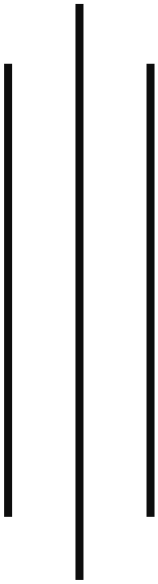
Tribhuvan University
Institute of Engineering
Pulchowk Engineering Campus

A Project Report on

3D Polygoneer



Submitted by:
Sandip Katel(078BCT077)
Saphal Rimal(078BCT078)
Sharad Pokharel(078BCT083)
Sijan Joshi(078BCT087)

Submitted to:
Department of Electronics and Computer Engineering
Pulchowk Engineering Campus, IOE Pulchowk,
Lalitpur

30th July, 2024

Abstract

This project presents the development of a sophisticated computer graphics application designed to facilitate the creation and manipulation of 3D objects. The primary objective of the application is to provide users with an intuitive and versatile tool for drawing 3D objects and applying various transformations such as rotation, scaling, and translation.

One of the core features of the application is its ability to render objects from multiple viewpoints, including both perspective and axonometric projections. Perspective projection allows for realistic visualization by simulating how objects appear to the human eye, with parallel lines converging at a vanishing point. In contrast, axonometric projection offers a more technical and scaled view, where the dimensions along the axes are preserved, and the objects do not appear distorted.

The application is built upon fundamental principles of computer graphics, incorporating algorithms for object representation, transformation matrices, and projection techniques. The user interface is designed to be accessible, enabling users to interact with the 3D environment seamlessly. Users can draw basic geometric shapes, manipulate these models in real time, and switch between different projection views.

Acknowledgment

It is a genuine pleasure to express our deep thanks and gratitude to Electronics and Computer Department which granted us this platform to make a graphics project. Our appreciation to **Asst. Prof. Mrs. Shanti Tamang**, Department of Electronics and Computer Engineering, our computer graphics teacher whose contribution in stimulating suggestions and encouragement helped us a lot for initiating this project. Her dedication and keen interest to help her students has been solely responsible for this work. We are also thankful to **Asst. Prof. Mrs. Bibha Sthapit ma'am**, our lab instructor, who inspires us constantly for making a project. Her scholarly advice and timely approach have helped us a great deal to begin this task.

Furthermore, we would also like to acknowledge with much appreciation the crucial role of the library of Pulchowk Campus, which provides us the permission to use all required equipment and the necessary materials. Pulchowk Campus family is to be thanked profusely for providing us a platform to grow and ultimately develop as an educated citizen. We are extremely thankful to our friends and family for becoming a source of constant encouragement and invaluable assistance for this process.

Table of Contents

1. Introduction

1.1 Background and Motivation

Computer graphics is a rapidly evolving field that plays a crucial role in various industries, including entertainment, design, simulation, and education. The ability to create, manipulate, and visualize 3D objects is fundamental to these applications. This project aims to develop a versatile computer graphics application that allows users to draw and transform 3D objects while viewing them from different projections.

1.2 Objectives of the Project

The primary objectives of this project are:

- To design and implement a user-friendly application for creating and manipulating 3D objects.
- To provide tools for performing geometric transformations, such as rotation, scaling, and translation.

1.3 Scope of the Project

The scope of this project includes the following key aspects:

- Development of a graphical user interface (GUI) that allows users to draw and edit basic 3D shapes.
- Implementation of transformation algorithms to modify the objects in real time.
- Integration of projection techniques to visualize objects from various viewpoints.
- Testing and validation of the application to ensure accuracy and usability.
- Provision of a user guide to assist users in utilizing the application's features effectively.

This project is designed to serve as both an educational tool for students and a foundation for more advanced applications in fields such as computer-aided design (CAD), game development, and virtual reality.

2. Theoretical Background

2.1 Tkinter

Tkinter is the standard GUI (Graphical User Interface) library for Python. It provides a powerful object-oriented interface to the Tk GUI toolkit, which is part of the standard Python distribution. Tkinter is widely used for developing desktop applications due to its simplicity and ease of use. It enables the creation of a variety of GUI applications that are both interactive and user-friendly.

