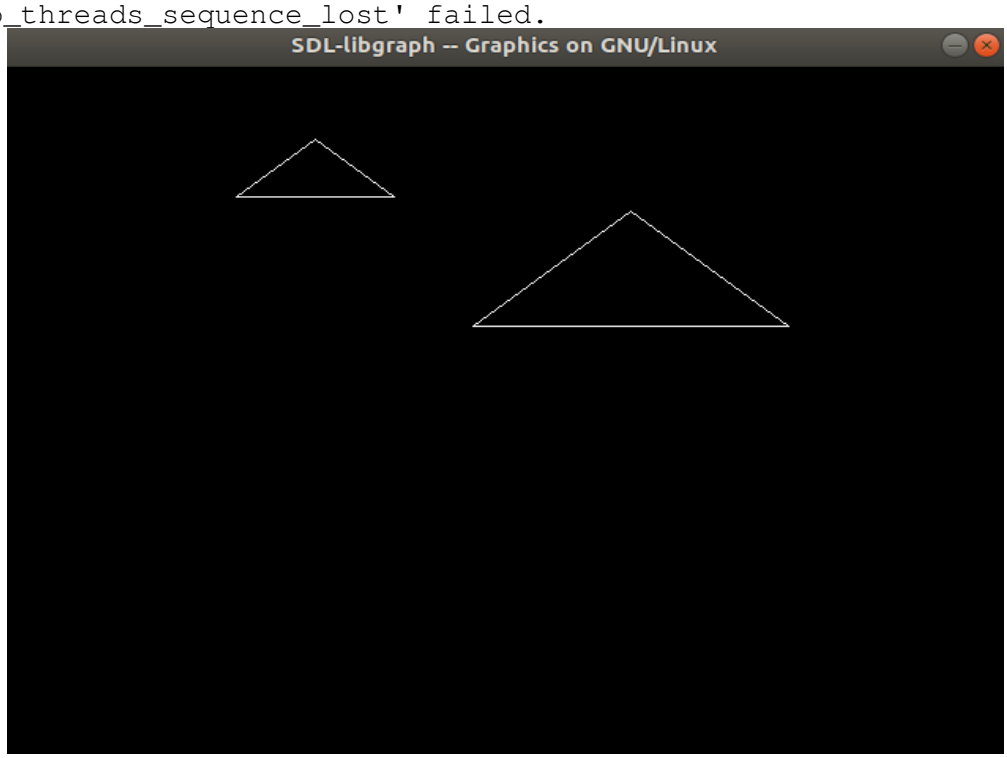
```
#include<iostream>
#include<math.h>
#include<graphics.h>
#define con 3.142857143/180
using namespace std;
int main()
{
int gd=DETECT,gm;
int a[3][3]; // object matrix
int I[3][3],ch;
int i,j,tx,ty,sx,sy;
double angle;
int k;
int prod[3][3];
char ans;

a[0][2]=1;
a[1][2]=1;
a[2][2]=1;
cout<<"\nEnter the Object Matrix: ";
for(i=0;i<3;i++)
{
      for(j=0;j<2;j++)
      {
            cin>>a[i][j];
      }
}
cout<<"\nObject Matrix: "<<endl;
for(i=0;i<3;i++)
{
      for(j=0;j<3;j++)
      {
            cout<<a[i][j]<<"\t";
      }
      cout<<"\n";
}
for(i=0;i<3;i++)
{
      for(j=0;j<3;j++)
      {
            if(i==j)
            {
                  I[i][j]=1;
            }
            else
            {
                  I[i][j]=0;
            }
      }
}
do
{
cout<<"\n1.Translate \n2.Scale \n3.Rotate \nEnter your choice: ";
cin>>ch;
switch(ch)
{
case 1:
cout<<"\nEnter tx and ty: ";
cin>>tx>>ty;
initgraph(&gd,&gm,NULL);
line(a[0][0],a[0][1],a[1][0],a[1][1]);
line(a[1][0],a[1][1],a[2][0],a[2][1]);
line(a[0][0],a[0][1],a[2][0],a[2][1]);
I[2][0]=tx;
I[2][1]=ty;
cout<<"\nTranslation Matrix: "<<endl;
```

```
for(i=0;i<3;i++)
{
      for(j=0;j<3;j++)
      {
            cout<<I[i][j]<<"\t";
      }
      cout<<"\n";
}


for(i=0;i<3;i++)
{
      for(j=0;j<3;j++)
      {
            prod[i][j]=0;
            for(k=0;k<3;k++)
            {
                  prod[i][j]=prod[i][j]+a[i][k]*I[k][j];
            }

      }
}
      cout<<"\n";
      for(i=0;i<3;i++)
      {
            for(j=0;j<3;j++)
            {
                  cout<<prod[i][j]<<"\t";
            }
            cout<<"\n";
      }
line(prod[0][0],prod[0][1],prod[1][0],prod[1][1]);
line(prod[1][0],prod[1][1],prod[2][0],prod[2][1]);
line(prod[0][0],prod[0][1],prod[2][0],prod[2][1]);
