205001085
SABARIVASAN V

Lab Exercise 5: 2D Transformations in C++ using OpenGL

Aim:

To apply the following 2D transformations on objects and to render the final output along with the original object.

1) Translation 2) Rotation

a) about origin

b) with respect to a fixed point (xr,yr) 3) Scaling with respect to

a) origin - Uniform Vs Differential Scaling

b) fixed point (xf,yf) 4) Reflection with respect to

a) x-axis
b) y-axis
c) origin
d) the line x=y

5) Shearing
a) x-direction shear

b) y-direction shear

Algorithm:

Application of a sequence of transformations to a point:

P’ = M2.M1.P

= M.P

Composite transformations are formed by calculating the matrix product of the individual transformations and forming products of the transformation matrix.

Code:

```
#include <stdio.h>
#include <iostream>
#include<GLUT/glut.h>
#include <cmath>
using namespace std;
void plot(int x1,int x2,int y1, int y2)
{
glBegin(GL_LINES);
glVertex2i(x1,y1);
glVertex2i(x2, y2);
glEnd();
}
void myInit (void)
{
glClearColor(1.0, 1.0, 1.0, 0.0);
glColor3f(0.0f, 0.0f, 0.0f);
glPointSize(4.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(-320.0,320.0,-240.0,240.0); }
void matmul(float M[3][3],int P[3][2],float res[3][2]){ res[0][0]=0;res[1][0]=0;res[2][0]=0;
res[0][1]=0;res[1][1]=0;res[2][1]=0;
for(int i=0;i<3;i++){
```

```
for(int j=0;j<2;j++){
for(int k=0;k<3;k++){
res[i][j]+=M[i][k]*P[k][j];
}
}
}
}
void matmul2(float A[][3],float B[][3],float M[][3]){ for(int i=0;i<3;i++){
for(int j=0;j<3;j++){
for(int k=0;k<3;k++){
M[i][j]+=A[i][k]*B[k][j];
}
}
}
}
void matmul3(float M[][3],int P[3][4],float res[3][4]){ res[0][0]=0;res[1][0]=0;res[2][0]=0;
res[0][1]=0;res[1][1]=0;res[2][1]=0;
res[0][2]=0;res[1][2]=0;res[2][2]=0;
res[0][3]=0;res[1][3]=0;res[2][3]=0;
for(int i=0;i<3;i++){
for(int j=0;j<4;j++){
for(int k=0;k<3;k++){
res[i][j]+=M[i][k]*P[k][j];
}
}
```

```
}

}

void translate(int x1,int y1,int x2,int y2,int tx,int ty){

plot(x1,x2,y1,y2);

int hom[3][2];

hom[0][0]=x1;hom[1][0]=y1;hom[2][0]=1;

hom[0][1]=x2;hom[1][1]=y2;hom[2][1]=1;

float T[3][3];

T[0][0]=1;

T[0][1]=0;

T[0][2]=tx;

T[1][0]=0;

T[1][1]=1;

T[1][2]=ty;

T[2][0]=0;

T[2][1]=0;

T[2][2]=1;

float res[3][2];

matmul(T,hom,res);

plot((int)round(res[0][0]),(int)round(res[0][1]),(int)round(res[1][0]),(int)round(res[1][1])); }

void rotation(int x1,int y1,int x2,int y2,int tx,int ty){

plot(x1,x2,y1,y2);

int hom[3][2];

hom[0][0]=x1;hom[1][0]=y1;hom[2][0]=1;

hom[0][1]=x2;hom[1][1]=y2;hom[2][1]=1;
```

```
float T[3][3]={0.0};

T[0][0]=1;

T[0][1]=0;

T[0][2]=tx;

T[1][0]=0;

T[1][1]=1;

T[1][2]=ty;

T[2][0]=0;

T[2][1]=0;

T[2][2]=1;

float rad;

cout<<"Enter angle in radians:";cin>>rad;

float T2[3][3];

T2[0][0]=cos(rad);

T2[0][1]=-sin(rad);

T2[0][2]=0;

T2[1][0]=sin(rad);

