3D Escape Room

CS352: Computer Graphics & Visualization Lab

Project Report

Course Instructor:

Dr. Somnath Dey

Submitted By:

Sreya Shanmukhi – 210001010

Gulshan Nag – 210001021

Harshavardhan – 210001029

Introduction:

The objective of this project is to design and develop a captivating 3D escape room experience using computer graphics and visualization techniques. Within the confines of this project, we will delve into the intricacies of 3D modeling, scene rendering, and user interaction, all orchestrated through the GLUT (OpenGL Utility Toolkit) library. Our scope encompasses the creation of a room with all different objects with various features implemented as per the functionalities as near as possible to actual escape room.

Specifications:

Headers Used

OpenGL Related Headers:

- GL/gl.h: Contains essential declarations for OpenGL's core functions and data types, enabling low-level graphics rendering tasks such as drawing primitives, applying transformations, and setting up lighting effects.
- GL/glu.h: Provides additional functionality through the OpenGL Utility Library (GLU), offering higher-level operations like matrix manipulation, surface tessellation, and curve rendering. GLU functions simplify complex tasks and enhance OpenGL's capabilities.
- GL/glut.h: Includes declarations for functions from the OpenGL Utility Toolkit (GLUT), which simplifies common tasks in OpenGL applications such as window creation, event handling (keyboard and mouse input), and managing the application's main loop. GLUT abstracts platform-specific details, making it easier to develop cross-platform OpenGL applications.

Standard C++ Libraries:

- cstdlib: Provides declarations for standard C library functions, including memory allocation, random number generation, and program termination.
- cstdio: Contains declarations for standard C I/O functions, such as file input/output operations and printing formatted output to the console.
- iostream: Declarations for standard C++ input/output streams, enabling console input/output and file streaming.
- string: Contains declarations for string manipulation functions in C++, allowing manipulation, comparison, and concatenation of string objects.
- sstream: Provides declarations for string stream classes in C++, which allow for string-based input/output operations.
- cmath: Contains declarations for mathematical functions in C++, including common mathematical operations like trigonometric functions, logarithms, and exponentiation.
- bits/stdc++.h: Includes all standard C++ library headers, offering a convenient way to include commonly used C++ headers in a single line.

To run the Code:

g++ file.cpp -lGL -lGLU -lglut (this will produce an ouput file)

then run ./a (to get final output)

Details of keys:

1) To open and close the locker: f

2)To move Eye point:

- w: up
- s: down
- a: left
- d: right
- i: zoom in
- o: zoom out

3)To move Camera point:

- j: up
- n: down
- b: left
- m: right
- l: move nearer
- k: move far

4)Press q to move to default position

5)For lighting:

Light source 1 [the light on the right on the screen

- 1: to turn on/off light one
- 4: to turn on/off ambient light one
- 5: to turn on/off diffusion light one
- 6: to turn on/off specular light one

Light source 2 [the light on the left on the screen

- 2: to turn on/off light two
- 7: to turn on/off ambient light two
- 8: to turn on/off diffusion light two
- 9: to turn on/off specular light two

Lamp light (spot light)

- 3: to turn on/off lamp
- e: to turn on/off ambient lamp light
- r: to turn on/off diffusion lamp light
- t: to turn on/off specular lamp light

Functionalities Implemented

Part 1: Setting up Scene and Basic Interaction

Camera Movement Implementation:

We initialized variables (eyeX, eyeY, eyeZ, refX, refY, refZ) to represent the position of the camera's eye and the reference point.

The myKeyboardFunc function was defined to handle keyboard input for camera movement. Upon receiving input ('w', 's', 'a', 'd', 'i', 'o'), it updated the camera's position and orientation accordingly.

Inside the display function, we utilized gluLookAt to set up the camera's view based on the eye and reference points.

Lighting Control Implementation:

We implemented functions (lightOne, lightTwo, lampLight) to configure different light sources and their properties using OpenGL's lighting functions.

These functions adjusted ambient, diffuse, and specular properties of each light source, providing control over their intensity and color.

Keyboard functions (myKeyboardFunc) were extended to handle toggling individual lights (1, 2), adjusting ambient, diffuse, and specular properties (4-9), and controlling the spotlight of a lamp (3, e-t).

Keyboard Controls Implementation:

The myKeyboardFunc function was defined to interpret keyboard input for various interactions.

Using conditional statements, it mapped key presses to specific actions, such as updating camera positions, toggling lights, or adjusting lighting properties.

