**Batch: D - 1 Roll No.: 16010122096**

**Experiment No. 06**

|  |
| --- |
| **TITLE**: Write a program to perform 2D and 3D transformation |

**AIM:**

Write a program to perform 2D and 3D transformation

a. Translation

b. Scaling

c. Rotation

d. Shear

e. Reflection **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

**CO2:** Implement Fill area Primitives, 2D Geometric Transformations and 2D viewing

**CO3:** Implement Clipping,3D Geometric Transformations and 3D viewing

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

<https://cse18-iiith.vlabs.ac.in/exp/transformations-rotation/>

<https://cse18-iiith.vlabs.ac.in/exp/transformations-scaling/>

<https://cse18-iiith.vlabs.ac.in/exp/transformations-translation/>

<https://cse18-iiith.vlabs.ac.in/exp/2d-demo/>

<https://cse18-iiith.vlabs.ac.in/exp/3d-articulated-arm/>

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Implementation details:**

1. **2D TRASNFORMATIONS**
2. **Translation:**

#include <GL/glut.h>

#include <cmath>

void drawAxes() {

    glBegin(GL\_LINES);

    glColor3f(0.0, 0.0, 0.0);

    glVertex2f(-1.0, 0.0); glVertex2f(1.0, 0.0);

    glVertex2f(0.0, -1.0); glVertex2f(0.0, 1.0);

    glEnd();

}

void drawFigure() {

    glBegin(GL\_LINE\_LOOP);

    glVertex2f(-0.2, -0.2);

    glVertex2f(0.2, -0.2);

    glVertex2f(0.2, 0.2);

    glVertex2f(-0.2, 0.2);

    glEnd();

}

void translate(float tx, float ty) {

    glTranslatef(tx, ty, 0.0);

}

void display() {

    glClear(GL\_COLOR\_BUFFER\_BIT);

    glLoadIdentity();

    drawAxes();

    glColor3f(1.0, 0.0, 0.0);

    glPushMatrix();

    drawFigure();

    glPopMatrix();

    glLoadIdentity();

    translate(0.5, 0.5);

    glColor3f(0.0, 1.0, 0.0);

    drawFigure();

    glFlush();

}

void init() {

    glClearColor(1.0, 1.0, 1.0, 1.0);

    gluOrtho2D(-1.0, 1.0, -1.0, 1.0);

}

int main(int argc, char\*\* argv) {

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

    glutInitWindowSize(500, 500);

    glutInitWindowPosition(200, 200);

    glutCreateWindow("2D Translation");

    init();

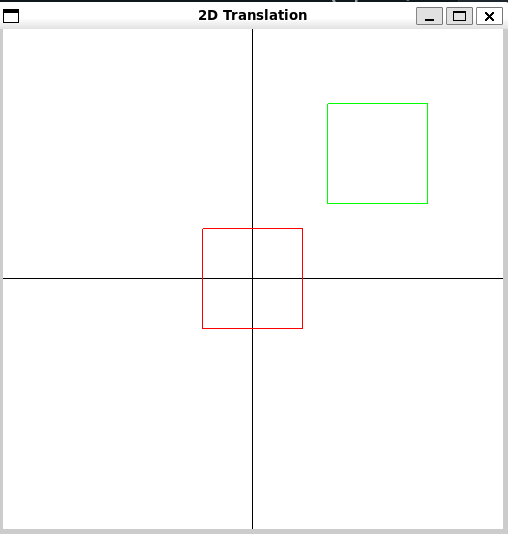
    glutDisplayFunc(display);

    glutMainLoop();

    return 0;

}

**Output:**

****

1. **Scaling:**

#include <GL/glut.h>

#include <cmath>

void drawAxes() {

    glBegin(GL\_LINES);

    glColor3f(0.0, 0.0, 0.0);

    glVertex2f(-7.0, 0.0); glVertex2f(7.0, 0.0);

    glVertex2f(0.0, -7.0); glVertex2f(0.0, 7.0);

    glEnd();

}

void drawFigure() {

    glBegin(GL\_LINE\_LOOP);

    glVertex2f(-0.2, -0.2);

    glVertex2f(0.2, -0.2);

    glVertex2f(0.2, 0.2);

    glVertex2f(-0.2, 0.2);

    glEnd();

}

void scaleRelativeToPoint(float sx, float sy, float px, float py) {

    int a = px, b = py;

    glTranslatef(px, py, 0.0);

    glScalef(sx, sy, 0.0);

    glTranslatef(-(a), -(b), 0.0);

}

void display() {

    glClear(GL\_COLOR\_BUFFER\_BIT);

    glLoadIdentity();

    drawAxes();

    glColor3f(1.0, 0.0, 0.0);

    glPushMatrix();

    drawFigure();

    glPopMatrix();

    glLoadIdentity();

    glColor3f(0.0, 1.0, 0.0);

    scaleRelativeToPoint(2.0, 2.0, 0.2, 0.2);

    drawFigure();

    glFlush();

}

void init() {

    glClearColor(1.0, 1.0, 1.0, 1.0);

    gluOrtho2D(-1.0, 1.0, -1.0, 1.0);

}

int main(int argc, char\*\* argv) {

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

    glutInitWindowSize(500, 500);

    glutInitWindowPosition(100, 100);

    glutCreateWindow("2D Scaling");

    init();

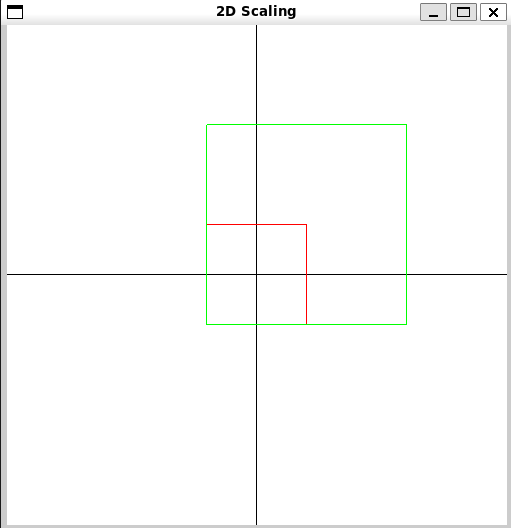
    glutDisplayFunc(display);

    glutMainLoop();

    return 0;

}

**Output:**

****

1. **Rotation:**

#include <GL/glut.h>

#include <cmath>

void drawAxes() {

    glBegin(GL\_LINES);

    glColor3f(0.0, 0.0, 0.0);

    glVertex2f(-1.0, 0.0); glVertex2f(1.0, 0.0);

    glVertex2f(0.0, -1.0); glVertex2f(0.0, 1.0);

    glEnd();

}

void drawFigure() {

    glBegin(GL\_LINE\_LOOP);

    glVertex2f(-0.2, -0.2);

    glVertex2f(0.2, -0.2);

    glVertex2f(0.2, 0.2);

    glVertex2f(-0.2, 0.2);

    glEnd();

}

void rotateRelativeToPoint(float angle, float px, float py) {

    glTranslatef(px, py, 0.0);

    glRotatef(angle, 0.0, 0.0, 1.0);

    glTranslatef(-px, -py, 0.0);

}

void display() {

    glClear(GL\_COLOR\_BUFFER\_BIT);

    glLoadIdentity();

    drawAxes();

    glColor3f(1.0, 0.0, 0.0);

    drawFigure();

    glLoadIdentity();

    glColor3f(0.0, 1.0, 0.0);

    rotateRelativeToPoint(-45, -0.2, -0.2);

    drawFigure();

    glFlush();

}

void init() {

    glClearColor(1.0, 1.0, 1.0, 1.0);

    gluOrtho2D(-1.0, 1.0, -1.0, 1.0);

}

int main(int argc, char\*\* argv) {

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

    glutInitWindowSize(500, 500);

    glutInitWindowPosition(100, 100);

    glutCreateWindow("2D Rotation");

    init();

