|  |  |
| --- | --- |
| **Assignment** | |
| **Course Code** | CSC309A |
| **Course Name** | Computer Graphics |
| **Programme** | B.Tech |
| **Department** | CSE |
| **Faculty** | FET |

|  |  |
| --- | --- |
| **Name of the Student** | Satyajit Ghana |
| **Reg. No.** | 17ETCS002159 |
| **Semester/Year** | 06/2020 |
| **Course Leader(s)** | Dr. Subarna Chatterjee |



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Declaration Sheet | | | | | | | | |
| Student Name | Satyajit Ghana | | | | | | | |
| Reg. No | 17ETCS002159 | | | | | | | |
| Programme | B.Tech | | | | | Semester/Year | 06/2020 | |
| Course Code | CSC309A | | | | | | | |
| Course Title | Computer Graphics | | | | | | | |
| Course Date |  | | to |  | | | | |
| Course Leader | Dr. Subarna Chatterjee | | | | | | | |
| **Declaration**  The assignment submitted herewith is a result of my own investigations and that I have conformed to the guidelines against plagiarism as laid out in the Student Handbook. All sections of the text and results, which have been obtained from other sources, are fully referenced. I understand that cheating and plagiarism constitute a breach of University regulations and will be dealt with accordingly. | | | | | | | | |
| Signature of the Student | |  | | | | | Date |  |
| Submission date stamp  (by Examination & Assessment Section) | |  | | | | | | |
| Signature of the Course Leader and date | | | | | Signature of the Reviewer and date | | | |
|  | | | | |  | | | |

# Contents

[Declaration Sheet ii](#_Toc37993809)

[Contents iii](#_Toc37993810)

[List of Figures iv](#_Toc37993811)

[1 Question 1 5](#_Toc37993812)

[1.1 Introduction 5](#_Toc37993813)

[1.2 Implementation of Transformation 5](#_Toc37993814)

[1.3 Results with Screenshot and Discussion 9](#_Toc37993815)

[Bibliography 16](#_Toc37993816)

# List of Figures

[Figure 1 Original Polygon 10](#_Toc37993804)

[Figure 2 Rotated Polygon 11](#_Toc37993805)

[Figure 3 Translated Polygon 12](#_Toc37993806)

[Figure 4 Scaled Polygon 13](#_Toc37993807)

[Figure 5 Final Transformed Polygon 14](#_Toc37993808)

# Question 1

Solution to Question No. 1

## Introduction

## Implementation of Transformation

**main.cpp**

#include <iostream>

#include <string>

#include <GL/freeglut.h>

#define WIN\_WIDTH 1000

#define WIN\_HEIGHT 1000

#define GRID\_X\_MIN -8

#define GRID\_X\_MAX 60

#define GRID\_Y\_MIN -8

#define GRID\_Y\_MAX 60

#define WIN\_TITLE "Graphics Assignement"

int state = 0;

const unsigned char state\_title[][20] = { "ORIGINAL", "ROTATED", "TRANSLATED", "SCALED", "FINAL TRANSFORMED" };

int total\_states = 5;

void render();

void reshape(int, int);

void keyboard(unsigned char, int, int);

void special\_keys(int key, int x, int y);

void init();

int main(int argc, char\* argv[]) {

    glutInit(&argc, argv);

    // initialize the display mode

    glutInitDisplayMode(GLUT\_DEPTH | GLUT\_DOUBLE | GLUT\_RGBA | GLUT\_MULTISAMPLE);

    // for anti-aliasing

    glEnable(GLUT\_MULTISAMPLE);

    // set the window position

    glutInitWindowPosition(100, 100);

    // set the window size

    glutInitWindowSize(WIN\_WIDTH, WIN\_HEIGHT);

    // now create the window

    glutCreateWindow(WIN\_TITLE);

    // register the core functions

    glutDisplayFunc(render);

    glutReshapeFunc(reshape);

    glutKeyboardFunc(keyboard);

    glutSpecialFunc(special\_keys);

    init();

    glutMainLoop();

    return 0;

}

void init() {

    glClearColor(164 / 255.0, 176 / 255.0, 190 / 255.0, 0.8);

    glFlush();

}

void reshape(int w, int h) {

    glMatrixMode(GL\_MODELVIEW);

    glLoadIdentity();

    gluOrtho2D(GRID\_X\_MIN, GRID\_X\_MAX, GRID\_Y\_MIN, GRID\_Y\_MAX);

