

| **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/>

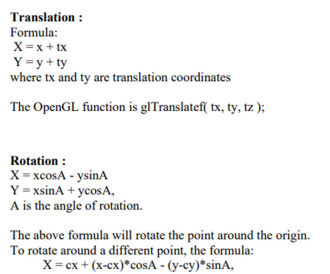
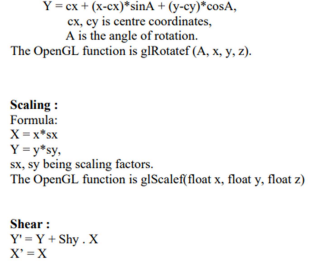
<https://cse18-iiith.vlabs.ac.in/exp/transformations-translation/>

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

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

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

**Algorithm/ Pseudo code for each process:**

**Implementation details:**

a. Translation

#include <GL/glut.h>

#include <iostream>

void drawLine(float x1, float y1, float x2, float y2) {

glBegin(GL\_LINES);

glVertex2f(x1, y1);

glVertex2f(x2, y2);

glEnd();

}

void translateAndDraw(float l1[4][2], float translate[2]) {

glColor3f(1.0f, 0.0f, 0.0f);

for (int i = 0; i < 4; ++i) {

float x1 = l1[i][0];

float y1 = l1[i][1];

float x2 = l1[(i + 1) % 4][0];

float y2 = l1[(i + 1) % 4][1];

drawLine(x1, y1, x2, y2);

}

for (int i = 0; i < 4; ++i) {

l1[i][0] += translate[0];

l1[i][1] += translate[1];

}

glColor3f(0.0f, 1.0f, 0.0f);

for (int i = 0; i < 4; ++i) {

float x1 = l1[i][0];

float y1 = l1[i][1];

float x2 = l1[(i + 1) % 4][0];

float y2 = l1[(i + 1) % 4][1];

drawLine(x1, y1, x2, y2);

}

}

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

float square[4][2] = { {10.0f, 10.0f}, {10.0f, 80.0f}, {80.0f, 80.0f}, {80.0f, 10.0f} };

float translate[2] = { 20.0f, 30.0f };

translateAndDraw(square, translate);

glFlush();

}

void init() {

glClearColor(0.0, 0.0, 0.0, 0.0);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0.0, 200.0, 0.0, 150.0);

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowPosition(50, 100);

glutInitWindowSize(400, 300);

glutCreateWindow("Translation");

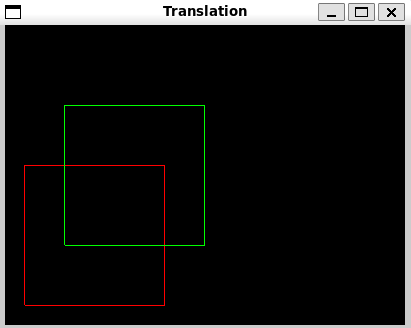
init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}



b. Scaling

#include <GL/glut.h>

#include <iostream>

void drawLine(float x1, float y1, float x2, float y2) {

glBegin(GL\_LINES);

glVertex2f(x1, y1);

glVertex2f(x2, y2);

glEnd();

}

void translateAndDraw(float l1[4][2], float l2[2]) {

glColor3f(1.0f, 0.0f, 0.0f);

for (int i = 0; i < 4; ++i) {

drawLine(l1[i][0], l1[i][1], l1[(i + 1) % 4][0], l1[(i + 1) % 4][1]);

}

for (int i = 0; i < 4; ++i) {

l1[i][0] \*= l2[0];

l1[i][1] \*= l2[1];

}

glColor3f(0.0f, 1.0f, 0.0f);

for (int i = 0; i < 4; ++i) {

drawLine(l1[i][0], l1[i][1], l1[(i + 1) % 4][0], l1[(i + 1) % 4][1]);

}

}

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

float square[4][2] = { {10.0f, 10.0f}, {10.0f, 80.0f}, {80.0f, 80.0f}, {80.0f, 10.0f} };

float scale[2] = { 1.5f, 1.5f };

translateAndDraw(square, scale);

glFlush();

}

void init() {

glClearColor(0.0, 0.0, 0.0, 0.0);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0.0, 200.0, 0.0, 150.0);

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowPosition(50, 100);

glutInitWindowSize(400, 300);

glutCreateWindow("Scaling");

init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}



c. Rotation

#include <GL/glut.h>#include <cmath>

const float PI = 3.14159265f;

void drawLine(float x1, float y1, float x2, float y2) {

glBegin(GL\_LINES);

glVertex2f(x1, y1);

glVertex2f(x2, y2);

glEnd();

}

void rotateAndDraw(float l1[4][2], float angle) {

float radians = angle \* (PI / 180.0f);

float cosTheta = cos(radians);

float sinTheta = sin(radians);

glColor3f(1.0f, 0.0f, 0.0f);

for (int i = 0; i < 4; ++i) {

float x1 = l1[i][0];

float y1 = l1[i][1];

float x2 = l1[(i + 1) % 4][0];

float y2 = l1[(i + 1) % 4][1];

drawLine(x1, y1, x2, y2);

}

for (int i = 0; i < 4; ++i) {

float x = l1[i][0];

float y = l1[i][1];

l1[i][0] = x \* cosTheta - y \* sinTheta;

l1[i][1] = x \* sinTheta + y \* cosTheta;

}

glColor3f(0.0f, 1.0f, 0.0f);

for (int i = 0; i < 4; ++i) {

float x1 = l1[i][0];

float y1 = l1[i][1];

float x2 = l1[(i + 1) % 4][0];

float y2 = l1[(i + 1) % 4][1];

drawLine(x1, y1, x2, y2);

}

}

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

float square[4][2] = { {10.0f, 10.0f}, {10.0f, 80.0f}, {80.0f, 80.0f}, {80.0f, 10.0f} };

float angle = 45.0f;

rotateAndDraw(square, angle);

glFlush();

}

void init() {

glClearColor(0.0, 0.0, 0.0, 0.0);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(-100.0, 180.0, -100.0, 180.0);

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowPosition(50, 100);

glutInitWindowSize(800, 600);

glutCreateWindow("Rotation");

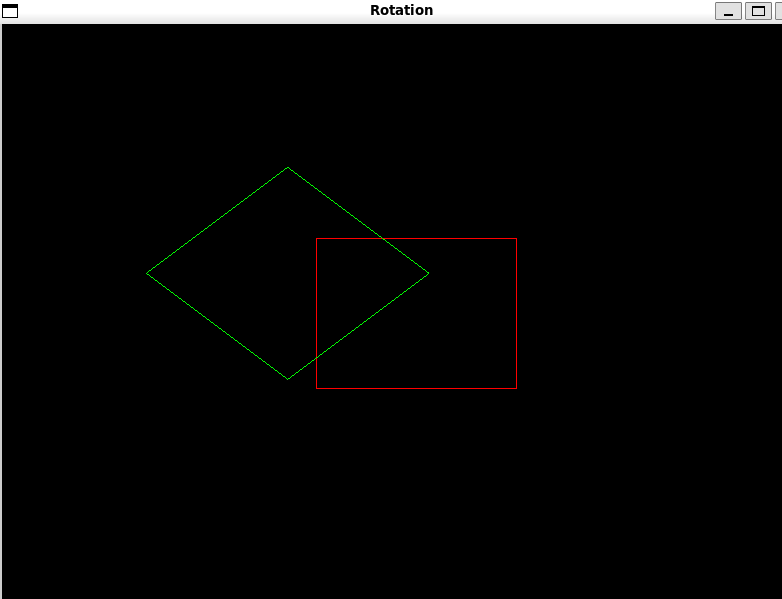
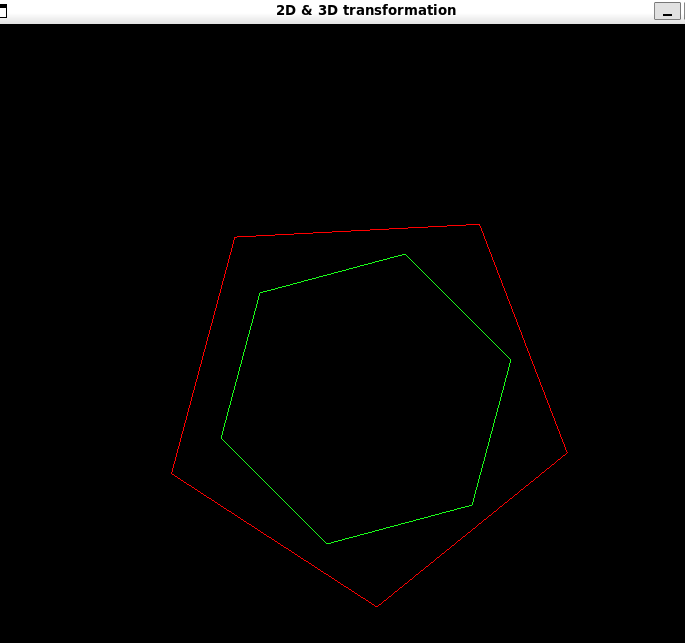
init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

