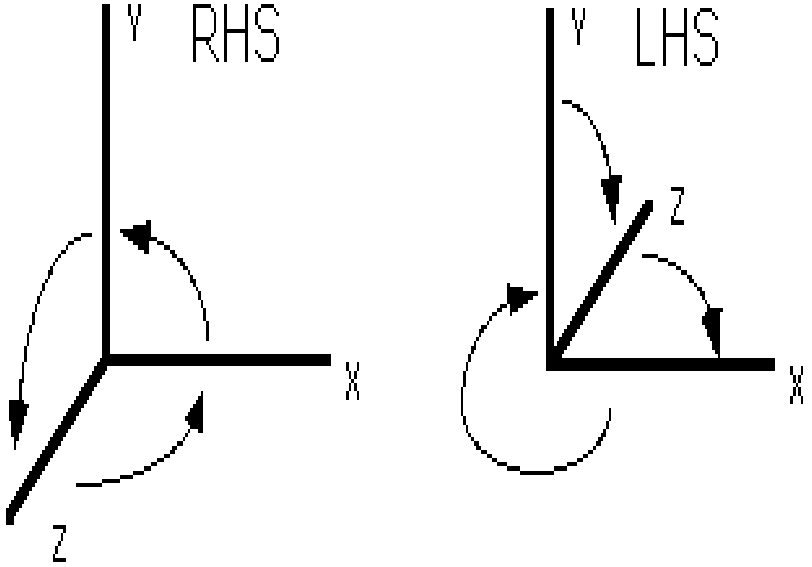
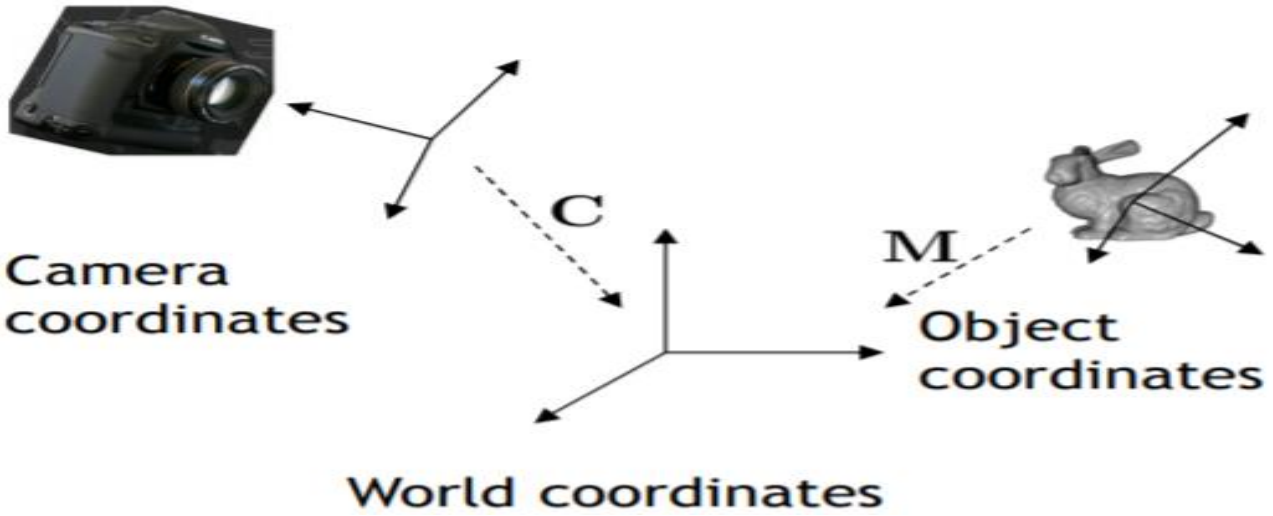
Unit-2: Line Drawing Technique

- ✓ 2.1 Co-ordinate Systems
- ✓ 2.2 The Simple DDA
- ✓ 2.3 The Symmetrical DDA
- ✓ 2.4 Bresenham’s line drawing Algorithm
- ✓ 2.5 Bresenham’s circle drawing Algorithm

