Music Visualizer

Strahil Peykov, April 2025

About This Project

This music visualizer creates a 3D visual experience that responds to audio in real-time. I created this project because most online visualizers lacked depth and responsiveness to music. Inspired by a Processing-based tunnel visualizer I found on YouTube, I expanded it with advanced audio analysis, multiple visual styles, and physical controls.

Inspiration & Credit

This project builds upon the foundation of a tunnel-based music visualizer created by user "KnukN" on YouTube. While maintaining the core tunnel concept, I've significantly enhanced the audio analysis, added new visualization modes, implemented chord/note detection, and incorporated physical controls.

Technical Features

- **Advanced Audio Analysis.** Real-time frequency analysis with beat detection, chord recognition, and section identification
- **Synesthetic Color System.** Musical notes mapped to specific colors based on Scriabin's system
- **Dual Input Modes.** Switch between MP3 playback and microphone input
- **Multiple Visual Styles.** Rectangular and circular visualization modes with smooth transitions
- **Physical Controls.** Arduino-based interface with sensors for intuitive interaction
- **3D Camera System.** Interactive viewpoint control with automatic and manual navigation

Visual Elements

- Dynamic star field background that reacts to music intensity
- Mid-layer objects (entities) that pulse with specific frequencies
- Structural elements that form tunnel boundaries
- Wave patterns that respond to bass and rhythm
- Color schemes that shift with musical notes and harmony

Getting Started

1. Place an MP3 file named "song.mp3" in the Processing sketch data folder
2. Install the Minim library in Processing if not already installed
3. Run the Processing sketch
4. Recommended: Connect Arduino for physical controls

Interactive Controls

Keyboard Controls

- `Space`: Toggle between rectangular/circular modes
- `V`: Toggle voice/music input
- `R`: Reset camera
- `A`: Toggle auto camera mode
- `Arrow keys`: Manual camera control
- `+/-`: Adjust visual intensity

Arduino Controls (Optional)

- **Rotary Encoder**: Adjust size/density
- **Encoder Button**: Cycle parameters
- **Joystick**: Camera movement
- **Joystick Button**: Reset camera
- **Mode Button**: Switch visualization mode
- **Ultrasonic Sensor**: Control movement speed