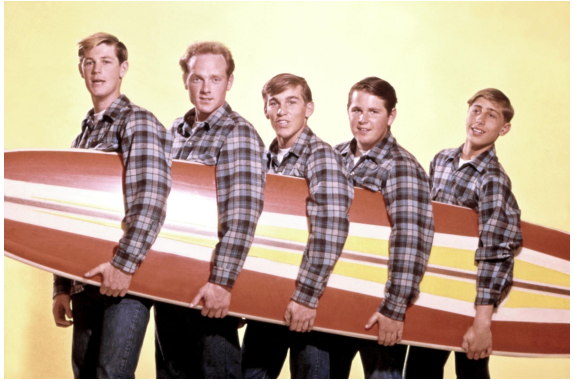
tanaguru

Tanaguru Contrast Finder

Brought to you
by
The Beach Boys



Why We're Here

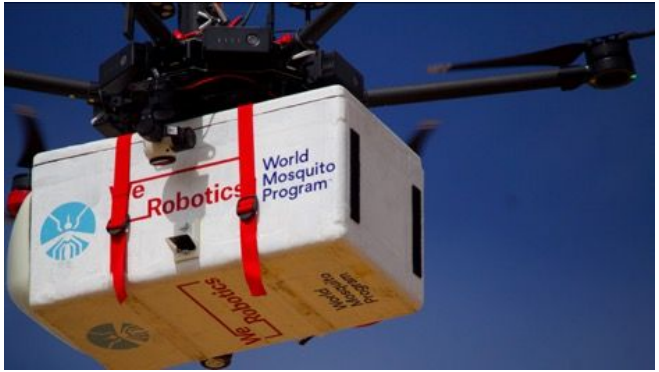
Our Goal

Our goal is to design and build an automated testing framework to run a series of test cases on a software project we have chosen as a group.

Our Candidates

1. Drone 4 Dengue
2. Epidemiological Modeler (STEM)
3. Tanaguru Contrast Finder

Drone 4 Dengue



Overview

- A software project that uses drone images to detect dengue mosquito breeding sites
- These mosquitoes can transmit a disease called Dengue Fever
- This disease is primarily transmitted in subtropical areas of the world

Running and compiling

- Due to the lack of updates to this project over the years we were not able to run it due to compatibility issues, causing us not to go with this project

Epidemiological Modeler (STEM)

Overview

- The Epidemiological Modeler (STEM) is a software tool that Was developed to help scientists and public health officials create models and run simulations of emerging infectious diseases. These simulations can help understand and prevent diseases before they could possibly become a worldwide pandemic. We found this project interesting because we thought it related to what's going on in the world today.

Running and Compiling

- Requires Eclipse IDE and numerous Eclipse plugins from the Eclipse Marketplace, along with one additional library
- The project successfully ran and compiled after resolving plugin issues

Tanaguru Contrast Finder

Overview

- Tanaguru is a project dedicated to enhancing the quality and accessibility of web and digital products.
- The Tanaguru Contrast Finder is a tool developed by the Tanaguru team and is meant to help find contrasting colors that promote design quality and accessibility for visually impaired users.

Running and Compiling

- The TCF is coded mainly in Java, a language each of our team members are familiar with, and comes with a multitude of methods that can be properly tested.

Why We Chose Tanaguru

Overview

- Tanaguru had the best documentation out of all of the projects
- The project is written in Java, a language everyone in group felt comfortable with.
- We managed to successfully build it and it was simpler to use than the other projects

Hardware and Software Requirements

- Linux System or VirtualBox 6.1 - https://www.virtualbox.org/
- Git - https://git-scm.com/downloads
- OpenJDK 14.01 - https://jdk.java.net/14/
- Tanaguru - https://github.com/Tanaguru/Contrast-Finder

Finding Test Subjects

- Looking into Tanaguru’s files we found a ColorConverter.java file that contained methods that we would test
- ColorConverter.java
    1. rgb2Hex()
        - Given an rgb value will output the hex equivalent
    2. hex2Rgb()
        - Given a hex value will output a Java Color object that matches this hex value
    3. getSaturation()
        - Given an rgb value will output the saturation of that color
    4. getHue()
        - Given an rgb value will output the hue of that color
    5. getBrightness()
        - Given an rgb value will output the brightness of that color

Our Plan

Testing Process

- To test these methods, we constructed driver classes for each method being tested. Then, by feeding information into the drivers through a script that reads through our test case files, our script compares the driver output to the expected output supplied in each test case. All results are then placed into a table and presented as an HTML file in your system's default browser.