delay(5000);
closegraph();
break;
case 2:
cout<<"\nEnter sx and sy: ";
cin>>sx>>sy;
initgraph(&gd,&gm,NULL);
line(a[0][0],a[0][1],a[1][0],a[1][1]);
line(a[1][0],a[1][1],a[2][0],a[2][1]);
line(a[0][0],a[0][1],a[2][0],a[2][1]);
I[0][0]=sx;
I[1][1]=sy;
cout<<"\nScaling Matrix: "<<endl;
for(i=0;i<3;i++)
{
      for(j=0;j<3;j++)
      {
            cout<<I[i][j]<<"\t";
      }
      cout<<"\n";
}

for(i=0;i<3;i++)
{
      for(j=0;j<3;j++)
      {
            prod[i][j]=0;
            for(k=0;k<3;k++)
            {
                  prod[i][j]=prod[i][j]+a[i][k]*I[k][j];
            }
```

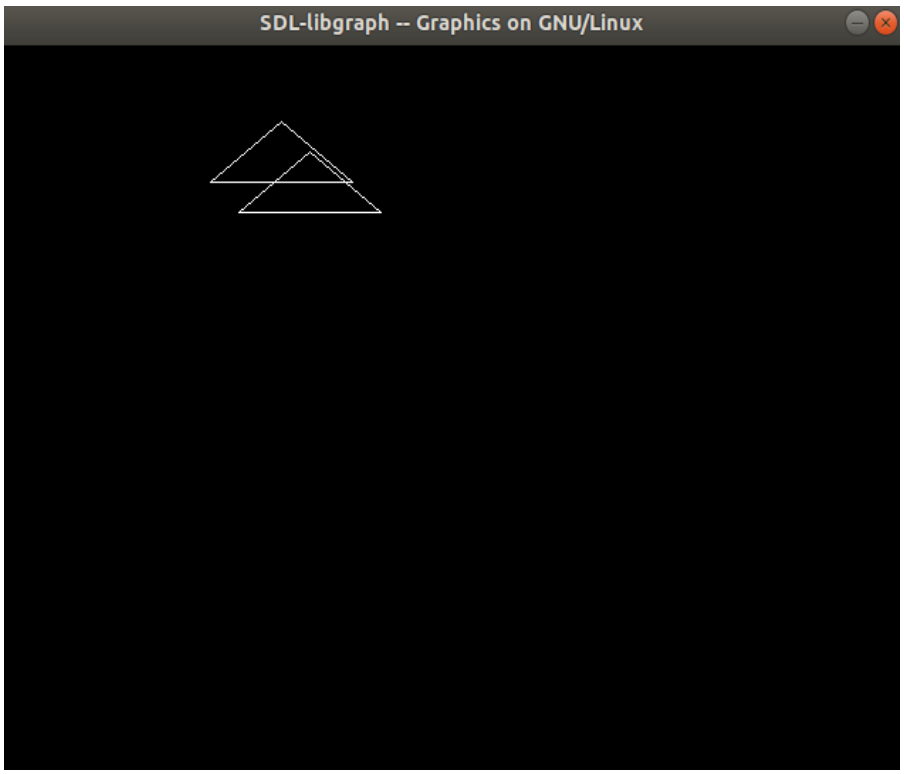
```
	}
}
	cout<<"\n";
	for(i=0;i<3;i++)
	{
		for(j=0;j<3;j++)
		{
			cout<<prod[i][j]<<"\t";
		}
		cout<<"\n";
	}
line(prod[0][0],prod[0][1],prod[1][0],prod[1][1]);
line(prod[1][0],prod[1][1],prod[2][0],prod[2][1]);
line(prod[0][0],prod[0][1],prod[2][0],prod[2][1]);
delay(5000);
closegraph();
break;
case 3:
double b[3][3];
double id[3][3];
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
b[i][j]=a[i][j];
}
}
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
id[i][j]=I[i][j];
}
}
cout<<"\nAngle";
cin>>angle;
angle = angle *con;
initgraph(&gd,&gm,NULL);
line(b[0][0],b[0][1],b[1][0],b[1][1]);
line(b[1][0],b[1][1],b[2][0],b[2][1]);
setcolor(BLUE);
line(b[0][0],b[0][1],b[2][0],b[2][1]);
id[0][0]=cos(angle);
id[0][1]=sin(angle);
id[1][0]=-sin(angle);
id[1][1]=cos(angle);
cout<<"\nRotational Matrix: "<<endl;
for(i=0;i<3;i++)
{
	for(j=0;j<3;j++)
	{
		cout<<id[i][j]<<"\t";
	}
	cout<<"\n";
}


for(i=0;i<3;i++)
{
	for(j=0;j<3;j++)
	{
		prod[i][j]=0;
		for(k=0;k<3;k++)
		{
			prod[i][j]=prod[i][j]+b[i][k]*id[k][j];
		}
```

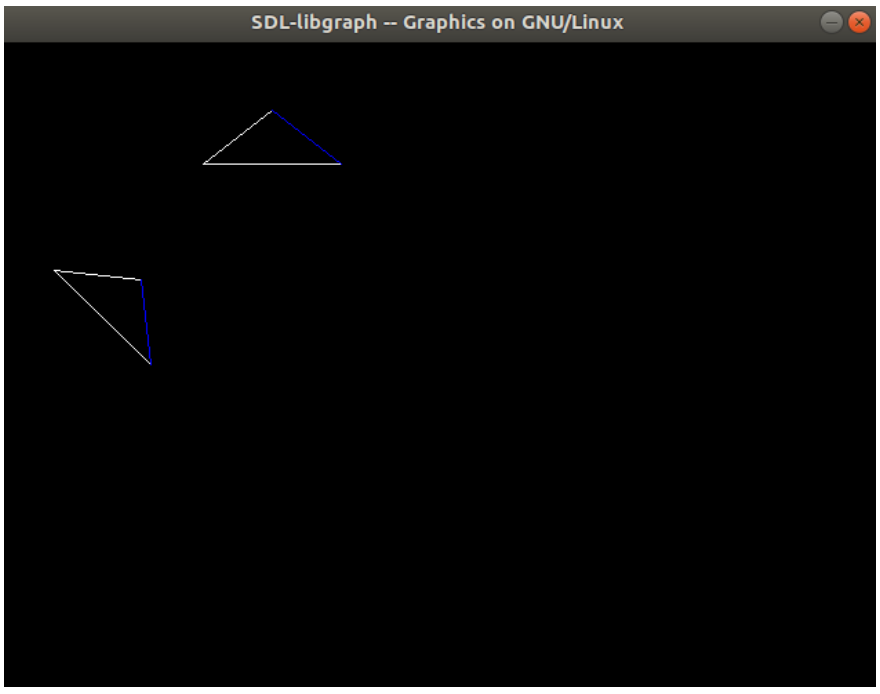
```
      }
}
      cout<<"\n";
      for(i=0;i<3;i++)
      {
            for(j=0;j<3;j++)
            {
