CSCI 362 Software Engineering

Testing Framework of the Tanaguru Contrast Finder

Team Gr8 - Ethan Graham, Daniel McBane, Chloe Stapleton

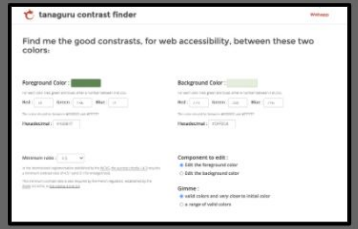
CSCI 362 Software Engineering: Testing Framework of the Tanaguru Contrast Finder

Ethan Graham, Daniel McBane, Chloe Stapleton

Tanaguru

The Tanaguru Contrast Finder is a web-based tool for finding suitable foreground and background colors for ease of use. The Contrast Finder is particularly useful in helping visually impaired readers and complies with the Web Content Accessibility Guidelines presented by the World Wide Web Consortium (W3C).

Our testing framework aims to find bugs and confirm reliability of calculations of the Tanaguru Contrast Finder.

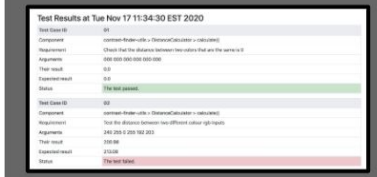


Testing Script

Our automatic testing framework consists of two files "runAllTests.sh" and "runtest.sh"

"runAllTests.sh" calls "runtest.sh" for each file in our test case directory. Before calling "runtest.sh" on each test case it creates an html file and sets up a table displays our test case and whether it passes or fails. Once its finished running "runtest.sh" for each test case it then opens the html file in the browser. Once it has successfully opened it then proceeds to delete the html file so that a new one is created each time it is run.

"runtest.sh" grabs information from the test case text file, it then updates the table in "runAllTests.sh" with the information from the test case. It then proceeds to find and compile the java files needed for the test case from the ../project/src directory where the Tanaguru file is and the ../testCaseExecutables directory where the driver file is. It then proceeds to compile the Tanaguru file in the same directory as the driver file. Once the files are compiled the script runs the driver file with the required inputs taken from the test case text file. When the file has been successfully run and has sent its output to the output.txt file it then cleans up the directories by deleting all the .class files and the output.txt file.



