Pygame OpenGL 3D Rendering Project Documentation

This document explains the structure and functionality of a Python project that uses Pygame and OpenGL to render a 3D scene with interactive camera controls and audio effects.

# 1. Project Overview

The project initializes a Pygame window with an OpenGL context, loads a background image as a texture, and renders 3D models loaded from external files. It uses shaders to handle lighting and special effects such as glowing. Audio playback includes background music and sound effects that fade in and out.

# 2. Code Structure

## 2.1 Initialization

The Pygame library is initialized along with its mixer module for audio. The OpenGL context is set up within a window created by Pygame, with depth testing enabled for proper 3D rendering.

## 2.2 Background Setup

A background image is loaded and converted into an OpenGL texture. A fullscreen quad is created with vertex positions and texture coordinates, rendered with a custom shader to display the background.

## 2.3 3D Model Loading and Rendering

3D objects are loaded from text files representing materials and geometry. A shader program manages projection, view, and model matrices to properly transform objects within the scene.

# 3. Camera Controls

The camera position and orientation are controlled using mouse input and the mouse wheel:  
- Scrolling the mouse wheel zooms in and out by adjusting camera distance.  
- Clicking and dragging with the left mouse button rotates the camera by changing rotation angles on the X and Y axes.  
These controls provide intuitive navigation around the 3D scene.

# 4. Audio System

Background music is loaded and played in a continuous loop. When the user presses keys 1, 2, or 3, sound effects specific to certain characters are played on a separate audio channel.  
To preserve the background music, the volume is smoothly faded down before playing the effect, then faded back up after a delay, rather than stopping the music.

# 5. Key Event Controls

|  |  |
| --- | --- |
| Key/Event | Action |
| P key | Prints current camera zoom and rotation to console and appends the information to 'view\_log.txt'. |
| 1 key | Plays the 'Charmander' sound effect and triggers a glow effect on the corresponding object while fading background music. |
| 2 key | Plays the 'Bulbasaur' sound effect and triggers a glow effect similarly with music fade. |
| 3 key | Plays the 'Squirtle' sound effect and triggers its glow effect with fading music. |
| Mouse wheel up | Zooms the camera in by decreasing the distance. |
| Mouse wheel down | Zooms the camera out by increasing the distance. |
| Left mouse button down | Starts camera rotation control via dragging. |
| Left mouse button up | Stops camera rotation control. |
| Mouse movement while dragging | Rotates the camera based on mouse movement delta. |

# 6. Rendering Loop

The main loop maintains a consistent frame rate and processes events. It handles camera updates, music fading logic, and renders the background and all 3D objects with their animations and glow effects.

# 7. Cleanup

On exit, all OpenGL resources such as vertex arrays, buffers, and textures are deleted, and Pygame is properly quit.