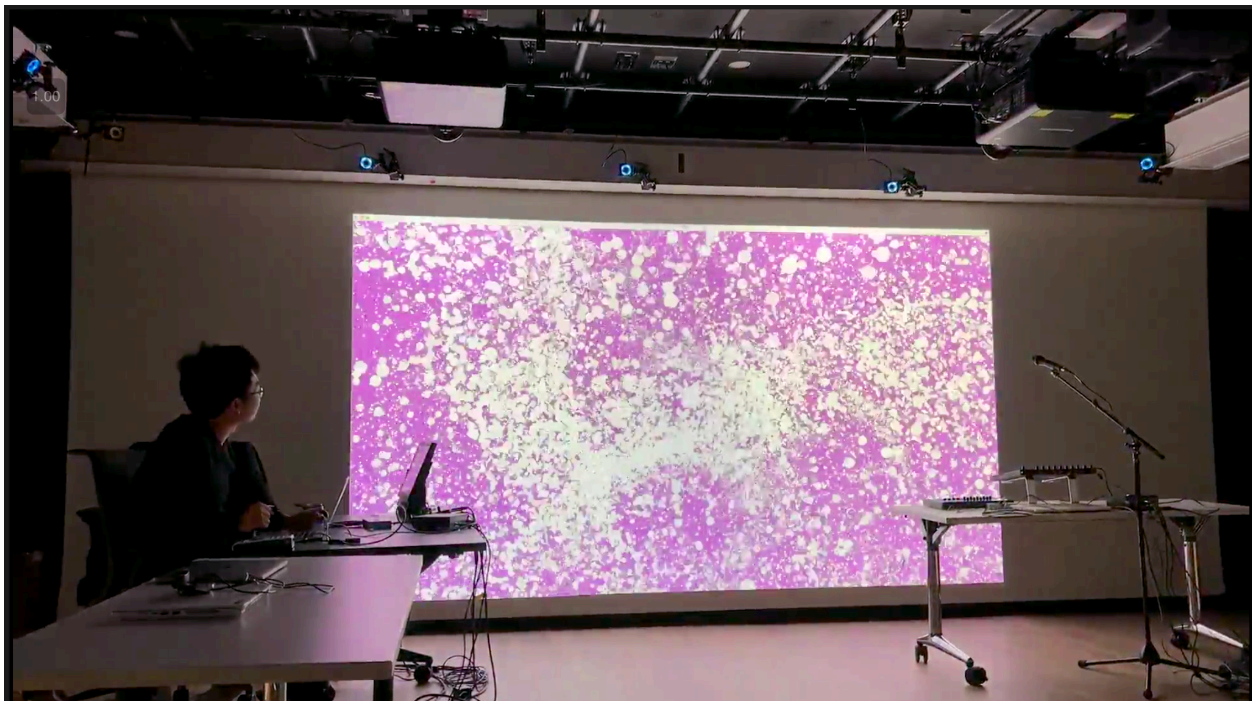
Max Patch Music Visual Project



Video link:
https://drive.google.com/file/d/16tCWxzaHRznmrbCB2rnZCm39Js0a8w_D/view?usp=drive_link

Workflow explained:

This Max patch is an audiovisual system I designed to generate dynamic particle visuals that respond to the beat of a selected piece of music. The patch combines audio analysis and real-time graphics generation, allowing synchronized and engaging visual effects alongside the music playback.

