Name : V Vikram **Class : CSE 'C'**

Reg. No. : 18 5001 194 **Date : 10/10/2021**

Subject : UCS1712---Graphics and Multimedia Lab

Source.cpp :

```
#include<GL/glut.h>
#include<stdlib.h>
#include<iostream>
#include<vector>
#include<cmath>
#include<iostream>
using namespace std;

int pntX1=100,pntY1=100,r=20;

vector<vector<double>> Trans_1(3,vector<double>(3,0));
vector<vector<double>> Trans_2(3,vector<double>(3,0));

#include "Header.h"

vector<pair<int,int>> line_pair;


//vector<vector<int>> triangle(3,vector<int>(2,0));

//triangle[0][0] = 100;
```

```
//triangle[0][1] = 110;


//triangle[1][0] = 110;
//triangle[1][1] = 90;


//triangle[2][0] = 90;
//triangle[2][1] = 90;




void myInit()
{
	glClearColor(1.0, 1.0, 1.0, 0.0 );
	glPointSize(20.0);

	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0.0,1000.0,0.0,500);


}
void myDisplay()
{
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(255.0f / 255.0f, 0.0f / 255.0f, 0.0f / 255.0f);

	draw_Floor();
	glBegin(GL_POINTS);

	glEnd();
```

```
	glColor3f(255.0f / 255.0f, 0.0f / 255.0f, 0.0f / 255.0f);
	Trans_1 = translate();
	Trans_2 = rotate();
	for(int i=0;i<3;i++){

		//draw_Triangle();
		line_pair = Transform_Polygon(Trans_1,Trans_2);
		midPointCircleAlgo();
	}
	pntX1 = line_pair[0].first;
	pntY1 = line_pair[0].second;
	glFlush();
}
int main(int argc, char* argv[]) {
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(1000, 500);
	glutCreateWindow("Sample");
	glutDisplayFunc(myDisplay);
	myInit();
	glutMainLoop();
	return 1;
}
```

Header.h:

```
#pragma once


void draw_Floor()
{
	//glPointSize(20.0);
	glColor3f(0.0,0.0,0.0);
	glBegin(GL_LINES);
	glVertex2f(0.0,50.0);
	glVertex2d(1000.0,50.0);
	glEnd();
}
void plot(int x, int y)
{
	glBegin(GL_POINTS);
	glVertex2i(x + pntX1, y + pntY1);
	glEnd();
}
vector<vector<double>> translate()
{
	//vector<vector<int>> Trans_1(3,vector<int>(3,0));
	Trans_1[0][2] = 100;
	Trans_1[1][2] = 100;
	Trans_1[2][2] = 1;
```

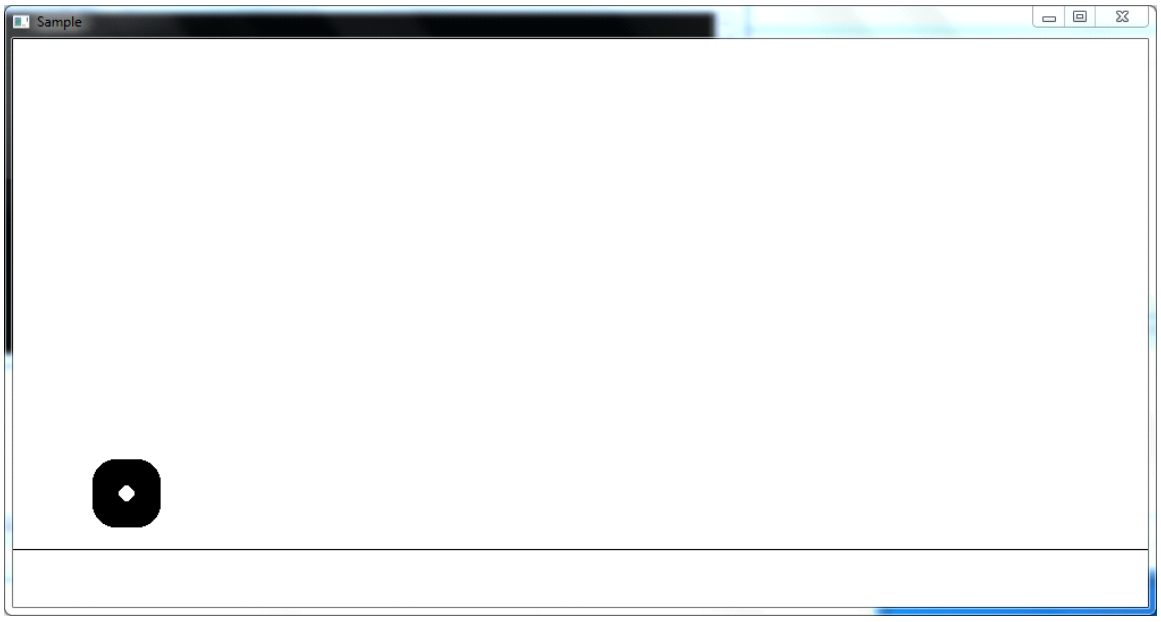
```
	cout << "\nTranslate\n";
	for (int i = 0; i < 3; i++) {
		for (int j = 0; j < 3; j++) {
			cout << Trans_1[i][j] << " ";
		}
		cout << "\n";
	}
	return Trans_1;
}

vector<vector<double>> rotate()
{
	//vector<vector<double>> Trans_2(3,vector<double>(3,0));
	double ang_rad = 45 * 3.14/180;
	Trans_2[0][0] = cos(ang_rad);
	Trans_2[1][1] = cos(ang_rad);
	Trans_2[0][1] = sin(ang_rad);
	Trans_2[1][0] = -1*sin(ang_rad);
	Trans_2[2][2] = 1;

		cout << "\nROtate\n";
	for (int i = 0; i < 3; i++) {
		for (int j = 0; j < 3; j++) {
			cout << Trans_2[i][j] << " ";
		}
		cout << "\n";
	}
	return Trans_2;
}
```