2.1 Co-ordinate Systems



- In a 2-D coordinate system the X axis generally points from left to right, and the Y axis generally points from bottom to top.
- When we add the third coordinate, Z, we have a choice as to whether the Z-axis points into the screen or out of the screen:



World **Coordinates**: Common reference frame for all objects in the scene. No standard for coordinate system orientation. If there is a ground plane, usually x/y is horizontal and z points up (height). Otherwise, x/y is often screen plane, z points out of the screen

Camera Coordinate System: Origin defines center of projection of camera. x-y plane is parallel to image plane. z-axis is perpendicular to image plane.

Object Coordinates: Local coordinates in which points and other object geometry are given. Often origin is in geometric center, on the base, or in a corner of the object. Depends on how object is generated or used.

World Coordinate System

- Also known as the "universe" or sometimes "model" coordinate system.
- This is the base reference system for the overall model
- Generally in 3D, all other model coordinates relate.

Object Coordinate System

- When each object is created in a modelling program, the modeller must pick some point to be the origin of that particular object, and the orientation of the object to a set of model axes.
- For example when modelling a desk, the modeller might choose a point in the center of the desk top for the origin, or the point in the center of the desk at floor level, or the bottom of one of the legs of the desk.
- When this object is moved to a point in the world coordinate system, it is really the origin of the object (in object coordinate system) that is moved to the new world coordinates, and all other points in the model are moved by an equal amount.
- Note that while the origin of the object model is usually somewhere on the model itself, it does not have to be.
- For example, the origin of a doughnut or a tire might be in the vacant space in the middle.

Hierarchical Coordinate Systems

Sometimes objects in a scene are arranged in a hierarchy, so that the "position" of one object in the hierarchy is relative to its parent in the hierarchy scheme, rather than to the world coordinate system.

For example, a hand may be positioned relative to an arm, and the arm relative to the torso.

When the arm moves, the hand moves with it, and when the torso moves, all three objects move together.

Viewpoint Coordinate System

- Also known as the "camera" coordinate system.
- This coordinate system is based upon the viewpoint of the observer, and changes as they change their view.
- Moving an object "forward" in this coordinate system moves it along the direction that the viewer happens to be looking at the time.

Model Window Coordinate System

- Not to be confused with desktop windowing systems (MS Windows or X Windows)
- This coordinate system refers to the subset of the overall model world that is to be displayed on the screen.
- Depending on the viewing parameters selected, the model window may be rectalinear or a distorted viewing frustrum of some kind.

Screen Coordinate System

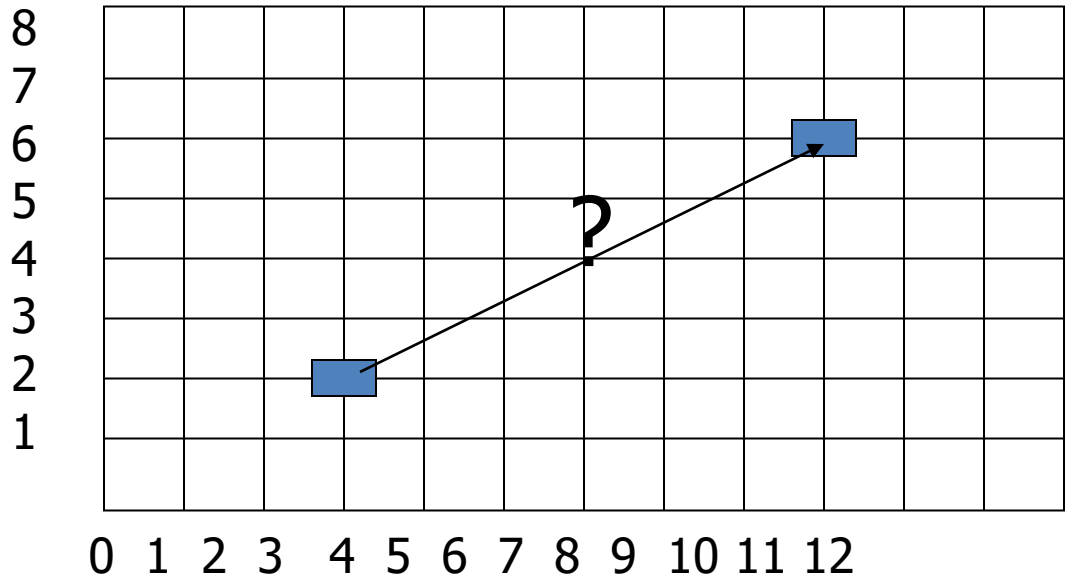
- This 2D coordinate system refers to the physical coordinates of the pixels on the computer screen, based on current screen resolution.
- Example: 1024x768

