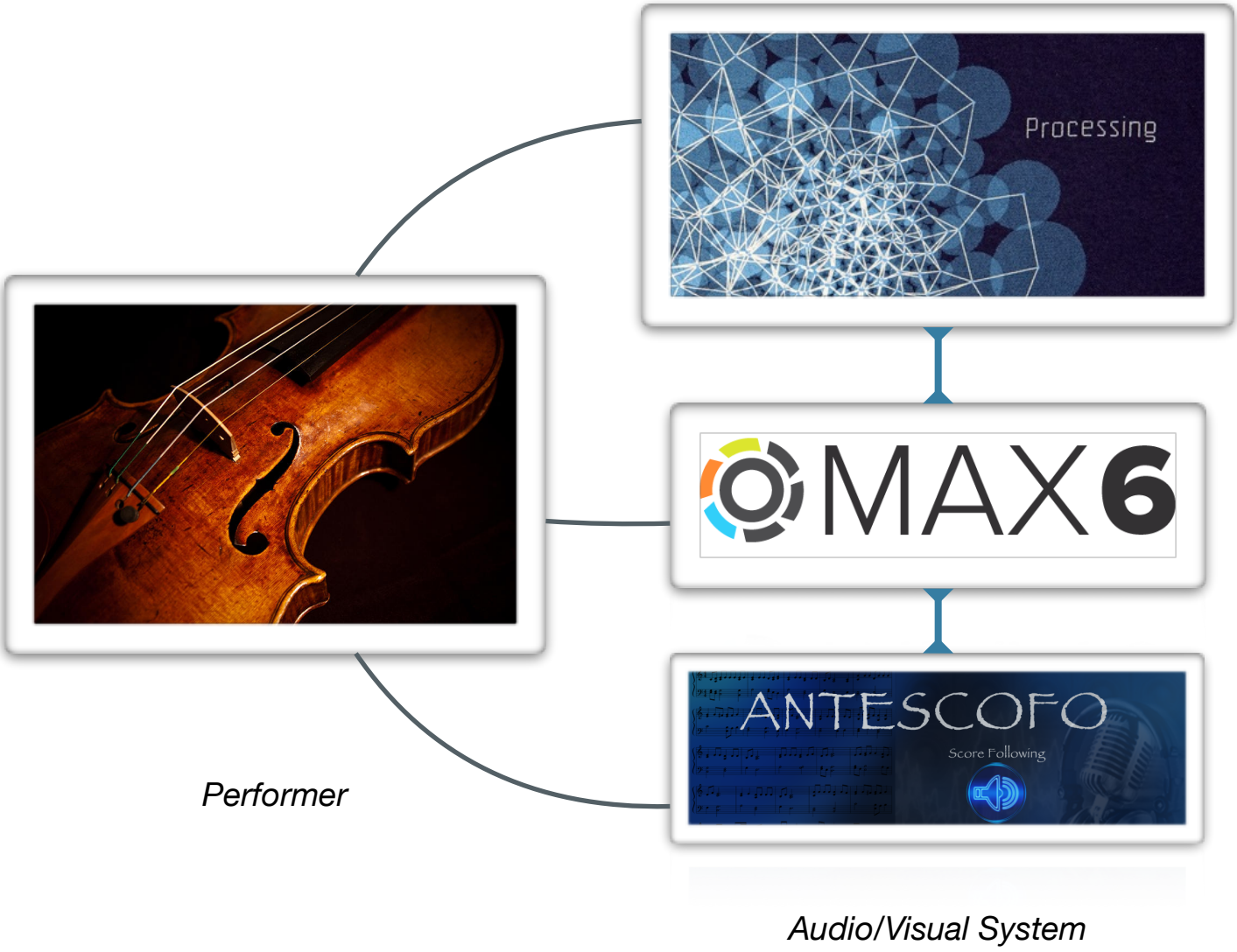
KEVIN GUTOWSKI



An Adaptive Audio-Visual Bridge

For MaxMSP and Processing

Prepared for: Wessel, David
May 16, 2014

KEVIN GUTOWSKI

THE VIOLIN: AN AUDIO-VISUAL ADVENTURE

Objective

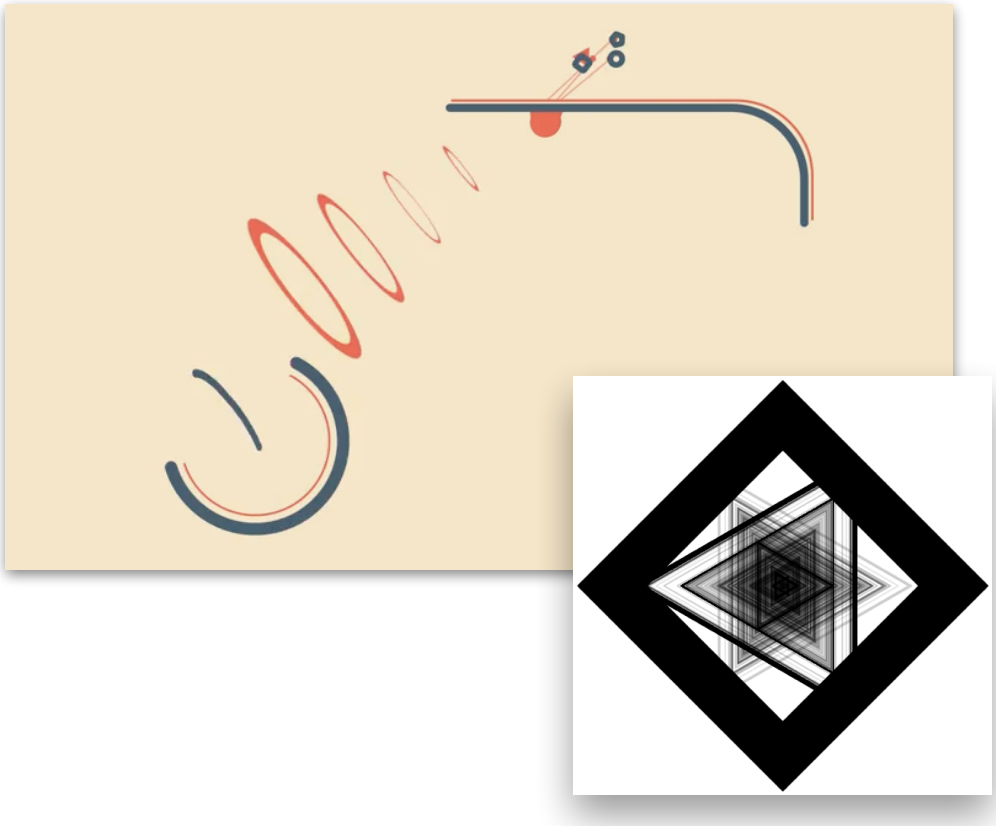
The main function of this project is to utilize audio-visual technologies in creative ways. Many of these applications are great by themselves however integrating them can create new potentials for ways we can interact and perform with various media.

Goals

To provide a flexible way to perform a piece and have audio and visuals accompany a violinist.

Inspiration

Visual Inspiration-



For my visuals I was heavily inspired by *A Ball and a Stick* by Oscar Pettersson (https://vimeo.com/86291629) and *Mork* by Phil Borst (https://vimeo.com/40006163) which utilized simple graphics with smooth animations. I wanted