Test Cases

TestID	Requirement	Component	Method	Inputs	Outcomes
1	Find distance between two colors	DistanceCalculator.java	calculate(color1,color2)	000 000 000, 000 000 000	A value of 0 should be returned
2	Find distance between two colors	DistanceCalculator.java	calculate(color1,color2)	240 255 000, 255 192 203	A value of 213.08 should be returned
3	Find distance between two colors	DistanceCalculator.java	calculate(color1,color2)	180 180 000, 180 180 060	A value of 60 should be returned
4	Find distance between two colors	DistanceCalculator.java	calculate(color1,color2)	255 255 255, 000 000 000	A value of 441.67 should be returned
5	Find Euclidean distance between two colors	ContrastChecker.java	distanceColor(color1,color2)	000 000 000, 000 000 000	A value of 0 should be returned
6	Find Euclidean distance between two colors	ContrastChecker.java	distanceColor(color1,color2)	240 255 000, 255 192 203	A value of 213.0 should be returned
7	Find Euclidean distance between two colors	ContrastChecker.java	distanceColor(color1,color2)	180 180 000, 180 180 060	A value of 60 should be returned
8	Find Euclidean distance between two colors	ContrastChecker.java	distanceColor(color1,color2)	255 255 255, 000 000 000	A value of 441.67 should be returned
9	Check method gives an Error from invalid input	DistanceCalculator.java	calculate(color1,colour2)	000 000 000, GGG GGG GGG	An Error should be returned
10	Check method gives Error from invalid input	DistanceCalculator.java	calculate(color1,color2)	000 000 000, -111 111 111	An Error should be returned
11	Convert RGB value to Hex value	ColorConverter.java	rgb2Hex(R/G/B Value)	128 000 128	A value of #800080 should be returned
12	Convert RGB value to Hex value	ColorConverter.java	rgb2Hex(R/G/B Value)	128 000 032	A value of #800020
13	Convert RGB value to Hex value	ColorConverter.java	rgb2Hex(R/G/B Value)	015 082 186	A value of #0F52BA should be returned
14	Convert Hex value to RGB value	ColorConverter.java	hex2Rgb(Hex Value)	#800080	java.awt.Color[r=128,g=0,b =128] should be returned
15	Convert Hex value to RGB value	ColorConverter.java	hex2Rgb(Hex Value)	#008000	java.awt.Color[r=0,g=128,b=0] should be returned
16	Check the methods response to non-numeric input	ColorConverter.java	hex2Rgb(Hex Value)	#00G000	An Error is expected
17	Check methods response to non-numeric input	ColorConverter.java	hex2Rgb(Hex Value)	128 000 FFF	An Error is expected
18	Check methods response to negative input	ColorConverter.java	hex2Rgb(Hex Value)	-128 000 000	An Error is expected
19	Check methods response to invalid input	ColorConverter.java	hex2Rgb(Hex Value)	"apple"	An Error is expected
20	Check methods response to invalid input	ColorConverter.java	rgb2Hex(R/G/B Value)	"apple"	An Error is expected
21	Calculate RGB number with offset	ColorConverter.java	offsetRgb(RGB1,RGB2)	200 200 200, 0 0 0	java.awt.Color[r=200,g=200,b=200] should be returned
22	Calculate RGB number with offset	ColorConverter.java	offsetRgb(RGB1,RGB2)	155 211 067, 6 8 25	java.awt.Color[r=161,g=219,b=32] should be returned
23	Calculate RGB number with offset	ColorConverter.java	offsetRgb(RGB1,RGB2)	155 211 067, -60 -8 -2	java.awt.Color[r=95,g=203,b=5] should be returned
24	Offset rgb number	ColorConverter.java	offsetRgb(RGB1,RGB2)	250 190 65, 6 0 0	An Error is expected
25	Offset rgb number	ColorConverter.java	offsetRgb(RGB1,RGB2)	250 190 65, 8 0 0	An Error is expected

Injecting Faults

To make sure our tests are properly working, we inject strategic faults and analyze the results of our automated testing framework.

Fault 1:
In the ColorConverter class we inserted a fault into the offsetRgbColor() method on line 118.
Fault injected: Subtract the blue offset rather than adding it
Fault 2:
In the ContrastChecker class we inserted a fault into the distanceColor() method on line 66.
Fault injected: Incorrectly multiply red values instead of subtracting them to find the Euclidean distance
Fault 3:
In the ColorConverter class we inserted a fault into the rgbToHex() method on line 189.
Fault injected: Add an extra 0 into the formatter
Fault 4:
In the ColorConverter class we inserted a fault into the getNewColor() method on line 166.
Fault injected: Parse the Hex color string incorrectly from R to B instead of R to G
Fault 5:
In the DistanceCalculator class we inserted a fault into the calculate method on line 30.
Fault injected: Set CUBIC variable to 0 instead of 3

With all five faults in code simultaneously, 10/25 test cases will pass.

Lessons Learned

- Bash scripting / parsing text files
- The purpose of a driver
- Github / Git version control
- Markdown
- Java compiling via the command line
- VSCode
- The importance of testing!

Acknowledgements

A special thanks to Dr. Bowring for his guidance and feedback as well as the Tanaguru project and its open source status for the code (found at https://github.com/Tanaguru/Contrast-Finder).

Tanaguru

The Tanaguru Contrast Finder is a web-based tool for finding suitable foreground and background colors for ease of use. The Contrast Finder is particularly useful in helping visually impaired readers and complies with the Web Content Accessibility Guidelines presented by the World Wide Web Consortium (W3C).

Our testing framework aims to find bugs and confirm reliability of calculations of the Tanaguru Contrast Finder.

