Final Project: An AudioVisual Player Documentation

Orsolya Szantho

For this project, my main interest and objective was to create some sort of an audiovisual player using Open Frameworks. My core inspiration behind this basic idea is that I work in live performance settings from time to time, creating visuals for musicians or theater pieces, and I have always been wanting to start bringing custom-coded live-reactive visualization into that practice.

In terms of specific visuals, I was hoping to explore meshes and wave/trigonometry based abstract animation, but I had no specific goal. I wanted to say open and look at several examples instead, mostly because I felt like I had to work some more on my general awareness of OF (even some basics).

I originally hoped to work with a kinect (as I have one and its potential seemed to match my interests in movement-based performance as well) to track a mover and generate sounds and visuals based on that input. Unfortunately, after quite a bit of struggling, I had to give up on this hope as I discovered that the device doesn't really want to connect reliably to my current computer, and it was very challenging for me to have consistent access to an alternative machine.

Instead, I decided to look into OF's OpenCV example and build a basic motion tracker -- once I figured out how the blob detection worked, I decided to just average blob positions to get a general sense of movement in the frame (Figure 1 - also Video 1).

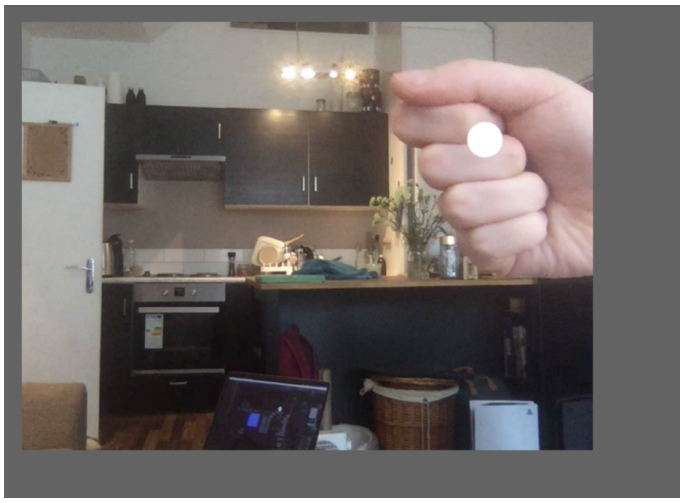

Figure 1: Basic movement tracking

Once this was working, I got rid of the camera image display, and just kept my white circle against a black background, a base visual for now. As I was feeling like the output remained quite jittery despite my efforts to ease position change, and that overall the movement tracking was okay but not extremely interesting, I decided to think of an additional input for the project. To stick with ideas that could help me build performance material in the future, I settled on using an audio input -- as I have achieved nice visual results using similar controls in pre-built software before, and I could see many possible applications. I then succeeded in making my circle react to incoming volume by changing its size (Figure 2 - also Video 2).

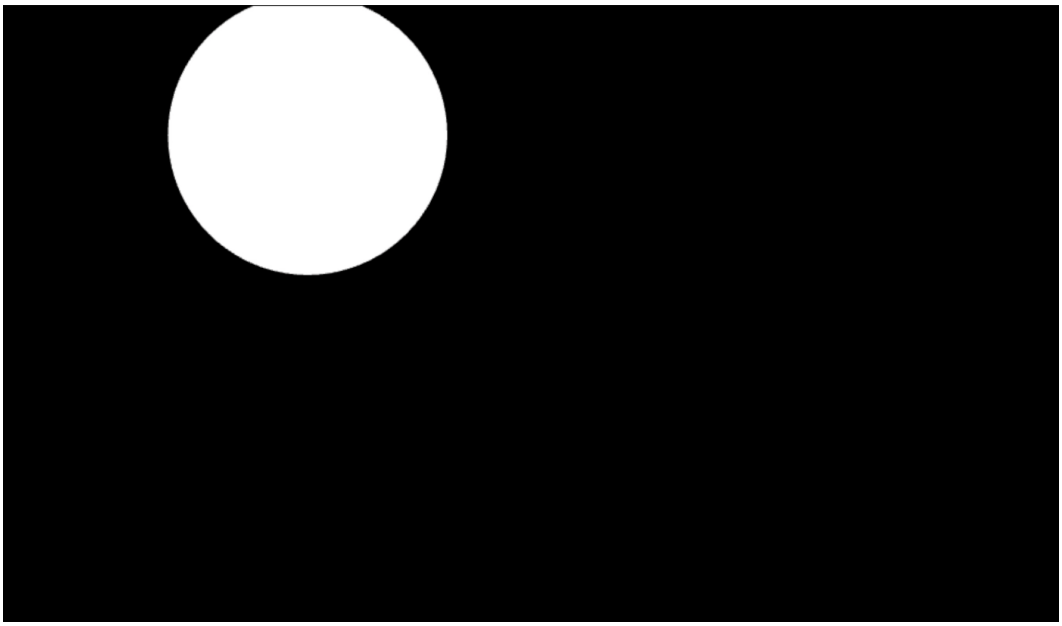

Figure 2: The circle grows larger in response to loud sounds

Now it was time to move on to creating some more interesting looks. I really liked the flower-effect we started exploring even in JavaScript last term, so I adjusted the OF audio output example - which by default shows the played audio in two horizontal waves. I took the L and R channels the example was using and created two different sin-based patterns that react to the mouse movements along with sound, to visualize the sound output. I then merged the two projects to have both the audio output and visualization and the audio input reactivity together (Figure 3 - also Video 3). This project had the white shape react to sound input from the mic as well as the camera's tracking, and the two pink/blue shapes react to the mouse, which also controlled L/R audio channels and the output sound's pitch.

