Application of OpenGL: Windmill

Aseem Yash
CCE
Manipal Institute of Technology
210953392

Aditya K. Jagadeesh
CCE
Manipal Institute of Technology
210953028

Shreyash B.S.
CCE
Manipal Institute of Technology
210953078

***Abstract*—This report presents a project that explores the use of the OpenGL graphics API and GLUT (OpenGL Utility Toolkit) libraries to create a 3D windmill scene. The primary objectives were to develop an understanding of drawing 3D shapes using OpenGL primitives and to create a simple yet visually appealing windmill model. The project involved constructing the various components of the windmill, such as the base, tower, and blades, by rendering polygons in 3D space.**

Keywords—OpenGL, 3D Graphics, Modeling, OpenGL Toolkit

I. INTRODUCTION

The **goal** of this project was to utilize OpenGL and the GLUT libraries to create a 3D windmill scene. By focusing on the fundamental tasks of drawing 3D shapes and arranging them in a cohesive manner, the project aimed to provide an introduction to the process of building 3D graphics applications using OpenGL.

The key **objectives** of the project included:

A. 3D Windmill Construction: The creation of the windmill's base, tower, and blades using various OpenGL primitive shapes, such as polygons, lines, and planes.

B. Camera Controls: The implementation of camera controls to allow the user to navigate and explore the 3D windmill scene.

C. Transformations and Rotations: The application of transformations, such as translation, rotation, and scaling, to position and orient the windmill components in the 3D space.

The key **contributions** in this project are Aseem Yash who designed the Windmill on paper helping the team come up with coordinates and techniques to use to draw various shapes, number of iterations to use for each shape. Aditya K. Jagadeesh who had the key role in coding these concepts using C++ programming language. Shreyash BS who perfected the code by adding rotation, translation and helped make OpenGL Utility Toolkit and OpenGL API Installations on Windows Software.

By successfully completing this project, we aimed to demonstrate our understanding of the basic OpenGL rendering pipeline and the GLUT library, as well as our ability to construct a simple yet visually appealing 3D scene using these tools.

II. METHODOLOGY

To achieve the goal of creating a 3D windmill scene using OpenGL and the GLUT library, the following methodology was followed:

A. Selection of OpenGL Library

After reviewing available options, the GLUT (OpenGL Utility Toolkit) library was chosen as the appropriate library to use for this project. GLUT provides a convenient set of functions and utilities for creating and managing OpenGL-based applications.

B. Windmill Shape Breakdown

The 3D windmill was broken down into its primary components, including the base, tower, and blades. Each of these elements was further analyzed to determine the best approach for rendering them using OpenGL primitives.

C. Coordinate Planning

On paper, the edge points for each curved and flat object were mapped to specific coordinate values. This step was crucial for ensuring the accurate placement and proportions of the windmill components within the 3D space.

D. Curved Object Rendering

For the curved objects, such as the windmill blades, an infinite number of `GL_LINES` were used to approximate the shape. The x and y coordinates were determined using the `cos(x)` and `sin(x)` functions, respectively, while the z coordinate was set to a fixed height `h`.

E. Flat Object Rendering

The flat objects, such as the base and tower, were rendered using `GL_POLYGON` with fixed coordinate points that were predetermined on paper.

F. Coding the Shapes in OpenGL

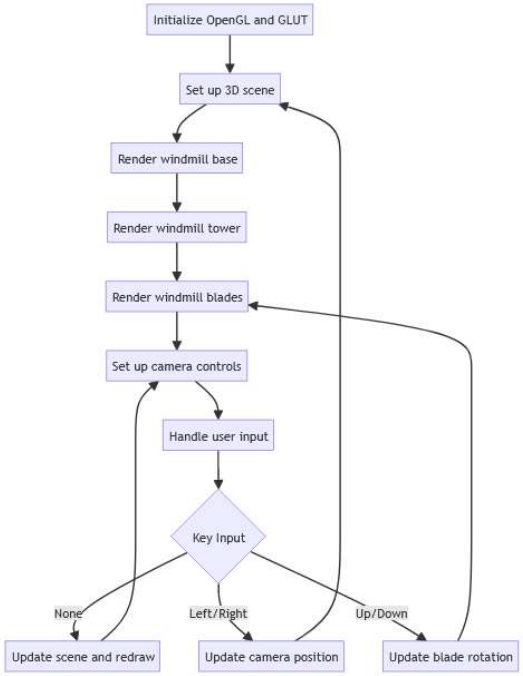
The planned polygons and lines were then implemented using OpenGL functions and commands to draw the individual components of the windmill.

