Test Report
Liquid Rescaling

Team 35 - Marshiel
Lab 03
Marlee Roth - rothm1
Daniel Wolff - wolffd
Harsh Shah - shahhk2

December 6, 2017

Contents

List of Figures

List of Tables

1 Revision History

Date	Developer	Changes Made	Revision
November 25, 2017	Harsh Shah	Initial draft	Revision 1.0
November 27, 2017	Daniel Wolff	Update white-box tests	Revision 1.1
November 28, 2017	Marlee Roth	Begin black-box testing	Revision 1.2
November 30, 2017	Marlee Roth	Finalize black-box tests	Revision 1.3
November 30, 2017	Daniel Wolff	Finalize white-box tests	Revision 1.4
December 5, 2017	Harsh Shah	Supporting documentation	Revision 1.5
December 6, 2017	Harsh Shah	Finalizing document	Revision 1.5

Table 1: Caption

2 Overview

2.1 Summary

This document is intended to provide a complete description of all tests performed on the Liquid Rescaling Application. Due to the nature of the application, the design of the project was very user interface focused. Therefore, the structure of the modules was carefully designed to isolate the core functionality of the program from the user interface implementations, allowing as much unit testing as possible. The user interface was tested through manual testing only. All tests were done dynamically.

2.2 Environment and Background

2.2.1 Module Overview

The Liquid Rescaling Application is built with four modules (excluding the testing module):

Table 2: Brief description of modules

Module	**Description**
dialog.h	Handles dialog box actions
draw.h	Handles image displaying
rescale.h	Algorithm used for image processing
ui.h	User interface functions and event handling

The main.cpp file is responsible for running the program or running automated tests, depending on its configuration (see Automated Testing).

The test.h file contains all unit tests implemented with an automated testing framework.

2.2.2 Automated Testing

The automated test framework that is used is the main unit test framework used for C++ applications: CPPUnit. This testing is initialized by the definition of the *__DO_TESTING__* flag in main.cpp, and the output of the test results can be found in a generated file named *LiquidRescaleTestResults.xml* (located in the same folder as the application executable). All automated tests will be conducted as white box testing.

2.2.3 Manual Testing

All manual testing has been conducted by Team Marshiel and has been conducted on multiple Linux machines. All manual tests will be black box tests.

3 Functions

Table 3: Function Overview

Item No.	Module	Function	Input	Output
1.1	ui.h	run		integer
1.2	ui.h	init		integer
1.3	ui.h	init_filters		integer
1.4	ui.h	init_ui		integer
1.5	ui.h	init_styles	string	integer
1.6	ui.h	init_handlers		integer
1.7	ui.h	on_load_image		
1.8	ui.h	on_save_image		
1.9	ui.h	on_scale_image		
2.1	dialog.h	GetFileExtension	const string&	string
2.2	dialog.h	displayMessage	Window&, const string&	bool
2.3	dialog.h	openImageDialog	Window&, RefPtr<FileFilter>, string&	bool
2.4	dialog.h	saveImageDialog	Window&, RefPtr<FileFilter>, RefPtr<Pixbuf>, string&	bool
3.1	draw.h	Dimensions::operator==	const Dimensions&	bool
3.2	draw.h	bestFitToDimensions	Dimensions, Dimensions, integer	Dimensions
3.3	draw.h	drawImage	Window&, RefPtr<Pixbuf>, Image, integer	
4.1	rescale.h	pixelFromPixbufData	guint8*, integer, integer	guchar*
4.2	rescale.h	pixbufFromCarver	LqrCarver*	RefPtr<Pixbuf>
4.3	rescale.h	bufferFromPixbuf	RefPtr<Pixbuf>	guchar*
4.4	rescale.h	liquidRescaleImage	RefPtr<Pixbuf>, integer, integer	RefPtr<Pixbuf>

4 Test Results

4.1 White Box (Automated) Testing

This section is a summary of all tests ran through automated testing in the test.h module. These tests exercised a white-box (structural) approach to testing. Exception tests are separated from normal tests.

