Name : Yogender Singh   Roll No : 1121173 (58)   Subject Code - PBC-602

Course : BCA - 6C   Subject : Computer Graphics

P1: <u>Bresenham Line Drawing Algorithm :</u>

```
#include <Stdio.h>
# include <graphics.h>
 int main ()
{
   int rou (float num)
    {
      return num < 0 ? num - 0.5 : num + 0.5;
    }
  int x1 = 100, x2 = 300 , y1 = 100 , y2 = 300 ;
   int gd = DETECT , gm;
   float p , pk , x , y , step;
    int dx = x2 - x1;
    int dy - y2 - y1;
      p = 2* dx - dy;
       if (dx > dy)
        step = dx;
        else
        step = dy;
      initgraph (&gd, &gm, " ");
        outtextxy (x1, y1, "A");
        outtextxy ( x2, y2, "B");
         putpixel (x1, y1, WHITE);
  x= x1 , y1;
 while (step >0)
{
  if (pk <0)
{
  pk = pk + 2* dy;
}
 else
 {
  pk = pk + 2*dy - 2* dx;

 y++;
}
putpixel (rou (x) , rou (y) , WHITE);
 x++;
Step-- ;
}
getch ();
 return 0;
}
```

Y singh

Algorithm:- Bresenham line drawing

Step1:- Start

Step2:- Declare x, y, x1, y1, x2, y2, steps, p, pk in float data type.

Step3:- Delare gm and initialize gd= DETECT and i=1.

Step4:- Enter coordinates x1 & y1 of first point.

Step5:- Enter coordinates x2 & y2 of second point.

Step6:- Initialize graph by using initgraph (&gd, &gm, " ").

Step7:- Calculate, dx = x2 - x1,
dy = y2 - y1,
Steps = dx - 1.

Step 8:- Initialize decision parameter
pk = (2* dy) - dx

Step9:- Initialize p=pk, x=x1, y=y1

Step10:- Repeat step 11 to step 13 while i<= steps

Step11:- Check if p<0, then
putpixel(x,y, WHITE);
x=x+1;
y=y;
p= ~~p~~ p+(2* dy);

otherwise go to step 12

```
step12:- putpixel (x, y, WHITE);
            x = x+1;
            y = y+1;
            p = p+ (2* dy) - (2* d~~y~~ x);

Step13:- Increment i by one

Step14:- Close the graph.
```

P2: Mid Paint Circle Drawing Algorithm :

```
# include <graphics.h>
# include <stdio.h>
 void midpoint (int midx, int midy, int r)
 {
     int x=0, y=r, gd=0, gm, di, dnext;
        initgraph (&gd, &gm, " ");
       di = 1.25-r;
       while (x<=y)
      {
          if (di>=0)
        {
           dnext = di+2*(x-y)+1;
            x++;
            y--;
       }
       else
        {
         dnext = di+2*x+1;
            x++;
      }
       putpixel (x+midx, y+midy, 5);
       putpixel (y+mid x, x+midy, 5);
       putpixel (-x+ mid x, -y+midy, 5);
       putpixel (-y+ mid x, -x+ midy, 5);
       putpixel (-y+ mid x, x+midy, 5);
        putpixel (y + mid x, -x + midy, 5);
        putpixel( x +mid x, -y + midy, 5);
```

Ysingh

```
        putpixel (-x+ midx, y+midy ,5);

          di= dnext;

      }
      ~~getch;~~ getch();
        closegraph ();
  }

     int main ()
     {
       int gd=0, gm;
       int midx=0, midy=0 ,r=0;
       printf(" enter the coordinates (x,y):");
       scanf("%d %d", &midx, &midy);
       printf ("enter the radius :");
         scanf("%d", &r);
         midpoint(midx, midy, r);
         return 0;
     }
```

Algorithm:- Mid-point Circle drawing:

Step1:- Start Algorithm

Step2:- Plot the center coordinates (p_0, q_0) follows.

$p_0 = 0, q_0 = r$

Step3:- Calculate, the init decision parameter $d_0 = 1 - r$;

Step4:- Assume the starting coordinates (p_n, p_k)

The next coordinates will be (p_{k+1}, q_{k+1})

Find the next point of first octant according to d_k.

Step5:- Follows thes cases:-

Case 1:- If $d_k < 0$, then

$p_{k+1} = p_k + 1$

$q_{k+1} = q_k$

$d_{k+1} = d_k + 2p_{k+1}$

Case 2:- if $d_k >= 0$, then

$p_{k+1} = p_k + 1$

$q_{k+1} = q_k - 1$

$d_{k+1} = d_k - 2(q_{k+1} + 2p_{k+1} + 1)$

Step6:- If center not (0,0) point will be

x coordinate = $x_c + p_0$

y coordinate = $y_c + q_0$

Step7:- Repeat Steps 5 & 6 untill $n >= s$

Step 8- stop.

Ysingh