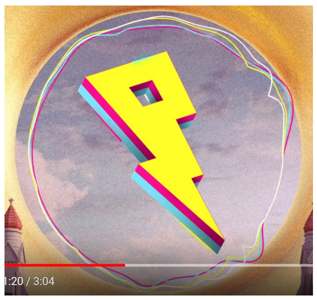
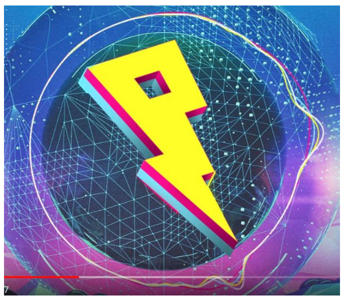
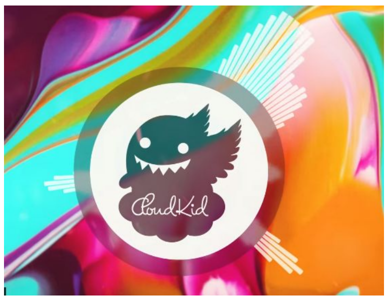
# JAW dropping Visual Effects via Audio Spectral Analysis

## 1. What is it?

***Overview***

This project is an audio based visual effect generator. By extracting frequencies and other features from an input audio, effects which alter the input image's color, shape, and position will be generated and superimposed onto the input image to create intriguing visuals. The inspiration comes from music videos that create spectrum analyzers wrapped in a circle through mapping frequencies from an input audio source. The images below demonstrate examples of the general effects we hope to implement. In order to break our projects into manageable chunks, we have divided the goals of the project into three levels (basic, expected, and stretch).



***Basic***

Our basic goals demonstrate a simple proof of concept. In the basic implementation, the input audio source will be from a short audio recording or from a cell phone's audio data stream. FFT and appropriate filtering will be applied onto the audio source to generate the appropriate number of frequency bins.

The input image will be a simple, close contour, high contrast, polygon image from a file. The contours of the image, found through edge detection within MatLab, will be pushed onto the FPGA. The FPGA will treat the incoming contours as a combination of sprites, with each sprite representing a section of the contour which maps to the appropriate frequency bin.

With the sprites and processed audio input, one or two transformations can be mapped. Several possible transformations include: a color change transformation to a segment whenever the average intensity of the associated frequency bin surpasses a threshold, movement of the entire sprite with a variable speed based on the intensity of the corresponding frequency bin, or manipulating the edge contours of the image in response to the corresponding frequency bin (as in the frequency above).

### *Expected*

Our expected goals build upon the basic goals. There will be an improved audio filtering, and the ability to extract or emphasize specific frequencies, such as the bass, will be added. The input image can be captured through a NTSC camera, and the inputted images can have a higher complexity than a simple polygon. Additionally, the image edge detection will be implemented in Verilog, as opposed to MatLab in the basic implementation. The image modulation will also have higher resolution (we plan to approximate curves using lines, so the number of lines used to approximate the same curve would increase)