Viewport Coordinate System

This coordinate system refers to a subset of the screen space where the model window is to be displayed.

Typically the viewport will occupy the entire screen window, or even the entire screen, but it is also possible to set up multiple smaller viewports within a single screen window.

Line Drawing Algorithm



Line: (4,2) -> (12,6)

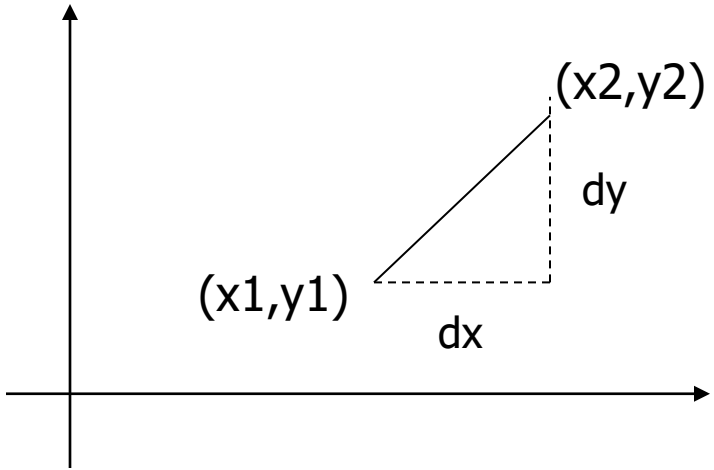
Which intermediate pixels to turn on?

Line Drawing Algorithm

- Slope-intercept line equation
  - y = mx + b
  - Given two end points (x1,y1), (x2, y2)

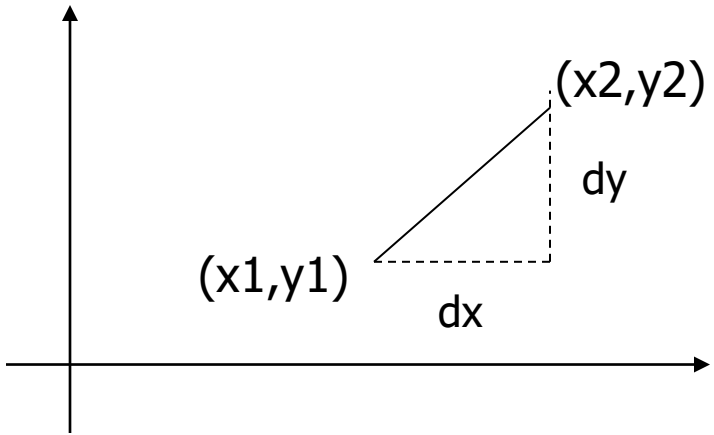
$$m = \frac{dy}{dx} = \frac{y2 - y1}{x2 - x1}$$

$$b = y1 - m * x1$$



Digital Differential Analyzer (DDA): Line Drawing Algorithm

- Line drawing starts from (x1,y1) and end upto (x2,y2)
- Condition→ x, y increments to values only in [0,1] range
- Case 1: x is incrementing faster (m < 1)
  Step in x=x+1, y=y+m
- Case 2: y is incrementing faster (m > 1)
  Step in x=x+1/m, y=y+1