Deciding on Tanaguru

- Java compatibility
- Accessibility interest

Issues at the start

- Invalid Directories
- Incorrect Paths
- Old Wiki

tanaguru contrast finder

Webapp

Find me the good constrasts, for web accessibility, between these two colors:

Foreground Color :

For each color (red, green and blue), enter a number between 0 et 255.

Red : 70 **Green :** 136 **Blue :** 71

The color should be between #000000 and #FFFFFF

Hexadecimal : #468847

Background Color :

For each color (red, green and blue), enter a number between 0 et 255.

Red : 223 **Green :** 240 **Blue :** 216

The color should be between #000000 and #FFFFFF

Hexadecimal : #DFF0D8

Minimum ratio : 4.5

In the international reglementation etablished by the WCAG, the success criteria 1.4.3 requires a minimum contrast ratio of 4.5:1 (and 3:1 for enlarged text).

This minimum contrast ratio is also required by the French regulation, established by the RGAA 3.0 2016, in the criteria 3.3 et 3.4.

Component to edit :

- (•) Edit the foreground color
- ( ) Edit the background color

Gimme :

- (•) valid colors and *very close* to initial color
- ( ) a *range* of valid colors

Testing Script

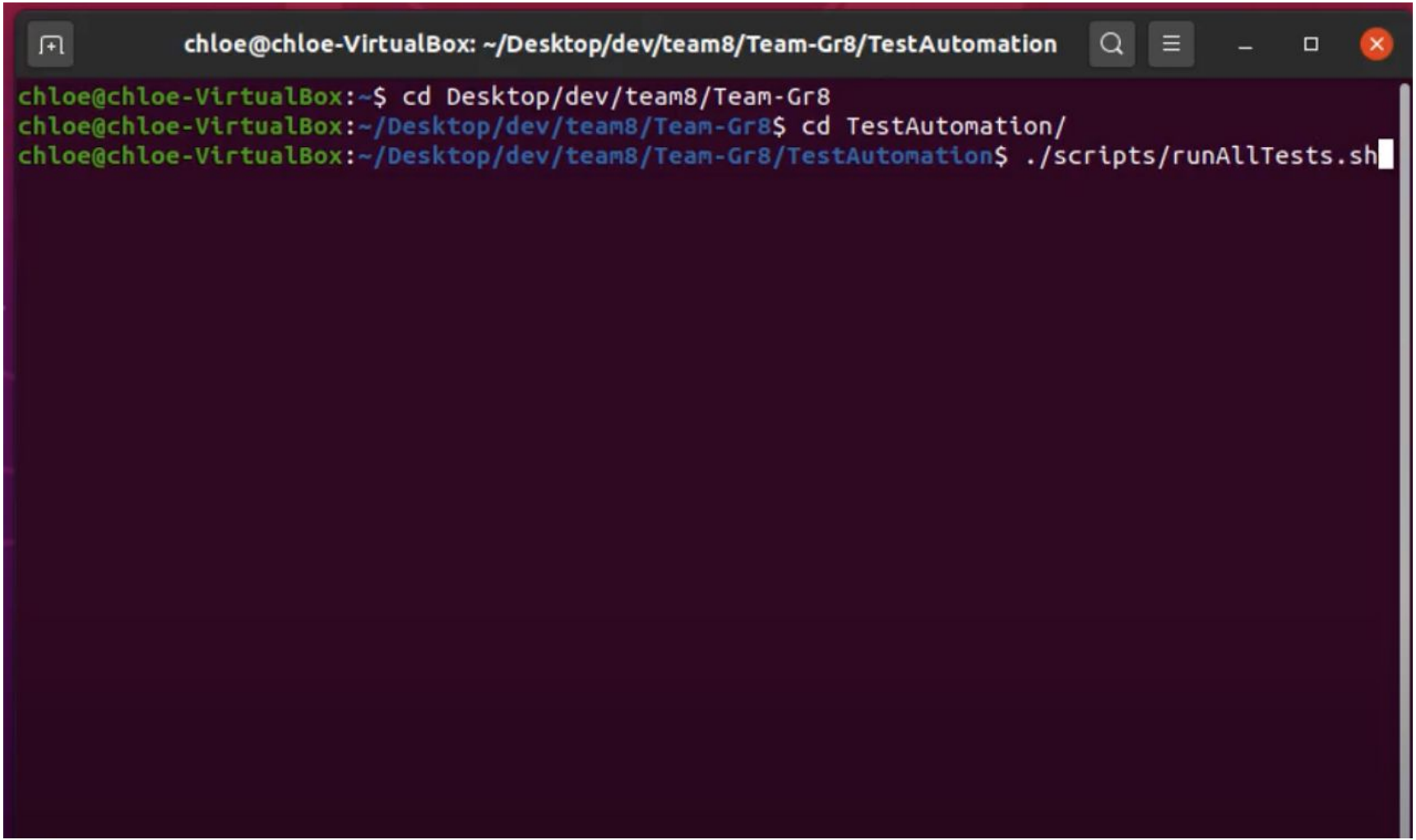
- runAllTests.sh
  - Calls runtest.sh for each file in TestCase folder
  - Adds output to table
  - Displays in browser
- Runtest.sh
  - Parses information from each line of the txt files of the test case
  - Compiles the driver and Tanaguru class needed for test case
  - Outputs to output.txt file and compares with oracle in test case
  - Cleans up directories

```
#!/bin/bash
# runAllTests.sh
# script that loops through all tests in testCases folder
# displays test results in a browser

cd oracles

# set up html to display in browser
cat >> "results.html" << EOF
<!DOCTYPE html>
  <html>
    <head>
      <title>Results</title>
      <meta charset="utf-8">
      <meta name="viewport" content="width=device-width, initial-scale=1">
      <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
      <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
      <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
      <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
    </head>
    <body>
