**SSN COLLEGE OF ENGINEERING, KALAVAKKAM  
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
UCS1712 – GRAPHICS AND MULTIMEDIA LAB ------------------------------------------------------------------------------------------------------------**

**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.

**Algorithm:**

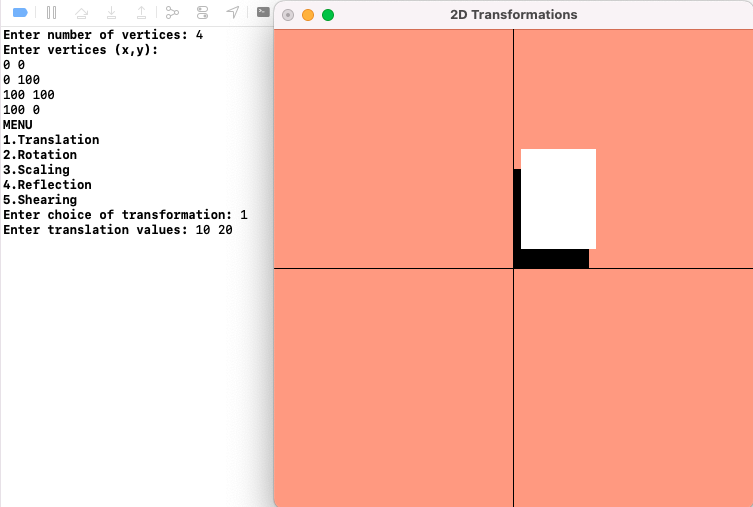
1. Input the number of vertices and the vertices
2. Plot the original shape (black color)
3. Input the transformation choice and all the other required values
4. Perform matrix multiplication with appropriate matrix for transformation
5. Plot the transformed shape (white color)