2.1.1 Key Features of Tkinter

1. **Widgets:** Tkinter offers a variety of widgets such as buttons, labels, frames, text boxes, and canvas, which can be used to build complex GUI applications. Each widget is highly customizable, allowing developers to modify attributes such as size, color, font, and behavior.
2. **Geometry Management:** Tkinter provides geometry managers like pack, grid, and place to control the layout of widgets within the window. This flexibility allows developers to design interfaces that are both functional and aesthetically pleasing.
3. **Event Handling:** Tkinter supports event-driven programming, enabling the application to respond to user actions such as mouse clicks, key presses, and other events. Event handlers can be easily associated with widgets to define their behavior.
4. **Canvas Widget:** One of the most powerful features of Tkinter is the canvas widget, which is used for drawing shapes, plotting graphs, and creating custom widgets. The canvas widget supports various drawing primitives such as lines, circles, polygons, and text, making it ideal for applications like a polygon generator.

In the 3D Polygoneer project**,** Tkinter is used to create a graphical user interface that allows users to interact with the application and customize the properties of the generated polygons. Below are some specific uses of Tkinter in this project:

1. **Input Fields:** Tkinter entry widgets are used to input parameters such as the number of sides, base radius, top radius, and height of the polygon. These inputs are then used by the application to generate the 3D shape.
2. **Sliders and Color Choosers:** Tkinter provides sliders (Scale widget) for adjusting the values of parameters and color choosers for selecting the color of the polygon. These widgets provide a user-friendly way to modify the appearance and dimensions of the polygon.
3. **Canvas for Rendering:** The canvas widget is utilized to draw and display the generated polygon. By leveraging the canvas's drawing capabilities, the application can render complex shapes and provide real-time updates as parameters are changed.

2.2 3D Transformation

In computer graphics, 3D transformations are essential operations that allow the manipulation of objects in three-dimensional space. These transformations enable the positioning, orientation, and scaling of objects to create a desired scene or animation. The fundamental types of 3D transformations include scaling, rotation, translation, and shearing.

2.2.1 Scaling

Scaling is a transformation that changes the size of an object. It can be uniform or non-uniform. Uniform scaling uses the same scaling factor for all three dimensions (x, y, z), thereby preserving the object's proportions. Non-uniform scaling applies different scaling factors to each dimension, which can alter the object's shape.

The scaling transformation is represented by a scaling matrix *S*:

$$S = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where s_x, s_y, and s_z are the scaling factors along the x, y, and z axes, respectively. When this matrix multiplies a point (x, y, z, 1) in homogeneous coordinates, the point's coordinates are scaled accordingly.

2.2.2 Rotation

Rotation involves rotating an object around a specific axis (x, y, or z) by a certain angle. Rotation transformations are critical for orienting objects correctly within a scene. The rotation matrices for rotating an object around the x, y, and z axes by an angle are as follows:

For rotation about the x-axis:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For rotation about the y-axis:

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For rotation about the z-axis:

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.2.3 Translation

Translation moves an object from one location to another within 3D space. This transformation is described by a translation vector (t_x, t_y, t_z), which specifies the distances to move the object along the x, y, and z axes. The translation matrix T is:

$$T = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.3 Axonometric View/Object

Axonometric projection is a type of orthographic projection used in computer graphics and technical drawing to represent three-dimensional objects in two dimensions. Unlike perspective projection, axonometric projection does not have a vanishing point, meaning parallel lines remain parallel and objects do not appear smaller as they recede into the

distance. This method is useful for maintaining the scale and proportions of objects without distortion, making it ideal for technical and engineering drawings.

There are three primary types of axonometric projections: isometric, dimetric, and trimetric projections.

2.3.1 Isometric Projection

In isometric projection, the three coordinate axes (x, y, and z) are equally foreshortened and the angles between any two of the axes are equal, each being 120 degrees. This results in a visually balanced and aesthetically pleasing representation of the object. The scale along each axis is the same, which simplifies measurements and construction.

2.3.2 Dimetric Projection

Dimetric projection is a form of axonometric projection in which two of the three axes have the same scale, while the third axis has a different scale. This type of projection allows for a greater emphasis on one plane of the object, providing a clearer view of certain features while still maintaining some degree of foreshortening.

2.3.3 Trimetric Projection

Trimetric projection is the most general form of axonometric projection where all three axes are scaled differently. This allows for the most flexible and accurate representation of an object's dimensions but can be more complex to construct and interpret due to the different scaling factors.

2.4 Illumination

Illumination in computer graphics refers to the simulation of light interactions with objects to create realistic scenes. Effective illumination models consider how light behaves when it strikes surfaces, resulting in various visual effects. Proper illumination enhances the realism and depth of 3D objects, making them appear more lifelike. There are three primary components of illumination: ambient light, diffuse reflection, and specular reflection.