EOF

# print title with time and date of test
DATETIME=`date`
echo "<div class=\"container pt-3\"><h3>Tanaguru Contrast-Finder Automated Testing</h3><h3>Test Results at $DATETIME</h3></div>" >> "results.html"

# get to TestAutomation folder
cd ..

# loop through test cases
for file in testCases/*
do
  testcasefile=`eval "echo $file | cut -d'/' -f2"`
  echo $testcasefile
  ./scripts/runtest.sh $testcasefile
done
cd oracles
# print results to html file
echo "</body>" >> "results.html"
echo "</html>" >> "results.html"

# open in browser
xdg-open "results.html"
sleep 10s

# remove results file after displaying them in the browser so a failed test run does not result in an old results page
rm $(find . -type f -name "results.html")
```

```
#!/bin/bash
# runtest.sh
# This runs a single test case

cd testCases

# if running a single testCase and there is no testCaseID argument then user must enter testCase#
if [ $# -eq 0 ]; then
    echo Enter the test case you would like to run: Format like testCase#
    read testcase
else
  # cut off the extension when running the full script
  testcase=`eval "echo $1 | cut -d'.' -f1"`
  fi

# ignore the README
if [ "$testcase.txt" == "README.txt" ]; then
  exit
  fi

cd ../oracles/
cd ../scripts/
TOPLEVEL="oracles"
ONEDOWN="../oracles"
TWODOWN="../../oracles"

testfile=../testCases/$testcase.txt

TESTCASEEXECDIR=/testCasesExecutables/
TESTCASEID=`cat $testfile | head -1 | tail -1`
COMPONENT=`cat $testfile | head -2 | tail -1`
REQUIREMENT=`cat $testfile | head -3 | tail -1`
TESTDRIVER=`cat $testfile | head -4 | tail -1`
TANAGURUFILE=`cat $testfile | head -5 | tail -1`
TESTMETHOD=`eval "echo $TESTDRIVER | cut -d'.' -f1"`
ARGS=`cat $testfile | head -7 | tail -1`
ORACLE=`cat $testfile | head -8 | tail -1`
TANAGURUFILEDIR=/project/src

# set up html formatting
echo "<div class=container>" >> "$ONEDOWN/results.html"

# go to the tanaguru file
cd ../$TANAGURUFILEDIR/

#compile the file into the directory with the driver
javac -d ../../$TESTCASEEXECDIR $TANAGURUFILE

# go to the folder that has the driver
cd ../../$TESTCASEEXECDIR/

# compile the driver
javac $TESTDRIVER

# Run the testCase file
java  $TESTMETHOD $ARGS > output.txt

OUTPUT=$(cat output.txt | tail -1)

