Test Plan Revision 1
Liquid Rescaling

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

December 6, 2017

Contents

List of Tables

1 Revision History

Table 1: Revision History

Date	Developer	Changes Made	Revision
October 27, 2017	Whole Team	Initial draft	Revision 0
December 2, 2017	Marlee Roth	Modified tools	Revision 1
December 3, 2017	Harsh Shah	Modified the Technologies Section	Revision 1
November 24, 2017	Daniel Wolff	modified the testing	Revision 1
December 2, 2017	Marlee Roth	Modified the Non Functional Requirements (Installability and Portability)	Revision 1

**Note blue text implies there has been a revision to that content. If a section name is blue then all the content in that section has been altered.*

2 Testing Tools

2.1 Tools used

The tests will be run using the CppUint testing built-in unit testing framework to perform automated testing on features that have a definite output given a certain input. For example, if an image is scaled to have a width of 500 and a height of 600, the resulting image must have a width of 500 and a height of 600.

2.2 Type of Tests

The types of tests that will be performed are automated dynamic tests using CPPunit as stated above, and manual testing for the image outputs. The automated tests will be designed using the white box technique to cover all edge cases of the features, and the manual tests will be black box technique since the implementation of the visual output of the scaled images are not known.

2.3 Plan for Tests

Automated tests will be run using CppUnit which will be initiated from the main.cpp file. This will happen when the testing flag is defined. Manual tests will consist of images being scaled using the application and GIMP's Liquid Rescaling plugin. The output of the two will be compared, and should be very similar. There will also be black box testing on the user interface to check if there is any inputs that may break the code. These test must be black box as it requires user interaction to initiate the result. A pass or fail will be given by the tester, by means of a visual check, if certain qualities have been met with each test.

3 System Test Cases

3.1 Black Box Test Cases

- **BT1**
  - Test scaling, preservation and, amplification on simple black and white images which consist of a solid coloured background with 2 or 3 objects of different shades. Objects should preferably be simple solid shapes such as a square, rectangle, circle, sphere, cylinder and/or cube.
- **BT2**
  - Test scaling, preservation and, amplification on simple coloured images which consist of a solid coloured background with 2 or 3 objects of different colours. Objects should preferably be simple solid shapes such as a square, rectangle, circle, sphere, cylinder and/or cube.
- **BT3**
  - Test scaling, preservation and, amplification on images which consist of a gradient background with only 1 object of different shades. Objects should preferably be simple solid shapes such as a square, rectangle, circle, sphere, cylinder and/or cube.
- **BT4**
  - Test scaling , preservation and, amplification on images which consist of a multicoloured background with only 1 object. Such as an image of the ocean and the sky with one bird or a small island.
- **BT5**
  - Test scaling, preservation and, amplification on images which consist of a solid background with only 1 complex object. Such as an image of a person or animal walking on the beach or in the water of the ocean.
- **BT6**
  - Test scaling, preservation and, amplification on images which consist of a solid background with multiple complex objects. Such as an image of a group of people playing volleyball on the beach.
- **BT7**
  - Test scaling, preservation and, amplification on colourful images which consist of multiple complex objects. Such as an image of the inside/outside of the a busy shopping centre which has lots of festive decorations.

- **BT8**
  - Test scaling, preservation and, amplification on a complicated and recognisable object. Such as an image of a face.
- **BT9**
  - Test if a transparent image file can be uploaded.
- **BT10**
  - Test if a 1x1 image can be uploaded.
- **BT11**
  - Test if a 2001x2001 image can be uploaded.
- **BT12**
  - Test that an image can be saved after it has been edited.
- **BT13**
  - Test that an image can be saved without a file ending.
- **BT14**
  - Test that an image can be saved with an incorrect file ending
- **BT15**
  - Test that an image can override an image with the same name.
- **BT16**
  - Test that controls activate only if an image is uploaded.
- **BT17**
  - Test if a 2x2 image can be loaded.
- **BT18**
  - Test if a 1x2000 image can be loaded.
- **BT19**
  - Test if a 2000x1 image can be loaded.
- **BT20**
  - Test scaling a 1000x1000 image to 1x1.
- **BT21**

    - Test scaling a 1000x1000 image to 1x2000.