2.4.1 Ambient Light

Ambient light represents the general illumination that is present in an environment. It is the light that has been scattered and reflected so many times that it appears to come from all directions, resulting in a uniform lighting effect.

1. **Uniform Lighting:** Ambient light is evenly distributed, ensuring that all parts of the scene receive some level of illumination.
2. **No Specific Source:** Unlike other types of light, ambient light does not originate from a specific source. It is a global light that affects the entire scene equally.

3. **Soft Shadows:** Since ambient light is uniform and non-directional, it does not create distinct shadows or highlights on objects.

Ambient light is used to ensure that no part of a scene is completely dark. It provides a base level of illumination, allowing other types of light to build upon it to create more complex lighting effects. In a polygon generator, ambient light ensures that the generated polygon is always visible, even if other light sources are not present

2.4.2 Diffuse Reflection

Diffuse reflection occurs when light strikes a rough or matte surface and is scattered uniformly in all directions. This type of reflection is responsible for the basic color and brightness of objects.

1. **Uniform Scattering:** Light is scattered evenly, resulting in a uniform appearance from all viewing angles.
2. **Surface Dependent:** The intensity of the diffuse reflection depends on the angle between the light source and the surface normal (a perpendicular vector to the surface).
3. **Lambert's Cosine Law:** The intensity is proportional to the cosine of the angle between the light direction and the surface normal.

In the context of this project, diffuse reflection is used to calculate the base color of the polygon surfaces based on the light source and the orientation of the surfaces.

2.4.3 Specular Reflection

Specular reflection occurs when light reflects off a smooth, shiny surface, creating a highlight. This type of reflection is highly directional and depends on the viewer's position relative to the light source and the surface.

1. **Directional Reflection:** Specular reflection creates bright spots, known as specular highlights, which move with the viewing angle.
2. **Surface Smoothness:** The intensity and sharpness of the specular reflection depend on the smoothness of the surface. Smoother surfaces produce sharper and more defined highlights.
3. **Phong Reflection Model:** The intensity of specular reflection can be calculated using the Phong reflection model, which considers the angle between the reflected light direction and the viewer's direction.

Specular reflection adds realism by simulating the shiny appearance of surfaces. It enhances the perception of material properties and surface texture. In this project, specular reflection is used to create highlights on the polygon surfaces, making them appear more three-dimensional and visually appealing.

2.5 Window Port and View Port

In computer graphics, the concepts of window port and view port are essential for the process of mapping a scene from world coordinates to device coordinates. This process involves defining which part of the scene will be displayed (window port) and where it will be displayed on the output device (view port).

2.5.1 Window Port

The window port, or window, is a rectangular area in world coordinates that defines the part of the scene that will be visible. Essentially, it acts as a "window" through which a portion of the entire scene is viewed. By adjusting the window port, different parts of the scene can be magnified, reduced, or panned across.

Mathematically, if we denote the coordinates of the window port by (x_{min}, y_{min}) and (x_{max}, y_{max}) these represent the lower-left and upper-right corners of the window port, respectively.

2.5.2 View Port

The view port is the rectangular area on the output device (such as a computer screen) where the contents of the window port are displayed. It is defined in device coordinates, which are typically pixels for screen displays. The view port specifies the location and size of the display area on the device.

The coordinates of the view port are often given by (x'_{min}, y'_{min}) and (x'_{max}, y'_{max}) representing the lower-left and upper-right corners of the view port, respectively.

2.5.3 Mapping from Window Port to View Port

To display the scene correctly, the coordinates in the window port need to be mapped to the view port. This involves a linear transformation that scales and translates the window port coordinates to fit within the view port. The transformation can be expressed as:

$$x' = x'_{\min} + \left(\frac{x - x_{\min}}{x_{\max} - x_{\min}} \right) (x'_{\max} - x'_{\min})$$

$$y' = y'_{\min} + \left(\frac{y - y_{\min}}{y_{\max} - y_{\min}} \right) (y'_{\max} - y'_{\min})$$

where (x,y) are the coordinates in the window port and (x',y') are the corresponding coordinates in the view port.

2.6 Viewing Reference Point (VRP)

The Viewing Reference Point (VRP) is a crucial concept in the process of viewing transformation in computer graphics. It defines the position of the viewer or camera in the world coordinate system. The VRP, along with other parameters such as the view plane normal and the view up vector, helps in defining the view coordinate system, which is used to transform the 3D scene for proper visualization.

