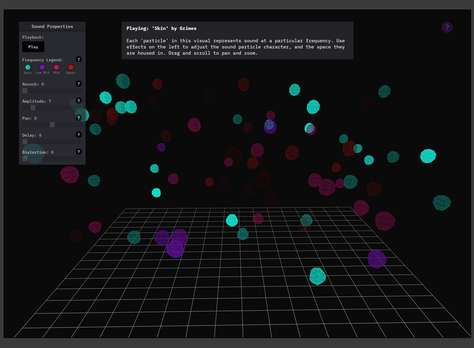
Sound and Space: Process & Context

Overview

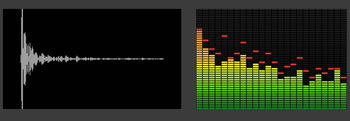
'Sound and Space' is a data visualization showing how sound (particularly, the song 'Skin' by Grimes) occupies sonic and visual space. In this project, sound is broken down into waveforms, and visualized as a set of pulsating particles. As the song progresses, these particles continually transform to reflect the track's sonic properties. The viewer can manipulate different aspects of the track and visualize changes in particle shape and movement.

Link: http://omarnema.com/parsons-info-aesthetics/assignment-3-sound/index.html
Tools: p5.js, p5-sound.js, quick settings.js



Context: State of Audio Visualization

Visuals are frequently paired with music in a variety of forms: music video, installation, lighting. However, audio visualization linked to sonic data is far more limited. Audio data visualization is typically only used in production or playback software. Visualizers embedded in software usually display sound as a bar or line waveform, and typically display a limited set of variables (frequency and amplitude). The result is that these visuals are flat: useful for technical purposes (albeit in a limited set of parameters), but not compelling as a visual accompaniment for a song.



Sample Audio Visuals

Visual Aims

Bearing in mind current limitations in audio visualization, my aims with 'Sound and Space' were to:

- Visualize how sound occupies space
- Create a compelling experience that reflects a chosen track's emotional and sonic properties
- Allow users to interact - manipulate sound and visualize the output
- Break down sound into a few individual sonic attributes, and educate the viewer about what these attributes mean

Design Choices

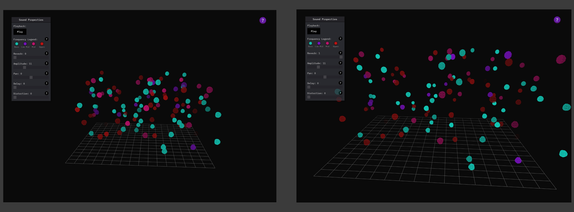
Functionality

The visual displays a series of sound 'particles' in 3D space. Each sphere corresponds to a particular frequency range. The opacity of the sphere reflects the amplitude, or activity of a frequency. These frequencies are intentionally scattered across the 3D space, to reflect the way that sound melds together across different frequencies.

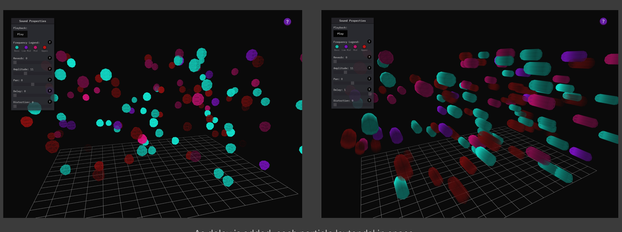
Users have the option to adjust a number five distinct sonic properties: reverb, amplitude, panning, delay, and distortion. As these properties are adjusted, the distribution, shape, and movement of sound particles shift.

For example, reverb refers to the size and quality of the 'space' a sound is housed in. As reverb is increased, the size of the virtual room in 'Sound and Space' expands. As the delay affect is added, particles extend further, reflecting how delay allows frequencies to linger. Amplitude, panning, and distortion are also included, each with their visual effects.

Interface for adjusting sound properties



As reverb is increased, particles become distributed over a larger space.



As delay is added, each particle 'extends' in space

Aesthetic Choices

The visual is intended to mirror the way sound occupies space - how different frequencies and textures can be drifting all over the room. For this reason, the visual is rendered in 3D. I used a dark theme, with small bright pulsing particles to represent the brooding, yet upbeat quality of the chosen track. The particles are somewhat jagged, reflecting the 'lo-fi' nature of the chosen track.

Spheres were used to visualize particles primarily due to their flexibility. I had confidence that I would be able to manipulate the rotation, size, and vibration of a sphere given that it is a core element of the p5.js library.

The visual is intentionally immersive. It takes up the user's full screen and allows for panning and zooming. Contextual information is available on request (by hovering over question marks), but stays in the background otherwise to ensure that the user is fully immersed in the experience. Additionally parameters to manipulate the visual are shown as a draggable, partly transparent overlay to ensure that the visual field is not disrupted.

Data & Code Structure

The p5.js library was used to parse and visualize sound. I used native p5.js functions to load the sound, and convert it to a waveform. I then then wrote code to manipulate, parse, and visualize the waveform. The sound parsing and visualization occurs roughly 20 times per second, this ensuring that the visual is in lockstep with the dynamic audio.

Additionally, quicksettings.js was used to render the sliders used for manipulating audio.

Code here: https://github.com/omar-nema/parsons-info-aesthetics/tree/master/assignment-3-sound/scripts