Title

2D Implementation (Scaling) and Translation for making object smaller to bigger or bigger to smaller.

Introduction

In this project, a simple 2D scaling transformation has been implemented using OpenGL and GLUT. The main objective is to modify a previously used translation-based program and extend it to allow an object to become **bigger** or **smaller** based on keyboard input. A circle shape is rendered at the center of the window, and interactive scaling is applied using the '+', '-', and 'r' keys. This project demonstrates how scaling transformations affect objects in a 2D coordinate system and how user input can be used to dynamically change the size of shapes in real time.

Contents

The following functions and shapes have been used in this project:

1. display()
   - This function handles all drawing operations.
   - Applies the scaling transformation using glScalef(scaleFactor, scaleFactor, 1.0f).
   - A yellow-colored square is drawn at the center of the screen using GL_TRIANGLE_FAN.
2. keyboard()
   - Handles keyboard input from the user.
   - Pressing '+' increases the scale factor, making the object bigger.
   - Pressing '-' decreases the scale factor, making the object smaller (with a minimum limit).
   - Pressing 'r' resets the scale factor to its original value (1.0).
   - Finally calls glutPostRedisplay() to update the screen.
3. main()
   - Creates the window using GLUT functions.
   - Sets up the 2D orthographic projection using gluOrtho2D(0, 800, 0, 600).
   - Registers the display and keyboard callback functions.
   - Starts the GLUT main loop.
4. Shape Used: Circle
   - A simple yellow circle is created with 100 segments using trigonometric functions.
   - Initially centered at the origin after translation.
   - Its size changes dynamically depending on the scaling factor.

Code

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

float scaleFactor = 1.0f;

void display()
{
   glClear(GL_COLOR_BUFFER_BIT);
   glLoadIdentity();
   glTranslatef(400.0f, 300.0f, 0.0f);
   glScalef(scaleFactor, scaleFactor, 1.0f);

   glBegin(GL_TRIANGLE_FAN);
   glColor3f(1.0f, 1.0f, 0.0f);
   glVertex2f(0, 0);

   float radius = 50.0f;
   int segments = 100;
   for(int i = 0; i <= segments; i++)
   {
      float angle = 2.0f * 3.14159f * i / segments;
      glVertex2f(radius * cos(angle), radius * sin(angle));
   }
```