**Source Code:**

| #include<GLUT/glut.h> #include<iostream> #include<cmath> #include<string> using namespace std; const double PI = 3.14159265; void drawString(float x, float y, const char \*string){  glRasterPos2f(x, y);  for(const char\* c = string;\*c!='\0';c++)  glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12, \*c); } void myInit() {  glClearColor(1.0,0.6,0.5,0.0);  glPointSize(1);  glMatrixMode(GL\_PROJECTION);  glLoadIdentity();  gluOrtho2D(-320.0,320.0,-240.0,240.0); } void pltFig(float arr[][3],int n){  glBegin(GL\_POLYGON);  for(int i=0;i<n;i++){  glVertex2f(arr[i][0],arr[i][1]);  }  glEnd(); } void matMul(float a[][3], float b[][3],float mult[][3],int n){  for(int i = 0; i < n; ++i)  for(int j = 0; j < 3; ++j)  for(int k = 0; k < 3; ++k)  {  mult[i][j] += a[i][k] \* b[k][j];  } } void translation(float arr[][3],int n, float tx, float ty,float mul[][3]){  float b[3][3]={{1,0,0},{0,1,0},{tx,ty,1}};  matMul(arr, b, mul, n); } void rotation(float arr[][3],int n, int a,float mul[][3],int xr=0,int yr=0){  double angle = a \* PI / 180.0;  float b[3][3]= {{float(cos(angle)),float(sin(angle)),0},{float(-1\*sin(angle)),float(cos(angle)),0},{xr\*(1-float(cos(angle)))+yr\*float(sin(angle)),yr\*(1-float(cos(angle)))-xr\*float(sin(angle)),1}};  matMul(arr, b, mul, n); } void scaling(float arr[][3],int n, float sx, float sy,float mul[][3],int xf=0,int yf=0){  float b[3][3]= {{sx,0,0},{0,sy,0},{xf\*(1-sx),yf\*(1-sy),1}};  matMul(arr, b, mul, n); } void reflection(float arr[][3],int n, float mul[][3],int choice){  if(choice==1){  float b[3][3]={{1,0,0},{0,-1,0},{0,0,1}};  matMul(arr, b, mul, n);  }  else if(choice==2){  float b[3][3]={{-1,0,0},{0,1,0},{0,0,1}};  matMul(arr, b, mul, n);  }  else if(choice==3){  float b[3][3]={{-1,0,0},{0,-1,0},{0,0,1}};  matMul(arr, b, mul, n);  }  else if(choice==4){  float b[3][3]={{0,1,0},{1,0,0},{0,0,1}};  matMul(arr, b, mul, n);  } } void shearing(float arr[][3],int n, float s,float mul[][3],int choice){  if(choice==1){  float b[3][3]={{1,0,0},{s,1,0},{0,0,1}};  matMul(arr, b, mul, n);  }  else{  float b[3][3]={{1,s,0},{0,1,0},{0,0,1}};  matMul(arr, b, mul, n);  } } void myDisplay(){  glClear(GL\_COLOR\_BUFFER\_BIT);  glColor3f(0.0,0.0,0.0);  /\* **To** plot axes \*/  glBegin(GL\_LINES);  glVertex2f(-320.0,0.0);  glVertex2f(320.0,0.0);  glVertex2f(0.0,-240.0);  glVertex2f(0.0,240.0);  glEnd();    int n,ch;  float arr[10][3],mul[10][3]={0};  cout<<"Enter number of vertices: ";  cin>>n;  cout<<"Enter vertices (x,y): "<<endl;  for(int i=0;i<n;i++){  cin>>arr[i][0]>>arr[i][1];  arr[i][2]=1;  }    pltFig(arr, n);    cout<<"MENU\n1.Translation\n2.Rotation\n3.Scaling\n4.Reflection\n5.Shearing\n";  cout<<"Enter choice of transformation: ";  cin>>ch;  switch(ch){  case 1:  {  float tx,ty;  cout<<"Enter translation values: ";  cin>>tx>>ty;  translation(arr,n,tx,ty,mul);  glColor3f(1.0,1.0,1.0);  pltFig(mul, n);  break;  }  case 2:  {  int angle,c,xr,yr;  cout<<"Enter rotation degree: ";  cin>>angle;  cout<<"1. about origin\n2. about fixed point\nChoice = ";  cin>>c;  if(c==1) rotation(arr,n,angle,mul);  else{  cout<<"Enter fixed point: ";  cin>>xr>>yr;  rotation(arr, n, angle, mul,xr,yr);  }  for(int i=0;i<n;i++)  cout<<mul[i][0]<<","<<mul[i][1]<<endl;  glColor3f(1.0,1.0,1.0);  pltFig(mul, n);  break;  }  case 3:  {  float sx,sy;  int c,xr,yr;  cout<<"Enter Scaling values: ";  cin>>sx>>sy;  cout<<"1. about origin\n2. about fixed point\nChoice = ";  cin>>c;  if(c==1) scaling(arr,n,sx,sy,mul);  else{  cout<<"Enter fixed point: ";  cin>>xr>>yr;  scaling(arr,n,sx,sy,mul,xr,yr);  }  for(int i=0;i<n;i++)  cout<<mul[i][0]<<","<<mul[i][1]<<endl;  glColor3f(1.0,1.0,1.0);  pltFig(mul, n);  break;  }  case 4:  {  int c,xr,yr;  cout<<"1. x-axis\n2. y-axis\n3. origin\n4. the line x=y\nChoice = ";  cin>>c;  reflection(arr,n,mul,c);  for(int i=0;i<n;i++)  cout<<mul[i][0]<<","<<mul[i][1]<<endl;  glColor3f(1.0,1.0,1.0);  pltFig(mul, n);  break;  }  case 5:  {  float s;  int c;  cout<<"1. x-direction shear\n2. y-direction shear\nChoice = ";  cin>>c;  cout<<"Enter shear value: ";  cin>>s;  shearing(arr,n,s,mul,c);  for(int i=0;i<n;i++)  cout<<mul[i][0]<<","<<mul[i][1]<<endl;  glColor3f(1.0,1.0,1.0);  pltFig(mul, n);  break;  }  default:  cout<<"Invalid Option"<<endl;  }  glFlush(); } int main(int argc,char\* argv[]) {  glutInit(&argc,argv);  glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);  glutInitWindowSize(480,480);  glutCreateWindow("2D Transformations");  glutDisplayFunc(myDisplay);  myInit();  glutMainLoop();  return 1; } |
| --- |

