DITAA Test Report

Version 1.0

4 December 2020

Kevin Porter

Adam Shiveley

Table of Contents

[Introduction 3](#_Toc57718147)

[Purpose 3](#_Toc57718148)

[Scope 3](#_Toc57718149)

[Test Plan 3](#_Toc57718150)

[Testing Environment 3](#_Toc57718151)

[Testing Methodology 3](#_Toc57718152)

[Test Cases 3](#_Toc57718153)

[Graphical User Interface 3](#_Toc57718154)

[Added Color Options 3](#_Toc57718155)

[Test Results 3](#_Toc57718156)

[Conclusion 4](#_Toc57718157)

# Introduction

## Purpose

The purpose of this document is to describe the test plan, test approach and methodology, test cases, and test results for this team’s implementation of the design for modifications to the Diagrams Through ASCII Art (DITAA) module. This document contains all the information a reader should need to understand what testing was planned for and accomplished, along with the results of those testing efforts.

## Scope

The scope of this document is limited specifically to content related to testing. Any questions with respect to specification, design, requirements, or implementation plans should be referred to those to documents. Additionally, this document does not describe any testing that already existed in the module prior to the team’s modifications, nor does it describe testing for any features that existed in the module prior to the team’s modifications.

# Test Plan

## Testing Environment

All testing for this module has been accomplished using Java version 8. As such, the module should be expected to run successfully on any modern operating system (Windows, Linux, macOS) with the appropriate version of Java. However, all testing has been done on Windows or Linux, as no support is offered for macOS. The program will be officially tested on Linux using Ubuntu 16.04.

## Testing Methodology

Testing for this project is divided into two sections: test cases and test results. Test cases are treated as descriptions of tests to be performed. A test case can be validated automatically through unit tests or manually through instrumentation of the module. The testing done for this project is not intended to be exhaustive or bulletproof, but rather a demonstration of the understanding of the amount of rigor and information required to execute a successful and useful testing scenario.

# Test Cases

## Command Line Interface

This test case refers to the Command Line Interface Use Case described earlier in this document.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Use Case Step | Description | Passing Result | Actual Result | Success/Fail |
| 1 | User generates and locates valid ASCII file | DITAA accepts ASCII file | DITAA accepts ASCII files with a certain format | Success |
| 2 | User launches DITAA from the shell | DITAA responds to inputs from the shell | DITAA responds to inputs from the shell | Success |
| 3 | User passes arguments to DITAA | DITAA processes arguments according to specification and documentation | DITAA processes arguments according to specification and documentation | Success |
| 4 | User retrieves processed output | DITAA processes inputs and generates output in the same directory as the input | DITAA processes inputs and generates a \*.png with the same name in the same directory as the input | Success |
| 5 | User is notified of program completion | DITAA exits gracefully | DITAA exits gracefully | Success |

## Graphical User Interface

This test case refers to the Graphical User Interface Use Case described earlier in this document.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Use Case Step | Description | Passing Result | Actual Result | Success/Fail |
| 1 | User generates and locates valid ASCII file | DITAA accepts ASCII file | DITAA accepts ASCII file with a certain format | Success |
| 2 | User double clicks the \*.jar file | DITAA GUI is presented to the user | DITAA GUI is presented to the user | Success |
| 3 | User can input optional arguments as mentioned in the companion documents | Input by User are processed by the program | Input by User are processed by the program | Success |
| 4 | User chooses the file with the ASCII art from a button click | DITAA presents a browse dialog and accepts a path for an input file | DITAA presents a browse dialog and accepts a path and input file | Success |
| 5 | The program is executed and a preview is displayed | The final result preview is shown to the user | The final results preview is shown to the user | Success |
| 6 | User can choose to save the final result as a \*.jpg or \*.png by clicking the appropriate button | A \*.jpg or \*.png with the User specified filename is created in the User specified folder | A \*.jpg or \*.png with the User specified filename is created in the User specified folder | Success |
| 7 | User exits the program | DITAA exits gracefully and notifies the User | DITAA exits gracefully and notifies the User | Success |

## 

## Added Color Options

This test case refers to the implementation of the additional color options described in Team 1’s specification document, which is the common specification document for all teams’ implementations.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Use Case Step | Description | Passing Result | Actual Result | Success/Fail |
| 1 | User generates an input ASCII file with new color options described in this document | ASCII file reflects a valid and compliant input file | ASCII file reflects a valid and compliant input file | Success |
| 2 | User proceeds through all steps of the CLI Test Case or the GUI Test Case | DITAA produces an output file with the appropriate colored sections as defined in the input file | DITAA produces an output file with the appropriate colored sections as defined in the input file | Success |

# Test Results

DITAA was executed from the command line with the following command:

Java -jar ditaa.jar “C:\DITAA\text\art3.txt”

The original art3.txt file contents is shown in Figure 1 and the results after processing with the DITAA program is shown in Figure 2.