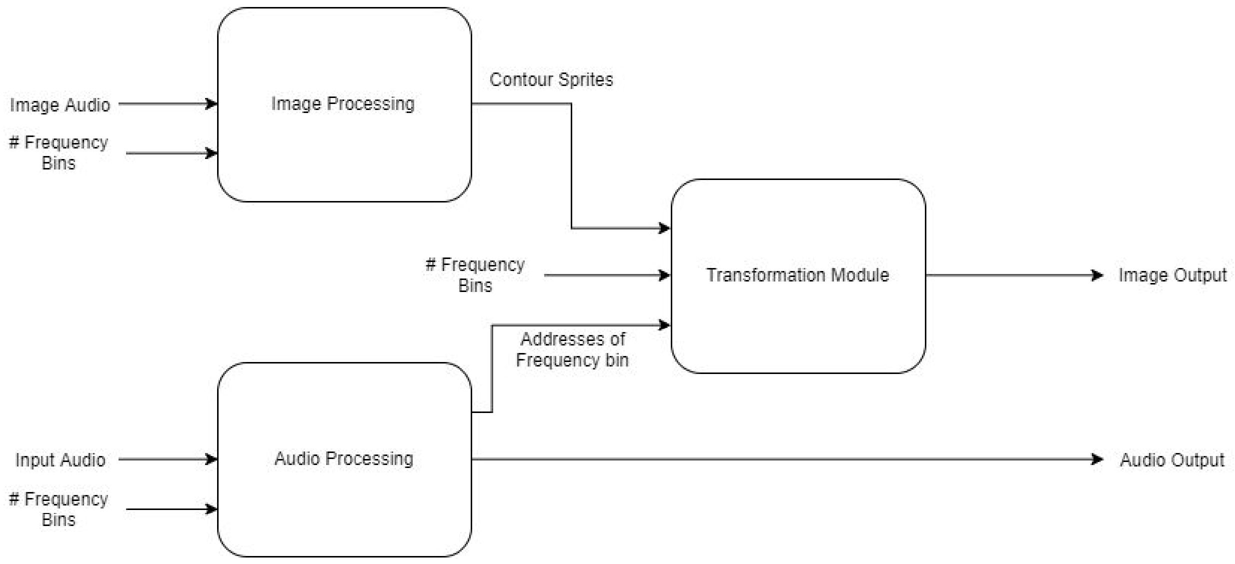
### *Stretch*

Our stretch goal is to incorporate all or most of the above features in live/real time. The visual effects will react to live audio sampled via a microphone, and the inputted image would be a stream of frames from the NTSC camera.

## 2. How will it work?

There are three major workstreams that can be completed concurrently. The audio input needs to be retrieved and processed. The image input needs to be retrieved and processed, and finally, a suitable mapping of transformations must be designed and implemented to the image before outputting the image on a monitor. Aaron will work on audio retrieval and processing. Wings will handle preliminary image processing and edge detection. Julian will use the processed audio and image to implement image transformations based on the filtered audio input mappings.