  
****

d. Shear

#include <GL/glut.h>

void drawLine(float x1, float y1, float x2, float y2) {

glBegin(GL\_LINES);

glVertex2f(x1, y1);

glVertex2f(x2, y2);

glEnd();

}

void shearAndDraw(float l1[4][2], float shearX, float shearY) {

glColor3f(1.0f, 0.0f, 0.0f);

for (int i = 0; i < 4; ++i) {

float x1 = l1[i][0];

float y1 = l1[i][1];

float x2 = l1[(i + 1) % 4][0];

float y2 = l1[(i + 1) % 4][1];

drawLine(x1, y1, x2, y2);

}

for (int i = 0; i < 4; ++i) {

float x = l1[i][0];

float y = l1[i][1];

l1[i][0] = x + shearX \* y;

l1[i][1] = y + shearY \* x;

}

glColor3f(0.0f, 1.0f, 0.0f);

for (int i = 0; i < 4; ++i) {

float x1 = l1[i][0];

float y1 = l1[i][1];

float x2 = l1[(i + 1) % 4][0];

float y2 = l1[(i + 1) % 4][1];

drawLine(x1, y1, x2, y2);

}

}

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

float square[4][2] = { {10.0f, 10.0f}, {10.0f, 80.0f}, {80.0f, 80.0f}, {80.0f, 10.0f} };

float shearX = 0.5f;

float shearY = 0.3f;

shearAndDraw(square, shearX, shearY);

glFlush();

}

void init() {

glClearColor(0.0, 0.0, 0.0, 0.0);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(-100.0, 180.0, -100.0, 180.0);

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowPosition(50, 100);

glutInitWindowSize(800, 600);

glutCreateWindow("Shear");

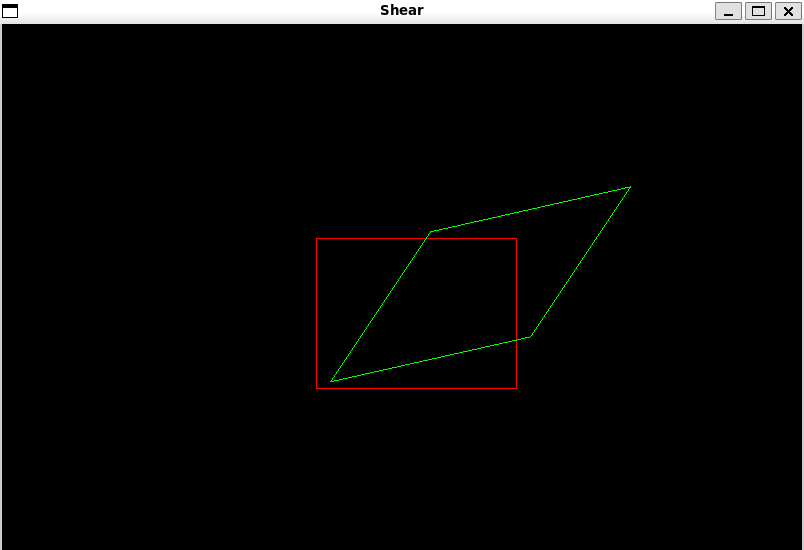
init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}