For instance, pressing 'w' or 's' moved the camera vertically, 'a' or 'd' moved it horizontally, and 'i' or 'o' zoomed in or out, respectively.

Part 2: Rendering Objects and Enhancing Visuals

Object Rendering Implementation:

We created rendering functions (CricketBall, Torus, diamond, bottle, wallshelf, etc.) to draw different objects using OpenGL's geometric primitives and transformations.

Each rendering function utilized translation, rotation, and scaling operations (glTranslatef, glRotatef, glScalef) to position, orient, and scale the objects appropriately within the scene.

Text Rendering Implementation:

Functions (building_name, building_name2) were employed to render text on the screen using GLUT's stroke fonts.

To render each character accurately, we translated, rotated, and scaled them individually within the text rendering functions, ensuring proper alignment and spacing.

User Interaction Implementation:

Mouse functions (myMouseFunc) were integrated to detect mouse clicks on interactive elements or regions within the scene.

These functions were responsible for handling mouse events and triggering corresponding actions, such as displaying additional information or advancing the narrative based on the user's interactions.

Part 3: Completing the Scene and Final Touches

Obtaining specificized objects with texture using default shapes like solid sphere, solid cone, solid cube etc., On giving parameters with respect to ambient light, diffuse light and shininess to the functions for specificized objects can help accessing the objects with desired texture.

Creating room, bed (with pillows, blanket), study table, chair, locker (with objects like cricket ball, doughnut, diamond), wall poster, window, window scenery, wall clock, lamplight, bed side drawer, bottle, teapot, books, door, shelves with different objects by using the functions and applying respective translations, scaling, rotations.

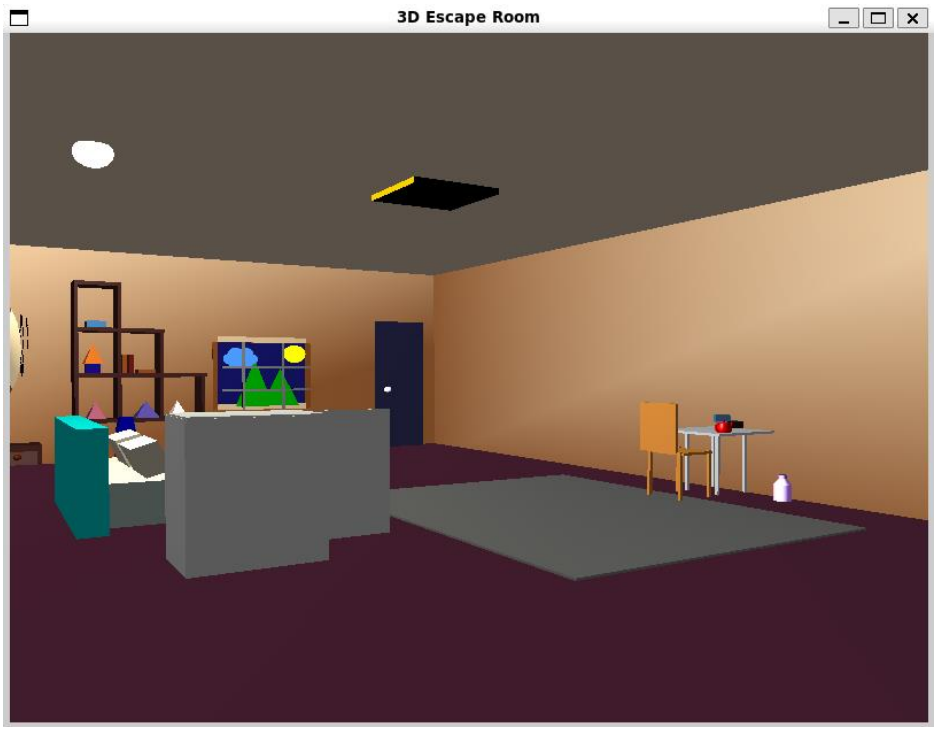
Reference: https://github.com/n-gauhar/3D-bedroom/tree/main

Used for basic understanding of lights,camera angles and room structure.

Output:

Turning on lightone:

Turning on lighttwo with changing view angle



Turning lamplight:

Changing camera angle using k, i on key board and focusing on clock side:

Zooming out book and getting clue:

Opening locker, zooming two of the objects by clicking on them:

Turning on lamp light and zooming out by changing camera position and look at point:

Zooming out to see the entire room: