

LAB MANUAL

For

Computer Graphics

Laboratory

2020 -2021

BTech 5th Sem

Dr. Dhananjay Bhakta (CSE)



भारतीय सूचना प्रौद्योगिकी संस्थान राँची

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, RANCHI

(An Institution of National importance under act of Parliament)

(Ranchi - 834010), Jharkhand

Department of Computer Science and Engineering

INDEX

Sr. No.	Experiment	Date
1	<i>DDA Algorithm</i>	
2	<i>Bresenham Line Algorithm</i>	
3	<i>Midpoint Circle Algorithm</i>	
4	<i>Bresenham Circle Algorithm</i>	
5	<i>Midpoint Eclipse Algorithm</i>	
6	<i>Stylish Text and Filling Algorithms</i>	
7	<i>Basic 2D Transformation</i>	
8	<i>Drawing a Car</i>	
9	<i>Polygon Clipping using Sutherland - Hodgeman Algorithm</i>	
10	<i>Line Clipping using Cohen - Sutherland Algorithm.</i>	
11	<i>Basic 3D Transformation</i>	

By Prithwiraj Samanta(2018UGCS002R)

EXPERIMENT NO.1

Aim

Draw a line using DDA algorithm.

Theory

A linear DDA starts by calculating the smaller of dy or dx for a unit increment of the other. A line is then sampled at unit intervals in one coordinate and corresponding integer values nearest the line path are determined for the other coordinate.

Considering a line with positive slope, if the slope is less than or equal to 1, we sample at unit x intervals ($dx=1$) and compute successive y values as

Subscript k takes integer values starting from 0, for the 1st point and increases by 1 until endpoint is reached. y value is rounded off to nearest integer to correspond to a screen pixel.

For lines with slope greater than 1, we reverse the role of x and y i.e. we sample at $dy=1$ and calculate consecutive x values as

Similar calculations are carried out to determine pixel positions along a line with negative slope. Thus, if the absolute value of the slope is less than 1, we set $dx=1$ if i.e. the starting extreme point is at the left.

Program

```
#include<bits/stdc++.h>
#include<graphics.h>
#define ll long long int
```

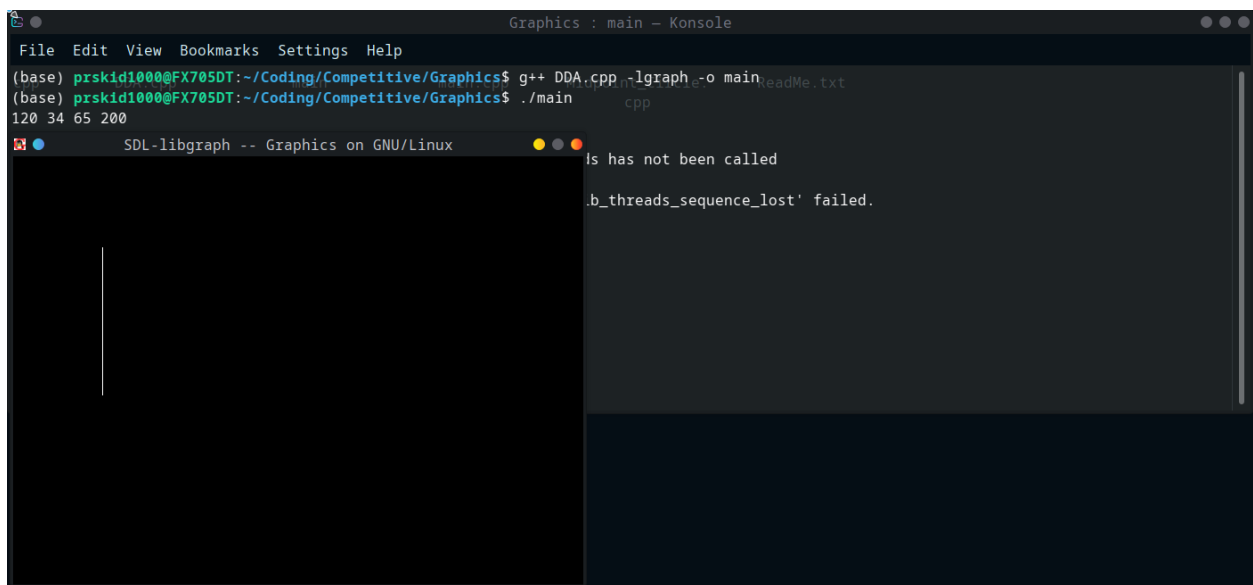
```

using namespace std;
int main()
{
int gd = DETECT, gm, tmp = 0;
ll x1 = 0, x2 = 0, y1 = 0, y2 = 0, dx = 0, dy = 0;
ll steps = 0, xinc = 0, yinc = 0, x = 0, y = 0;
cin >> x1 >> y1;
cin >> x2 >> y2;
dx = x2 - x1;
dy = y2 - y1;
if(abs(dx) > abs(dy)) steps = abs(dx);
else steps = abs(dy);
xinc = dx / steps;
yinc = dy / steps;
x += xinc;
y += yinc;
//declare all variables before it
initgraph(&gd,&gm, NULL);
//draw here
for(ll i = 0; i < steps; i++)
{
x += xinc;
y += yinc;
putpixel(100 + x, 100 + y, 15);
delay(300);
}

```

```
//draw ends  
getche();  
closegraph();  
return 0;  
}
```

Output



```
Graphics : main - Konsole  
File Edit View Bookmarks Settings Help  
(base) prskid1000@FX705DT:~/Coding/Competitive/Graphics$ g++ DDA.cpp -lgraph -o main  
(base) prskid1000@FX705DT:~/Coding/Competitive/Graphics$ ./main  
120 34 65 200  
SDL-libgraph -- Graphics on GNU/Linux  
ls has not been called  
'b_threads_sequence_lost' failed.
```

EXPERIMENT NO.2

Aim

Draw a line using Bresenham's algorithm

Theory

Let's define $f(x,y) = 0 = ax + by + c$

$$D = f(x_0 + 1, y_0 + 0.5) - f(x_0, y_0)$$

If $D > 0$ choose $(x_0 + 1, y_0)$ and $\text{del}(D) = \text{del}(y)$

If $D > 0$ choose $(x_0 + 1, y_0 + 1)$ and $\text{del}(D) = \text{del}(y) - \text{del}(x)$

One performance issue is the $\frac{1}{2}$ factor in the initial value of D. Since all of this is about the sign of the accumulated difference, then everything can be multiplied by 2 with no consequence

Program

```
#include<bits/stdc++.h>
#include<graphics.h>
#define ll long long int
using namespace std;
int main()
{
int gd = DETECT, gm, tmp = 0;
```

```

ll x1 = 0, x2 = 0, y1 = 0, y2 = 0, dx = 0, dy = 0, p;
cin >> x1 >> y1;
cin >> x2 >> y2;
dx = x2 - x1;
dy = y2 - y1;
//declare all variables before it
initgraph(&gd,&gm, NULL);
//draw here
if(dy > dx)
{
p = 2*(dy - dx);
for(; x1 <= x2; x1++)
{
if(p < 0)
{
p = p + 2 * dy;
}
else
{
y1++;
p = p + 2 * dy - dx;
}
putpixel(100 + x1, 100 + y1, 15);
delay(300);
}
}
else

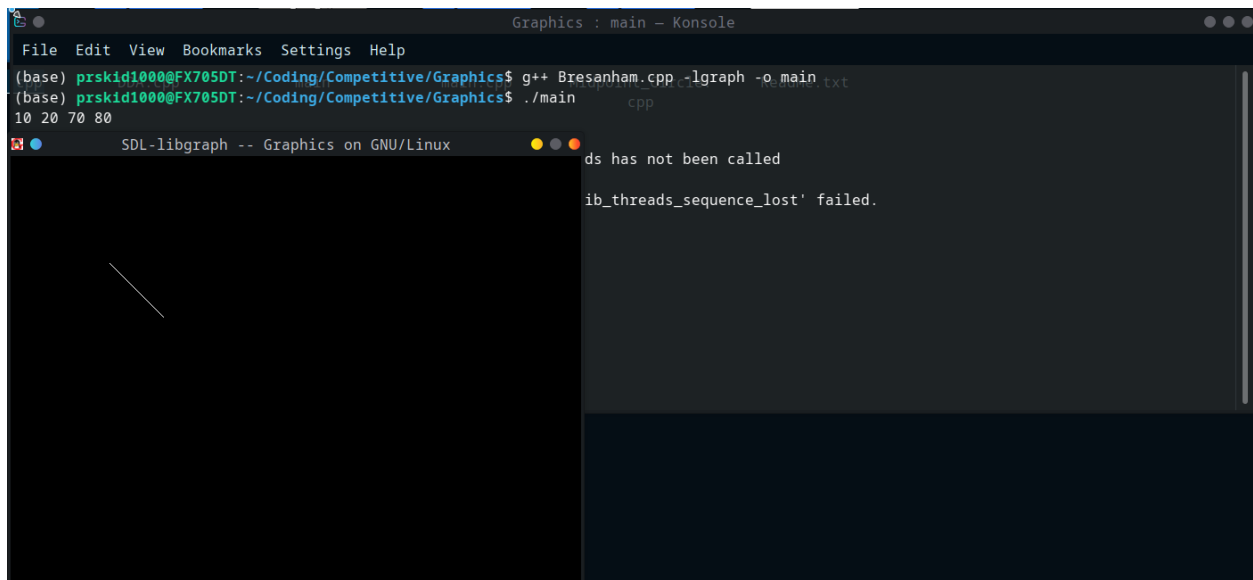
```

```

{
p = 2*(dx - dy);
for(; y1 <= y2; y1++)
{
if(p < 0)
{
p = p + 2 * dx;
}
else
{
x1++;
p = p + 2 * dx - dy;
}
putpixel(100 + x1, 100 + y1, 15);
delay(300);
}
}
//draw ends
getche();
closegraph();
return 0;
}

```


Output



```
Graphics : main - Konsole
File Edit View Bookmarks Settings Help
(base) prskid1000@FX705DT:~/Coding/Competitive/Graphics$ g++ Bresenham.cpp -lgraph -o main
(base) prskid1000@FX705DT:~/Coding/Competitive/Graphics$ ./main
10 20 70 80
SDL-libgraph -- Graphics on GNU/Linux
ds has not been called
ib_threads_sequence_lost' failed.
```

EXPERIMENT NO.3

Aim

Draw a circle using midpoint algorithm.

Theory

The function of a circle i.e.: $f_{\text{circle}}(x,y) = x^2 + y^2 - r^2$

- If $f_{\text{circle}} < 0$ then x, y is inside the circle boundary.
- If $f_{\text{circle}} > 0$ then x, y is outside the circle boundary.
- If $f_{\text{circle}} = 0$ then x, y is on the circle boundary.

$p_k = f_{\text{circle}}(x_{k+1}, y_{k-1/2})$ where p_k is a decision parameter and in this $\frac{1}{2}$ is taken because it is a midpoint value through which it is easy to calculate value of y_k and y_{k-1} . i.e. $p_k = (x_{k+1})^2 + (y_{k-1/2})^2 - r^2$

If $p_k < 0$ then midpoint is inside the circle in this condition we select y is y_k otherwise we will select next y as y_{k-1} for the condition of $p_k > 0$.

If $p_k < 0$ then $y_{k+1} = y_k$, by this the plotting points will be (x_{k+1}, y_k) . By this the value for the next point will be given as: $P_{k+1} = p_k + 2(x_{k+1}) + 1$

If $p_k > 0$ then $y_{k+1} = y_{k-1}$, by this the plotting points will be (x_{k+1}, y_{k-1}) . By this the value of the next point will be given as: $P_{k+1} = p_k + 2(x_{k+1}) + 1 - 2(y_{k+1})$

$$P_0 = f_{\text{circle}}(1, r-1/2)$$

This is taken because of $(x_0, y_0) = (0, r)$

i.e. $p_0 = 5/4 - r$ or $1 - r$, ($1 - r$ will be taken if r is integer)

Program

```
#include<bits/stdc++.h>
#include<graphics.h>
#define ll long long int
#define ld long double
using namespace std;
int main()
{
    int gd = DETECT, gm, tmp = 0;
    ld r = 0;
    cin >> r;
    ld x = 0, y = r, d = 1.25 - r;
    //declare all variables before it
    initgraph(&gd,&gm, NULL);
    //draw here
    do
    {
        putpixel(100 + x, 100 + y, 15);
        putpixel(100 - x, 100 - y, 15);
        putpixel(100 - x, 100 + y, 15);
        putpixel(100 + x, 100 - y, 15);
        putpixel(100 + y, 100 + x, 15);
        putpixel(100 - y, 100 - x, 15);
        putpixel(100 - y, 100 + x, 15);
```

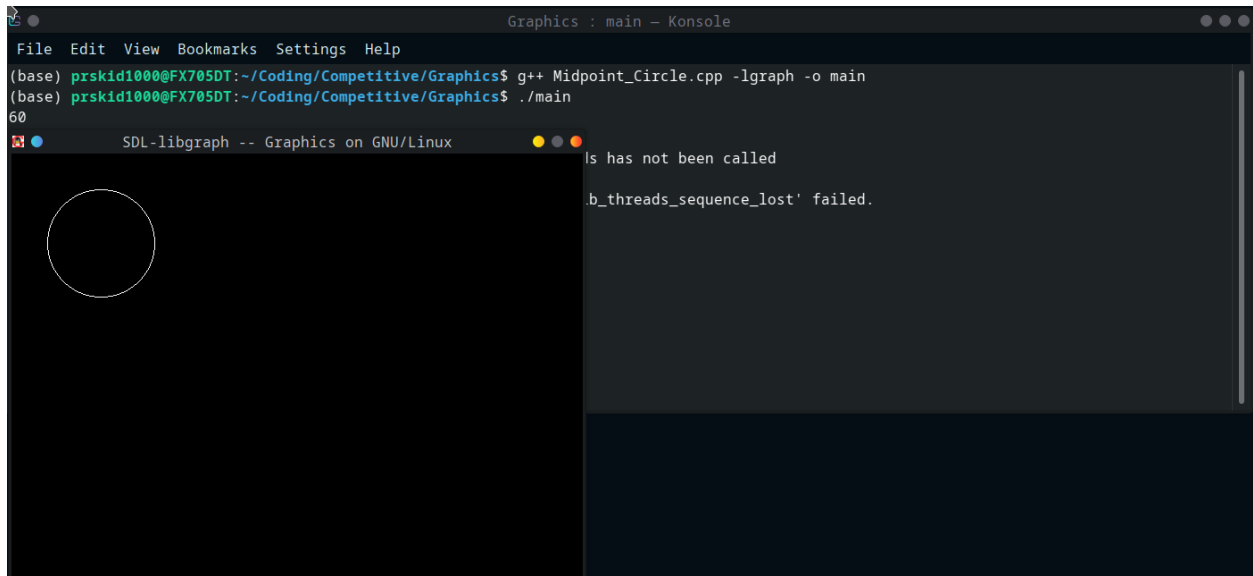
```

putpixel(100 + y, 100 - x, 15);
if(d < 0)
{
x = x + 1;
y = y;
d = d + 2 * x + 1;
}
else
{
x = x + 1;
y = y - 1;
d = d + 2 * x - 2 * y + 1;
}
delay(300);
}while(x < y);

//draw ends
getche();
closegraph();
return 0;
}

```

Output



```
Graphics : main - Konsole
File Edit View Bookmarks Settings Help
(base) prskid1000@FX705DT:~/Coding/Competitive/Graphics$ g++ Midpoint_Circle.cpp -lgraph -o main
(base) prskid1000@FX705DT:~/Coding/Competitive/Graphics$ ./main
60
SDL-libgraph -- Graphics on GNU/Linux
ls has not been called
b_threads_sequence_lost' failed.
```

EXPERIMENT NO.4

Aim

Draw a circle using Bresenham's algorithm

Theory

We know the equation of the circle is $f_c(x, y) = x^2 + y^2 = r^2$

We assume,

The distance between point P_3 and circle boundary = d_1

The distance between point P_2 and circle boundary = d_2

Now, if we select point P_3 then circle equation will be

$$d_1 = (x_k + 1)^2 + (y_k)^2 - r^2$$

If we select point P_2 then circle equation will be-

$$d_2 = (x_k + 1)^2 + (y_k - 1)^2 - r^2$$

Now, we will calculate the decision parameter $(d_k) = d_1 + d_2$

$$d_k = 2(x_k + 1)^2 + (y_k)^2 + (y_k - 1)^2 - 2r^2$$

If $d_k < 0$ then $(x_{k+1}, y_k) = (x_k + 1, y_k)$

If $d_k \geq 0$ then $(x_{k+1}, y_k) = (x_k + 1, y_k - 1)$

Now, we will find the next decision parameter (d_{k+1})

$$d_{k+1} = 2(x_{k+1} + 1)^2 + (y_{k+1})^2 + (y_{k+1} - 1)^2 - 2r^2$$

If $d_k < 0$ then $y_{k+1} = y_k$ (We select point P_3)

If $d_k \geq 0$ then $y_{k+1} = y_k - 1$ (We select point P_3)

Now, we calculate initial decision parameter (d_0)

$$d_0 = d_1 + d_2$$

$$d_0 = 3 - 2r$$

Program

```
#include<bits/stdc++.h>
#include<graphics.h>
#define ll long long int
#define ld long double
using namespace std;
int main()
{
    int gd = DETECT, gm, tmp = 0;

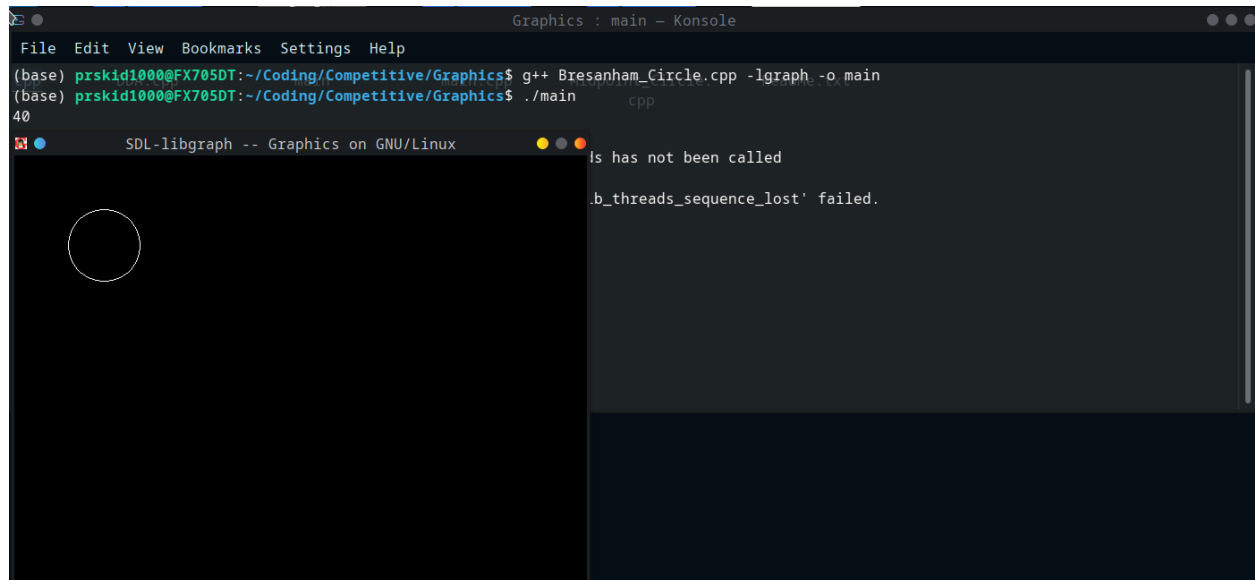
    ld r = 0;
    cin >> r;
    ld x = 0, y = r, d = 3 - 2 * r;
    //declare all variables before it
    initgraph(&gd,&gm, NULL);
    //draw here
    do
    {
        putpixel(100 + x, 100 + y, 15);
        putpixel(100 - x, 100 - y, 15);
```

```

putpixel(100 - x, 100 + y, 15);
putpixel(100 + x, 100 - y, 15);
putpixel(100 + y, 100 + x, 15);
putpixel(100 - y, 100 - x, 15);
putpixel(100 - y, 100 + x, 15);
putpixel(100 + y, 100 - x, 15);
if(d < 0)
{
x = x + 1;
y = y;
d = d + 4 * x + 6;
}
else
{
x = x + 1;
y = y - 1;
d = d + 4 * x - 4 * y - 10;
}
delay(300);
}while(x < y);
//draw ends
getche();
closegraph();
return 0;
}

```


Output



```
Graphics : main - Konsole
File Edit View Bookmarks Settings Help
(base) prskid1000@FX705DT:~/Coding/Competitive/Graphics$ g++ Bresenham_Circle.cpp -lgraph -o main
(base) prskid1000@FX705DT:~/Coding/Competitive/Graphics$ ./main
40
SDL-libgraph -- Graphics on GNU/Linux
ls has not been called
.b_threads_sequence_lost' failed.
```

EXPERIMENT NO.5

Aim

Draw an ellipse using midpoint algorithm.

Theory

Take input radius along x axis and y axis and obtain centre of ellipse.

Initially, we assume ellipse to be centred at origin and the first point as : $(x_0, y_0) = (0, r_y)$.

Obtain the initial decision parameter for region 1 as: $p_{10} = r_y^2 + 1/4 r_x^2 - r_x^2 r_y$

For every x_k position in region 1 :

- If $p_{1k} < 0$ then the next point along the is (x_{k+1}, y_k) and $p_{1k+1} = p_{1k} + 2r_y^2 x_{k+1} + r_y^2$
- Else, the next point is (x_{k+1}, y_{k-1})

$$\text{And } p_{1k+1} = p_{1k} + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_y^2$$

Obtain the initial value in region 2 using the last point (x_0, y_0) of region 1 as:

$$p_{20} = r_y^2 (x_0 + 1/2)^2 + r_x^2 (y_0 - 1)^2 - r_x^2 r_y^2$$

At each y_k in region 2 starting at $k = 0$ perform the following task.

