**DESIGN DOCUMENTATION**

***Vybe is an interactive music visualizer created in python3, with an easy-to-use and intuitive user experience and interface.***

• Modules used: PyAudio, Aubio and Pydub.

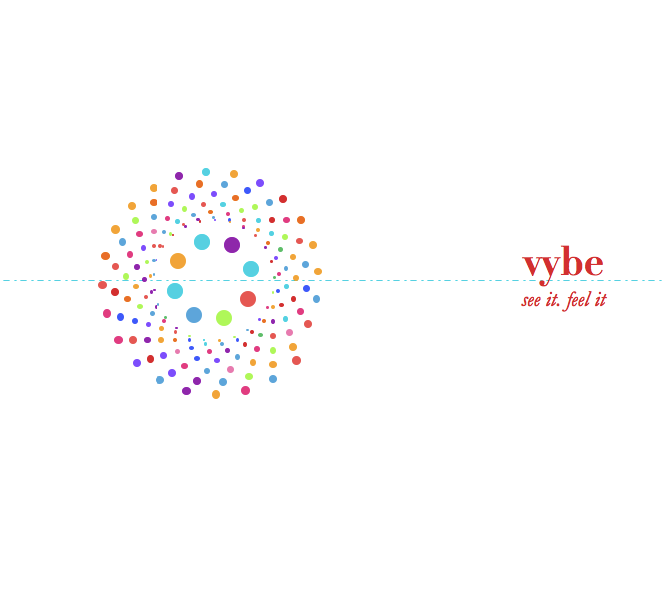
• All my graphics were drawn using principles of physics and were all done in tkinter!

**• Audio Analysis:**

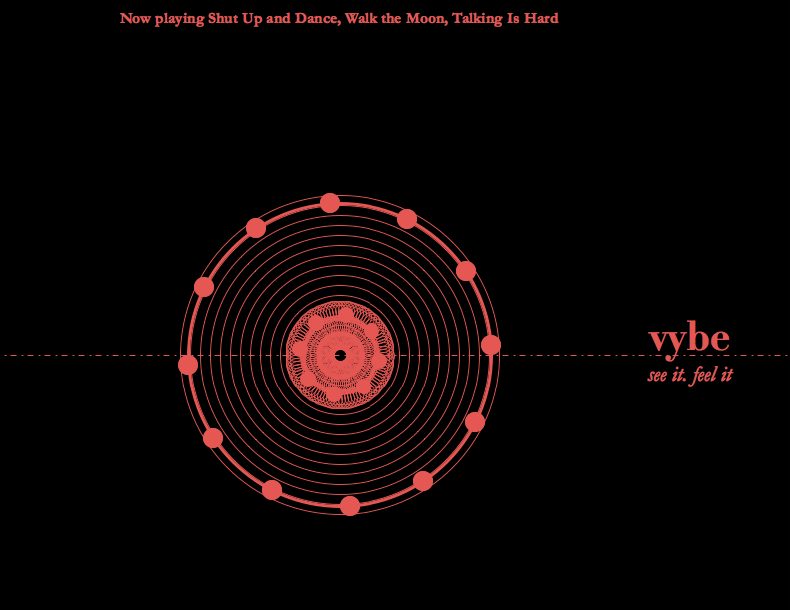
* **class Player** had all the functions used to play and analyze the audio.
* PyAudio is used to play the wav files of the songs.
* Aubio helps detect the beats, pitch and onsets of the song.
* Using the callback function in PyAudio, the function **def** **pyaudio\_callback,** I “threaded” the music to the visuals in tkinter.

**• Visualizations: (**Created using principles of physics!)

* **Colors**: Color change is dependent on onsets in a song. The colors chosen are randomly chosen from a list of colors chosen by me.
* **Logo**: The spinning circles and spirals on the Splash Screen (as seen below; in two different themes).



* **Theme “Classic”:**
  + Has a black background.
  + **class Wheel­**– Has all the functions used to draw a wheel, which resembles planets (In circular motions) connected by an orbit rotating and revolving around the center. It pulses and changes its radius depending on the beat. The rotation is dependent on the pitch.
  + **class Dots**– Dots emanating from the center, according to onsets and pitch of the song. During the visualizations, due to the density and high number of dots emanating, it is a design that resembles floral pattern. The number of dots is dependent on the pitch.
  + **class Waves**– Concentric circles coming out of the center, and depends on the onsets of the song.

