Experiment 9

implementation of 2-D transformation with help of graphic primitive

- translation
- scaling
- rotation
- reflection
- shearing

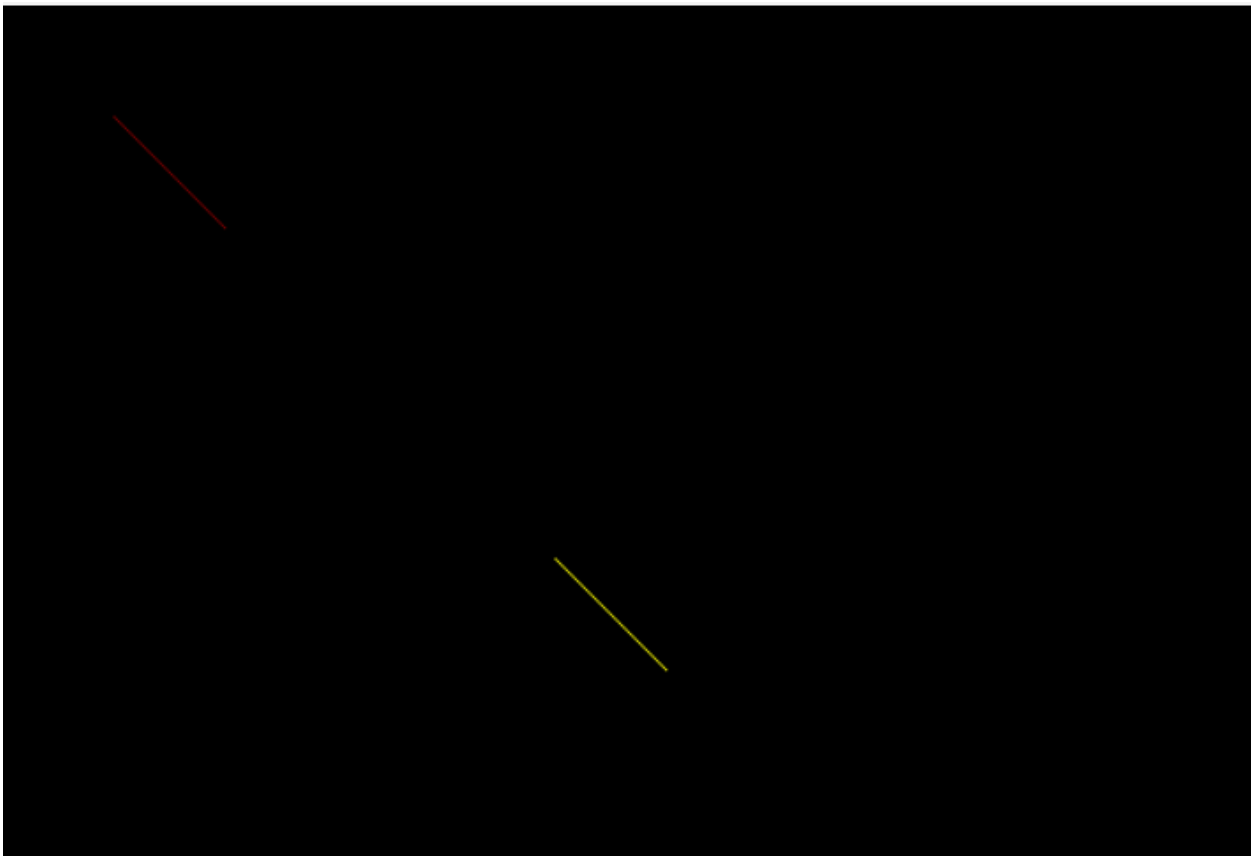
translation -

```
#include <conio.h>
#include <graphics.h>
#include <iostream>
#include <stdio.h>
int gd = DETECT, gm;
int n, xs[100], ys[100], i, ty, tx;
void draw();
void translate();
using namespace std;
int main()
{
```

```
    cout << "Enter no. of sides in polygon: ";
    cin >> n;
    cout << "Enter coordinates x, y for each vertex: ";
    for (i = 0; i < n; i++) {
        cin >> xs[i] >> ys[i];
    }
    cout << "Enter distances for translation (in x and y "
            "directions): ";
    cin >> tx >> ty;
    initgraph(&gd, &gm, (char*)"");
    cleardevice();
    setcolor(RED);
    draw();
    translate();
    setcolor(YELLOW);
    draw();
    getch();
    closegraph();
    return 0;
}
void draw()
{
    for (i = 0; i < n; i++) {
        line(xs[i], ys[i], xs[(i + 1) % n],
             ys[(i + 1) % n]);
    }
}
void translate()
{
    for (i = 0; i < n; i++) {
        xs[i] += tx;
        ys[i] += ty;
    }
}
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS C:\Users\JAYA\Desktop\cg\programs> ./2dtrans
Enter no. of sides in polygon: 3
Enter coordinates x, y for each vertex: 50
50
50
50
100
100
Enter distances for translation (in x and y directions): 200
200
```



