Project Name
Test Plan

Table of Contents

History of Changes

Version	**Date**	**Change**
First Draft	10/10/2005	
Second Draft	10/11/2005	Types of testing
Third Draft	10/18/2005	Added Test
Final Draft	11/22/2005	Revised procedure, added tests

Related Documents

Requirements Document: http://swiki.cc.gatech.edu:8080/cs4911b-fl05/143

Test Team

Testing Manager: Dustin Roberts
Sound Tester: John Burton
Image Tester: Andrew Nagel
GUI Tester: Sam Gawthrop

Testing Strategy

Initially, all tests will be written in a black box fashion. The tests will include unit tests from a previous semester's team, and all sample code in "Introduction to Media Computation" by Mark Guzdial. If the software can run all of the sample code in this book, it has met the acceptance criteria. When major bugs are discovered, we will then write specific tests for that function. For these specific tests, we will use white box testing, so that every line of code can be run in those functions.

A major bug is one that takes more than 30 minutes to repair, or that has to employ a work around in order to work with currently available libraries.

In addition to functionality testing, we will also test performance. In order to test performance, we will use the "Time" function in Python to record how long a program takes to run. We will use the sample code from Mark Guzdial's book to get the benchmarks. First, we will run the code on JES and record the times. These tests will be run on both Windows and Mac. We will then run the code using the new Media.py and record those times. These tests will also be run on both Windows and Mac. All results will be shown graphically using a bar chart.

Testing will be done in a "Top-down" fashion. This is inherently true because the examples in the book start out doing very broad tasks, which use large portions of the library, and slowly dwindle down into the specific function.

Dustin Roberts is responsible for writing all of the initial tests. This does not mean that each individual will not write tests; in fact they are required to write a test if they discover a bug, or are unsure if something is running correctly. Dustin Roberts is responsible for getting with each programmer and running tests that pertain to their code. Team members may run the tests on their code alone, but their results are not certified as correct until Dustin Roberts has run all the tests with them. Once all tests have been run once, the programmer will make necessary repairs, and write extra tests for any major bug. The programmer will then contact Dustin Roberts, and they will meet for another test run. This will occur until all tests are passed satisfactory with Dustin Roberts present.

Items Not Covered by These Test Cases

PyGames will not be tested by the test cases. This software is a third party library, that is assumed to have already been tested.
CPython will not be tested because it is the virtual machine, which will run our code, and it is also assumed to have already been tested by its developers.
WxWindows will not be tested by our code either, because it too is a third party library and is assumed to be tested by its developers.

Bug Tracking

An Excel spreadsheet will be used to track bugs. It will contain the following information: description of the bug, person who discovered it, when it was discovered, who is responsible for it, how it was fixed, and when it was fixed. Each teammate is responsible for recording bugs in the Excel spreadsheet when they discover them. Andrew Nagel is responsible for logging all bugs reported by beta testers.

Quality Control

All team members, prior to its submittal, will review the test plan. The advisor will also review the test plan and add anything he thinks would be beneficial to the team.

The customer decided that he would like to test the software on DrPython, since a class in Australia will be using DrPython for their students. This requires us to run all unit tests and tests from the book on DrPython in addition to CPython.

Adequacy Criterion

When all tests from "Introduction to Media Computation" have been passed, then beta testing with the customer will begin. After a 2-week period, beta testing will end and all reports of bugs will be repaired. Once all bugs are repaired, we will run the tests again to make sure no functionality has been broken. Mark Guzdial will then accept the product once all tests are passed in his presence.

Planned Test Cases

Test #	Purpose	Action and Input	Expected Result	Actual Result	P/F	Notes

FUNCTIONAL						
1	Test the MakeColor, distance, newColor, makeLighter, makeDarker functions.	Use the ColorTest.py file. When run it will perform actions using the Media.py and output results to screen	Data should match everything written on the screen			Print statements tell you exactly what should be printed form the result. If they match, you have passed.
2	MakePicture, getMediaPath, getHeight, getPixel, setColor	Flower1.jpg and Flower2.jpg should be in the directory where media.py is loaded, use createCollage.py	The two flowers should be combined into 1 picture, they will be side by side			See p. 96 of the book for expected result
3	GetRed, setRed	Barbara.jpg should be used and the decreaseRed.py	Red levels should be reduced by ½.			Pay close attention to red levels
4	AddText, addLine, addRectFilled, addRect	Use 640x480.jpg	A piece of text, a line, and rectangles on the screen.			
5	GetPixels, setColor	Use the Barbara.jpg file and the greyscale.py file.	This will convert the color image into a grey scale image			If you test on another picture, use a color one.
6	PickAFile, various setters and getters	Make sure a jpeg image is available	Output should match the printed text.			The output will follow a print statement. The print statement has the expected result.
7	MakeLighter "for" loop	Use the Barbara.jpg file and the "lighten loops.py" file	Picture will be lighter by 1/3 of its current shade			
8	Tests using math functions on pixel values	Use the daisies.jpg file and the lineDetect.py file.	New picture will be an outline of the old.			Look at the old and the new picture together
9	Using "range" with pixel	Use the Barbara.jpg	Draws a grid on the pictuere that			