2.6.1 Role of VRP in the Viewing Process

1. **Defining the Viewing Position**: The VRP sets the location of the viewer or camera in the scene. It is the point from which the scene is observed.

2. **Transforming to View Coordinates**: The VRP, along with the view direction and view up vector, is used to define a coordinate transformation from world coordinates to view coordinates. This transformation aligns the view plane normal to the z-axis of the view coordinate system, making it easier to project the 3D scene onto a 2D plane.

3. **Clipping and Projection**: Once the scene is transformed into view coordinates, the window port is defined in this coordinate system. The VRP ensures that the window port is aligned correctly with the viewer's perspective. The transformed scene is then clipped to the window port and projected onto the view port.

3. Literature Review

3.1 Overview of Existing Technologies and Tools in Computer Graphics

Computer graphics is a field that has seen significant advancements over the years. Various technologies and tools have been developed to facilitate the creation and manipulation of 3D objects. This section provides an overview of the key technologies and tools that have influenced the development of our application.

1. **3D Modeling Software:** Programs such as Blender, AutoCAD, and Maya offer extensive features for creating and manipulating 3D models. These tools provide inspiration for the user interface and functionality of our application.
2. **Graphics Libraries and APIs:** Libraries such as OpenGL, DirectX, and WebGL are crucial for rendering 3D graphics. These libraries provide the foundation for implementing the rendering and transformation algorithms in our project.
3. **Game Engines:** Engines like Unity and Unreal Engine offer powerful tools for 3D graphics and game development. The principles and techniques used in these engines inform the design and implementation of our application's features.

3.2 Relevant Research and Prior Work

The development of our application is informed by existing research in the field of computer graphics. This section reviews key studies and prior work that have contributed to our understanding and implementation of 3D object manipulation and projection techniques.

1. **Geometric Transformations:** Research on algorithms for rotation, scaling, and translation of 3D objects provides the mathematical foundation for our application. Key papers and textbooks, such as "Computer Graphics: Principles and Practice" by Foley et al., are essential references.
2. **Projection Techniques:** Studies on different projection methods, including perspective and axonometric projections, guide the implementation of these features in our application. Important works include "Fundamentals of Computer Graphics" by Marschner and Shirley.
3. **User Interface Design for Graphics Applications:** Research on effective UI design for graphics applications ensures that our application is user-friendly and accessible. Relevant studies include usability guidelines and design principles for graphical software.

3.3 Comparison with Similar Projects or Applications

To highlight the uniqueness and value of our project, it is essential to compare it with similar existing applications. This section examines how our application stands out in terms of features, usability, and educational value.

- **Educational Tools:** Our application is designed with a strong educational focus, providing a hands-on tool for learning computer graphics concepts. Compared to other educational tools, our project offers a unique combination of drawing, transforming, and projecting 3D objects.
- **Professional Software:** While professional 3D modeling and graphics software offer extensive features, they can be complex and intimidating for beginners. Our application aims to bridge this gap by providing an intuitive and accessible interface for learning and experimentation

4. Methodology

This project is a 3D polygon generator developed using Tkinter in Python. Users can input parameters to generate a 3D polygon and adjust various graphical properties, including lighting and projection settings. The application also allows for object manipulation and scene management. The following methodology outlines the development and testing processes for this project.

4.1 Requirements Gathering

4.1.1 Functional Requirements:

1. Users can input parameters to define the polygon:
    a. Base radius
    b. Top radius
    c. Number of sides
    d. Height
2. Users can adjust lighting settings:
    a. Ambient lighting (Ka)
    b. Diffuse reflection (Kd)
    c. Specular reflection (Ks)
3. Users can adjust the View Reference Point (VRP) and projection settings.
4. Users can update the lighting of a selected object.
5. Users can delete a selected object.
6. Users can clear the screen to reset all data.

4.1.2 Non-Functional Requirements:

1. The application should be responsive and provide real-time updates to the 3D scene.
2. The UI should be intuitive and easy to use.
3. The application should handle invalid inputs gracefully.

4.2 Design

4.2.1 Architecture:

1. **Frontend:** Tkinter for the graphical user interface.
2. **Backend:** Python for processing user inputs and rendering the 3D polygon.

