Assignment-04

2D transformations

```
#include<iostream>
#include<math.h>
#include<graphics.h>
using namespace std;
class matrix
{
public:
   int n,i,j,tx,ty,k,sum,sx,sy;
   double
a[6][3],b[6][3],mult[6][3],mat3[6][3];
   double p,q,r;
   double ang=0,angle=0;
public:
   void get()
      {
   cout<<"\n enter the number of vertices
of polygon : ";
      cin>>n;
```

```
     // cout<<"n Entering user matix\n";
      for(i=0;i<n;i++)
      {
       cout<<"enter x n y co ordinates";
          cin>>b[i][0];
          cin>>b[i][1];
          b[i][2]=1;
      }
//display object matrix
      cout<<"\n original  co ordinates
are"<<"\n";

      for(i=0;i<n;i++)
      {
          for(j=0;j<3;j++)
          {
              cout<<b[i][j]<<"\t";
          }cout<<"\n";
      }
      }
void identitymat()
{
```

```
for(i=0;i<n;i++)
       {
               for(j=0;j<3;j++)
               {
                   if(i==j)
                   {
                       a[i][j]=1;
                   }
                   else
                   {
                       a[i][j]=0;
                   }
               }
       }
}
   void trans()
   {

       cout<<"enter values of tx and ty";
       cin>>tx>>ty;
       a[2][0]=tx;
       a[2][1]=ty;
       cout<<"matrix is"<<"\n";
       for(i=0;i<n;i++)
       {
           for(j=0;j<3;j++)
           {
               cout<<a[i][j]<<"\t";
           }cout<<"\n";
       }
    }
   void scale()
   {
       cout<<"\n Enter the values of sx and
sy";
       cin>>sx>>sy;
       a[0][0]=sx;
       a[1][1]=sy;
   cout<<"\n Matrix is:"<<"\n";
//To display scaling matrix
       for(i=0;i<3;i++)
       {
               for(j=0;j<3;j++)
       {
                       cout<<a[i][j]<<"\t";
       }cout<<"\n";
       }
   }
   void rot()
      {
       cout<<"Enter the angle";
      cin>>ang;
      angle=(ang*3.142)/180;
      q=sin(angle);
      p=cos(angle);
      r=-sin(angle);
      a[0][0]=p;
      a[0][1]=q;
      a[1][0]=r;
      a[1][1]=p;
      cout<<"tranformationmatrix is"<<"\n";

      for(i=0;i<3;i++)
```

```
        {
            for(j=0;j<3;j++)
            {
                cout<<a[i][j]<<"\t";
            }cout<<"\n";
        }
        }
    void multi()
             {
                 cout<<"\nMultiplying two
matrices...";
                     for(i=0; i<n; i++)
                     {
                         for(j=0; j<3; j++)
                         {
                             sum=0;
                             for(k=0; k<3;
k++)
                             {
                                 sum = sum +
b[i][k] * a[k][j];
```

```
                            }
                            mat3[i][j] =
sum;
                        }
                    }
           }
           void display()
           {
                   cout<<"\nMultiplication
of two Matrices : \n";
                   for(i=0; i<n; i++)
                   {
                       for(j=0; j<3; j++)
                       {

cout<<mat3[i][j]<<" ";
                       }
                       cout<<"\n";
                   }

int gd=DETECT,gm;
initgraph(&gd,&gm,NULL);
```

```
for(int i=0;i<n-1;i++)
{

line(b[i][0],b[i][1],b[i+1][0],b[i+1][1]);
}
line(b[2][0],b[2][1],b[0][0],b[0][1]);
for(int i=0;i<n-1;i++)
{

line(mat3[i][0],mat3[i][1],mat3[i+1][0],mat3[i+1][1]);
}
line(mat3[2][0],mat3[2][1],mat3[0][0],mat3[0][1]);
delay(5000);
closegraph();
           }
   };
int main()
{
   matrix g;
   int ch;
   char ans;
   g.get();
   g.identitymat();
   do
   {
cout<<"menu\n1.translation\n2.scaling\n3.rotation";
   cin>>ch;
   switch(ch)
   {
   case 1:
       g.trans();
       g.multi();
       g.display();
       break;
   case 2:
       g.scale();
       g.multi();
       g.display();
       break;
   case 3:
       g.rot();
       g.multi();
       g.display();
       break;
   }cin>>ans;
   }while(ans=='Y'&& ans=='y');
return 0;
}
```

```
Enter coordinate x1 : 50
Enter coordinate y1 : 50
Enter coordinate x2 : 200
Enter coordinate y2 : 200
Menu :
1. Translate
2. Scale
3. Rotate
4. Exit
Enter your choice : 1
Enter coordinate for point x : 30
Enter coordinate for point y : 40
Do you want to continue? (y/n): y
Menu :
1. Translate
2. Scale
3. Rotate
4. Exit
Enter your choice : 2
Enter coordinate for point x : 2
Enter coordinate for point y : 2
Do you want to continue? (y/n): y
Menu :
1. Translate
2. Scale
3. Rotate
4. Exit
Enter your choice : 3
Enter the angle to rotate the line : 45
Do you want to continue? (y/n): n
```

Explanation of the Code

This C++ program demonstrates 2D transformations (translation, scaling, and rotation) of a polygon using matrix operations and the graphics.h library for graphical output. It allows the user to input the vertices of a polygon and apply different transformations to it.

Key Components of the Code:

1.Matrix Class:

•The class matrix encapsulates methods for handling 2D transformations using matrices.

•It includes methods for inputting vertices, creating transformation matrices, performing matrix multiplication, and displaying results.

2.Attributes:

•b[][]: A 2D array to store the original vertices of the polygon.

- a[][]: A 2D array to hold the transformation matrix.
- mat3[][]: A 2D array to store the result after transformations.

3.Methods:

- get(): Prompts the user to enter the number of vertices and their coordinates.
- identitymat(): Initializes an identity matrix for transformations.
- trans(): Constructs the translation matrix using translation values tx and ty.
- scale(): Constructs the scaling matrix using scaling factors sx and sy.
- rot(): Constructs the rotation matrix using an angle of rotation.
- multi(): Multiplies the transformation matrix by the original vertex matrix to compute the transformed coordinates.
- display(): Displays the transformed coordinates and draws the original and

transformed polygons using the graphics.h functions.

4.Graphics Functions:

- The program uses the graphics.h library to draw lines between the vertices of the original polygon and the transformed polygon.

5.Main Function:

- The program prompts the user to select a transformation type (translation, scaling, or rotation), applies the transformation, and displays the result.