e. Reflection

**#include <GL/glut.h>**

**void drawLine(float x1, float y1, float x2, float y2) {**

**glBegin(GL\_LINES);**

**glVertex2f(x1, y1);**

**glVertex2f(x2, y2);**

**glEnd();**

**}**

**void reflectAndDraw(float l1[4][2], int reflectionType) {**

**glColor3f(1.0f, 0.0f, 0.0f);**

**for (int i = 0; i < 4; ++i) {**

**float x1 = l1[i][0];**

**float y1 = l1[i][1];**

**float x2 = l1[(i + 1) % 4][0];**

**float y2 = l1[(i + 1) % 4][1];**

**drawLine(x1, y1, x2, y2);**

**}**

**for (int i = 0; i < 4; ++i) {**

**float x = l1[i][0];**

**float y = l1[i][1];**

**switch (reflectionType) {**

**case 1:**

**l1[i][0] = x;**

**l1[i][1] = -y;**

**break;**

**case 2:**

**l1[i][0] = -x;**

**l1[i][1] = y;**

**break;**

**case 3:**

**l1[i][0] = y;**

**l1[i][1] = x;**

**break;**

**}**

**}**

**glColor3f(0.0f, 1.0f, 0.0f);**

**for (int i = 0; i < 4; ++i) {**

**float x1 = l1[i][0];**

**float y1 = l1[i][1];**

**float x2 = l1[(i + 1) % 4][0];**