4.1.1 Testing Function 2.1: GetFileExtension

Table 4: Normal tests for GetFileExtension

Test ID	Test case	Expected result	Actual result	Result
WT1	"afe.c.bcd"	"bcd"	"bcd"	Passed
WT2	"a.b"	"b"	"b"	Passed
WT3	".b"	"b"	"b"	Passed
WT4	"a."	""	""	Passed
WT5	"a"	""	""	Passed
WT6	"a.b.c"	"c"	"c"	Passed
WT7	"a.b.."	""	""	Passed
WT8	""	""	""	Passed
WT9	"."	""	""	Passed

Table 5: Exception tests for GetFileExtension

Test case	Expected result	Actual result	Result

4.1.2 Testing Function 3.1: Dimensions::operator==

Table 6: Normal tests for Dimensions::operator==

Test ID	Test case	Expected result	Actual result	Result
WT10	x(100,100), y(100,200)	false	false	Passed
WT11	x(100,100), y(200,100)	false	false	Passed
WT12	x(100,200), y(100,100)	false	false	Passed
WT13	x(200,100), y(100,100)	false	false	Passed
WT14	x(100,100), y(100,100)	true	true	Passed

Table 7: Exception tests for Dimensions::operator==

Test case	Expected result	Actual result	Result

4.1.3 Testing Function 3.2: bestFitToDimensions

Table 8: Normal tests for bestFitToDimensions

Test ID	Test case	Expected result	Actual result	Result
WT15	src(100,100), display(100,100), buff=0	(100,100)	(100,100)	Passed
WT16	src(100,100), display(100,100), buff=10	(80,80)	(80,80)	Passed
WT17	src(50,100), display(100,100), buff=10	(40,80)	(40,80)	Passed
WT18	src(1,2), display(100,100), buff=10	(40,80)	(40,80)	Passed
WT19	src(100,100), display(100,40), buff=10	(20,20)	(20,20)	Passed
WT20	src(100,100), display(40,100), buff=10	(20,20)	(20,20)	Passed
WT21	src(200,400), display(100,40), buff=10	(10,20)	(10,20)	Passed
WT22	src(200,400), display(40,100), buff=10	(20,40)	(20,40)	Passed

Table 9: Exception tests for bestFitToDimensions

Test ID	Test case	Expected result	Actual result	Result
ET1	src.width < 1	invalid_argument	invalid_argument	Passed
ET2	src.height < 1	invalid_argument	invalid_argument	Passed
ET3	display.width < 1	invalid_argument	invalid_argument	Passed
ET4	display.height < 1	invalid_argument	invalid_argument	Passed
ET5	buffer < 0	invalid_argument	invalid_argument	Passed
ET6	display.width - buffer * 2 < 1	invalid_argument	invalid_argument	Passed
ET7	display.height - buffer * 2 < 1	invalid_argument	invalid_argument	Passed

4.1.4 Testing Function 4.2: pixbufFromCarver

Table 10: Normal tests for pixbufFromCarver

Test ID	**Test case**	**Expected result**	**Actual result**	**Result**
WT23	carver(someImage)	result[i] = carver[i]	result[i] = carver[i]	Passed

Table 11: Exception tests for pixbufFromCarver

Test case	**Expected result**	**Actual result**	**Result**

4.1.5 Testing Function 4.3: bufferFromPixbuf

Table 12: Normal tests for bufferFromPixbuf

Test ID	**Test case**	**Expected result**	**Actual result**	**Result**
WT24	pixbuf(someImage)	result[i] = pixbuf[i]	result[i] = pixbuf[i]	

Table 13: Exception tests for bufferFromPixbuf

Test case	**Expected result**	**Actual result**	**Result**

4.1.6 Testing Function 4.4: liquidRescaleImage

Table 14: Normal tests for liquidRescaleImage

Test ID	Test case	Expected result	Actual result	Result
WT25	someImage(100,100), newW=100, newH=100	res.w=newW, res.h=newH	res.w=newW, res.h=newH	Passed
WT26	someImage(100,100), newW=200, newH=300	res.w=newW, res.h=newH	res.w=newW, res.h=newH	Passed
WT27	someImage(100,100), newW=200, newH=50	res.w=newW, res.h=newH	res.w=newW, res.h=newH	Passed
WT28	someImage(100,100), newW=50, newH=300	res.w=newW, res.h=newH	res.w=newW, res.h=newH	Passed
WT29	someImage(100,100), newW=50, newH=25	res.w=newW, res.h=newH	res.w=newW, res.h=newH	Passed
WT30	someImage(100,100), newW=2, newH=2	res.w=newW, res.h=newH	res.w=newW, res.h=newH	Passed

Table 15: Exception tests for liquidRescaleImage

Test ID	Test case	Expected result	Actual result	Result
ET8	someImage(100,100), newW=1, newH=10	invalid_argument	invalid_argument	Passed
ET9	someImage(100,100), newW=10, newH=1	invalid_argument	invalid_argument	Passed
ET10	someImage(100,100), newW=1, newH=1	invalid_argument	invalid_argument	Passed
ET11	someImage(1,1), newW=10, newH=10	invalid_argument	invalid_argument	Passed
ET12	someImage(1,10), newW=10, newH=10	invalid_argument	invalid_argument	Passed
ET13	someImage(10,1), newW=10, newH=10	invalid_argument	invalid_argument	Passed

4.2 Black Box (Manual) Testing

This section is a summary of tests regarding the system as a whole using a black-box (functional) approach. The system was tested with the black-box tests outlined in the Test Plan. All of the following tests are conducted on the initial state defined by the description of the test in the Test Plan.

