Daniela Muniz
Austin Douglas Crow
Justin Phillips
Max McKinney

Project Group 22

Self-Assembling Off-World Colony

Site url
http://web.engr.oregonstate.edu/~phillij6/spaceforce/

Login = admin
Password = 1234

USER STORIES

Manual Control:
Teammates that worked on it: Max and Daniela

Relevant Unit Tests:
Input of Commands to web page
- -Input string to command AI unit
- -Display string to screen
- -Press Submit Command Button
- -Command is successfully input to queue

Display of Command to Command Queue
- -Take a command that is already input
- -Display command to screen in correct order
- -Intuitive storage process

Display of Time of Command Issuance to screen
- -Record time of command once input
- -Associate time with correct command
- -display time to screen with correct format

Main Problems:
Difficulty determining what procedure to use for storing and retrieving the information. Teammates had different ideas on how to do it based on their own experience and had to work out a way that they would both be able to be effective.

How long did it take?
The decision process on how to manage the data took about 3-4 hours. After that the programming itself only took 1-2 hours.

Current Status:
Complete

Announce Construction:
Teammates that worked on it: Max and Daniela

Relevant Unit Tests

Create Entries to Construction Log
- -Show button for construction check
- -Display modules to check
- -Create construction list once button is pressed

Display Entries to screen
- -Display construction check button
- -display module names to screen
- -display construction status once status has been created by the construction check button

Display status of entry to the screen
- -Status has been created by check status button
- -status is associated with correct module
- -status is correctly displayed to the screen

Main Problems:
Difficulty determining what procedure to use for storing and retrieving the information. Teammates had different ideas on how to do it based on their own experience and had to work out a way that they would both be able to be effective.

How long did it take?
The decision process on how to manage the data took about 3-4 hours. The programming took another 2-3 hours.

Current Status:
Complete

Warn System

Teammates that worked on it: Max and Daniela

Relevant Unit Tests

Create Entries to System Warn Log
- -Show input box for string input
- -Display button for submitting warning
- -Display modules to check
- -Create construction list once button is pressed

Display warning entries to screen
- -Store entries in retrievable format
- -Retrieve Entries in correct order and format
- -display warning entry to the screen

Display number of entries to the screen
- -Entry has been created by send warning button
- -warning is stored in warning queue
- -number of items in warning queue calculated
- -number of items in warning queue displayed to screen correctly.

Main Problems:
Difficulty determining what procedure to use for storing and retrieving the information. Teammates had different ideas on how to do it based on their own experience and had to work out a way that they would both be able to be effective.

How long did it take?
The decision process on how to manage the data took about 3-4 hours. The programming took another 2-3 hours.

Current Status:
Complete

Review of spikes and UML Sequence diagrams

Manual Control

We had a discussing last week with the customer and wanted us to create the manual control feature in week. The spike we performed was useful, we discussed it as a team and decided to create a UI that allowed the user to send commands to a queue and would be performed based on the user's preference. The UML diagram needed updated a little to accommodate these changes. Like many of the user stories we could only simulate this scenario. This story still fits with in all the planning and met the customer's expectations.

Announce Construction

This user story was easy to implement we followed the UML diagram which would create a solid set up for this story. The story had to be created after we had the manual control operational. Once completed we performed our spike for additional information. The implementation took less time than we thought but we were able to meet all the customer's expectations.

Warn System

This user story was like "announce construction" and needed the manual story completed. This story had a straightforward implantation. Where we just simulated the scenario of a warning and took the amount of time, we had originally thought it would. The stories were all integrated and the implementation met all of our planning and exceeded the customer's expectations.

Refactoring:

This week we had to do a significant amount of refactoring to our initial plan for the week. Originally we had thought that shifting to a command line application would allow for the additional functionality that we would like for the additional user stories for this week. Once we added up all of the additional work that would be required to reimplement the user stories from last week in a new environment we decided against it. Do to this planned change, and then unchange, we had to rethink how we would go about implementing our user stories in the html format. We had a group meeting at the

beginning of the week and thought about the ways that we could make the html format work with the ideas that we had for the command line app.

Manual Control:
This was originally going to be a command line interface that allowed for specific commands to be typed into it. This would then take the commands and put them into a queue for later use by the AI colony. Once we decided against the command prompt we had to find another way to accomplish this. With the online app we decided that a list of buttons could be presented and the choices could be concatenated together to form a string of commands that would then be fed to the AI colony. The commands that we decided on were different objectives or priorities for the colony as a whole. Having individual command of a unit that is located several light years away, that is autonomous, didn't seem like it was very practical or feasible.

Announce Construction Message:
When the plan was to build a command line app, this was initially going to be a timed response after selecting items from the manual command prompt. If one of the options selected was to construct something, the program would determine how long the construction would take, and display a construction complete message once it was completed. After switching to the html interface we decided to use different objects to accomplish this task. We created a construction alert that will display if construction has successfully completed once the system receives confirmation that the project is done.

