# Minecraft 3D Castle

## INTRODUCTION

This project demonstrates rendering a textured 3D castle model inside a Python application using Pygame and OpenGL 3.3. It features a live video background, smooth camera controls, and audio playback to create an immersive exploration experience.

### How to Run and Use the Program:

1. Run the program by executing the main Python script:

**python main.py**

1. **Auto mode** will play for 25 seconds after opening/running the main.py and will then switch to manual mode **automatically**.
2. Explore the 3D scene using the following controls:

* Toggle Auto/Manual Camera Mode: **Press M**
* Auto mode cycles through preset camera views automatically.
* Manual mode lets you control the camera view.
* Navigate Views (Manual Mode):
* Press **Arrow keys** or **WASD** to cycle through available views.
* Press **number keys** **1–9 or 0** to jump directly to a specific view.
* Press **R** or **Spacebar** to jump to a random view.
* Camera Zoom: Use the mouse scroll wheel to zoom in and out within limits.
* Camera Rotation: Click and drag the left mouse button to rotate the camera smoothly.
* Adjust Camera Movement Speed: Press + to increase and - to decrease the smoothness of camera motion.

1. Exit the program by closing the window.

## OVERVIEW

This project renders a 3D castle model over a live video background inside a Pygame window using OpenGL. It includes smooth camera controls, automatic and manual view modes, zooming, rotation, and background music playback. The background video is played using OpenCV and displayed as an OpenGL texture mapped onto a full screen quad. The 3D castle model is loaded from custom vertex and index files, textured, and rendered using GLSL shaders.

## FEATURES

* 3D model rendering with textures using OpenGL 3.3
* Live video background rendered with OpenCV frames
* Multiple camera views with automatic cycling and manual control
* Smooth mouse-controlled camera rotation and zooming
* Background music playback with intro and loop tracks
* Zoom limit feedback with beep sound

## REQUIREMENTS

* Python 3.x
* Pygame
* PyOpenGL
* Pillow (PIL)
* NumPy
* OpenCV (cv2)
* pyglm

## PROJECT STRUCTURE

* **main.py**  
  This is the entry point of the application. It contains the main event loop that handles user input, updates the scene state, and triggers the rendering of 3D models and the video background. It sets up the OpenGL context, initializes shaders, loads 3D models and textures, manages camera and lighting transformations using the glm library, and coordinates frame updates at a fixed frame rate. Additionally, it handles integration with OpenCV for video playback as a dynamic background.
* **model\_loader.py**  
  This module is responsible for loading 3D model data from external files (such as .txt or .obj) and preparing it for rendering in OpenGL. It parses vertex positions, normals, texture coordinates, and indices, then creates and binds the necessary Vertex Buffer Objects (VBOs), Element Buffer Objects (EBOs), and Vertex Array Objects (VAOs). This ensures efficient GPU-side storage and retrieval of mesh data for high-performance rendering.
* **texture\_loader.py**  
  This component manages loading image files into OpenGL textures. It handles image decoding, format conversion, and sets texture parameters like filtering and wrapping modes. It prepares textures for efficient sampling in the shader programs, enabling the application of detailed surface patterns and colors onto 3D models.
* **textured\_shader.py**  
  Contains the GLSL shader source code and compilation routines necessary to render textured 3D models. This includes vertex shaders for transforming vertices and fragment shaders for texture sampling and lighting calculations. The module compiles, links, and validates the shader programs and provides interfaces for setting uniform variables such as transformation matrices and lighting parameters.
* **bg\_loader.py**  
  This directory contains all the external assets used by the project, organized by type: index, vertices, texture, videos, music. Each asset is loaded at runtime by the respective loader modules to be integrated into the rendering pipeline.

## CONTROLS SUMMARY

|  |  |
| --- | --- |
| **Control Key/Button** | **Action** |
| M | Toggle auto/manual camera mode |
| Arrow keys / WASD | Cycle through views (manual mode) |
| Number keys 1–9, 0 | Jump to specific view |
| R or Spacebar | Random view |
| Mouse Wheel | Zoom camera in/out |
| Left Mouse Drag | Rotate camera |
| + / - | Adjust camera smoothness (lerp speed) |
| Window Close | Exit program |