T2[1][1]=cos(rad);

T2[1][2]=0;

T2[2][0]=0;

T2[2][1]=0;

T2[2][2]=1;

float res[3][2]={0.0},M[3][3]={0.0},M2[3][3]={0.0};

matmul2(T,T2,M);

T[0][0]=1;
```

```
T[0][1]=0;
T[0][2]=-tx;
T[1][0]=0;
T[1][1]=1;
T[1][2]=-ty;
T[2][0]=0;
T[2][1]=0;
T[2][2]=1;
matmul2(M, T,M2);
matmul(M2, hom, res);
plot((int)round(res[0][0]),(int)round(res[0][1]),(int)round(res[1][0]),(int)round(res[1][1])); }
void scaling(int x1,int y1,int x2,int y2,int tx,int ty){
plot(x1,x2,y1,y2);
int hom[3][2];
hom[0][0]=x1;hom[1][0]=y1;hom[2][0]=1;
hom[0][1]=x2;hom[1][1]=y2;hom[2][1]=1;
float T[3][3]={0.0};
T[0][0]=1;
T[0][1]=0;
T[0][2]=tx;
T[1][0]=0;
T[1][1]=1;
T[1][2]=ty;
T[2][0]=0;
T[2][1]=0;
```

```
T[2][2]=1;
float sx,sy;
cout<<"Enter sx,sy:";cin>>sx>>sy;
float T2[3][3];
T2[0][0]=sx;
T2[0][1]=0;
T2[0][2]=0;
T2[1][0]=0;
T2[1][1]=sy;
T2[1][2]=0;
T2[2][0]=0;
T2[2][1]=0;
T2[2][2]=1;
float res[3][2]={0.0},M[3][3]={0.0},M2[3][3]={0.0};
matmul2(T,T2,M);
T[0][0]=1;
T[0][1]=0;
T[0][2]=-tx;
T[1][0]=0;
T[1][1]=1;
T[1][2]=-ty;
T[2][0]=0;
T[2][1]=0;
T[2][2]=1;
matmul2(M, T,M2);
```

```
matmul(M2, hom, res);
plot((int)round(res[0][0]),(int)round(res[0][1]),(int)round(res[1][0]),(int)round(res[1][1])); }
void reflection(int x1,int y1,int x2,int y2){
plot(x1,x2,y1,y2);
int hom[3][2];
hom[0][0]=x1;hom[1][0]=y1;hom[2][0]=1;
hom[0][1]=x2;hom[1][1]=y2;hom[2][1]=1;
float T[3][3]={0.0};
T[0][0]=1;
T[0][1]=0;
T[0][2]=0;
T[1][0]=0;
T[1][1]=-1;
T[1][2]=0;
T[2][0]=0;
T[2][1]=0;
T[2][2]=1;
float res[3][2]={0.0};
matmul(T, hom, res);
plot((int)round(res[0][0]),(int)round(res[0][1]),(int)round(res[1][0]),(int)round(res[1][1]));
T[0][0]=-1;
T[0][1]=0;
T[0][2]=0;
T[1][0]=0;
T[1][1]=1;
```

```
T[1][2]=0;
T[2][0]=0;
T[2][1]=0;
T[2][2]=1;
matmul(T, hom, res);
plot((int)round(res[0][0]),(int)round(res[0][1]),(int)round(res[1][0]),(int)round(res[1][1]));
T[0][0]=-1;
T[0][1]=0;
T[0][2]=0;
T[1][0]=0;
T[1][1]=-1;
T[1][2]=0;
T[2][0]=0;
T[2][1]=0;
T[2][2]=1;
matmul(T, hom, res);
plot((int)round(res[0][0]),(int)round(res[0][1]),(int)round(res[1][0]),(int)round(res[1][1]));
T[0][0]=0;
T[0][1]=1;
T[0][2]=0;
T[1][0]=1;
T[1][1]=0;
T[1][2]=0;
T[2][0]=0;
T[2][1]=0;
```

```
T[2][2]=1;
matmul(T, hom, res);
plot((int)round(res[0][0]),(int)round(res[0][1]),(int)round(res[1][0]),(int)round(res[1][1])); }
void shearing(int x1,int y1,int x2,int y2,int x3,int y3,int x4,int y4){ glBegin(GL_QUADS);
glVertex2d(x1, y1);
glVertex2d(x2, y2);
glVertex2d(x3, y3);
glVertex2d(x4, y4);
glEnd();
int hom[3][4];
hom[0][0]=x1;hom[1][0]=y1;hom[2][0]=1;
hom[0][1]=x2;hom[1][1]=y2;hom[2][1]=1;
hom[0][2]=x3;hom[1][2]=y3;hom[2][2]=1;
hom[0][3]=x4;hom[1][3]=y4;hom[2][3]=1;
float shx,shy;
cout<<"Enter shx,shy:";
cin>>shx>>shy;
float T[3][3]={0.0};
T[0][0]=1;
T[0][1]=shx;
T[0][2]=0;
T[1][0]=0;
T[1][1]=1;
T[1][2]=0;
T[2][0]=0;
```

