Assignment - 3

Name : Emmadi Sumith Kumar
Branch : CSE - B1
Subject : Computer Graphics
Roll No : UI20CS21

Write a program to perform 2D transformation using switch case
1. Translation
2. Rotation
3. Scaling
4. Reflection
5. Shearing

```
#include <iostream>
#include <graphics.h>
#include <cmath>
using namespace std;
void translation(float a, float b, float ans[])
{
    float tran[2];
    tran[0] = a;
    tran[1] = b;
    for (int i = 0; i < 2; i++)
    {
        ans[i] = ans[i] + tran[i];
    }
}
void scaling(float x, float y, float ans[])
{
    float scale[2][2];
    scale[0][0] = x;
    scale[0][1] = 0;
    scale[1][1] = y;
    scale[1][0] = 0;
    for (int i = 0; i < 2; i++)
    {
        ans[i] = ans[0] * scale[0][i] + ans[1] * scale[1][i];
    }
```

```
}
void rotation(float x, int opt, float ans[])
{
    float scale[2][2];
    switch (opt)
    {
    case 1:
        scale[0][0] = cos(x);
        scale[0][1] = sin(x);
        scale[1][1] = cos(x);
        scale[1][0] = -sin(x);
        for (int i = 0; i < 2; i++)
        {
            ans[i] = ans[0] * scale[0][i] + ans[1] * scale[1][i];
        }
        break;
    case 2:
        scale[0][0] = cos(x);
        scale[0][1] = sin(x);
        scale[1][1] = cos(x);
        scale[1][0] = -sin(x);
        for (int i = 0; i < 2; i++)
        {
            ans[i] = ans[0] * scale[0][i] + ans[1] * scale[1][i];
        }
        break;
    default:
        cout << " Invalid output " << endl;
    }
}
void shearing(float x, float y, float ans[])
{
    float scale[2][2];
    scale[0][0] = 1;
    scale[0][1] = y;
    scale[1][1] = 1;
    scale[1][0] = x;
    for (int i = 0; i < 2; i++)
    {
```

```
        ans[i] = ans[0] * scale[0][i] + ans[1] * scale[1][i];
    }
}
void reflection(int opt, float ans[])
{
    float scale[2][2];
    switch (opt)
    {
    case 1:
        scale[0][0] = 1;
        scale[0][1] = 0;
        scale[1][1] = -1;
        scale[1][0] = 0;
        for (int i = 0; i < 2; i++)
        {
            ans[i] = ans[0] * scale[i][0] + ans[1] * scale[i][1];
        }
        break;
    case 2:
        scale[0][0] = -1;
        scale[0][1] = 0;
        scale[1][1] = 1;
        scale[1][0] = 0;
        for (int i = 0; i < 2; i++)
        {
            ans[i] = ans[0] * scale[i][0] + ans[1] * scale[i][1];
        }
        break;
    case 3:
        scale[0][0] = -1;
        scale[0][1] = 0;
        scale[1][1] = -1;
        scale[1][0] = 0;
        for (int i = 0; i < 2; i++)
        {
            ans[i] = ans[0] * scale[i][0] + ans[1] * scale[i][1];
        }
        break;
    case 4:
```

```
        scale[0][0] = 0;
        scale[0][1] = 1;
        scale[1][1] = 0;
        scale[1][0] = 1;
        for (int i = 0; i < 2; i++)
        {
            ans[i] = ans[0] * scale[i][0] + ans[1] * scale[i][1];
        }
        break;
    case 5:
        scale[0][0] = 0;
        scale[0][1] = -1;
        scale[1][1] = 0;
        scale[1][0] = -1;
        for (int i = 0; i < 2; i++)
        {
            ans[i] = ans[0] * scale[i][0] + ans[1] * scale[i][1];
        }
        break;
    case 6:
        break;
    default:
        cout << "Invalid input " << endl;
    }
}
int main(void)
{
    int gd = DETECT, gm;
    initgraph(&gd, &gm, NULL);

    line(getmaxx() / 2, 0, getmaxx() / 2, getmaxy());
    line(0, getmaxy() / 2, getmaxx(), getmaxy() / 2);

    line(480, 80, 580, 150);
    line(480, 80, 380, 150);
    line(380, 150, 580, 150);

    float org_x = getmaxx() / 2.0, org_y = getmaxy() / 2.0;
```

