Navi Music Visualizer

Final Project Requirements

Team: David Wells

# Project Description

This is a desktop application that receives audio and minor adjustment settings and outputs a visual representation of the given audio in a decently objective manner. A way to describe the visuals would be “dancing lights with rippling, intermingling color trails”. Visual effects include crepuscular rays, progressive blur, glow, water-like ripples, and most importantly the glowing orbs referred to as fairy lights or “FLiT”s. Settings are purposefully sparse, as this is meant to be an automatic translation of audio to video. Framerate (30fps) and audio sample rate (48 kHz) are fixed, but users will have the option of setting the color palette (including a black-and-white mode for color overlaying in other programs), the random-seed number, and the option of text display of the title, the lyrics, or both. Finally, users can choose any resolution, for which the video preview will be 16 times smaller. For example, 1920x1080 output will produce a 480x270 preview. FLiT flight paths will be affected by resolution, as they try to stay in camera as much as possible.

# Section 1: Introduction: Purpose of Project

This program is meant to be a more objective and visually interesting music visualizer than the other ones on the market. It is much more than just some pulsing frequency chart with randomized colors. It takes influence from early 2000’s screensavers which had a reputation of being difficult to look away from, and it mixes that with basic concepts around group dance.

# Section 2a: Must Have Requirements

1. Automated Audio-to-Video Pipeline
   * ✅Convert audio input into a dynamic video output automatically.
   * Work effectively across a wide range of musical genres (4+).
2. FLiT (Fairy Light Trail) Generation
   * ✅Generate FLiTs based on audio analysis.
   * Control FLiT properties and movement based on:
     + ✅Volume
     + Vibrato
     + ✅Reverb
     + Frequency
     + Tempo
     + ✅Sustain
     + Timbre
     + Proximity and behavior of nearby FLiTs
3. Audio Analysis Engine
   * ✅Detect and break audio into distinct components.
   * ✅Compute relevant audio features for FLiT behavior mapping.
   * ✅Automatically determine the medium volume level of a track (with optional manual adjustment).
4. Speech Transcription and Application
   * Song lyrics may optionally be shown and interacted with by FLiTs
5. User Customization Options
   * Color Options
     + Choose color palette.
     + Set color usage mode (e.g., by instrument, emotion, or other)
   * Technical Settings
     + Select video resolution.
     + ✅Specify input/output file locations.
     + Select the random-seed
   * Speech Transcription
     + Enable or disable speech transcription (simple toggle).
     + Display transcribed words in time with audio
6. Export/Output
   * ✅Generate final video output from audio input and FLiT visualizations.

# Section 2b: Stretch Requirements

1. Video Editing Features
   * Include camera movement options
   * Integrate word placement tools for transcription display
   * Ability to save program settings as a program file.
2. Other Visual Settings
   * Select background image.
   * Adjust image durability under FLiT influence.
   * Choose from multiple visual styles (e.g., 8-bit, watercolor, plain).
   * Set FLiT synchronization behavior (e.g., synchronized movement).
   * Set text interaction mode (walls, FLiT children, turned on by FLiTs, etc.)
3. Free Public Release
   * Distribute the application for free for one year.
   * Include a Patreon link on the official website.
4. Other Technical Settings
   * Set CPU/GPU usage preference.

# Section 3: Overview of the Product:

## Workflow:

The program starts as a small, windowed application with a preview screen and the list of settings shown under it. Limited input includes resolution, input file location, export location, video titling, a 1-to-8-color palette with a color picker, a text display slider for showing the title and/or lyrics and/or nothing, and a random-seed number.

Once you have determined the settings you want, there is a preview button and an export button at the bottom of the page.

All these are inputs for the final video, which will output as an MP4 file in the given resolution at the audio sample rate of 48 kHz. You can preview the resulting video or export it, but you cannot do both at the same time.

## GUI:



## Section 4: Verification: How will the project be tested.

I will select a test audience to compare the outputs of my program to the outputs of other music visualizers. They will score it on stunningness, attention hold, enjoyment, and rewatchability.

The audio must never disconnect from the video or fall behind. Obvious requirements like having a video output that has the audio in it must be met. All buttons must be functional in the GUI.