G. Integration with GLUT

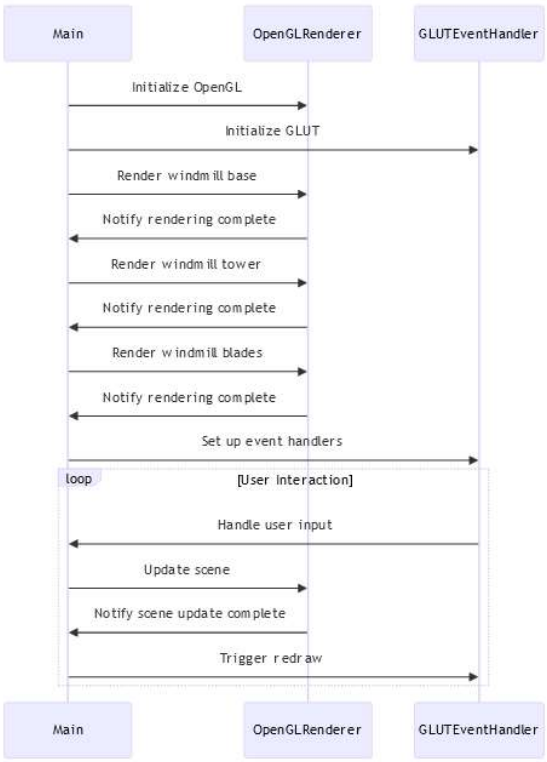
The OpenGL shapes were integrated with the GLUT library to provide additional functionality, such as camera controls and object transformations (rotation, translation, and scaling).

H. *Designing Of UML Diagrams*

1. UML ACTIVITY DIAGRAM



2. UML SEQUENCE DIAGRAM



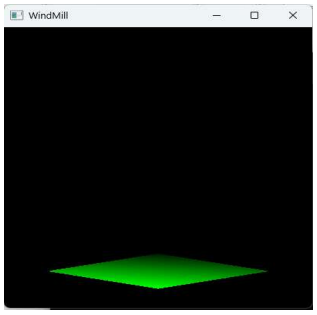
By following this methodology, the project aimed to leverage the capabilities of OpenGL and GLUT to create a visually appealing and interactive 3D windmill scene. The detailed planning and step-by-step implementation helped ensure the coherence and accuracy of the final 3D model.

III. RESULTS ANALYSIS

The results of the OpenGL windmill project are presented by showcasing the output of each individual element, breaking down the corresponding code segments that were implemented to achieve the desired 3D model.

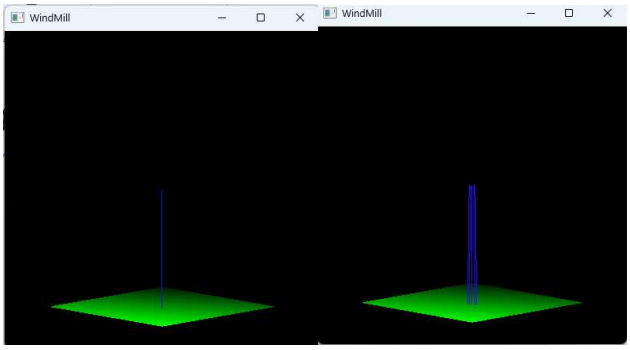
A. *The Ground*

```
for(int i=0;i<1000;++i){

   glBegin (GL_LINES);
   glColor3f  (0,.1+.3*i/1000.0,0);
   glVertex3f  (-50,  50,   0);
   glVertex3f  ( 0,   50,    0);
   glVertex3f  ( 0,   -50,   0);
   glVertex3f  ( 50,  50,    0);
   glEnd();
}
```



B. *The Shaft*

```
void Shaft(radius_bottom, radius_top, height)
{
 int steps = 1000;
 for(int i=0;i<steps;++i)
  {
  double temp = (2*PI/steps)*i;
  glBegin (GL_LINES);
  double c = .7 + cos(th)*.2; //2*pi*r
  glColor3f  (0.5, 0.5, 0.5); //Grey
  glVertex3f(r_top*cos(th),r_top*sin(th),0);
  glVertex3f(r_bot*cos(th),r_bot*sin(th),0);
  glEnd ();
  }
}
```

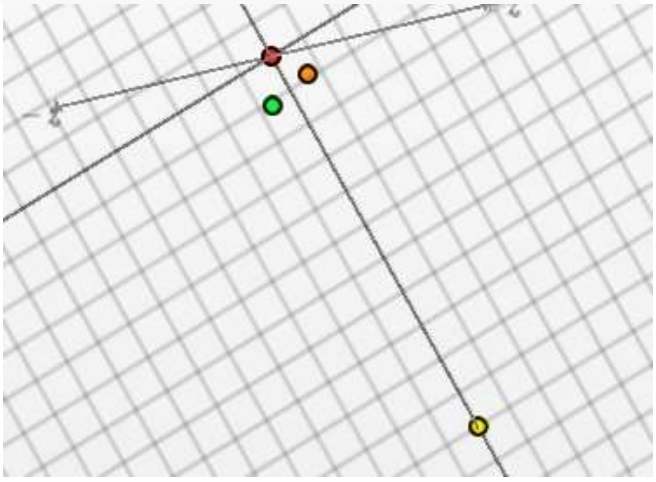


steps = 1 *steps = 100*

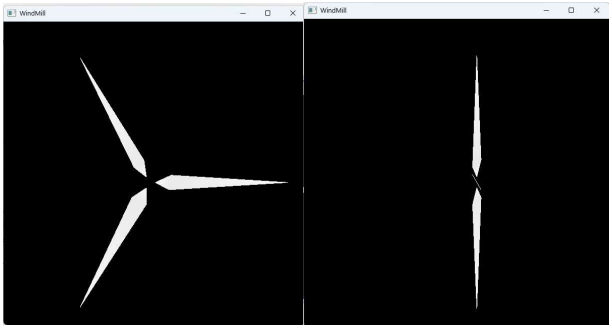
C. *The Wings*

1. *Front Frame*

```
glBegin(GL_Polygon);
        glColor3f(0.5,0.5,0.5);
        glVertex3f(x+2,   y+10,          z);
glVertex3f(x-2,    y+10                ,z+10);
glVertex3f(x,      y+50,               z+15);
glVertex3f(x+2,    y+40,               z+45);
glVertex3f(x-2,    y+55,               z+30);

glEnd();
```