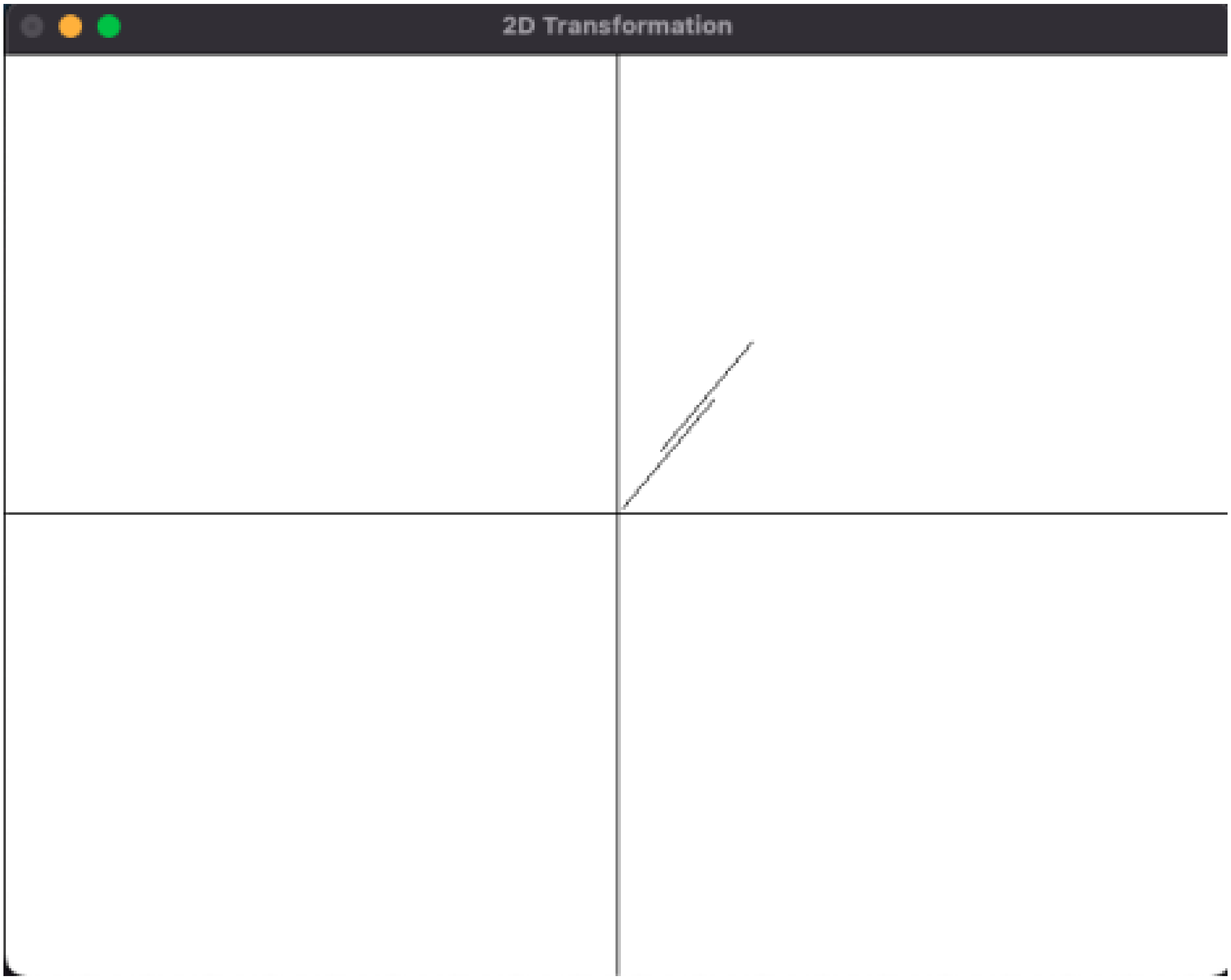
```
T[2][1]=0;

T[2][2]=1;

float res[3][4]={0.0};

matmul3(T, hom, res);

glBegin(GL_QUADS);

glVertex2d((int)round(res[0][0]), (int)round(res[1][0]));

glVertex2d((int)round(res[0][1]), (int)round(res[1][1]));

glVertex2d((int)round(res[0][2]), (int)round(res[1][2]));

glVertex2d((int)round(res[0][3]), (int)round(res[1][3]));

glEnd();

T[0][0]=1;

T[0][1]=0;

T[0][2]=0;

T[1][0]=shy;

T[1][1]=1;

T[1][2]=0;

T[2][0]=0;

T[2][1]=0;

T[2][2]=1;

matmul3(T, hom, res);

glBegin(GL_QUADS);

glVertex2d((int)round(res[0][0]), (int)round(res[1][0])); glVertex2d((int)round(res[0][1]),
(int)round(res[1][1])); glVertex2d((int)round(res[0][2]), (int)round(res[1][2]));
glVertex2d((int)round(res[0][3]), (int)round(res[1][3])); glEnd();

}

void myDisplay(void)
```