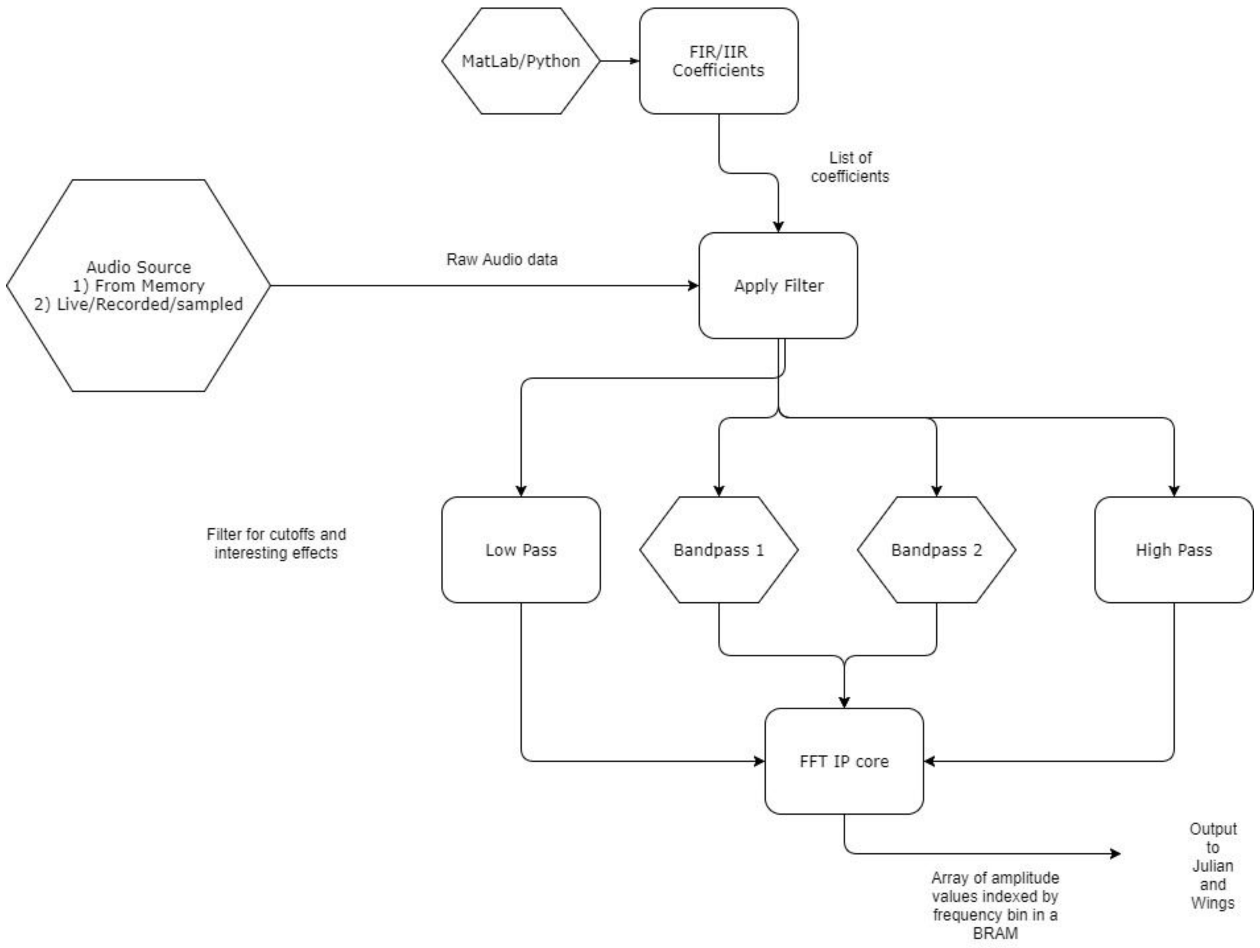
***High Level Block Diagram***



***Audio: Aaron***

The audio portion will be a three step process: retrieve the audio data, apply the necessary filters, and apply an FFT. Retrieving the audio data will be accomplished one of two ways: either recording/loading the audio data onto the FPGA memory or using a microphone and live sampling an external audio source (music being played, talking, etc). After the the audio data is accessed by the FPGA, Aaron will be able to use the FPGA to implement a multitude of low pass, bandpass, and high pass filters to make the mapping of frequencies to transformations as easy as possible. The coefficients for implementing an FIR filters will be generated in Matlab/Python and the structure and complexity  will be similar to that of lab 5 except with different cutoff frequencies designed to give smooth transitions between bins or to generate interesting frequency effects. Finally, applying an FFT using the IP core will be necessary as the input to the transformations and mappings module that Julian will design and implement. The external hardware required for this portion is limited to microphones and an audio source (cellphone, laptop, voice, etc). Testing this module will involve making sure the input audio regardless of the source being either sampled live audio or stored audio in memory can be reproducibly filtered and transformed using the FFT IP core. The output of the FFT should be able to be graphed and make sense (spikes at the appropriate frequencies). The internal memory BRAMs or ZBTs will be used to store the audio and

