Final Project: Chaotic Harmony - Documentation
Rabab Altarazi
301387828

Github link: (the latest version of the project can be found here)
https://github.com/rababta/final-project-IAT460/tree/main
(The zip file submitted does not have the color adjustments or looping)

.....

Project Overview

- Goal: Create an interactive 3D particle system that:
- Forms/shapes into user-provided images.
- Reacts to audio input (files/microphone).
- Features smooth transitions between states (formation/dispersal).
- Customization options for the user

Tools

p5.js (WebGL + p5.sound).

Web Audio API (via p5.sound abstractions).

HTML/CSS for UI controls.

Key Features & Technical Implementation

A. Particle Swarm Engines

Feature	Implementation	Challenge
Image Formation	Particles lerp from random 3D positions to pre-calculated image-mapped coordinates.	Balancing between speed smoothness
Audio Reactivity	FFT analyzes bass/treble; particles scale/pulse based on frequency energy.	Normalizing audio input ranges.

B. Audio Pipeline

```
A[Audio Input] --> B[Microphone/File]
B --> C[p5.FFT Analysis]
C --> D[Bass/Treble Extraction]
D --> E[Particle Motion/Scale]
```

C. State Management

Loop System:
formationProgress → isDispersing → disperseProgress → (repeat).
Configurable duration via GUI.

Development Process

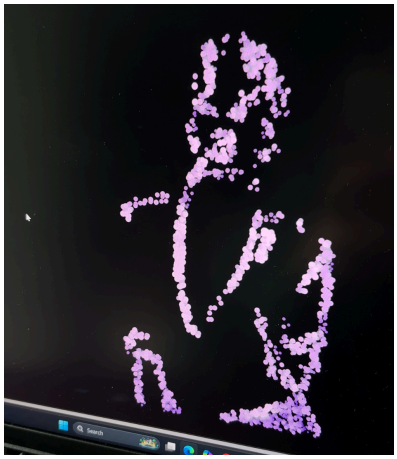
- Iteration 1:

I created a basic swarm system that reacts to mouse interaction and evolves the more the user clicks somewhere and in that direction, however, I wanted to make the interaction and the visuals more interesting.

- Iteration 2:

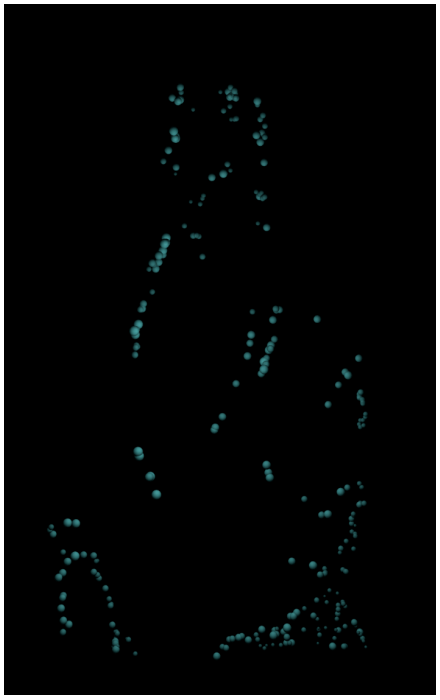
- Used core Swarm Mechanics

- Basic particle system with 2D image formation that traces the outline of elements in the image.



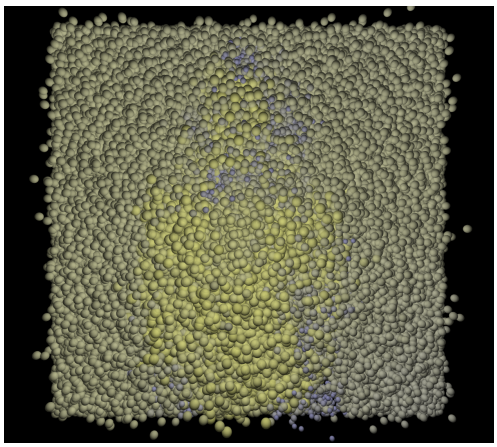

- Interaction 3:

- Sampled image pixels to create 3D particles in target positions in 3D space
- Still used the outline of elements in the image
- Used lerp() for smooth transitions.
- Per-particle speed/offset was needed to avoid robotic movement.



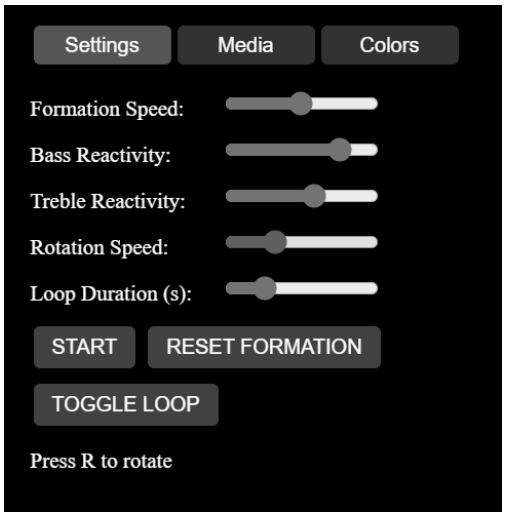

- Iteration 4:

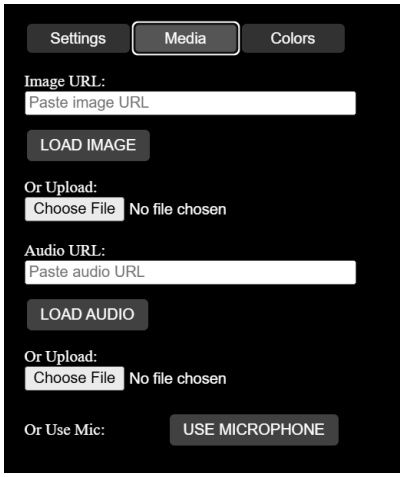
- Particles draw the fall image instead of only the outline
- Audio Integration
- Make particles react to music/mic.
- Hooked p5.FFT to analyze frequencies.
- Mapped bass/treble to particle Z-offset and size.

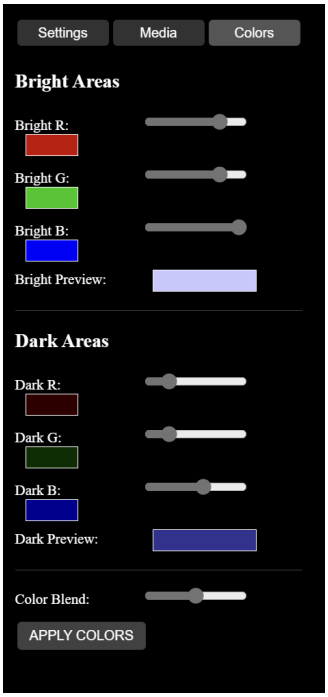


- Iteration 3:

On top of the main user interaction of uploading an image and audio I added the following:







Sample results:

- Source code is available in the GitHub link provided at the beginning of the document.