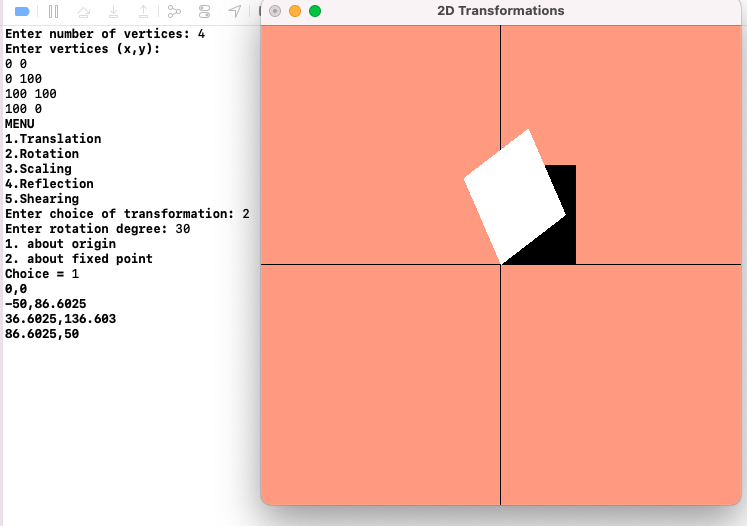
**Output:**

1) Translation

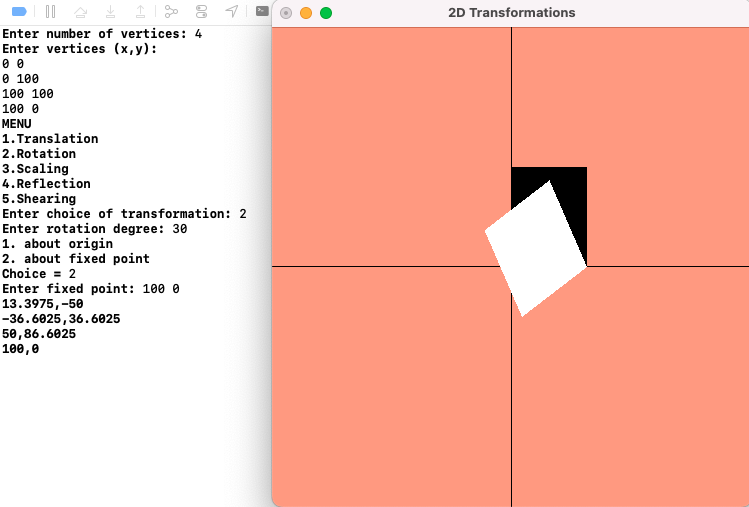


2) Rotation

a) about origin

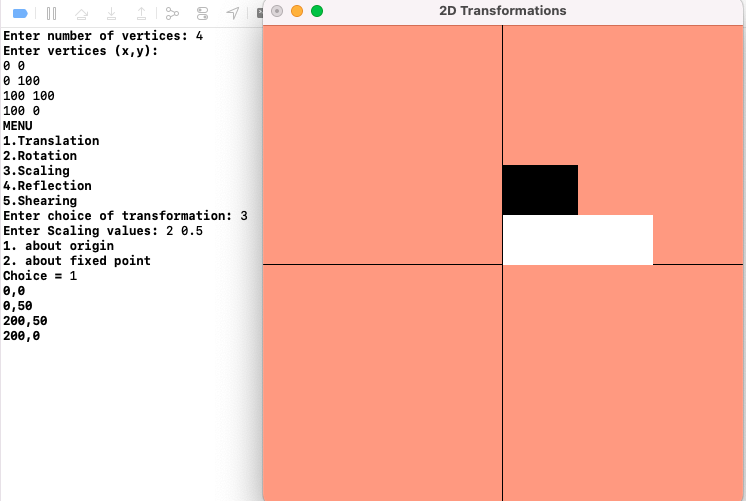


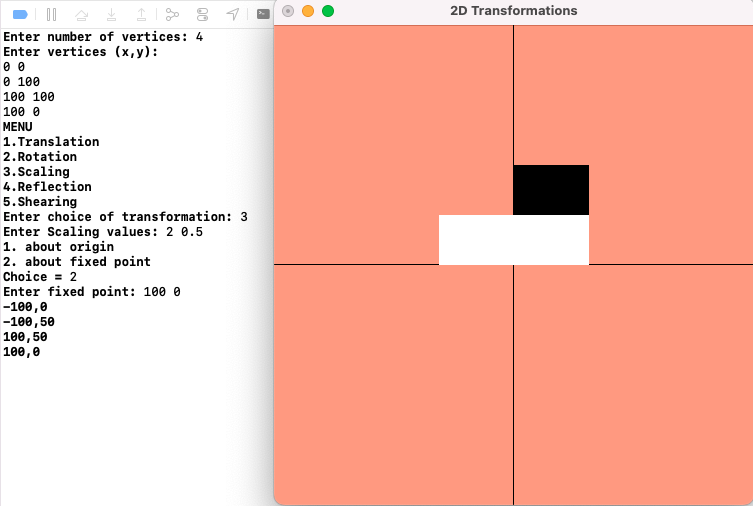
b) with respect to a fixed point (xr,yr)