**float y2 = l1[(i + 1) % 4][1];**

**drawLine(x1, y1, x2, y2);**

**}**

**}**

**void display() {**

**glClear(GL\_COLOR\_BUFFER\_BIT);**

**float square[4][2] = { {10.0f, 10.0f}, {10.0f, 80.0f}, {80.0f, 80.0f}, {80.0f, 10.0f} };**

**int reflectionType = 1;**

**reflectAndDraw(square, reflectionType);**

**glFlush();**

**}**

**void init() {**

**glClearColor(0.0, 0.0, 0.0, 0.0);**

**glMatrixMode(GL\_PROJECTION);**

**gluOrtho2D(-100.0, 180.0, -100.0, 180.0);**

**}**

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

**glutInit(&argc, argv);**

**glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);**

**glutInitWindowPosition(50, 100);**

**glutInitWindowSize(800, 600);**

**glutCreateWindow("Reflection");**

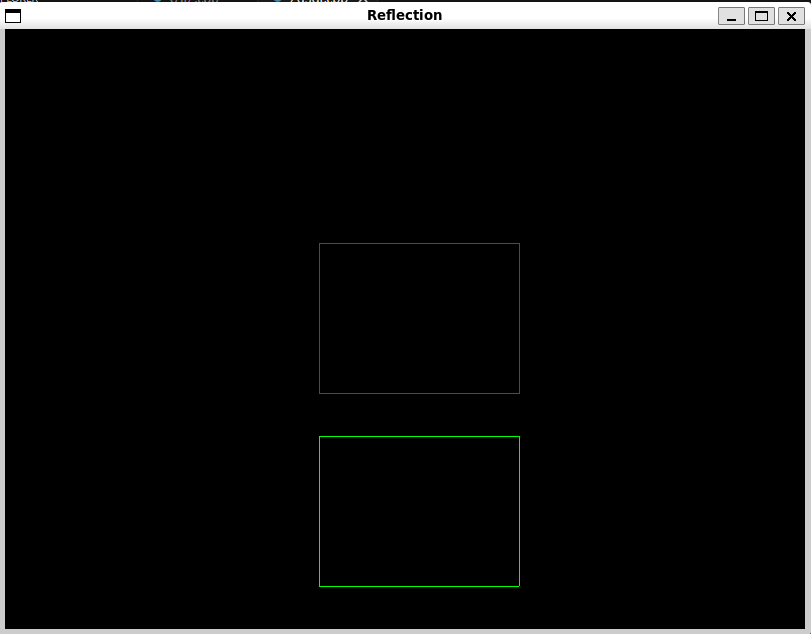
**init();**

**glutDisplayFunc(display);**

**glutMainLoop();**

**return 0;**

**}**

****

**1.** **3D TRANSFORMATIONS**

**a.** **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);**

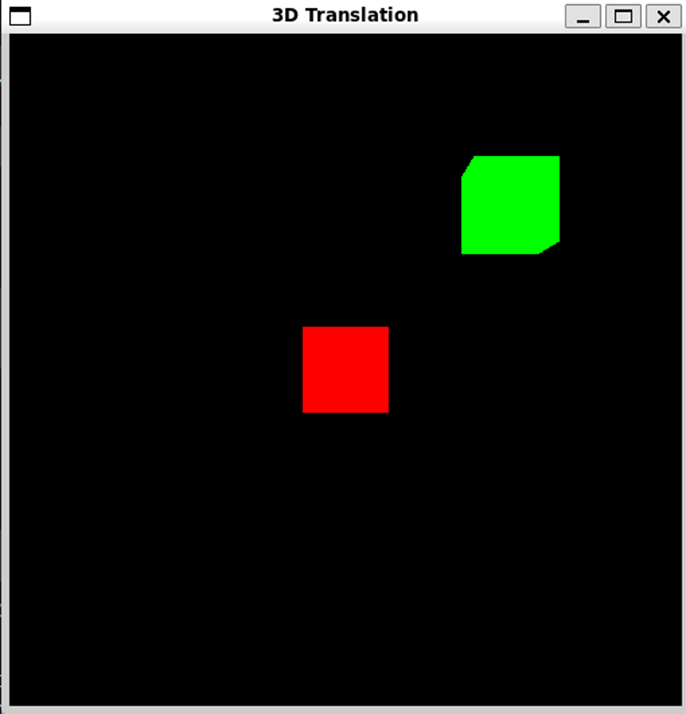
