Sound and Visuals Synthesizer

Brief outline of what it does

This project is a visual and interactive program written in Processing. It creates a scene with particles and sparkles that move around the screen. The movement and appearance of the particles and sparkles are influenced by the sound volume, which can be adjusted using increase and decrease buttons. The program uses the Minim library for audio synthesis and playback.

The audio aspect of the program is powered by the remarkable Minim library. It brings to life an oscillating waveform that complements the visuals, immersing you in an all-encompassing sensory experience. The increase and decrease buttons provide a simple yet effective means to fine-tune the audio, allowing you to orchestrate a symphony of colors and sounds.

The project creates a visual display of moving particles and sparkles. The sound volume can be adjusted using buttons, which affect the animation and movement of the visual elements.

Brief outline of the code

- ❏ Import necessary libraries: `ddf.minim` for audio synthesis and playback.

- ❏ Initialize variables and objects:
- `Minim minim`, `AudioOutput audioOut`, `Oscil oscillator`: Audio-related objects.
- `ArrayList<Particle> particles`, `ArrayList<Sparkle> sparkles`: Collections for particles and sparkles.
- `int numParticles`, `int numSparkles`: Number of particles and sparkles.
- `boolean isSoundOn`: Sound state.
- `float volume`, `float volumeIncrement`: Volume level and increment.

- ❏ Setup the environment:
- `setup()`: Initialize the sketch, audio objects, and buttons.

- ❏ Draw loop:
- `draw()`: Update and display particles, sparkles, and buttons.

- ❏ User interaction:
- `mouseClicked()`: Handle button clicks and volume changes.

- ❏ Particle class: Represents particles with position, size, speed, and color.

- ❏ Sparkle class: Represents sparkles with position, size, speed, color, and angle.

- ❏ Button class: Represents buttons with position, size, label, and color. Provides animation and interaction functionality.

Notes on the design decision № 1

During the creation of the project, I came up with a few ideas to enhance the design. I decided to incorporate arrow buttons for sound control, resembling the arrows you find on a computer keyboard.

Why are they orange, you may wonder? Well, I have a profound love for fireplaces, and gazing into a blazing fire is my eternal bliss. That's precisely why I opted for a campfire motif in the prototype of my project, and consequently, the buttons are adorned in shades of orange.

Moreover, to captivate users' interest, I resolved to add a subtle animation to these arrow buttons. When pressed, they instantly change their hue to red, mirroring the association between flames and the color red.

Notes on the design decision № 2

The second idea to enhance the project was to visually represent fire in the form of an explosive burst when the code starts running.

- Firstly, I wanted to create a shape that resembled flames. After numerous attempts and creative explorations, I settled on using circles.
- Secondly, it was crucial to determine the colors of these circles. This part was relatively straightforward as I aimed to capture the fiery hues, such as orange, yellow, and a touch of red.
- And finally, as a third step, I decided to incorporate gleams to provide structure and add a captivating visual impact to the explosion.

Notes on the design decision № 3

The next idea to enhance the design involves creating new functionality for the project, specifically related to controlling the volume using arrow buttons. I have decided to incorporate a visual explosion that changes based on whether the sound is being increased or decreased.

Therefore, when the up arrow button is pressed, the visual explosion of flames will accelerate, accompanied by an increase in volume. Pressing the down arrow button will gradually decrease the sound, and if held for an extended period, it will be completely muted.

Meanwhile, the visual effect will either slow down, resembling slow-motion footage, or freeze entirely, as if paused.

For the final touch, I have decided to refine the visual representation of the explosion. I concluded that the explosion should originate from the center, with circles bursting outwards, emulating the effect of a detonation. These elements will then move randomly across the screen, as fire disperses chaotically in various directions after an explosion. This will introduce an element of unpredictability and enhance the realism of the fire's movement.

Explanation on how to operate a tool

This code creates a visual and interactive program using particles and sparkles on a canvas. The program also includes buttons to control the volume of a sound.

When you run the code, you will see a window with a black background and various colorful particles and sparkles moving around. The particles are represented by small circles, and the sparkles are represented by even smaller circles.

There are two buttons displayed on the screen: an upward arrow ("▲") and a downward arrow ("▼"). These buttons are used to control the volume of a sound.

To increase the volume, click on the upward arrow button. Each click will increase the volume slightly. As you click, the button will animate by changing color to red and slightly increasing in size.

To decrease the volume, click on the downward arrow button. Each click will decrease the volume slightly. Similar to the upward arrow button, the downward arrow button will animate by changing color to red and slightly increasing in size.

The particles and sparkles will respond to the volume changes. When the volume is increased, the particles and sparkles will move faster, creating a more dynamic visual effect. When the volume is decreased, the particles and sparkles will slow down.

You can experiment with clicking the buttons multiple times to adjust the volume and observe the changes in the particle and sparkle movement.