Faculty of Computers & artificial Intelligence
Beni-Suef University

CIRCLE DRAWING ALGORITHMS USING MIDPOINT ALGORITHM

Ahmed Mahmoud Sultan

Agenda

- Mid Point Algorithm
- Mid Point Pseudo Code.
- Mid Point Example.
- Mid Point Code.
- Examples of OpenGL.

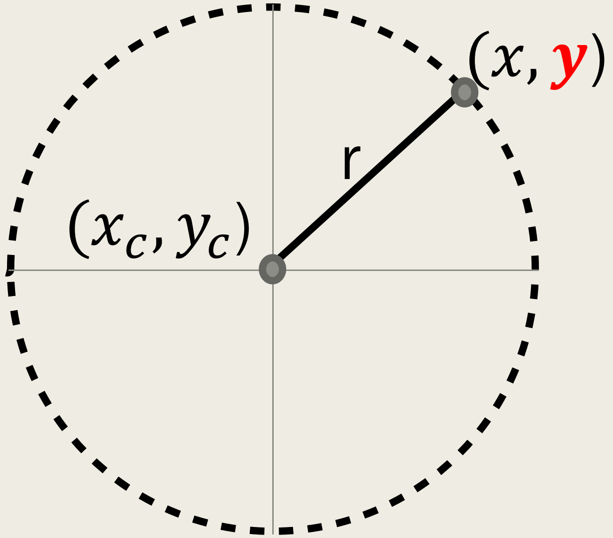
What is a Circle

- A circle is defined as a set of points that are all have the same distance from a given center (Xc , $Y\ c$).

- This distance relationship is expressed by the pythagorean theorem in **Cartesian coordinates** as.

$$(x - x_c)^2 + (y - y_c)^2 = \ r^2$$

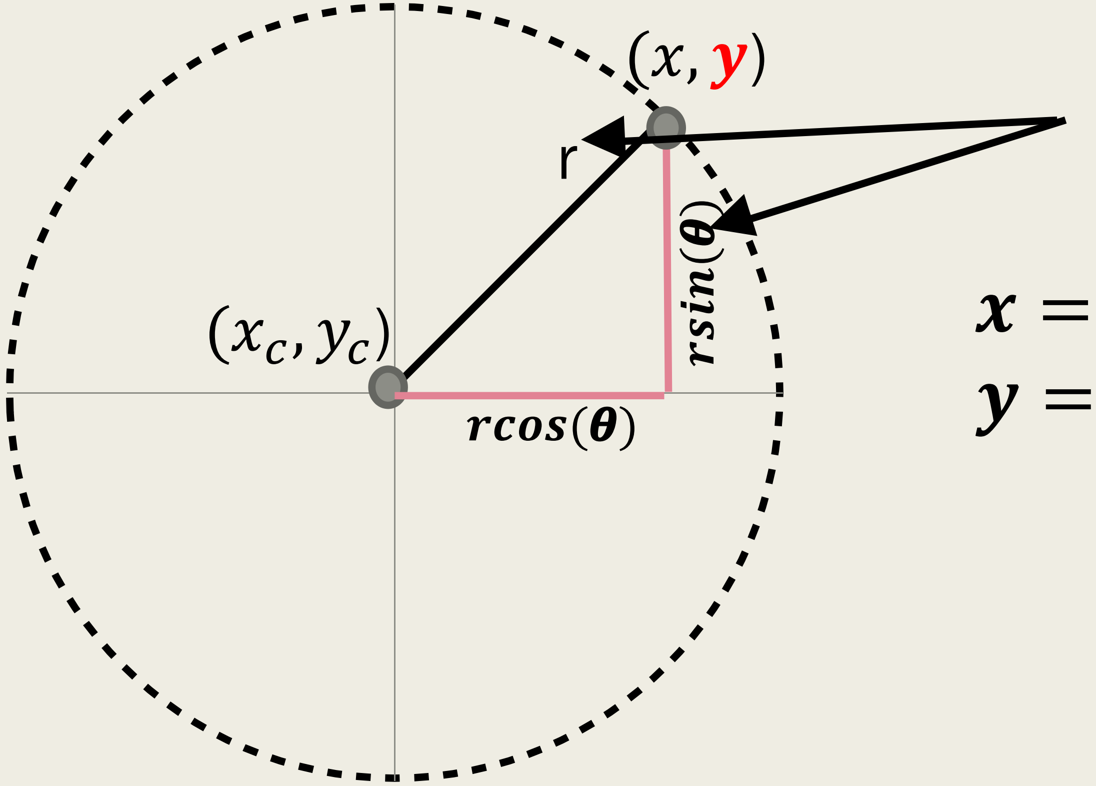
$$y = \ y_c \pm \sqrt{r^2 - (x_c - x)^2}$$