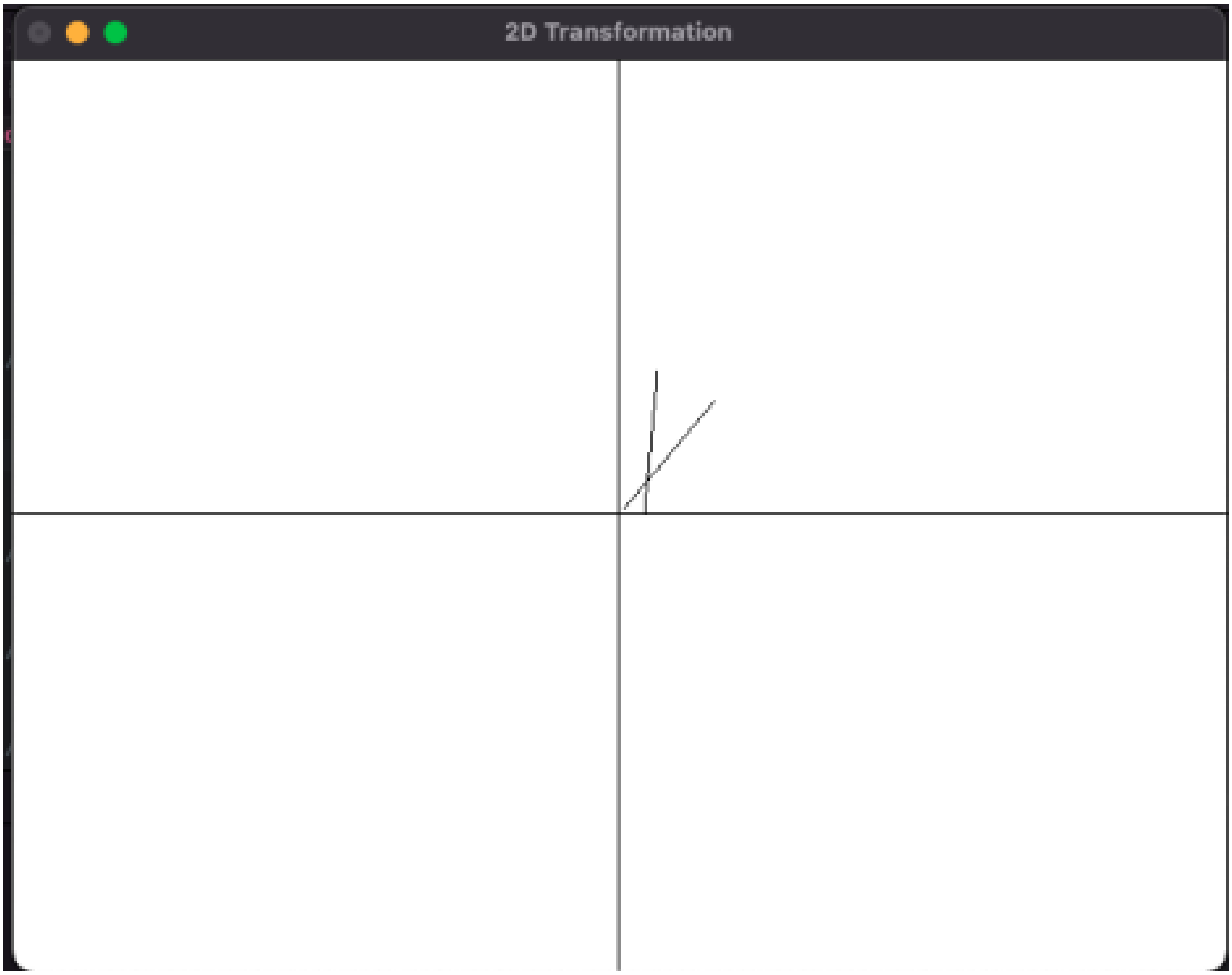
```
{
glClear (GL_COLOR_BUFFER_BIT);
glColor3f (0.0, 0.0, 0.0);
glPointSize(1.0);
glBegin(GL_LINES);
glVertex2i(-320, 0);
glVertex2i(320, 0);
glEnd();
glBegin(GL_LINES);
glVertex2i(0, -240);
glVertex2i(0,240 );
glEnd();
int x1,x2,y1,y2;
cout << "X-coordinate 1 : "; cin >> x1;
cout << "\nY-coordinate 1 : "; cin >> y1;
cout << "X-coordinate 2 : "; cin >> x2;
cout << "\nY-coordinate 2 : "; cin >> y2;
int tx,ty;
// cout<<"Enter tx,ty:";cin>>tx;cin>>ty;
// translate(x1,y1,x2,y2,tx,ty);
//rotation
// rotation(x1,y1,x2,y2,tx,ty);
//scaling
// scaling(x1,y1,x2,y2,tx,ty);
//reflection
```

```
// reflection(x1,y1,x2,y2);

//shearing

int x3,y3,x4,y4;

cout<<"X-coordinate 3";

cin>>x3;

cout<<"Y-coordinate 3";

cin>>y3;

cout<<"X-coordinate 4";

cin>>x4;

cout<<"Y-coordinate 4";

cin>>y4;

shearing(x1,y1,x2,y2,x3,y3,x4,y4);

glFlush ();

}

int main(int argc, char** argv)

{

glutInit(&argc, argv);

glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);

glutInitWindowSize (640, 480);

// glutInitWindowPosition (100, 150);

glutCreateWindow ("2D Transformation");

glutDisplayFunc(myDisplay);

myInit ();

glutMainLoop();

return 0;
```