to stick to simple design practices so that the animation could be easy enough to execute and flexible enough for Antescofo. Some other notables for inspiration were Will Fortanbary's *Motion Graphics Fun 3!* (https://vimeo.com/88283748) and audio-visual pieces from KNOWER (https://www.youtube.com/watch?v=itRWssMSuvw).

Audio Inspiration-

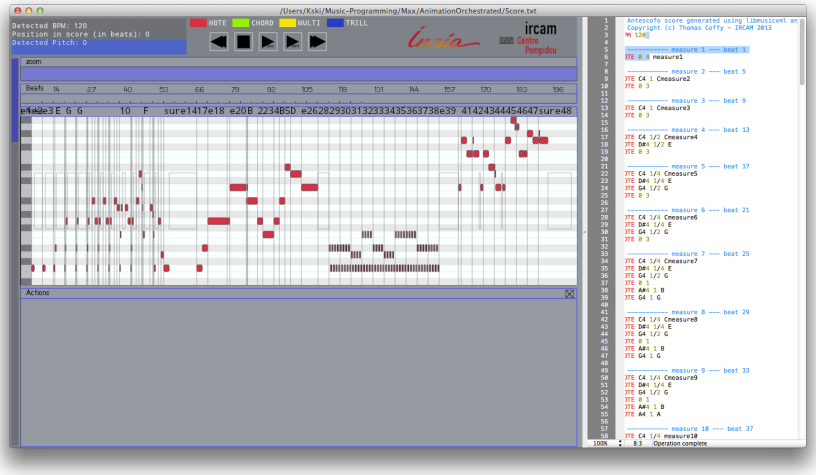
Wiley Webb has been a huge inspiration due to his design of interesting electronic music. He utilizes tension and release with interesting textures and audio clips to make for a captivating sound. My favorite use of these techniques are in the first few minutes of his fourth piece titled *Sensual Art* (https://soundcloud.com/wileywebb/eclecticism4).