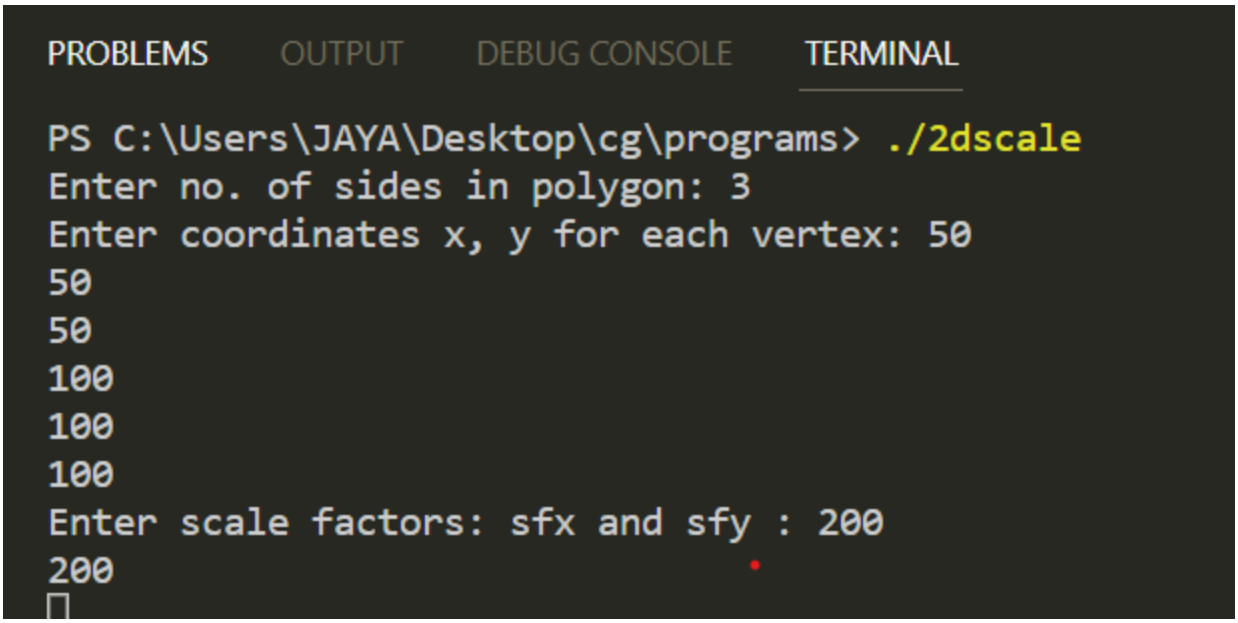
scaling

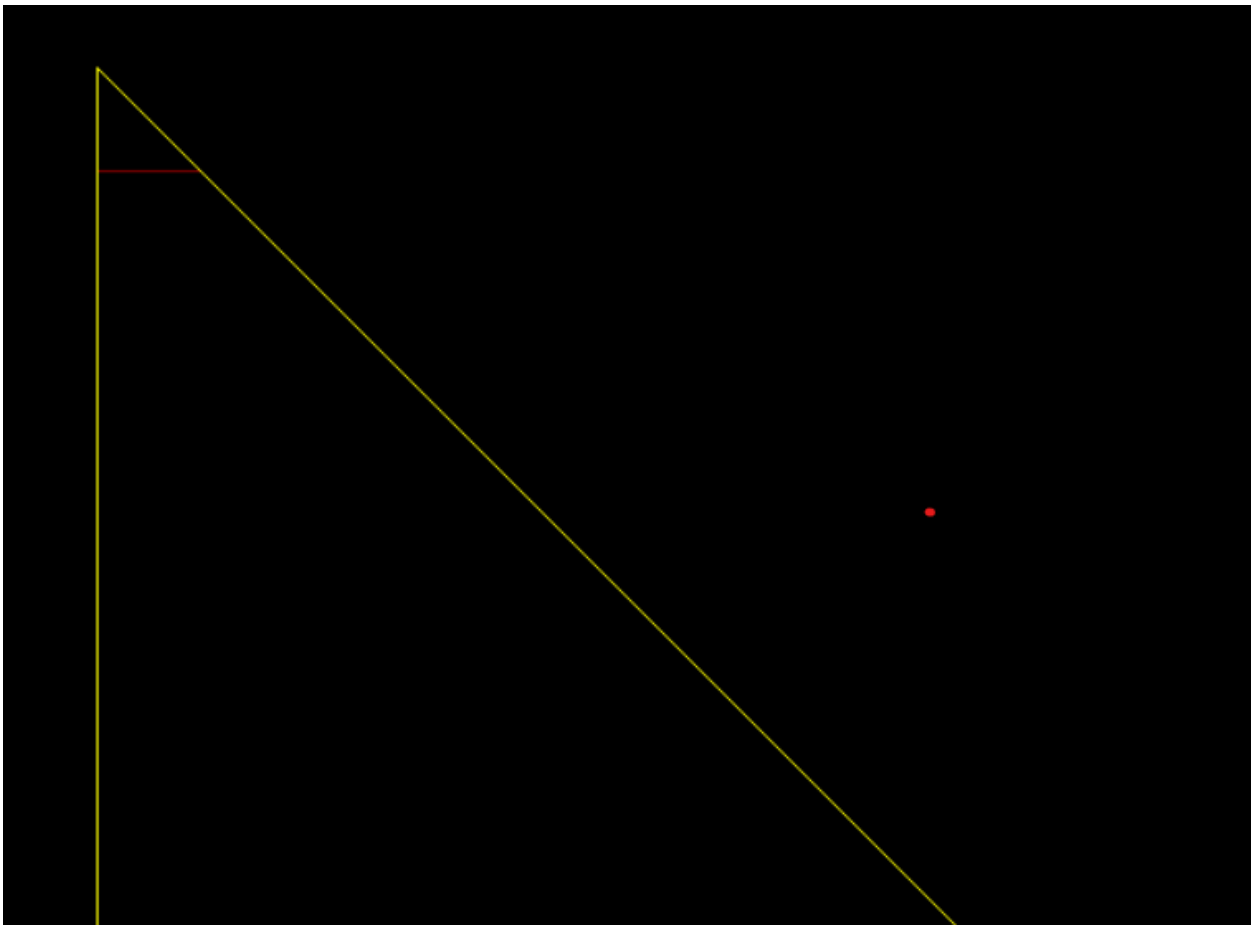
```
#include <conio.h>
#include <graphics.h>
#include <iostream>
#include <math.h>
#include <stdio.h>

int gd = DETECT, gm;
int n, x[100], y[100], i;
float sfx, sfy;
void draw();
void scale();
using namespace std;

int main()
{
    cout << "Enter no. of sides in polygon: ";
    cin >> n;
    cout << "Enter coordinates x, y for each vertex: ";
    for (i = 0; i < n; i++) {
        cin >> x[i] >> y[i];
    }
    cout << "Enter scale factors: sfx and sfy : ";
    cin >> sfx >> sfy;
    initgraph(&gd, &gm, (char*)"");
    cleardevice();
    setcolor(RED);
    draw();
    scale();
    setcolor(YELLOW);
    draw();
    getch();
    closegraph();
    return 0;
}
```

```
void draw()
{
    for (i = 0; i < n; i++) {
        line(x[i], y[i], x[(i + 1) % n], y[(i + 1) % n]);
    }
}
void scale()
{
    for (i = 0; i < n; i++) {
        x[i] = x[0] + (int)((float)(x[i] - x[0]) * sfx);
        y[i] = y[0] + (int)((float)(y[i] - y[0]) * sfx);
    }
}
```



