Team International
Report: Deliverable #4
Project: Celestia-g2
Date: 11.14.2016
Members: Gui Costa, Megan Landau, Tony Tang

Task: Complete the design and implementation of your testing framework as specified in **Deliverable #3**. You are now ready to test your project in earnest. You will create 25 test cases that your framework will automatically use to test your H/FOSS project.

Team Experience and Process:

1. **More test cases:**
    - For this deliverable, we added more test cases to the initial cases we had from **Deliverable#3** for a total of 25 test cases. Initially, we had already written 12 test cases for Deliverable #3. We began by adding 13 more test cases for **Deliverable #4** and checking to see if our test cases really tested a good variety of inputs. We added 3 more test cases for the **square** method which is found in the **mathlib.h** library of the Celestia repository. Then we added 2 more test cases for the **cube** method. We added 2 more test cases for the **sphere area** method and 3 more test cases for the **circle area** method.
2. **Finding the last method:**
    - Lastly, we had to find one more method and create 5 test cases for a complete total of 25 test cases for the entire project.
        - Initially, we wanted to test the method, **readChar**, which is found in the **3dsread.cpp** file. This method is supposed to read an input stream and convert it to a char correctly. The readChar method has inputs that are of variable type char. The return type for the readChar method is a boolean. Unfortunately, the file 3dsread.cpp had too many dependencies and we were unable to create a test driver that could successfully perform the readChar method.
    - After failing to use the readChar method, we decided to look into libraries other than the mathlib.h library to find a method that was suitable for testing. First, we found a method that used a timer. We ended up finding a method in the **astro.h** library called **secsToDays**, this method would convert time in seconds to a number of days. The input variable is a double. However, this method also had too many dependencies and we were unable to run the test driver to display the correct results. In the end, we decided to ask for permission from our professor if we could use another method found in the mathlib.h library because we knew what most of the expected outcomes would be and were able to successfully calculate them.
        - We finally settled on a method called **radToDeg** found in the mathlib.h library which converts radians to degrees based on a radian input. This method worked similarly to other methods we have been using. We decided to test

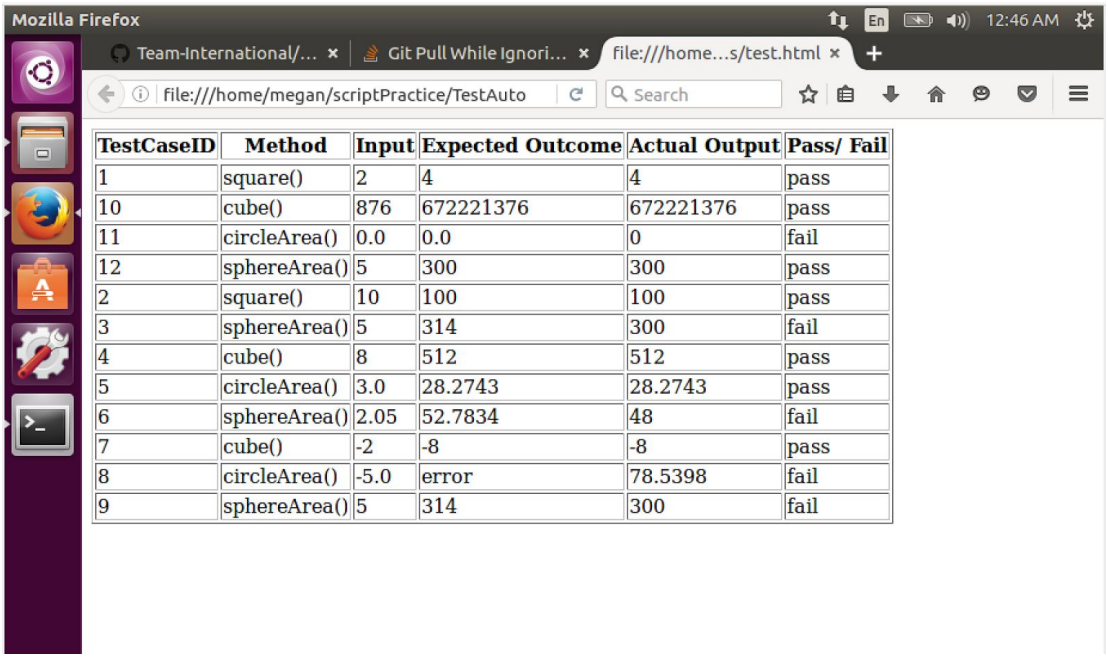
the radToDeg method with input variables of 0, 5, 1, -57, and 125,000,000 to test a wide variety of possibilities.

3. **Editing the HTML page with CSS:**
    - Next, we decided to add to the aesthetics of our HTML table. Below are a "Before" and "After" screenshot of the table (See *Figure 1* and *Figure 2*).
    - First, we wanted to display our team name and logo above the results table because we thought this would make our table look more professional and cohesive with our Wiki page.
    - Next, we changed the font for the data in the table to Helvetica because it adds a simple, clean look to the page.
        - We wanted to style the HTML table for it to be easier to read so we decided to alternate the row colors between the rows between gray and white. We made the even rows to be #EAEAEA which is a grayish color.
        - Then, we wanted to write a little description of what the table results show. For example, the table shows the number of the test cases, the name of the method, the expected outcomes, the actual outcome, and whether the test case pass or failed.
        - We also added the name of the group members that worked on the project and the original project.
        - Next, for additional styling we added a page border in gray. We plan to add links at the bottom of the HTML table to our GitHub account and blogs.
4. **Analyzing our test results:**
    - Lastly, we started looking at the results from the tests. Some of our tests cases were failing due to rounding errors or problems with how the math library defines the number Pi.
    - We decided to make a note of these specifications on the HTML page because it is important to note what restrictions the Celestia project places on mathematical formulas.
    - We also wanted to make sure all decimals were rounded to four places, which is how the Celestia math library uses decimal point precision.
    - Another problem we had was when dealing with the exponential notation. We did not know what was the minimum value that a integer value will be converted into exponential notation, but we think it is to the 10^5th power.

