UCS1712 - GRAPHICS AND MULTIMEDIA LAB

EX - 5A: TRANSLATION, ROTATION, AND SCALING

NAME: KEERTHANA T
REGISTER NUMBER: 185001074
CLASS-SEC: CSE-B
DATE: 22/08/2021

AIM :

To write a C++ menu-driven program using OpenGL to perform 2D transformations – translation, rotation, scaling for line and polygon.

ALGORITHM :

- Start
- Import GL library as a header file
- Create a function void drawPolygon() to draw a polygon for the given vertices.
- Create a function void drawPolygonTrans() to draw the translated polygon for the given translation factors.
- Create a function void drawPolygonScale() to draw the scaled polygon for the given scaling factors.
- Create a function void drawPolygonRotation() to draw the rotated polygon for the given angle for rotation.
- Create a function void myInit() for the initial configuration of the display window
- Create a function void myDisplay()
- Now create a menu-driven program for choosing the kind of operation to be done for the given polygon.

- If the choice is 1, assign translation; if the choice is two, assign scaling, if the choice is 3, assign rotation.
- Create the void main() function
- Get parameters for the initial polygon.
- Implement the menu-driven program and get the input choice for the operation to be executed.
- After choosing the operation take in the input parameters for the operation to be executed.
- End

CODE:

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

int pntX1, pntY1, choice = 0, edges;
vector<int> pntX;
vector<int> pntY;
int transX, transY;
double scaleX, scaleY;
double angle, angleRad;


double round(double d)
{
	return floor(d + 0.5);
}

void drawPolygon()
{
	glBegin(GL_POLYGON);
```

```
	glColor3f(1.0, 0.0, 0.0);
	for (int i = 0; i < edges; i++)
	{
		glVertex2i(pntX[i], pntY[i]);
	}
	glEnd();
}


void drawPolygonTrans(int x, int y)
{
	glBegin(GL_POLYGON);
	glColor3f(0.0, 1.0, 0.0);
	for (int i = 0; i < edges; i++)
	{
		glVertex2i(pntX[i] + x, pntY[i] + y);
	}
	glEnd();
}

void drawPolygonScale(double x, double y)
{
	glBegin(GL_POLYGON);
	glColor3f(0.0, 0.0, 1.0);
	for (int i = 0; i < edges; i++)
	{
		glVertex2i(round(pntX[i] * x), round(pntY[i] * y));
	}
	glEnd();
}

void drawPolygonRotation(double angleRad)
{
	glBegin(GL_POLYGON);
	glColor3f(0.0, 0.0, 1.0);
```

```
	for (int i = 0; i < edges; i++)
	{
		glVertex2i(round((pntX[i] * cos(angleRad)) - (pntY[i] *
sin(angleRad))), round((pntX[i] * sin(angleRad)) + (pntY[i] * cos(angleRad))));
	}
	glEnd();
}
/*void drawPolygonMirrorReflection(char reflectionAxis)
{
	glBegin(GL_POLYGON);
	glColor3f(0.0, 0.0, 1.0);

	if (reflectionAxis == 'x' || reflectionAxis == 'X')
	{
		for (int i = 0; i < edges; i++)
		{
			glVertex2i(round(pntX[i]), round(pntY[i] * -1));

		}
	}
	else if (reflectionAxis == 'y' || reflectionAxis == 'Y')
	{
		for (int i = 0; i < edges; i++)
		{
			glVertex2i(round(pntX[i] * -1), round(pntY[i]));
		}
	}
	glEnd();
}