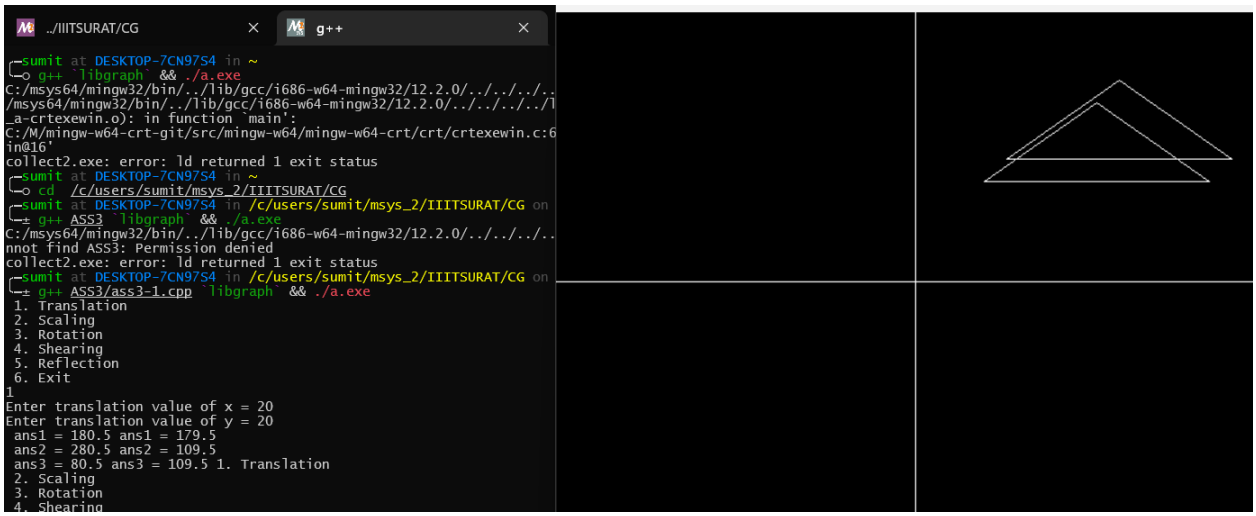
```
    float ans1[2];
    ans1[0] = 480 - org_x;
    ans1[1] = org_y - 80;
    float ans2[2];
    ans2[0] = 580 - org_x;
    ans2[1] = org_y - 150;
    float ans3[2];
    ans3[0] = 380 - org_x;
    ans3[1] = org_y - 150;
    while (1)
    {
        cout << " 1. Translation " << endl
             << " 2. Scaling " << endl
             << " 3. Rotation " << endl
             << " 4. Shearing " << endl
             << " 5. Reflection " << endl
             << " 6. Exit " << endl;
        int opt, opt1;
        float a, b;
        cin >> opt1;
        switch (opt1)
        {
        case 1:
            cout << "Enter translation value of x = ";
            cin >> a;
            cout << "Enter translation value of y = ";
            cin >> b;
            translation(a, b, ans1);
            translation(a, b, ans2);
            translation(a, b, ans3);

            for (int i = 0; i < 2; i++)
            {
                cout << " ans1 = " << ans1[i];
            }
            cout << endl;
            for (int i = 0; i < 2; i++)
            {
                cout << " ans2 = " << ans2[i];
```

```
            }
            cout << endl;
            for (int i = 0; i < 2; i++)
            {
                cout << " ans3 = " << ans3[i];
            }

            line(org_x + ans1[0], org_y - ans1[1], org_x + ans2[0], org_y -
ans2[1]);
            line(org_x + ans1[0], org_y - ans1[1], org_x + ans3[0], org_y -
ans3[1]);
            line(org_x + ans2[0], org_y - ans2[1], org_x + ans3[0], org_y -
ans3[1]);
            // getch();
            // closegraph();
            break;
        case 2:
            cout << "Enter scaling factor in x = " << endl;
            cin >> a;
            cout << "Enter scaling factor in y = " << endl;
            cin >> b;
            scaling(a, b, ans1);
            scaling(a, b, ans2);
            scaling(a, b, ans3);

            line(org_x + ans1[0], org_y - ans1[1], org_x + ans2[0], org_y -
ans2[1]);
            line(org_x + ans1[0], org_y - ans1[1], org_x + ans3[0], org_y -
ans3[1]);
            line(org_x + ans2[0], org_y - ans2[1], org_x + ans3[0], org_y -
ans3[1]);
            // getch();
            // closegraph();
            break;
        case 3:
            cout << "Enter angle by which we want to rotate about origin = " <<
endl;
            float z;
            cin >> z;
```