Requirement Traceability

- getBrightness requires a Color class as input and returns a float value. Test Cases 1-5
- getSaturation requires a Color class as input and returns a float value. Test Cases 6-10
- getHue requires a Color class as input and returns a float value. Test Cases 11-15
- rgb2Hex requires a Color class as input and returns a string. Test Cases 16-20
- hex2Rgb requires a hex string of a color as input and returns a string. Test Cases 21-25

Testing Framework

- Compile all test case drivers
- Creates an empty html file
- Script runs through each test case
  - Passes information to correct driver
- Driver outputs processed information to the html file
- Results are compared with their expected values
- Passing or Failing results is written to the html file
- Html page is opened to display the results

```
javac testCaseExecutables/*.java
echo "All drivers have been successfully compiled"
```

```
mkdir -p reports
touch reports/results.html
> reports/results.html
```

```
declare i=1
declare firstLine
declare numberOfFiles=$( ls testCases -1 | wc -l )
numberOfFiles=$((numberOfFiles+1))

while [ $i -lt $numberOfFiles ]
do
  for file in testCases/*.txt
  do
    firstLine=$( head -1 $file )
    if [[ $firstLine == $i ]]
    then
      echo $file >> temp/fileOrder.txt
      break
    fi
  done
  i=$((i+1))
done
```

```
if [[ $output == $expected ]]
then
  echo \<td\>"Passed"\<\/td\> >> reports/results.html
else
  echo \<td\>"Failed"\<\/td\> >> reports/results.html
fi
```

```
xdg-open  reports/results.html
```

Example Test Case and Results Table

6 lines (6 sloc) | 73 Bytes

```
6
outputs brightness of color
ColorConverter
getBrightness
0x00FF00
1.00
```

Activities | Firefox Web Browser | Oct 22 16:50

Mozilla Firefox

csci-362-02-2020/Beach | /home/amonspm/Documen

file:///home/amonspm/Documents/Beach-Boys-master/TestAutomation/reports/results.html