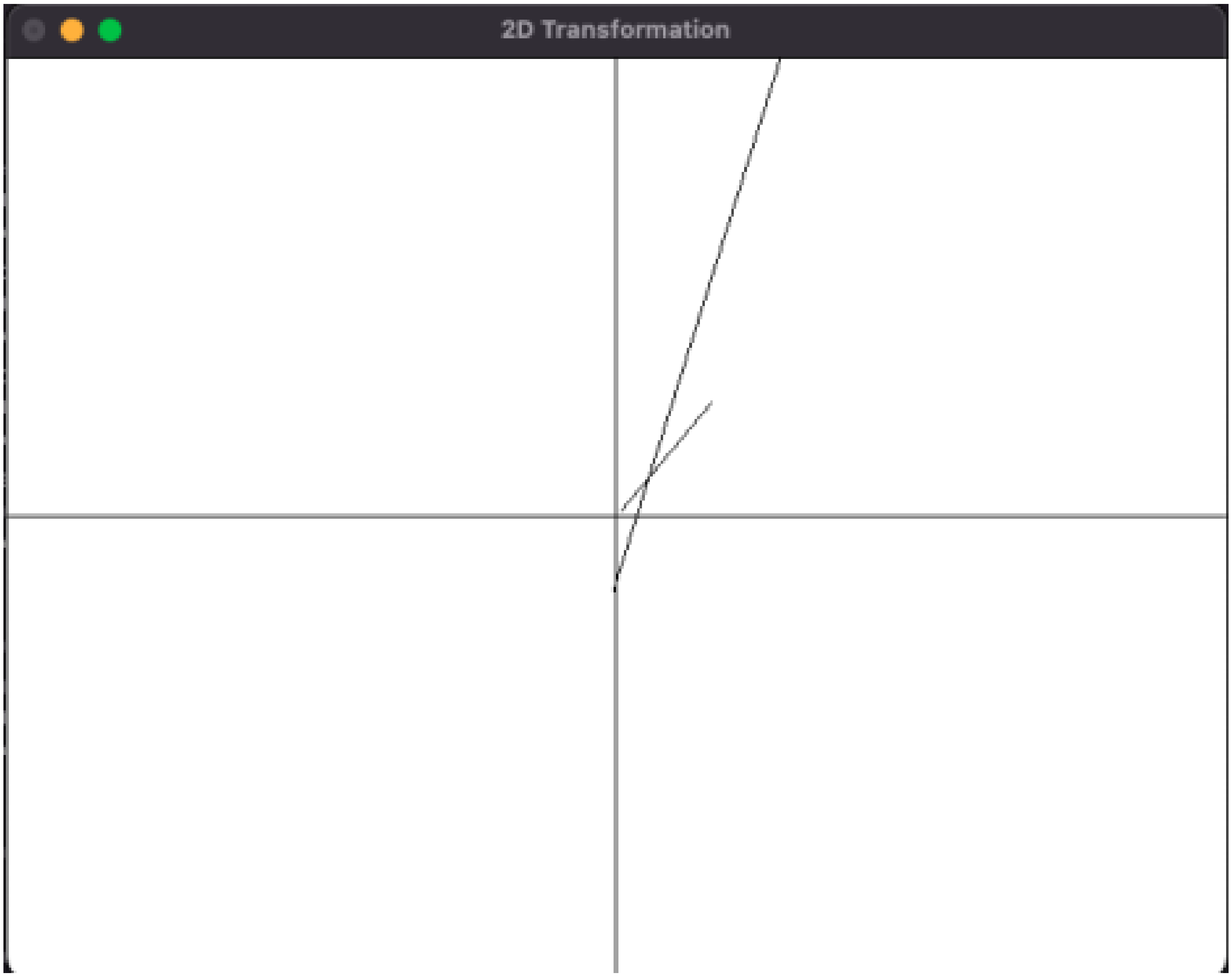
```
}
```