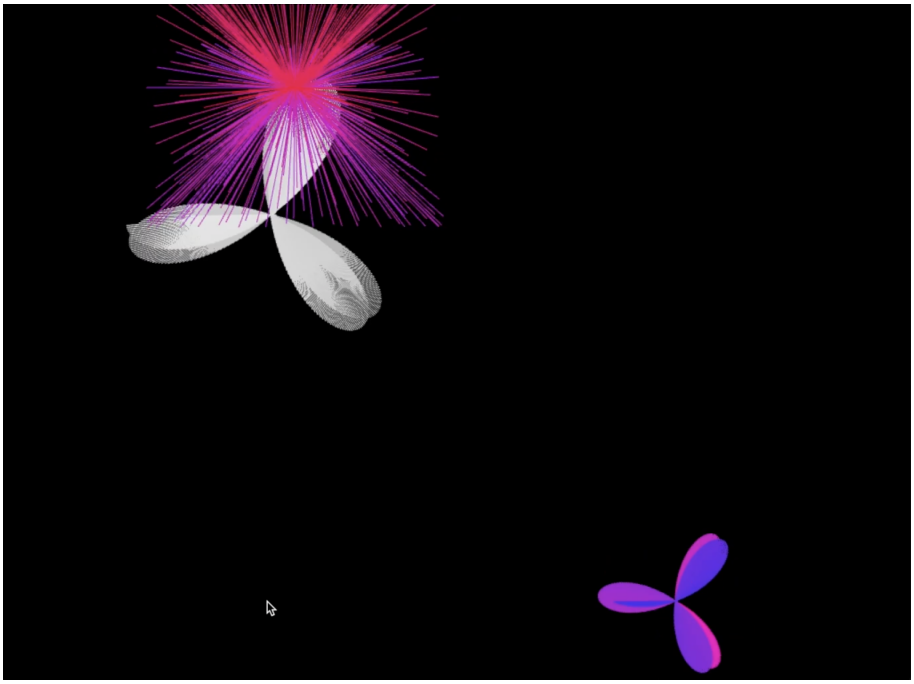

Figure 2: The circle grows larger in response to loud sounds

While this interaction was enjoyable in a way -it's definitely fun(ny) to be singing/whistling along with the computer's very digital tones- I felt like I wanted to go in a more pleasant-sounding direction, and perhaps also bring some more visual options in. I looked into playing tones and some more OF ideas and found videos and examples online that helped me understand the system a bit better and I decided to build in a tone player. I created some "magical" sounds using choir tones from an example player in the folder linked above, and added some wind chimes and harmonies to have my own samples sound more charming and playful.

I also wanted to make sure to visualize the sound player, but I didn't want it to look too much like a classic visualizer (eg bars around a circle or a soundwave or such) so I created a visualizer that draws circles along a really squashed wave shape with varying sizes based on OF frequency analysis (the volume of each frequency group). I was really pleased with how the resulting pattern almost looks like a mix between falling rain and twinkling lights.

An additional aspect of OF I was interested in that I didn't feel like I had had a chance to look at was meshes and distortion, so I also looked into those OF examples, to build a palette of different looks or moments for this piece. I looked for some noise examples (and then noticed the actual OF example of mesh distortion), which I found to be quite nice, so for one "look" of the project I adjusted this only slightly. That said, I wanted to also see if I could use the camera image as a distortion to create an almost 3d-tracked-effect -of say, a dancer in a performance- since I already was using some camera interaction. I managed to get this working quite easily and quite well, and I was mostly happy with the effect but I had two issues I couldn't resolve. One, I couldn't figure out a smooth transition between the two distortion maps, which was also stemming from: two, I couldn't

figure out a way to effectively pre-smooth the camera image output enough to create a smooth distortion mesh from it -- so I had to keep that with the vertices being a bit more noisy and creating a sharper mesh.

The final step was to bring these bits all together. In this phase of the project, I aimed to clean up my code a bunch and introduce some functions for drawing different elements, which I also wanted to be controllable using different letter keys on the keyboard (since the sounds were already being played using numbers on there). I luckily did not face as many issues with this step as I had anticipated, really the majority of my troubles were with the non-functional kinect, the camera's mapping, graphics mapping, and making the flower shapes using a wave generation without the sound output being audible. Below are some screenshots of the final project, which you can also see in Video 4.