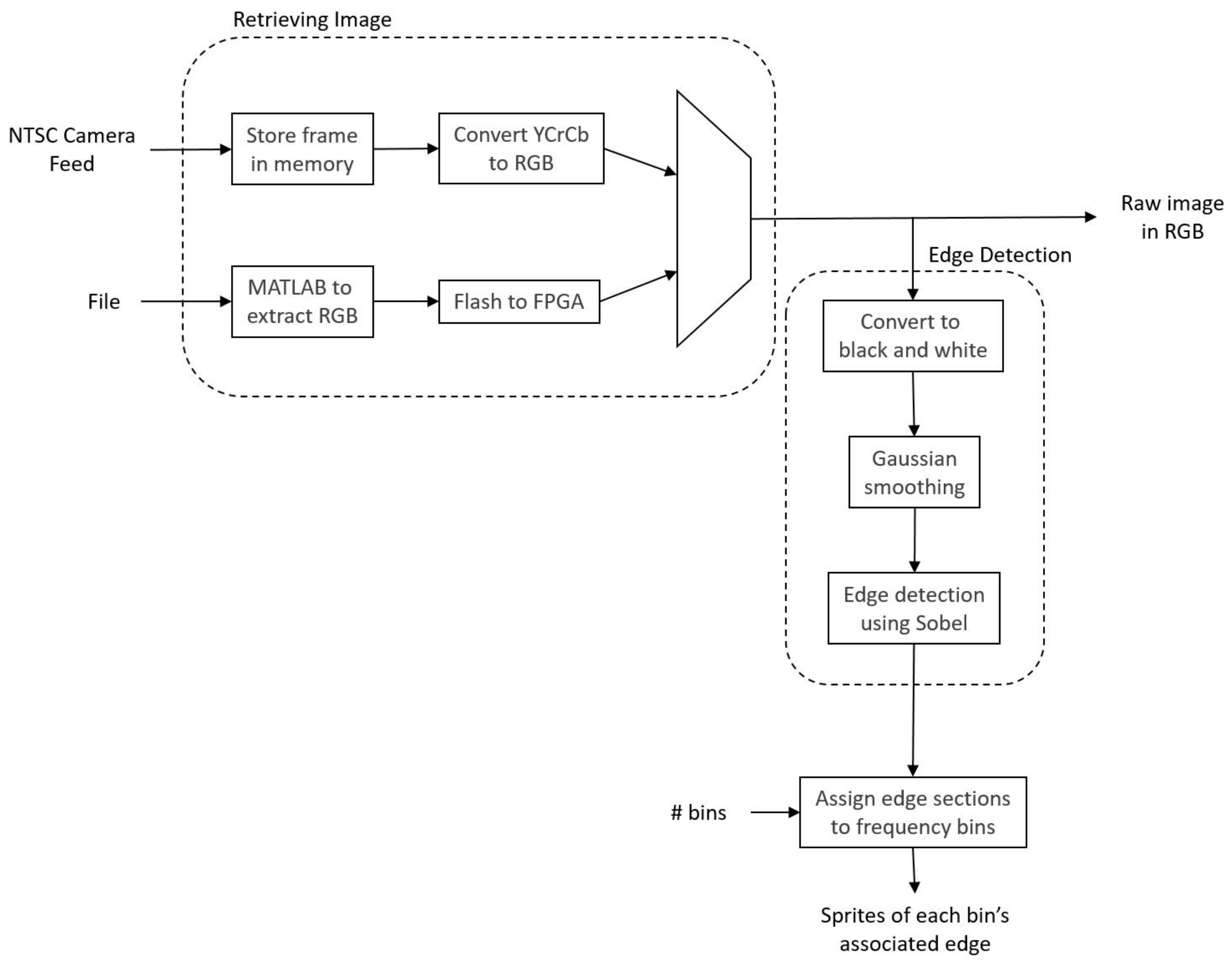
FFT output indexing the FFT output by the number of frequency bins to allow for easy access by other modules.


MatLab/Python
FIR/IIR Coefficients
List of coefficients
Audio Source 1) From Memory 2) Live/Recorded/sampled
Raw Audio data
Apply Filter
Filter for cutoffs and interesting effects
Low Pass
Bandpass 1
Bandpass 2
High Pass
FFT IP core
Array of amplitude values indexed by frequency bin in a BRAM
Output to Julian and Wings

***Edge Detection & Image Processing: Wings***

Retrieving the image data can be done in two ways: through a NTSC Camera feed or through a file flashed onto the FPGA. The raw image will then go through several different modules in order to extract the edges. The image should first be converted from RGB to black and white, then processed through a Gaussian smoothing filter, and lastly the edge can be extracted using the Sobel edge detection algorithm. The raw image will also be outputted to the Transformation stage of the project.

The next step is to the match the frequency bins to sections of the edge. In this module, sprites corresponding to one edge section representing a single frequency bin will be outputted to the Transformation phase of the project.



### ***Transformation Design and Implementation: Julian***

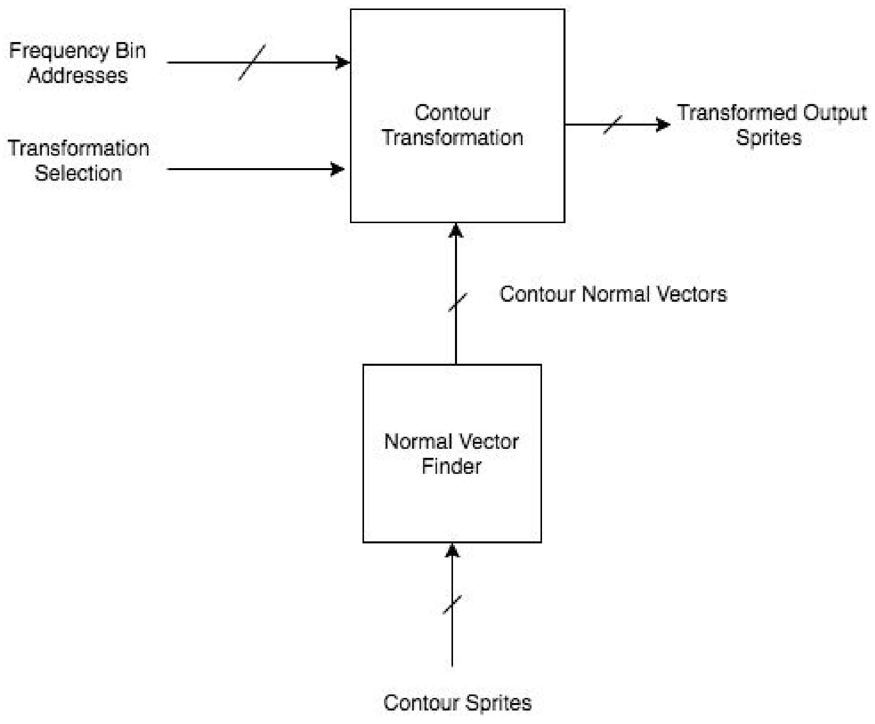
The transformation on the image will consist of two main steps. First is to find normal vectors to the different sections of the image, which each map to different frequency bins. Second is to output image pixels according to the direction of the normal vector, and the amplitude values in the frequency bins.