**glutReshapeFunc(reshape);**

**glutMainLoop();**

**return 0;**

**}**

**Output:**

****

**b.** **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);**

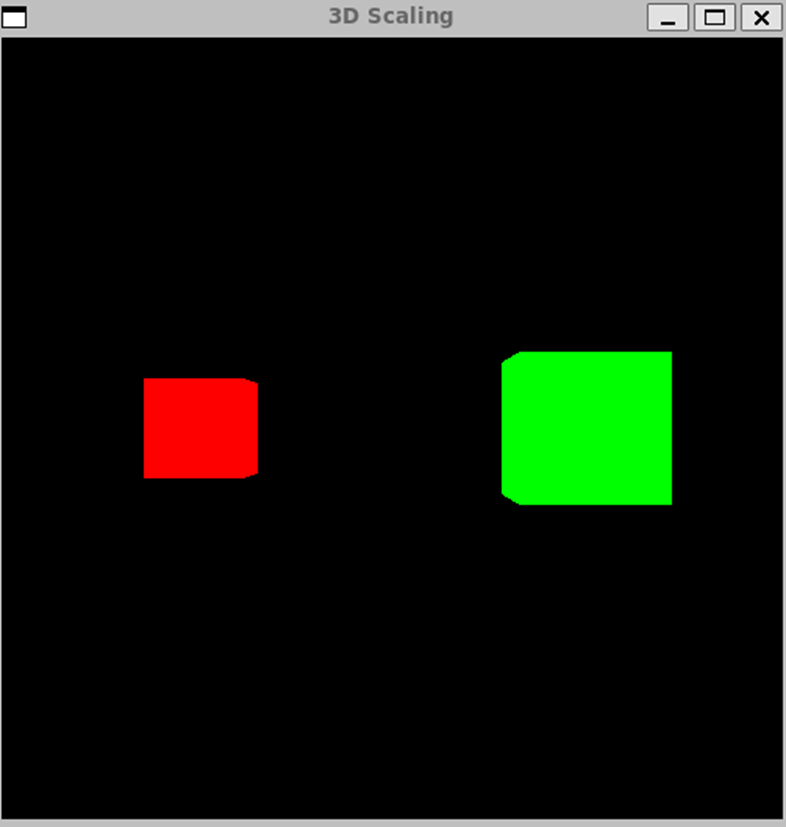
**glutReshapeFunc(reshape);**

**glutMainLoop();**

**return 0;**

**}**

**Output:**

****

**c.** **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);**

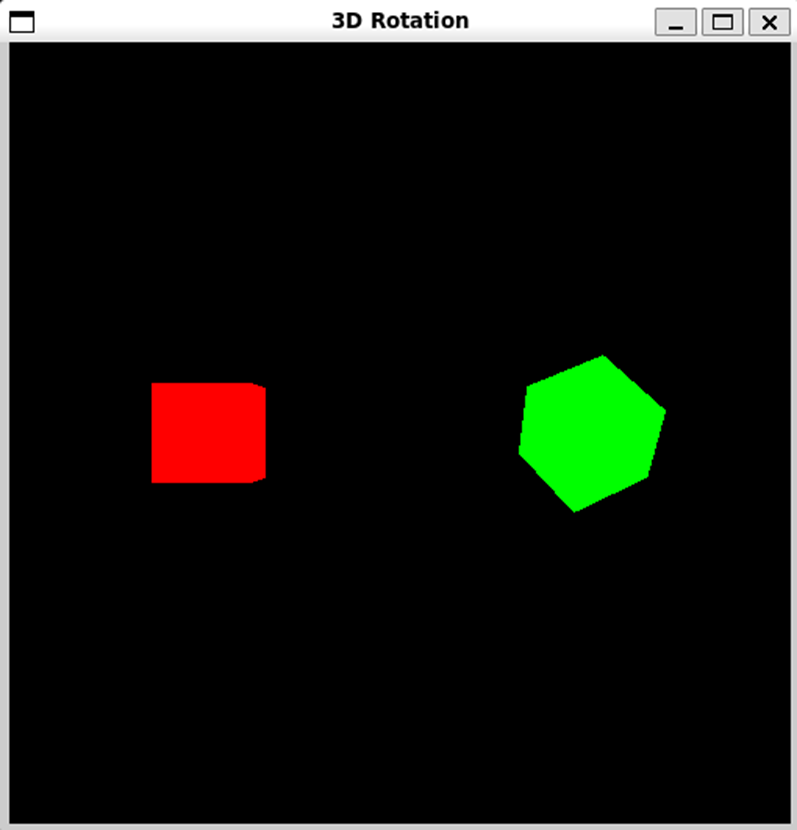
**glutReshapeFunc(reshape);**

**glutMainLoop();**

**return 0;**

**}**

**Output:**

****

**d.** **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);**

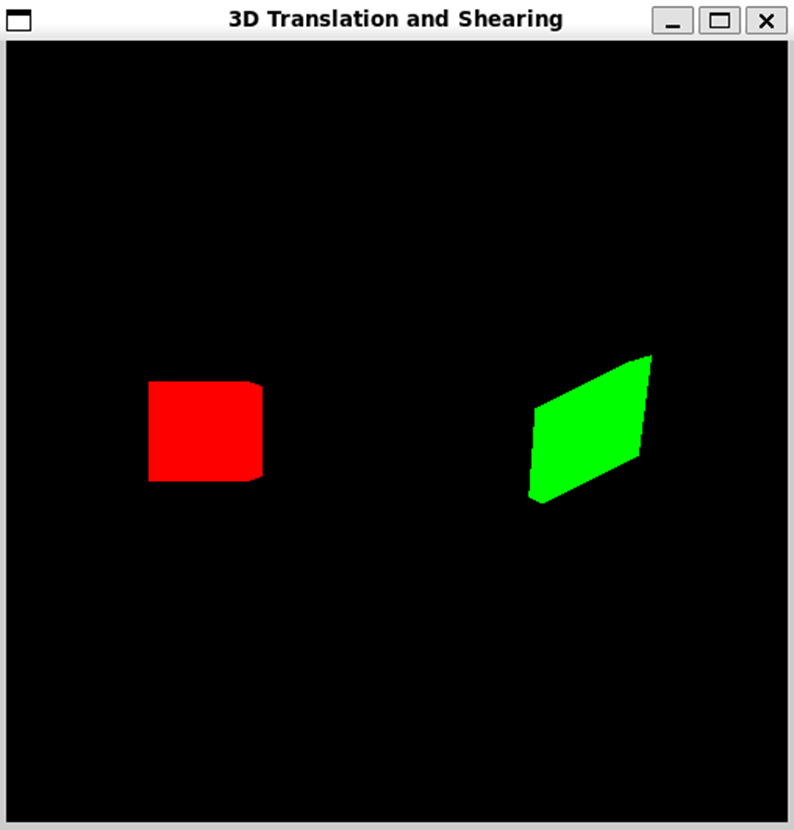
**glutReshapeFunc(reshape);**

**glutMainLoop();**

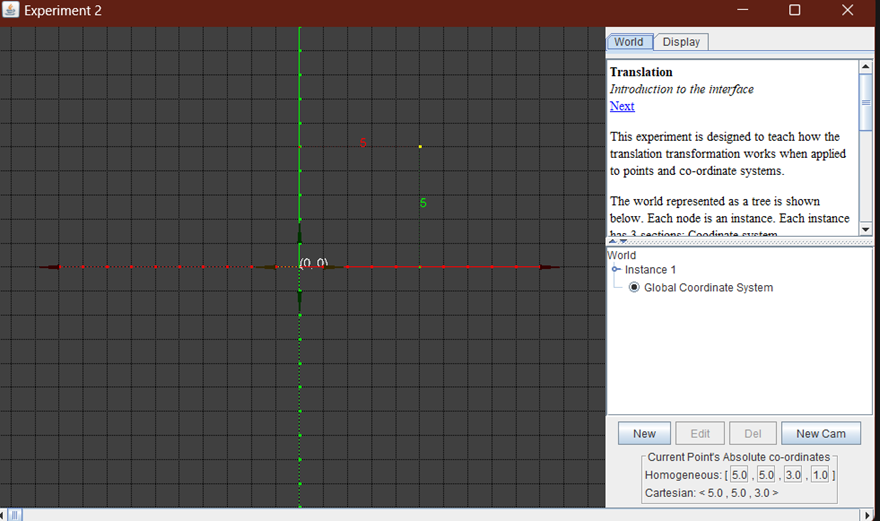
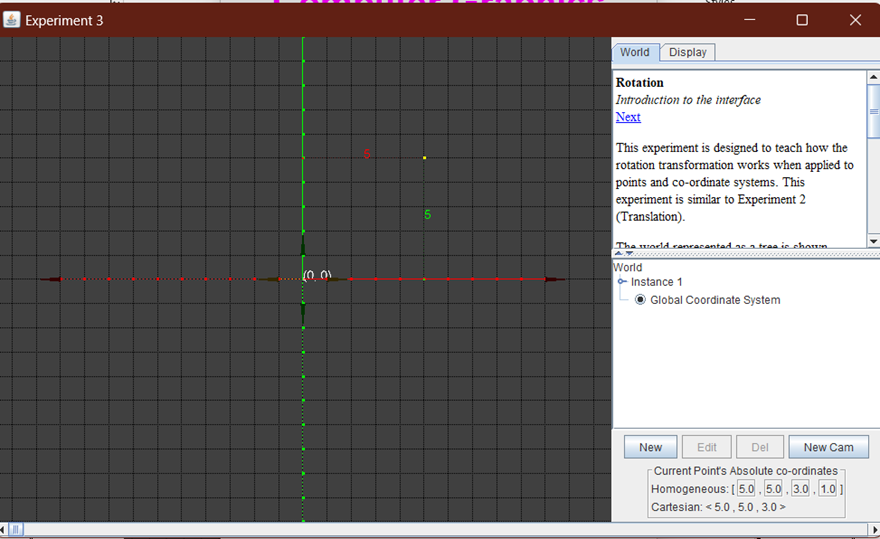
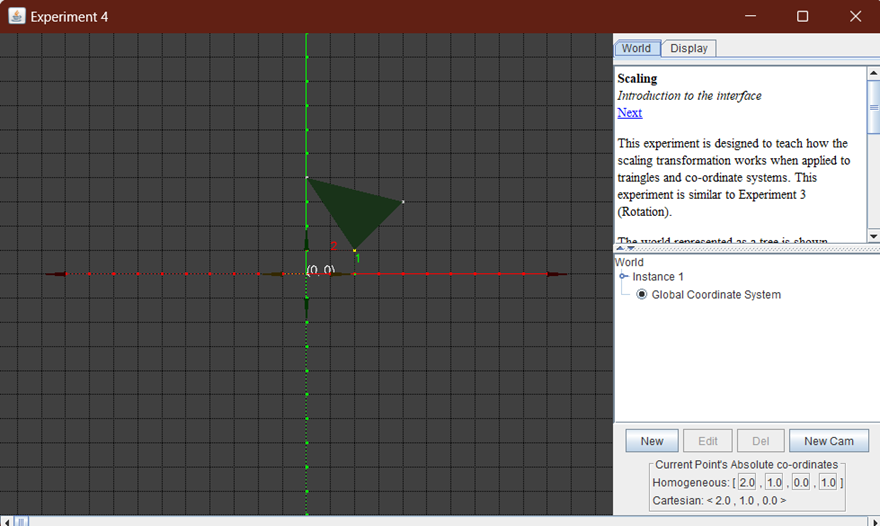
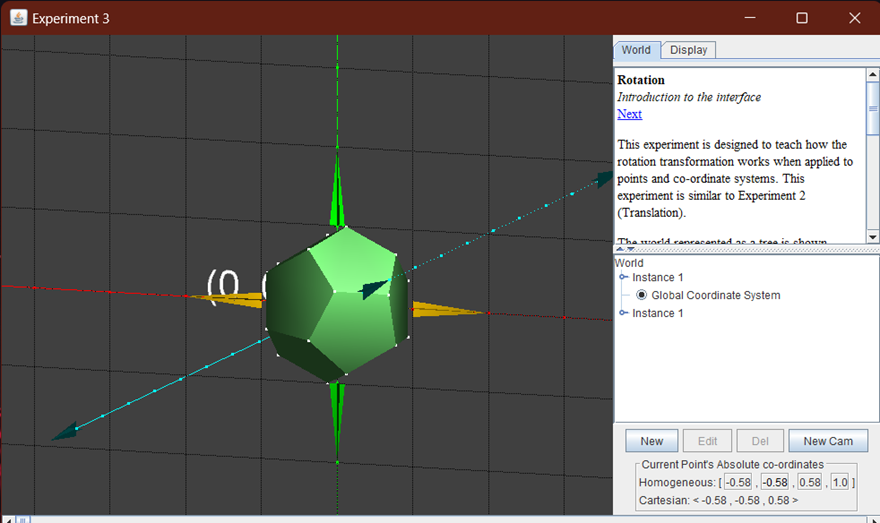
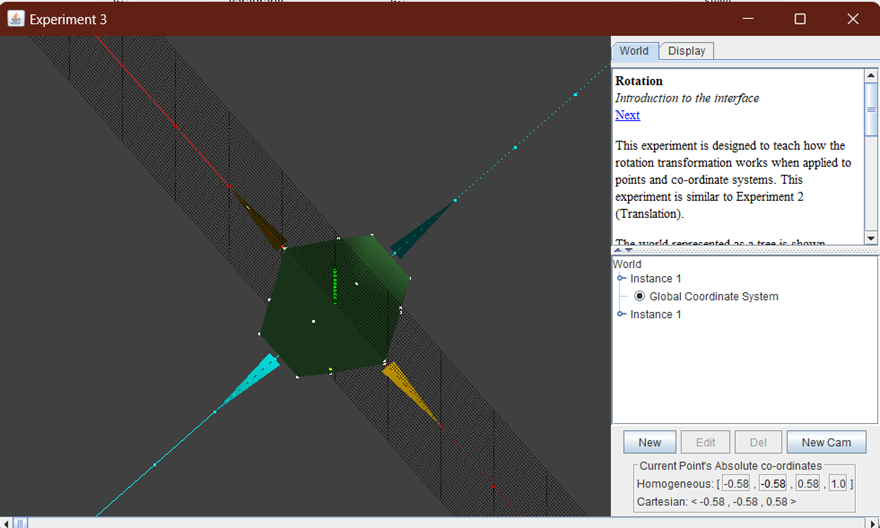
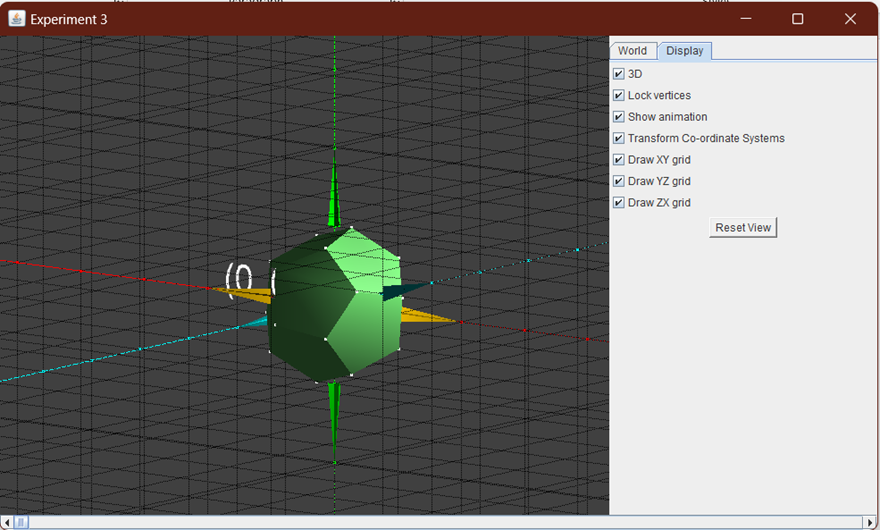
**return 0;**