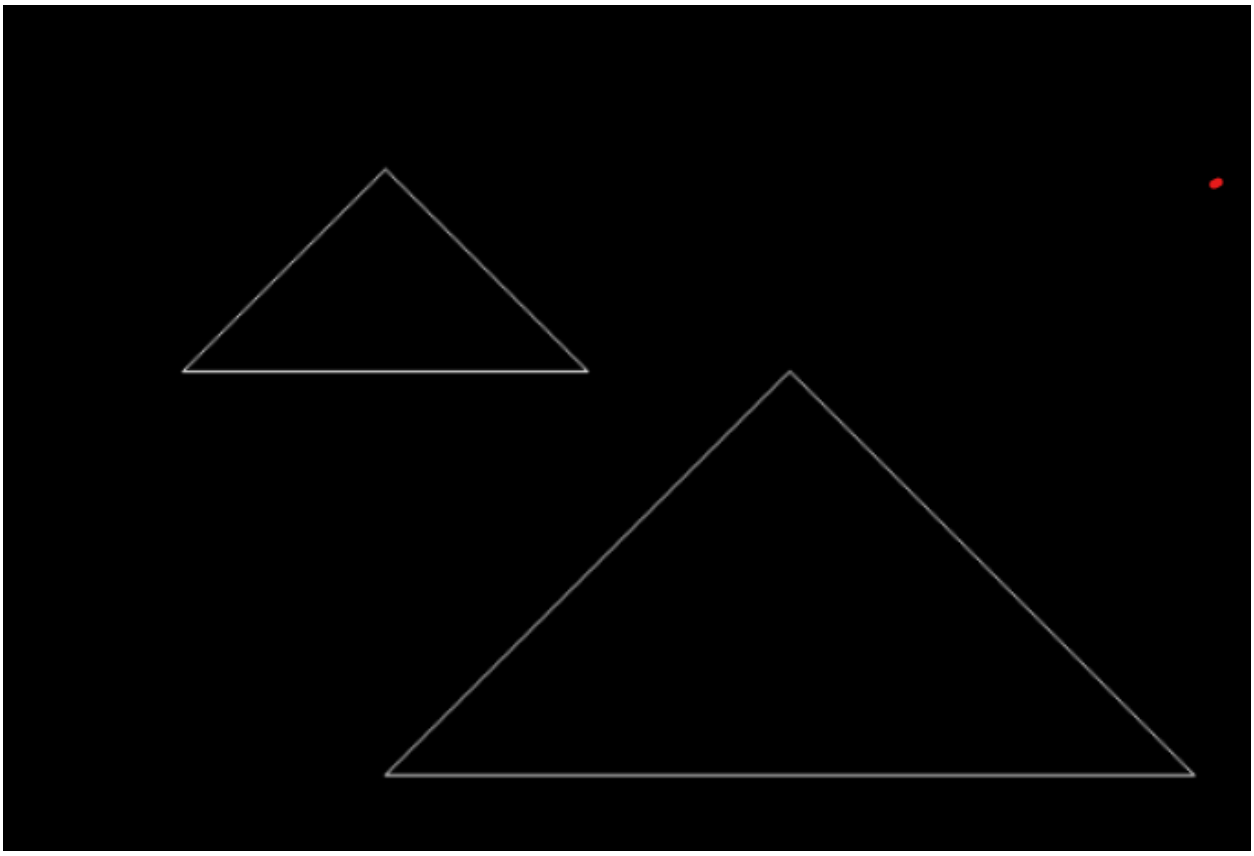


rotation

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
#include<iostream>
using namespace std;
int main()
{
 int gd=0,gm,x1,y1,x2,y2,x3,y3;
 double s,c, angle;
 initgraph(&gd, &gm, (char*)"");
 setcolor(RED);
 cout<<"Enter coordinates of triangle: ";
```

