Oct 26, 2015 Assignment 2 Chazz Young
CMPT 365 D100 chazzy@sfu.ca
Prof. Mark. Drew 301209440

Design

My version of Assignment 2 (such as it is), was written in Java. It utilizes the ImageJ library (http://imagej.nih.gov/ij/), as well as the AVI_Reader class from the same site. The ImageJ library was chosen due to it's simplicity and power to easily read uncompressed .avi files, as well as separate it into frames. The AVI_Reader and ImageProcessor classes are most notably used in the project's main class, ImageToSound. A Tone class has also been included for a simple way to generate sounds. On top of this, a simple GUI was created to easily run the program.

Instructions

1) Extract to your preferred directory
2) Open a2-blund.jar
3) Click on the "Select AVI File" button to select the file to decompress. The pause and play buttons are disabled until you do.
4) If the file is read successfully, a small dialog box will appear. Ensure that the "Use Virtual Stack" and "Convert To Greyscale' buttons are checked, and press OK. **WARNING** There is no error checking here!
5) Hit "Play" to begin listening to the soudns

Known Bugs/Issues

1: The text area does not display the outpout volumes for the generated iamge as intended

2: The image

3: The generated sounds could not be outputted concurrently, creating serious lag issues as well as a column being sounded in 4 seconds, as opposed ot an image. Regrettably, I ran out of time in figuring this out with Java.

4: The ImageProcessor class appears to return non-standard integer pixel values (i.e. black has a color value of [240, 0, 0] from java.awt.Color). This will likely change the effects of output sound.

5: The play button currently calls the run() function, which freezes the app until the entire image stack is played, or the process is terminated externally

Description of Classes

1: ImageToSound – main class nearly entirely written by me with reference to the ImageJ API documentation. It contains an AVI_Reader object, and is responsible for generating the resulting ImageStack, literally a stack of images generated by AVI_Reader. It abstracts away the major details of dealing with AVI_Reader and only returns an AWT Image and an int[][] array (of grey intensities or volumes for the tones) for a given image/frame number, as well as the total number of frames (starting from 1).

2: Applet – This is the primary GUI class, mostly generated code from Netbeans's GUI builder. It's responsibility is to connect the ImageToSound and Tone classes, while also providing the GUI. This class is responsible for calculating the values of the frequencies, displaying the image for the current frame, and running the Tone sounds for that frame. File selection is done here, as well as play and pause functions (not 100% working)

3: Tone – Taken from https://community.oracle.com/thread/1273219?start=0&tstart=0 and simplified to only show the one method generateTone(). This method is responsible for playing the tones with the given parameters (frequency, length, volume, false) (4th parameter is a Boolean to add an octave, which isn't necessary).

4: AVI_Reader – Taken from http://rsb.info.nih.gov/ij/plugins/download/AVI_Reader.java. This class is used to read in uncompressed AVI files. Unfortunately, it is very limited, and AVIs had to be uncompressed using VirtualDub http://www.virtualdub.org/ and http://forum.videohelp.com/threads/289379-VirtualDub-being-stupid-(VFW-codec-)

5 – a2blind: Simple main class to start the applet.

6: toneThread() The last attempt to concurrently play the tones. Taken from http://www.javacodegeeks.com/2013/01/java-thread-pool-example-using-executors-and-threadpoolexecutor.html with no success

Proof of Excellence*

* works subject to known bugs

Language: Java

Video size; any video size, resized within program

Video type: Particular videos: only uncompressed AVI files. It works for individual images as well if the code is modified