Go-Gitters Deliverable 2: Celestia

Test plan for Celestia:

By: Lara Brooksbank, Jacob Mattox, Kyle Cooper, and Alexander Swanson

Testing Process:

Celestia is primarily a graphical application that creates a replica of our universe. Our testing process will involve interacting with the GUI via Python and Bash scripting. The Python library PyAutoGui allows for the system to take control of the keyboard and mouse functions and use them to navigate the screen using pixels for reference. We are leveraging this library to test what Celestia displays after certain navigational functions and scripts are run.

Requirements Traceability:

Celestia is an open-source project that was not designed by our group and therefore we do not have direct access to the requirements of its design. However, we can infer some requirements based on the existing product. It would appear that a requirement is for the user to be able to interact with the universe in a tangible way. Another requirement could be to help the user learn more about our solar system. Since these are the requirements we can infer based on our use of system, these are the types of things we will be testing.

Tested Items:

We are testing the installation process which, to a new user, could seem complex and create a barrier to entry for using the system. Additional tests will be of existing functions within the Celestia packages, as well as, the GUI.

Testing Schedule:

10/15: Deliverable 2 - 5 test cases created
11/12: Deliverable 3 - Initial test framework with 5 tests created and demonstrated
11/19: Deliverable 4 - Test framework with 25 tests finalized and demonstrated
11/28: Deliverable 5 - 5 faults injected into codebase to create failed test cases, demonstrated
11/30: Project completed and demonstrated again, with final thoughts on complete process

Test Recording Procedure:

Our test cases are documented in a test log text file that lists the date and time of the test and the output, whether a success or failure, in a readable format. Due to the inherent nature of our program (being mostly graphical), a large portion of our test cases will only return “Success” if the test script has been executed completely. Any failure would most likely cause a failure to the entire program and would require a restart.

Hardware and Software Requirements:

Operating System: All code was tested in Linux using the Ubuntu 18.04 distribution

Hardware: All tests performed on a 13” MacBook Pro within a VirtualBox virtual environment. Graphical tests use the MacBook Pro’s native resolution for pixel density to find the specified location on the screen.

Software: VirtualBox, Python, Bash, Ubuntu 16.04, PyAutoGui(python library)

Constraints:

Time is a major constraint on this project. We only have one semester and each of us have other classes and assignments to complete during the semester. Total number of people is also a concern. There are four people working on the project, but as with any testing, the more people the better for creating useful test cases.

System Tests:

5 Test Cases

Test 1: Installation - When initially building Celestia in Ubuntu 18.04, test for the necessary libraries required to run ‘./configure --with-gtk’.

Test 2: Installation - When initially building Celestia in Ubuntu 18.04, test for the appearance of the ‘Make’ file after running the ./Configure command to compile from Celestia source code.

Test 3: Installation - After running ‘sudo make install’, test for the appearance of the celestia.exe file.

Test 4: Test that Celestia graphical interface properly fixes onto the sun using hotkeys.

Test 5: Test that Celestia graphical interface properly zooms out to show the Milky Way galaxy.

Report:

Thus far, getting Celestia to compile in our virtual machines has been the most difficult and time consuming part of the project. In the Celestia Github and Celestia Forums there is no support for compiling Celestia from the source code in Linux/Ubuntu 18.04. We decided as a team that this would be a worthwhile place to begin before diving into the rest of Celestia itself. Our first few test cases revolve around compiling and installing Celestia from the official source code. These tests include checking for the necessary libraries, the location and appearance of the ‘Make’ file, and also the location and appearance of the celestia.exe after compiling.