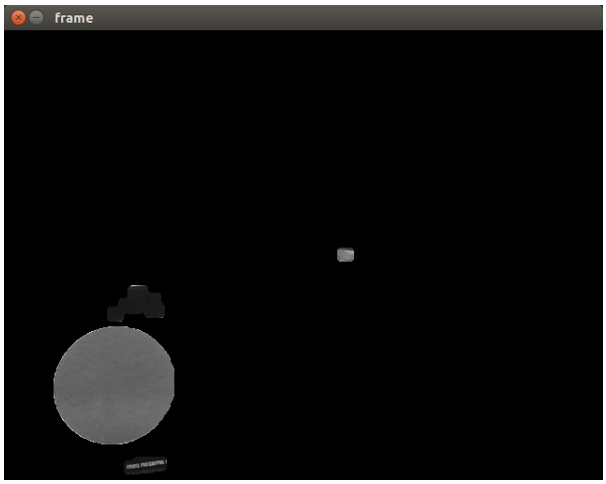
Rocoo DiVerdi
Manik Singh Sethi
Interactive Programming – Theremin

Project Overview – We have tried to create a Theremin device that tracks a purple ball and then determines the location of the ball on the screen in two axes. The horizontal axis is then used to change the frequency of the output sound and the vertical axis corresponds to volume of the output sound. Different modules like OpenCV and Nsound were used to create the project in two parts – a video tracking part and an audio output part.

Results: The final result is a functioning Theremin like device that responds to the user's use of placement of a purple ball. It is able to distinctly increase the frequency of the sound produced (pitch) as the ball moves left in view of the web cam and it is able to distinctly able to increase the volume of the produced sound as the ball moves up in view of the web cam. This was the intended functionality of the program. The ball is clearly tracked by the web cam based on the color purple and then the output is changed in real time until the program is quit. It is possible to create interesting musical outputs if the user tries various experiments with the ball, like juggling three simultaneously because there is movement is the x axis and the y axis.



The best way to really see the resutlts is to actually try the program and hear the Theremin program in use. The image above shows the screen the user will see when the program is active. The Circle on the bottom is an example of a purple circle image found on the Internet. This shows that there is not much interference from the surroundings but some small amount of shading and purples should be expected to show up without compromising the performance greatly. When the program is running, the sound clip is continuously played so there is no continuous hum of the tone, rather, the sound keeps playing one file after the other but the real time reaction of the program is not compromised.

Implementation: There is no inheritance in the project for there is no interesting UML diagram. However, there are two classes that can be explained a little more closely. Most importantly for ball tracking, there is a ball object which has color limitations, an x position and a y position. The second object in the program is a note class that maintains the playback and usage of sound output. That object has a pitch, and a volume specifically. The controller would be the manipulations of the ball by the user model would be considered to be the positions of the ball(s) in the system, the view would be the output sound. The reason a purple ball is chosen is to simplify the object tracking system to just look for the color purple (chosen arbitrarily) and the circle shape of a ball is simpler to average to find a

singular position. Another major decision taken was to create files of the different tones beforehand and only access those from the program, played at specified volumes. This is to save time. In the original case, the program would write the sound file and then play it every time it received input.

The first major progress for analysis from object tracking is to isolate the color purple and that was done by a combination of eroding, dilating, and blurring the image. Then, the center of the shape needs to be found. Previously, the program would try to get the center of as many circles it could draw in the shape which cause significant amounts of lag. The program was improved to use Hough Circles to create a weighted average to safe time and be more efficient. This is primarily done using the OpenCV package. That position is then given to the note object that uses the x position to call a file containing a corresponding pitch and the system's playback volume is adjusted by the y position. The sound work is mainly powered by the NSound and Pygame packages.

Reflection: In the barebones of the project, it was quite manageable but the usability and sensitivities of using video and audio manipulations created difficulties for both partners of the project so it was well suited for the both of us. We had deemed success as a system that could take in user input based on video and output varied pitch and volume. This was achieved through much work and integration of efforts. Until we both did the OpenCV and sound toolboxes, it was hard to head forward but both provided great bases for customization and specialization. Specifically for the audio portion, it was quite difficult to get a start because there are so many packages for making and playing audio but finding one best for this application was not clear until the toolox was completed. The approach to pair programming we took was interesting, with Rocco taking on the video manipulation and Manik taking on the audio portion. Toward the very end of the project, the two components were put together. However, throughout the process, we both conferred on progress and what we were learning and how the model could be adjusted to reflect those changes. That seemed to be a useful aspect of our teamwork.

We did not get to achieve our stretch goals, which were to graph out the data inputs over a single run of the program and to track a hand instead of a simple object. We learned about the patients and nuances that can often change the effectiveness of a program that depends entirely on video and audio inputs/outputs. Next time, it may be better to get more time together and work simultaneously on single parts. Still, division of labor was useful for us.