Another artist that has inspired me has been Keith Kenniff. You would know his work from many different ads ranging from ads for Instagram, Apple, and Facebook. He also does work for film, television, dance, and performance art. I find that his piano motifs with layered strings or synths gives his music a "homey" feeling and a sense of closeness. I always find his tunes inspiring and refreshing. One of my favorite tracks is *To Be* (https://soundcloud.com/keithkenniff/to-be) and it subtly reminds me of *Nemo Egg* by Thomas Newman from *Finding Nemo*.

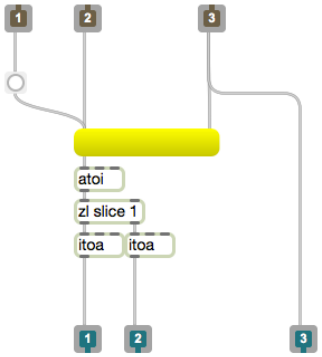
Project Outline

Performers looking to add visuals and audio effects to their shows often resort to either hiring other engineers to help them accomplish this live (which usually makes for a more loose audio-visual performance) or having to stay on a click track which events can occur at specific moments in time. My solution to this is to develop a piece of software that utilizes Antescofo such that the performers no longer need to stay adhered to a click track. The performers can slow down or speed up (and even improvise if programmed) and the accompanying audio and visuals will still play on cue. This idea revolves around two main technologies and one other for the visualizations, which are Max MSP, Antescofo, and Processing.

Code Analysis



Starting in Max, I have the ADC hooked up to Antescofo. This takes the violin's audio and maps it to the position within the score. Once the score of the song is loaded, Antescofo can now follow along with whatever piece of music you feed it. I then go in and edit the Antescofo code to output particular IDs of what note is supposed to be played. I don't examine what note is being played because sometimes that note is slightly off what is supposed to be played and the animation that is mapped to the



correct note will not play. Antescofo has some nice features for practicing with it and I highly recommend creating the correct message boxes above antescofo when trying to get the score following and animations just right.

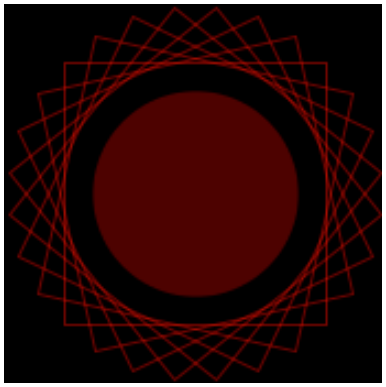
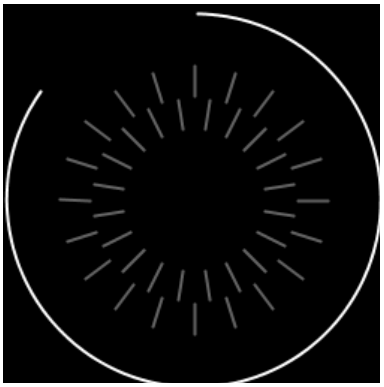
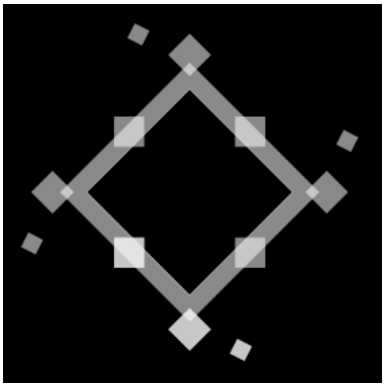
Once outside of Antescofo the measure data and note data are paired together so I parse them out by converting the text to numbers and then back to letters so that I can do a basic slice of the note and the measure. I am not particularly convinced that this is the best method to pass both note data and measure data but it works alright. I then create the note bindings and map them to different "keys." I chose keys because I was originally testing with number 1, 2, 3, etc. and didn't want to change the code over in processing so the convention stuck. Now that Processing can get our note data we can do animations but before I talk about the processing side of things I want to talk about how I do my additional audio.

