Department: Information Technology Engineering
Course: Computer Graphic
Name: Beun Bunleap

Lecture 02

1. Introduction

This assignment focuses on implementing and understanding **2D transformations**, specifically **translation** and **rotation**, using OpenGL. Translation involves moving an object from one position to another, while rotation involves turning an object around a fixed point. Both transformations preserve the shape and size of the object.

2. Theory

2.1. Translation

Definition: Translation moves an object from one position to another by adding displacement values (tx, ty) to its coordinates.

Matrix Representation:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- (x, y): Original position
- (x’, y’): New position after translation
- tx: Displacement along the x-axis.
- ty: Displacement along the y-axis.

Key Points:

- Translation is additive.
- Order of transition does not matter.
- Reversible by using negative displacement values.

2.2. Rotation

Definition: Rotation turns an object around a fixed point (usually the origin) by a specified angle 0.

Matrix Representation:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- θ: Rotation angle (positive for counterclockwise, negative for clockwise).
- (x, y): Original position.
- (x', y'): New position after rotation.

Key Points:

- Rotation is around the origin (0, 0).
- Order of transformations matters when combining with other transformations.
- A full rotation is 360°.

3. Implementation

3.1 Translation Demo

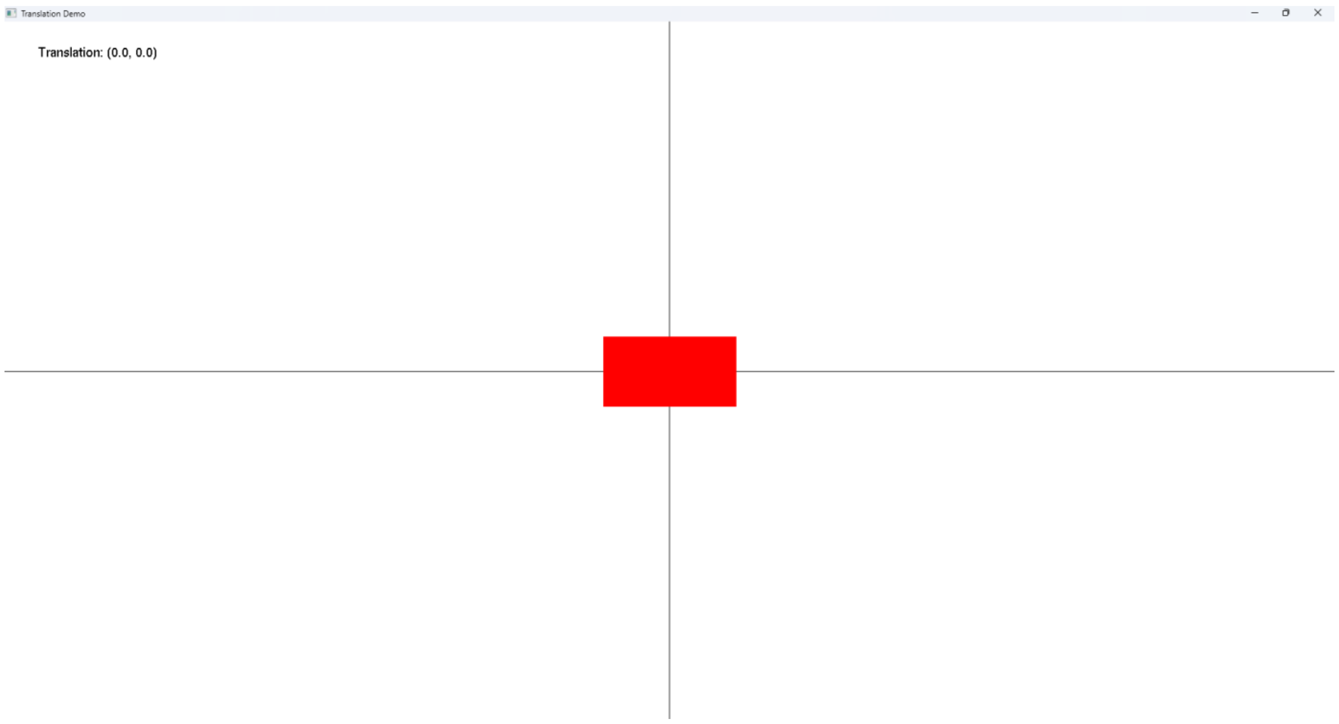
The provided code implements a translation demo using OpenGL. Below is the explanation of the code:

```
#include <GL/glut.h>
#include <math.h>
#include <stdio.h>

// Global variables for translation
float translateX = 0.0f;
float translateY = 0.0f;

// Initialize window and OpenGL settings
void init() {
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f); // White background
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-10.0, 10.0, -10.0, 10.0);
}

// Draw a colored square
void drawSquare() {
    glBegin(GL_POLYGON);
    glColor3f(1.0f, 0.0f, 0.0f); // Red
    glVertex2f(-1.0f, -1.0f);
    glVertex2f(1.0f, -1.0f);
    glVertex2f(1.0f, 1.0f);
    glVertex2f(-1.0f, 1.0f);
    glEnd();
```

```
}

// Draw coordinate axes
void drawAxes() {
    glColor3f(0.0f, 0.0f, 0.0f); // Black
    glBegin(GL_LINES);
    // X-axis
    glVertex2f(-10.0f, 0.0f);
    glVertex2f(10.0f, 0.0f);
    // Y-axis
    glVertex2f(0.0f, -10.0f);
    glVertex2f(0.0f, 10.0f);
    glEnd();
}

// Display function
void display() {
    glClear(GL_COLOR_BUFFER_BIT);
    // Draw axes
    drawAxes();
    // Apply translation and draw square
    glPushMatrix();
    glTranslatef(translateX, translateY, 0.0f);
    drawSquare();
    glPopMatrix();
    // Display current translation values
    glColor3f(0.0f, 0.0f, 0.0f);
    glRasterPos2f(-9.5f, 9.0f);
    char buffer[50];
    sprintf_s(buffer, sizeof(buffer), "Translation: (%.1f,
%.1f)", translateX, translateY);
    for (char* c = buffer; *c != '\0'; c++) {
        glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, *c);
    }
    glutSwapBuffers();
}

// Keyboard control
void keyboard(unsigned char key, int x, int y) {
    switch (key) {
    case 'w': translateY += 0.5f; break; // Move up
    case 's': translateY -= 0.5f; break; // Move down
    case 'a': translateX -= 0.5f; break; // Move left
    case 'd': translateX += 0.5f; break; // Move right
    case ' ': // Reset position
        translateX = 0.0f;
        translateY = 0.0f;
        break;
    }
    glutPostRedisplay();
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize(800, 800);
    glutCreateWindow("Translation Demo");
    init();
    glutDisplayFunc(display);
    glutKeyboardFunc(keyboard);
    glutMainLoop();
    return 0;
}
```