Test Number	Requirement	Component	Method	Input	Expected Output	Output	Passed?
6	outputs brightness of color	ColorConverter	getBrightnessDriver	0x00FF00	1.00	1.00	Passed
7	outputs brightness of color	ColorConverter	getBrightnessDriver	0xFF0000	1.00	1.00	Passed
8	outputs brightness of color	ColorConverter	getBrightnessDriver	0x0000FF	1.00	1.00	Passed
9	outputs brightness of color	ColorConverter	getBrightnessDriver	0x000000	0.00	0.00	Passed
10	outputs brightness of color	ColorConverter	getBrightnessDriver	0x800080	0.50	0.50	Passed
16	outputs hue of color	ColorConverter	getHueDriver	0xFF0000	0.00	0.00	Passed
17	outputs hue of color	ColorConverter	getHueDriver	0x00FF00	0.33	0.33	Passed
18	outputs hue of color	ColorConverter	getHueDriver	0x0000FF	0.67	0.67	Passed
19	outputs hue of color	ColorConverter	getHueDriver	0x000000	0.00	0.00	Passed
20	outputs hue of color	ColorConverter	getHueDriver	0x800080	0.83	0.83	Passed
11	outputs saturation of color	ColorConverter	getSaturationDriver	0x000000	0.00	0.00	Passed
12	outputs saturation of color	ColorConverter	getSaturationDriver	0xFFFFFF	0.00	0.00	Passed
13	outputs saturation of color	ColorConverter	getSaturationDriver	0xFF0000	1.00	1.00	Passed
14	outputs saturation of color	ColorConverter	getSaturationDriver	0xFFFF00	1.00	1.00	Passed
15	outputs saturation of color	ColorConverter	getSaturationDriver	0x800080	1.00	1.00	Passed
1	Converts hex value to rgb	ColorConverter	hex2rgbDriver	FF0000	Red Value: 255; Green Value: 0; Blue Value: 0	Red Value: 255; Green Value: 0; Blue Value: 0	Passed
2	Converts hex value to rgb	ColorConverter	hex2rgbDriver	00FF00	Red Value: 0; Green Value: 255; Blue Value: 0	Red Value: 0; Green Value: 255; Blue Value: 0	Passed
3	Converts hex value to rgb	ColorConverter	hex2rgbDriver	0000FF	Red Value: 0; Green Value: 0; Blue Value: 255	Red Value: 0; Green Value: 0; Blue Value: 255	Passed
4	Converts hex value to rgb	ColorConverter	hex2rgbDriver	0x000000	Red Value: 0; Green Value: 0; Blue Value: 0	null	Failed
5	Converts hex value to rgb	ColorConverter	hex2rgbDriver	800080	Red Value: 128; Green Value: 0; Blue Value: 128	Red Value: 128; Green Value: 0; Blue Value: 128	Passed
21	outputs the hex from color given	ColorConverter	rgb2hexDriver	0x0000FF	#0000FF	#0000FF	Passed
22	outputs the hex from color given	ColorConverter	rgb2hexDriver	0x00FF00	#00FF00	#00FF00	Passed
23	outputs the hex from color given	ColorConverter	rgb2hexDriver	0xFF0000	#FF0000	#FF0000	Passed
24	outputs the hex from color given	ColorConverter	rgb2hexDriver	0x000000	#000000	#000000	Passed
25	outputs the hex from color given	ColorConverter	rgb2hexDriver	0x800080	#800080	#800080	Passed

Feedback Changes

After our initial presentation of our project, we were given some feedback on changes to make to our script in order to better meet the requirements of the project. These changes were in regards to our script's knowledge of the driver names and included tweaking a line in our test cases that held the method's driver name, as well as removing an if-else statement in the script that determined the driver to be used. This if-else statement was condensed to a single line that now allows for new drivers and test cases to be implemented without the need to alter the script.

```
output=$(java testCaseExecutables.$method$driver "$input")
```

```
hue = hsbValues[HUE];
//hue = hsbValues[1];
```

```
Color.RGBtoHSB(color.getRed(), color.getGreen(), color.getBlue(), hsbValues);
//Color.RGBtoHSB(color.getRed(), color.getGreen(), color.getGreen(), hsbValues);
```

Fault Injection

As part of developing our testing framework, our team injected five faults throughout the ColorConverter.java class in order to test if our framework was properly comparing expected and actual outputs. These faults included:

- Changing the RGB values passed into the getHue() and getBrightness() methods
- Indexing the wrong HSB value in getHue()
- Editing an if-else statement in hex2Rgb() to no longer check for a preceding “#” in hexadecimal inputs
- Checking for an incorrect input length in hex2Rgb()

```
if (str.matches(HEXADECIMAL_DICTIONNARY)
        && str.length() == RGB_HEXA_LENGTH) {
        //&& str.length() == 5) {
```

```
Color.RGBtoHSB(color.getRed(), color.getGreen(), color.getBlue(), hsbValues);
//Color.RGBtoHSB(color.getRed(), color.getGreen(), color.getGreen(), hsbValues);
```

```
//Remove the two // before each block comment indicator in the above portion of the if statement and delete the "else" from
//to introduce a fault where the method no longer accepts inputs containing a # before the hexstring
else if (colorStr.matches(HEXADECIMAL_DICTIONNARY)) {
    if (colorStr.length() == RGB_HEXA_LENGTH) {
        return getNewColor(colorStr);
    } else if (colorStr.length() == RGB_SHORT_HEXA_LENGTH) {
        return getNewColorShortHexa(colorStr);
    }
```

What We Learned

- Github
- Bash
- Working with open source
- Working within a VM
- Basics of Testing
  - Drivers
  - Test Cases
  - Script
- Teamwork