For this build I decided to start audio on the down beat of particular measures. It is possible to start them on whatever note you like however this was the convention that I chose. I output the measure data from Antescofo and then convert it to an OSC packet. Afterwards, I define what measures I want audio to begin and definite the switches accordingly. I then map these up to **sfplay~** objects to keep things easy. Just like that, we have additional audio effects mapped to particular events in the score.

Now that our data is all being sent over **udpsend** we can easily parse and map this data to trigger our animations. We define the globals that we would like to be affected and include them in the animation functions. To troubleshoot each animation I individually crafted each animation in a different sketch and then brought them all together into one. One difficultly I had when doing this was making sure global variables were correct. It's easy to use matching globals for two different animations. Another difficultly that I ran into was implementing physics within the Processing environment. Either it was two processing intensive (probably due to my coding skills) or it was much more involved to implement. I settled for an animation library called Ani (http://www.looksgood.de/libraries/Ani/). It allows me to easily move things around the screen and define different easing to get from point A to point B or phrase an animation. You can find all the curves that are available at (http://www.looksgood.de/libraries/Ani/Ani_Cheat_Sheet.pdf).

To summarize, I used external libraries Ani and OscP5, initialized them properly, and created globals for which variables I wanted affected. My main hurdle was getting all the code for each animation to mesh together. It's quite simple to set this up for one animation phrase but much more laborious to get five together. I am sure there is a better way to implement this than what I have done, however that will be left for an exercise for the future. Below, I will include a couple screenshots of the animations that I was able to create.

Last but not least was the composition. I did the composition in Garageband due to the fact that I could iterate really quickly and it had some nice synth samples. I really wish I had a composer on board to do this part however finding one with the a parallel idea of a electronic related score proved to be more difficult than I had realized.

Execution, Implementation, & Reflection

Overall I think this was a good first attempt at trying to integrate these three technologies. Considering that amount of programming I knew at the beginning of the semester and now, I think I have accomplished a lot during that time. While I couldn't exactly get my code to work in a live setting (antescofo was a little bit more difficult to work with than I first anticipated), I learned a lot about the development of an AV system. Here the flexibility in hooking up whatever data you want is freeing to the programmer but limits the availability to just programmers. If my target audience are musicians than I should refine, skin, and narrow down the openness of the platform to make it easier to see an end product that they can use or edit. I think the idea of using antescofo to direct audio and visual control over a production is very doable however at this moment the product needs to be developed more to include redundancies and easier access to animations and mappings from a non-programmer standpoint.

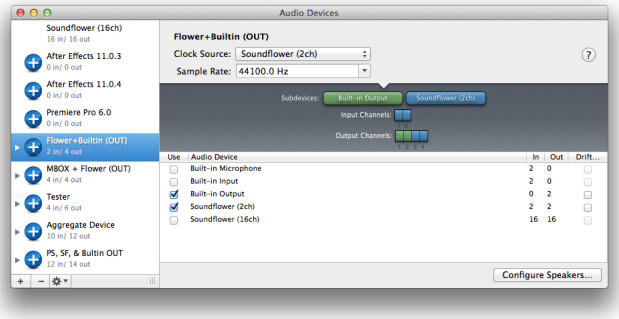
Side Projects

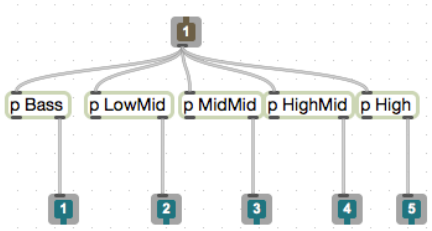
Throughout the development process I took many detours that led me to a finer analysis of a A/V system. This was in part due to the Multimedia Orchestra and wanting to make a presentation at their concert.