```
 cin>>x1>>y1;
cin>>x2>>y2;
cin>>x3>>y3;
 setbkcolor(WHITE);
 cleardevice();
 line(x1,y1,x2,y2);
 line(x2,y2, x3,y3);
 line(x3, y3, x1, y1);
 cout<<"Enter rotation angle: ";
 cin>>angle;
 c = cos(angle *M_PI/180);
 s = sin(angle *M_PI/180);
 x1 = floor(x1 * c + y1 * s);
 y1 = floor(-x1 * s + y1 * c);
 x2 = floor(x2 * c + y2 * s);
 y2 = floor(-x2 * s + y2 * c);
 x3 = floor(x3 * c + y3 * s);
 y3 = floor(-x3 * s + y3 * c);
 setcolor(GREEN);
 line(x1, y1 ,x2, y2);
 line(x2,y2, x3,y3);
 line(x3, y3, x1, y1);
 getch();
 closegraph();
 return 0;
}
```

```
PS C:\Users\JAYA\Desktop\cg\programs> ./2drotate
Enter coordinates of triangle: 50
50
50
100
100
100
Enter rotation angle: 180
PS C:\Users\JAYA\Desktop\cg\programs> 180
180
```

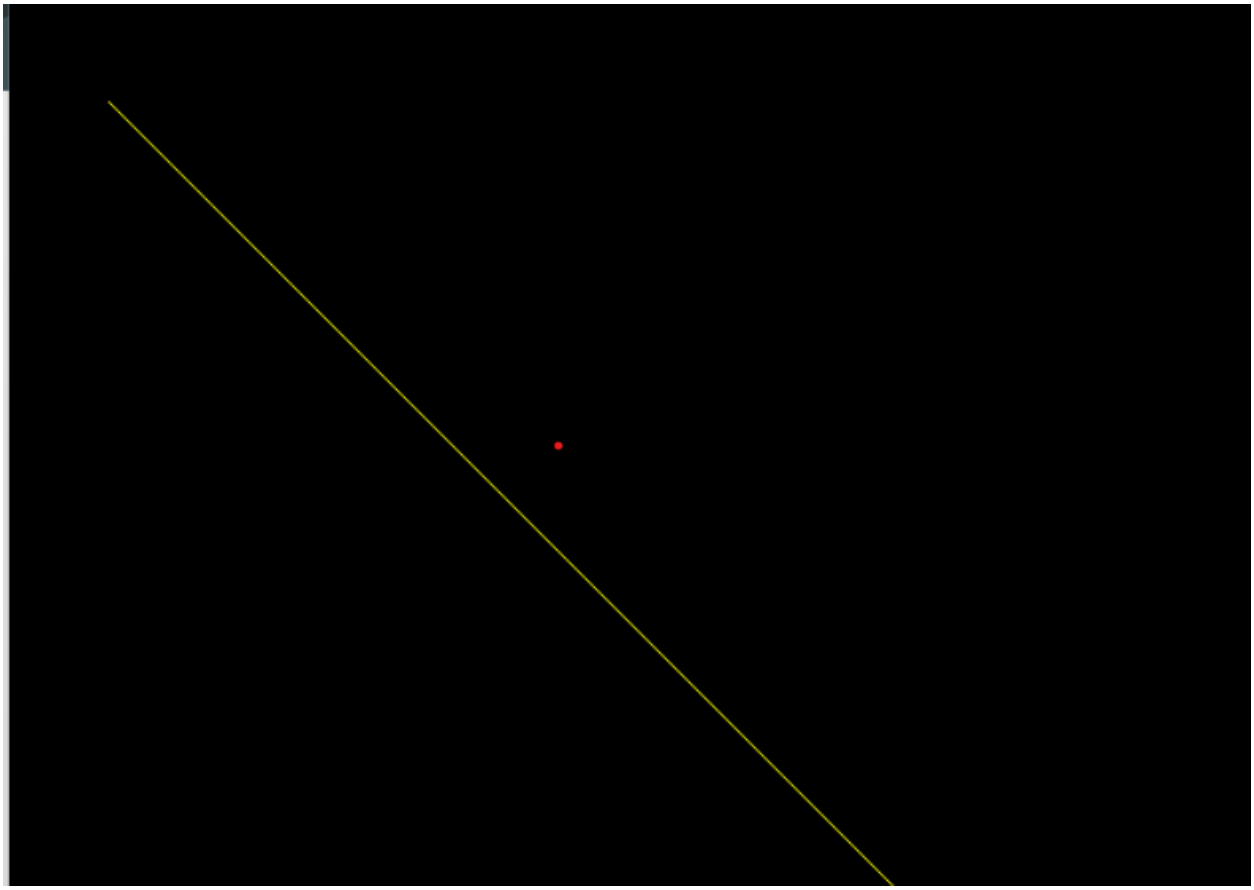


shearing

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
#include<iostream>
#include<math.h>
```

```
int gd= DETECT, gm;
int n,x[100],y[100],i;
float sfx, sfy;
void draw();
void scale();
using namespace std;
int main(){
cout<<"Enter no. of sides in polygon: ";
cin>>n;
cout<<"Enter coordinates x, y for each vertex: ";
for(i=0;i<n;i++){
cin>>x[i]>>y[i];}
cout<<"Enter scale factors: sfx and sfy : ";
cin>>sfx>>sfy;
initgraph(&gd, &gm, (char*)"");
cleardevice();
setcolor(RED);
draw();
scale();
setcolor(YELLOW);
draw();
getch();
closegraph();
return 0;
}
void draw(){
for(i=0; i<n; i++){
line(x[i],y[i],x[(i+1)%n],y[(i+1)%n]);}
}
void scale(){
for(i=0; i<n; i++){
x[i]=x[0]+(int)((float)(x[i]-x[0])*sfx);
y[i]=y[0]+(int)((float)(y[i]-y[0])*sfx);
}
}
```

```
PS C:\Users\JAYA\Desktop\cg\programs> ./2dshear
Enter no. of sides in polygon: 3
Enter coordinates x, y for each vertex: 50
50
50
50
100
100
Enter scale factors: sfx and sfy : 50
50
```



