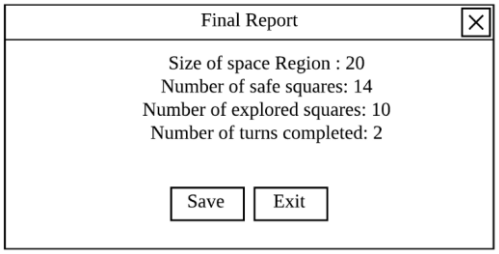

Figure 3 Final report popup window

The flow of operations is as follows:

- First, user selects “Load Scenario File”, which will initialize the space region and objects in it.

- Then, user can either click “Run Continuously” to let the software run all the actions continuously or click “Next Action” button to step through the simulation run one action at a time.
- For each action, the simulation visualization panel will update the position/direction of the drones and UFO and the status panel will update the simulation statistics as well. The simulation statistics include the current drone ID, the current action. If the current action is scan, the scan result will also be shown here. Also, a progress half pie shows how many squares have been explored from the safe squares and a progress bar shows how many turns have been completed from the total number of turns.
- If in the scenario file a particular drone uses manual strategy, then the status bar at the bottom of the window will alert the user to do manual control by choosing an action from the manual control panel and input the action parameters. (Figure 1 shows steer control, Figure 2 shows thrust and scan control). This panel will only be enabled when user input is required in order to avoid interference with any other actions the program may be executing concurrently.
- After the simulation is done, a popup window (Figure 3) will show up with the final report information. If user chooses “save”, another window will pop up asking for saving directory information. If user chooses “exit”, then final report will not be saved.

(Semi-)Realistic Modifications to the Space Region (Wormholes/Warp gates)

We are introducing wormholes to the space region. If a drone or UFO (our third modification) flies into a wormhole, it will be teleported to another region of space. Our new approach of adding wormholes will break the linearity we have right now with drone movements. For example, if a drone is one square away from Gate A, and then enters Gate A by thrusting 3 squares ahead, then it will come out of the corresponding Gate B and finish the last two squares of the thrust. Since drones are allowed to perform multiple square thrusts through the warp gates, their direction of entry will be preserved on exit from the corresponding gate. The wormholes will work in pairs and be bidirectional, meaning drones can enter through both ends of a wormhole (Gate A will lead to Gate B, and Gate B will lead back to Gate A).

Users will have the flexibility to specify the number of warp gate pairs and their initial locations via the scenario file. Doing so will be similar to how suns are currently configured. Somewhere on the scenario file, the first line of the wormhole description must start with "w<X>", where "w" indicates wormholes, and <X> indicates the number of warp gate pairs to be generated. Then below this line, for each wormhole, set the coordinates for each pair of warp gates in the following format: "(x1,y1)(x2,y2)", where (x1,y1) indicates the position of Gate A, and (x2,y2) indicates the position for Gate B. If the wormhole config is not specified, the simulation will randomly generate them.

There is no support for wormholes in the current implementation. The current practice is limited by linear and predictable movements as well as barriers. If we are successful, the presence of these warp gates will enable drones and the UFO to move between non-adjacent squares, which can potentially allow them to explore the space region more quickly if used intelligently.

(Semi-)Realistic Additions of Other Spacecraft or Objects (UFO)

We are introducing a UFO that will fly around the space region that will destroy drones upon contact. This is a new feature that does not currently exist. In current practice, the only danger to drones are suns and other drones. The UFO will be destroyed if it comes into contact with the sun, and can also take advantage of the wormhole. For each turn, the UFO will only be allowed to thrust one space. There will only be one UFO per simulation, and its starting position can be either specified from the scenario file, or be randomized for each simulation.

Adding the UFO object will allow a more realistic simulation as a real space region will contain a dangerous object other than a sun that poses a threat to the drones. If successful, this

modification will allow users to mimic the space region in a more realistic way. We believe this will take medium level of effort as we will most likely have to create a new object for a UFO and modify existing classes to include the UFO to appear in scan and crash methods and include it in the final report logic.

Heilmeier's Catechism (common for all 3 modifications):

Who cares?

Our proposed modifications will enhance the user experience for drone testers/space explorers who use the current version of the simulation system. The GUI will give a better visual representation of the simulation that testers will appreciate, and the addition of a UFO and wormholes will give explorers a more dynamic and realistic simulation.