void drawPolygonShearing()
{
	glBegin(GL_POLYGON);
	glColor3f(0.0, 0.0, 1.0);
```

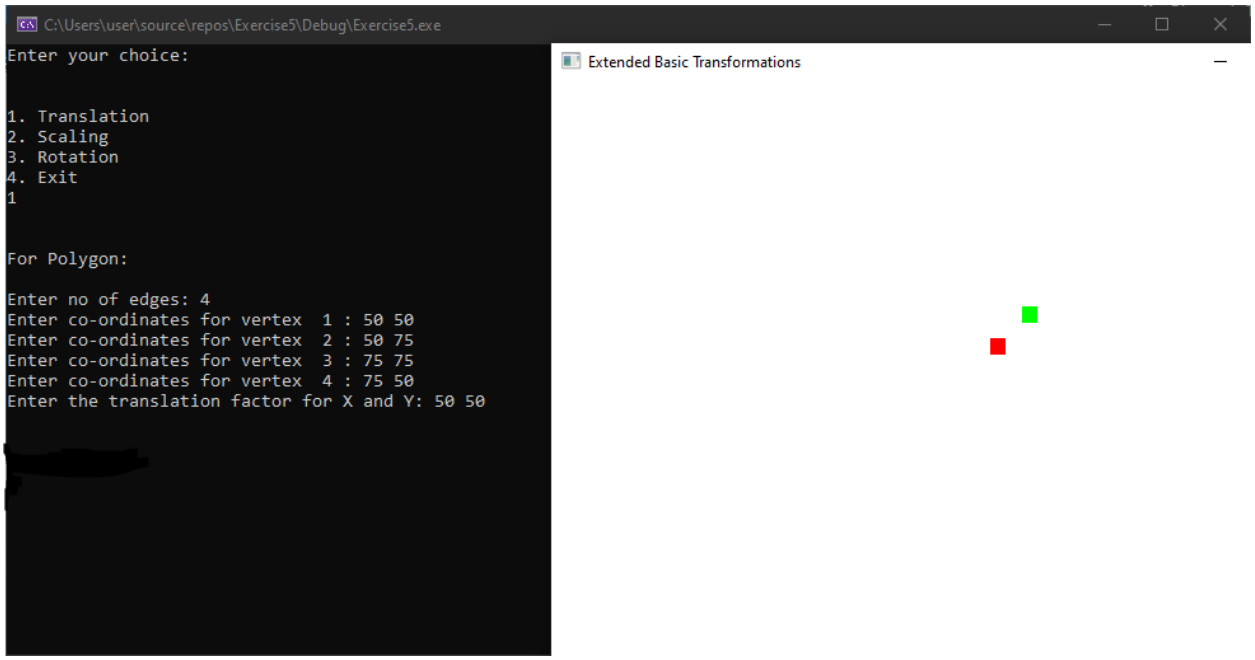
```
	if (shearingAxis == 'x' || shearingAxis == 'X')
	{
		glVertex2i(pntX[0], pntY[0]);

		glVertex2i(pntX[1] + shearingX, pntY[1]);
		glVertex2i(pntX[2] + shearingX, pntY[2]);

		glVertex2i(pntX[3], pntY[3]);
	}
	else if (shearingAxis == 'y' || shearingAxis == 'Y')
	{
		glVertex2i(pntX[0], pntY[0]);
		glVertex2i(pntX[1], pntY[1]);

		glVertex2i(pntX[2], pntY[2] + shearingY);
		glVertex2i(pntX[3], pntY[3] + shearingY);
	}
	glEnd();
}*/


void myInit(void)
{
	glClearColor(1.0, 1.0, 1.0, 0.0);
	glColor3f(0.0f, 0.0f, 0.0f);
	glPointSize(1.0);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(-640.0, 640.0, -480.0, 480.0);
}