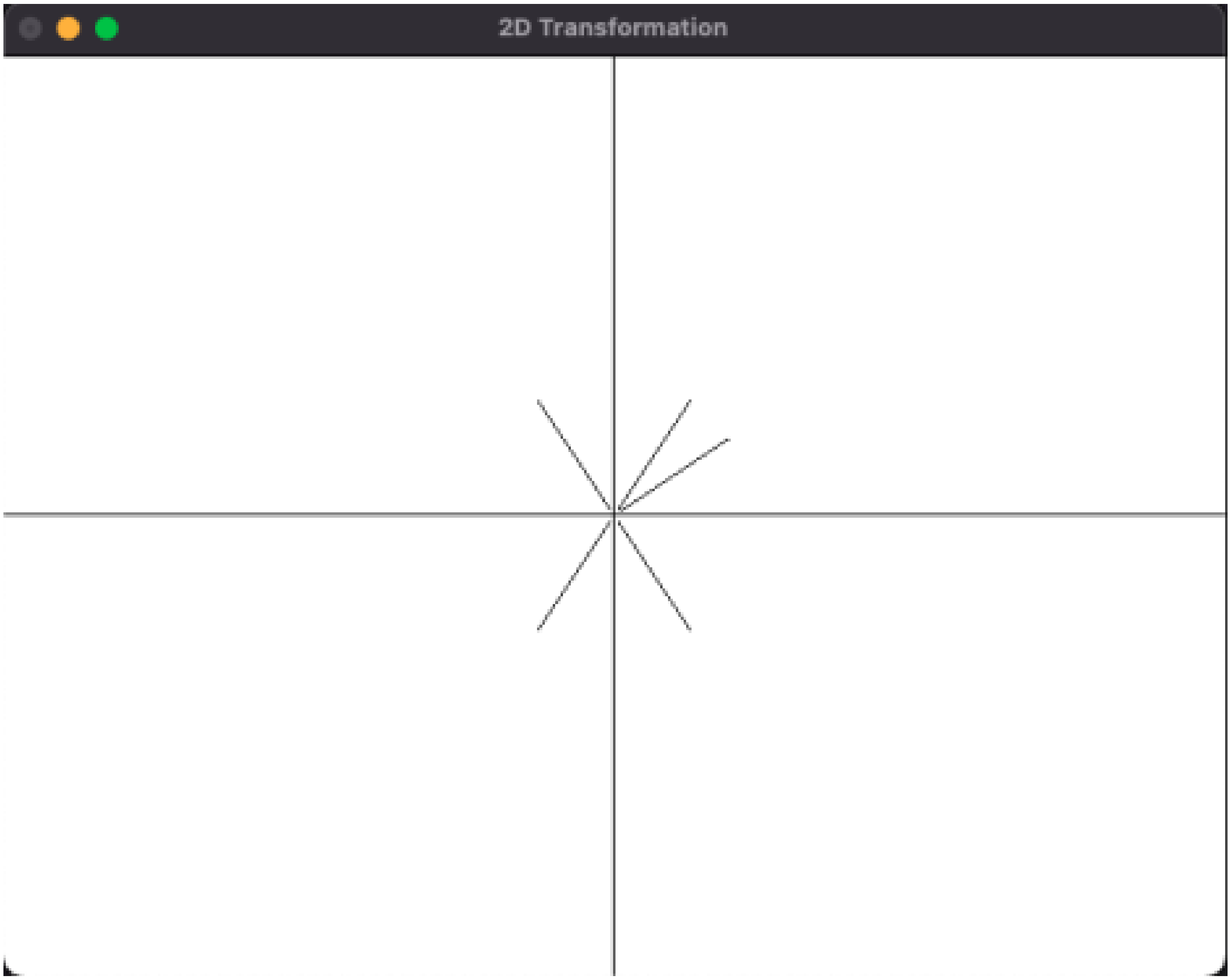
Output:



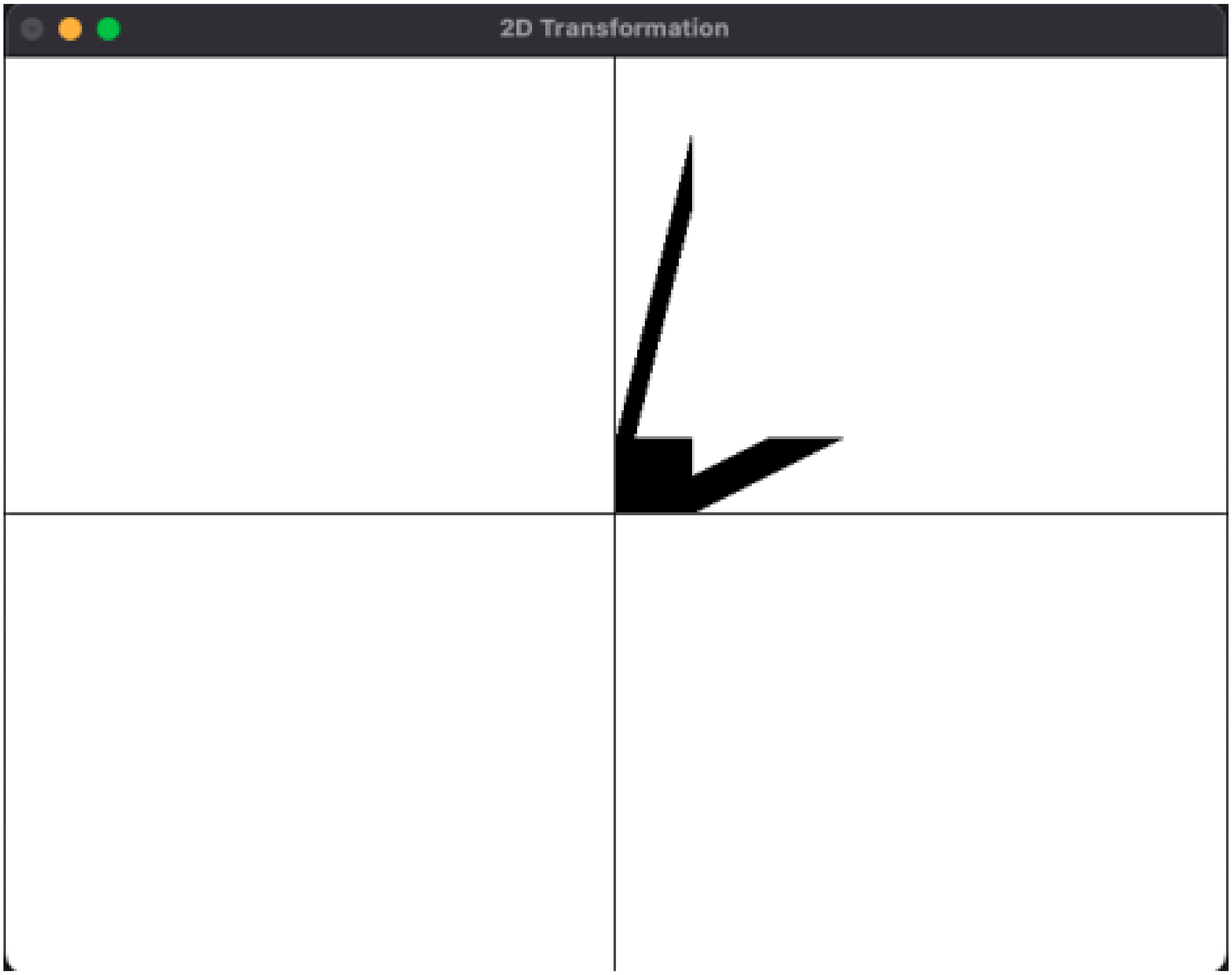


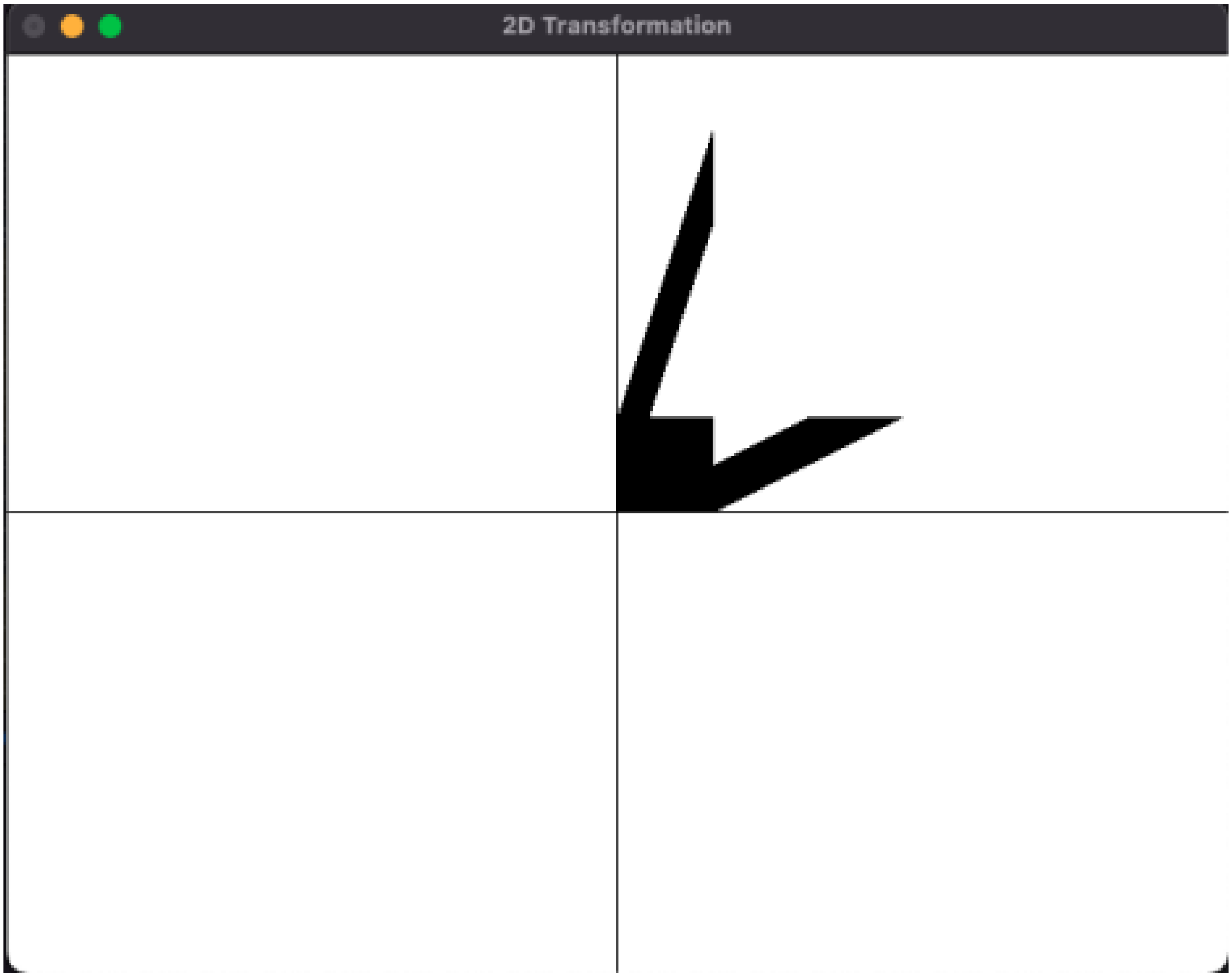
2D Transformation


2D Transformation



2D Transformation


2D Transformation



Learning Outcome:

Learnt to do composite transformations.

Learnt to do translation, reflection, shearing, rotation and scaling.