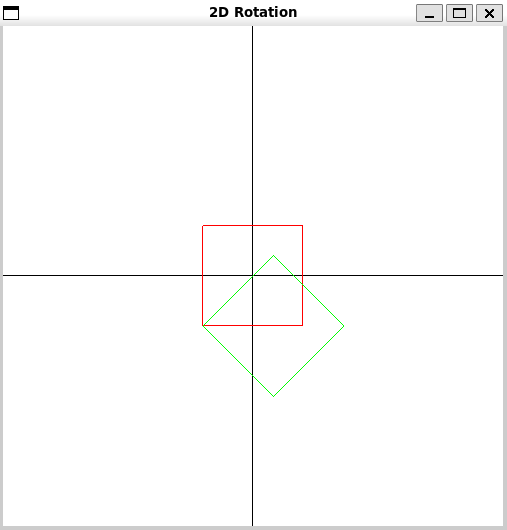
    glutDisplayFunc(display);

    glutMainLoop();

    return 0;

}

**Output:**

****

1. **Shear:**

#include <GL/glut.h>

#include <cmath>

void drawAxes() {

    glBegin(GL\_LINES);

    glColor3f(0.0, 0.0, 0.0);

    glVertex2f(-1.0, 0.0); glVertex2f(1.0, 0.0);

    glVertex2f(0.0, -1.0); glVertex2f(0.0, 1.0);

    glEnd();

}

void drawFigure() {

    glBegin(GL\_LINE\_LOOP);

    glVertex2f(-0.2, -0.2);

    glVertex2f(0.2, -0.2);

    glVertex2f(0.2, 0.2);

    glVertex2f(-0.2, 0.2);

    glEnd();

}

void shear(float shx, float shy, float tx, float ty) {

    glMatrixMode(GL\_MODELVIEW);

    glLoadIdentity();

    glTranslatef(-tx, -ty, 0.0);

    GLfloat matrix[16] = {

        1.0, shy, 0.0, 0.0,

        shx, 1.0, 0.0, 0.0,

        0.0, 0.0, 1.0, 0.0,

        0.0, 0.0, 0.0, 1.0

    };

   glLoadMatrixf(matrix);

   glTranslatef(0.1, 0.0, 0.0);

}

void display() {

    glClear(GL\_COLOR\_BUFFER\_BIT);

    glLoadIdentity();

    drawAxes();

    glColor3f(1.0, 0.0, 0.0);

    drawFigure();

    glLoadIdentity();

    shear(0.5, 0.0, -0.2, -0.2);

    glColor3f(0.0, 1.0, 0.0);

    drawFigure();

    glFlush();

}

void init() {

    glClearColor(1.0, 1.0, 1.0, 1.0);

    gluOrtho2D(-1.0, 1.0, -1.0, 1.0);

}

int main(int argc, char\*\* argv) {

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

    glutInitWindowSize(500, 500);

    glutInitWindowPosition(100, 100);

    glutCreateWindow("2D Shear");

    init();

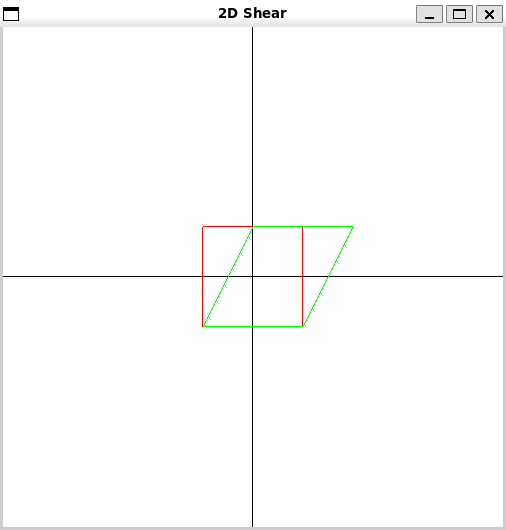
    glutDisplayFunc(display);

    glutMainLoop();

    return 0;

}

**Output:**

****

1. **3D TRANSFORMATIONS**
2. **Translation:**

#include <GL/glut.h>

void init() {

    glClearColor(0.0, 0.0, 0.0, 1.0);

    glEnable(GL\_DEPTH\_TEST);

}

void display() {

    glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

    glMatrixMode(GL\_MODELVIEW);

    glLoadIdentity();

    glPushMatrix();

    glTranslatef(0.0f, 0.0f, 0.0f);

    glColor3f(1.0f, 0.0f, 0.0f);

    glutSolidCube(0.5);

    glPopMatrix();

    glPushMatrix();

    glTranslatef(1.0f, 1.0f, 0.0f);

    glColor3f(0.0f, 1.0f, 0.0f);

    glutSolidCube(0.5);

    glPopMatrix();

    glutSwapBuffers();