Problems and Issues:

- Our script running the drivers had to be executed from inside the /scripts directory. We had to change it to meet the project requirements: the project requires the script to be executed from the /TestAutomation directory using the command ./scripts/runAllTests.sh. It was a learning experience learning how to change directories using Bash, but at this point we are very comfortable with Bash. The team worked together to overcome this problem and we were able to provide a solution to make the script work as required.

- The math library was also generating answers with only 4 decimals. We had to run tests and we determined that, by default, the precision of the custom math library used by celestia was only to 4 decimal places.
- The team spent a lot of time on the HTML report. We felt that it was an important characteristic of the project to provide an exceptional report. We had major difficulties customizing the HTML, since for some of the team members it was their first experience with HTML. We learned how to properly use tags and how to format text to provide a nice visual effect on the report. The HTML report reflects the hard work and determination of the team as we made sure that it is of good quality and easy to understand. Adding a picture to the HTML report proved to be a challenge since we had to make changes to the script itself to apply the picture on the report. After many different combinations of table styles, font sizes, and colors we all agreed that the final HTML report was perfect.
- The team is running into problems adding a pdf file to the GitHub wiki main page. We have converted Deliverables #1-3 into Markdown type but we want to add the actual pdf files to the main wiki page.



TestCaseID	Method	Input	Expected Outcome	Actual Output	Pass/ Fail
1	square()	2	4	4	pass
10	cube()	876	672221376	672221376	pass
11	circleArea()	0.0	0.0	0	fail
12	sphereArea()	5	300	300	pass
2	square()	10	100	100	pass
3	sphereArea()	5	314	300	fail
4	cube()	8	512	512	pass
5	circleArea()	3.0	28.2743	28.2743	pass
6	sphereArea()	2.05	52.7834	48	fail
7	cube()	-2	-8	-8	pass
8	circleArea()	-5.0	error	78.5398	fail
9	sphereArea()	5	314	300	fail

Figure 1. Before screenshot of HTML report presented on deliverable #3.

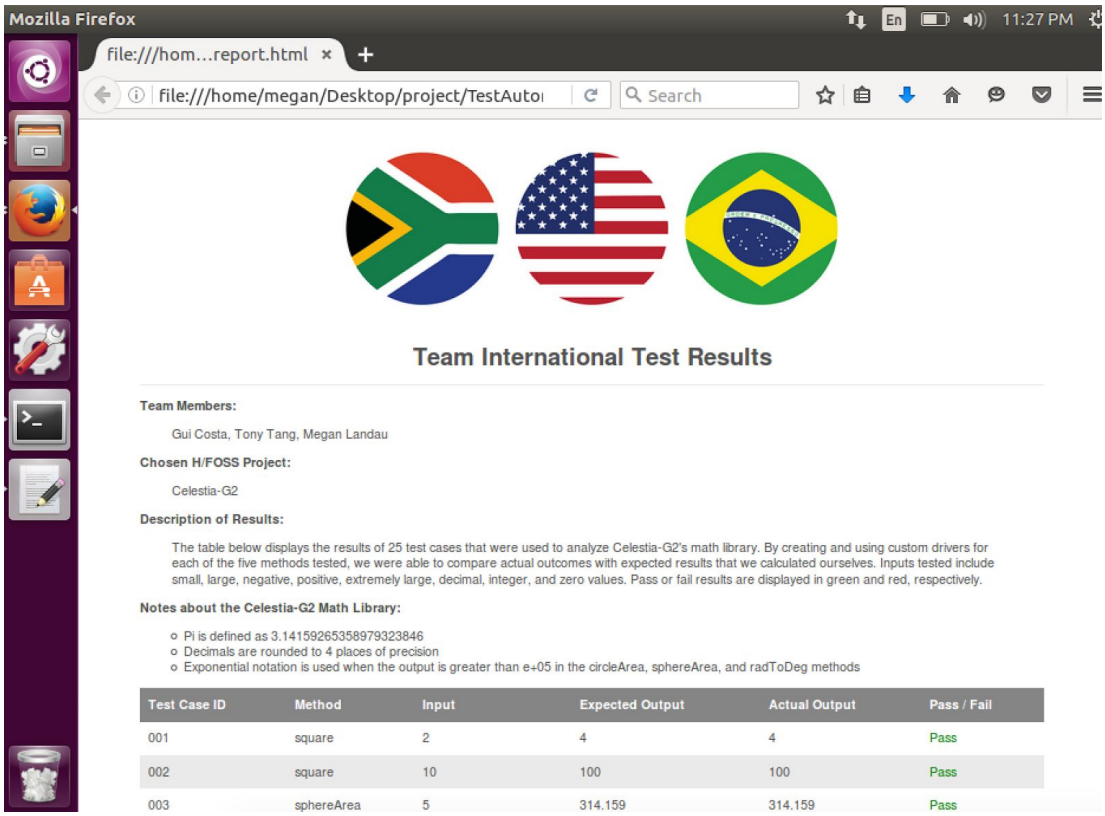


Figure 2. After screenshot of HTML report after it has been altered for deliverable #4.