What are the risks and the payoffs?

Risks are:	• Coronavirus effect • Bugs (processing wrong variables, thread synchronization, etc.), wrong algorithms, poor design • Redesign needed to accommodate the new functionality may cause the current implementation to break in the short term
Payoffs:	• Unique selling points feature implementation: Wormholes and UFOs. • Have a new feature that works in a very different way from anything the implementation can handle right now • Adds a new element of risk vs reward (using a warp gate could teleport a drone near an unexpected UFO, but could also allow faster exploration)

Table 1. risks and payoffs table

How much will it cost?

Due to our student status, the development, deployment, and maintenance can be completed for free resulting in zero cost. The analysis of cost is shown in Figure 4.

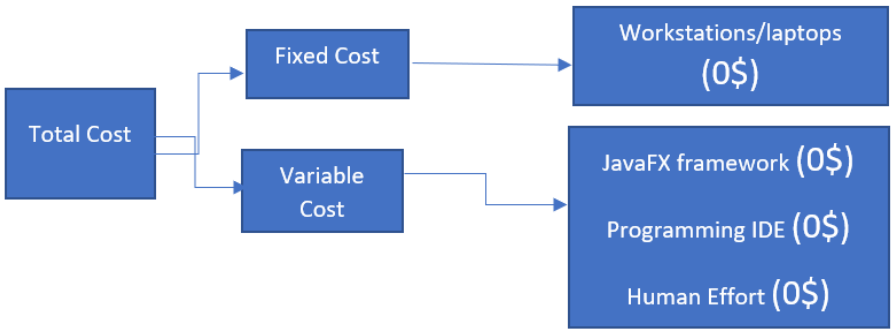

Figure 4. Cost Analysis Chart

How long will it take?

As is shown in Figure 5, this project will take 18 days (80 hrs * 5 group members = 400 work hours). We have three milestones. First milestone: generate UML diagrams such as class diagram, sequence diagram and state diagram. Second milestone: first version of the software accomplishing basic features from previous assignments. Third milestone: add the additional two features and achieve the final version of the software. Finally, we will do our final polish of the UML and software, generate the user manual and make a video presentation.

The work breakdown structure is shown in Figure 6. The estimate of the difficulty will lie in implementing the GUI because none of the group members have previous knowledge in JavaFX but two weeks for 2nd and 3rd milestone will be enough for us to learn JavaFx and use it.