```
    glEnd();
    glutSwapBuffers();
}

void keyboard(unsigned char key, int x, int y)
{
    if (key == '+')
    {
        scaleFactor += 0.1f;
        std::cout << "Bigger: Scale = " << scaleFactor << std::endl;
    }
    else if (key == '-')
    {
        scaleFactor -= 0.1f;
        if (scaleFactor < 0.1f) scaleFactor = 0.1f;
        std::cout << "Smaller: Scale = " << scaleFactor << std::endl;
    }
    else if (key == 'r')
    {
        scaleFactor = 1.0f;
        std::cout << "Reset: Scale = 1.0" << std::endl;
    }
    glutPostRedisplay();
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize(800, 600);
    glutCreateWindow("2D Scaling - Circle");
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0, 800, 0, 600);
    glMatrixMode(GL_MODELVIEW);
    glutDisplayFunc(display);
    glutKeyboardFunc(keyboard);
    glutMainLoop();
    return 0;
}
```

Output

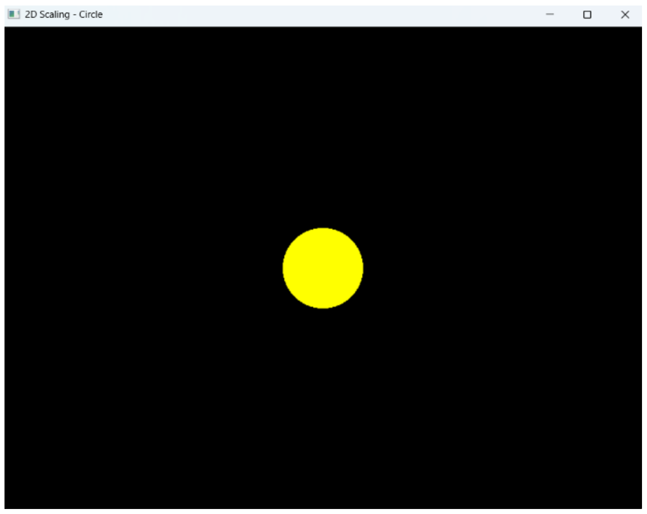


Figure 1: Initial Position and after pressing "r"

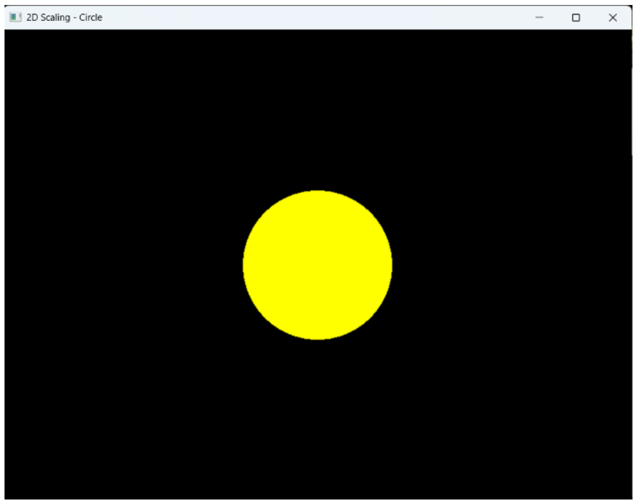
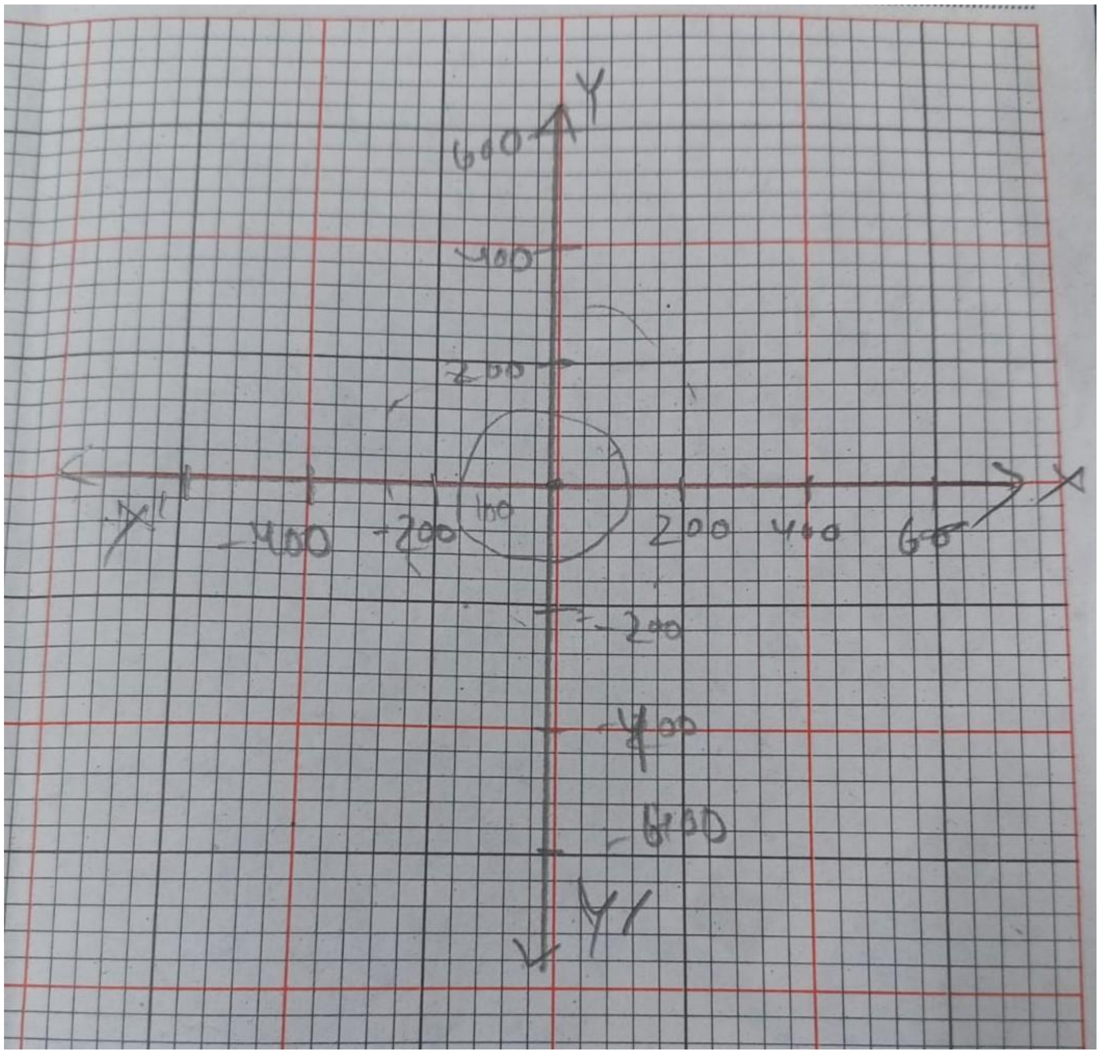


Figure 2: After pressing "+".

Graph



Discussion

The main objective of the task is to modify a translation-based program and implement interactive 2D scaling. Scaling transforms the size of an object while keeping its shape proportional. In this program, scaling is applied around the center of the window by first translating the coordinate system to the center and then applying glScalef().

Whenever the user presses the '+' key, the scale factor increases, making the square larger. Pressing the '–' key decreases the size but does not allow it to shrink below a safe minimum value. The 'r' key resets the object to its default size. This dynamic interaction demonstrates how transformations can be controlled in real time, which is useful in animations, simulations, and interactive graphics applications.