DDA Line Drawing Algorithm (Case a: m < 1)

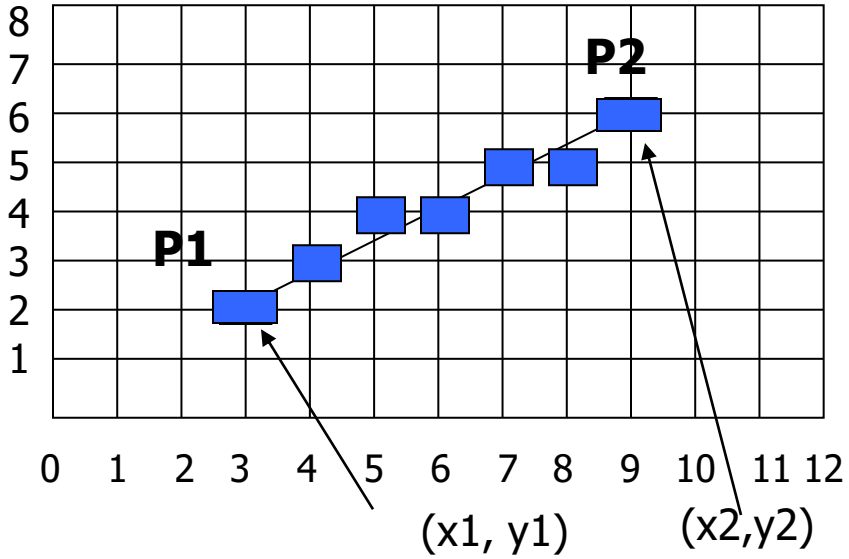
$$y_{k+1} = y_k + m$$

Draw line segment (P1 to P2)

P1(x1, y1) P2(x2, y2)
P1(3 , 2) P2(9 , 6)

dx = 9 – 3= **6**
dy = 6 – 2 =**4**

Step = max (abs(dx), abs(dy)) = max (6 , 4)
Step = 6



xinc=6/6=1 yinc=4/6=0.6
x=x+xinc y=y+yinc

X	Y	
3	2	
4	2.6	3
5	3.2	4
6	3.8	4
7	4.4	5
8	5	5
9	5.6	6

```
Algorithm DDA(x1,y1,x2,y2)
{
dx=x2-x1
dy=y2-y1
if (abs(dx) >  abs(dy))
    step = abs(dx)
else
     step = abs(dy)
xinc=dx/step
yinc=dy/step
x=x1;
y=y1;
for(i=1; i<=step; i++)
{
  putpixe(x,y)
  x=x+xinc;
  y=y+yinc;
}
}
```

DDA Line Drawing Algorithm (Case 2: m > 1)

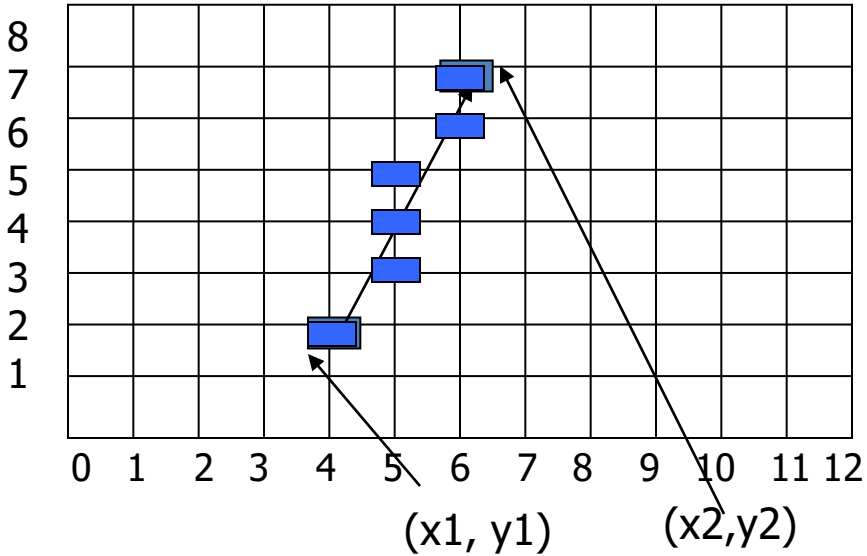
$$x_{k+1} = x_k + \frac{1}{m}$$

Draw line segment (P1 to P2)