Project's Gantt Chart and Work break down structure

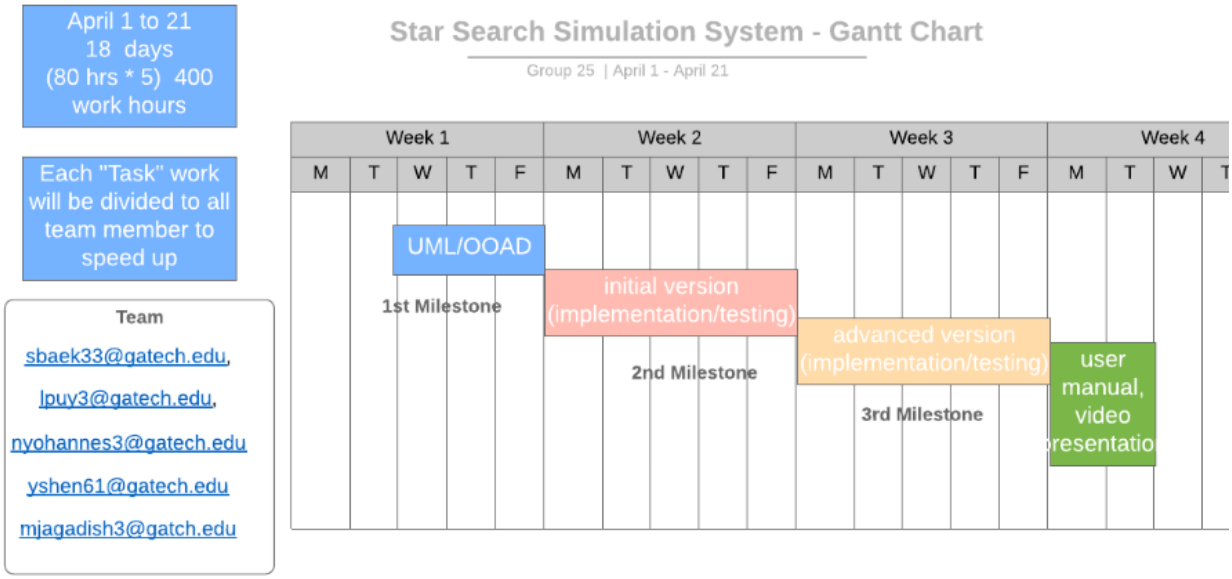

Figure 5. Timeline for the project

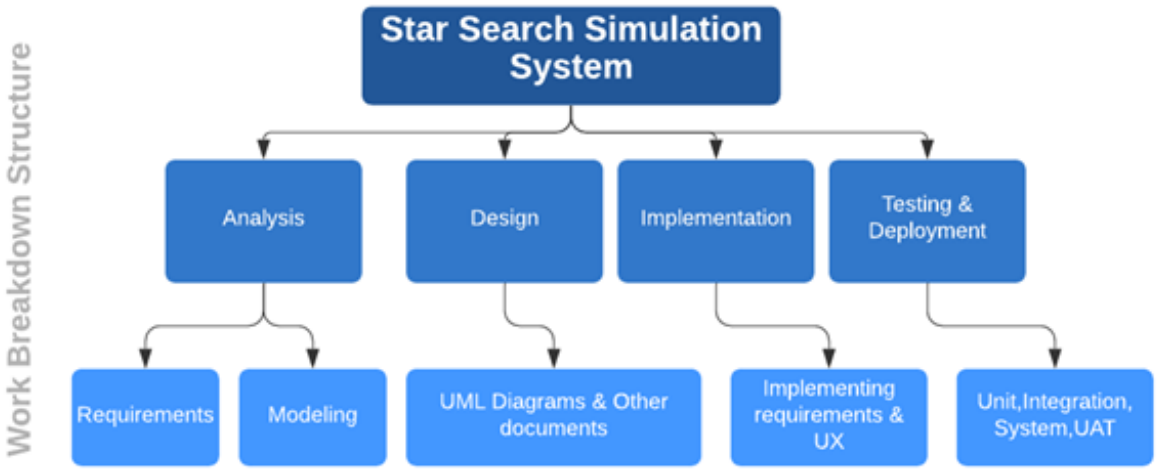

Figure 6. Work break down structure

What are the midterm and final "exams" to check for success?

The midterm corresponds to the second milestone in Figure 5. In this stage, the initial version of the software should be done according to the UML diagrams produced in the first milestone. The check for success includes the following:

1) GUI should accomplish basic tasks without issues. For example, load the scenario file, display and update the simulation visualization panel/status panel, enable users to control drones manually, and generate the final report.

2) GUI should be able to generate an alert when user accidentally pushes the wrong button instead of getting crashed.

3) Test using a variety of scenario files with different experimental settings. For example, check that for each action the software gives the correct response and the final report is accurate. Different experimental settings include varying the number of drones and suns, varying drone initial position/direction, varying grid dimension.

The final exam corresponds to the third milestone in Figure 5. In this stage, the UFO and warp gates will be added. In addition to the midterm validations, we will check the following:

1) The warp gates and the UFO should be displayed correctly in the simulation visualization panel. We will also validate their behavior. For example, verify the drone flies through wormholes properly, UFO can destroy a drone without being destroyed as well, etc.

2) User should be able to specify the UFO's starting position and the number of warp gates and their initial positions from the scenario file. If these elements are not specified, a pair of wormholes and a UFO will be created and positioned at random.

3) User should be able to control the pace of the simulation.

4) The status panel displays the correct results and the final report is correct when UFO and warp gates are taken into consideration.

5) The additional documents such as user manual and video presentation are organized and clear.

6) Finally, we'll verify how well the code reflects our initial UML diagrams.

Software Development Technical Stack:

Operating System	**Linux**
Programming Language	**Java v11**
Programming IDE	**IntelliJ Idea**
GUI framework	**JavaFX, Java 1.8**
UML IDE	**Lucidchart, UML v2.0**
Source Control	**Github (GaTech Enterprise repository)**

Table 2. Software development technical stack