

SE 3XA3: Test Plan Gifitti

Team #2,Gifitti
Nicolai Kozel kozeln
Riley McGee mcgeer
Student 3 name and macid

October 24, 2016

Contents

1	General Information	1
1.1	Purpose	1
1.2	Scope	1
1.3	Acronyms, Abbreviations, and Symbols	1
1.4	Overview of Document	2
2	Plan	2
2.1	Software Description	2
2.2	Test Team	2
2.3	Automated Testing Approach	2
2.4	Testing Tools	2
2.5	Testing Schedule	2
3	System Test Description	2
3.1	Tests for Functional Requirements	2
3.1.1	Open GIF	2
3.1.2	Save GIF	4
3.1.3	Save GIF as Sprite Spreadsheet	5
3.2	Tests for Nonfunctional Requirements	6
3.2.1	Area of Testing1	6
3.2.2	Area of Testing2	6
4	Tests for Proof of Concept	6
4.1	Opening a GIF file for playback	6
4.2	Saving a GIF file's frames	7
5	Comparison to Existing Implementation	7
5.1	Graphics/UI	7
5.2	Performance	8
6	Unit Testing Plan	8
6.1	Unit testing of internal functions	8
6.2	Unit testing of output files	8
7	Appendix	9
7.1	Symbolic Parameters	9
7.2	Usability Survey Questions	9

List of Tables

1	Revision History	i
2	Table of Abbreviations	1
3	Table of Definitions	1

List of Figures

Table 1: **Revision History**

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

This document describes the test plan for the Gifitti application developed for 3XA3 at McMaster University.

1 General Information

1.1 Purpose

The purpose of the testing plan is to establish a set of tests that will test the product in its entirety to ensure that it fulfills the intended purpose. This would be accomplished through verifying if Gifitti satisfies the different functional and non-functional requirements that were assigned to it. Having test plans for any product is essential to be able to understand how well the product is satisfying the clients needs and if there are rooms for improvement.

1.2 Scope

This testing plan is utilizing different testing methods, automated and user created, and various techniques, black box and white box testing, to establish if the project has any need of improvement. Two different products will be analyzed through these tests, the Proof of Concept and the first iteration of the final product. Proof of Concept will be tested to ensure that a basic representation of the product was demonstrated while the requirements should be tested when the first iteration of the final product is completed.

1.3 Acronyms, Abbreviations, and Symbols

Table 2: Table of Abbreviations

Abbreviation	Definition
Abbreviation1	Definition1
Abbreviation2	Definition2

1.4 Overview of Document

The testing plan is broken up into distinct parts. Under the heading Plan, the basic information will be given on the product and the tests. System Test

Table 3: **Table of Definitions**

Term	Definition
Term1	Definition1
Term2	Definition2

Description will contain the specific tests for the functional requirements stated for the product. These tests will broken down into what type of tests they are and the results they achieve depending on their specific input. Test for nonfunctional requirments will follow the same format where input will be given and the output will be measured for all of the provided nonfunctional requirements. Under tests for proof of concept, the same format will be utilized as the functional testing but it will not be testing the requirements for the project but for the goals of the proof of concept. Furthermore there will be tests to compare Gifitti to the original product it was based on and unit testing plans to ensure correct output is achieved through proper internal functions.

2 Plan

2.1 Software Description

Gifitti is a software prodcut

2.2 Test Team

The team to implement the test plan for the project will be Pavle Arezina, Riley McGee, Nicolai Kozel

2.3 Automated Testing Approach

This test plan will not utilize an automated testing approach towarded Gifitti since the project centers around a graphical manipulation of the GIF.

2.4 Testing Tools

Tools to be utilized will be

2.5 Testing Schedule

See Gantt Chart at the following url ...

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 Open GIF

The User is Able to Open a GIF from a specified location

1. Select proper formatted gif- id1

Type: Manual Functional. Initial State: Program loaded; no GIF Loaded. Input: File name. Output: System loads gif into memory, displays it to the user in the gif view.

How test will be performed:

- (a) Launch the program
- (b) Select Open
- (c) From the Open dialog specify a path to a known gif image
- (d) After the image is loaded verify that it is being displayed, and resides in system memory

2. No File Selected in File Dialog-id2

Type: Manual Functional. Initial State: Program loaded; no GIF Loaded. Input: No file path. Output: No image loaded.

How test will be performed:

- (a) Launch the program
- (b) Select Open
- (c) Select open option with no file path specified
- (d) Verify program remains open, and no image is loaded

3. Close File Dialog-id3

Type: Manual Functional. Initial State: Program loaded; no GIF Loaded. Input: None. Output: No image loaded.