- If $p_{2k} > 0$ the next point is (x_k, y_{k-1}) and $p_{2k+1} = p_{2k} - 2r_x^2 y_{k+1} + r_x^2$
- Else, the next point is (x_{k+1}, y_{k-1}) and $p_{2k+1} = p_{2k} + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_x^2$

Now obtain the symmetric points in the three quadrants and plot the coordinate value as: $x = x + x_c$, $y = y + y_c$

Repeat the steps for region 1 until $2r_y^2 x > 2r_x^2 y$

Program

```

#include<bits/stdc++.h>
#include<graphics.h>
#define ll long long int
#define ld long double
using namespace std;
int main()
{
    int gd = DETECT, gm, tmp = 0;

    ld dx, dy, d1, d2, x = 0, y, rx, ry, xc, yc;
    cin >> rx >> ry >> xc >> yc;
    y = ry;
    //declare all variables before it
    initgraph(&gd,&gm, NULL);
    //draw here
    d1 = (ry * ry) - (rx * rx * ry) + (0.25 * rx * rx);
    dx = 2 * ry * ry * x;
    dy = 2 * rx * rx * y;
    while (dx < dy)
    {
        putpixel(100 + x + xc, 100 + y + yc, 15);
        putpixel(100 - x + xc, 100 + y + yc, 15);
        putpixel(100 + x + xc, 100 - y + yc, 15);
        putpixel(100 - x + xc, 100 - y + yc, 15);
        if (d1 < 0)
        {
            x++;

```

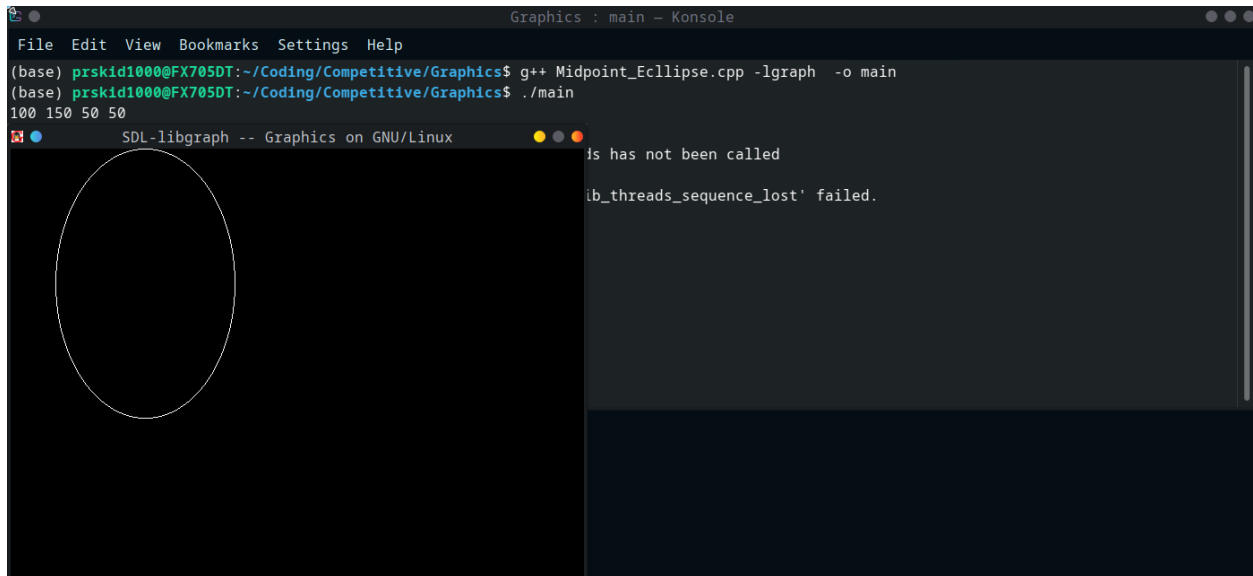
```

    dx = dx + (2 * ry * ry);
    d1 = d1 + dx + (ry * ry);
}
else
{
    x++;
    y--;
    dx = dx + (2 * ry * ry);
    dy = dy - (2 * rx * rx);
    d1 = d1 + dx - dy + (ry * ry);
}
delay(300);
}
d2 = ((ry * ry) * ((x + 0.5) * (x + 0.5))) + ((rx * rx) * ((y - 1) * (y - 1))) - (rx *
rx * ry * ry);
while (y >= 0)
{
    putpixel(100 + x + xc, 100 + y + yc, 15);
    putpixel(100 - x + xc, 100 + y + yc, 15);
    putpixel(100 + x + xc, 100 - y + yc, 15);
    putpixel(100 - x + xc, 100 - y + yc, 15);
    if (d2 > 0)
    {
        y--;
        dy = dy - (2 * rx * rx);
        d2 = d2 + (rx * rx) - dy;
    }
}

```

```
else
{
    y--;
    x++;
    dx = dx + (2 * ry * ry);
    dy = dy - (2 * rx * rx);
    d2 = d2 + dx - dy + (rx * rx);
}
delay(300);
}
//draw ends
getche();
closegraph();
return 0;
}
```

Output



```
Graphics : main - Konsole
File Edit View Bookmarks Settings Help
(base) prskid1000@FX705DT:~/Coding/Competitive/Graphics$ g++ Midpoint_Eclipse.cpp -lgraph -o main
(base) prskid1000@FX705DT:~/Coding/Competitive/Graphics$ ./main
100 150 50 50
SDL-libgraph -- Graphics on GNU/Linux
is has not been called
ib_threads_sequence_lost' failed.
```

EXPERIMENT NO.6a

Aim

Draw text using library functions

Theory

- **setcolor():** It will set the cursor colour and hence anything written on the output screen will be of the colour as per setcolor().
- **settextstyle():** It set the text font style, its orientation(horizontal/ vertical) and size of font.
- **outtextxy() :** It will print message passed to it at some certain coordinate (x,y).

Program

Note: Program was run in turboc++ inside dosbox


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

```
void printMsg()
{
```

```
int gdriver = DETECT,gmode,i;
initgraph(&gdriver,&gmode,"C:\\TC\\BGI");
for (i=3; i<7; i++)
{
setcolor(i);
settextstyle(i,0,i);
outtextxy(100,20*i,"Geeks");
delay(500);
}
delay(2000);
}
```

```
int main()
{
printMsg();
return 0;
}
```


Output

 DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip



EXPERIMENT NO.6b

Aim

Implementing boundary fill algorithm

Theory

Boundary Fill Algorithm starts at a pixel inside the polygon to be filled and paints the interior proceeding outwards towards the boundary. This algorithm works only if the color with which the region has to be filled and the color of the boundary of the region are different. If the boundary is of one single color, this approach proceeds outwards pixel by pixel until it hits the boundary of the region.

Boundary Fill Algorithm is recursive in nature. It takes an interior point(x, y), a fill color, and a boundary color as the input. The algorithm starts by checking the color of (x, y). If it's color is not equal to the fill color and the boundary color, then it is painted with the fill color and the function is called for all the neighbours of (x, y). If a point is found to be of fill color or of boundary color, the function does not call its neighbours and returns. This process continues until all points up to the boundary color for the region have been tested.

The boundary fill algorithm can be implemented by 4-connected pixels or 8-connected pixels.

Program

```
#include<bits/stdc++.h>
```

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

```

#define ll long long int

#define ld long double

//For outer loop

#define fr(i, start, stop, increment) for(i = start; i < stop; i += increment)

#define dfr(i, start, stop ,decrement) for( i = start; i >= stop; i -= decrement)

using namespace std;

void boundaryFill(ll x, ll y, ll fc, ll bc)

{

if(getpixel(x, y) != bc && getpixel(x, y) != fc)

{

putpixel(x, y, fc);

boundaryFill(x + 1, y, fc, bc);

boundaryFill(x, y + 1, fc, bc);

boundaryFill(x - 1, y, fc, bc);

boundaryFill(x, y - 1, fc, bc);

boundaryFill(x - 1, y - 1, fc, bc);

boundaryFill(x - 1, y + 1, fc, bc);

boundaryFill(x + 1, y - 1, fc, bc);

boundaryFill(x + 1, y + 1, fc, bc);

}

```

```

}

int main()

{

int gd = DETECT, gm;

ll n = 0, m = 0, i, j;

cin >> n >> m;

vector<vector<ll>> v(n, vector<ll>(m, 0));

fr(i, 0, n, 1)

{

v[i][0] = 1;

v[i][m - 1] = 1;

}

fr(i, 0, m, 1)

{

v[0][i] = 1;

v[n - 1][i] = 1;

}

//declare all variables before it

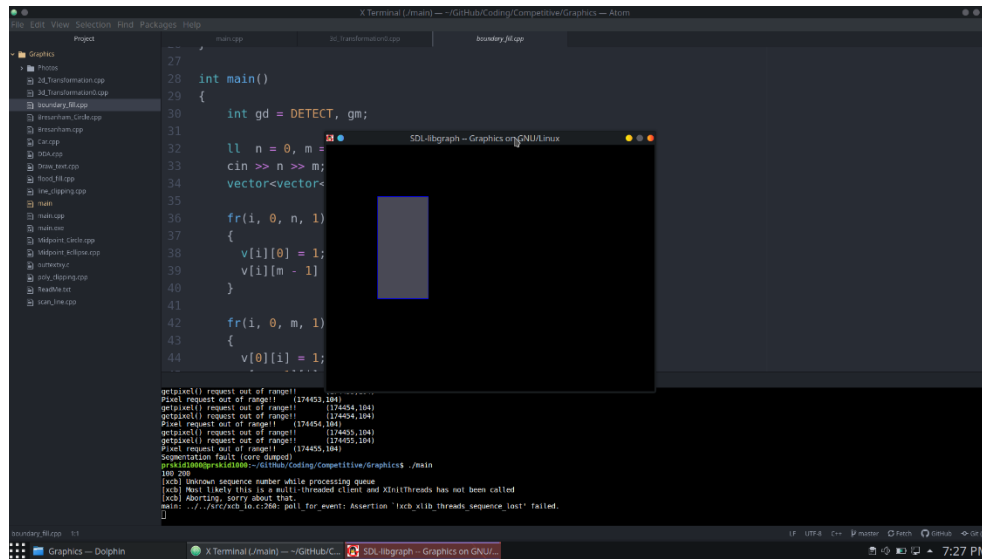
initgraph(&gd,&gm, NULL);

//draw here

```

```
fr(i, 0, n, 1){  
    fr(j, 0, m, 1)  
    {  
        putpixel(100 + i, 100 + j, v[i][j]);  
    }  
}  
  
boundaryFill(104, 104, 8, 1);  
  
//draw ends  
  
getche();  
  
closegraph();  
  
return 0;
```

Output



EXPERIMENT NO.6c

Aim

Implementing flood fill algorithm

Theory

- Take the position of the starting point.
- Decide whether you want to go in 4 directions (N, S, W, E) or 8 directions (N, S, W, E, NW, NE, SW, SE).
- Choose a replacement color and a target color.
- Travel in those directions.
- If the tile you land on is a target, replace it with the chosen color.
- Repeat 4 and 5 until you've been everywhere within the boundaries

Program

```
#include<bits/stdc++.h>
```

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

```
#define ll long long int
```

```
#define ld long double
```

```
//For outer loop
```

```
#define fr(i, start, stop, increment) for(i = start; i < stop; i += increment)
```

```

#define dfr(i, start, stop ,decrement) for( i = start; i >= stop; i -= decrement)

using namespace std;

void floodFillUtil(vector<vector<ll>> &v, ll x, ll y, ll pc, ll nc)
{
    if(x < 0 || x >= v.size() || y < 0 || y >= v[0].size()) return;
    if(v[x][y] != pc) return;
    if(v[x][y] == nc) return;
    v[x][y] = nc;
    floodFillUtil(v, x+1, y, pc, nc);
    floodFillUtil(v, x-1, y, pc, nc);
    floodFillUtil(v, x, y+1, pc, nc);
    floodFillUtil(v, x, y-1, pc, nc);
}

void floodFill(vector<vector<ll>> &v, ll x, ll y, ll nc)
{
    ll pc = v[x][y];
    floodFillUtil(v, x, y, pc, nc);
}

int main()
{

```



```

int gd = DETECT, gm;

ll n = 0, m = 0, i, j;

cin >> n >> m;

vector<vector<ll>> v(n, vector<ll>(m, 0));

fr(i, 0, n, 1)

{

v[i][0] = 2;

v[i][m - 1] = 3;

}

fr(i, 0, m, 1)

{

v[0][i] = 4;

v[n - 1][i] = 5;

}

//declare all variables before it

initgraph(&gd,&gm, NULL);

//draw here

fr(i, 0, n, 1)

{

fr(j, 0, m, 1)