The different sections of the image are referred to as contour sprites, as they are considered individual portions of the image to be separately modulated according to the different frequency bins. The need for the normal vectors to these contour sprites is so that we can map amplitude values (in the frequency bins) to modulated distance along the normal to the original sprites.

Each output sprite will have an associated array indicating which pixels are set so as to reflect the current state of the sprite. The array will be updated by the contour transformation block, according to the process above. The output sprite will reflect a transformed version of the original contour sprite at every frame.

On each frame, the pixels of all output sprites, and the original image will be ORed together, following a scheme similar to Lab 3 for generating the VGA display output.

Different transformations will be selected by the user, and the output sprites will be modulated according to this choice. Basic transformations will be implemented first. Once those are working in an integrated system, we will consider more complex transformations.



## 3. When will it get done?

Week of October 29th:

- Develop project outline, divide up tasks, research feasibility of different goals

Week of November 5th:

- Aaron - Learn how to interface with FFT and clock wizard IP core, decide on frequency bins set up, determine memory size and type; implement
- Wings - Finish edge detection with MatLab and flash processed image onto the FPGA. Segment the contour and assign the appropriate frequency bins to each section.
- Julian - Consult staff on Normal Vector Finder, write output sprite modules that are able to be updated.

Week of November 12th:

- Aaron - Figure out how to sample appropriately for live audio source, start working on filters for frequency effects
- Wings - Work on input image using the NTSC. Start implementing edge detection using verilog.
- Julian - Work on skeleton module for Contour Transformation, and connect with output sprite modules to implement simple transformation (color gradient) on pre-defined contour, outputting on simple background.

Week of November 19th:

- Aaron - Finalize filter modules for frequency effects, and integrate with FFT and audio data retrieval
- Wings - Continue implementing edge detection on verilog.
- Julian - Start work on more complicated contour transformations and outputting on original image; integrate color transformation with baseline version of other modules.

Week of November 26th:

- Aaron - Debugging and integration of modules
- Wings - Debugging and integration of modules
- Julian - Debugging and integration of modules; continue work on more complicated contour transformations; continue work on outputting to original image.

Week of December 3rd:

- Aaron - Debugging and integration of modules. Final report.
- Wings - Debugging and integration of modules. Final report.
- Julian - Debugging and integration of modules. Final report.

## 4. What could go wrong?

The major source of uncertainty is the unbounded difficulty of the image modulation in a smooth manner. Color change and coarse image alterations seem manageable, but creating smooth transitions of a sprite based on the shape of an FFT may be too computationally intensive of the FPGA(and us). The main concern here are distortions due to resolution issues on the edge detection on the original image. In theory these transformations would look nice, but it will be difficult to tell until we reach that step.

Another source of possible issues will be the memory allocation for the image and audio simultaneously. Finally, the edge detection may also be too computationally intensive for our skills.

Timing is also a big concern in terms of sampling audio and outputting to the VGA display correctly. This includes synchronization issues, as well as issues with time averaging the FFT in order to have changes that are perceivable on the VGA display.

## 5. Equipment needed

We plan on implementing the project on the Nexys 4 Board. To handle the audio portion of the project, we will need to borrow a microphone and a speaker. For the image portion of the project, we will need to borrow a NTSC camera and a desktop monitor. All these equipments are already available to us in lab.