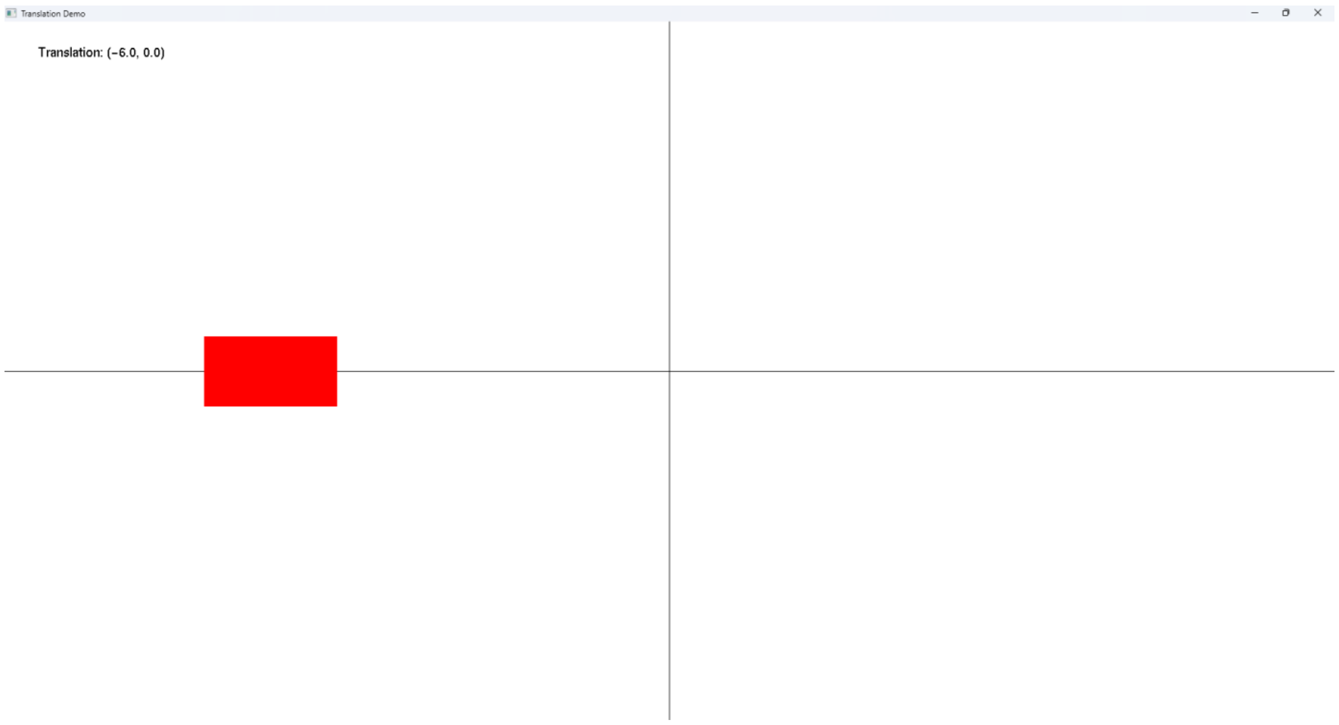
- **Translation Demo**:
  - The square moves smoothly in the specified direction when pressing W, A, S, or D.
  - The position resets to the origin when pressing the spacebar.