}

void show\_grid() {

    glBegin(GL\_LINES);

    const GLubyte prestigeBlue[] = { 47.0, 53.0, 66.0, 1.0 };

    const GLubyte pureRed[] = { 255.0, 0.0, 0.0, 1.0 };

    for (float i = GRID\_X\_MIN; i <= GRID\_X\_MAX; i += 1.0) {

        glColor4ubv(prestigeBlue);

        if (i == 0)

            glColor4ubv(pureRed);

        glVertex2f(i, GRID\_Y\_MIN);

        glVertex2f(i, GRID\_Y\_MAX);

    }

    for (float i = GRID\_Y\_MIN; i <= GRID\_Y\_MAX; i += 1.0) {

        glColor4ubv(prestigeBlue);

        if (i == 0)

            glColor4ubv(pureRed);

        glVertex2f(GRID\_X\_MIN, i);

        glVertex2f(GRID\_X\_MAX, i);

    }

    glEnd();

}

void polygon() {

    glBegin(GL\_POLYGON);

    const GLubyte jalapenoRed[] = { 183.0, 21.0, 64.0, 1.0 };

    const GLubyte darkSapphire[] = { 12.0, 36.0, 97.0, 1.0 };

    const GLubyte forestBlues[] = { 10.0, 61.0, 98.0, 1.0 };

    const GLubyte reefEncounter[] = { 7.0, 153.0, 146.0, 1.0 };

    glColor4ubv(jalapenoRed);

    glVertex2f(-1.0, 1.0);

    glColor4ubv(darkSapphire);

    glVertex2f(-1.0, -1.0);

    glColor4ubv(forestBlues);

    glVertex2f(1.0, -1.0);

    glColor4ubv(reefEncounter);

    glVertex2f(1.0, 1.0);

    glEnd();

}

void show\_original() {

    glPushMatrix();

    polygon();

    glPopMatrix();

}

void show\_transformed() {

    glPushMatrix();

    glScalef(5.0, 6.0, 1.0);

    glTranslatef(4.0, 8.0, 0.0);

    glRotatef(90, 0.0, 0.0, 1.0);

    polygon();

    glPopMatrix();

}

void show\_text() {

    glPushMatrix();

    glColor3f(0.0, 0.0, 0.0);

    glTranslatef(30.0, 30.0, 1.0);

    glRasterPos2f(0.0, 0.0);

    const unsigned char \*string = state\_title[state];

    glutBitmapString(GLUT\_BITMAP\_HELVETICA\_18, string);

    glPopMatrix();

}

void render() {

    glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

    show\_text();

    show\_grid();

    switch (state) {

        //ORIGINAL

    case 0:

        glPushMatrix();

        polygon();

        glPopMatrix();

        break;

        // ROTATED

    case 1:

        glPushMatrix();

        glRotatef(90, 0.0, 0.0, 1.0);

        polygon();

        glPopMatrix();

        break;

        // TRANSLATED

    case 2:

        glPushMatrix();

        glTranslatef(4.0, 8.0, 0.0);

        polygon();

        glPopMatrix();

        break;

        // SCALED

    case 3:

        glPushMatrix();

        glScalef(5.0, 6.0, 1.0);

        polygon();

        glPopMatrix();

        break;

        // FINAL TRANSFORMED

    case 4:

        show\_grid();

        show\_original();

        show\_transformed();

        break;

    }

    glFlush();

    glutSwapBuffers();

}

void keyboard(unsigned char c, int x, int y) {

    switch (c) {

    case 13:

    case 'q':

    case 'Q':

        exit(EXIT\_SUCCESS);

        break;

    }

}

void special\_keys(int key, int x, int y) {

    switch (key) {

    case GLUT\_KEY\_RIGHT:

        state = ((state + 1) % total\_states + total\_states) % total\_states;

        glutPostRedisplay();

        break;

    case GLUT\_KEY\_LEFT:

        state = ((state - 1) % total\_states + total\_states) % total\_states;

        glutPostRedisplay();

        break;