4.3 Components:

1. **Input Form:**
    a. Text fields and sliders for polygon parameters and lighting settings.
    b. Buttons for creating, updating, and deleting objects, and clearing the screen.
2. **3D Rendering Canvas:**
    a. A canvas area to display the generated 3D polygons.

b. Real-time rendering updates based on user inputs.

3. **Control Logic:**
    a. Functions to handle user inputs, update the 3D scene, and manage objects.

4.4 Implementation

1. **Setup Environment:**
    a. Install necessary libraries (Tkinter, NumPy, etc.).
    b. Create a basic Tkinter window with a canvas.
2. **Input Handling:**
    a. Implement text fields and sliders for user inputs.
    b. Add validation for input values.
3. **3D Polygon Generation:**
    a. Use mathematical formulas to generate vertices and faces of the polygon based on user inputs.
    b. Render the polygon on the canvas.
4. **Lighting and Projection:**
    a. Implement lighting calculations (ambient, diffuse, specular).
    b. Allow users to adjust VRP and other projection parameters.
5. **Object Management:**
    a. Enable selection of objects for updating or deleting.
    b. Implement object creation, updating, and deletion logic.
6. **Clear Functionality:**
    a. Implement a clear button to reset the canvas and input fields.

4.5 Context Diagram

A context diagram provides a high-level overview of the system, showing the system boundaries, external entities that interact with the system, and the major information flows between the entities and the system. Below is the context diagram for the Polygon Generator project:

Explanation:

1. **User Interface (Tkinter UI)**: The primary interface for user interaction, allowing users to input parameters for the polygon, adjust settings, manage objects, and perform actions like creating, updating, and deleting objects.

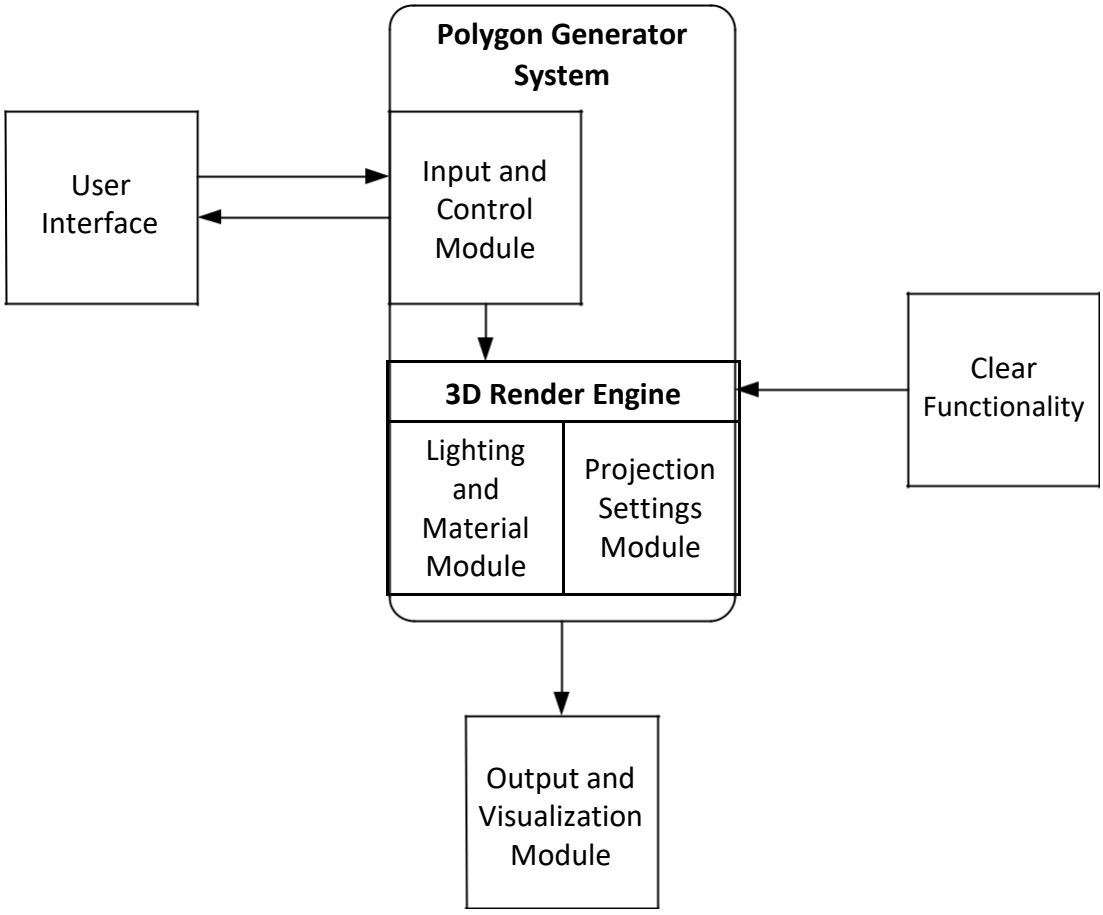


Fig 4.5: Context Diagram of 3D polygon generator

2. **Polygon Generator System**: The core system that processes user inputs, generates 3D polygons, and handles various functionalities:

    a. **Input & Control Module**: Manages user inputs and coordinates the flow of data and commands within the system.
    b. **3D Render Engine**: Processes the input data to generate and render the 3D polygons.
    c. **Lighting & Material Module**: Handles the settings for ambient lighting, diffuse reflection, and specular reflection, allowing users to adjust the appearance of the 3D objects.
    d. **Projection Settings Module**: Manages the view reference point (VRP) and other projection parameters to control how the 3D scene is viewed.

3. **Output & Visualization Module:** Manages the display of the generated 3D polygons and any updates based on user interactions.
4. **Clear Functionality**: Provides a mechanism to reset the canvas and all input fields, allowing users to start fresh.

5. Result

Some of the sample outcome as result of our project are attached below demonstrating the tool's ability to create basic 3D shapes and adjust their colors and lighting.

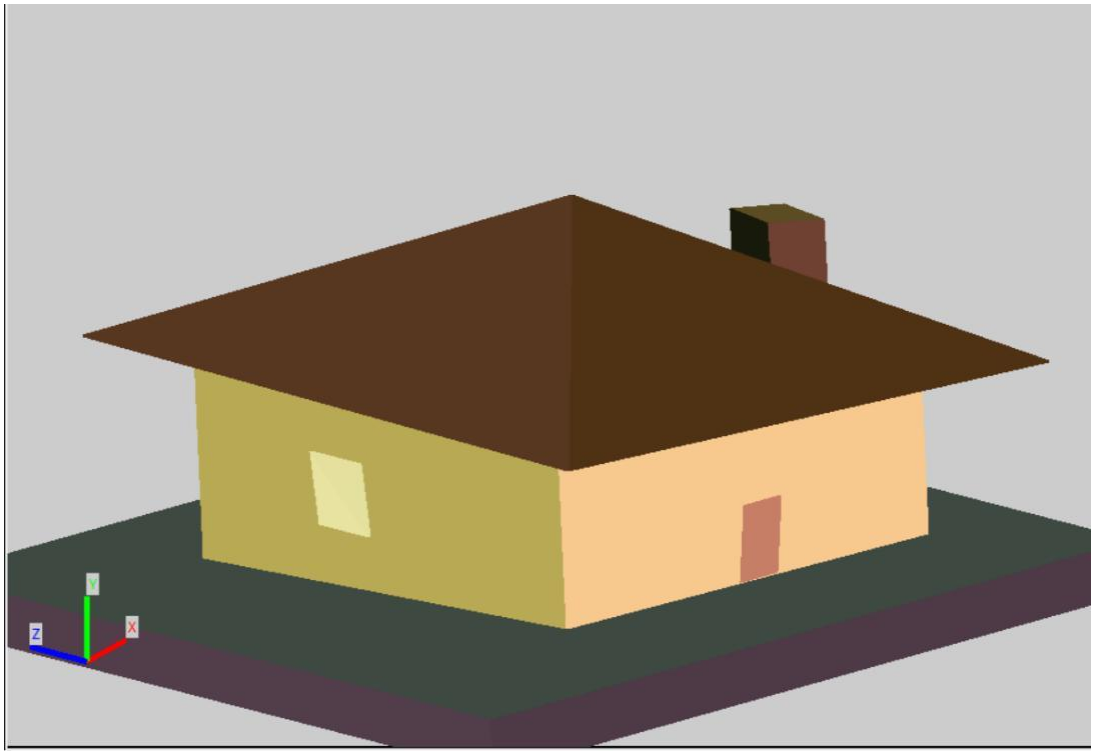


Fig 5.1: A sample hut modeled using multiple polygons and coloring

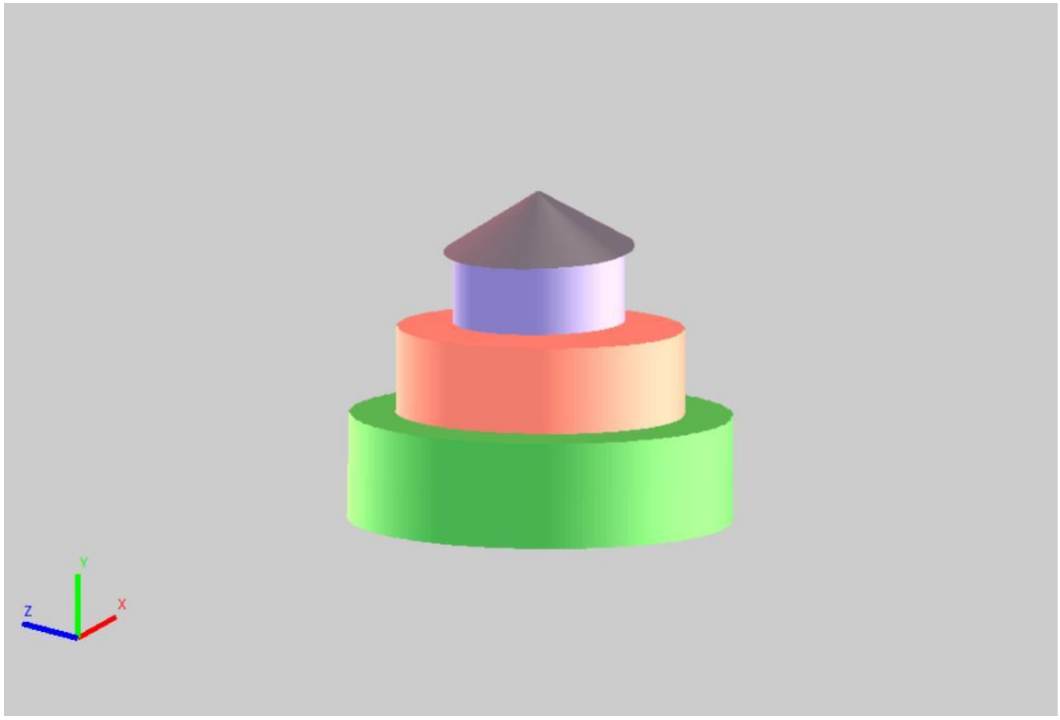


Fig 5.2: Stack of cylinders and cone using different colors and lighting

6. Applications

This project has various applications, some of the main application fields are below

1. **Drawing Tools:**

    a. **Visualization of Geometric Shapes**: This tool can be used by artists and designers to generate and visualize 3D geometric shapes such as pyramids and prisms.
    b. **Multiple Views**: Users can view the shapes from different angles by adjusting the view-up and viewing reference point parameters, providing top, side, front, or any other angle of view.