1. First screen



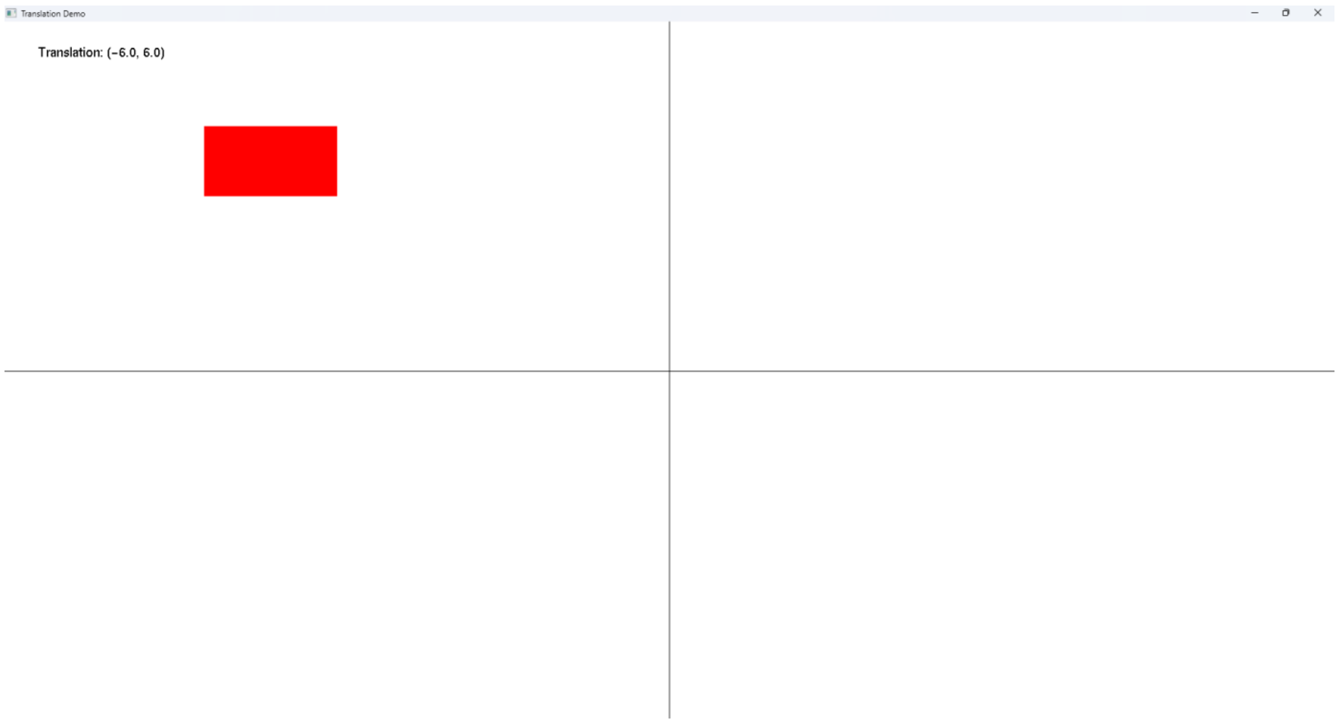
Translation: (0.0, 0.0)