How test will be performed:

- (a) Launch the program
- (b) Select Open
- (c) Close the file dialog
- (d) Verify program remains open, and no image is loaded

4. Open random non gif file-id4

Type: Manual Functional. Initial State: Program loaded; no GIF Loaded. Input: File that is not a GIF. Output: No image loaded.

How test will be performed:

- (a) Launch the program
- (b) Select Open
- (c) Try and select a file that is not a GIF or specify a file path to a known file
- (d) Verify program remains open, and no image is loaded

3.1.2 Save GIF

The user is able to save a GIF to a specified location

1. Save GIF to known location-id1

Type: Manual Functional.

Initial State: Program loaded; GIF Loaded.

Input: File path, GIF file.

Output: GIF file saved to specified location.

How test will be performed:

- (a) Launch the program
- (b) Open a GIF
- (c) Select save as
- (d) Specify a known system file path and a saved image name
- (e) Verify the loaded and saved GIFs are identical

2. Save GIF to no existant location- id2

Type: Manual Functional.

Initial State: Program loaded; GIF Loaded.

Input: File path, GIF file.

Output: GIF file not saved to specified location.

How test will be performed:

- (a) Launch the program
- (b) Open a GIF
- (c) Select save as
- (d) Specify a system file path known not to exist and a saved image name
- (e) Verify the user is informed that the file path does not work

3. Save GIF to Opened Location- id3

Type: Manual Functional.

Initial State: Program loaded; GIF Loaded.

Input: File path, GIF file.

Output: GIF file saved to specified location.

How test will be performed:

- (a) Launch the program
- (b) Open a GIF

- (c) Modify the GIF
- (d) Save image
- (e) Verify new GIF is saved over the originally opened GIF

3.1.3 Save GIF as Sprite Spreadsheet

The user is able to save a GIF as a sprite spreadsheet in a specified location

1. Save Sprite Spreadsheet to known location-id1

Type: Manual Functional.

Initial State: Program loaded; GIF Loaded.

Input: File path, GIF file.

Output: GIF file saved as a Sprite Spreadsheet to specified location.

How test will be performed:

- (a) Launch the program
- (b) Open a GIF
- (c) Choose to export the image as a sprite spreadsheet
- (d) Verify that the GIF is a single image representation of the GIF via frames

3.2 Tests for Nonfunctional Requirements

3.2.1 Area of Testing1

Title for Test

1. test-id1

Type:

Initial State:

Input/Condition:

Output/Result:

How test will be performed:

2. test-id2

Type: Functional, Dynamic, Manual, Static etc.

Initial State:

Input:

Output:

How test will be performed:

3.2.2 Area of Testing2

...

4 Tests for Proof of Concept

4.1 Opening a GIF file for playback

Open GIF

1. OpenGif-01

Type: Manual Functional

Initial State: Program must be in normal state (form window is open and playback window is blank).

Input: File

Output: GIF is shown in playback window of the form.

How test will be performed: Click open button, select a file of type .gif, and verify that the GIF loads and begins to playback within the form window.

4.2 Saving a GIF file's frames

Save GIF

1. SaveGif-01

Type: Manual Functional

Initial State: Program must be in playback state (form window is open and playback window is playing GIF).

Input: GIF File

Output: GIF's frames are saved as .bmp in folder specified.

How test will be performed: Click save frames button, select a folder, and verify that the GIF's frames are saved to the specified folder as .bmp.

5 Comparison to Existing Implementation

5.1 Graphics/UI

1. GIF playback resolution is the same or better than Gif Viewer.
2. Program has the same button scheme as Gif Viewer (Open button to open file, Extract Frames button to save frames).
3. Program has a help menu available that is similiar to Gif Viewer, but is available at all times.
4. Program's color scheme and design resembles Gif Viewer.

5.2 Performance

1. GIF playback is at the same smoothness/framerate or better than Gif Viewer.
2. Opening and saving a file takes the same amount of time or less than Gif Viewer.

6 Unit Testing Plan

6.1 Unit testing of internal functions

6.2 Unit testing of output files

7 Appendix

This section contains symbolic parameters for this document and the usability survey that will be delivered to a focus group upon initial completion of the application.

7.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

7.2 Usability Survey Questions

The survey will be delivered in the same format as the Questionnaire for User Interface Satisfaction. This questionnaire is composed of various questions pertaining to several sub categories on a 0-9 scale. This includes the screen, terminology and system information, learning, and system capabilities. It also allows the user to list the most positive and negative aspects of the program. The questionnaire can be found at garyperلمان.com