```

Testing Cases

- DistanceCalculator > calculate()
  - incorrect - proved need for test cases
- ContrastChecker > distanceColor()
  - started with getComposantValue()
  - switched to distanceColor() to work with a more understandable method
- ColorConverter > rgb2Hex(), hex2Rgb(), offsetRgb()
  - easily confirmable using standard calculators for finding values


chloe@chloe-VirtualBox: ~/Desktop/dev/team8/Team-Gr8/TestAutomation
chloe@chloe-VirtualBox:~$ cd Desktop/dev/team8/Team-Gr8
chloe@chloe-VirtualBox:~/Desktop/dev/team8/Team-Gr8$ cd TestAutomation/
chloe@chloe-VirtualBox:~/Desktop/dev/team8/Team-Gr8/TestAutomation$ ./scripts/runAllTests.sh

Injecting Faults

To make sure our tests are working properly, we inject strategic faults and analyze the results of our automated testing framework.

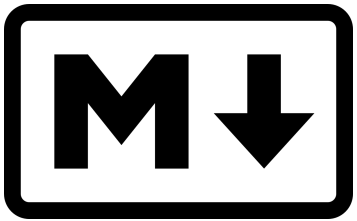
Fault 1:
In the ColorConverter class we inserted a fault into the offsetRgbColor() method on line 118.
Fault injected: Subtract the blue offset rather than adding it
Fault 2:
In the ContrastChecker class we inserted a fault into the distanceColor() method on line 66.
Fault injected: Incorrectly multiply red values instead of subtracting them to find the Euclidean distance
Fault 3:
In the ColorConverter class we inserted a fault into the rgbToHex() method on line 189.
Fault injected: Add an extra 0 into the formatter
Fault 4:
In the ColorConverter class we inserted a fault into the getNewColor() method on line 166.
Fault injected: Parse the Hex color string incorrectly from R to B instead of R to G
Fault 5:
In the DistanceCalculator class we inserted a fault into the calculate() method on line 30.
Fault injected: Set CUBIC variable to 0 instead of 3

With all five faults in code simultaneously, 10/25 test cases will pass (before faults 23/25 passed).

```
    /**
     *
     */
//    private ContrastChecker() {
//    }

    public static double distanceColor(final Color fgColor, final Color bgColor) {
        int redFg = fgColor.getRed();
        int redBg = bgColor.getRed();
        int greenBg = bgColor.getGreen();
        int greenFg = fgColor.getGreen();
        int blueFg = fgColor.getBlue();
        int blueBg = bgColor.getBlue();
        return (Math.sqrt(Math.pow(redFg * redBg, 2) + Math.pow(greenFg - greenBg, 2) + Math.pow(blueFg - blu
eBg, 2)));
    }

    /**
     *
     * @param fgColor
     * @param bgColor
     * @param coefficientLevel
     * @return
     */
    public static boolean isContrastValid(final Color fgColor, final Color bgColor, Float coefficientLevel) {
        return getConstrastRatio(fgColor, bgColor) > coefficientLevel;
    }
```

Lessons Learned

- Bash scripting / parsing text files
- The purpose of a driver
- Github / Git version control
- Markdown
- Java compiling via the command line
- VSCode
- The importance of testing!



Acknowledgements

A special thanks to Dr. Bowring for his guidance and feedback as well as the Tanaguru project and its open source status for the code (found at https://github.com/Tanaguru/Contrast-Finder).

Questions