Kathford International College of Engineering and Management
Balkumari, Lalitpur

4th Semester BEI

A PROJECT REPORT OF "3D House"

Submitted To:
Department of Computer and Electronics

Submitted By:
Bishal Humagain (KIC078BEI010)
Nishant Bhandari (KIC078BEI015)

March 5, 2024

ACKNOWLEDGEMENT

First and foremost, we would like to express our deepest appreciation to our subject instructors **Er. Chandra Shekar** sir and **Er. Prabin Kumar Jha** sir whose guidance, expertise, and unwavering support have been instrumental throughout this journey. Their dedication to our learning and their insightful feedback have enriched our understanding and helped us to move forward. Furthermore, special thanks to **Department of Computer and Electronics** whose expertise and guidance have provided invaluable insights and direction at critical junctures of the project and provided us advice and suggestion towards the selection of our project topic, content necessary and for continuous support throughout the course of this project.

We would also like to extend our gratitude to our peers and fellow students for their collaboration, and encouragement. The exchange of ideas, discussions, and collective brainstorming sessions have played a pivotal role in shaping the trajectory of this project.

ABSTRACT

The purpose of the project entitled as "**3D HOUSE**" is to explore into the concept of 3D house design, where we design a virtual environment using computer graphics techniques. This project helps in constructing a virtual representation of a house using rendering techniques and interactive elements.

Through the help of perspective projection and along with the principles of 3D space transformation, we ensured that our virtual house accurately simulates the perspective view of a real-world observer. Through modeling, we constructed every aspect of the house employing techniques such as polygonal modeling and spline-based curves to achieve real like detail. Through the help of various illumination model, lighting also served as a cornerstone of our project, as we explore the relation between light and shadow to evoke realism and atmosphere within the virtual environment.

Throughout the project, we prioritized optimization and performance, through OpenGL's hardware-accelerated rendering pipeline to achieve real-time rendering speeds while maintaining visual quality. Our ultimate goal was to create a virtual house that invites users to explore, interact, and appreciate the beauty of digital architecture.

Table of Content

Introduction:

The field of computer graphics plays a crucial role in the creation of realistic and immersive virtual environments. Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D and 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly and efficiently. Interactive computer graphics is the most important means of producing pictures since the invention of photography and television. It has the added advantage that, with the computer, we can make pictures not only of concrete real-world objects but also of abstract, synthetic objects, such as mathematical surfaces and of data that have no inherent geometry, such as survey results.

OpenGL (open graphics library) is a standard specification defining a cross language cross platform API for writing applications that produce 2D and 3D computer graphics. The interface consists of over 250 different function calls which can be used to draw complex 3D scenes from simple primitives. OpenGL was developed by silicon graphics Inc. (SGI) in 1992 and is widely used in CAD, virtual reality, scientific visualization, information visualization and flight simulation. It is also used in video games, where it competes with direct 3D on Microsoft Windows platforms. OpenGL is managed by the non-profit technology consortium, the Chronos group, Inc.

OpenGL serves two main purposes:

- To hide the complexities of interfacing with different 3D accelerators, by presenting programmer with a single, uniform API
- To hide the differing capabilities of hardware platforms, by requiring that all Implementations support the full OpenGL, feature set.

Many OpenGL functions are used for rendering and transformation purposes.

Transformations functions like glRotate (), glTranslate (), glScaled () can be used.

OpenGL provides a powerful but primitive set of rendering command, and all higher-level drawing must be done in terms of these commands. There are several libraries that allow you to simplify your programming tasks, including the following:

OpenGL Utility Library (GLU) contains several routines that use lower-level OpenGL commands to perform such tasks as setting up matrices for specific viewing orientations and projections and rendering surfaces.

OpenGL Utility Toolkit (GLUT) is a window-system-independent toolkit, written by Mark Kill guard, to hide the complexities of differing window APIs.

To achieve the objective of the project, information related to the light sources is required with OpenGL we can manipulate the lighting and objects in a scene to create many different kinds of effects. It explains how to control the lighting in a scene, discusses the OpenGL conceptual model of lighting, and describes in detail how to set the numerous illumination parameters to achieve certain effects. This concept is being obtained from.