}

void reshape(int w, int h) {

    glViewport(0, 0, w, h);

    glMatrixMode(GL\_PROJECTION);

    glLoadIdentity();

    gluPerspective(45.0, (double)w / (double)h, 1.0, 100.0);

    glTranslatef(0.0, 0.0, -5.0);

}

int main(int argc, char\*\* argv) {

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

    glutInitWindowSize(500, 500);

    glutCreateWindow("3D Translation");

    init();

    glutDisplayFunc(display);

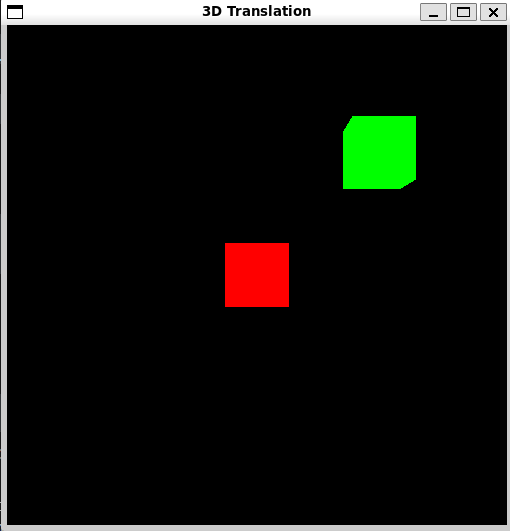
    glutReshapeFunc(reshape);

    glutMainLoop();

    return 0;

}

**Output:**

****

1. **Scaling:**

#include <GL/glut.h>

void init() {

    glClearColor(0.0, 0.0, 0.0, 1.0);

    glEnable(GL\_DEPTH\_TEST);

}

void display() {

    glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

    glMatrixMode(GL\_MODELVIEW);

    glLoadIdentity();

    glPushMatrix();

    glTranslatef(-1.0f, 0.0f, 0.0f);

    glColor3f(1.0f, 0.0f, 0.0f);

    glutSolidCube(0.5);

    glPopMatrix();

    glPushMatrix();

    glTranslatef(1.0f, 0.0f, 0.0f);

    glScalef(1.5f, 1.5f, 1.5f);

    glColor3f(0.0f, 1.0f, 0.0f);

    glutSolidCube(0.5);

    glPopMatrix();

    glutSwapBuffers();

}

void reshape(int w, int h) {

    glViewport(0, 0, w, h);

    glMatrixMode(GL\_PROJECTION);

    glLoadIdentity();

    gluPerspective(45.0, (double)w / (double)h, 1.0, 100.0);

    glTranslatef(0.0, 0.0, -5.0);

}

int main(int argc, char\*\* argv) {

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

    glutInitWindowSize(500, 500);

    glutCreateWindow("3D Scaling");

    init();

    glutDisplayFunc(display);

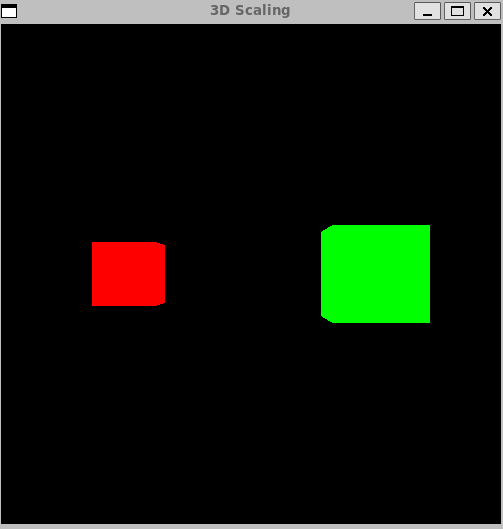
    glutReshapeFunc(reshape);

    glutMainLoop();

    return 0;

}

**Output:**



1. **Rotation:**

#include <GL/glut.h>

void init() {

    glClearColor(0.0, 0.0, 0.0, 1.0);

    glEnable(GL\_DEPTH\_TEST);