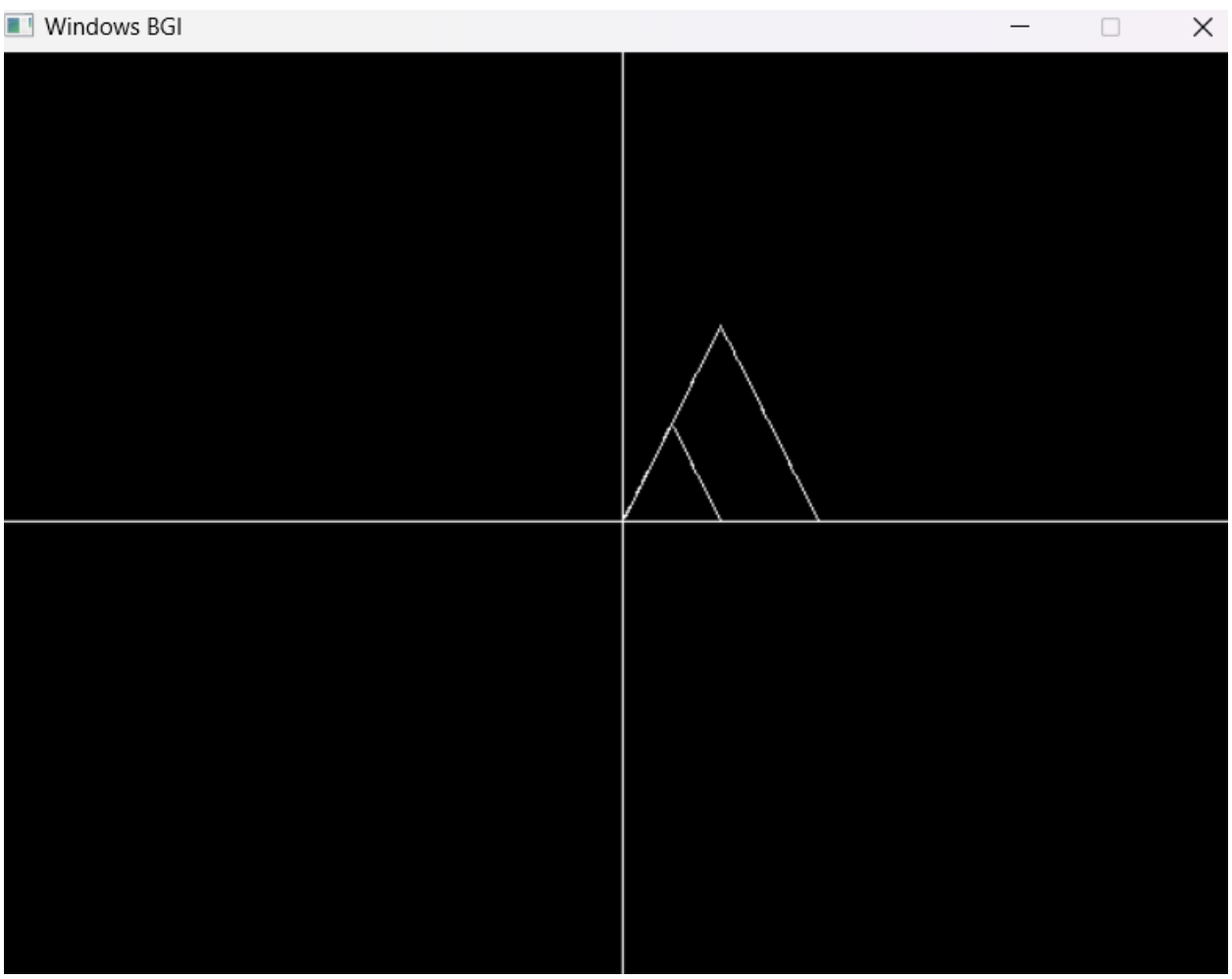
```
            cout << "Choose option of rotation = " << endl
                 << " 1: Anti-clockwise "
                 << " 2: Clockwise " << endl;
            cin >>
                opt;
            rotation(z, opt, ans1);
            rotation(z, opt, ans2);
            rotation(z, opt, ans3);

            line(org_x + ans1[0], org_y - ans1[1], org_x + ans2[0], org_y -
ans2[1]);
            line(org_x + ans1[0], org_y - ans1[1], org_x + ans3[0], org_y -
ans3[1]);
            line(org_x + ans2[0], org_y - ans2[1], org_x + ans3[0], org_y -
ans3[1]);
            // getch();
            // closegraph();
            break;
        case 4:
            cout << "Choose shearing factor along x direction = " << endl;

            cin >> a;
            cout << "Choose shearing factor along y direction = " << endl;

            cin >> b;
            shearing(a, b, ans1);
            shearing(a, b, ans2);
            shearing(a, b, ans3);

            line(org_x + ans1[0], org_y - ans1[1], org_x + ans2[0], org_y -
ans2[1]);
            line(org_x + ans1[0], org_y - ans1[1], org_x + ans3[0], org_y -
ans3[1]);
            line(org_x + ans2[0], org_y - ans2[1], org_x + ans3[0], org_y -
ans3[1]);
            // getch();
            // closegraph();*/
            break;
```

```
        case 5:
            cout << "Choose reflection around axis :- " << endl
                 << " 1: x-axis " << endl
                 << " 2: y- axis " << endl
                 << " 3 : origin " << endl
                 << " 4 : y = x axis " << endl
                 << " 5 : y = -x axis " << endl;
            cin >> opt;
            reflection(opt, ans1);
            reflection(opt, ans2);
            reflection(opt, ans3);

            line(org_x + ans1[0], org_y - ans1[1], org_x + ans2[0], org_y -
ans2[1]);
            line(org_x + ans1[0], org_y - ans1[1], org_x + ans3[0], org_y -
ans3[1]);
            line(org_x + ans2[0], org_y - ans2[1], org_x + ans3[0], org_y -
ans3[1]);

            break;
        case 6:
            return 0;
        default:
            cout << "Invalid input " << endl;
        }
    }
    getch();
    closegraph();
    return 0;
}
```