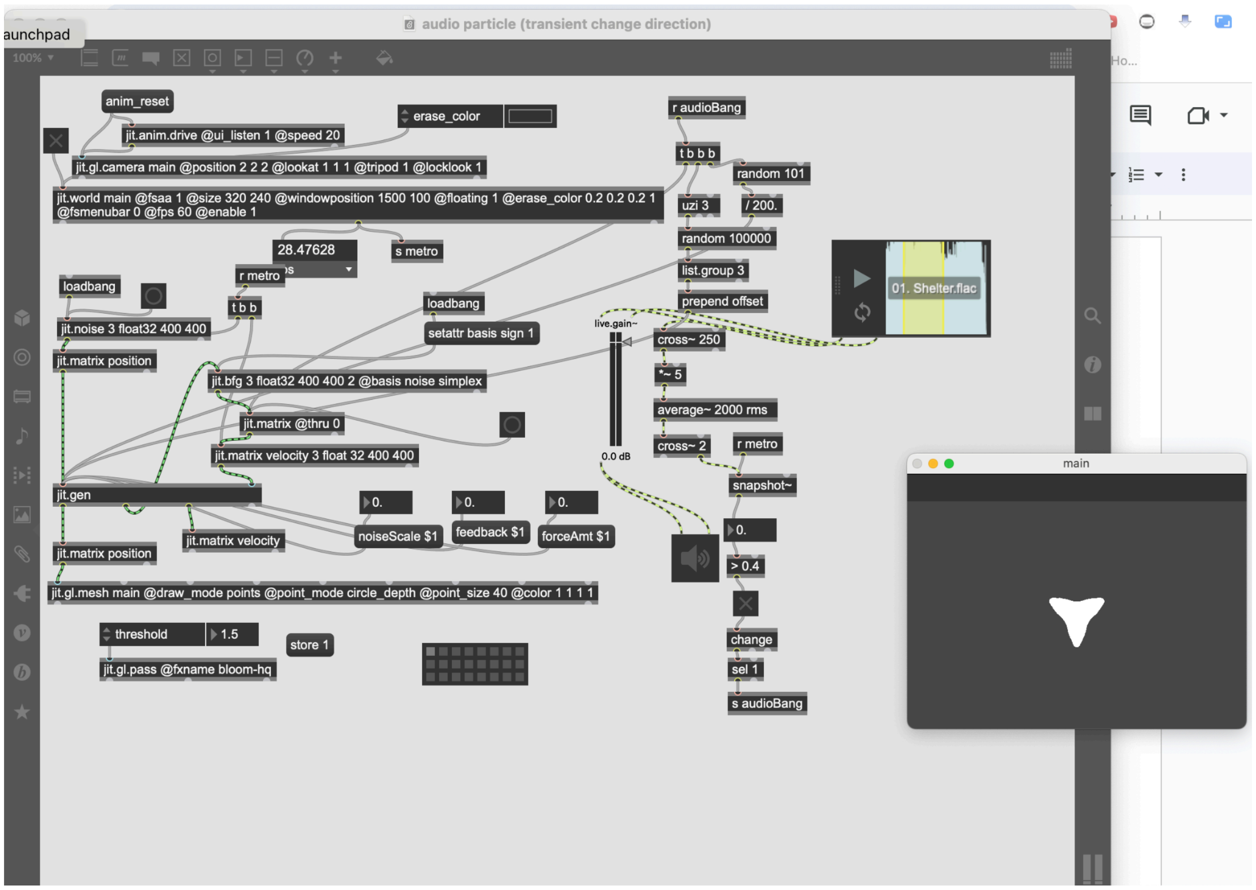
The core of the system is its audio analysis. The music file is played using the live.gain~ object, which adjusts the volume and allows signal monitoring. I use real-time signal processing with objects like snapshot~ and average~ to measure the RMS level of the audio. This provides a smooth representation of the signal's intensity over time, allowing me to detect beats in the music. I can trigger specific events whenever the RMS level crosses a particular value by setting a threshold. These events are then used to control the visual elements of the patch dynamically.

On the visual side, the particle system is built using Jitter objects. I use jit.noise and jit.bfg to generate a 3D matrix of particle positions and velocities, creating either random or procedurally determined particle dynamics. These matrices are then processed using jit.gen, where I compute motion dynamics influenced by parameters like noiseScale, feedback, and forceAmt. The particles are rendered as points in 3D space using jit.gl.mesh, with additional settings for size and color to enhance their appearance. The system allows for various effects as the particles' behavior changes based on the music's rhythm and intensity.

To further link the visuals to the audio, I implemented an audioBang mechanism. When a beat is detected, the system triggers changes in the particle dynamics or other visual properties like colors or animations. This connection ensures that the visuals react in real-time to the music, creating a tightly synchronized audiovisual experience.

To polish the output, I incorporated visual effects such as bloom using the jit.gl.pass object, which adds a layer of refinement to the particle rendering. Camera movement and view transitions are handled through jit.anim.drive and jit.gl.camera, allowing for smooth navigation of the 3D visual space. A metro object ensures the timing of all components remains consistent, driving updates at a regular frame rate.

In summary, this patch explores real-time interaction between audio and visuals. It combines procedural noise-based graphics with beat-driven dynamic changes to create a responsive, immersive system that reacts to the music being played. Through this project, I've built a system that enhances the listening experience and artistically bridges sound and visuals.



audio particle (transient change direction)
aunchpad
anim_reset
jit.anim.drive @ui_listen 1 @speed 20
jit.gl.camera main @position 2 2 2 @lookat 1 1 1 @tripod 1 @locklook 1
jit.world main @fsaa 1 @size 320 240 @windowposition 1500 100 @floating 1 @erase_color 0.2 0.2 0.2 1 @fsmenubar 0 @fps 60 @enable 1
erase_color
28.47628
s metro
r metro
t b b
loadbang
jit.noise 3 float32 400 400
jit.matrix position
loadbang
setattr basis sign 1
jit.bfg 3 float32 400 400 2 @basis noise.simplex
jit.matrix @thru 0
jit.matrix velocity 3 float 32 400 400
jit.gen
jit.matrix velocity
jit.matrix position
jit.gl.mesh main @draw_mode points @point_mode circle_depth @point_size 40 @color 1 1 1 1
noiseScale $1
feedback $1
forceAmt $1
threshold
1.5
store 1
jit.gl.pass @fxname bloom-hq
r audioBang
t b b b
random 101
uzi 3
/ 200.
random 100000
list.group 3
prepend offset
01. Shelter.flac
live.gain~
0.0 dB
cross~ 250
*~ 5
average~ 2000 rms
cross~ 2
r metro
snapshot~
> 0.4
change
sel 1
s audioBang
main