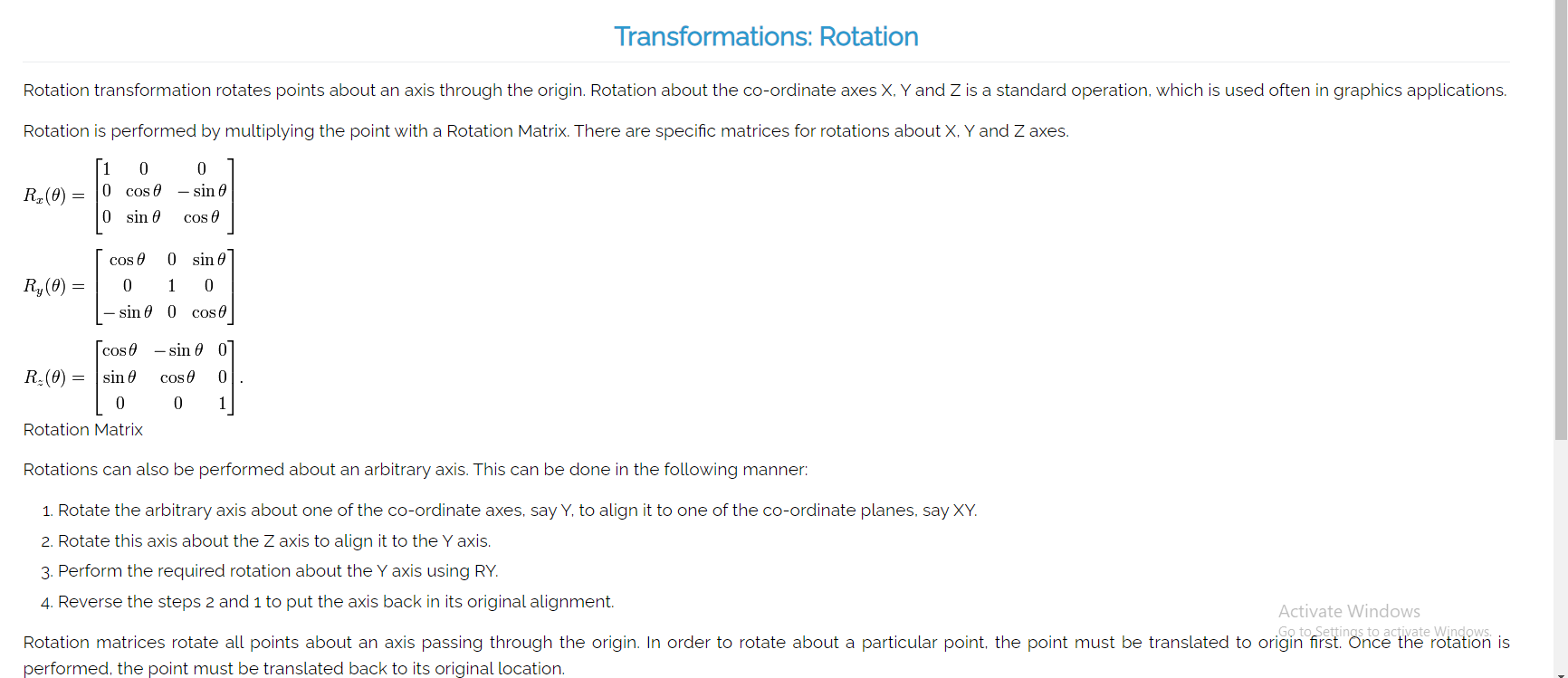
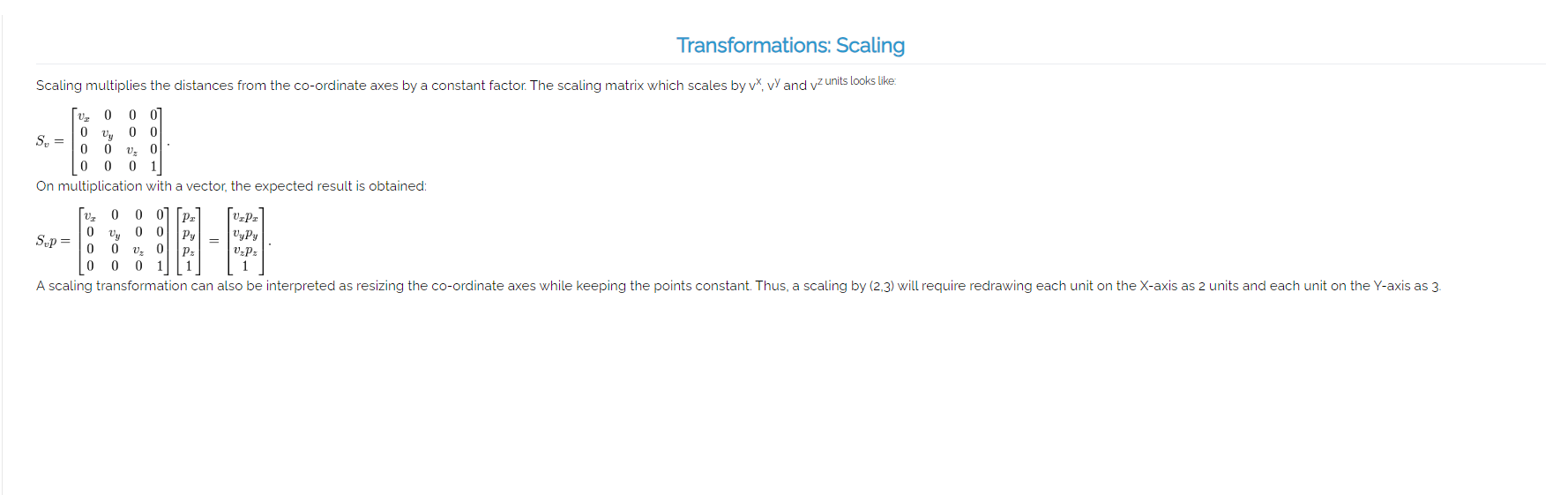
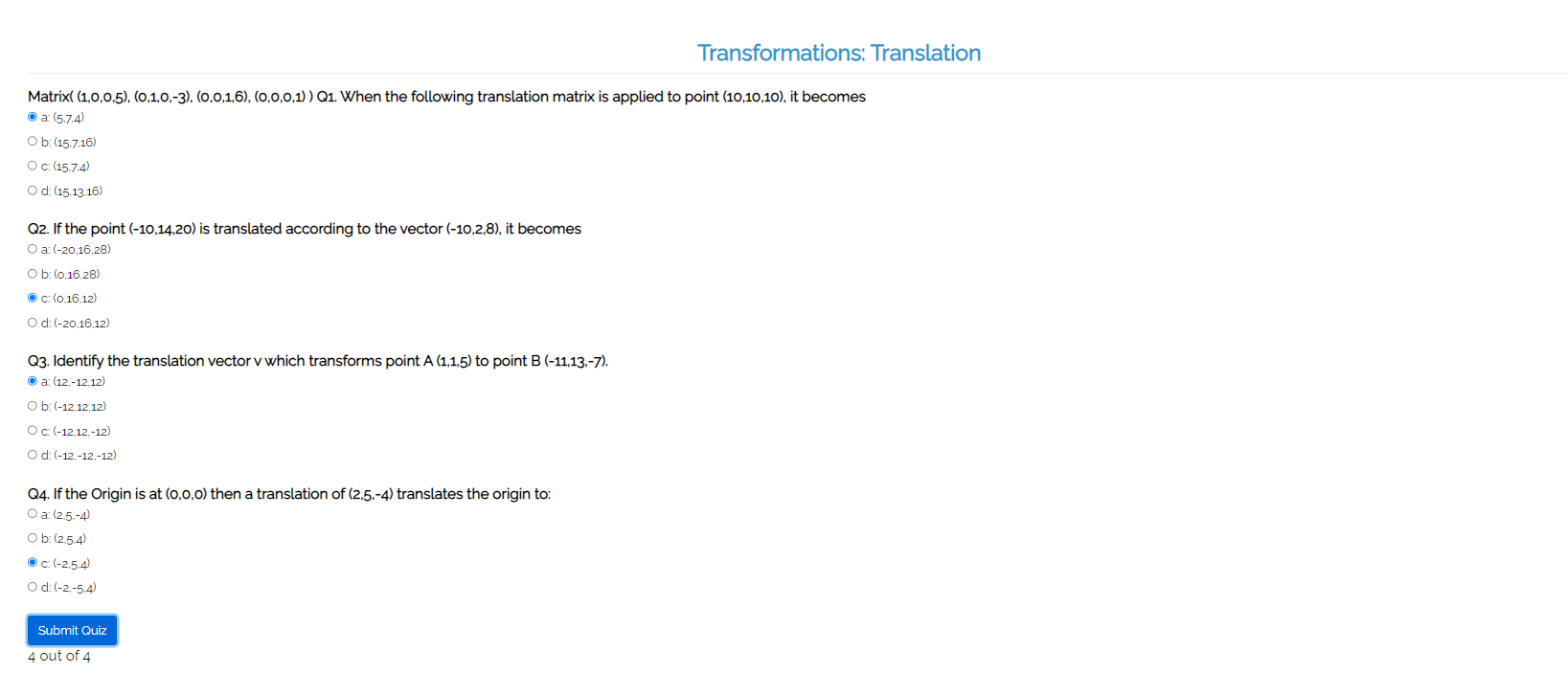
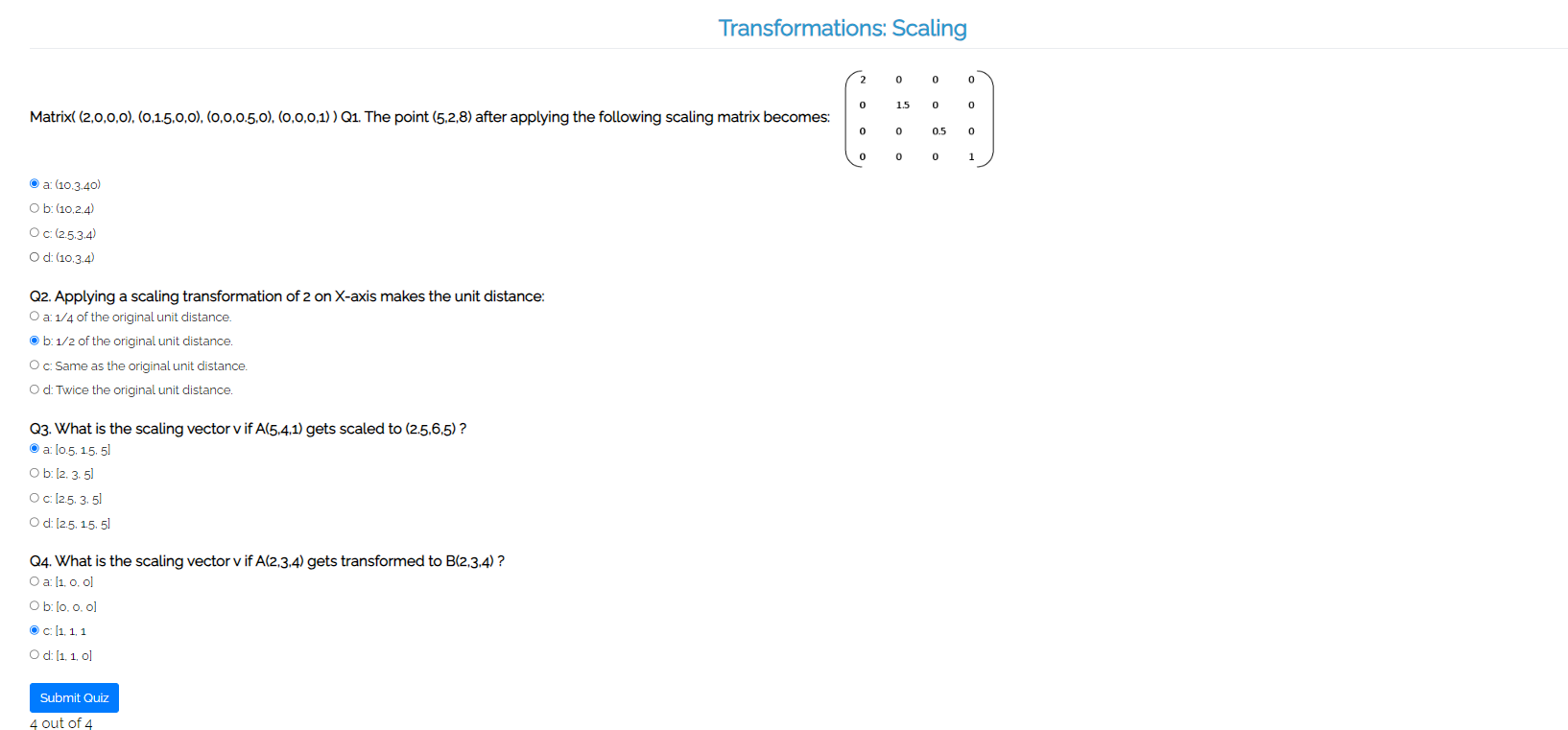
**}**