Draw Backs

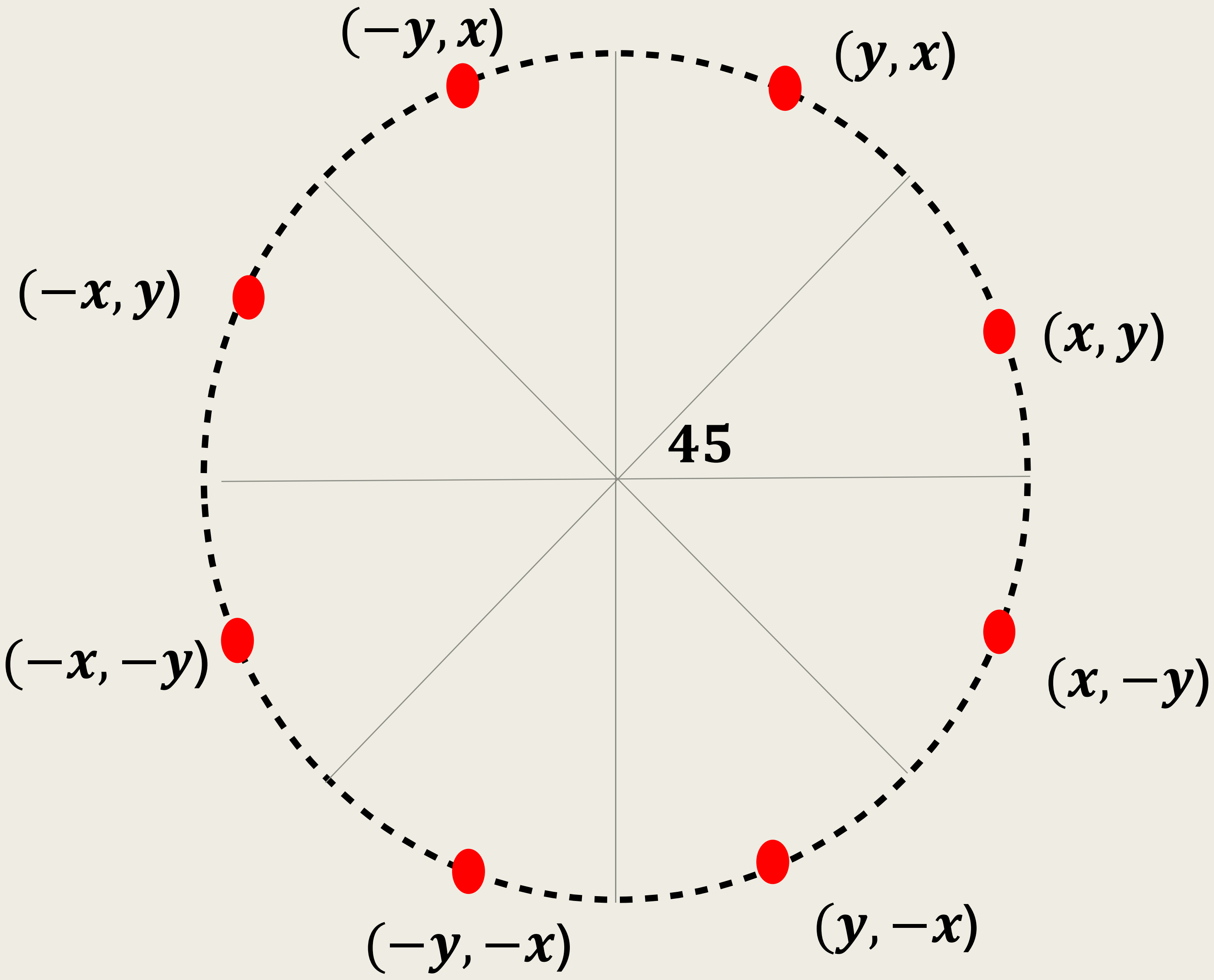
- Considerable amount of computation.
- Spacing between plotted pixels is not uniform.