- **BT22**
    - Test scaling a 1000x1000 image to 2000x1.
- **BT23**
    - Test scaling a 1000x1000 image to 2000x2000.
- **BT24**
    - Test scaling a 1000x1000 image to 3000x3000.
- **BT25**
    - Test scaling a 1000x1000 image to 2x2, and then back up to 1000x1000.
- **BT26**
    - Test saving an image that hasn't been edited.
- **BT27**
    - Test saving an image with two file endings (rightmost is correct).
- **BT28**
    - Test saving an image with two file endings (rightmost is incorrect).
- **BT29**
    - Test saving an image by overwriting a non-image file.
- **BT30**
    - Test saving an image with special characters in its file name.
- **BT31**
    - Test saving an image with a blank file name.

3.2 White Box Test Cases

3.2.1 Function: GetFileExtension

Table 2: Normal tests for GetFileExtension

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

3.2.2 Function: Dimensions::operator==

Table 3: Normal tests for Dimensions::operator==

Test case	Expected result	Test Number
x(100,100), y(100,200)	false	WT10
x(100,100), y(200,100)	false	WT11
x(100,200), y(100,100)	false	WT12
x(200,100), y(100,100)	false	WT13
x(100,100), y(100,100)	true	WT14

3.2.3 Function: bestFitToDimensions

Table 4: Normal tests for bestFitToDimensions

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

3.2.4 Function: pixbufFromCarver

Table 5: Normal tests for pixbufFromCarver

Test case	Expected result	Test Number
carver(someImage)	result[i] = carver[i]	WT23

3.2.5 Function: bufferFromPixbuf

Table 6: Normal tests for bufferFromPixbuf

Test case	Expected result	Test Number
pixbuf(someImage)	result[i] = pixbuf[i]	WT24

3.2.6 Function: liquidRescaleImage

Table 7: Normal tests for liquidRescaleImage

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

3.3 Exception Testing

** Please note that there are many more possible exceptions that have not uncovered yet here are just a few basic ones.*

- The program shall generate an exception if the user inputs a file with an unacceptable file format (i.e. not an image file).
- The program shall generate an exception if the user requests to save the output file in an a non-existing directory.
- The program will not initiate the control panel unless an image has been uploaded.
- The program will inform the user if an image does not save.
- Bellow the known exceptions for the functions.

3.3.1 Function: bestFitToDimensions

Table 8: Exception tests for bestFitToDimensions

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

3.3.2 Function: liquidRescaleImage

Table 9: Exception tests for liquidRescaleImage

Test case	**Expected result**	**Test Number**
someImage(100,100), newW=1, newH=10	invalid_argument	ET8
someImage(100,100), newW=10, newH=1	invalid_argument	ET9
someImage(100,100), newW=1, newH=1	invalid_argument	ET10
someImage(1,1), newW=10, newH=10	invalid_argument	ET11
someImage(1,10), newW=10, newH=10	invalid_argument	ET12
someImage(10,1), newW=10, newH=10	invalid_argument	ET13

4 Testing Proof of Concept

The proof of concept will be testing the applications ability to accurately scale and preserve an image. The application should be able to perform content aware image re-sizing on any image chosen. The main aspect that need to be tested are the images clarity, and likeness to the original image. This cannot be done using automated testing therefore it will need to be done manually. The tests done on the application are specified in Section 3.1.

5 Non Functional Requirements

To ensure the program is working at its best, non-functional requirements of the program will be tested. **See section 3.1 for more details**

1. Appearance
   Is tested manually by having a person check to see that the design is sleek and simple with a exit button in the top corner

2. Usability
   Have multiple different users from different age groups perform a series of tasks using the application without guidance. If the users are able

to perform the tasks without much trouble then the application will be considered user friendly.

3. Safety
   Images can be saved with the same file name as an existing file to see if a notification is given, the file can be given the wrong file ending to see if it still saves. The image can also be edited but not saved to make sure the program does not save images unless specified.

4. Privacy
   The program does not save any of the data it is given out side of the local machine.

5. Performance
   The launch speed of the program can be tested with a timer. To test its capabilities large images can be processed to 2x2 image while being timed and vice verse. to check response time a user can click the UI buttons and see if there is any noticeable lagging.

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

7. Portability
   The program can be run off the executable file and see if it works on operating systems other than Linux

8. Learning The program has a total 6 button all of which say exactly what they do on them.

6 Test Plan Schedule

The updated Gantt Chart can be found in the git repository marked Revision 1. See File **Liquid Rescaling Development Schedule - Version 4.pdf**