```

```

{
putpixel(100 + i, 100 + j, v[i][j]);

}

}

floodFill(v, 4, 4, 8);

fr(i, 0, n, 1)

{

fr(j, 0, m, 1)

{

putpixel(100 + i, 100 + j, v[i][j]);

}

}

//draw ends

getche();

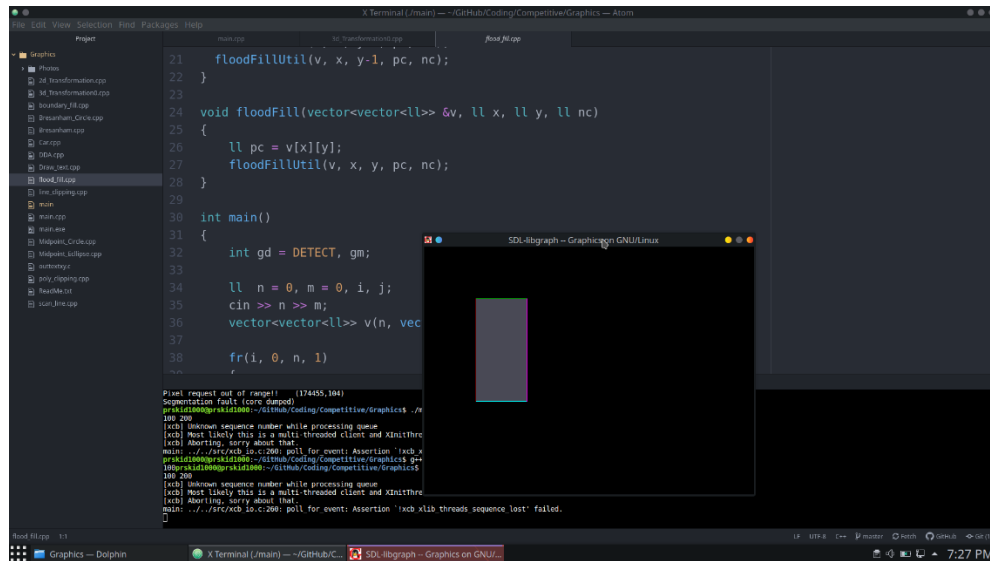
closegraph();

return 0;

}

```

Output



EXPERIMENT NO.6d

Aim

Implementing scan line algorithm

Theory

It is an image space algorithm. It processes one line at a time rather than one pixel at a time. It uses the concept area of coherence. This algorithm records edge list, active edge list. So accurate bookkeeping is necessary. The edge list or edge table contains the coordinate of two endpoints. Active Edge List (AEL) contain edges a given scan line intersects during its sweep. The active edge list (AEL) should be sorted in increasing order of x. The AEL is dynamic, growing and shrinking.

- Start algorithm
- Initialize the desired data structure
- Create a polygon table having color, edge pointers, coefficients. Establish edge table contains information regarding, the endpoint of edges, pointer to polygon, inverse slope. Create Active edge list. This will be sorted in increasing order of x. Create a flag F. It will have two values either on or off.
- Perform the following steps for all scan lines
 - i) Enter values in Active edge list (AEL) in sorted order using y as value
 - ii) Scan until the flag, i.e. F is on using a background color.
 - iii) When one polygon flag is on, and this is for surface S1 enter color intensity as I1 into refresh buffer
 - iv) When two or image surface flag are on, sort the surfaces according to depth and use intensity value S_n for the nth surface. This surface will have least z depth value

- v) Use the concept of coherence for remaining planes.
- Stop Algorithm

Program

```
#include <bits/stdc++.h>

#include <graphics.h>

using namespace std;

class point

{

public:

int x,y;

};

class poly

{

private:

point p[20];

int inter[20],x,y;

int v,xmin,ymin,xmax,ymax;

public:

int c;
```

```

void read();

void calcs();

void display();

void ints(float);

void sort(int);

};

void poly::read()

{

int i;

cout<<"\n Enter the no of vertices of polygon:";

cin>>v;

if(v>2)

{

for(i=0;i<v; i++)

{

cout<<"\nEnter the co-ordinate no.- "<<i+1<<" : ";

cout<<"\n\tx"<<(i+1)<<"=";

cin>>p[i].x;

cout<<"\n\ty"<<(i+1)<<"=";

cin>>p[i].y;

```

```

    }

    p[i].x=p[0].x;

    p[i].y=p[0].y;

    xmin=xmax=p[0].x;

    ymin=ymax=p[0].y;

    }

    else

    cout<<"\n Enter valid no. of vertices.";

    }

    void poly::calcs()

    {

    for(int i=0;i<v;i++)

    {

    if(xmin>p[i].x)

    xmin=p[i].x;

    if(xmax<p[i].x)

    xmax=p[i].x;

    if(ymin>p[i].y)

    ymin=p[i].y;

    if(ymax<p[i].y)

```

```

ymax=p[i].y;

}

}

void poly::display()

{

int ch1;

char ch='y';

float s,s2;

do

{

cout<<"\n\nMENU:";

cout<<"\n\n\t1 . Scan line Fill ";

cout<<"\n\n\t2 . Exit ";

cout<<"\n\nEnter your choice:";

cin>>ch1;

switch(ch1)

{

case 1:

s=ymin+0.01;

delay(100);

```



```

cleardevice();

while(s<=ymax)

{

ints(s);

sort(s);

s++;

}

break;

case 2:

exit(0);

}

cout<<"Do you want to continue?: ";

cin>>ch;

}while(ch=='y' || ch=='Y');

}

void poly::ints(float z)

{

int x1,x2,y1,y2,temp;

c=0;

for(int i=0;i<v;i++)

```

```
{  
x1=p[i].x;  
y1=p[i].y;  
x2=p[i+1].x;  
y2=p[i+1].y;  
if(y2<y1)  
{  
temp=x1;  
x1=x2;  
x2=temp;  
temp=y1;  
y1=y2;  
y2=temp;  
}  
if(z<=y2&& z>=y1)  
{  
if((y1-y2)==0)  
x=x1;  
else  
{
```

```

x=((x2-x1)*(z-y1))/(y2-y1);

x=x+x1;

}

if(x<=xmax && x>=xmin)

inter[c++]=x;

}

}

}

void poly::sort(int z)

{

int temp,j,i;

for(i=0;i<v;i++)

{

line(p[i].x,p[i].y,p[i+1].x,p[i+1].y);

}

delay(100);

for(i=0; i<c;i+=2)

{

delay(100);

line(inter[i],z,inter[i+1],z); // Used to fill the polygon ....

```

```

}

}

int main()

{

int gd=DETECT,gm;

detectgraph(&gd,&gm);

initgraph(&gd,&gm,NULL);

int cl;

poly x;

x.read();

x.calcs();

cout<<"\n\tEnter the colour u want:(0-15)->"; //Selecting colour

cin>>cl;

setcolor(cl);

x.display();

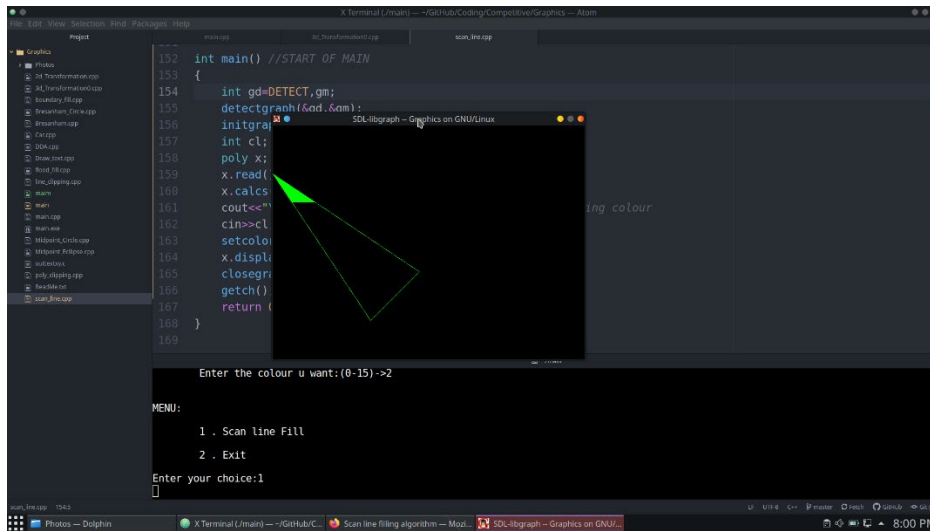
closegraph();

getch();

return 0;}

```

Output

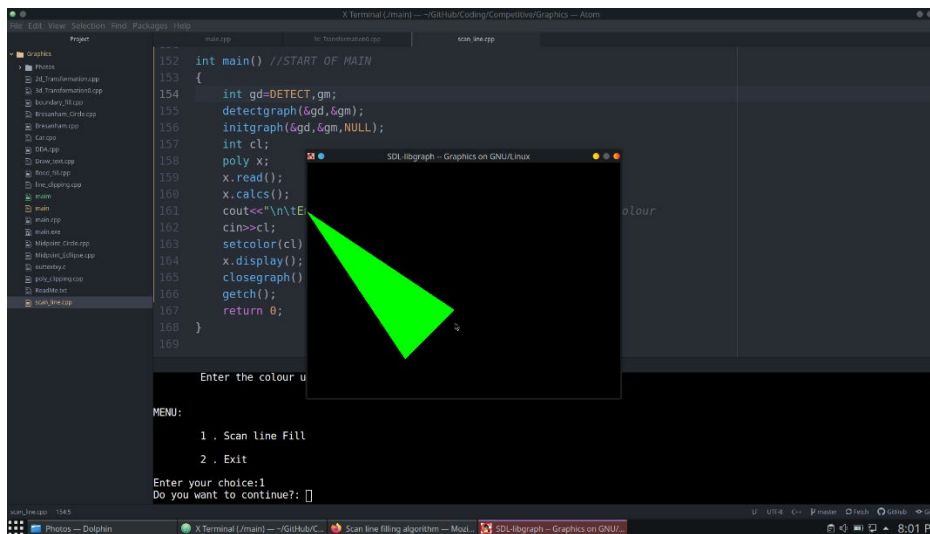


The screenshot shows a C++ IDE with a project named "SDL-ibgraph - Graphics on GNU/Linux". The code in `main.cpp` is as follows:

```
152 int main() //START OF MAIN
153 {
154     int gd=DETECT,gm;
155     detectgraph(&gd,&gm);
156     initgraph(&gd,&gm);
157     int cl;
158     poly x;
159     x.read();
160     x.calcs();
161     cout<<"\nTE
162     cins>>cl;
163     setcolor(cl);
164     x.display();
165     closegraph();
166     getch();
167     return 0;
168 }
```

The program prompts the user to "Enter the colour u want:(0-15)->2". The user enters "2", and the program displays a green triangle on the screen. The menu options are:

```
MENU:
1 . Scan line Fill
2 . Exit
Enter your choice:1
```



The screenshot shows the same C++ IDE with the same code in `main.cpp`. The program prompts the user to "Enter the colour u want:(0-15)->2". The user enters "2", and the program displays a green triangle on the screen. The menu options are:

```
MENU:
1 . Scan line Fill
2 . Exit
Enter your choice:1
Do you want to continue?:
```

EXPERIMENT NO.7

Aim

Write a program for performing the basic 2D transformations such as translation, scaling, rotation, shearing and reflection for 2D object.