Warn System Message:
When the plan was to build a command line app, this was initially going to display if a manual command was issued, but the system determined that it either was unable to perform the command, or the command was performed unsuccessfully. Once we switched to the html app we decided to handle this similarly to the announce construction message. After switching to the html app it was found that it was actually significantly easier to implement this story.

Additional Customer Questions

The customer was emailed on the 12/04/19 with a proposal of how we plan on developing the stories for this week. We suggested a different method on how we were going to develop the story instead of using a command prompt we were going to handle

via UI. Since we never heard back within the 24 hour window we just went forward with it.

Description of Integration Tests

More functionalities were implemented this week. For the data storage, we decided to use an array inside the project's javascript code. We figured that it made an appropriate placeholder for our data in the meantime while we were without a database or a JSON file. We were experiencing some difficulties with the JSON file connection to the code, that really drove us to use the previously mentioned array. Either way, the array approach still allowed us to perform concrete integration tests.

An example of how we were able to do integration tests would be like how we got the manual controls table to populate as the user added in a command that they wanted to send out to the system and rover. Basically, a user would just input whatever command they want to send into the text box on the manual control page that they navigated to from the main page. Then on the click of the send button, the inputted command would be saved into the data array that's our database substitute. Not only would the data be saved, but it will also call the method to display the sent command along with its timestamp and the amount of commands that were sent in the queue. This would let the user know that their command was received and was going to be executed in some time. The way we tested this part of the application was firstly having each member of the team test this flow from the main page to the manual control page. Then each of us also input multiple commands and checked if they all displayed correctly with different timestamps for each command. Afterward this testing, we were pleased to see that the flow went as expected and the functionalities did what they were supposed to do.

Having the table available for was important to us, because we wanted to show the user that they were actually interacting with something outside of the application and that their communications were being received. This was especially requested by our customer. In our email discussion with him about this week's user stories to complete, he expressed a lot of interest in showing the application talking to external entities. It is obvious here that if we accomplished having the application display its connectivity with external entities, then we have accomplished integration with the back end.

Another user story our customer requested for was the one about the construction announcement. Specifically, he wanted a system check to be ran. Then a construction announcement would be made if the construction process was ready to start. In our implementation, we display the different modules that are involved in the system check.

When the user clicks on the construction check button, each of the modules tell us its status. The different statuses it gives us is “Building...”, “Complete!”, and “Idle”. When the module shows “Complete!”, then construction is ready to start. However, what we noticed after the implementation was that we did not make that very clear to the user. At least what we achieved here was presenting the application’s ability to check external modules from the back end. Additionally, the flow from the main page to the construction announcement page also went as expected. This again demonstrated evident back end integration with the front end of the program.

Finally, there was the integration involved with the warn system that our customer also requested from us. He basically wanted to see that the user was able to communicate with the system and rovers whenever they wanted to send out a warning to either things. This part of the program actually works similarly to the manual control section that we’ve discussed earlier. Again, there is a text box that a user uses to input the kind of message, in this case a warning, they want to send to whatever they want to warn. There’s also a queue as well, and a table that shows what warnings have been sent out. The team practically tested this part of the application the same way our manual control page was tested. Going from the main page to the warn system page flowed as expected and the functionalities were also working the way we wanted them to. The testing also included inputting many different sorts of warnings.

Overall, we found the testing for this week to be very useful. We all agreed that executing these sorts of test proved to be time efficient, and it most likely would have been rather time consuming if we did not do any tests whatsoever as we kept adding on more components, features and functionality to the application. More testing will definitely be applied as we work on the next user stories.

Implementation Plan for Week 8

User Stories

1. Map
2. Satellite Image
3. Location Data

Schedule:

Monday	Austin and Justin
~12 hours	Create program that gathers data from the planet and create a map and create mock data to test it.
Tuesday - Wednesday	Austin and Justin
~2 hours	Write code to run a system check to see if the if the colony location is reporting back properly.
Tuesday - Thursday	Maxwell and Daniela
~10 hours	Create a page to allow the user to view a satellite image. Write code to have the system move accordingly depending on the user's request.
~5 hours	Create a method for the user to be able to select a safe location for the system so that it can relocate.

Customer

Our customer was able to communicate with us via email and was also quick to respond when we had more questions.

Team Member Contributions

Austin: User stories formatted and unit tests created. Followed through on unit tests and verified results.

Daniel: Description of Integration Tests, pair programmed with Max manual control, warning system and construction complete
Justin: Review of spikes and UML Sequence diagrams, customer and Additional Customer Questions, implementation plan
Max: Description of Refactoring, pair programmed with Daniela manual control, warning system and construction complete