2. Click on **A**



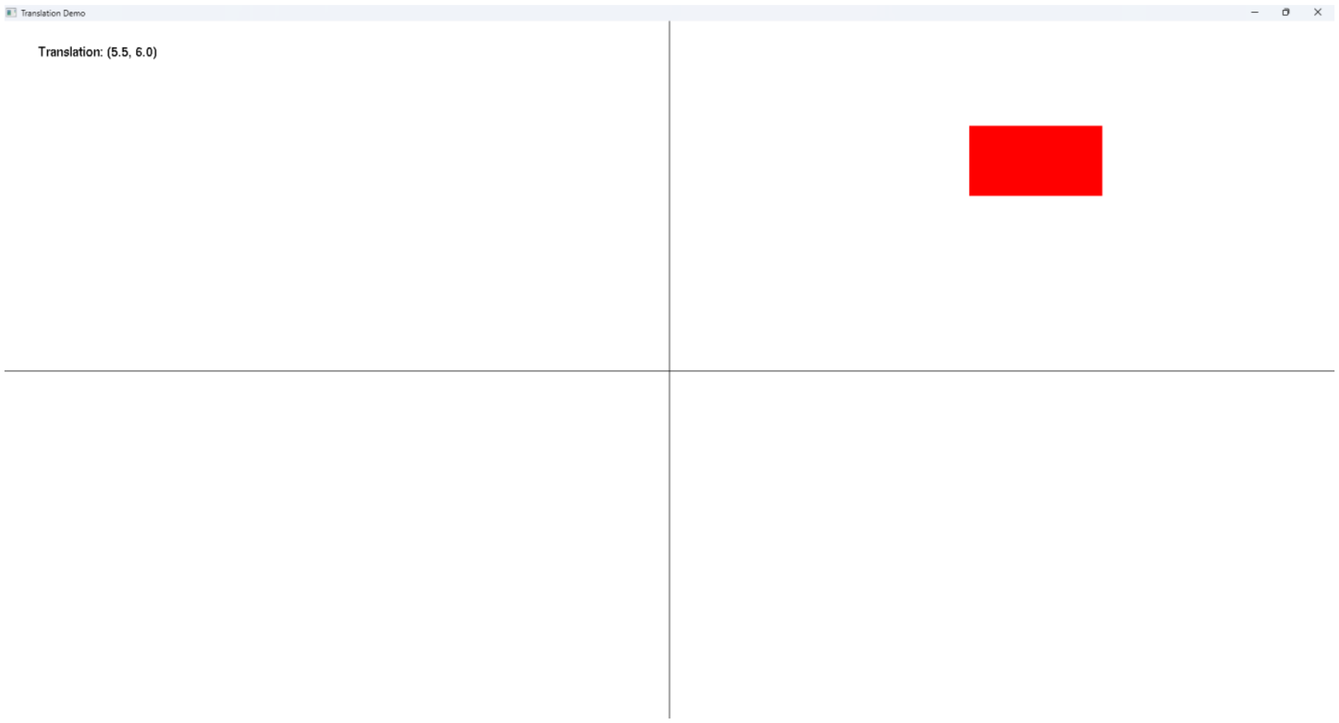
Translation: (−6.0, 0.0)

