

## **Experiment No: 7**

### **Experiment Name:**

**Magnifying a Circle, a Triangle and a Rectangle about a Point**

---

### **Objective:**

The objective of this experiment is to study and implement scaling (magnification) of a circle, triangle, and rectangle about a fixed reference point using computer graphics techniques.

---

### **Theory:**

Magnification (or scaling) is a basic geometric transformation in computer graphics. It changes the size of an object either by enlarging or shrinking it.

Scaling can be:

- Uniform scaling – same factor in x and y directions
- Non-uniform scaling – different factors in x and y directions

To magnify an object about a specific point:

1. Translate the object so the point lies at the origin
  2. Apply scaling
  3. Translate the object back to its original position
- 

### **Software Requirements:**

- Code::Blocks
- GLUT Library
- C++ Language

## **Procedure / Description of Code:**

1. Initialize the OpenGL window using GLUT.
2. Set up a 2D orthographic view using gluOrtho2D().
3. Draw a **circle**, **triangle**, and **rectangle** using OpenGL primitives.
4. Choose a fixed point as the **scaling (magnification) point**.
5. Translate the object so that the scaling point moves to the origin.
6. Apply scaling using glScalef() to magnify the object.
7. Translate the object back to its original position.
8. Display the magnified shapes on the screen.

## **SOURCE CODE:**

```
#include <windows.h>

#include <GL/glut.h>

#include <cmath>

float scaleFactor = 1.0f;

bool increase = true;

// ----- Draw Circle -----

void drawCircle(float cx, float cy, float r)

{

    glBegin(GL_POINTS);

    for (int i = 0; i < 360; i++)

    {

        float theta = i * 3.1416 / 180;

        glVertex2f(cx + r * cos(theta), cy + r * sin(theta));

    }

    glEnd();

}
```

```
void display()
{
    glClear(GL_COLOR_BUFFER_BIT);

    // Pivot Point
    float px = 250, py = 250;

    glPushMatrix();
    glTranslatef(px, py, 0);
    glScalef(scaleFactor, scaleFactor, 1);
    glTranslatef(-px, -py, 0);

    // Circle
    glColor3f(1, 0, 0);
    drawCircle(200, 300, 40);

    // Triangle
    glColor3f(0, 0, 1);
    glBegin(GL_TRIANGLES);
    glVertex2f(300, 300);
    glVertex2f(350, 300);
    glVertex2f(325, 350);
```

```
glEnd();

// Rectangle

glColor3f(0, 1, 0);

glBegin(GL_QUADS);

glVertex2f(200, 150);

glVertex2f(300, 150);

glVertex2f(300, 200);

glVertex2f(200, 200);

glEnd();

glPopMatrix();

glFlush();

}

void timer(int)

{

    if (increase)

        scaleFactor += 0.02f;

    else

        scaleFactor -= 0.02f;

    if (scaleFactor >= 2.0f)

        increase = false;
```

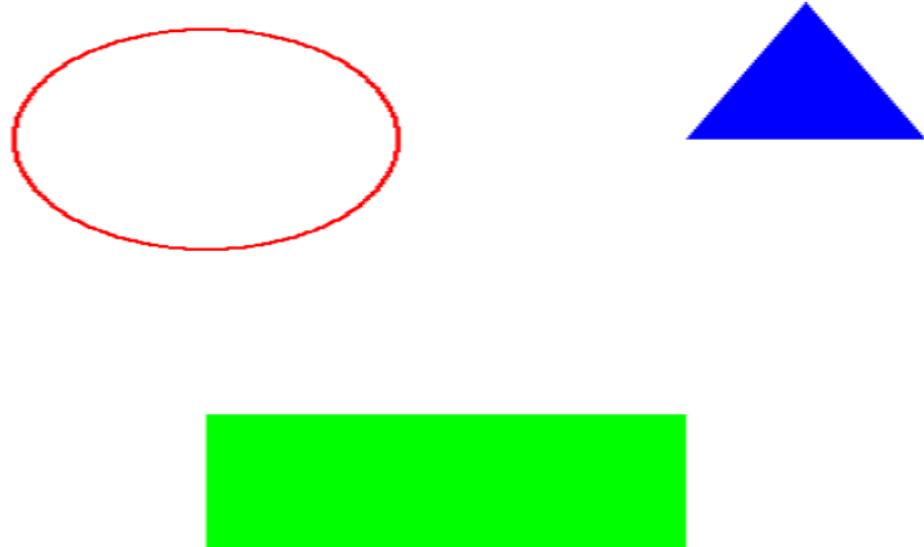
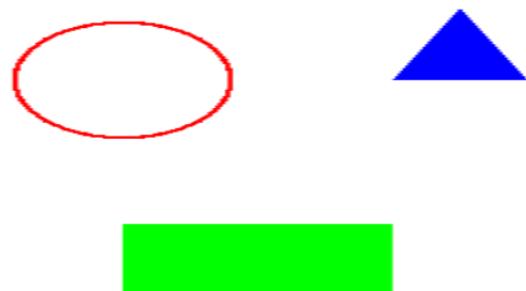
```
if (scaleFactor <= 1.0f)
    increase = true;

glutPostRedisplay();
glutTimerFunc(50, timer, 0);
}

void init()
{
    glClearColor(1, 1, 1, 1);
    glPointSize(2);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0, 500, 0, 500);
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Experiment 7: Magnification of Objects");
    init();
    glutDisplayFunc(display);
    glutTimerFunc(0, timer, 0);
}
```

```
glutMainLoop();  
return 0;  
}
```



## **Conclusion / Discussion:**

The circle, triangle, and rectangle were successfully magnified about the given point. This experiment demonstrates how **scaling transformation** is used to magnify objects in computer graphics. By combining translation and scaling, objects can be enlarged about any chosen point. This concept is important in zooming, animation, and graphical modeling applications.