Theory

Algorithm for scaling

- For each point P (x, y) do the following

$$\begin{matrix} x' & Sx & 0 & 0 & x \\ y' & 0 & Sy & 0 & x & y \\ 1 & 0 & 0 & 1 & 1 \end{matrix}$$

- Then finally plot the lines joining those points

Algorithm for Rotation

- For each point P (x, y) do the following

$$\begin{matrix} x' & \cos(a) & -\sin(a) & 0 & x \\ y' & \sin(a) & \cos(a) & 0 & x & y \\ 1 & 0 & 0 & 1 & 1 \end{matrix}$$

- Then finally plot the lines joining those points

Algorithm for Shearing

- For each point P (x, y) do the following

$$\begin{matrix} x' & 1 & Sx & 0 & x \\ y' & Sy & 1 & 0 & x & y \\ 1 & 0 & 0 & 1 & 1 \end{matrix}$$

- Then finally plot the lines joining those points

Algorithm for Reflection

- For each point P (x, y) do the following

$$\begin{matrix} x' & rx & 0 & 0 & x \\ y' & 0 & ry & 0 & x & y \\ 1 & 0 & 0 & 1 & 1 \end{matrix}$$

- Then finally plot the lines joining those points

Algorithm for Translation

- For each point P (x, y) do the following

$$\begin{matrix} x' & 1 & 0 & Tx & x \\ y' & 0 & 1 & Ty & x & y \\ 1 & 0 & 0 & 1 & 1 \end{matrix}$$

- Then finally plot the lines joining those points

Program

Write a program

```
#include<bits/stdc++.h>
```

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

```
#define ll long long int
```

```
#define ld long double
```

```
using namespace std;
```

```
void tranlate(float x1, float y1, float x2, float y2, float x3, float y3, float tx, float ty)
```

```
{
```

```

float mat[3][3] = {{1, 0, tx},{0, 1, ty},{0, 0, 1}};

float pt[3][3] = {{x1, y1, 1},{x2, y2, 1},{x3, y3, 1}};

float npt[3][3] = {{0, 0, 0},{0, 0, 0},{0, 0, 0}};

for(int k = 0; k < 3; k++)

{

for(int i = 0; i < 3; i++)

{

for(int j = 0; j < 3; j++)

{

npt[k][i] += mat[i][j] * pt[k][j];

}

}

}

for(int i = 0; i < 3; i++)

{

line(100 + pt[i][0], 100 + pt[i][1], 100 + pt[(i + 1) % 3][0], 100 + pt[(i + 1) %
3][1]);

}

for(int i = 0; i < 3; i++)

{

```



```

line(300 + npt[i][0], 300 + npt[i][1], 300 + npt[(i + 1) % 3][0], 300 + npt[(i + 1) %
3][1]);

}

}

void rotate(float x1, float y1, float x2, float y2, float x3, float y3, float ang)

{

float a = (3.14 * ang) / 180;

float mat[3][3] = {{cos(a), -sin(a), 0}, {sin(a), cos(a), 0}, {0, 0, 1}};

float pt[3][3] = {{x1, y1, 1},{x2, y2, 1},{x3, y3, 1}};

float npt[3][3] = {{0, 0, 0},{0, 0, 0},{0, 0, 0}};

for(int k = 0; k < 3; k++)

{

for(int i = 0; i < 3; i++)

{

for(int j = 0; j < 3; j++)

{

npt[k][i] += mat[i][j] * pt[k][j];

}

}

}

```

```

for(int i = 0; i < 3; i++)

{

line(100 + pt[i][0], 100 + pt[i][1], 100 + pt[(i + 1) % 3][0], 100 + pt[(i + 1) %
3][1]);

}

for(int i = 0; i < 3; i++)

{

line(300 + npt[i][0], 300 + npt[i][1], 300 + npt[(i + 1) % 3][0], 300 + npt[(i + 1) %
3][1]);

}

}

void scale(float x1, float y1, float x2, float y2, float x3, float y3, float sx, float sy)

{

float mat[3][3] = {{sx, 0, 0},{0, sy, 0},{0, 0, 1}};

float pt[3][3] = {{x1, y1, 1},{x2, y2, 1},{x3, y3, 1}};

float npt[3][3] = {{0, 0, 0},{0, 0, 0},{0, 0, 0}};

for(int k = 0; k < 3; k++)

{

for(int i = 0; i < 3; i++)

{

for(int j = 0; j < 3; j++)

```

```

{
npt[k][i] += mat[i][j] * pt[k][j];
}
}
}

for(int i = 0; i < 3; i++)
{
line(200 + pt[i][0], 200 + pt[i][1], 200 + pt[(i + 1) % 3][0], 200 + pt[(i + 1) %
3][1]);
}

for(int i = 0; i < 3; i++)
{
cout << 100 + npt[i][0] << " " << 100 + npt[i][1] << " " << 100 + npt[(i + 1) %
3][0] << " " << 100 + npt[(i + 1) % 3][1] << "\n";
line(300 + npt[i][0], 300 + npt[i][1], 300 + npt[(i + 1) % 3][0], 300 + npt[(i + 1) %
3][1]);
}
}

void reflect(float x1, float y1, float x2, float y2, float x3, float y3, float rx, float ry)
{

```

```

float mat[3][3] = {{rx, 0, 0},{0, ry, 0},{0, 0, 1}};

float pt[3][3] = {{x1, y1, 1},{x2, y2, 1},{x3, y3, 1}};

float npt[3][3] = {{0, 0, 0},{0, 0, 0},{0, 0, 0}};

for(int k = 0; k < 3; k++)

{

for(int i = 0; i < 3; i++)

{

for(int j = 0; j < 3; j++)

{

npt[k][i] += mat[i][j] * pt[k][j];

}

}

}

for(int i = 0; i < 3; i++)

{

line(100 + pt[i][0], 100 + pt[i][1], 100 + pt[(i + 1) % 3][0], 100 + pt[(i + 1) %
3][1]);

}

for(int i = 0; i < 3; i++)

{

```

```

line(300 + npt[i][0], 300 + npt[i][1], 300 + npt[(i + 1) % 3][0], 300 + npt[(i + 1) %
3][1]);

}

}

void shear(float x1, float y1, float x2, float y2, float x3, float y3, float sx, float sy)
{
float mat[3][3] = {{1, sx, 0},{sy, 1, 0},{0, 0, 1}};

float pt[3][3] = {{x1, y1, 1},{x2, y2, 1},{x3, y3, 1}};

float npt[3][3] = {{0, 0, 0},{0, 0, 0},{0, 0, 0}};

for(int k = 0; k < 3; k++)
{
for(int i = 0; i < 3; i++)
{
for(int j = 0; j < 3; j++)
{
npt[k][i] += mat[i][j] * pt[k][j];
}
}
}

for(int i = 0; i < 3; i++)

```

```

{
line(100 + pt[i][0], 100 + pt[i][1], 100 + pt[(i + 1) % 3][0], 100 + pt[(i + 1) %
3][1]);
}

for(int i = 0; i < 3; i++)

{
line(300 + npt[i][0], 300 + npt[i][1], 300 + npt[(i + 1) % 3][0], 300 + npt[(i + 1) %
3][1]);
}

}

int main()

{

int gd = DETECT, gm, tmp = 0;

cout << "Enter the points = " << "\n";

float x1, y1, x2, y2, x3, y3, sx, sy, rx, ry, tx, ty, ang;

cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3;

int ch = 0;

cout << "Enter:" << "\n";

cout << "1: Scaling" << "\n";

cout << "2: Rotation" << "\n";

cout << "3: Translation" << "\n";

```

```
cout << "4: Shearing" << "\n";

cout << "5: Reflection" << "\n";

cin >> ch;

switch(ch)

{

case 1:

cin >> sx >> sy;

break;

case 2:

cin >> ang;

break;

case 3:

cin >> tx >> ty;

break;

case 4:

cin >> sx >> sy;

break;

case 5:

cin >> rx >> ry;
```

```
break;

default:

cout << "Invalid Choice\n";

}

//declare all variables before it

initgraph(&gd,&gm, NULL);

//draw here

switch(ch)

{

case 1:

scale(x1, y1, x2, y2, x3, y3, sx, sy);

break;

case 2:

rotate(x1, y1, x2, y2, x3, y3, ang);

break;

case 3:

tranlate(x1, y1, x2, y2, x3, y3, tx, ty);

break;

case 4:

shear(x1, y1, x2, y2, x3, y3, sx, sy);
```



```
break;

case 5:

reflect(x1, y1, x2, y2, x3, y3, rx, ry);

break;

default:

cout << "";

}

//draw ends

getche();

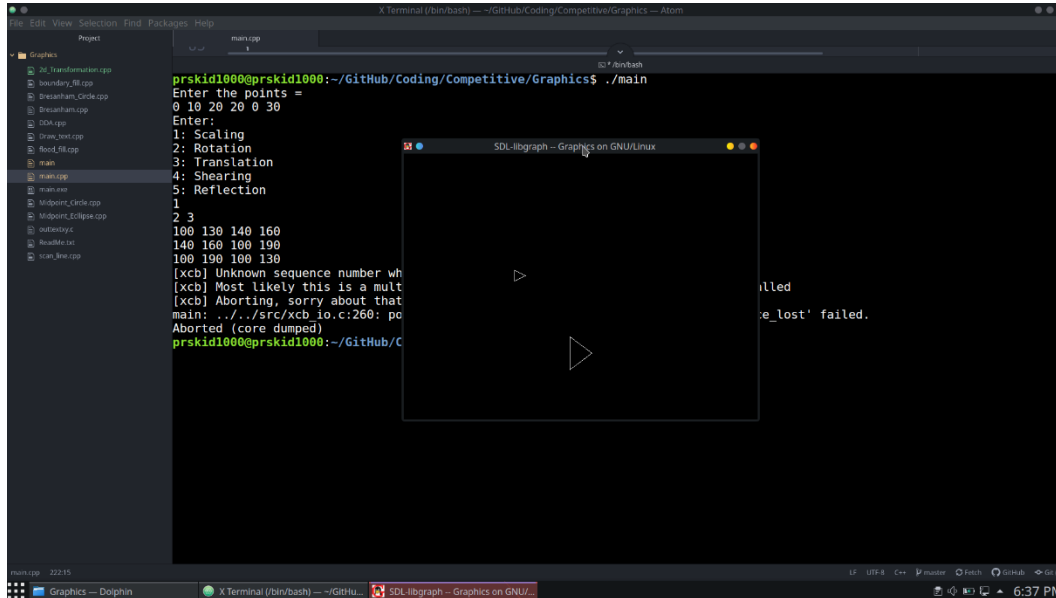
closegraph();

return 0;

}
```