}

void display() {

    glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

    glMatrixMode(GL\_MODELVIEW);

    glLoadIdentity();

    glPushMatrix();

    glTranslatef(-1.0f, 0.0f, 0.0f);

    glColor3f(1.0f, 0.0f, 0.0f);

    glutSolidCube(0.5);

    glPopMatrix();

    glPushMatrix();

    glTranslatef(1.0f, 0.0f, 0.0f);

    glRotatef(45.0f, 1.0f, 0.0f, 0.0f);

    glRotatef(30.0f, 0.0f, 1.0f, 0.0f);

    glColor3f(0.0f, 1.0f, 0.0f);

    glutSolidCube(0.5);

    glPopMatrix();

    glutSwapBuffers();

}

void reshape(int w, int h) {

    glViewport(0, 0, w, h);

    glMatrixMode(GL\_PROJECTION);

    glLoadIdentity();

    gluPerspective(45.0, (double)w / (double)h, 1.0, 100.0);

    glTranslatef(0.0, 0.0, -5.0);

}

int main(int argc, char\*\* argv) {

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

    glutInitWindowSize(500, 500);

    glutCreateWindow("3D Rotation");

    init();

    glutDisplayFunc(display);

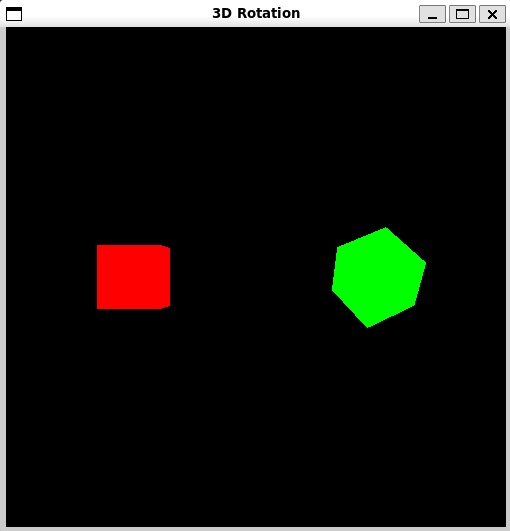
    glutReshapeFunc(reshape);

    glutMainLoop();

    return 0;

}

**Output:**

****

1. **Shear:**

#include <GL/glut.h>

void init() {

    glClearColor(0.0, 0.0, 0.0, 1.0);

    glEnable(GL\_DEPTH\_TEST);

}

void setShearMatrix() {

    GLfloat shearMatrix[16] = {

        1.0f, 0.5f, 0.0f, 0.0f,  // x shear

        0.0f, 1.0f, 0.5f, 0.0f,  // y shear

        0.0f, 0.0f, 1.0f, 0.0f,  // z shear

        0.0f, 0.0f, 0.0f, 1.0f   // homogeneous coordinate

    };

    glMultMatrixf(shearMatrix);

}

void display() {

    glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

    glMatrixMode(GL\_MODELVIEW);

    glLoadIdentity();

    glPushMatrix();

    glTranslatef(-1.0f, 0.0f, 0.0f);

    glColor3f(1.0f, 0.0f, 0.0f);

    glutSolidCube(0.5);

    glPopMatrix();

    glPushMatrix();

    glTranslatef(1.0f, 0.0f, 0.0f);

    setShearMatrix();

    glColor3f(0.0f, 1.0f, 0.0f);

    glutSolidCube(0.5);

    glPopMatrix();

    glutSwapBuffers();

}

void reshape(int w, int h) {

    glViewport(0, 0, w, h);

    glMatrixMode(GL\_PROJECTION);

    glLoadIdentity();

    gluPerspective(45.0, (double)w / (double)h, 1.0, 100.0);

    glTranslatef(0.0, 0.0, -5.0);

}

int main(int argc, char\*\* argv) {

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

    glutInitWindowSize(500, 500);

    glutCreateWindow("3D Shear");

    init();

    glutDisplayFunc(display);