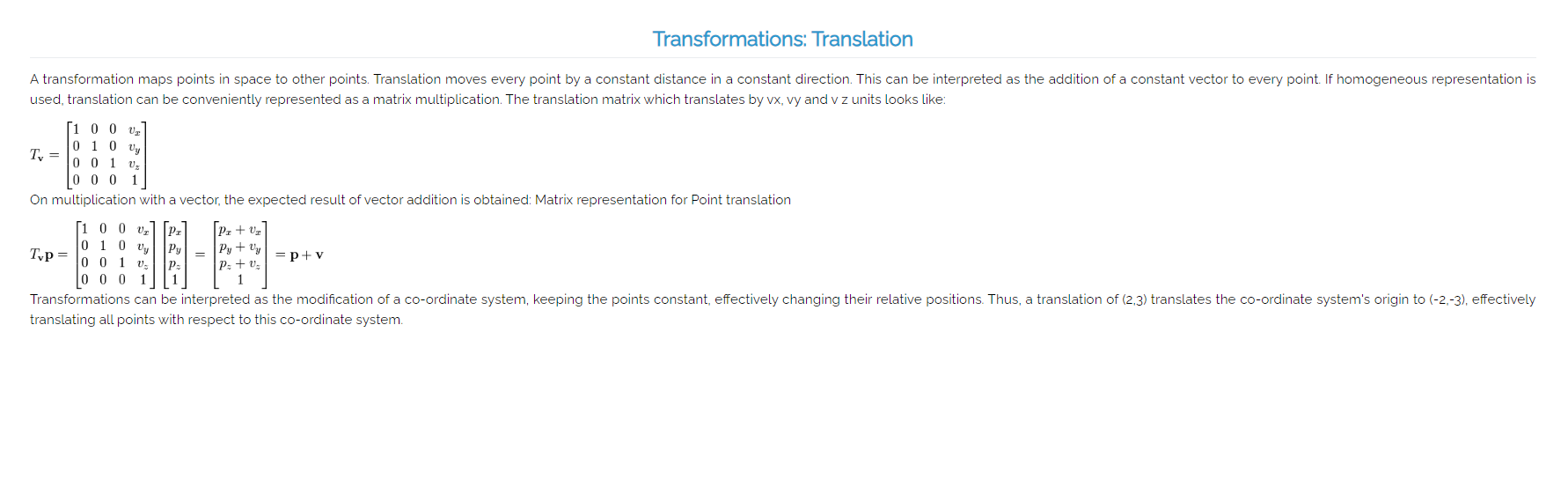
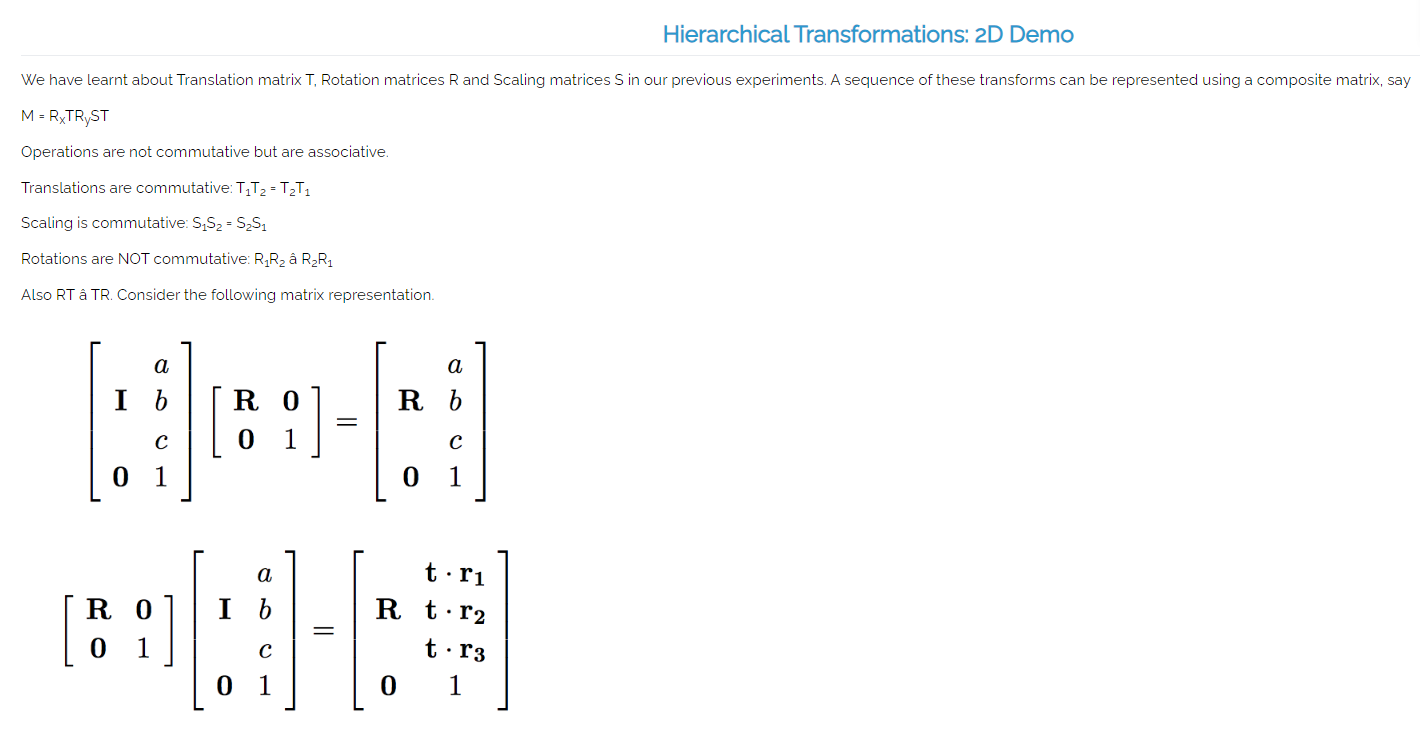
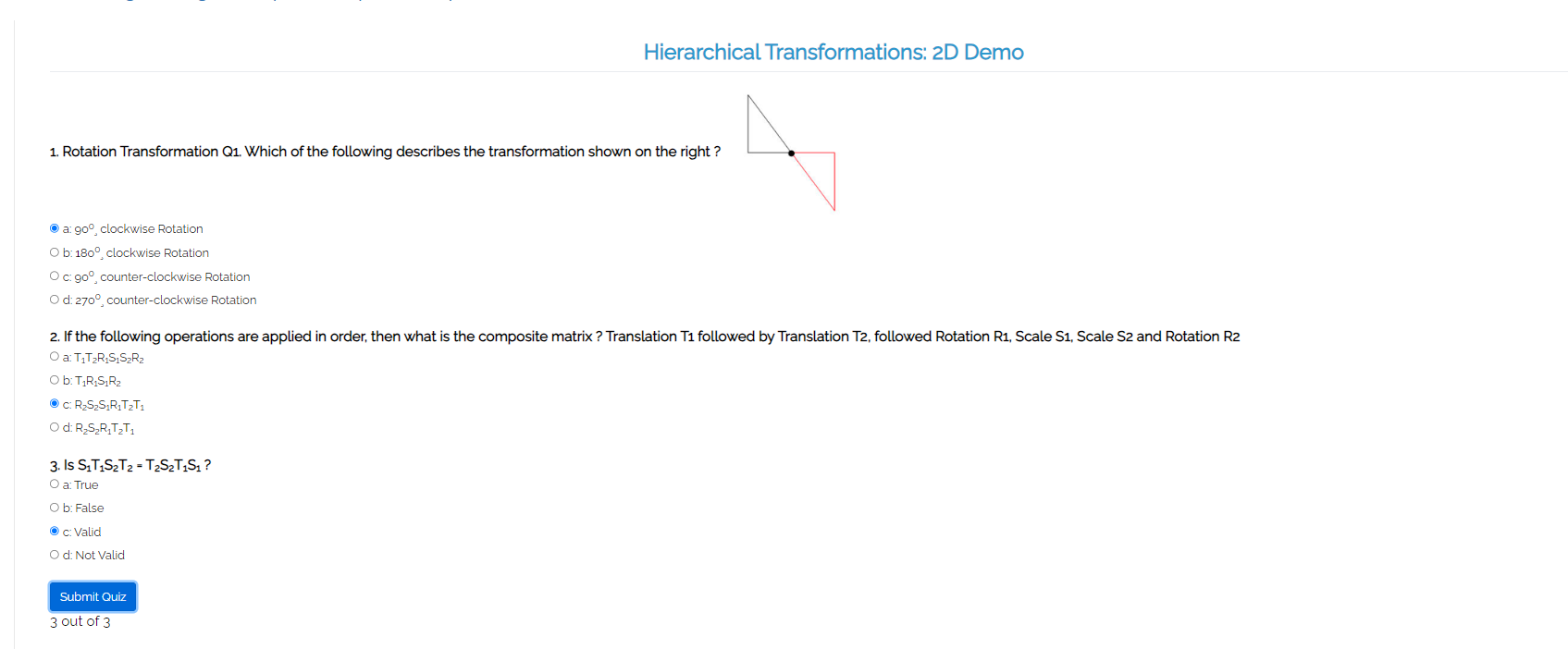
**Output:**