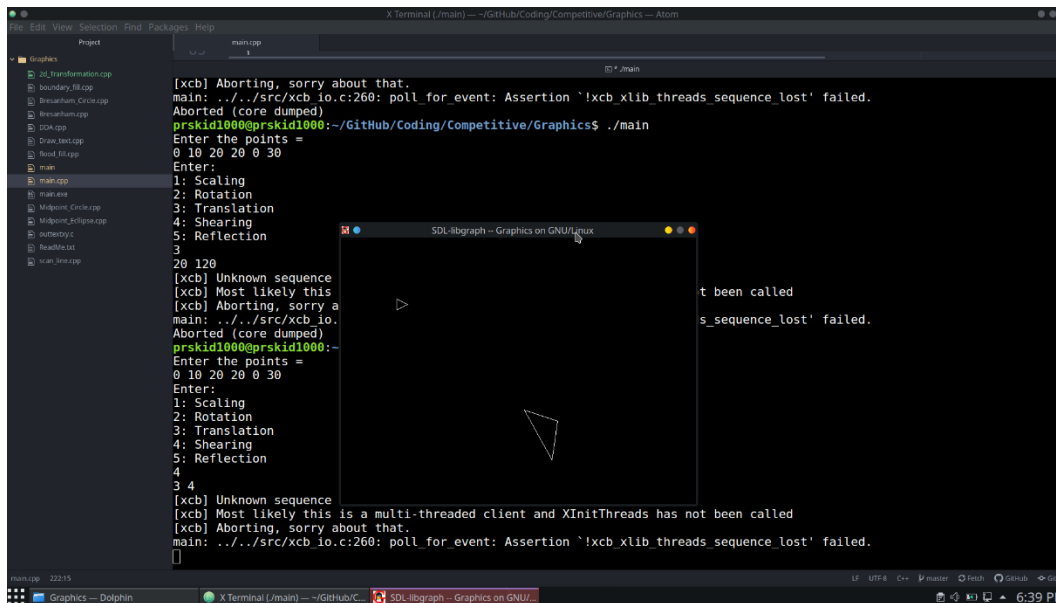
Output

Scaling



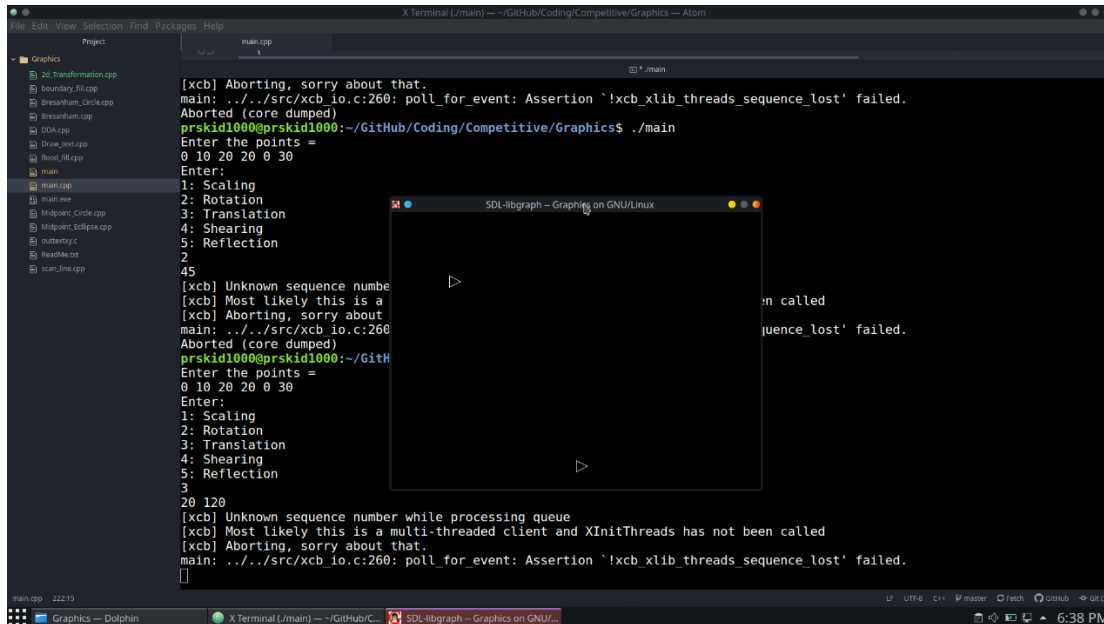
```
X Terminal (/bin/bash) -- ~/GitHub/Coding/Competitive/Graphics -- Atom
File Edit View Selection Find Packages Help
Project
main.cpp
Graphics
2d_Transformation.cpp
boundary_fill.cpp
Bresenham_Circle.cpp
Bresenham_Line.cpp
DDA.cpp
Draw_Text.cpp
Flood_Fill.cpp
main.cpp
main.exe
Midpoint_Circle.cpp
Midpoint_Ellipse.cpp
outlet.txt
ReadMe.txt
scan_line.cpp
prskid1000@prskid1000:~/GitHub/Coding/Competitive/Graphics$ ./main
Enter the points =
0 10 20 20 0 30
Enter:
1: Scaling
2: Rotation
3: Translation
4: Shearing
5: Reflection
1
100 130 140 160
140 160 100 190
100 190 100 130
[xcb] Unknown sequence number wh
[xcb] Most likely this is a mult
[xcb] Aborting, sorry about that
main: ../../src/xcb_io.c:260: po
Aborted (core dumped)
prskid1000@prskid1000:~/GitHub/Coding/Competitive/Graphics$
```

Shearing



```
X Terminal (/main) -- ~/GitHub/Coding/Competitive/Graphics -- Atom
File Edit View Selection Find Packages Help
Project
main.cpp
Graphics
2d_Transformation.cpp
boundary_fill.cpp
Bresenham_Circle.cpp
Bresenham_Line.cpp
DDA.cpp
Draw_Text.cpp
Flood_Fill.cpp
main.cpp
main.exe
Midpoint_Circle.cpp
Midpoint_Ellipse.cpp
outlet.txt
ReadMe.txt
scan_line.cpp
prskid1000@prskid1000:~/GitHub/Coding/Competitive/Graphics$ ./main
Enter the points =
0 10 20 20 0 30
Enter:
1: Scaling
2: Rotation
3: Translation
4: Shearing
5: Reflection
4
20 120
[xcb] Unknown sequence
[xcb] Most likely this is a multi-threaded client and XInitThreads has not been called
[xcb] Aborting, sorry about that
main: ../../src/xcb_io.c:260: poll_for_event: Assertion '!xcb_xlib_threads_sequence_lost' failed.
prskid1000@prskid1000:~/GitHub/Coding/Competitive/Graphics$
```

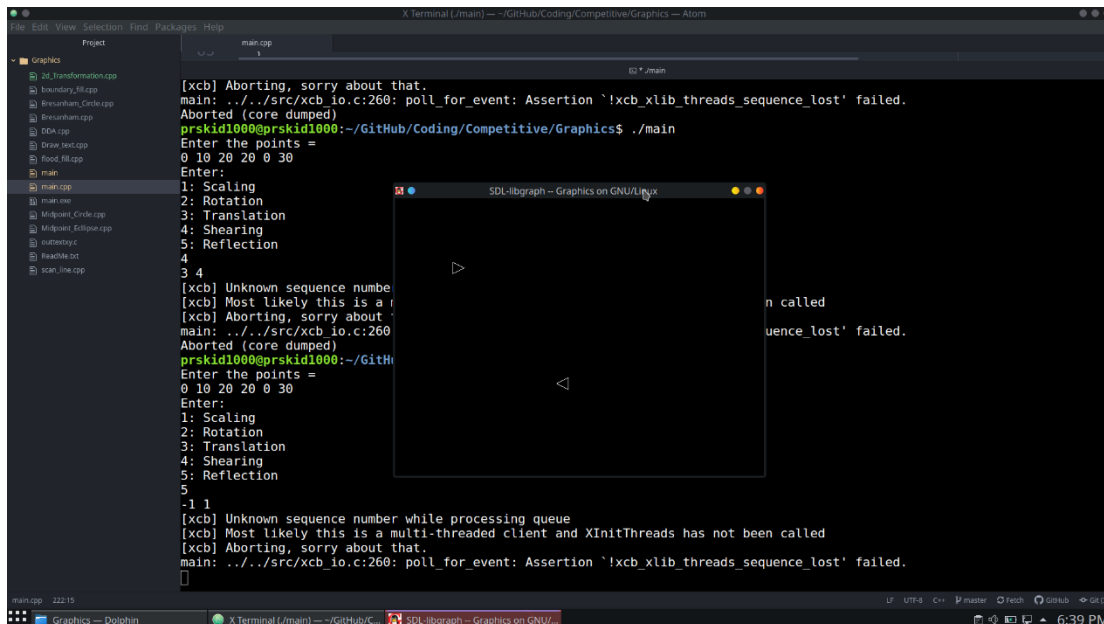
Translation



The screenshot shows a terminal window with a file explorer on the left. The terminal output shows a program crash with the message: [xcb] Aborting, sorry about that. main: ../../src/xcb_io.c:260: poll_for_event: Assertion '!xcb_xlib_threads_sequence_lost' failed. Aborted (core dumped). The user then runs ./main and enters the points = 0 10 20 20 0 30. The program then enters a loop of transformations: 1: Scaling, 2: Rotation, 3: Translation, 4: Shearing, 5: Reflection. The program then crashes again with the same error message. A graphics window titled 'SDL-ibgraph - Graphics on GNU/Linux' is visible in the background, showing a simple coordinate system with a triangle.

```
[xcb] Aborting, sorry about that.
main: ../../src/xcb_io.c:260: poll_for_event: Assertion '!xcb_xlib_threads_sequence_lost' failed.
Aborted (core dumped)
prskid1000@prskid1000:~/GitHub/Coding/Competitive/Graphics$ ./main
Enter the points =
0 10 20 20 0 30
Enter:
1: Scaling
2: Rotation
3: Translation
4: Shearing
5: Reflection
2
45
[xcb] Unknown sequence number
[xcb] Most likely this is a
[xcb] Aborting, sorry about
main: ../../src/xcb_io.c:260: poll_for_event: Assertion '!xcb_xlib_threads_sequence_lost' failed.
Aborted (core dumped)
prskid1000@prskid1000:~/GitHub/Coding/Competitive/Graphics$ ./main
Enter the points =
0 10 20 20 0 30
Enter:
1: Scaling
2: Rotation
3: Translation
4: Shearing
5: Reflection
3
20 120
[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.
main: ../../src/xcb_io.c:260: poll_for_event: Assertion '!xcb_xlib_threads_sequence_lost' failed.
Aborted (core dumped)
```

Reflection



The screenshot shows a terminal window with a file explorer on the left. The terminal output shows a program crash with the message: [xcb] Aborting, sorry about that. main: ../../src/xcb_io.c:260: poll_for_event: Assertion '!xcb_xlib_threads_sequence_lost' failed. Aborted (core dumped). The user then runs ./main and enters the points = 0 10 20 20 0 30. The program then enters a loop of transformations: 1: Scaling, 2: Rotation, 3: Translation, 4: Shearing, 5: Reflection. The program then crashes again with the same error message. A graphics window titled 'SDL-ibgraph - Graphics on GNU/Linux' is visible in the background, showing a simple coordinate system with a triangle.

```
[xcb] Aborting, sorry about that.
main: ../../src/xcb_io.c:260: poll_for_event: Assertion '!xcb_xlib_threads_sequence_lost' failed.
Aborted (core dumped)
prskid1000@prskid1000:~/GitHub/Coding/Competitive/Graphics$ ./main
Enter the points =
0 10 20 20 0 30
Enter:
1: Scaling
2: Rotation
3: Translation
4: Shearing
5: Reflection
4
3 4
[xcb] Unknown sequence number
[xcb] Most likely this is a
[xcb] Aborting, sorry about
main: ../../src/xcb_io.c:260: poll_for_event: Assertion '!xcb_xlib_threads_sequence_lost' failed.
Aborted (core dumped)
prskid1000@prskid1000:~/GitHub/Coding/Competitive/Graphics$ ./main
Enter the points =
0 10 20 20 0 30
Enter:
1: Scaling
2: Rotation
3: Translation
4: Shearing
5: Reflection
5
-1 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.
main: ../../src/xcb_io.c:260: poll_for_event: Assertion '!xcb_xlib_threads_sequence_lost' failed.
Aborted (core dumped)
```

EXPERIMENT NO.8

Aim

Write a program to create two-dimensional object car as shown in figure.

Theory

- Use **line()**, **arc()** and **circle()** function to draw the car

Program

```
#include<bits/stdc++.h>

#include<graphics.h>

#define ll long long int

#define ld long double

using namespace std;

int main()

{

int gd = DETECT, gm, tmp = 0;

//declare all variables before it

initgraph(&gd,&gm, NULL);

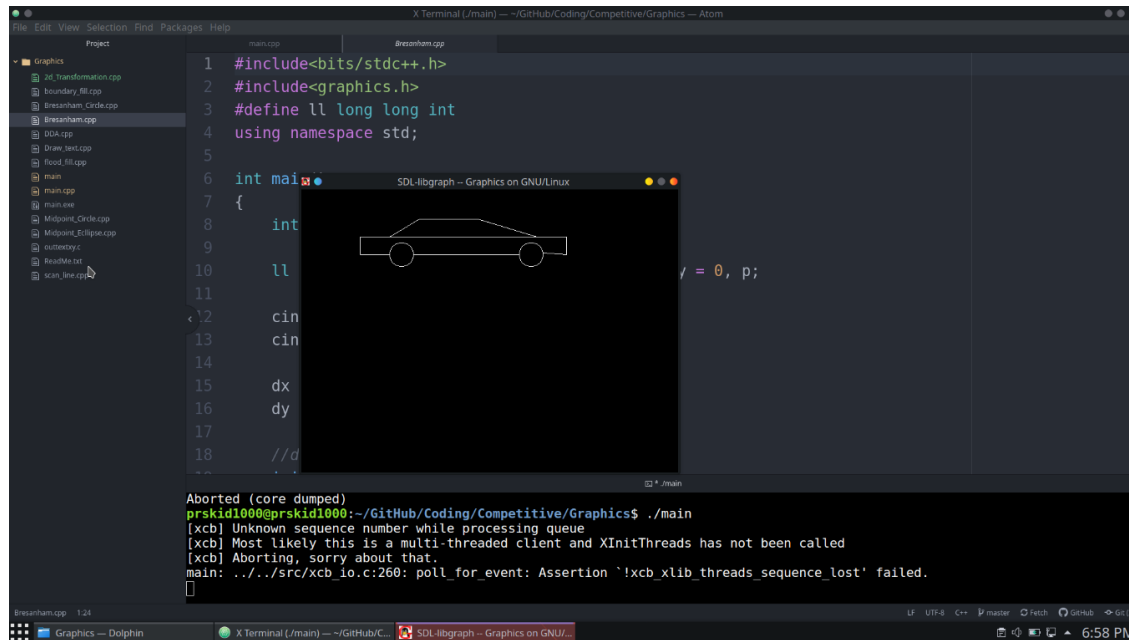
//draw here

line(200,50, 300, 50);

line(200, 50, 150, 80);
```

```
line(300, 50, 400, 80);  
  
line(100, 80, 450, 80);  
  
line(100, 80, 100, 110);  
  
line(450, 80, 450, 110);  
  
circle(170, 110, 20);  
  
circle(390, 110, 20);  
  
line(100, 110, 150, 110);  
  
line(190, 110, 370, 110);  
  
line(410, 107, 450, 110);  
  
//draw ends  
  
getche();  
  
closegraph();  
  
return 0;  
  
}
```