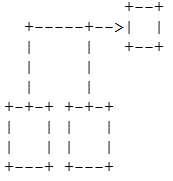
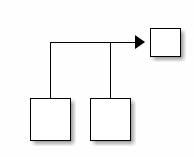
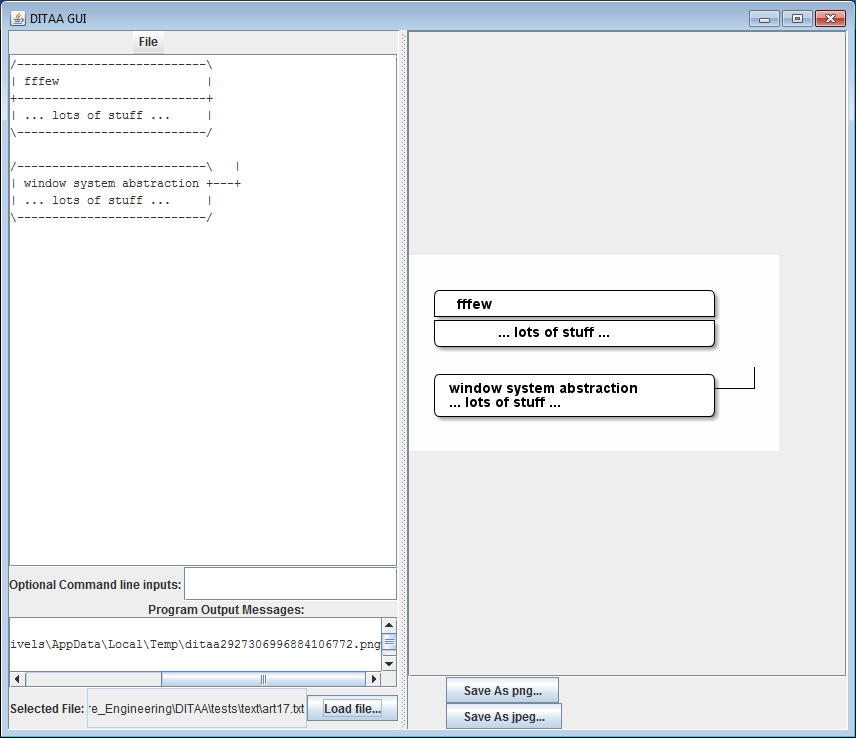
 

Figure 1: File contents for art3.txt Figure 2: File contents for art3.txt

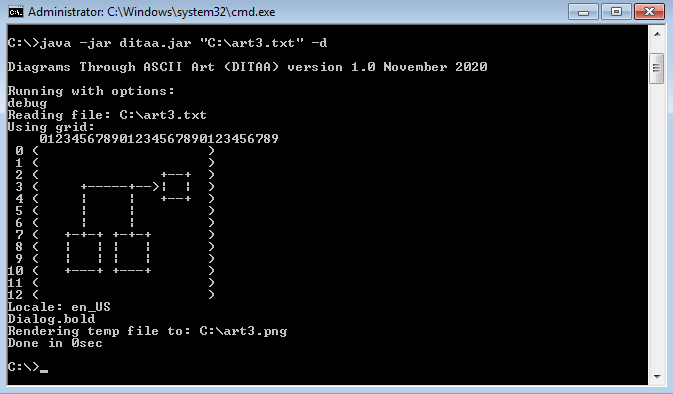
DITAA was also executed using the Graphical User Interface (GUI). A screenshot of the parameters used while operating in GUI mode is shown in Figure 3.