    }

}

## Results with Screenshot and Discussion

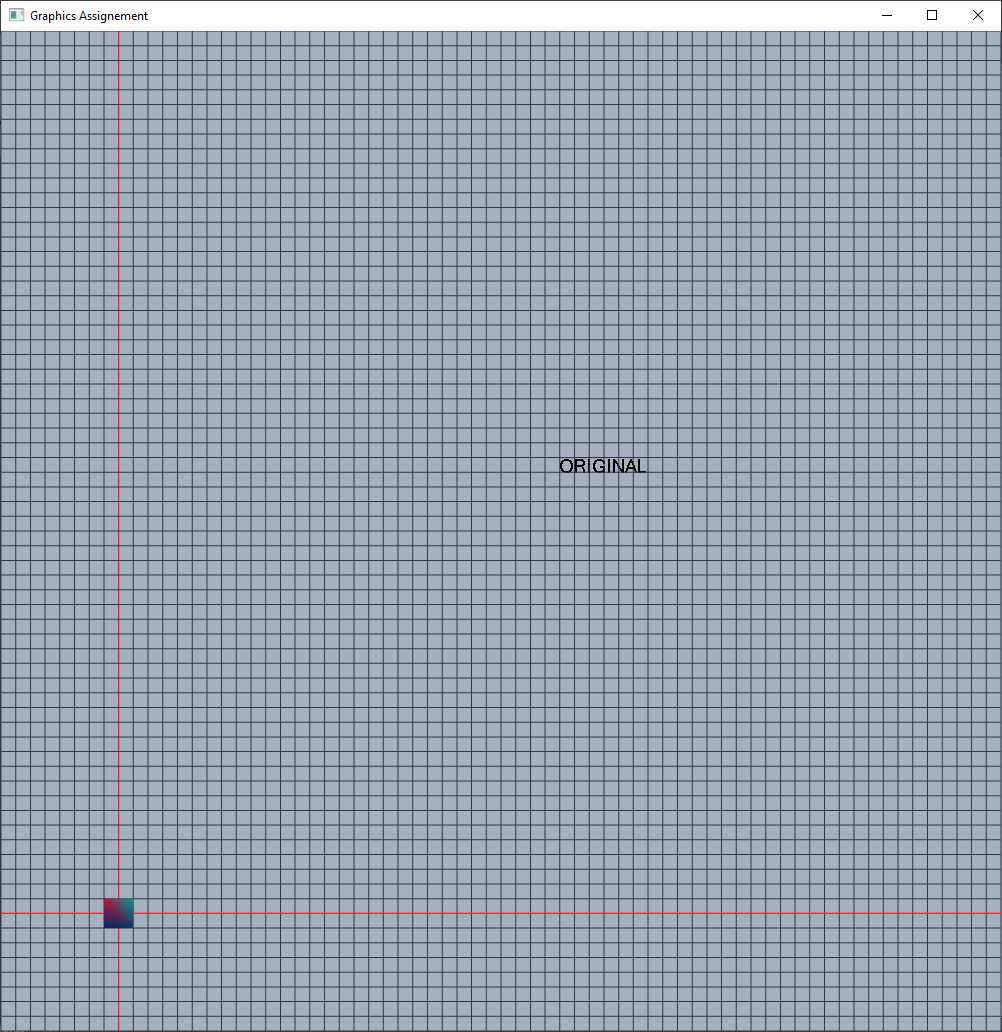
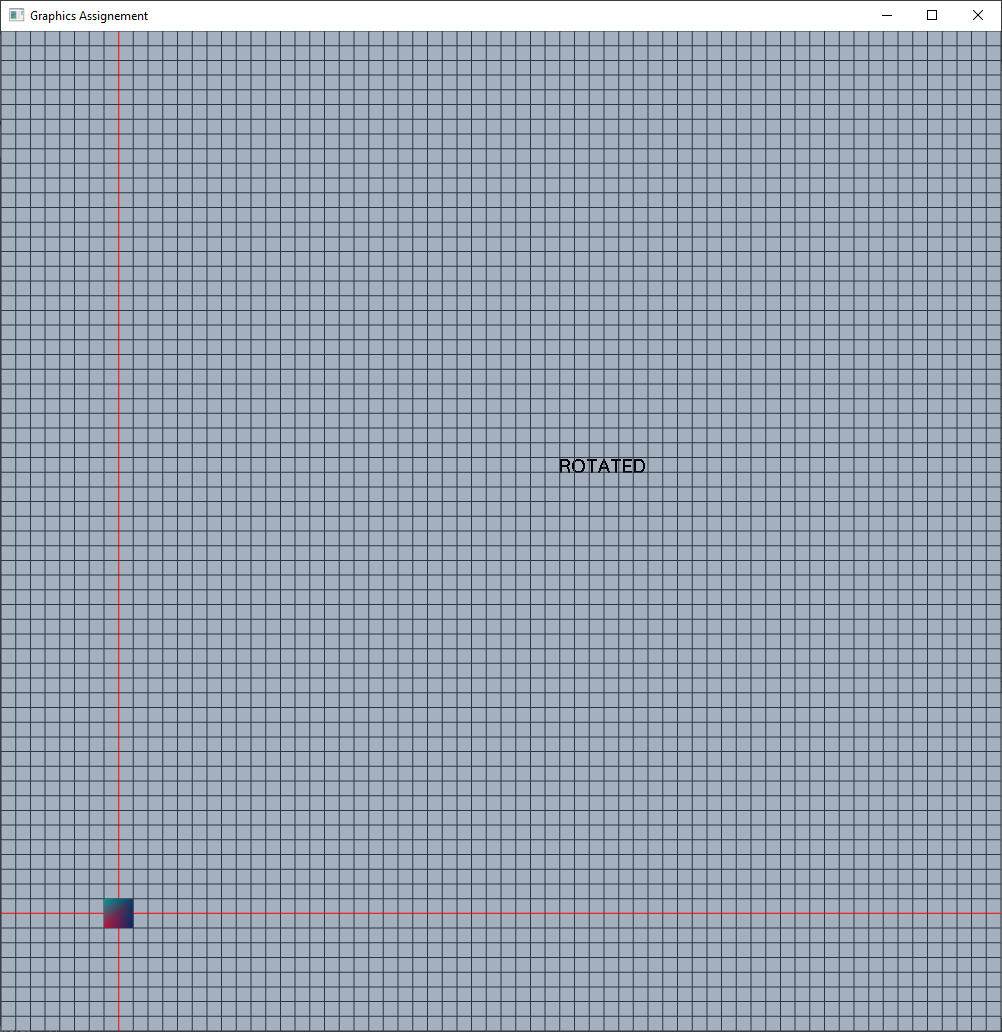