void myDisplay(void)
{
	glClear(GL_COLOR_BUFFER_BIT);
```

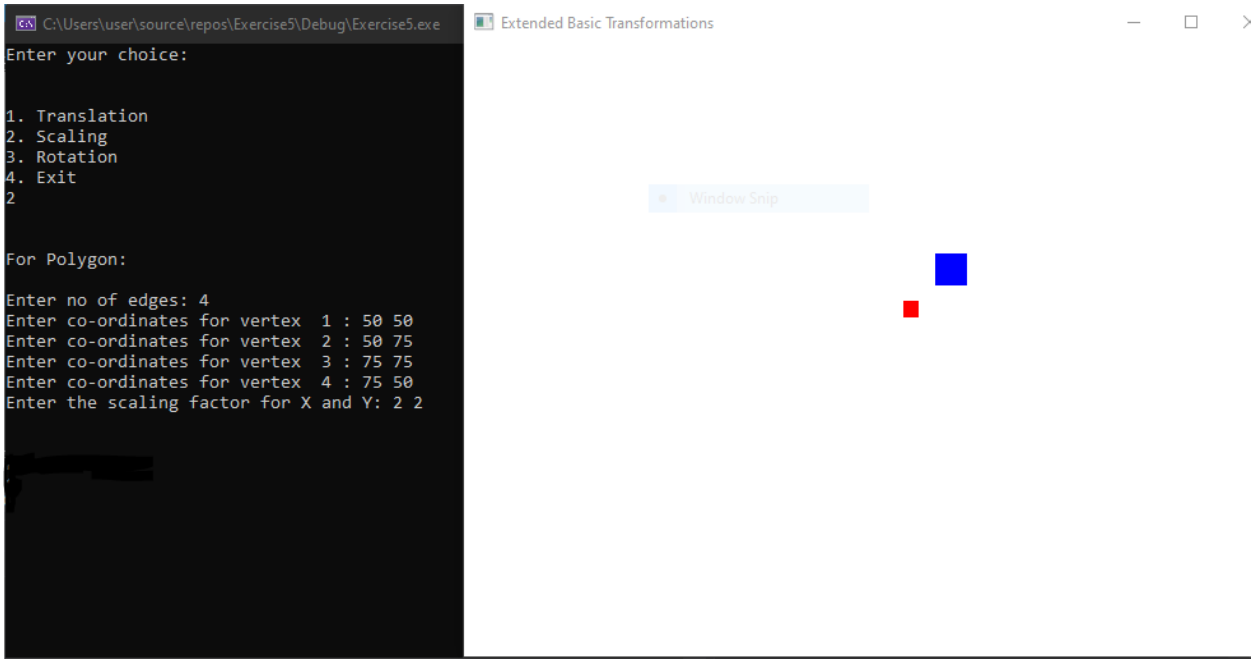
```
	glColor3f(0.0, 0.0, 0.0);

	if (choice == 1)
	{
		drawPolygon();
		drawPolygonTrans(transX, transY);
	}
	else if (choice == 2)
	{
		drawPolygon();
		drawPolygonScale(scaleX, scaleY);
	}
	else if (choice == 3)
	{
		drawPolygon();
		drawPolygonRotation(angleRad);
	}
	/*else if (choice == 4)
	{
		drawPolygon();
		drawPolygonMirrorReflection(reflectionAxis);
	}
	else if (choice == 5)
	{
		drawPolygon();
		drawPolygonShearing();
	}
	*/
	glFlush();
}

void main(int argc, char** argv)
{
	cout << "Enter your choice:\n\n" << endl;
```

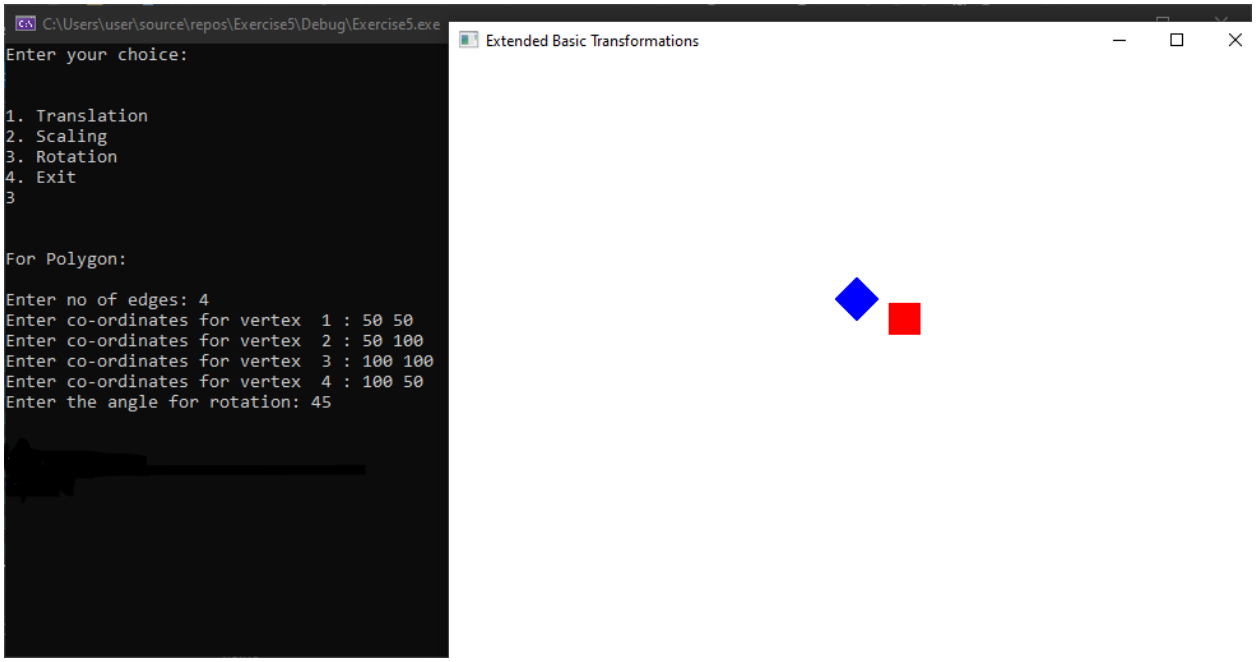
```
	cout << "1. Translation" << endl;
	cout << "2. Scaling" << endl;
	cout << "3. Rotation" << endl;
	//cout << "4. Mirror Reflection" << endl;
	//cout << "5. Shearing" << endl;
	cout << "4. Exit" << endl;