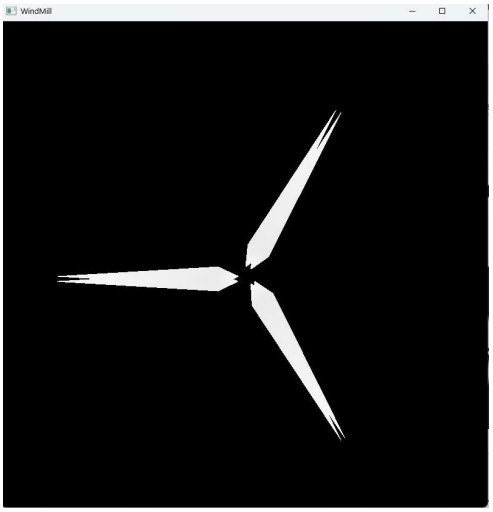
Coordinates Simulation



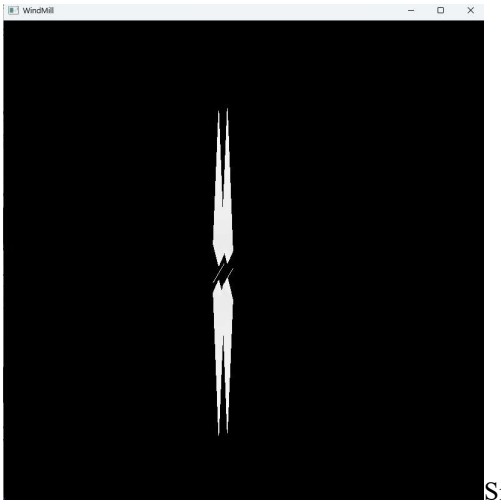
Front View Side View

2. *Back Frame*

Similar to Front Frame with an offset value set for all the coordinates in depth axis (x).



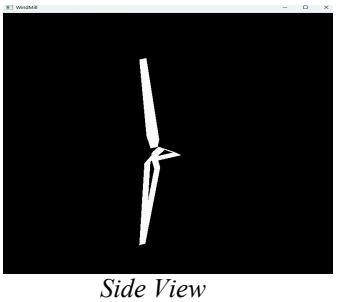
FrontView



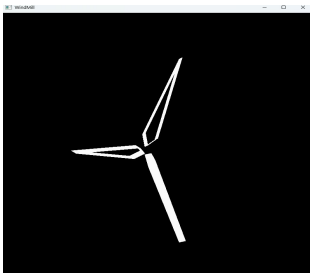
SideView

3. *Side Frames*

Fixed coordinates with coordination simulation on 3D Point Plotting software.



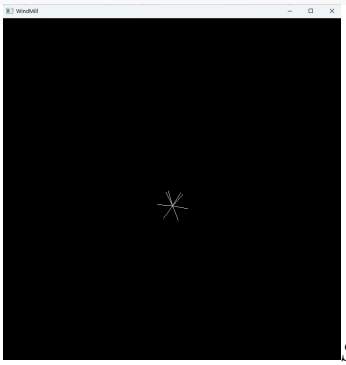
Side View



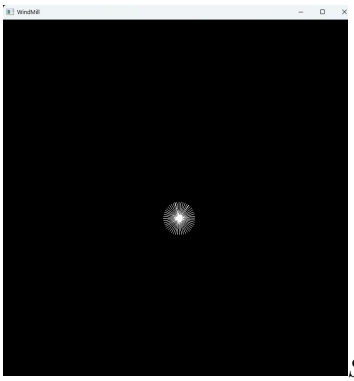
Front View

D. *The Nose*

```
void Circle(radius, center_z)
{
  int height = center_z;
  int steps = 1000;
  for(int i=0;i<steps;++i)
  {
    double ang = (360.0/steps)*i;
    glColor3f  (1,1,1);
    glBegin(GL_LINES);
    glVertex3f(rad*cos(ang),rad*sin(ang),0);
    glVertex3f(0,0,height);
    glEnd ();
  }
}
```