To demonstrate the transformation and lightening, effects, different polygons have to be used. Polygons are typically drawn by filling in all the pixels enclosed within the boundary, but we can also draw them as outlined polygons or simply as points at the vertices.

The properties of a light source like its material, diffuse, emissive, has to mention in the project. So, to design the light source and the objects, programming guide of an OpenGL is used.

As a student learning computer graphics, the proposed project, titled "3D House," aims to explore and demonstrate advanced techniques in 3D modeling, rendering, and visualization to create a detailed and visually appealing digital representation of a house.

OBJECTIVE:

The primary objective of the 3D House project is to implement cutting-edge computer graphics technologies to construct a three-dimensional virtual model of a house. This involves the integration of various components such as architectural design, realistic texturing, lighting, and animation to produce an interactive and visually stunning digital experience.

1. **Modelling:** To craft a detailed 3D model of a house, encompassing architectural elements such as walls, windows, doors, and roofing structures.
2. **Texturing:** To apply realistic textures to the surfaces of the 3D model, simulating materials, thereby enhancing visual fidelity.
3. **Lighting:** To implement lighting techniques to illuminate the virtual environment, creating realistic light and shadow effects that enhance the overall visual appeal.
4. **Rendering:** To employ rendering algorithms and technologies to generate high-quality images of the 3D house model, ensuring realism and aesthetic coherence.
5. **Interactivity:** To incorporate interactive elements into the virtual environment, enabling users to navigate through the 3D space, explore different areas of the house, and interact with objects and architectural features.
6. **User Experience:** To prioritize user experience by designing intuitive controls and interfaces that facilitate seamless navigation and interaction within the virtual house environment.
7. **Realism:** To strive for a high degree of realism in the virtual representation of the house, paying attention to details such as scale, proportion, texture resolution, and lighting accuracy.

SYSTEM REQUIREMENTS SPECIFICATION:

HARDWARE REQUIREMENTS

Minimum hardware specification:

- Microprocessor: 2.0 GHz and above CPU based on either AMD or INTEL Microprocessor Architecture
- Main memory: 512 MB RAM
- Hard Disk : 40 GB
- Hard disk speed in RPM:5400 RPM
- Keyboard: QWERTY Keyboard
- Mouse :2 or 3 Button mouse
- Monitor: 1024 x 768 display resolution

SOFTWARE REQUIREMENTS

Minimum software specification:

- Operating system : Windows 7
- IDE: Dev C++ /Code Blocks
- OPENGL and FREE GLUT Library
- X86
- X64(WOW)
- Mouse Driver
- Graphics Driver
- C Language

DESIGN:

EXISTING SYSTEM:

Existing system for a graphics is the Turbo C++. This system will support only the 2D graphics. 2D graphics package being designed should be easy to use and understand. It should provide various options such as free hand drawing, line drawing, polygon drawing, filled polygons, flood fill, translation, rotation, scaling, clipping etc. Even though these properties were supported, it was difficult to render 2D graphics cannot be very difficult to get a 3-Dimensional object. Even the effects like lighting, shading cannot be provided.

PROPOSED SYSTEM:

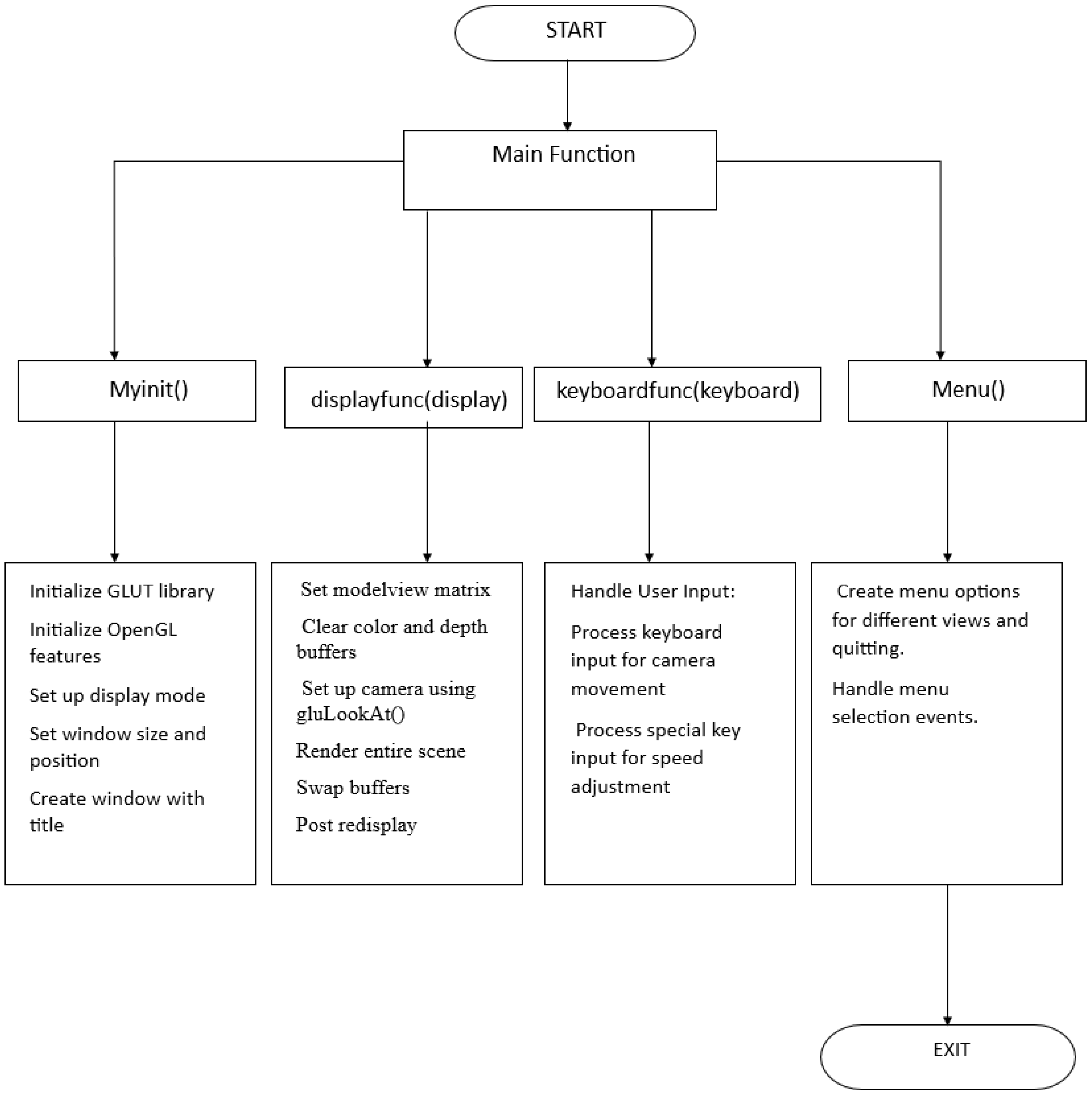
To achieve three dimensional effects, open GL software is proposed. It is software which provides a graphical interface. It is an interface between application program and graphics hardware. The advantages are:

- Open GL is designed as a streamlined.
- It's a hardware independent interface i.e. it can be implemented on many different hardware platforms.
- With OpenGL we can draw a small set of geometric primitives such as points, lines and polygons etc.
- It provides double buffering which is vital in providing transformations.
- It is event driven software.
- It provides call back function.

Algorithm:

1. **Initialization:**
- Initialize GLUT library.
- Set up display mode to include double buffering, RGB color, and depth buffer.
- Set window size and position.
- Create window with a title.
2. **Initialization of OpenGL Features:**
- Enable lighting and depth testing.
- Set shading model to smooth.
- Enable normalization of normal vectors.
- Set clear color for the background.
3. **Define Material Properties:**
- Define functions for setting material properties such as ambient, diffuse, specular colors, and shininess.
4. **Define Light Properties:**
- Define functions for setting up light properties such as position and intensity.
5. **Define Drawing Functions:**
- Define functions for drawing various components of the scene using OpenGL primitives like cubes and cylinders. These functions include drawing walls, roofs, gates, steps, etc.
6. **Define Camera Movement Functions:**
- Define functions to handle user input for camera movement and speed adjustment. This includes functions for keyboard input and special key input.
7. **Define Display Function:**
- Define a display function responsible for rendering the entire scene.
- Within this function:
- Set the modelview matrix and load the identity matrix.
- Clear the color and depth buffers.
- Set up the camera using gluLookAt() function.
- Call functions to render different components of the scene.
8. **Define Menu Function**: For creating menu options to switch between different views of the house and to quit the program.
9. **Main Function:**
- Initialize OpenGL features and register callback functions.
- Enter the GLUT main loop to handle events such as user input and rendering

Flowchart of Low-Level Design:


START
Main Function
Myinit()
displayfunc(display)
keyboardfunc(keyboard)
Menu()
Initialize GLUT library
Initialize OpenGL features
Set up display mode
Set window size and position
Create window with title
Set modelview matrix
Clear color and depth buffers
Set up camera using gluLookAt()
Render entire scene
Swap buffers
Post redisplay
Handle User Input:
Process keyboard input for camera movement
Process special key input for speed adjustment
Create menu options for different views and quitting.
Handle menu selection events.
EXIT

Implementations:

Some of the Functions used in the program:

- **glColor3f(float, float, float):** This function will set the current drawing colour.
- **gluOrtho2D(GLdoubleleft,GLdouble right,GLdouble bottom, GLdouble top):**which defines a two dimensional viewing rectangle in the plane z=0.
- **glClear():**Takes a single argument that is the bitwise OR of several values indicating which buffer is to be cleared.
- **glClearColor():**Specifies the red, green, blue, and alpha values used by glClear to clear the color buffers.
- **glLoadIdentity( ):**the current matrix with the identity matrix.
- **glMatrixMode(mode):** Sets the current matrix mode, mode can be GL_MODELVIEW, GL_PROJECTION or GL_TEXTURE.
- **Void glutInit (int *argc, char**argv):** Initializes GLUT, the arguments from main are passed in and can be used by the application.
- **Void glutInitDisplayMode (unsigned int mode):** Requests a display with the properties in mode. The value of mode is determined by the logical OR of options including the color model and buffering.
- **Void glutInitWindowSize (int width, int height):** Specifies the initial position of the top left corner of the window in pixels
- **Int glutCreateWindow (char *title):** A window on the display. The string title can be used to label the window. The return value provides references to the window that can be used when there are multiple windows.
- **Void glutMouseFunc(void *f(int button, int state, int x, int y):**Register the mouse callback function f. The callback function returns the button, the state of button after the event and the position of the mouse relative to the top-left corner of the window.
- **Void glutKeyboardFunc(void(*func) (void)):** This function is invoked when keyboard keys are pressed.
- **Void glutDisplayFunc (void (*func) (void)):** Register the display function func that is executed when the window needs to be redrawn.
- **glut PostReDisplay ( ) :**which requests that the display callback be executed after the current callback returns.
- **Void MouseFunc (void (*func) void)):** This function is invoked when mouse keys are pressed.
- **Void glutMainLoop ():** Cause the program to enter an event-processing loop. It should be the last statement in main function.

Result and Snapshots:

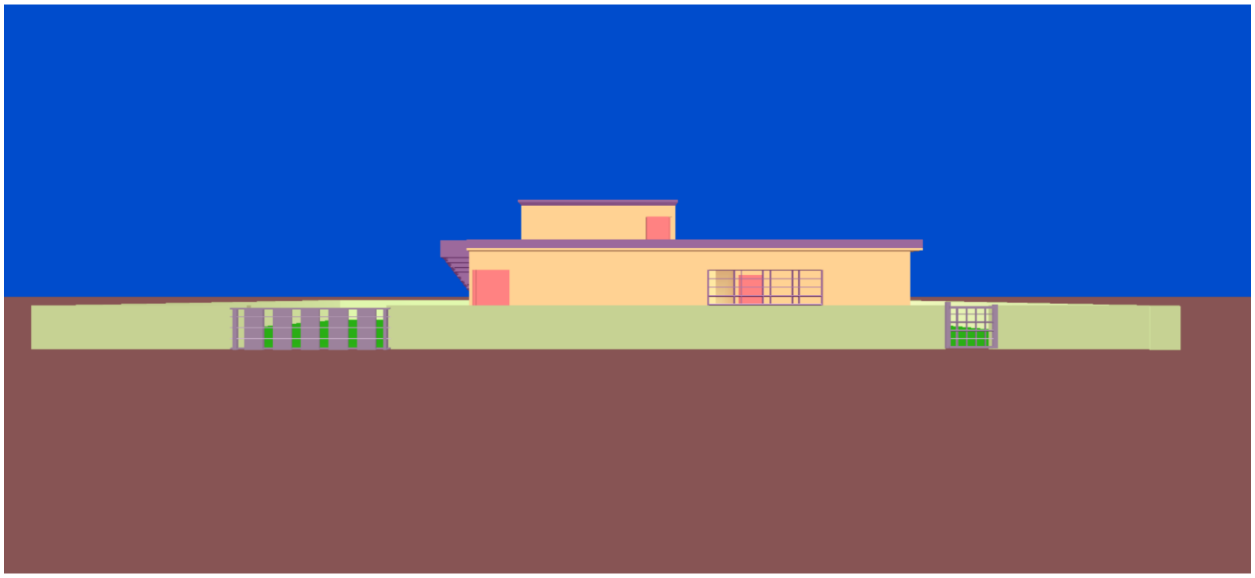


Figure: After execution the code.

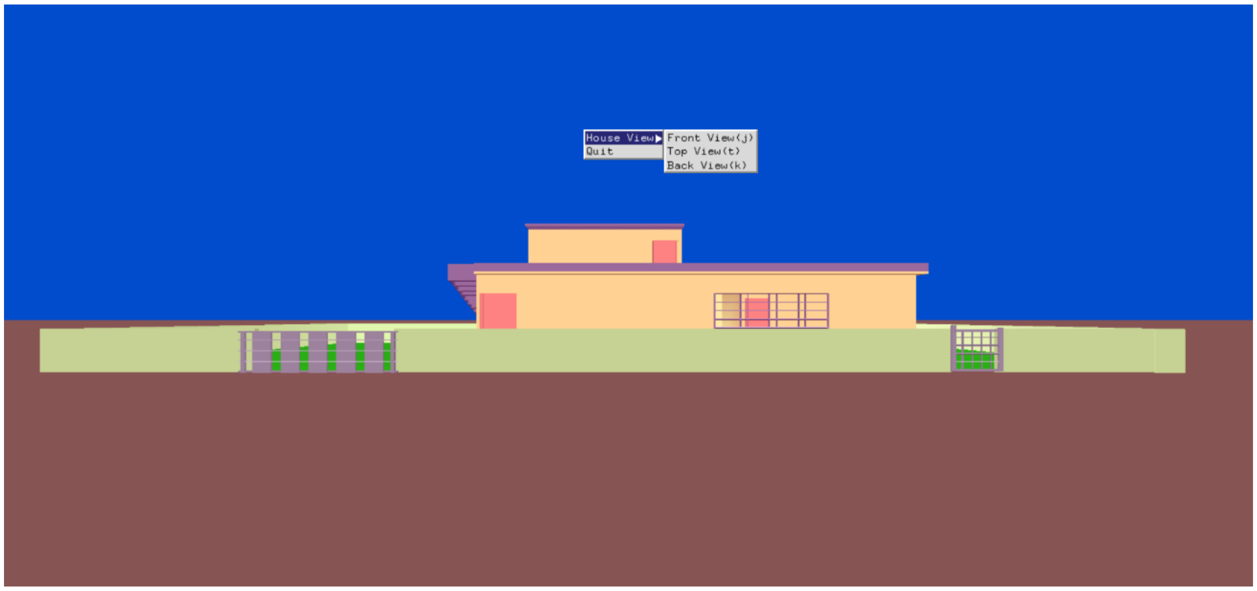


Figure: After Right click, showing the options.