	values	file and the line.py file	is 5 pixels by 5 pixels.			
10	Tests math functions on getPixel	Use the barabara.jpg file and the makeNegative.py file	This will make the negative of the picture.			
11	Tests conditional statements with pixels.	Use the students-on-tour.jpg and the posterize.py	The image will be posterized			This normalizes the colors.
12	Tests deleting and adding pixels onto a canvas	Use the barabara.jpg file and the scaleDown.py file	This will only capture Barbara's head and scale it by 2/3.			
13	GetSampleValueAt, getLength, makeSound	Use the preamble.wav file and the backwards.py file	This will reverse the sound.			When playing, it will sound garbled, because it is backwards.
14	BlockingPlay	Use the preamble.wav file	You will ONLY hear the preamble once, but the system tries to play it 3 times at once			There should only be one copy playing at a time
15	SetSampleValueAt	Use the preamble.wav file and the inc dec.py file	This will be loud until the middle of the sound, then it will get quieter			
16	Increasing the volume	Use the preamble wav and the IncreaseVol.py file	This will increase the intensity of the file			Remember that an increase in intensity does not increase the volume by the same amount
17	GetSampleObjectAt	Use the preamble.wav and the mirror.py file	This will mirror the sound file			
			Performance Tests (Non functional)			
1	SetRed, setBlue,	Use the Barbara.jpg	This will blur the image, it use to			If it is functioning

	setGreen, and set Color	file and the blur.py file	take a very long time on JES, make sure to record the time			correctly, then be sure to record the time.
2	Mirror the image	Use the santa.jpg file and the mirror.py file	This will mirror the santa image halfway through the picture.			The picture will be vertically mirrored; horizontal uses the same functions.
3	Rotate the image 90 degrees	Use the barabara.jpg file and the rorate.py file	This will rotate the image 90 degrees counterclockwise			Use to take a long time on JES
4	Mirror another image	Use the Temple.jpg and the mirror.py file	This will mirror an image and make it look like you have repaired the damage to it			Look at results on p. 86 of the book
5	Normalize a sound	Use the bassoon-c4.wav file and the normalize.py file	This will take out the peaks and valleys and make it sound more flat			
6	Combine 2 sounds	Use guzdial.wav and is.wav with splice.py file	This will combine the 2 wav files			Should say "Guzdial is"
			Error testing			
1	Access pixels out of bounds	Use the Barbara.jpg file and try to access a pixel past the end of the file	Gives error to the using saying "Pixel value out of bounds"			
2	Accessing samples out of bounds	Use the "is.wav" file and try to access a sample past the last index	Should display message saying "Sample value out of bounds"			
3	Try to open a picture that does not exist	Try to open a jpeg file that doesn't exist.	Error displayed will say "Image does not exist"			
4	Try to open a sound that doesn't exist	Open an image file that does not exist	Error displayed saying "File does not exist"			

Usability tests						
1	Beta Testing	Since the program will be extensively beta tested, we will send a survey to each tester.	Survey will ask to rank on likert scale: ease of install, performance, would they recommend it, would they use it for their students.			These questions will just help us know if people like the new Media.py, results will be given to the customer
2	Customer use	We will have the customer use the software and have a short Q&A session with him	This is to be performed at the beginning of the beta testing, so we can learn what the customer would like to be added, change, or get rid of.			

- It should be noted that the Media.py system is already in use by the CS1315 classes and are copying its functionality; therefore the current students and the professor have already established usability.
- Also note that performance benchmarks will be taken on every functional test as well as every performance test and will be measured in seconds and tenths of seconds.

Legend

Test #	Test Case Number / Identifier
Purpose	Reason that the test case is being run. For black-box tests, this is the requirement that the test cases are validating (number / identifier). For white-box tests, this is the code segment that is being exercised.
Action and Input	Scripted set of steps to perform test along with input data to use (or a pointer to a test case file)
Expected Result	Result expected when action is complete; output data values
Actual Result	What was actually seen. Failed cases should be marked with the date and time of the failure, and the associated test track number. When the failed cases is fixed, the date and time of the retest should be noted.
P / F	Pass / Fail indicator. Checkmark = Pass. “F” = Fail
Notes	Additional notes, error messages, or other information about the test.