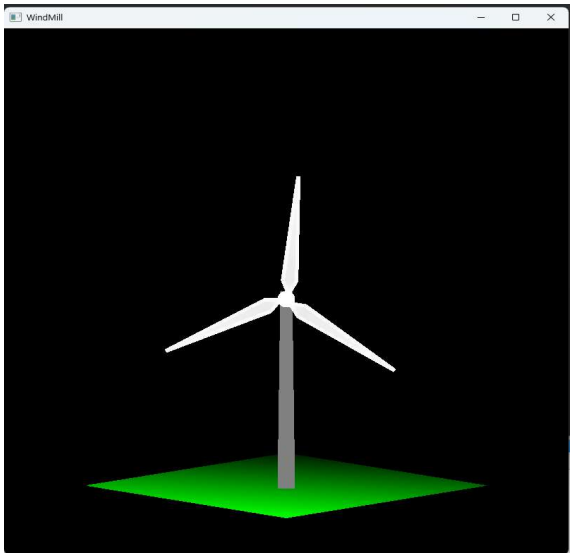


Steps = 10



Steps = 100

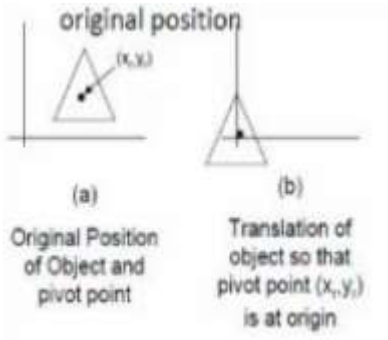
E. *Final 3D Shape Outcome*



F. *Adding GLUT Transformation Capabilities*

1. *Reset Identity*

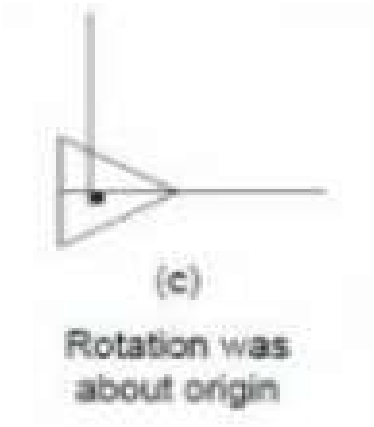
```
void resetIdentity()
{
   glLoadIdentity();
   glTranslatef(0.0,world_y_trans,0.0);
   glScalef(scale,scale,scale);
}
```



2. Transformation

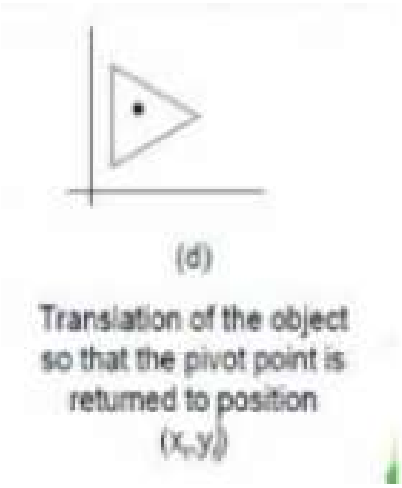
```
void rotate()
{
   ResetIdentity();
   double acc = stepsize*cos(ANGLE/180*PI);
   wing_pos += acc;

  glRotatef(0,Cos(wing_pos),Sin(wing_pos);
```

```
    RestoreIdentity();
    glutPostRedisplay();
}
```



3. *RestoreIdentity*

```
void restoreIdentity()
{
    RestoreIdentity();
    glutPostRedisplay();
}
```



.

IV. CONCLUSION AND FUTURE SCOPE

The OpenGL windmill project successfully demonstrated the ability to create a 3D scene using OpenGL primitives and the GLUT library. By breaking down the windmill into its core components, such as the base, shaft, and blades, we were able to render each element using a combination of lines and polygons. The integration of camera controls and object transformations allowed for the exploration and manipulation of the 3D environment, further enhancing the interactive nature of the project.

The implementation of this project has provided us with a deeper understanding of the OpenGL rendering pipeline and the fundamental techniques required to construct 3D models. The use of the GLUT library has also highlighted the benefits of leveraging existing tools and utilities to expedite the development process.

While the current project has achieved its primary objective of creating a basic 3D windmill scene, there are several areas of **future scope**:

1. **Texture Mapping**: Incorporate texture mapping to add more visual realism and detail to the windmill components.
2. **Advanced Lighting and Shading**: Explore more advanced lighting and shading techniques, such as the use of directional lighting,gradient coloring.
3. **User Interaction**: Expand the user interaction capabilities, allowing for more complex manipulations of the scene, such as zooming, panning, or selecting specific elements for further examination.
4. **Modularisation:** To create a library that can just input coordinates and create 3D shapes automatically

By addressing these potential areas for improvement, the OpenGL windmill project could be further developed into a more sophisticated and visually captivating 3D application, showcasing the versatility and power of the OpenGL graphics API.

REFERENCES

1. OpenGLUT Documentation. (n.d.). Retrieved from https://www.opengl.org/resources/libraries/glut/
2. FreeGLUT Documentation. (n.d.). Retrieved from http://freeglut.sourceforge.net/docs/api.php