Polar co-ordinates for a circle



$$x = x_c + rcos(\theta)$$
$$y = y_c + rsin(\theta)$$

Polar co-ordinates for a circle



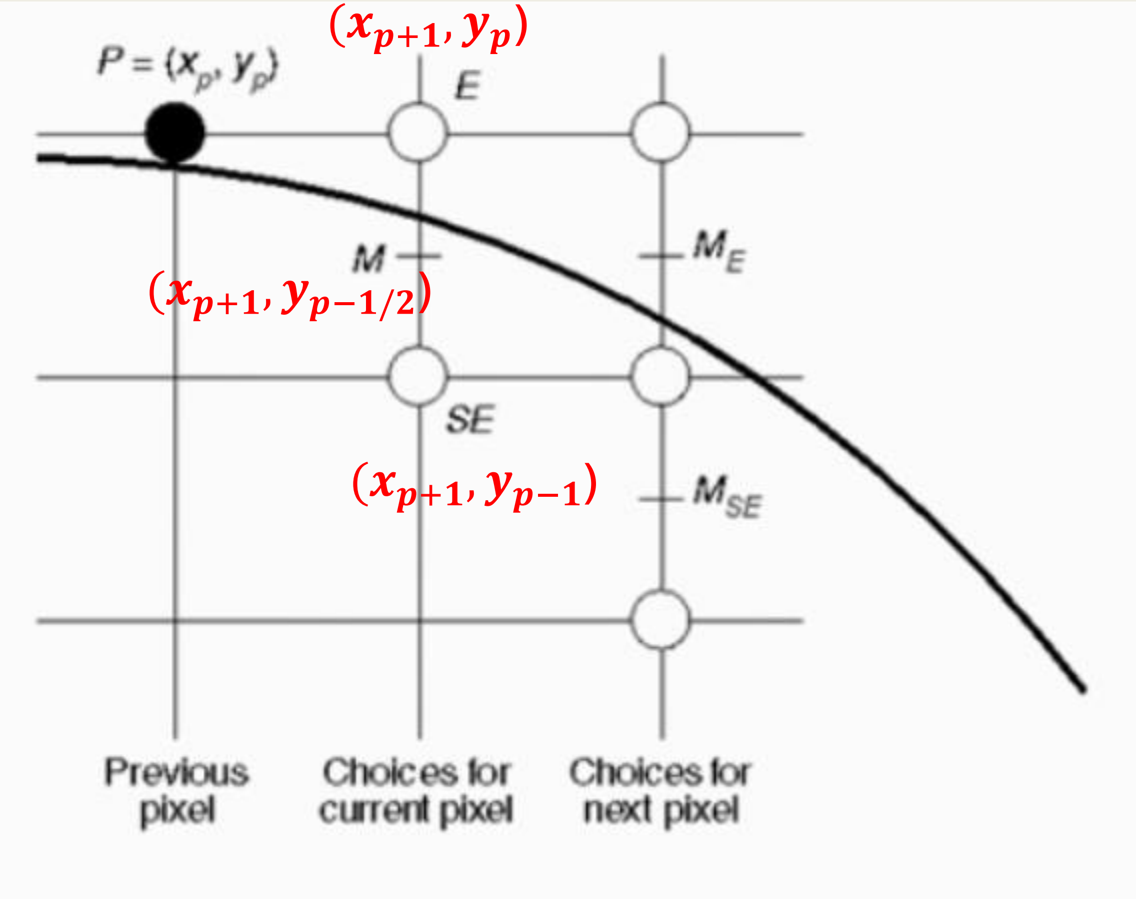
❑ But, note that circle sections in adjacent octants within one quadrant are symmetric with respect to the 45° line dividing the two octants

❑ But This method is still computationally expensive

Mid point Algorithm (Bresenham concept)

- ❑ We will first calculate pixel positions for a circle centered around the origin (0,0). Then, each calculated position (x,y) is moved to its proper screen position by adding xc to x and yc to y

- ❑ $f(x,y) = x^2 + y^2 - r^2 = 0$
- ❑ $f(x,y) > 0$ (point outside circle)
- ❑ $f(x,y) < 0$ (point inside circle)
- ❑ $f(x,y) = 0$ (point on circle)
- ❑ $f(M) = d = f(x_{p+1}, y_{p-\frac{1}{2}})$
- ❑ $f(x,y) = (x_{p+1})^2 + (y_{p-\frac{1}{2}})^2 - r^2$