Live Settings, Past and Present Information -

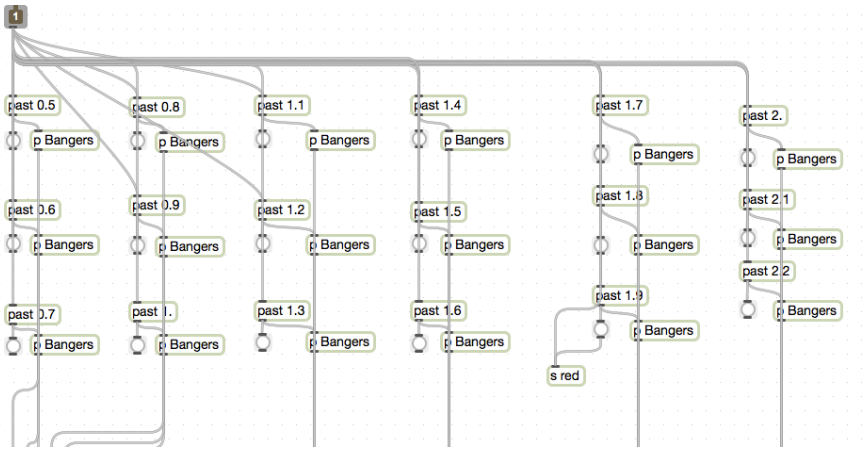
One large deviation that I took was an exploration of sending some audio signal and have some visuals be reactive to that audio. In other words, I explored the basics of visualizers. I often DJ for the Multimedia Orchestra and I wanted some interesting art the reacted to the audio output of Traktor. I use Soundflower to build an Aggregate Device which I can then route my audio from Traktor over to Max to do beat detection and audio analysis.

Now over in Max I first take the incoming audio and split it into 5 different audio regimes; Bass, LowMid, MidMid, HighMid, and High. I **pak** this data together and send it out over **udpsend** so that the visualizer can utilize this data (in this case processing). In parallel I use **peakamp~** to report the peak amplitude. I then smooth this data out and multiply the smoothed data out with the original signal. By doing this I'm able to retain the sudden changes in the stream while getting at the overall tendencies in the stream. After this I do some thresholding on the incoming





stream and parse this out into parts where I try to detect “drops” in the music which I classify as moments after a build and after a bit of silence or very little sound followed by a big beat. I then **pak** all this data and send it out over **udpsend**. In another part of the patch I also do thresholding in parallel and can get some nice characteristics of the incoming stream by just looking at where the threshold objects are triggered and how many are triggered at once. This data is also mapped to OSC variables and send out over **udpsend**. All this data gets sent out over to Processing for data visualizations.



Audio Stream Visualizations -

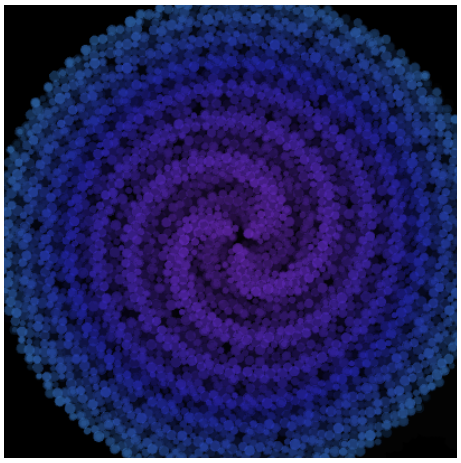
There are many solutions to creating visuals for audio productions. Too often do I see in mainstream music that these productions are either performed by artists or not really audio reactive. For example. at EDC the lights and animations used were part done by engineers for the drops however outside of that were animations that sold the effect of sound responsiveness. Sure, they pulsed and zoomed around the screen but they had no regard to the beat or build of the piece. Anyways,

```
void oscEvent(OscMessage m) {
  println(m.addrPattern());
  println(m.arguments());
  synchronized(this) {
    if(m.checkAddrPattern("/bass")) {
      bass = (m.get(0).floatValue())*5;
      }
    if(m.checkAddrPattern("/lowmid")) {
      lowmid = (m.get(0).floatValue())*10;
```

now that we have all this data about the incoming stream we can map it different functions within our processing sketches. My first attempt was to just get the basics down and how to properly do the mappings. First the external library oscP5 needed to be initialized properly and variables that are desired to be mapped need to be turned into global variables. Lastly, the oscEvent function needs to be properly called and **m.checkAddrPattern()** is utilized to pull the OSC data from the udp stream. These are then assigned to your globals which then manipulate the sketch. I then created a simple target that would take the mappings of the "/bass", "/lowmid", etc. and change the size of the circles. By choosing a pleasing color palate I created a simple, yet effective visualizer that shows information about the incoming stream and is reactive to the big changes in amplitude such as drops.