reflection

```
#include <conio.h>
#include <graphics.h>
#include <stdio.h>
```

```
// Driver Code
int main()
{
    // Initialize the drivers
    int gm, gd = DETECT, ax, x1 = 100;
    int x2 = 100, x3 = 200, y1 = 100;
    int y2 = 200, y3 = 100;


    initgraph(&gd, &gm, (char *)"");
    cleardevice();

    // Draw the graph
    line(getmaxx() / 2, 0, getmaxx() / 2,
         getmaxy());
    line(0, getmaxy() / 2, getmaxx(),
         getmaxy() / 2);

    // Object initially at 2nd quadrant
    printf("Before Reflection Object"
           " in 2nd Quadrant");

    // Set the color
    setcolor(14);
    line(x1, y1, x2, y2);
    line(x2, y2, x3, y3);
    line(x3, y3, x1, y1);
    getch();

    // After reflection
    printf("\nAfter Reflection");

    // Reflection along origin i.e.,
    // in 4th quadrant
    setcolor(4);
```

```
    line(getmaxx() - x1, getmaxy() - y1,
         getmaxx() - x2, getmaxy() - y2);

    line(getmaxx() - x2, getmaxy() - y2,
         getmaxx() - x3, getmaxy() - y3);

    line(getmaxx() - x3, getmaxy() - y3,
         getmaxx() - x1, getmaxy() - y1);

    // Reflection along x-axis i.e.,
    // in 1st quadrant
    setcolor(3);
    line(getmaxx() - x1, y1,
         getmaxx() - x2, y2);
    line(getmaxx() - x2, y2,
         getmaxx() - x3, y3);
    line(getmaxx() - x3, y3,
         getmaxx() - x1, y1);

    // Reflection along y-axis i.e.,
    // in 3rd quadrant
    setcolor(2);
    line(x1, getmaxy() - y1, x2,
         getmaxy() - y2);
    line(x2, getmaxy() - y2, x3,
         getmaxy() - y3);
    line(x3, getmaxy() - y3, x1,
         getmaxy() - y1);
    getch();

    // Close the graphics
    closegraph();
}
```