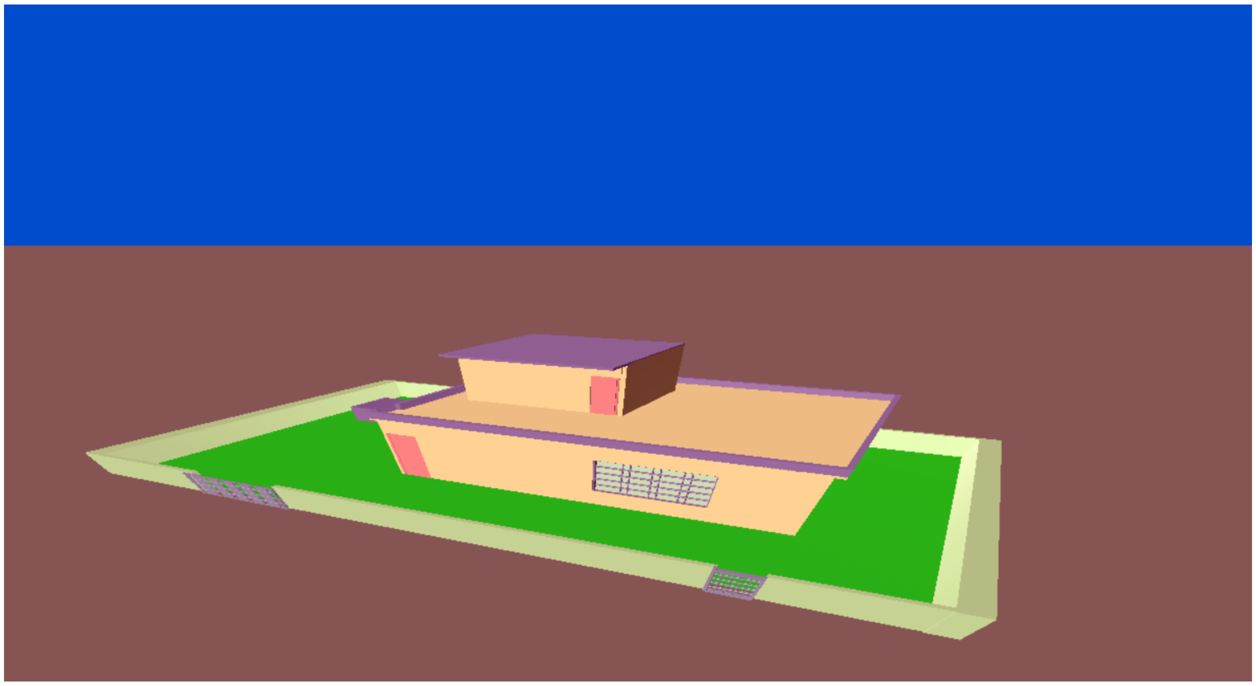


Figure: Top View of the house.