2. **Modeling:**

    a. **3D Model Generation**: This tool can be used in the field of 3D modeling to quickly generate basic 3D shapes that can be used as the foundation for more complex models.
    b. **Prototyping**: Designers can prototype simple 3D models to visualize their ideas before moving to more complex modeling software.

3. **Illumination:**

    a. **Lighting Studies**: The tool can be used to study the effects of different lighting conditions on 3D objects. Users can adjust ambient, diffuse, and specular lighting to see how these settings impact the appearance of the object.
    b. **Real-time Adjustments**: Users can make real-time adjustments to the lighting settings, providing an interactive way to learn about and experiment with illumination techniques.

4. **Education:**

    a. **Teaching Geometry and Computer Graphics**: This tool is valuable in educational settings for teaching concepts of geometry, computer graphics, and 3D rendering. Students can visualize geometric transformations and understand how different lighting settings affect 3D objects.
    b. **Interactive Learning**: Provides an interactive platform for students to experiment with and learn about 3D shapes, lighting, and rendering techniques.

5. **Architectural Visualization:**

    a. **Conceptual Design**: Architects can use this tool to create and visualize basic 3D shapes for conceptual designs and preliminary visualizations.
    b. **Lighting Analysis**: By adjusting the lighting settings, architects can study how different lighting conditions will affect the appearance of their designs.