Output



```
1 #include<bits/stdc++.h>
2 #include<graphics.h>
3 #define ll long long int
4 using namespace std;
5
6 int main
7 {
8     int
9
10    ll
11
12    cin
13    cin
14
15    dx
16    dy
17
18    //d
19
20    / = 0, p;
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

Aborted (core dumped)

prskid1000@prskid1000:~/Github/Coding/Competitive/Graphics\$./main

[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.

main: ../../src/xcb_io.c:260: poll_for_event: Assertion '!xcb_xlib_threads_sequence_lost' failed.

EXPERIMENT NO.9

Aim

Implementation of Line Clipping using Cohen - Sutherland Algorithm.

Theory

Nine regions are created, eight "outside" regions and one "inside" region.

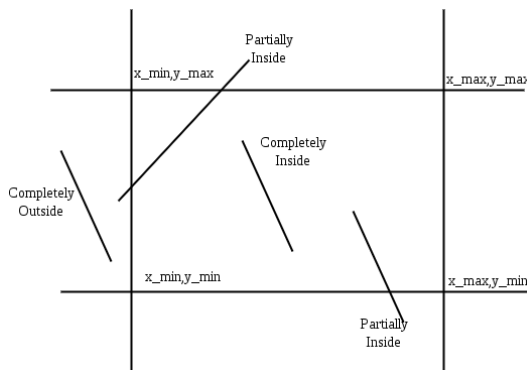
For a given line extreme point (x, y), we can quickly find its region's four-bit code. Four-bit code can be computed by comparing x and y with four values (x_min, x_max, y_min and y_max).

- If x is less than x_min then bit number 1 is set.
- If x is greater than x_max then bit number 2 is set.
- If y is less than y_min then bit number 3 is set.
- If y is greater than y_max then bit number 4 is set

There are three possible cases for any given line.

- Completely inside the given rectangle: Bitwise OR of region of two end points of line is 0 (Both points are inside the rectangle)
- Completely outside the given rectangle: Both endpoints share at least one outside region which implies that the line does not cross the visible region. (bitwise AND of endpoints! = 0).

- Partially inside the window: Both endpoints are in different regions. In this case, the algorithm finds one of the two points that is outside the rectangular region. The intersection of the line from outside point and rectangular window becomes new corner point and the algorithm repeats



- Assign a region code for two endpoints of given line.
- If both endpoints have a region code 0000 then given line is completely inside.
- Else, perform the logical AND operation for both region codes.
 - If the result is not 0000, then given line is completely outside.
 - Else line is partially inside.
 - Choose an endpoint of the line that is outside the given rectangle.
 - Find the intersection point of the rectangular boundary (based on region code)
 - Replace endpoint with the intersection and update the region code.
 - Repeat step 2 until we find a clipped line either trivially accepted or trivially rejected.
 - Repeat step 1 for other lines

Program

```
#include <bits/stdc++.h>

#include <graphics.h>

using namespace std;

int xmin, xmax, ymin, ymax;

struct lines {

int x1, y1, x2, y2;

};

int sign(int x)

{

if (x > 0)

return 1;

else

return 0;

}

void clip(struct lines mylines)

{

int bits[4], bite[4], i, var;

setcolor(RED);

bits[0] = sign(xmin - mylines.x1);
```

```

bite[0] = sign(xmin - mylines.x2);
bits[1] = sign(mylines.x1 - xmax);
bite[1] = sign(mylines.x2 - xmax);
bits[2] = sign(ymin - mylines.y1);
bite[2] = sign(ymin - mylines.y2);
bits[3] = sign(mylines.y1 - ymax);
bite[3] = sign(mylines.y2 - ymax);
string initial = "", end = "", temp = "";
for (i = 0; i < 4; i++) {
    if (bits[i] == 0)
        initial += '0';
    else
        initial += '1';
}
for (i = 0; i < 4; i++) {
    if (bite[i] == 0)
        end += '0';
    else
        end += '1';
}

```

```

float m = (mylines.y2 - mylines.y1) / (float)(mylines.x2 - mylines.x1);

float c = mylines.y1 - m * mylines.x1;

if (initial == end && end == "0000") {

line(mylines.x1, mylines.y1, mylines.x2, mylines.y2);

return;

}

else {

for (i = 0; i < 4; i++) {

int val = (bits[i] & bite[i]);

if (val == 0)

temp += '0';

else

temp += '1';

}

if (temp != "0000")

return;

for (i = 0; i < 4; i++) {

if (bits[i] == bite[i])

continue;

if (i == 0 && bits[i] == 1) {

```

```
var = round(m * xmin + c);  
  
mylines.y1 = var;  
  
mylines.x1 = xmin;  
  
}  
  
if (i == 0 && bite[i] == 1) {  
  
var = round(m * xmin + c);  
  
mylines.y2 = var;  
  
mylines.x2 = xmin;  
  
}  
  
if (i == 1 && bits[i] == 1) {  
  
var = round(m * xmax + c);  
  
mylines.y1 = var;  
  
mylines.x1 = xmax;  
  
}  
  
if (i == 1 && bite[i] == 1) {  
  
var = round(m * xmax + c);  
  
mylines.y2 = var;  
  
mylines.x2 = xmax;  
  
}  
  
if (i == 2 && bits[i] == 1) {
```

```

var = round((float)(ymin - c) / m);

mylines.y1 = ymin;

mylines.x1 = var;

}

if (i == 2 && bite[i] == 1) {

var = round((float)(ymin - c) / m);

mylines.y2 = ymin;

mylines.x2 = var;

}

if (i == 3 && bits[i] == 1) {

var = round((float)(ymax - c) / m);

mylines.y1 = ymax;

mylines.x1 = var;

}

if (i == 3 && bite[i] == 1) {

var = round((float)(ymax - c) / m);

mylines.y2 = ymax;

mylines.x2 = var;

}

bits[0] = sign(xmin - mylines.x1);

```

```

bite[0] = sign(xmin - mylines.x2);
bits[1] = sign(mylines.x1 - xmax);
bite[1] = sign(mylines.x2 - xmax);
bits[2] = sign(ymin - mylines.y1);
bite[2] = sign(ymin - mylines.y2);
bits[3] = sign(mylines.y1 - ymax);
bite[3] = sign(mylines.y2 - ymax);
}

initial = "", end = "";

for (i = 0; i < 4; i++) {

if (bits[i] == 0)

initial += '0';

else

initial += '1';

}

for (i = 0; i < 4; i++) {

if (bite[i] == 0)

end += '0';

else

end += '1';

```

```

}

if (initial == end && end == "0000") {

line(mylines.x1, mylines.y1, mylines.x2, mylines.y2);

return;

}

else

return;

}

}

int main()

{

int gd = DETECT, gm;

xmin = 40;

xmax = 100;

ymin = 40;

ymax = 80;

for(int i = 0; i < 4; i++)

{

int x1, y1, x2, y2;

cin >> x1 >> y1 >> x2 >> y2;

```

```

mylines[i].x1 = x1;

mylines[i].y1 = y1;

mylines[i].x2 = x2;

mylines[i].y2 = y2;

}

initgraph(&gd, &gm, NULL);

struct lines mylines[4];

line(xmin, ymin, xmax, ymin);

line(xmax, ymin, xmax, ymax);

line(xmax, ymax, xmin, ymax);

line(xmin, ymax, xmin, ymin);

for (int i = 0; i < 4; i++) {

line(mylines[i].x1, mylines[i].y1,

mylines[i].x2, mylines[i].y2);

delay(1000);

}

for (int i = 0; i < 4; i++)

{

clip(mylines[i]);

delay(1000);

```



```
}  
  
delay(4000);  
  
getch();  
  
// For Closing the graph.  
  
closegraph();  
  
return 0;  
  
}
```

Output

```
160  xmin = 40;
161  xmax = 100;
162  ymin = 40;
163  ymax = 80;
164
165  /*for(int i = 0; i < 4; i++)
166  {
167      int x1, y1, x2, y2;
168      cin >> x1 >> y1 >> x2 >> y2;
169
170      mylines[i].x1 =
171      mylines[i].y1 =
172      mylines[i].x2 =
173      mylines[i].y2 =
174  }*/
175
176  initgraph(&gd, &gr
177
prskid1000@prskid1000:~/GitHub/
prskid1000@prskid1000:~/GitHub/
[xcb] Unknown sequence number w
[xcb] Most likely this is a mul
[xcb] Aborting, sorry about tha
ain: ../../src/xcb_io.c:260: p
-o main
alled
ce_lost' failed.
```

EXPERIMENT NO.10

Aim

Implementation of Polygon Clipping using Sutherland- Hodgeman Algorithm.

Theory

- For each edge e clip the given polygon against e
- The edge (of clipping area) is extended infinitely to create a boundary and all the vertices are clipped using this boundary. The new list of vertices generated is passed to the next edge of the clip polygon in clockwise fashion until all the edges have been used. There are four possible cases for any given edge of given polygon against current clipping edge e.
 - ❖ **Both vertices are inside:** Only the second vertex is added to the output list
 - ❖ **First vertex is outside while second one is inside:** Both the point of intersection of the edge with the clip boundary and the second vertex are added to the output list
 - ❖ **First vertex is inside while second one is outside:** Only the point of intersection of the edge with the clip boundary is added to the output list
 - ❖ **Both vertices are outside:** No vertices are added to the output list
- If the vertices of the clipper polygon are given in clockwise order then all the points lying on the right side of the clipper edges are inside that polygon. This

can be calculated using

Given that the line starts from (x_1, y_1) and ends at (x_2, y_2)

$$P = (x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1)$$

if $P < 0$, the point is on the right side of the line

$P = 0$, the point is on the line

$P > 0$, the point is on the left side of the line

- If two points of each line(1,2 & 3,4) are known, then their point of intersection can be calculated using the formula :-

$$(P_x, P_y) = \left(\frac{(x_1 y_2 - y_1 x_2)(x_3 - x_4) - (x_1 - x_2)(x_3 y_4 - y_3 x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}, \frac{(x_1 y_2 - y_1 x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 y_4 - y_3 x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \right)$$

Program

```
#include <bits/stdc++.h>
```

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

```
#define TRUE 1
```

```
#define FALSE 0
```

```
using namespace std;
```

```
typedef unsigned int outcode;
```

```
outcode CompOutCode(float x,float y);
```

```
enum{
```

```
TOP = 0x1,
```

```

BOTTOM = 0x2,

RIGHT = 0x4,

LEFT = 0x8

};

float xmin,xmax,ymin,ymax;

void clip(float x0,float y0,float x1,float y1)