Figure Original Polygon



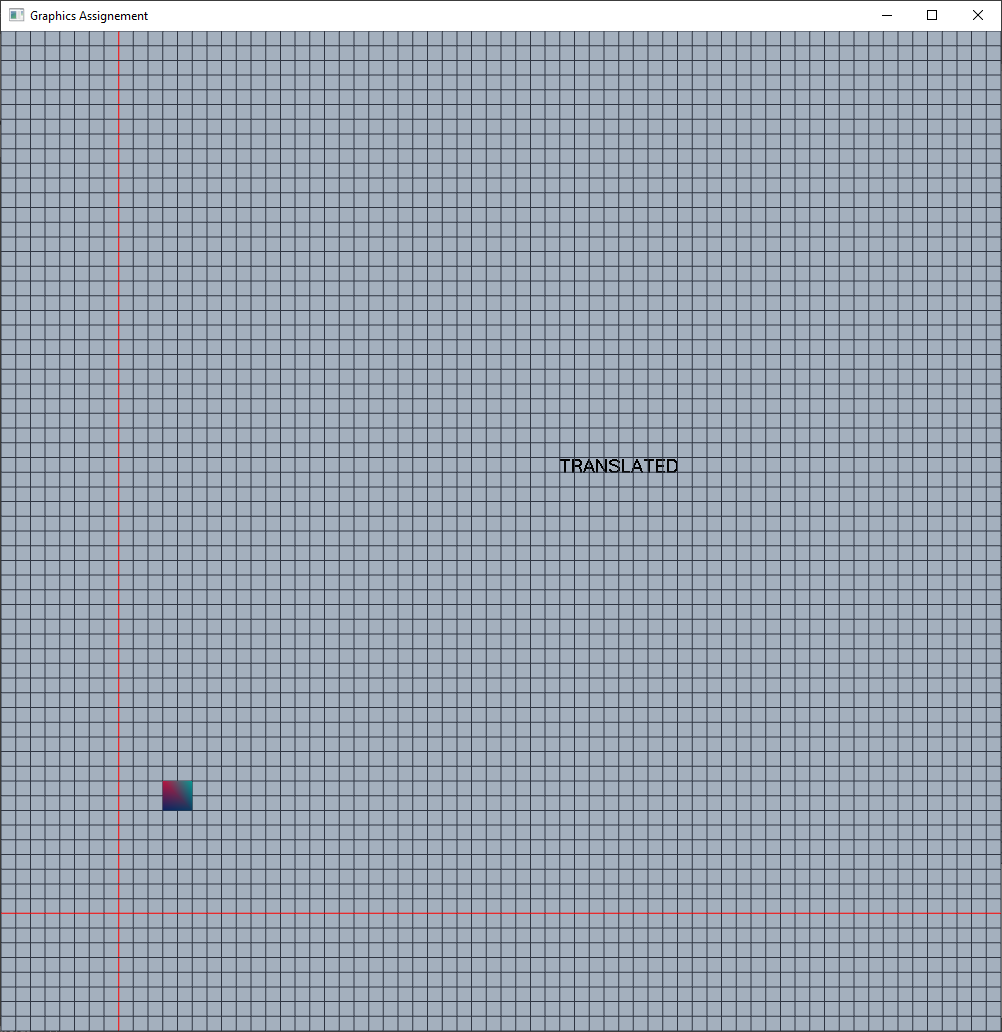
**(-1, 1)**

**(1, -1)**

**(-1, 1)**

**(1, 1)**

Figure Rotated Polygon



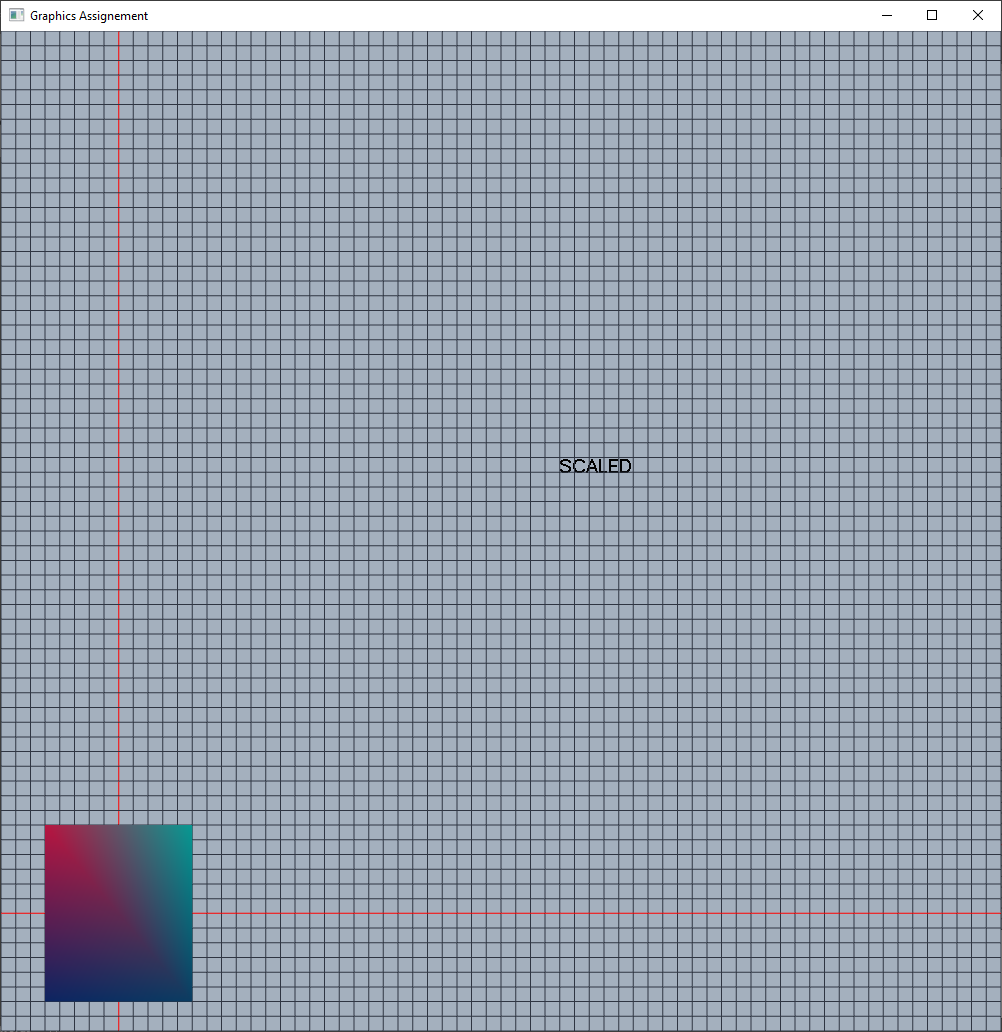
**(3, 9)**

**(5, 9)**

**(5, 7)**

**(3, 7)**

Figure Translated Polygon



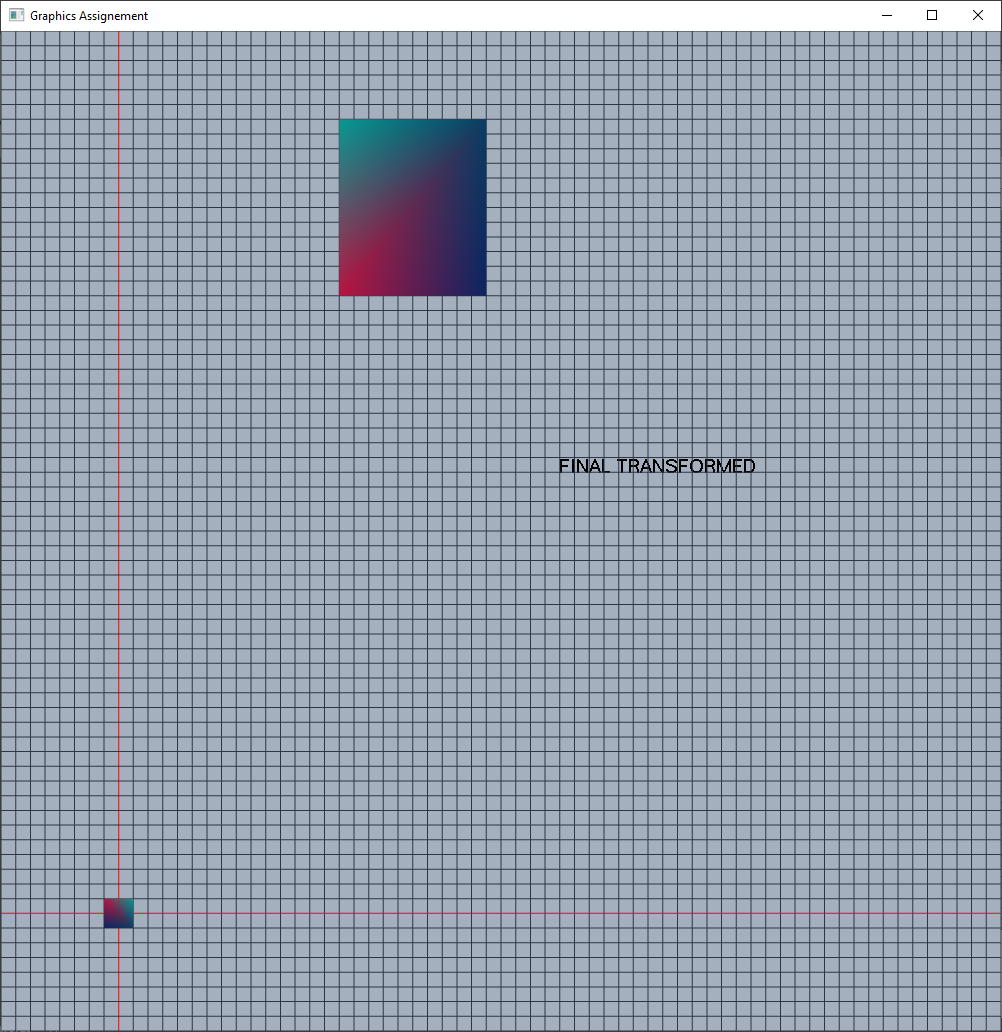
**(5, -6)**

**(-5, -6)**

**(5, 6)**

**(-5, 6)**

Figure Scaled Polygon



**(25, 42)**

**(25, 54)**

**(15, 42)**

**(15, 54)**

Figure Final Transformed Polygon

All the transformed applied sequentially,

Rotate:

Translate:

Scale:

Final Polygon:

# Bibliography