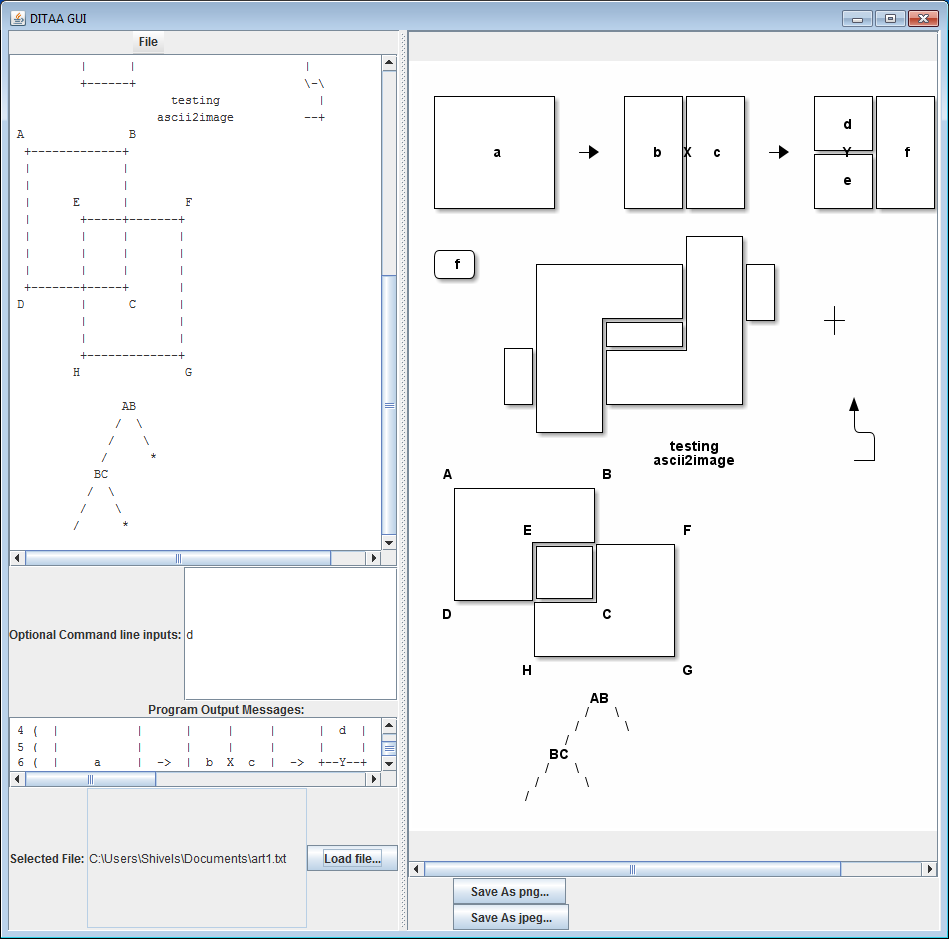
*Figure 3: DITAA GUI during execution*

As shown in Figure 3, the original file contents of the ASCII file is shown on the left panel and the resulting image after DITAA execution is shown on the right panel. The user has the option of saving as either a \*.png or \*.jpg based on which button the user presses.

Optional arguments can be sent to the program via the command line and GUI. These options are described in a companion document. Figures 4 and 5 show the program operating with optional operations, respectively.



*Figure 4: DITAA executed with debug option from the command line*



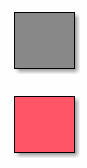
*Figure 5: DITAA executed using the GUI with debug option*

To demonstrate the custom colors work as intended, custom colors were added outside of the colors hardcoded into the program by default. The custom colors are detected based on an 8-bit hex code. For instance, the color White would be “FFF” and the color Black would be “000.” Table 1 shows the custom colors added to the program.

|  |  |  |
| --- | --- | --- |
| Color | Color Abbreviation | Hex Code |
| White | WHT | FFF |
| Orange | ORN | E82 |
| Purple | PRP | A0D |
| Brown | BRN | 641 |
| Indigo | IDG | 50D |
| Cyan | CYN | 3AA |
| Gray | GRY | 888 |
| Rose | RSE | F56 |

*Table 1: Custom Color codes added to the DITAA program*

The custom colors in table 1 were also hardcoded into the program. Figures 6 and 7 shows a simple object colored in Gray and Rose, respectively.

*Figure 6: File contents using custom colors in ASCII Figure 7: DITAA rendering using custom colors Gray (cGRY) and Rose (cRSE)*

# Conclusion

As shown in the Test Results section, the DITAA program performed as expected. Both the command line interface and the GUI performed as intended and met the requirements and specifications listed in the companion documents. Improvements were added to the GUI to enhance the user experience. Instead of a single “Save” button, two save buttons are used to allow the user to choose which format the resulting image should be saved as. Custom color codes were also added to the program. These values are hardcoded with descriptions included in the Readme. Overall, the program met or exceeded the testing matrix designed to ensure program effectiveness.