Table 16: Black-box System Tests

Test ID	**Expected result**	**Actual result**	**Result**
BT1	Acceptable output image	Acceptable output image	Passed
BT2	Acceptable output image	Acceptable output image	Passed
BT3	Acceptable output image	Acceptable output image	Passed
BT4	Acceptable output image	Acceptable output image	Passed
BT5	Acceptable output image	Acceptable output image	Passed
BT6	Acceptable output image	Acceptable output image	Passed
BT7	Acceptable output image	Acceptable output image	Passed
BT8	Acceptable output image	Distorted output image	Failed
BT9	Transparent image loaded	Transparent image loaded	Passed
BT10	Error notification shown	Error notification shown	Passed
BT11	Image loaded successfully	Image loaded successfully	Passed
BT12	Image saved successfully	Image saved successfully	Passed
BT13	Error notification shown	Error notification shown	Passed
BT14	Error notification shown	Error notification shown	Passed
BT15	Override prompt appears	Overwrite prompt appears	Passed
BT16	Controls activate on load	Controls activate on load	Passed
BT17	Image loaded successfully	Image loaded successfully	Passed
BT18	Error notification shown	Error notification shown	Passed
BT19	Error notification shown	Error notification shown	Passed
BT20	UI doesn't allow	UI doesn't allow	Passed

Table 17: Black-box System Tests (continued)

Test ID	Expected result	Actual result	Result
BT21	UI doesn't allow	UI doesn't allow	Passed
BT22	UI doesn't allow	UI doesn't allow	Passed
BT23	Image scales successfully	Image scales successfully	Passed
BT24	UI doesn't allow	UI doesn't allow	Passed
BT25	Image quality is maintained	Image quality is maintained	Passed
BT26	Image saved successfully	Image saved successfully	Passed
BT27	Image saved successfully	Image saved successfully	Passed
BT28	Error notification shown	Error notification shown	Passed
BT29	Error notification shown	Overwrite prompt appears, error notification shown after	Passed
BT30	Error notification shown	Error notification shown	Passed
BT31	Error notification shown	UI does't allow	Passed

5 Analysis of Test Results

5.1 Changes Made

The execution of the test results recorded above (both automated and manual), provided many opportunities to change the project code to be more robust and reliable under abnormal situations. The following are the major changes added to the system due to unexpected test results:

- Many abnormal scenarios when saving a file were tested - most of which provided a reason to make major changes to how the system handles unexpected save requests. For example, unexpected characters in a file name and trying to overwrite a non-image file were two scenarios that broke earlier versions of the application. Thankfully, due to testing, later versions of the project were able to handle these situations appropriately.

- Slight edge cases on image sizes were overlooking during earlier development of the system. As a result of testing, it was clear that the usual minimum side length of an image (1 pixel) was not supported by the Liquid Rescaling library that the project relied on, causing the system to crash when a user requested to rescale an image to have a side length less than 2 pixels long.

- Extreme conditions such as scaling to a 2000x1 image were tested as well, which actually resulted in an unusual situation where the program would try and scale it into the display window, but by doing so scaled one edge of the image down to zero. In light of this, a prompt was created to warn users about this scenario, and it allows them to expand the window to fit the original image.

5.2 Non-Functional Requirements Satisfied

Many of the black-box tests were inspired by the non-functional requirements defined by the development team at the beginning of the project.

1. Appearance To test the appearance requirements of the application human centred testing was used. A person looked at the user interface and confirmed a static design with an exit button in the top corner.

2. Usability To test usability of the application random people of random age where asked to use the application without guidance. If they could use it without instruction it would mean high usability.

3. Safety To test safety the program was loaded with an image. This image was edited and not saved. If the app overrode the existing data with the non-saved data it would mean it was not safe.

4. Privacy The application does not require privacy testing as it does not save any files outside of the local machine.

5. Performance The performance of the app can be tested by inputting a large image and scaling it to be very small while timing it. The launch time can be tested in the same way.

6. Installability The application runs off a single executable file making it very installable.

7. Portability To test portability the executable file can be launched on different operating systems to see if the application will run.

8. Learning The application only has 6 buttons all of which say their function on them making the learning requirement satisfied.

6 Gantt Chart

The updated Gantt chart can be found in the get repository named Revision 1.
See file LiquidRescalingDevelopmentSchedule - Version 4.pdf