P1(x1, y1) P2(x2, y2)
P1(4 , 2) P2(6 , 7)

dx = 6 – 4 = 2
dy = 7 – 2 = **5**

Step = max (abs(dx), abs(dy)) = max (2 , 5)
Step = 5



xinc=2/5=0.4 yinc=5/5=1
x=x+xinc y=y+yinc

X	R(X)	Y
4		2
4.4	5	3
4.8	5	4
5.2	5	5
5.6	6	6
6		7

```
Algorithm DDA(x1,y1,x2,y2)
{
dx=x2-x1
dy=y2-y1
if (abs(dx) >  abs(dy))
    step = abs(dx)
else
     step = abs(dy)
xinc=dx/step
yinc=dy/step
x=x1;
y=y1;
for(i=1; i<=step; i++)
{
  putpixe(x,y)
  x=x+xinc;
  y=y+yinc;
}
}
```

✓ 2.4 Bresenham’s line drawing Algorithm

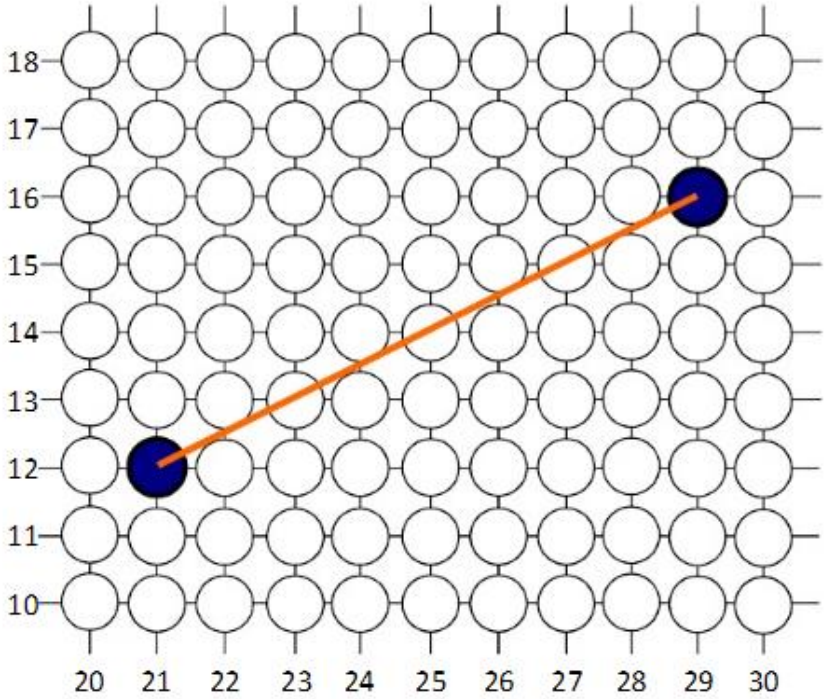
✓ 2.4 Bresenham's line drawing Algorithm

```
Void LineBres( int x1, int y1, int x2, int y2)
{
 int dx =abs(x2-x1), dy= abs(y2-y1)
 int p=2* dy- dx ;
 int x, y, Xend;     //  Xends is similar to steps in DDA
   If(x1>x2)
   {
       x=x2;y=y2;Xend=x1;
    }
  else
   {
     x=x1;y=y1;Xend=x2;
   }
 setPexel( x, y );
```

```
While( x < Xend)
{
   x++;
    if( p < 0)
       p=p+2dy;
    else
     {
        y++;
        p=p+2dy-2dx;
     }
    setPixel(x,y);
  }   // End While

}   //End LineBres
```