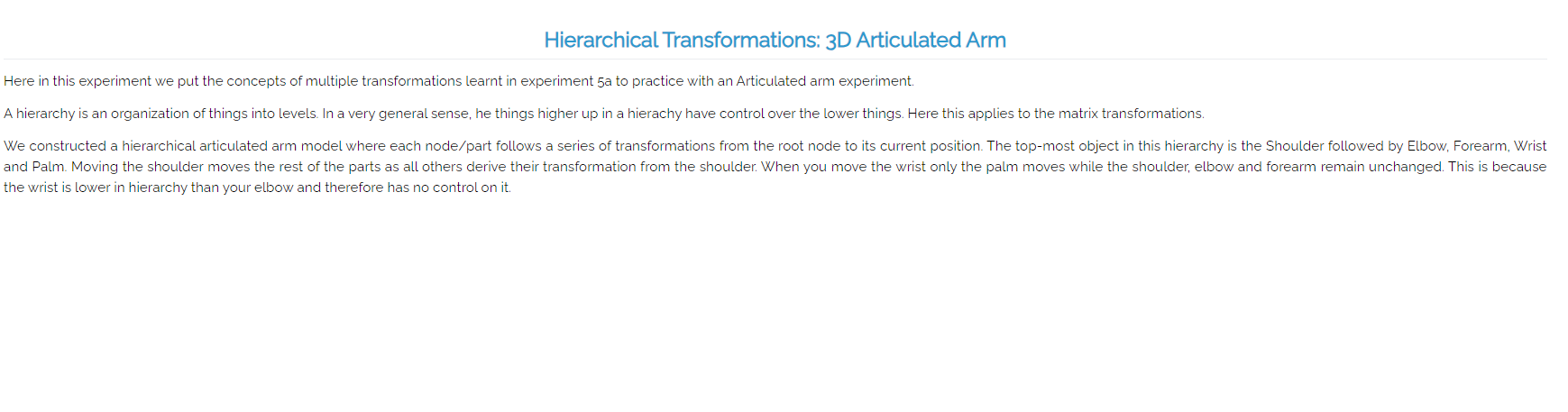
****

**Screenshots from VLab(if any):**

**  
  
**  
  
  


**  
  
  
  
**

**  
  
  
  
**

****

**Conclusion and discussion:**

Learned to perform 2D and 3D Transformations - scaling, shearing, rotation, reflection , translation - using openGL.

**Date: 29/8/24**

**Signature of faculty in-charge**

Post lab

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

**#include <GL/glut.h>  
GLfloat vertices[3][2] = {**

**{0.0f, 1.0f},**

**{-1.0f, -1.0f},**

**{1.0f, -1.0f}**

**};**

**void drawSierpinski(GLfloat v1[2], GLfloat v2[2], GLfloat v3[2], int depth) {**

**if (depth == 0) {**

**glBegin(GL\_TRIANGLES);**

**glVertex2fv(v1);**

**glVertex2fv(v2);**

**glVertex2fv(v3);**

**glEnd();**

**} else {**

**GLfloat v12[2], v23[2], v31[2];**

**v12[0] = (v1[0] + v2[0]) / 2.0f;**

**v12[1] = (v1[1] + v2[1]) / 2.0f;**

**v23[0] = (v2[0] + v3[0]) / 2.0f;**

**v23[1] = (v2[1] + v3[1]) / 2.0f;**

**v31[0] = (v3[0] + v1[0]) / 2.0f;**

**v31[1] = (v3[1] + v1[1]) / 2.0f;**

**drawSierpinski(v1, v12, v31, depth - 1);**

**drawSierpinski(v12, v2, v23, depth - 1);**

**drawSierpinski(v31, v23, v3, depth - 1);**

**}**

**}**

**void display() {**

**glClear(GL\_COLOR\_BUFFER\_BIT);**

**glColor3f(1.0f, 1.0f, 1.0f);**

**GLfloat v1[2] = {0.0f, 1.0f};**

**GLfloat v2[2] = {-1.0f, -1.0f};**

**GLfloat v3[2] = {1.0f, -1.0f};**

**int depth = 5;**

**drawSierpinski(v1, v2, v3, depth);**

**glutSwapBuffers();**

**}**

**void init() {**

**glClearColor(0.0f, 0.0f, 0.0f, 1.0f);**

**glMatrixMode(GL\_PROJECTION);**

**glLoadIdentity();**

**gluOrtho2D(-1.5, 1.5, -1.5, 1.5);**

**glMatrixMode(GL\_MODELVIEW);**

**}**

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

**glutInit(&argc, argv);**

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

**glutInitWindowSize(800, 800);**

**glutCreateWindow("Sierpinski Gasket");**

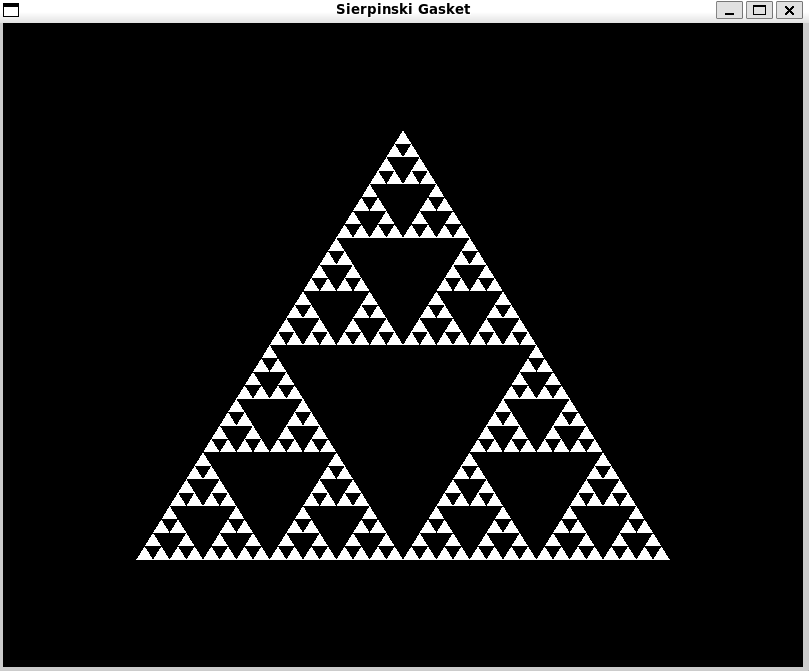
**init();**

**glutDisplayFunc(display);**

**glutMainLoop();**

**return 0;**

**}**

****

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