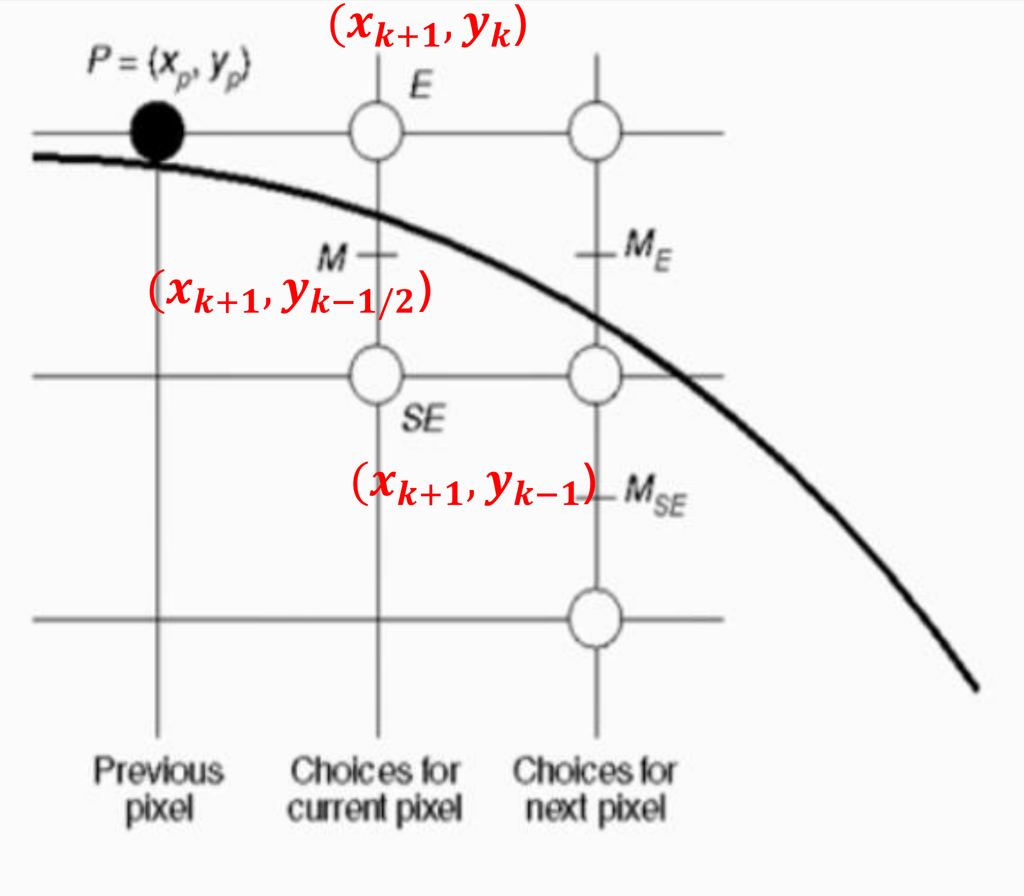


Mid point Algorithm Cont…

- ❑ Assuming we have just plotted the pixel at x_k , y_k we next need to decide which one of the following two pixels is closer to the circle:

$$(x_{k+1}, y_k) \text{ or } (x_{k+1}, y_{k-1}).$$

- ❑ **Note** : Our decision parameter is the circle function evaluated at the midpoint between these two pixels



Mid point Algorithm Cont…

$$p_k = f_{circle}(x_{k+1}, y_k - \frac{1}{2})$$

$$p_k = (x_{k+1})^2 + (\ y_k - \frac{1}{2})^2 - r^2$$

- ❑ We have now two decisions
  - ❑$p_k > \mathbf{0}$ (point outside circle)
  - ❑$p_k < \mathbf{0}$ (point inside circle)

Mid point Algorithm Cont…

$$p_{k+1} = f_{circle}(x_{k+1} + 1, y_{k+1} - \frac{1}{2})$$

$$p_{k+1} = (x_{k+1} + 1)^2 + (y_{k+1} - \frac{1}{2})^2 - r^2$$

$$p_{k+1} = p_k + 2(x_k + 1) + \left({y_{k+1}}^2 - {y_k}^2 \right) - (y_{k+1} - y_k) + 1$$

Mid point Algorithm Cont...

$$p_{k+1} = p_k + 2(x_{k+1} + 1) + \left({y_{k+1}}^2 - {y_k}^2 \right) - (y_{k+1} - y_k) + 1$$

$if\ p_k < \mathbf{0}$

- ❑ $y_{k+1} = y_k$
- ❑ $p_{k+1} = p_k + 2(x_k + 1) + \left({y_k}^2 - {y_k}^2 \right) - (y_k - y_k) + 1$
- ❑ $p_{k+1} = p_k + 2(x_k + 1)$**+1**
- ❑ $p_{k+1} = p_k + 2x_{k+1} + 1$

