CSCI 362 Fall 2016
Software Engineering
Joey Baldwin
Eric Hofesmann
Marge Marshall

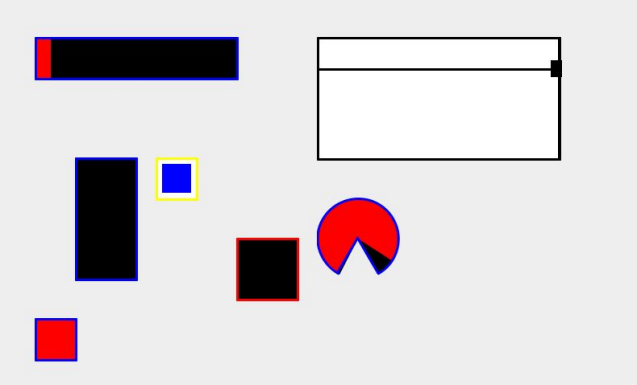
Cygnus Inter Anates Deliverable 1

JythonMusic is an open source music software project written in Python, created with Jython, and using several embedded Java software packages, such as jMusic and jSyn. jMusic is a package for music composition, and jSyn is for audio synthesis. These two packages provide a way to utilize the JythonMusic code. Our compile process was not complicated; a jython shell is included, which is used to compile files into Java bytecode, and the code is then run the same as Java code would be.

JythonMusic seems like it would be a useful tool for musicians, music composers, and programmers interested in music. The JythonMusic project gives the developer the means to compose, connect to external devices such as midi and osc, and also includes some simple gui functionality. There is also a user interface, the JEM editor, included in the source. The API is fairly simple and easy to follow.

There are nine files that make up the JythonMusic library, these are audio.py, gui.py, guicontrols.py, image.py, midi.py, music.py, osc,py, timer.py, and zipf.py. Six of these JythonMusic programs contain unit tests. These tests are positioned in main methods at the bottom of the files that they concern. The tests expose a variety of features within JythonMusic including UI elements and business functions. Tests are not ubiquitous throughout the project. Some files remain untested. These tests also are not complete, they appear to only test the essentials. There remain functionalities that have not been tested.

Gui.py is based on Java's Swing library, to give Jython GUI functionality. The test required image and sound downloads to work, as the test referenced files that did not exist and weren't included. However, once replacements were found for these, it generated 6 displays demonstrating control functions, animation and interaction of objects, and handling of input within the gui.



Guicontrols.py includes some extra gui widgets created with audio design in mind. The test demonstrated a range of UI features, but the control demonstrated by the test case was not as intuitive as one would expect from a quality UI -- vertical sliders did not respond to the mouse as it was over the slider, but

when it was at the very edge of the display. Similarly, the circle did not respond when the mouse was over it, but as it moved around the full screen in a circular motion.

Midi.py takes input from a midi controller. Thus this has not been able to be thoroughly tested. It was able to be compiled properly. We have midi controller devices available to test input in JythonMusic.

Osc.py is a program that incorporates funcitonality for osc (open sound control) devices. It contains OSC input, output, and a listener. It connects to a specific ip address and port number which it will either listen for input on or send output to. The test sets up a connection and sends a simple “hello world”.

Timer.py is a program that will take a function and repeat it on a set basis. The unit test involved creating a timer that counts seconds. It only tested one of the timer classes present in the file. The test performed what was expected of it, counting seconds, but it did not incorporate a broad testing range.

Zipf.py is a program that is used to calculate the slope and R^2 value the trendline of a zipf distribution. It works by entering an array of numbers and it calculates the zipfian distribution. There are multiple phenomena that were tested including white noise, brown noise, pink noise, monotonous, and a general case. These had to be tested separately by uncommenting the intended array. The slope and R^2 values were printed without errors.