{

outcode outcode0,outcode1,outcodeOut;

int accept = FALSE,done = FALSE;

outcode0 = CompOutCode(x0,y0);

outcode1 = CompOutCode(x1,y1);

do

{

if(!(outcode0|outcode1))

{

accept = TRUE;

done = TRUE;

}

else

if(outcode0 & outcode1)

```

```

done = TRUE;

else

{

float x,y;

outcodeOut = outcode0?outcode0:outcode1;

if(outcodeOut & TOP)

{

x = x0+(x1-x0)*(ymax-y0)/(y1-y0);

y = ymax;

}

else if(outcodeOut & BOTTOM)

{

x = x0+(x1-x0)*(ymin-y0)/(y1-y0);

y = ymin;

}

else if(outcodeOut & RIGHT)

{

y = y0+(y1-y0)*(xmax-x0)/(x1-x0);

x = xmax;

}

```

```

else

{

y = y0+(y1-y0)*(xmin-x0)/(x1-x0);

x = xmin;

}

if(outcodeOut==outcode0)

{

x0 = x;

y0 = y;

outcode0 = CompOutCode(x0,y0);

}

else

{

x1 = x;

y1 = y;

outcode1 = CompOutCode(x1,y1);

}

}

}while(done==FALSE);

if(accept)line(x0,y0,x1,y1);

```

```

rectangle(xmin,ymin,xmax,ymax);

}

outcode CompOutCode(float x,float y)

{

outcode code = 0;

if(y>ymax)

code|=TOP;

else if(y<ymin)

code|=BOTTOM;

if(x>xmax)

code|=RIGHT;

else if(x<xmin)

code|=LEFT;

return code;

}

int main( )

{

float x1,y1,x2,y2;

int gdriver = DETECT, gmode, n,poly[14],i;

```



```

cout << "Enter the no of sides of polygon:";

cin >> n;

cout << "\nEnter the coordinates of polygon\n";

for(i=0;i<2*n;i++)cin >> poly[i];

poly[2*n]=poly[0];

poly[2*n+1]=poly[1];

cout << "Enter the rectangular coordinates of clipping window\n";

cin >> xmin >> ymin >> xmax >> ymax;

initgraph(&gdriver, &gmode, NULL);

drawpoly(n+1,poly);

rectangle(xmin,ymin,xmax,ymax);

delay(5000);

for(i=0;i<n;i++)

{

clip(poly[2*i],poly[(2*i)+1],poly[(2*i)+2],poly[(2*i)+3]);

}

getch( );

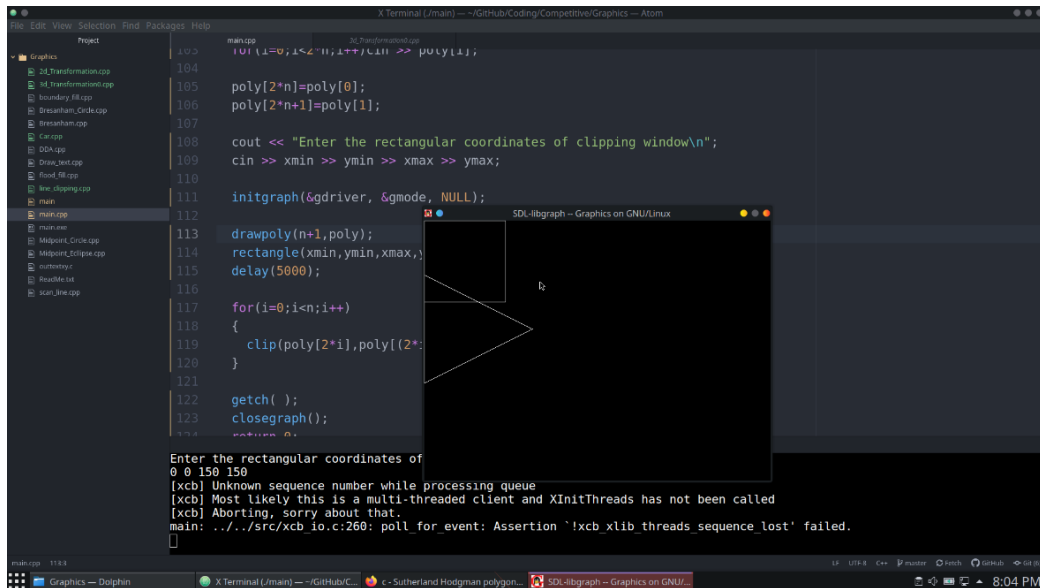
closegraph();

return 0;

}

```

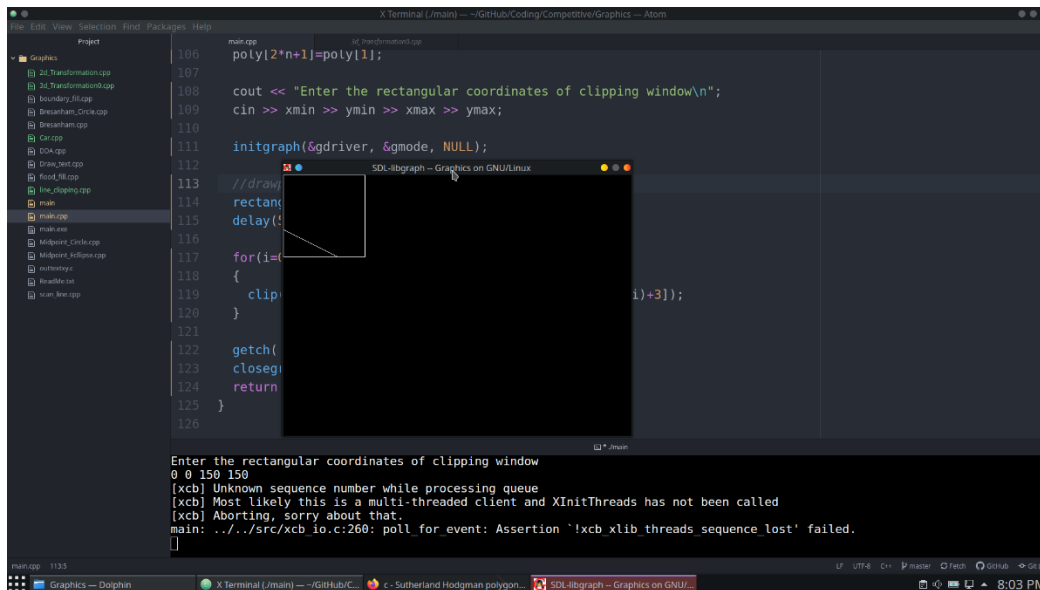
Output



```
main.cpp 103: for(i=0;i<2*n;i++) cin >> poly[i];
104:
105: poly[2*n]=poly[0];
106: poly[2*n+1]=poly[1];
107:
108: cout << "Enter the rectangular coordinates of clipping window\n";
109: cin >> xmin >> ymin >> xmax >> ymax;
110:
111: initgraph(&gdriver, &gmode, NULL);
112:
113: drawpoly(n+1,poly);
114: rectangle(xmin,ymin,xmax,ymax);
115: delay(5000);
116:
117: for(i=0;i<n;i++)
118: {
119:     clip(poly[2*i],poly[(2*i+1)+3]);
120: }
121:
122: getch();
123: closegraph();
124: return 0;
125: }
```

Enter the rectangular coordinates of clipping window\n0 0 150 150

[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.
main: ../../src/xcb_io.c:260: poll_for_event: Assertion '!xcb_xlib_threads_sequence_lost' failed.



```
main.cpp 106: poly[2*n+1]=poly[1];
107:
108: cout << "Enter the rectangular coordinates of clipping window\n";
109: cin >> xmin >> ymin >> xmax >> ymax;
110:
111: initgraph(&gdriver, &gmode, NULL);
112:
113: //draw
114: rectangle(xmin,ymin,xmax,ymax);
115: delay(5000);
116:
117: for(i=0;i<n;i++)
118: {
119:     clip(poly[2*i],poly[(2*i+1)+3]);
120: }
121:
122: getch();
123: closegraph();
124: return 0;
125: }
```

Enter the rectangular coordinates of clipping window\n0 0 150 150

[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.
main: ../../src/xcb_io.c:260: poll_for_event: Assertion '!xcb_xlib_threads_sequence_lost' failed.

EXPERIMENT NO.11

Aim

Write a C-program for performing the basic transformations such as Translation, Scaling, Rotation for a given 3D object.

Theory

Algorithm for scaling

- For each point P (x, y, z) do the following
- $x' = x * tx$
- $y' = y * ty$
- $z' = z * tz$
- Then finally plot the lines joining those points

Algorithm for Rotation

- For each point P (x, y, z) multiply the Point matrix with Rotation matrix

$$\begin{array}{c} \text{X-Rotation in 3D} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \quad \begin{array}{c} \text{Y-Rotation in 3D} \\ \begin{bmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \quad \begin{array}{c} \text{Z-Rotation in 3D} \\ \begin{bmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

- Then finally plot the lines joining those points

Algorithm for Translation

- For each point P (x, y, z) do the following
 - $x' = x + tx$
 - $y' = y + ty$
 - $z' = z + tz$
- Then finally plot the lines joining those points

Program

```
#include<bits/stdc++.h>

#include<graphics.h>

using namespace std;

int maxx,maxy,midx,midy;

void axis()

{

getch();

cleardevice();

line(midx,0,midx,maxy);

line(0,midy,maxx,midy);

}

void translation()
```

```

int x,y,z,o,x1,x2,y1,y2;

int gd=DETECT,gm;

detectgraph(&gd,&gm);

initgraph(&gd,&gm,NULL);

maxx=getmaxx();

maxy=getmaxy();

midx=maxx/2;

midy=maxy/2;

axis();

bar3d(midx+50,midy-100,midx+60,midy-90,10,1);

cout << "Enter translation factor\n";

cin >> x >> y;

cout << "After translation:\n";

bar3d(midx+x+50,midy-(y+100),midx+x+60,midy-(y+90),10,1);

cin >> x;

closegraph();

}

void scaling()

{

float x,y,z,o,x1,x2,y1,y2;

```

```

int gd=DETECT,gm;

detectgraph(&gd,&gm);

initgraph(&gd,&gm,NULL);

maxx=getmaxx();

maxy=getmaxy();

midx=maxx/2;

midy=maxy/2;

axis();

bar3d(midx+50,midy-100,midx+60,midy-90,5,1);

cout << "Enter scaling factors\n";

cin >> x >> y >> z;

cout << "After scaling\n";

bar3d(midx+(x*50),midy-(y*100),midx+(x*60),midy-(y*90),5*z,1);

delay(3000);

cin >> x;

closegraph();

}

void rotation()

{

float x,y,z,o,x1,x2,y1,y2;

```

```

int gd=DETECT,gm;

detectgraph(&gd,&gm);

initgraph(&gd,&gm,NULL);

maxx=getmaxx();

maxy=getmaxy();

midx=maxx/2;

midy=maxy/2;

axis();

bar3d(midx+50,midy-100,midx+60,midy-90,5,1);

cout << "Enter rotating angle\n";

cin >> o;

x1=50*cos(o*3.14/180)-100*sin(o*3.14/180);

y1=50*sin(o*3.14/180)+100*cos(o*3.14/180);

x2=60*cos(o*3.14/180)-90*sin(o*3.14/180);

y2=60*sin(o*3.14/180)+90*cos(o*3.14/180);

cout << "After rotation about z axis\n";

bar3d(midx+x1,midy-y1,midx+x2,midy-y2,5,1);

cin >> o;

cout << "After rotation about x axis\n";

bar3d(midx+50,midy-x1,midx+60,midy-x2,5,1);

```

```
cin >> o;

cout << "After rotation about yaxis\n";

bar3d(midx+x1,midy-100,midx+x2,midy-90,5,1);

cin >> o;

closegraph();

}

int main()

{

translation();

rotation();

scaling();

return 0;

}
```


Output

Translation

Rotation

[illegible]