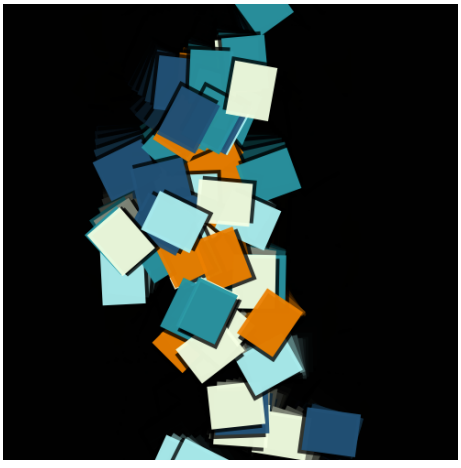
For my next visualization, I studied how plants grew and came across information about the golden ratio. With this knowledge I wanted to create something that pulsed while being built. At first this was a struggle but then I realized the power of altering the index of a for loop as time progressed. As for generating the rest of the flower, it only took a few more lines of code to draw each step within the flower. Here, **nprime** changes with time and allows the flower to build out. **r** is the distance from one circle out to the next, **radius** defines the average size of

```
for (int n = 1; n < nprime*3; n++) {;
  float r = c*sqrt(n);
  float radius = 9;
  float theta = n*PI*(3-sqrt(5));
  fill(62,map(r/2,1,width,0,200),138,100);
  float pulse = pow(sin(t*PI/3-n*PI/(t%100)),1.5);
  pushMatrix();
  ellipse(r*cos(theta)/4,r*sin(theta)/4,pulse*radius,pulse*radius);
  popMatrix();
```



the circles that make up the flower, **theta** is the golden ratio, **fill** is just coloring the circles based off their distance from the center. **pulse** is an effect that propagates through the flower, **pushMatrix() and popMatrix()** are redundant in this instance and are leftover from earlier versions of code. **ellipse()** then draws out all the ellipses that make up the flower. In my sound reactive version I make many of these global variables which are manipulated by the incoming stream.

For my last live animation, I wanted to make something a little more different. After watching a tutorial from Fun Programming (http://funprogramming.org) I was inspired to utilize **noise()** and create something a bit more randomized. I then created a bunch of squares on screen that utilized 2D noise() for their moment and utilized the incoming stream for their size. Then I had them rotate after a specified time and change into ellipses after two rotations. I also added a dark square behind each one to add a drop shadow and give the piece depth.



RESOURCES

Books-

Adobe Creative Team. Adobe After Effects CS6 classroom in a Book : The Official training workbook from Adobe Systems.

Adobe Creative Team. Adobe illustrator CS6: Classroom in a Book - The Official Training Workbook from Adobe Systems.

Bohnacker, Hartmut. Generative Design: Visualize, Program, and Create with Processing.

Botello, Chris. Advanced Adobe Photoshop CS6 Revealed.

Brie, Gyncild. Adobe Photoshop CS6: Classroom in a Book.

Chavez, Conrad. Adobe Creative Suite 6 Design & Web Premium Classroom in a Book: The Official Training Workbook.

Cipriani, Alessandro. Electronic Music and Sounds Design: Theory and Practice with Max MSP.

Foster, Jeff. After Effects and Photoshop: Animation and Production Effects for DV and Film.

Glassner, Andrew. Processing for Visual Artists: How to Create Expressive Images and Interactive Art.

Greenberg, Ira. Processing: Creative Coding and Generative Art in Processing 2.

Pearson, Matt. Generative Art: A Practical Guide To Using Processing.

Reas, Casey. A Programming Handbook for Visual Designers and Artists.

Shiffman, Daniel. The Nature of Code. (http://natureofcode.com)

Williams, Richard. The Animator's Survival Kit: A Manual of Methods, Principles, and Formulas for Classical, Computer, Games, Stop Motion, and Internet Animators.

Links-

Abduzeedo Staff, *Abduzeedo* - http://abduzeedo.com

aBe, *Fun Programming* - http://funprogramming.org

Borst, Phil . *Mork* - https://vimeo.com/40006163

Herr, Jerome. *p5art* - http://p5art.tumblr.com & http://www.openprocessing.org/user/28663

Kenniff, Keith. *To Be* - https://soundcloud.com/keithkenniff/to-be

KNOWER, *Time Traveler* - https://www.youtube.com/watch?v=itRWssMSuvw

KNOWER, *Daft Punk-Get Lucky Cover (Electronic vs. Live Instruments)* - https://www.youtube.com/watch?v=itRWssMSuvw

Pettersson, Oscar. *A Ball & A Stick* - https://vimeo.com/86291629

Tarakajian, Sam. Delicious Max MSP Tutorials - https://www.youtube.com/channel/UCHen0AQzqOMaPbpjYBhfsTA

Webb, Wiley. *Eclecticism v4: Sensual Art* - https://soundcloud.com/wileywebb/eclecticism4