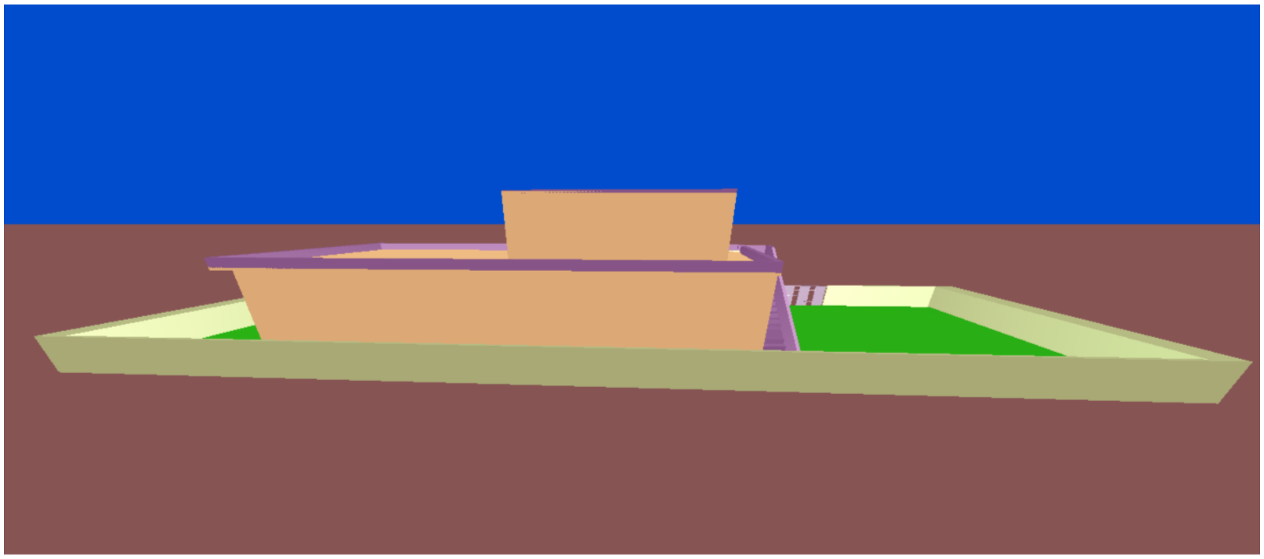


Figure: Back View of the house.

CONCLUSION:

In conclusion, the project on 3D house implemented in the context of our computer graphics course has been a valuable learning experience. Through this project, we have gained practical insights into various fundamental concepts and techniques in computer graphics.

First and foremost, we have explored the fundamentals of 3D modeling and rendering using OpenGL and GLUT libraries. We have learned how to create and manipulate 3D objects, apply textures and materials, and implement lighting and shading effects to enhance realism. Moreover, this project has provided us with hands-on experience in implementing user interaction mechanisms, such as camera control and menu systems. We have learned how to handle keyboard input and utilize special keys to navigate through the 3D environment, offering a dynamic and interactive user experience.

Furthermore, the project has highlighted the importance of modularity and code organization in graphics programming. By breaking down the implementation into smaller, manageable functions, we have improved code readability, maintainability, and scalability. This modular approach has also facilitated the incorporation of additional features and enhancements, paving the way for future development.

FUTURE ENHANCEMENTS:

1. **Texture Mapping:** Implement texture mapping to add realistic surface textures to the walls, floor, and other elements of the 3D house. This would involve applying bitmap images or procedural textures to the surfaces to add details such as brick patterns, wood grain, or tiled floors.
2. **Advanced Lighting Effects:** Explore more advanced lighting techniques such as ambient occlusion, global illumination, or HDR rendering to enhance the realism of the scene. Experiment with different types of light sources, such as spotlights or area lights, to create dynamic lighting scenarios.
3. **Animation:** Introduce animation to the 3D house project, allowing for interactive elements such as opening doors, moving objects, or changing lighting conditions. This could involve implementing keyframe animation or skeletal animation techniques to bring the environment to life.
4. **Dynamic Environments:** Extend the project to create dynamic environments with changing weather conditions, day-night cycles, or seasonal variations. This would require implementing procedural generation techniques or integrating real-time simulation systems

LIMITATIONS:

1. **Rendering Complexity:** The rendering technique used in the project may struggle to handle highly detailed scenes with complex geometry and textures. This could limit the project's ability to accurately represent intricate architectural designs or realistic environments.
2. **Realism:** While the project aims to create a 3D representation of a house, the level of realism may be limited compared to professional architectural visualization software.
3. **Interactivity**: The project's interactivity may be limited to basic navigation and exploration, with minimal support for interactive elements such as opening doors, turning on lights, or moving objects within the scene. This could reduce the project's appeal for users seeking more immersive experiences.
4. **Scalability:** Scaling the project to handle larger or more complex environments may present challenges, particularly in terms of rendering performance and memory usage. This could limit the project's suitability for applications requiring expansive virtual worlds or detailed architectural simulations.

References:

1. Wikipedia
2. ChatGPT OpenAI
3. https://vtupulse.com/cgv-mini-projects/3d-house-computer-graphics-project-in-opengl-source-code/
4. https://www3.ntu.edu.sg/home/ehchua/programming/opengl/CG_Examples.html