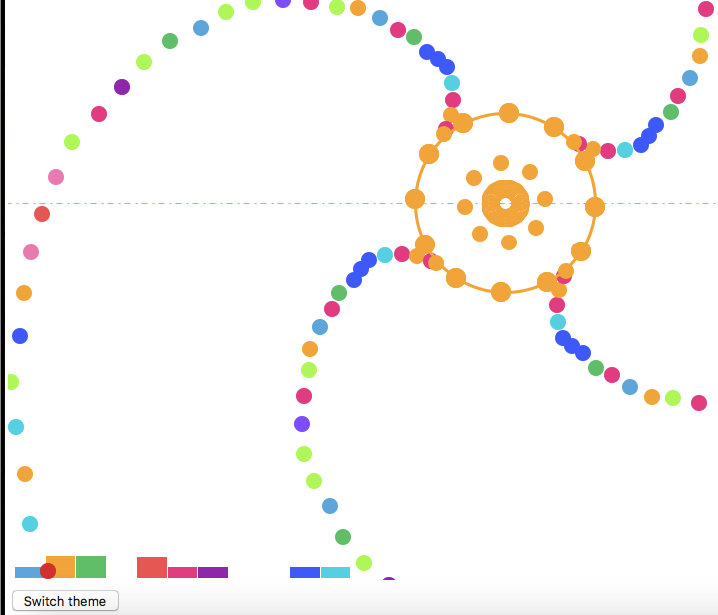


* **Theme “Fun”:**
  + Has a white background.
  + Contains the Wheel and the Dots as well.
  + **class Spiral**– Four spirals that are drawn every time there is a beat. It reverses its movement if there is an onset in the song. The dots in the spiral randomly change color depending on the onset. The spiral radius is also tied to the pitch and is redrawn every time there is a change in pitch as well.



**• Features:**

* **Equalizer:** 
  + Tracks the frequencies (pitches) in the song, in packets of audio data.
  + Is visualized at the bottom right corner of the screen, as a set of rectangles increasing/decreasing with the change in frequency.
  + Drawn using the **def drawEqualizer** and is normalized to be visualized accurately.

****

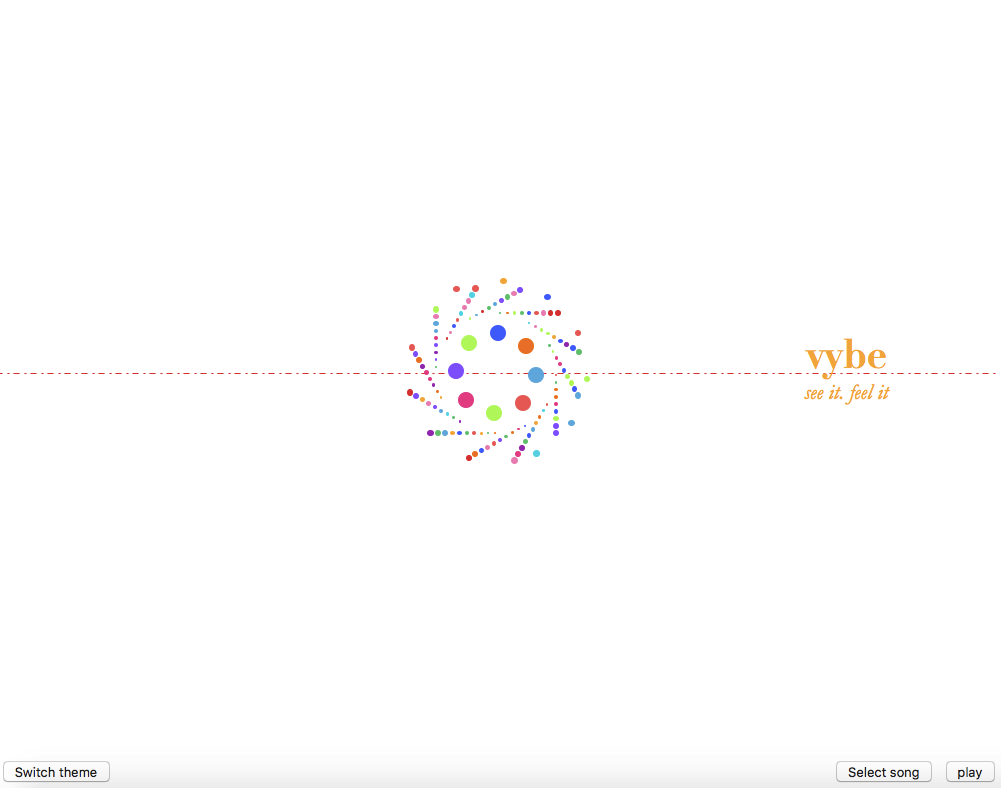
EQUALIZER

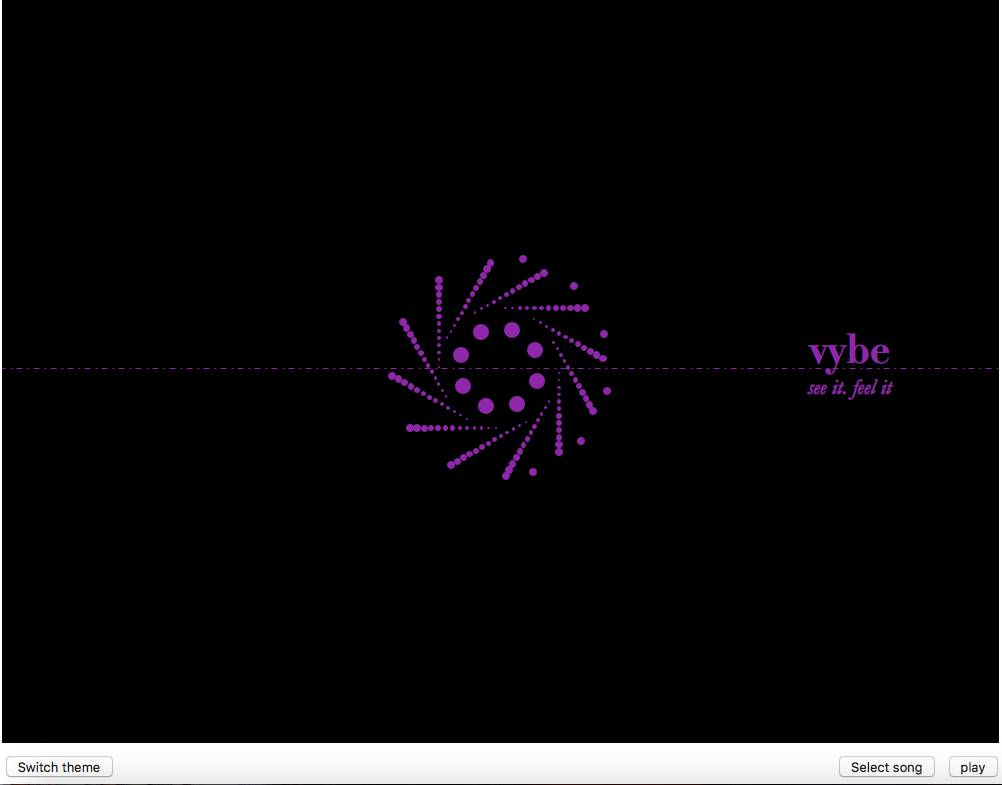
* **Meta data:**
  + Shows the song information at the top of the screen.
  + Displayed using the def **drawMeta** function.

**• User Interface and Experience (UI/UX):**

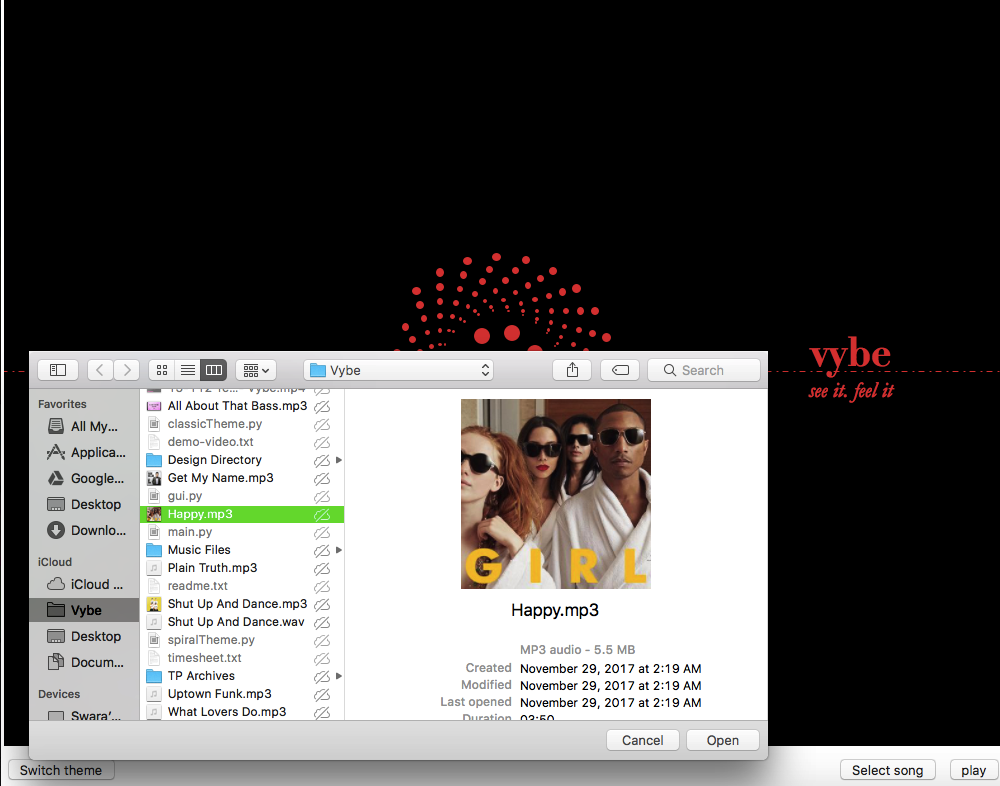
The user interface of the program is intuitive and simple to navigate. There are three buttons that are always present at the bottom of the screen always.

* **Splash Screen: (**class Splash)
  + It is the default home screen.
  + Displays the logo, the buttons and the name of the project.





* **Play/Pause button**:
  + Can play the song and pause the song at any time.
  + Every time the song is paused, the screen reverts to the Splash Screen.
* **Select Song:**
  + Can select any song saved as an mp3 file in your computer.
    - NOTE: the files must be saved in the same folder as your code.
  + Converts the mp3 file to a wav file and then plays it. If it is already a wav file, then it simply queues it up to play.
  + Conversion is done in the function **def openFile.**

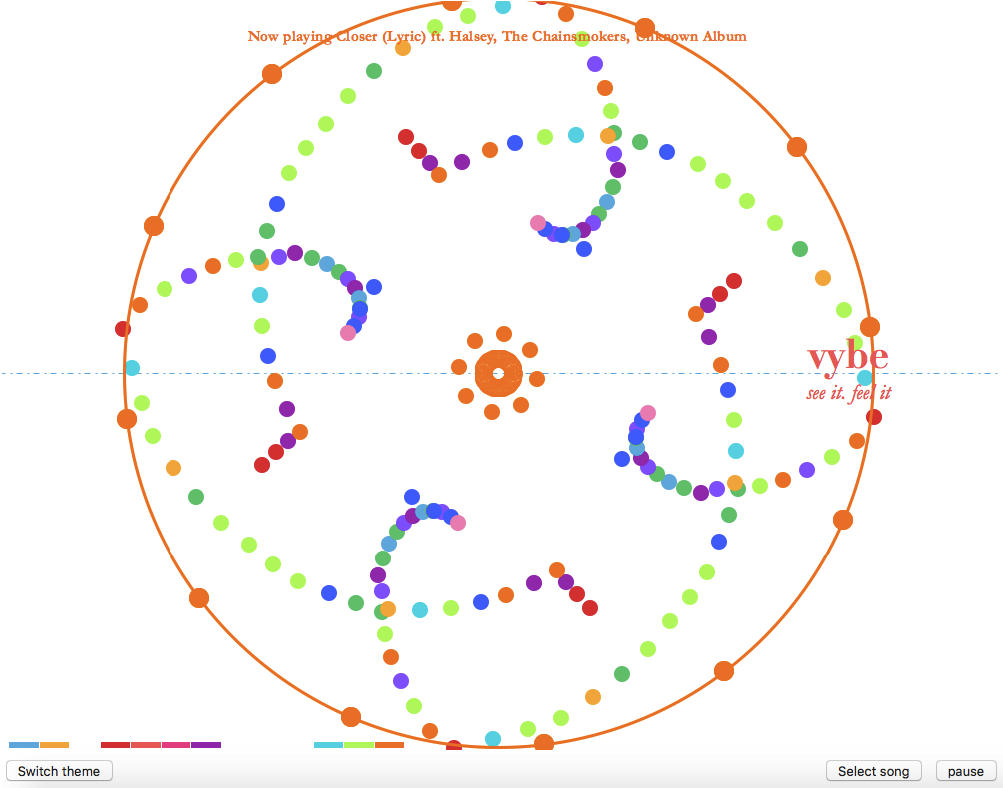


* **Switch Theme:**
  + The user can switch between two themes- “Classic” and “Fun”, without interfering with the song and the continuity of the visuals.

**Classic theme:**

****

**Fun theme:**

****

That’s it! ☺ ~ ***Swara Srinivasan***