Bresenham Exercise (cont…)



k	p_k	(x_{k+1}, y_{k+1})
0		
1		
2		
3		
4		
5		
6		
7		
8		

The starting point (x0, y0)=(30,20) and the successive pixel positions are given in the following table

- Two End points P1(30,20) and P2(40,28)
- dx=x2-x1=40-30=10 dy= y2-y1=28-20=8
- P0=2dy-dx=2X8-10 = 6 >0 pt (31,21)
- Pk+1= pk+2dy-2dx= 6+16-20 =2 >0 (32,22)
- Pk+1=2+16-20 =-2 <0 (33,22)
- Pk+1= pk+2dy =-2+16 = 14 >0 (34,23)
- Pk+1= pk+2dy-2dx=14+16-20=10>0 (35,24)
- Pk+1=10+16-20=6 >0 (36,25)
- Pk+1= 6+16-20=2>0 (37,26)
- Pk+1=2+16-20=-2 <0 (38,26)
- Pk+1= pk+2dy=-2+16=14>0 (39,27)
- Pk+1=pk+2dy-2dx= 14+16-20=10>0 (40,28)

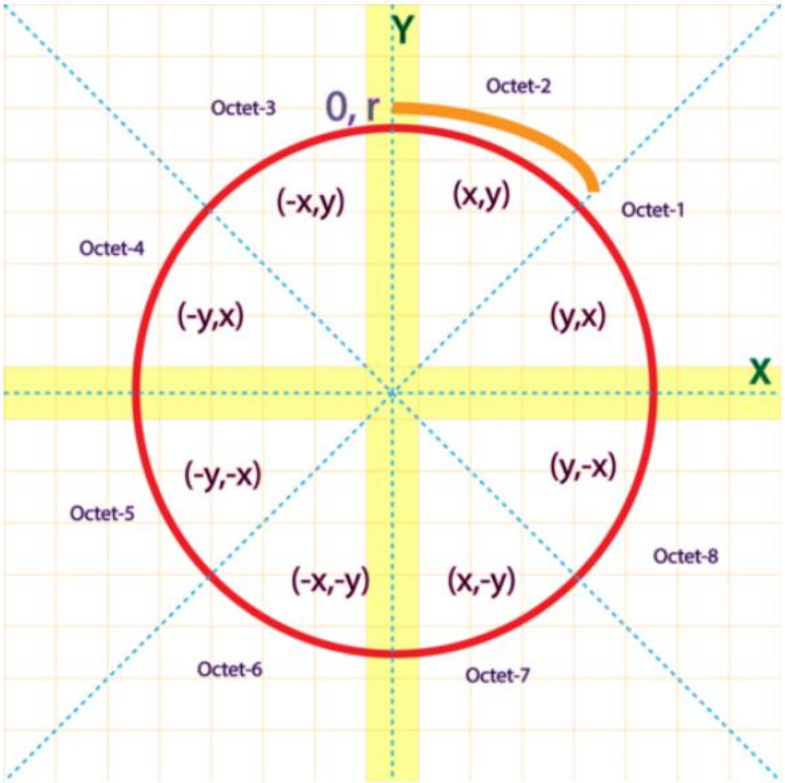
K	P_k	(X_{k+1}, Y_{k+1})
0	6	(31,21)
1	2	(32,22
2	-2	33,22
3	14	34,23
4	10	35,24
5	6	36,25
6	2	37,26
7	-2	38,26
8	14	39,27
9	10	40,28