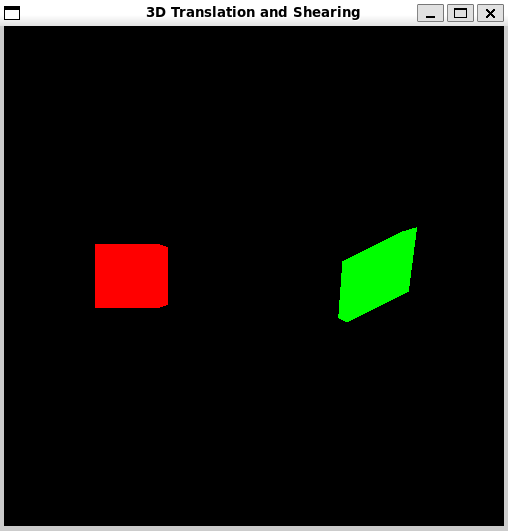
    glutReshapeFunc(reshape);

    glutMainLoop();

    return 0;

}

**Output:**



**Conclusion and discussion:**

In this experiment, various 2D and 3D transformations, including translation, scaling, rotation, shear, and reflection, were demonstrated using OpenGL. The implementations showcased how different transformations affect geometric objects in both 2D and 3D spaces. This exercise highlighted the versatility of transformation matrices in graphical programming and their essential role in creating dynamic and interactive visual applications.

**Date: 29 / 08 / 2024 Signature of faculty in-charge**

**Post lab**

Implement the code to **draw the Sierpinski Gasket**

**CODE:**

#include <GL/glut.h>

void drawSierpinski(float x1, float y1, float x2, float y2, float x3, float y3, int depth) {

    if (depth == 0) {

        glBegin(GL\_TRIANGLES);

        glVertex2f(x1, y1);

        glVertex2f(x2, y2);

        glVertex2f(x3, y3);

        glEnd();

    } else {

        float mid1x = (x1 + x2) / 2.0;

        float mid1y = (y1 + y2) / 2.0;

        float mid2x = (x2 + x3) / 2.0;

        float mid2y = (y2 + y3) / 2.0;

        float mid3x = (x3 + x1) / 2.0;

        float mid3y = (y3 + y1) / 2.0;

        drawSierpinski(x1, y1, mid1x, mid1y, mid3x, mid3y, depth - 1);

        drawSierpinski(mid1x, mid1y, x2, y2, mid2x, mid2y, depth - 1);

        drawSierpinski(mid3x, mid3y, mid2x, mid2y, x3, y3, depth - 1);

    }

}

void display() {

    glClear(GL\_COLOR\_BUFFER\_BIT);

    glLoadIdentity();

    glColor3f(1.0, 1.0, 1.0);

    drawSierpinski(-0.8f, -0.8f, 0.8f, -0.8f, 0.0f, 0.8f, 4);

    glutSwapBuffers();

}

void reshape(int w, int h) {

    glViewport(0, 0, w, h);

    glMatrixMode(GL\_PROJECTION);

    glLoadIdentity();

    gluOrtho2D(-1.0, 1.0, -1.0, 1.0);

    glMatrixMode(GL\_MODELVIEW);

}

int main(int argc, char\*\* argv) {

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

    glutInitWindowSize(500, 500);

    glutCreateWindow("Sierpinski Gasket");

    glutDisplayFunc(display);

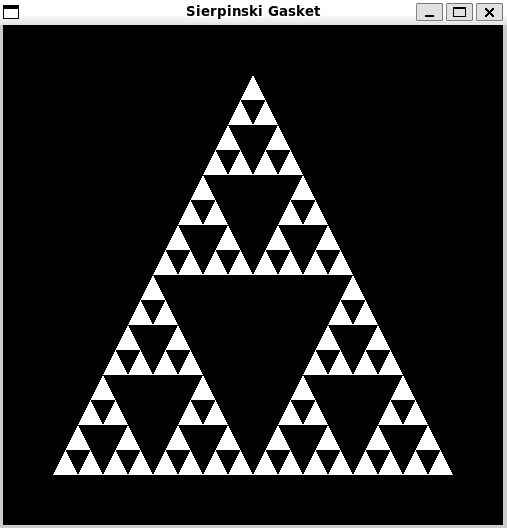
    glutReshapeFunc(reshape);

    glutMainLoop();

    return 0;

}

**Output:**



Draw dinosaur using Dino.dat Assignment (Search over internet for .dat file)