3. Click on **W**



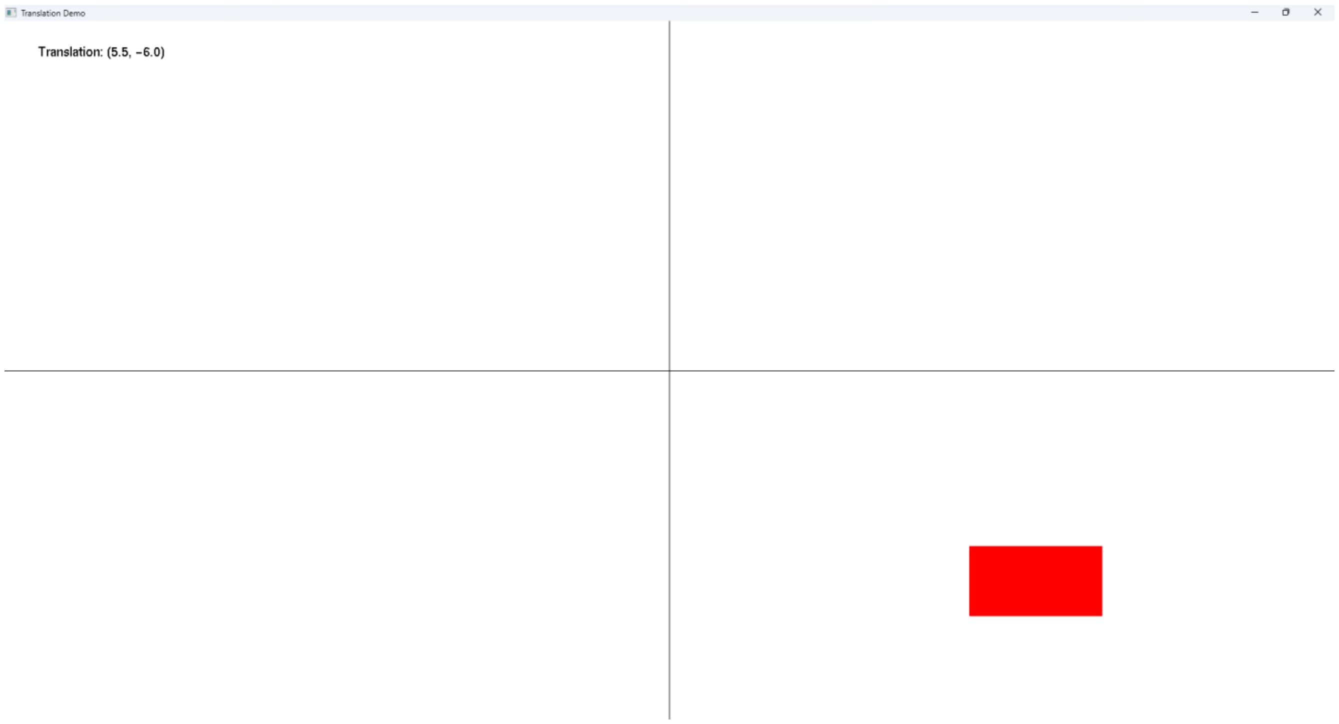
Translation: (−6.0, 6.0)

4. Click on **D**



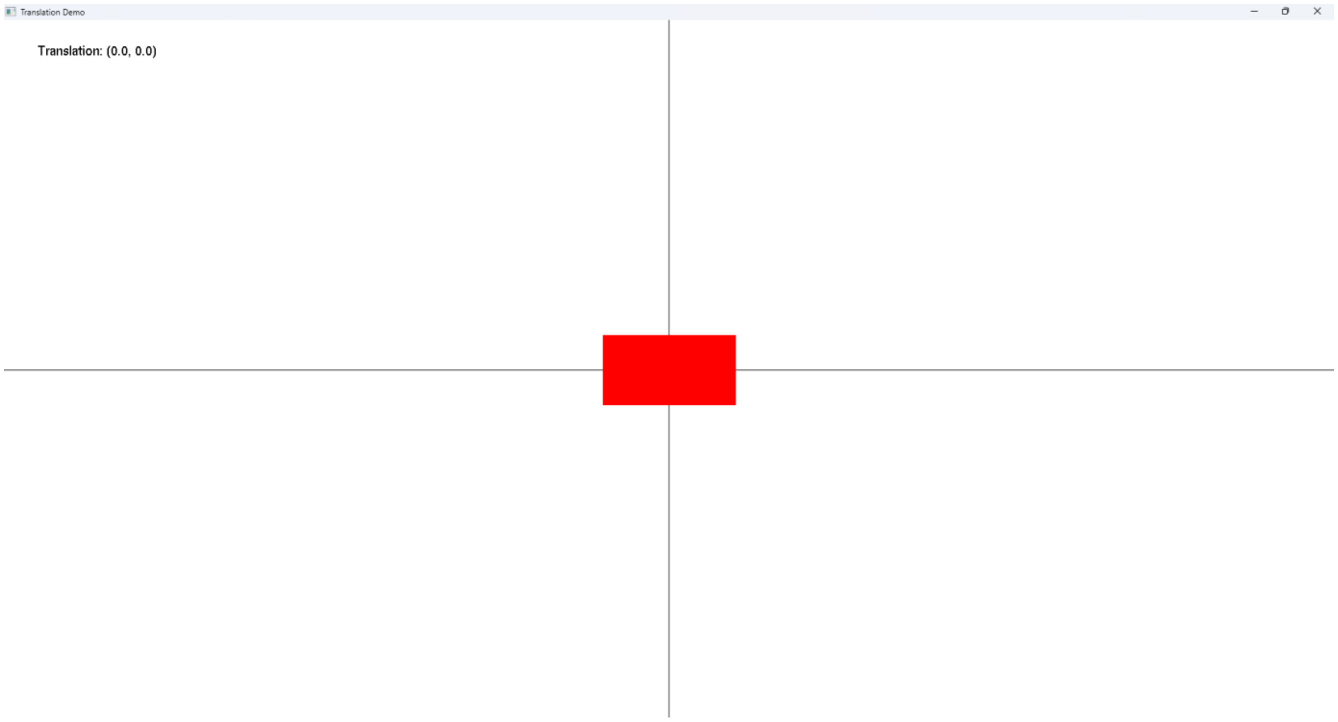
Translation: (5.5, 6.0)

5. Click on **S**



Translation: (5.5, −6.0)

6. Click on **Space**



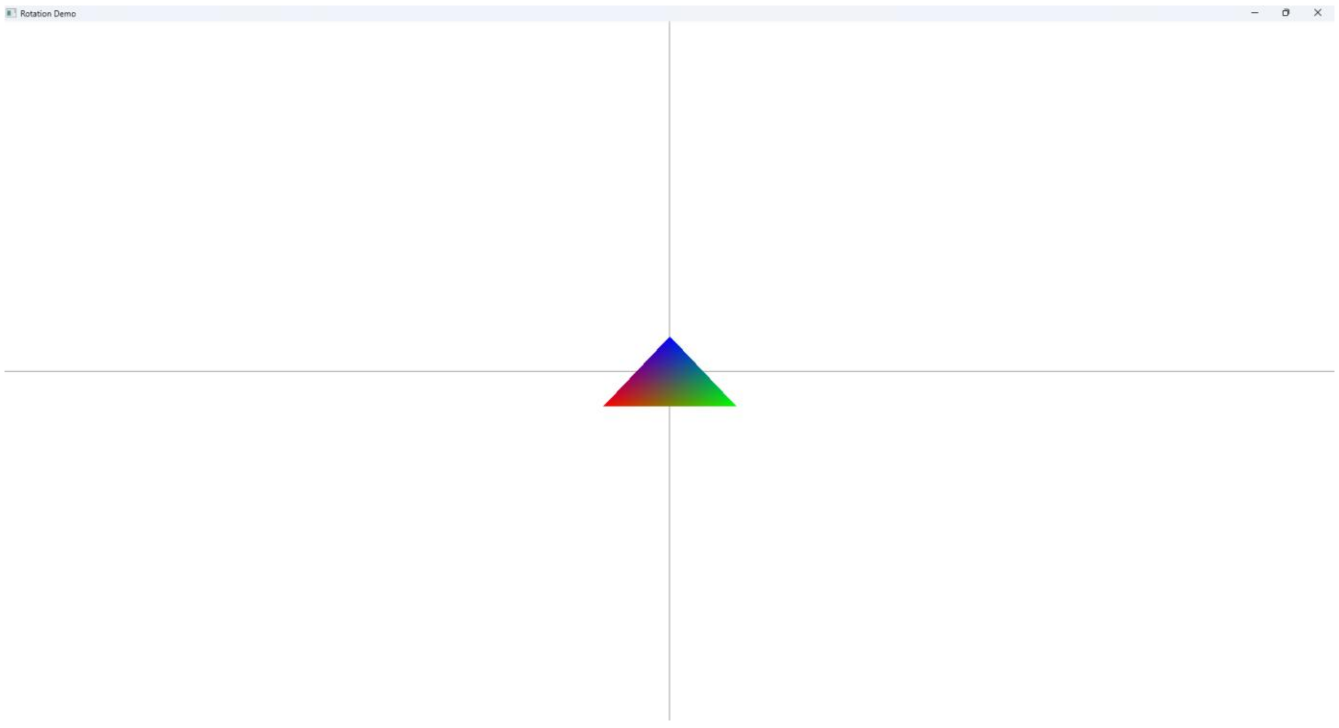
Translation: (0.0, 0.0)

3.2. Rotation Demo

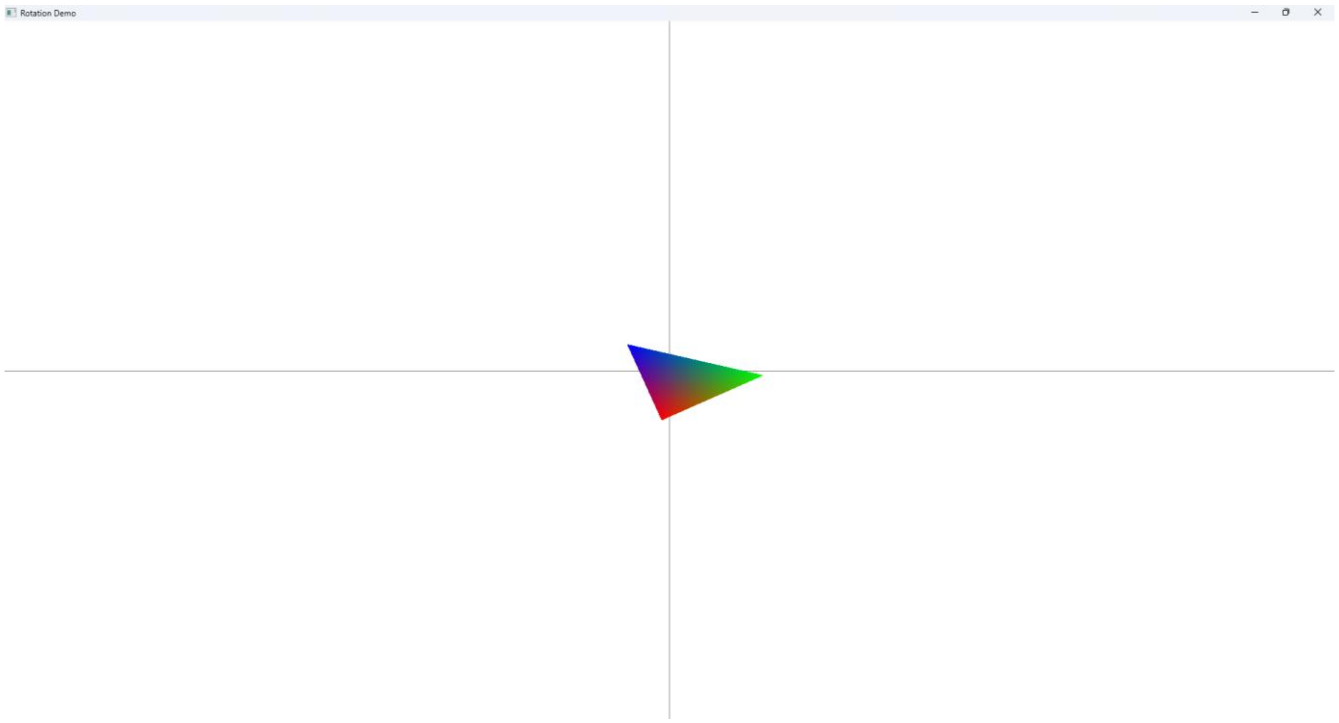
The provided code implements a rotation demo using OpenGL. Below is the explanation of the code:

```
#include <GL/glut.h>
#include <math.h>

// Global variables for rotation
float rotateAngle = 0.0f;
bool autoRotate = false;

// Initialize window and OpenGL settings
void init() {
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f); // White background
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-10.0, 10.0, -10.0, 10.0); // Set coordinate
system
}

// Draw a triangle with different colored vertices
void drawTriangle() {
    glBegin(GL_TRIANGLES);
    glColor3f(1.0f, 0.0f, 0.0f); // Red
    glVertex2f(-1.0f, -1.0f);
    glColor3f(0.0f, 1.0f, 0.0f); // Green
    glVertex2f(1.0f, -1.0f);
    glColor3f(0.0f, 0.0f, 1.0f); // Blue
    glVertex2f(0.0f, 1.0f);
    glEnd();
}

// Draw rotation center indicator
void drawCenter() {
    glPointSize(5.0f);
    glColor3f(0.0f, 0.0f, 0.0f); // Black
    glBegin(GL_POINTS);
    glVertex2f(0.0f, 0.0f);
    glEnd();
}

// Draw coordinate axes
void drawAxes() {
    glColor3f(0.5f, 0.5f, 0.5f); // Gray
    glBegin(GL_LINES);
    glVertex2f(-10.0f, 0.0f);
    glVertex2f(10.0f, 0.0f);
    glVertex2f(0.0f, -10.0f);
    glVertex2f(0.0f, 10.0f);
    glEnd();
}

// Display function
void display() {
    glClear(GL_COLOR_BUFFER_BIT);
    drawAxes(); // Draw axes
    drawCenter(); // Draw center point
    glPushMatrix();
    glRotatef(rotateAngle, 0.0f, 0.0f, 1.0f); // Apply rotation
    drawTriangle(); // Draw triangle
    glPopMatrix();
    glutSwapBuffers();
}

// Keyboard control
void keyboard(unsigned char key, int x, int y) {
    switch (key) {
    case 'r': rotateAngle += 5.0f; break; // Rotate clockwise
    case 'R': rotateAngle -= 5.0f; break; // Rotate counter-
clockwise
    case ' ': rotateAngle = 0.0f; break; // Reset rotation
    case 'a': autoRotate = !autoRotate; break; // Toggle auto-
rotation
    }
    // Keep angle between 0 and 360
    if (rotateAngle >= 360.0f) rotateAngle -= 360.0f;
    if (rotateAngle < 0.0f) rotateAngle += 360.0f;
    glutPostRedisplay(); // Redraw the scene
}

// Timer function for auto-rotation
void update(int value) {
    if (autoRotate) {
        rotateAngle += 2.0f;
        if (rotateAngle >= 360.0f) rotateAngle -= 360.0f;
        glutPostRedisplay();
    }
    glutTimerFunc(16, update, 0); // ~60 FPS
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize(800, 800);
    glutCreateWindow("Rotation Demo");
    init();
    glutDisplayFunc(display);
    glutKeyboardFunc(keyboard);
    glutTimerFunc(0, update, 0);
    glutMainLoop();
    return 0;
}
```

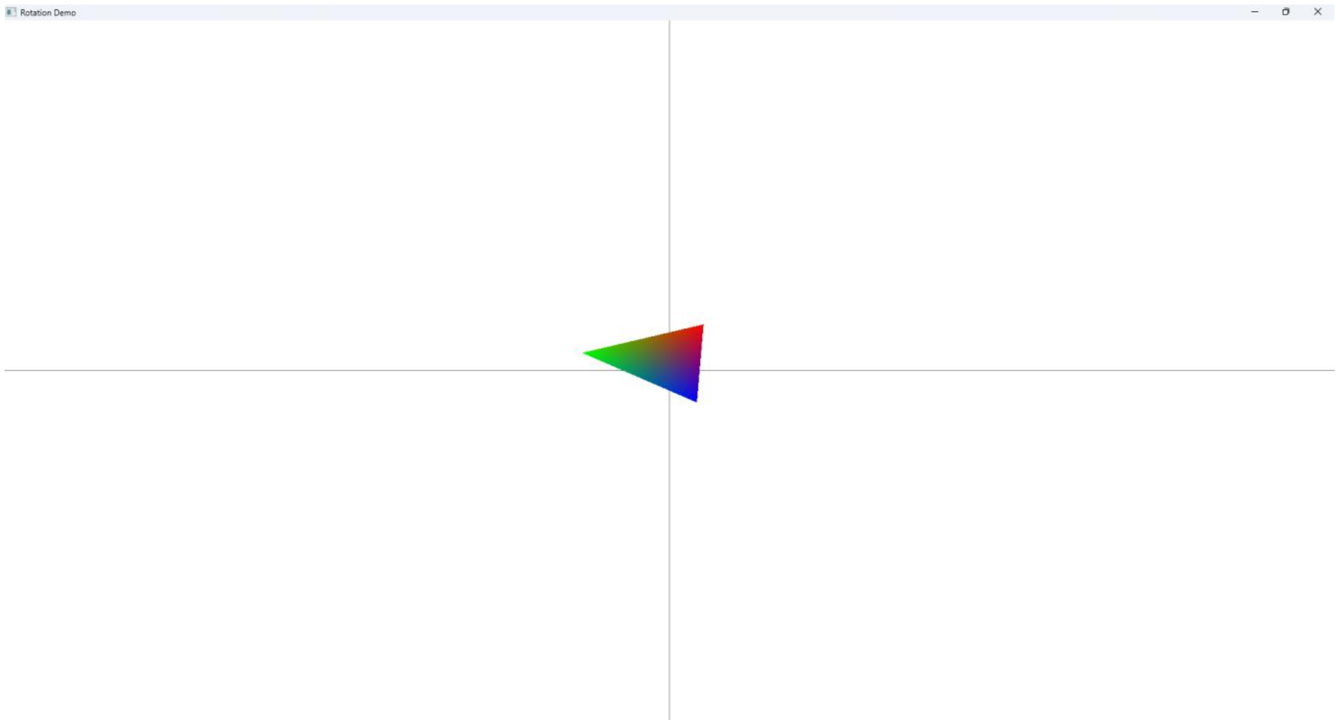
1. First Screen



2. Click on **R** or **Shift + R**



3. Click on **A** **(it will rotate as loop auto, and when click on A again it will stop)**



Rotation Demo:

- The triangle rotates clockwise or counter-clockwise when pressing R or Shift+R.
- Auto-rotation can be toggled with the A key.
- The rotation resets to 0° when pressing the spacebar.