7. Further Enhancement

As with any software project, there are numerous potential enhancements that can be implemented to improve the functionality, usability, and overall experience of the polygon generator. Here are some suggested future enhancements:

1. **Advanced 3D Modeling Features:** Introducing spheres, Boolean operations, extrusion, and beveling tools will enable the creation of more complex shapes. Adding lattice deformation and curve-based modeling techniques will allow users to manipulate polygons in more intricate ways.
2. **Collaboration and Export Options:** Integrating cloud storage will enable collaborative work on projects. Supporting various 3D file formats for export and allowing high-resolution image and animation exports will facilitate the use of models in other applications.
3. **Show Surface Development:** Implementing surface development capabilities will allow users to unfold 3D shapes into 2D representations, aiding in understanding the geometry and creating physical models.
4. **Show Intersection:** Adding the ability to show intersections between different polygons will enhance the analytical capabilities of the polygon generator. Users will be able to visualize and calculate the points or areas where shapes intersect, which is valuable for tasks such as collision detection in simulations, creating complex geometries in modeling, and performing Boolean operations.
5. **Include Other Transformations:** Incorporating additional transformation options such as shearing, tapering, and twisting will expand the range of modifications users can apply to their polygons. These transformations will enable more creative and complex design possibilities, allowing users to achieve specific aesthetic or functional requirements in their models.
6. **Enhanced User Interface:** Integrating a material editor and an advanced color picker will improve customization options. Usability can be further enhanced by adding undo/redo functionality and providing a library of pre-defined shapes, materials, and lighting setups for quick application.

8. Conclusion

The development of this computer graphics application has successfully met its core objectives by providing a tool for creating and manipulating 3D objects with a focus on educational value and user accessibility. The project incorporated essential concepts in computer graphics, such as geometric transformations and projection techniques, while leveraging Python's Tkinter and related modules for a functional and user-friendly interface.

The application enables users to draw, transform, and view 3D objects from multiple perspectives, demonstrating its versatility and practicality. Despite facing challenges related to mathematical accuracy, GUI design, and performance, the project has achieved significant milestones and offers a solid foundation for further development.

In summary, this project not only serves as an effective educational tool but also provides a basis for future enhancements and broader applications. The work accomplished paves the way for more advanced features and continued growth, reflecting the potential for further contributions to the field of computer graphics.

References

[1] **Foley, J. D., van Dam, A., Feiner, S. K., & Hughes, J. F.** (1996). *Computer Graphics: Principles and Practice*. Addison-Wesley.

[2] **Marschner, S., & Shirley, P.** (2015). *Fundamentals of Computer Graphics*. CRC Press.

[3] **Python Software Foundation.** (n.d.). *tkinter Documentation*. Retrieved from https://docs.python.org/3/library/tkinter.html

[4] **Gomez, J. C., & Gotsman, A.** (2016). *User Interface Design for Graphics Applications*. IEEE Computer Society.

[5] **Python Package Index (PyPI).** (n.d.). *tooltip Documentation*. Retrieved from https://pypi.org/project/tooltip/

User Guide

1. Create a single object

ii. Click on the object information
iii. Enter the number of sides of the faces (must be equal or greater than 3 else become 3 by default)
iv. Enter the base radius of the object
v. Enter the top radius of the object(if top radius is zero it will be pyramid/cone)
vi. Enter the color as Ka, Kd, Ks
vii. Now you can create the object

2. Transformation of Object

Select the object you want to apply transformation and use as following table

Key	Transformation
A	Move selected object left by 1px
D	Move selected object right by 1px
W	Move selected object up by 1px
S	Move selected object down by 1px
R	Scale down object along X-axis to 0.9 times
F	Scale up object along X-axis to 1.1 times
T	Scale down object along Z-axis to 0.9 times
G	Scale up object along Z-axis to 1.1 times
Y	Scale down object along Y-axis to 0.9 times
H	Scale Up object along Z-axis to 1.1 times
U	Rotate along X-axis by 5 degree
J	Rotate along X-axis by 5 degree
I	Rotate along Z-axis by 5 degree
K	Rotate along Z-axis by 5 degree
O	Rotate along Y-axis by 5 degree
L	Rotate along Y-axis by 5 degree

3. Creating multiple object

Once a object is created you can create as many object as you want following the same procedure.

4. Projection of object

i. Select the object you want to project.
ii. Select the type of projection you want.
iii. Select the view reference point as required.
iv. Enter the appropriate window and viewport limits as required

5. Deleting Object

You can select the object and click clicking “delete” button at bottom of the screen.to remove the object

6. Clearing the screen

You clear the whole screen and set every entry points to initial value and start form scratch by clicking “clear” button at bottom of the screen.