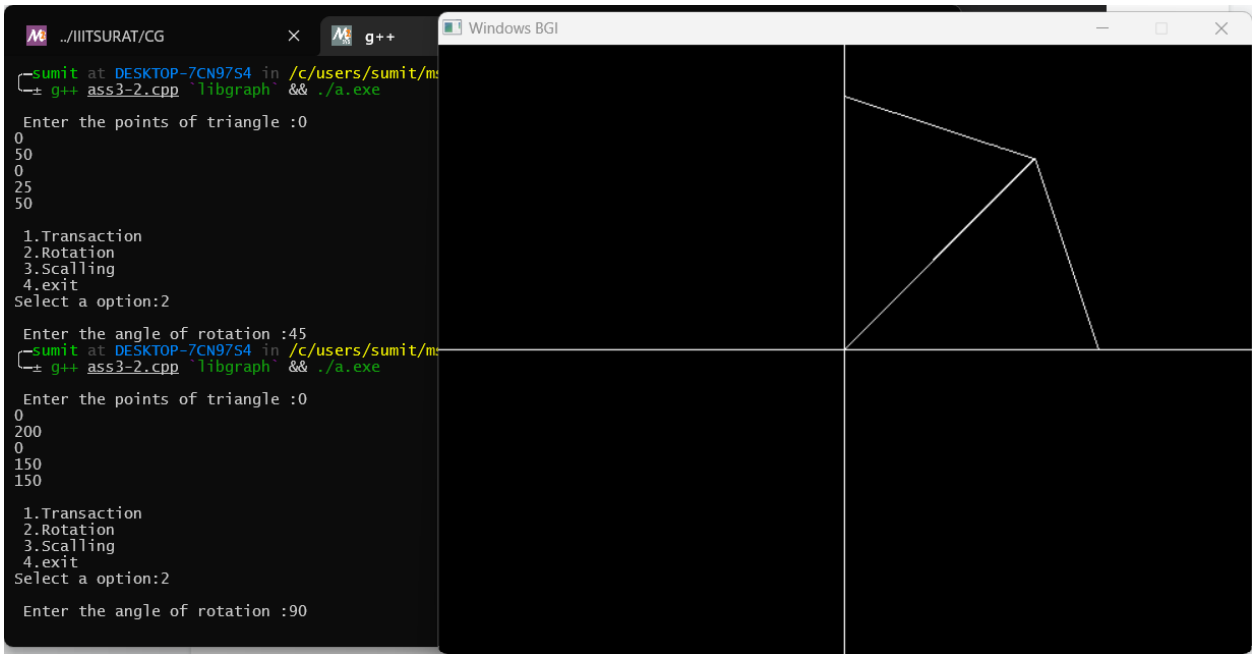
1. Translation



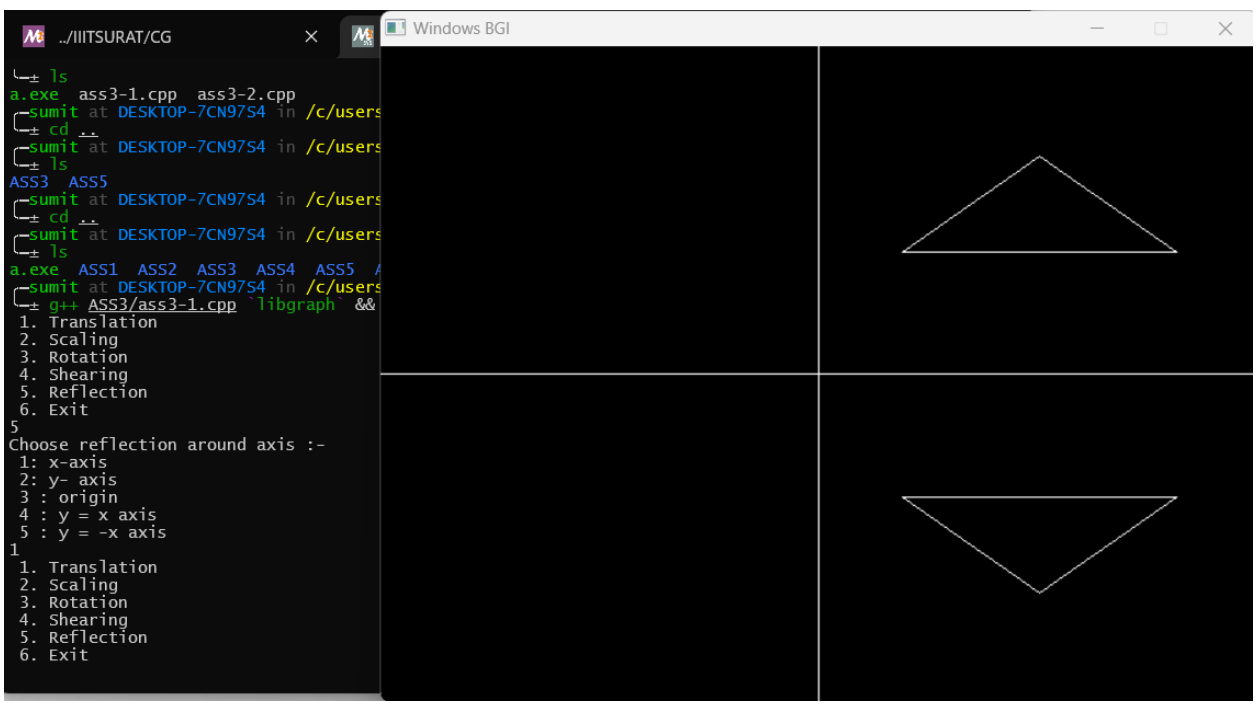
2. Scaling



1. Rotation



2. Reflection



3. **Shearing**