3) Scaling with respect to

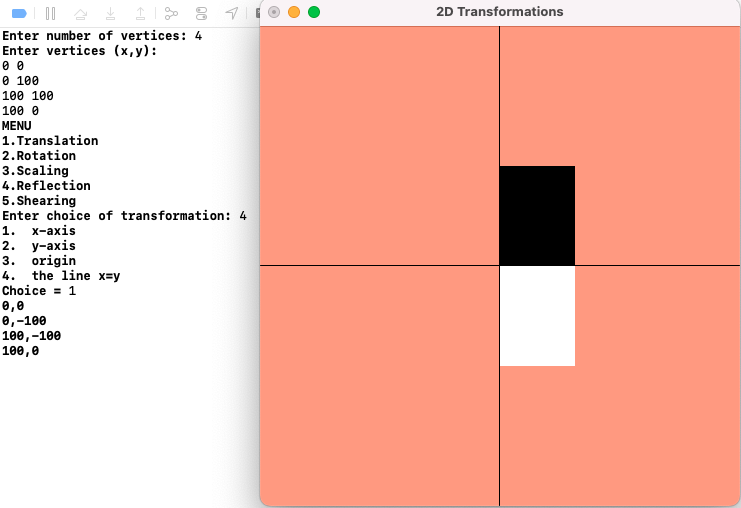
a) origin



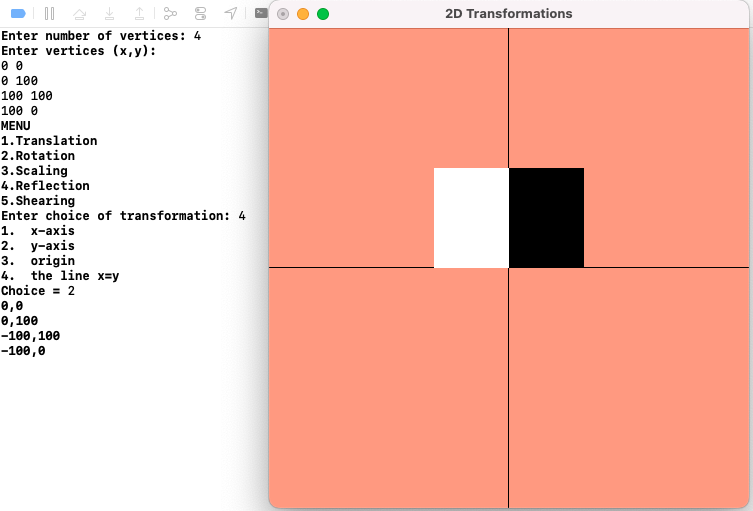
b) fixed point (xf,yf) 