2.5 Bresenham's Circle Drawing Algorithm

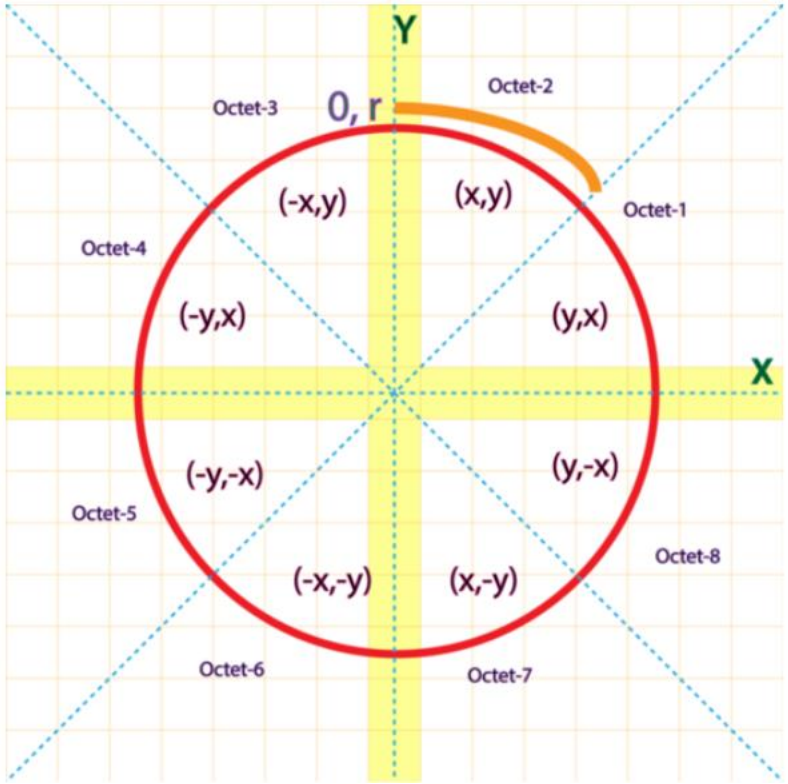
A circle is made up of 8 Equal Octets so we need to find only coordinates of any one octet rest we can conclude using that coordinates.

We took octet-2. Where X and Y will represent the pixel

Let us make a function Circle() with parameters coordinates of Centre (Xc,Yc) and pixel point (X,Y) that will plot the pixel on screen.



We will find pixels assuming that Centre is at Origin (0,0) then we will add the coordinates of centre to corresponding X and Y while drawing circle on screen.




```
Circle (Xc,Yc,X,Y) {

Plot (Y+Xc , X+Yc)      ……Octet-1
Plot (X+Xc , Y+Yc)      ……Octet-2
Plot (-X+Xc , Y+Yc)     ……Octet-3
Plot (-Y+Xc , X+Yc)     …..Octet-4
Plot (-Y+Xc , -X+Yc)    ……Octet-5
Plot (-X+Xc , -Y+Yc)    ……Octet-6
Plot (X+Xc , -Y+Yc)     ……Octet-7
Plot (Y+Xc , -X+Yc)     ……Octet-8

}
```

Bresenham's Circle Drawing C++ Program

```
#include<conio.h>
#include<Graphics.h>
#include<stdio.h>
void main() {
int gd = DETECT, gm;
initgraph (&gd, &gm, "C:\\TC\\BGI");
int xc, yc, x, y, r, p;
printf ("Enter Radius of Circle: ");
scanf ("%d", &r);
printf ("Enter coordinates of centre of Circle: ");
scanf ("%d%d", &xc, &yc);
x=0;    y=r;
p=3-(2*r);          //Initial Decision parameter
while (x<=y) {
putpixel (y+xc, x+yc, 15 );          //……octet-1
putpixel (x+xc, y+yc, 15 );          //……octet-2
putpixel (-x+xc, y+yc, 15 );         //……octet-3
putpixel (-y+xc, x+yc, 15 );         //……octet-4
putpixel (-y+xc, -x+yc, 15 );        //……octet-5
putpixel (-x+xc, -y+yc, 15 );        //……octet-6
putpixel (x+xc, -y+yc, 15 );         //……octet-7
putpixel (y+xc, -x+yc, 15 );         //……octet-8
if (p<0) {
p=p+4*x+6;       //Next Decision Parameter
x++;
}
else {
p=p+4*(x-y)+10;       //Next Decision Parameter
x++;
y--;
}
}
getch ();
closegraph ();
}
```