	cin >> choice;

	if (choice == 4) {
		return;
	}

	cout << "\n\nFor Polygon:\n" << endl;

	cout << "Enter no of edges: "; cin >> edges;

	for (int i = 0; i < edges; i++)
	{
		cout << "Enter co-ordinates for vertex  " << i + 1 << " : "; cin >>
pntX1 >> pntY1;
		pntX.push_back(pntX1);
		pntY.push_back(pntY1);
	}

	if (choice == 1)
	{
		cout << "Enter the translation factor for X and Y: "; cin >> transX >>
transY;
	}
	else if (choice == 2)
	{
		cout << "Enter the scaling factor for X and Y: "; cin >> scaleX >>
scaleY;
	}
```

```
	else if (choice == 3)
	{
		cout << "Enter the angle for rotation: "; cin >> angle;
		angleRad = angle * 3.1416 / 180;
	}
	/*else if (choice == 4)
	{
		cout << "Enter reflection axis ( x or y ): "; cin >> reflectionAxis;
	}
	else if (choice == 5)
	{
		cout << "Enter reflection axis ( x or y ): "; cin >> shearingAxis;
		if (shearingAxis == 'x' || shearingAxis == 'X')
		{
			cout << "Enter the shearing factor for X: "; cin >> shearingX;
		}
		else
		{
			cout << "Enter the shearing factor for Y: "; cin >> shearingY;
		}
	}*/
	cout << "\n\nPoints:" << pntX[0] << ", " << pntY[0] << endl;
	cout << angleRad;

	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(640, 480);
	glutInitWindowPosition(100, 150);
	glutCreateWindow("Extended Basic Transformations");
	glutDisplayFunc(myDisplay);
	myInit();
	glutMainLoop();

}
```

OUTPUT:


C:\Users\user\source\repos\Exercise5\Debug\Exercise5.exe
Enter your choice:
1. Translation
2. Scaling
3. Rotation
4. Exit
1
For Polygon:
Enter no of edges: 4
Enter co-ordinates for vertex 1 : 50 50
Enter co-ordinates for vertex 2 : 50 75
Enter co-ordinates for vertex 3 : 75 75
Enter co-ordinates for vertex 4 : 75 50
Enter the translation factor for X and Y: 50 50
Extended Basic Transformations


C:\Users\user\source\repos\Exercise5\Debug\Exercise5.exe
Enter your choice:
1. Translation
2. Scaling
3. Rotation
4. Exit
2
For Polygon:
Enter no of edges: 4
Enter co-ordinates for vertex 1 : 50 50
Enter co-ordinates for vertex 2 : 50 75
Enter co-ordinates for vertex 3 : 75 75
Enter co-ordinates for vertex 4 : 75 50
Enter the scaling factor for X and Y: 2 2
Extended Basic Transformations



RESULT:

Thus we have successfully implemented the basic transformation such as Translation, Rotation, and scaling for the given polygon.