Computer Graphics Lab Report

CSE4204

Submitted To :

Md. Tahmid Hasan
(Lecturer)

Submitted By :

Abrity Paul Chowdhury
ID : 0692130005101005
Batch : CSE17

Submission Date : April 06,2025

Moving Rocket

Code

```
/*
 * GLUT Shapes Demo
 *
 * Written by Nigel Stewart November 2003
 *
 * This program is a test harness for the sphere, cone
 * and torus shapes in GLUT.
 *
 * Spinning wireframe and smooth shaded shapes are
 * displayed until the ESC or q key is pressed. The
 * number of geometry stacks and slices can be adjusted
 * using the + and - keys.
 */

#include <windows.h> // This includes the Windows API header file, which contains declarations for all Windows functions, constants, and types.

#ifdef __APPLE__
#include <GLUT/glut.h> // If compiling on a Mac, this includes the GLUT header specific to Mac.
#else
#include <GL/glut.h> // If compiling on other platforms, this includes the standard GLUT header.
#endif

#include <stdlib.h> // Includes standard library functions, such as memory allocation, process control, and conversions.
#include <unistd.h> // Includes POSIX API functions, which provide access to the operating system API.
#include <math.h> // Includes mathematical functions, like sine, cosine, and square root.

static GLfloat spin = 0.0; // A global variable to store the current rotation angle of the shape.
static float tx = 0.0; // A global variable to store the translation amount along the x-axis.
static float ty = 0.0; // A global variable to store the translation amount along the y-axis.

//FOR CIRCLE
void DrawCircle(float cx, float cy, float rx,float ry, int num_segments)
{
glBegin(GL_TRIANGLE_FAN);
for(int ii = 0; ii < num_segments; ii++)
{
float theta = 2.0f * 3.1415926f * float(ii) / float(num_segments);//get the current angle

float x = rx * cosf(theta);//calculate the x component
float y = ry * sinf(theta);//calculate the y component

glVertex2f(x + cx, y + cy);//output vertex

}
glEnd();
}
```

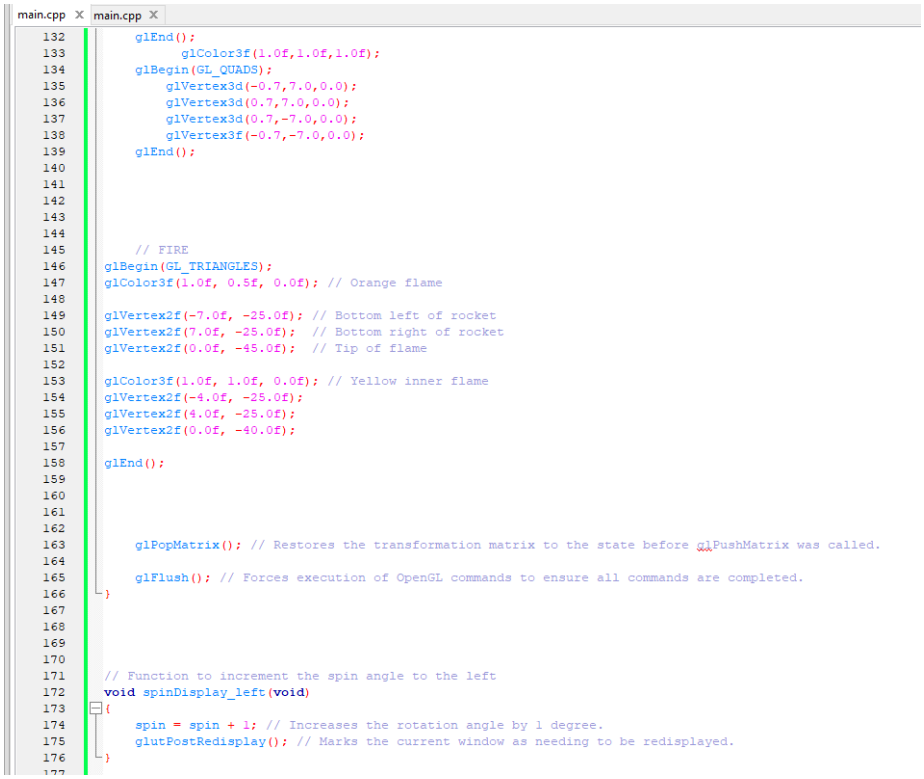
Figure1: code part 1

```
}

// Function to display the content on the screen

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT); // Clears the window to the current background color.


        //CLOUD

// Cloud 1
glColor3f(1.0, 1.0, 1.0);
DrawCircle(40, 25, 20, 10, 100);
DrawCircle(52, 27, 18, 10, 100);
DrawCircle(46, 35, 15, 10, 100);
DrawCircle(34, 30, 14, 9, 100);

// Cloud 2
DrawCircle(-40, 30, 20, 10, 100);
DrawCircle(-30, 34, 18, 10, 100);
DrawCircle(-38, 40, 15, 10, 100);
DrawCircle(-50, 36, 16, 9, 100);

// Cloud 3
DrawCircle(0, 60, 22, 10, 100);
DrawCircle(10, 63, 20, 10, 100);
DrawCircle(5, 70, 18, 9, 100);
DrawCircle(-8, 65, 14, 8, 100);

// Cloud 4
DrawCircle(-70, 55, 18, 9, 100);
DrawCircle(-60, 60, 16, 9, 100);
DrawCircle(-68, 66, 12, 8, 100);

// Cloud 5 (low altitude)
DrawCircle(50, -40, 22, 9, 100);
DrawCircle(60, -36, 20, 9, 100);
DrawCircle(56, -30, 18, 8, 100);

// Cloud 6 (low left)
DrawCircle(-55, -45, 20, 8, 100);
DrawCircle(-65, -42, 18, 9, 100);
DrawCircle(-60, -35, 15, 8, 100);

    glPushMatrix(); // Saves the current state of the transformation matrix.
```

Figure2: code part 2

```
glPushMatrix(); // Saves the current state of the transformation matrix.

glRotatef(spin, 0.0, 0.0, 1.0); // Rotates the matrix by 'spin' degrees around the z-axis (2D rotation).
glColor3f(0.2, 0.5, 0.5); // Sets the current color to white (RGB: 1.0, 1.0, 1.0).

glTranslatef(tx, ty, 0); // Translates the matrix by tx units in the x direction and ty units in the y direction.

glRectf(-10.0, -25.0, 10.0, 25.0); // Draws a rectangle centered at the origin with width and height of 50 units.

glBegin(GL_TRIANGLES);
glColor3f(1.0f, 1.0f, 0.0f); // Yellow triangle
glVertex2f(-10.0f, 25.0f);  // Bottom left corner (same as top-left of the rectangle)
glVertex2f(10.0f, 25.0f);   // Bottom right corner (same as top-right of the rectangle)
glVertex2f(0.0f, 45.0f);    // Top middle point
glEnd();

glBegin(GL_TRIANGLES);
glColor3f(1.0f, 1.0f, 0.0f); // Yellow triangle
glVertex2f(-10.0f, -5.0f);  // Bottom left corner (same as top-left of the rectangle)
glVertex2f(-30.0f, -35.0f);   // Bottom right corner (same as top-right of the rectangle)
glVertex2f(-10.0f, -25.0f);    // Top middle point
glEnd();

glBegin(GL_TRIANGLES);
glColor3f(1.0f, 1.0f, 0.0f); // Yellow triangle
glVertex2f(10.0f, -5.0f);  // Bottom left corner (same as top-left of the rectangle)
glVertex2f(30.0f, -35.0f);   // Bottom right corner (same as top-right of the rectangle)
glVertex2f(10.0f, -25.0f);    // Top middle point
glEnd();

glColor3f(0.7, 0.2, 0.3);

DrawCircle(0,0,7,8,100);

//WINDOWS
  glColor3f(1.0f,1.0f,1.0f);
glBegin(GL_QUADS);
    glVertex3d(-6.0,0.0,0.0);
    glVertex3d(6.0,0.0,0.0);
    glVertex3d(7.0,1.0,0.0);
    glVertex3f(-7.0,1.0,0.0);
glEnd();
      glColor3f(1.0f,1.0f,1.0f);
glBegin(GL_QUADS);
    glVertex3d(-0.7,7.0,0.0);
    glVertex3d(0.7,7.0,0.0);
```

Figure3: code part 3



```
main.cpp  ×  main.cpp  ×
132         glEnd();
133               glColor3f(1.0f,1.0f,1.0f);
134         glBegin(GL_QUADS);
135             glVertex3d(-0.7,7.0,0.0);
136             glVertex3d(0.7,7.0,0.0);
137             glVertex3d(0.7,-7.0,0.0);
138             glVertex3f(-0.7,-7.0,0.0);
139         glEnd();
140
141
142
143
144
145         // FIRE
146     glBegin(GL_TRIANGLES);
147     glColor3f(1.0f, 0.5f, 0.0f); // Orange flame
148
149     glVertex2f(-7.0f, -25.0f); // Bottom left of rocket
150     glVertex2f(7.0f, -25.0f);  // Bottom right of rocket
151     glVertex2f(0.0f, -45.0f);  // Tip of flame
152
153     glColor3f(1.0f, 1.0f, 0.0f); // Yellow inner flame
154     glVertex2f(-4.0f, -25.0f);
155     glVertex2f(4.0f, -25.0f);
156     glVertex2f(0.0f, -40.0f);
157
158     glEnd();
159
160
161
162
163         glPopMatrix(); // Restores the transformation matrix to the state before glPushMatrix was called.
164
165         glFlush(); // Forces execution of OpenGL commands to ensure all commands are completed.
166     }
167
168
169
170
171     // Function to increment the spin angle to the left
172     void spinDisplay_left(void)
173     {
174         spin = spin + 1; // Increases the rotation angle by 1 degree.
175         glutPostRedisplay(); // Marks the current window as needing to be redisplayed.
176     }
```

Figure4: code part 4

```c
void spinDisplay_left(void)
{
    spin = spin + 1; // Increases the rotation angle by 1 degree.
    glutPostRedisplay(); // Marks the current window as needing to be redisplayed.
}

// Function to decrement the spin angle to the right
void spinDisplay_right(void)
{
    spin = spin - 1; // Decreases the rotation angle by 1 degree.
    glutPostRedisplay(); // Marks the current window as needing to be redisplayed.
}

// Function to initialize the OpenGL environment
void init(void)
{
    glClearColor(0.0, 0.0, 1.0, 0.0); // Sets the clear color to red (RGB: 1.0, 0.0, 0.0, Alpha: 0.0).
    glOrtho(-100.0, 100.0, -100.0, 100.0, -1.0, 1.0); // Sets up a 2D orthographic viewing region with the given boundaries.
}

// Function to handle keyboard input
void my_keyboard(unsigned char key, int x, int y)
{
    switch (key) {
        case 'l': // If 'l' key is pressed
            spinDisplay_left(); // Call the function to spin left.
            break;
        case 'r': // If 'r' key is pressed
            spinDisplay_right(); // Call the function to spin right.
            break;
        case 's': // If 's' key is pressed
            glutIdleFunc(NULL); // Stop any ongoing animation.
            break;
        default:
            break;
    }
}

// Function to handle special key input (arrow keys)
void spe_key(int key, int x, int y)
{
    switch (key) {
        case GLUT_KEY_UP: // If up arrow key is pressed
            ty += 5; // Move the rectangle up by 5 units.
            PlaySound("Horn Honk-SoundBible.com-1162546405.wav", NULL, SND_ASYNC | SND_FILENAME); // Play a honk sound.
            glutPostRedisplay(); // Mark the window to be redisplayed.
```

Figure4: code part 5

```c
// Function to handle special key input (arrow keys)
void spe_key(int key, int x, int y)
{
    switch (key) {
        case GLUT_KEY_UP: // If up arrow key is pressed
            ty += 5; // Move the rectangle up by 5 units.
            PlaySound("Horn Honk-SoundBible.com-1162546405.wav", NULL, SND_ASYNC | SND_FILENAME); // Play a honk sound.
            glutPostRedisplay(); // Mark the window to be redisplayed.
            break;
        case GLUT_KEY_DOWN: // If down arrow key is pressed
            ty -= 5; // Move the rectangle down by 5 units.
            PlaySound("Horn Honk-SoundBible.com-1162546405.wav", NULL, SND_ASYNC | SND_FILENAME); // Play a honk sound.
            glutPostRedisplay(); // Mark the window to be redisplayed.
            break;
        case GLUT_KEY_RIGHT: // If right arrow key is pressed
            tx += 5; // Move the rectangle right by 5 units.
            PlaySound("Horn Honk-SoundBible.com-1162546405.wav", NULL, SND_ASYNC | SND_FILENAME); // Play a honk sound.
            glutPostRedisplay(); // Mark the window to be redisplayed.
            break;
        case GLUT_KEY_LEFT: // If left arrow key is pressed
            tx -= 5; // Move the rectangle left by 5 units.
            PlaySound("Horn Honk-SoundBible.com-1162546405.wav", NULL, SND_ASYNC | SND_FILENAME); // Play a honk sound.
            glutPostRedisplay(); // Mark the window to be redisplayed.
            break;
        default:
            break;
    }
}
```

Figure4: code part 6

```
// Main function to set up the GLUT environment and enter the event processing loop
int main(int argc, char *argv[])
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB); // Initialize the display mode with single buffering and RGB color model.
    glutInitWindowSize(500, 500); // Set the window size to 500x500 pixels.
    glutInitWindowPosition(100, 100); // Set the initial window position.
    glutCreateWindow("ROCKET BY ABRITY"); // Create a window with the title "mist".
    init(); // Call the initialization function.

    glutDisplayFunc(display); // Register the display callback function.
    glutKeyboardFunc(my_keyboard); // Register the keyboard callback function.
    glutSpecialFunc(spe_key); // Register the special key callback function.
    glutMouseFunc(my_mouse); // Register the mouse callback function.
    glutMainLoop(); // Enter the GLUT event processing loop.
    return 0; // Exit the program.
}
```

Figure4: code part 7

Output

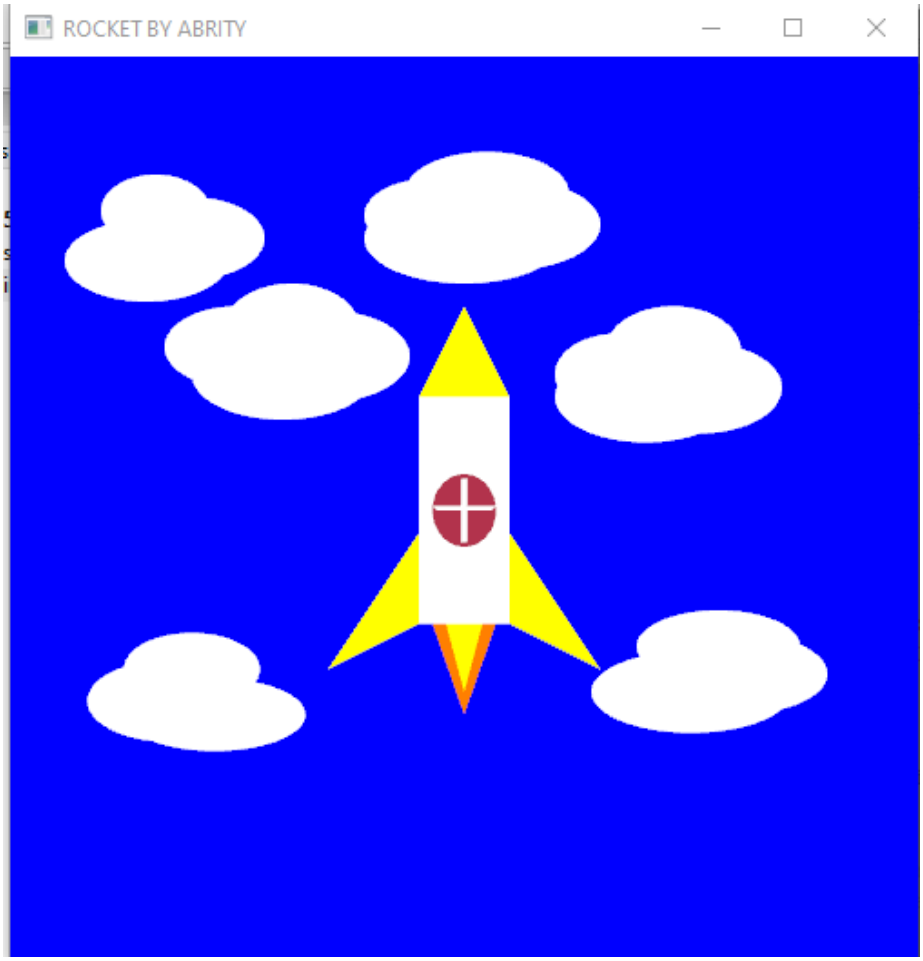


Figure5: Output 1

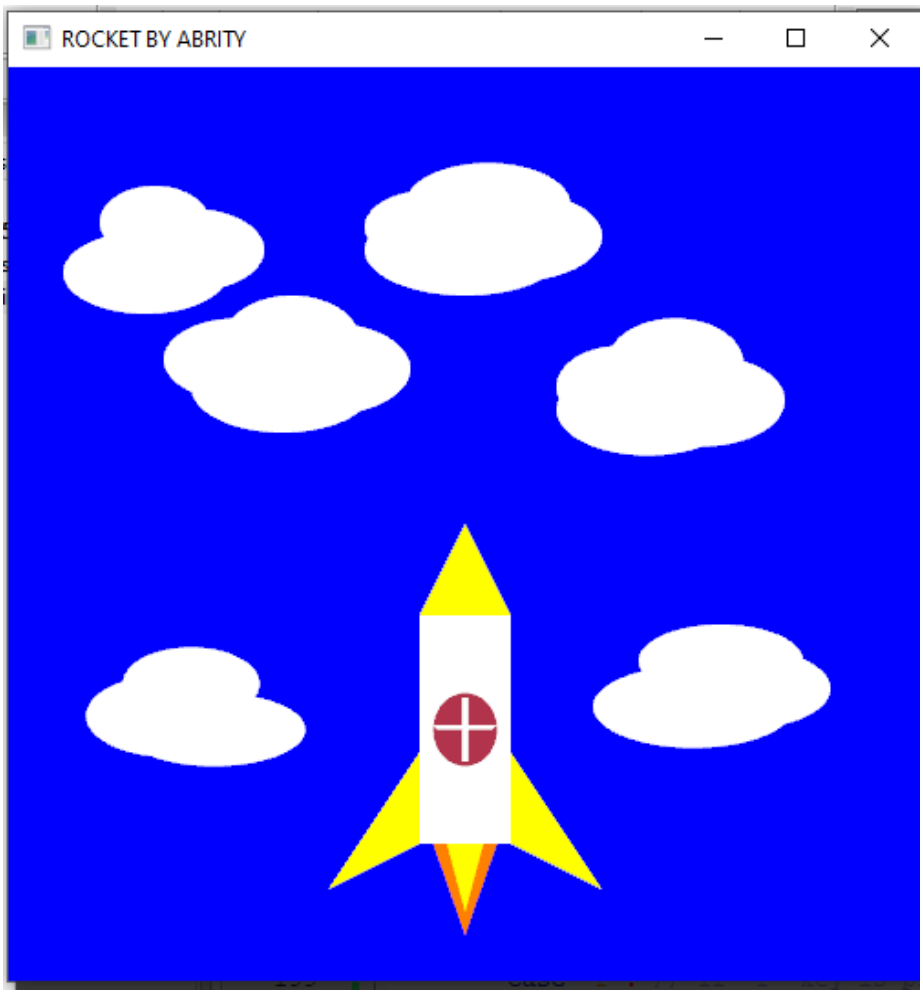


Figure5: Output 2

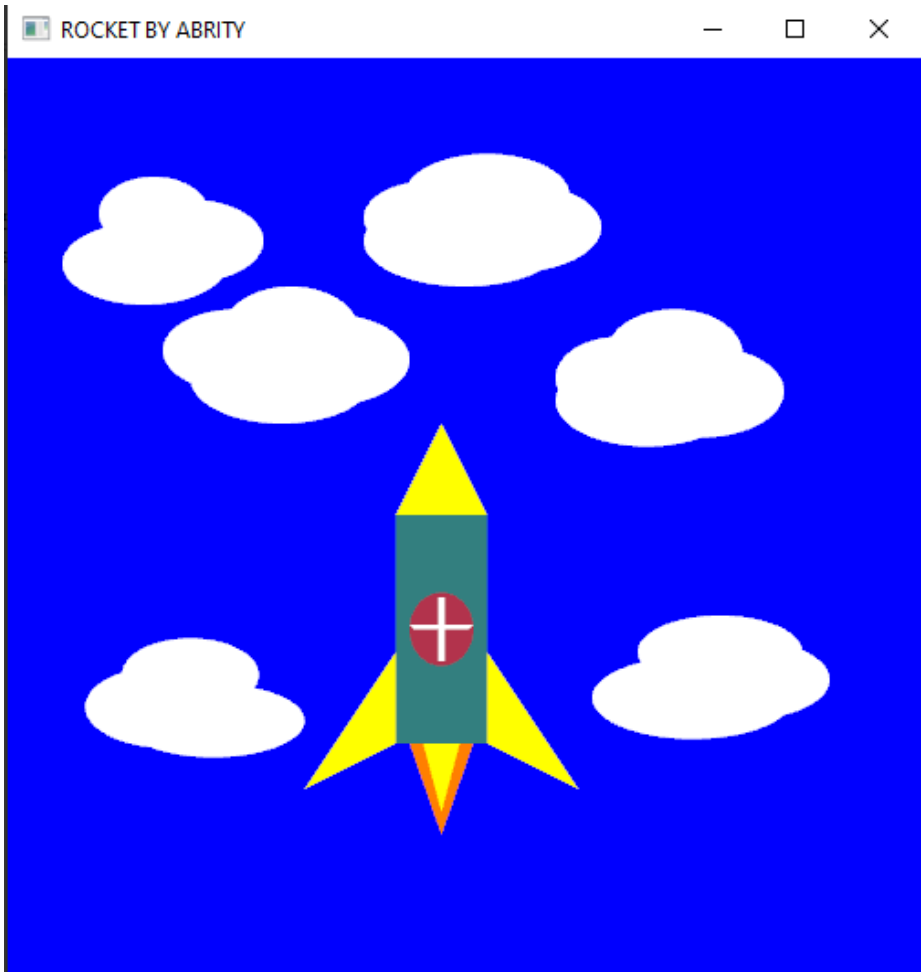


Figure5: Output 3