                  cout<<prod[i][j]<<"\t";
            }
            cout<<"\n";
      }
setcolor(WHITE);
line(prod[0][0],prod[0][1],prod[1][0],prod[1][1]);
line(prod[1][0],prod[1][1],prod[2][0],prod[2][1]);
setcolor(BLUE);
line(prod[0][0],prod[0][1],prod[2][0],prod[2][1]);
delay(5000);
closegraph();
break;
}

cout<<"\nDo you want to continue?(y/n): ";
cin>>ans;
}while(ans == 'y' || ans == 'Y');
return 0;
}
/*
Output:
Enter the Object Matrix: 200
50
150
90
250
90

Object Matrix:
200   50    1
150   90    1
250   90    1

1.Translate
2.Scale
3.Rotate
Enter your choice: 1

Enter tx and ty: 20
20

Translation Matrix:
1     0     0
0     1     0
20    20    1

220   70    1
170   110   1
270   110   1
[xcb] Unknown sequence number while processing queue
[xcb] Most likely this is a multi-threaded client and XInitThreads has
not been called
[xcb] Aborting, sorry about that.
a.out: ../../src/xcb_io.c:259: poll_for_event: Assertion `!
xcb_xlib_threads_sequence_lost' failed.

Do you want to continue?(y/n): y
```

```
1.Translate
2.Scale
3.Rotate
Enter your choice: 2

Enter sx and sy: 2
2

Scaling Matrix:
2     0     0
0     2     0
20    20    1

420   120   1
320   200   1
520   200   1

Do you want to continue?(y/n): y

1.Translate
2.Scale
3.Rotate
Enter your choice: 3

Angle45

Rotational Matrix:
0.706883   0.70733    0
-0.70733   0.706883   0
0     0     1

105   176   1
42    169   1
112   239   1
[xcb] Unknown sequence number while processing queue
[xcb] Most likely this is a multi-threaded client and XInitThreads has
not been called
[xcb] Aborting, sorry about that.
a.out: ../../src/xcb_io.c:259: poll_for_event: Assertion `!
xcb_xlib_threads_sequence_lost' failed.
```




```
Do you want to continue?(y/n): n*/
```


SDL-libgraph -- Graphics on GNU/Linux


SDL-libgraph -- Graphics on GNU/Linux