4) Reflection with respect to

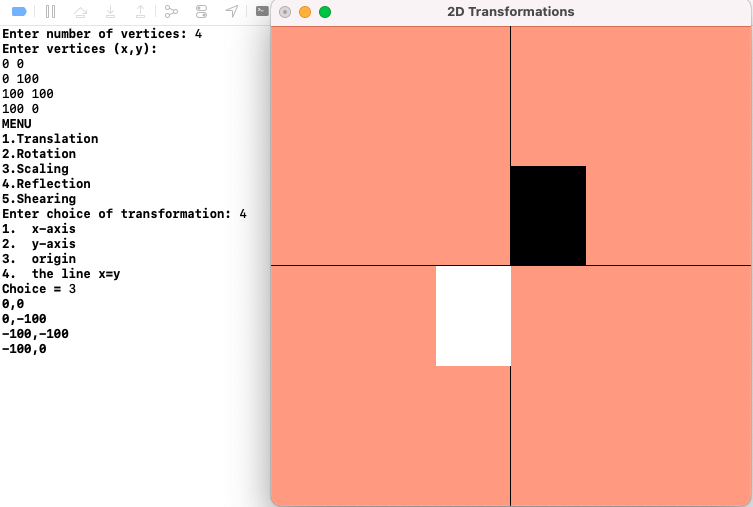
a) x-axis

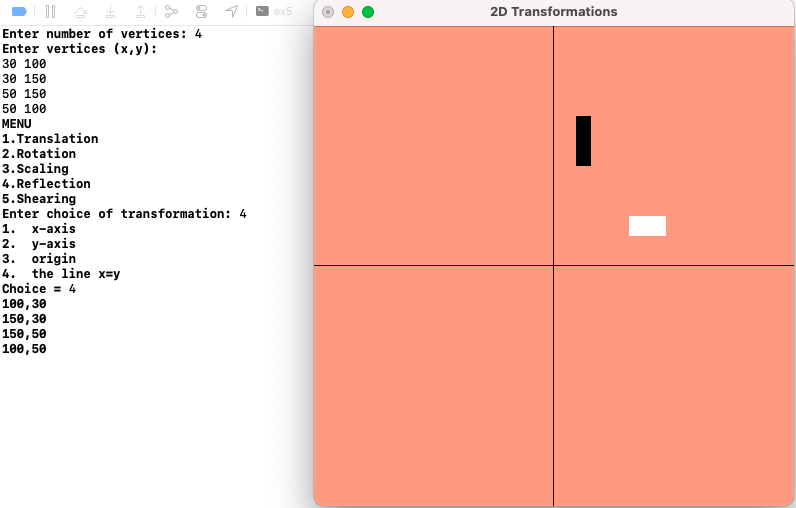


b) y-axis

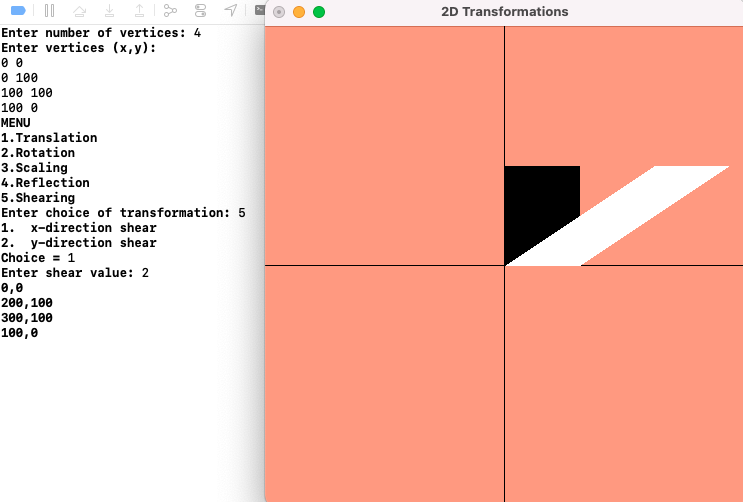


c) origin

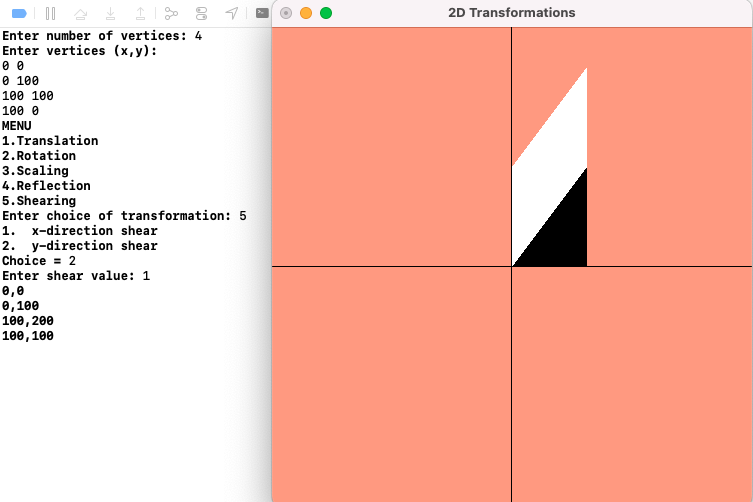


d) the line x=y

5) Shearing

a) x-direction shear

b) y-direction shear



**Learning Outcomes:**

Learnt to perform all 2D transformations in C++ using OpenGL