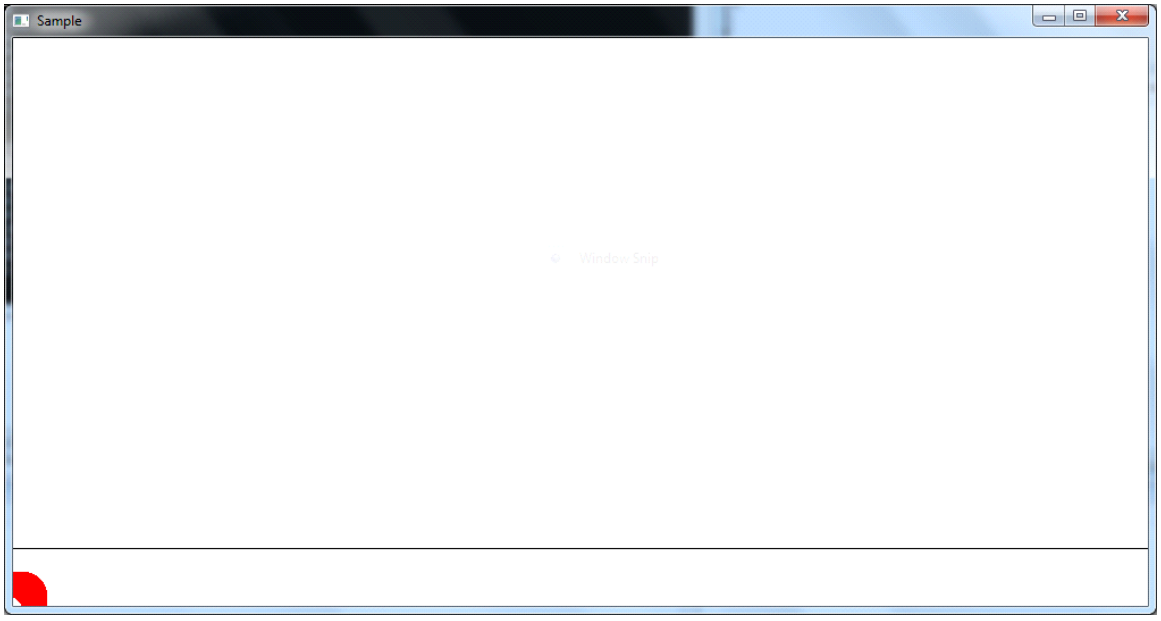
```
void draw_Triangle()
{
	glColor3f(0.0,0.0,0.0);
	glBegin(GL_LINE_LOOP);
	for(int i=0;i<3;i++){
		//glVertex2d(triangle[i][0],triangle[i][1]);
	}
	glEnd();
}
void midPointCircleAlgo()
{
	int x = 0;
	int y = r;
	int decision = 1 - r;
	plot(x, y);
	while (y >= x){
		if (decision < 0){
			x++;
			decision += 2 * x + 1;
		}
		else{
			y--;
			x++;
			decision += 2 * (x - y) + 1;
		}
		plot(x, y);
		plot(x, -y);
		plot(-x, y);
		plot(-x, -y);
```

```
            plot(y, x);
            plot(-y, x);
            plot(y, -x);
            plot(-y, -x);
      }
}

vector<pair<int,int>> Transform_Polygon(vector<vector<double>>
Trans_1,vector<vector<double>> Trans_2) {
      /*Multiply the two transformation matrices to find the
      final TRANSFORMATION matrix*/
      vector<vector<double>> Transformation(3,vector<double>(3,0));
cout << "\nTransforme\n";
      for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                  for (int k = 0; k < 3; k++) {
                        Transformation[i][j] += Trans_1[i][k] *Trans_2[k][j];
                        cout << Transformation[i][j] << " ";
                  }
                  cout << "\n";
            }
      }
      cout << "\n";

      vector<double> curpoint(3, 0), matProduct(3, 0);
      //for (int i = 0; i < 3; i++) {
            curpoint[0] = pntX1;
            curpoint[1] = pntY1;
            curpoint[2] = 1;
```

```
            for (int j = 0; j < 3; j++) {
                  matProduct[0] = 0;
                  matProduct[1] = 0;
                  matProduct[2] = 0;
                  for (int k = 0; k < 3; k++) {
                        //Apply the TRANSFORMATION matrix to all the vertices
                        matProduct[j] += Transformation[j][k] *curpoint[k];
                  }
            }

      //}
      pntX1 = abs(matProduct[0]);
      pntY1 = abs(matProduct[1]);
      vector<pair<int,int>> line_pair;
      line_pair.push_back(make_pair(abs(matProduct[0]),abs(matProduct[1])));
      return line_pair;
}
```

OUTPUT SNAP SHOTS :


Sample


Sample