Mid point Algorithm Cont…

$$p_{k+1} = p_k + 2(x_{k+1} + 1) + \left({y_{k+1}}^2 - {y_k}^2 \right) - (y_{k+1} - y_k) + 1$$

$$\boldsymbol{if\ p_k > 0}$$

□ $y_{k+1} = y_k - 1$

□ $p_{k+1} = p_k + 2(x_k + 1) + \left((y_k - 1)^2 - {y_k}^2 \right) - (y_k - 1 - y_k) + 1$

□ $p_{k+1} = p_k + 2(x_k + 1) + (-2y_k + 1) + 1 + 1$

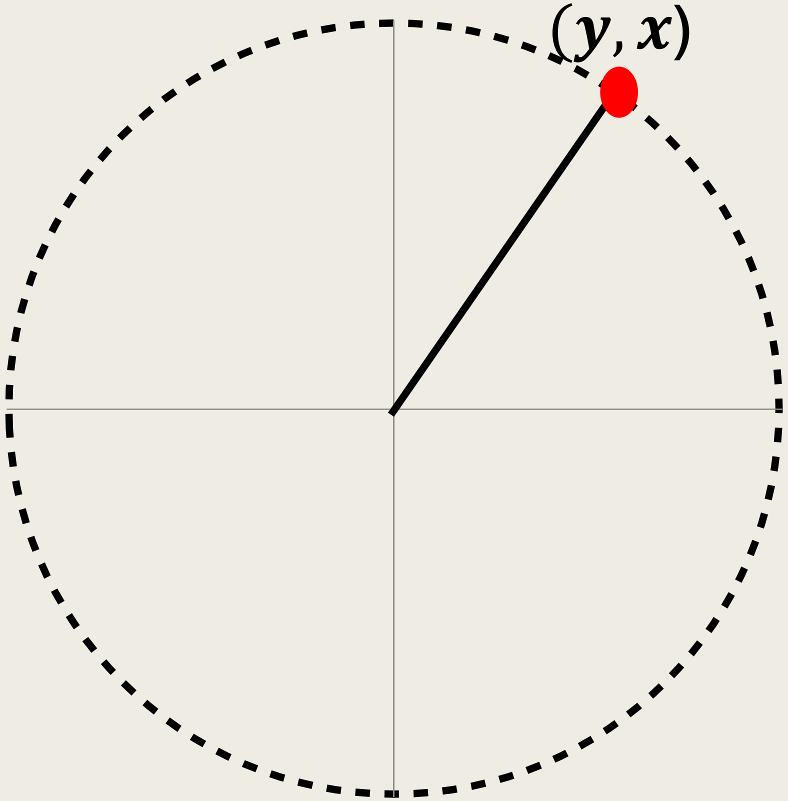
□ $p_{k+1} = p_k + 2(x_k + 1) - 2y_k + 2 + 1$

□ $p_{k+1} = p_k + 2(x_k + 1) - 2(y_k - 1) + 1$

□ $p_{k+1} = p_k + 2x_{k+1} - 2y_{k+1} + 1$

Mid Point Algorithm

- ❑ *Input radius r and circle center* $(\boldsymbol{x_0}, \boldsymbol{y_0})$ *and obtain the first point on the circumstances of a circle centered on origin as :*
  - ❑ $(\boldsymbol{x_0}, \boldsymbol{y_0})$.
- ❑ Calculate the initial value of the decision parameter as :
  - ❑ $\boldsymbol{p_0 = \frac{5}{4} - r}$
  - ❑ $\boldsymbol{p_0 = f_{circle}(1, r - \frac{1}{2})}$
  - ❑ $\boldsymbol{p_0 = 1 + (r - \frac{1}{2})^2 - r^2}$
- ❑ *If the radius r is specified as an integer we*
- ❑ *can simply round p0 to*
  - ❑ $\boldsymbol{p_0 = 1 - r}$



Mid Point Algorithm cont …

- ❑ *At each* $\boldsymbol{x_k}$ *position starting at k=0 perform the following test if*

$\boldsymbol{p_k} < 0$ *the next point along the circle centered on(0,0) is* $(\boldsymbol{x_{k+1}}, \boldsymbol{y_k})$

$$p_{k+1} = p_k + 2x_k + 1$$

- ❑ *Otherwise , the next point along the circle is* $(\boldsymbol{x_{k+1}}, \boldsymbol{y_{k+1}})$

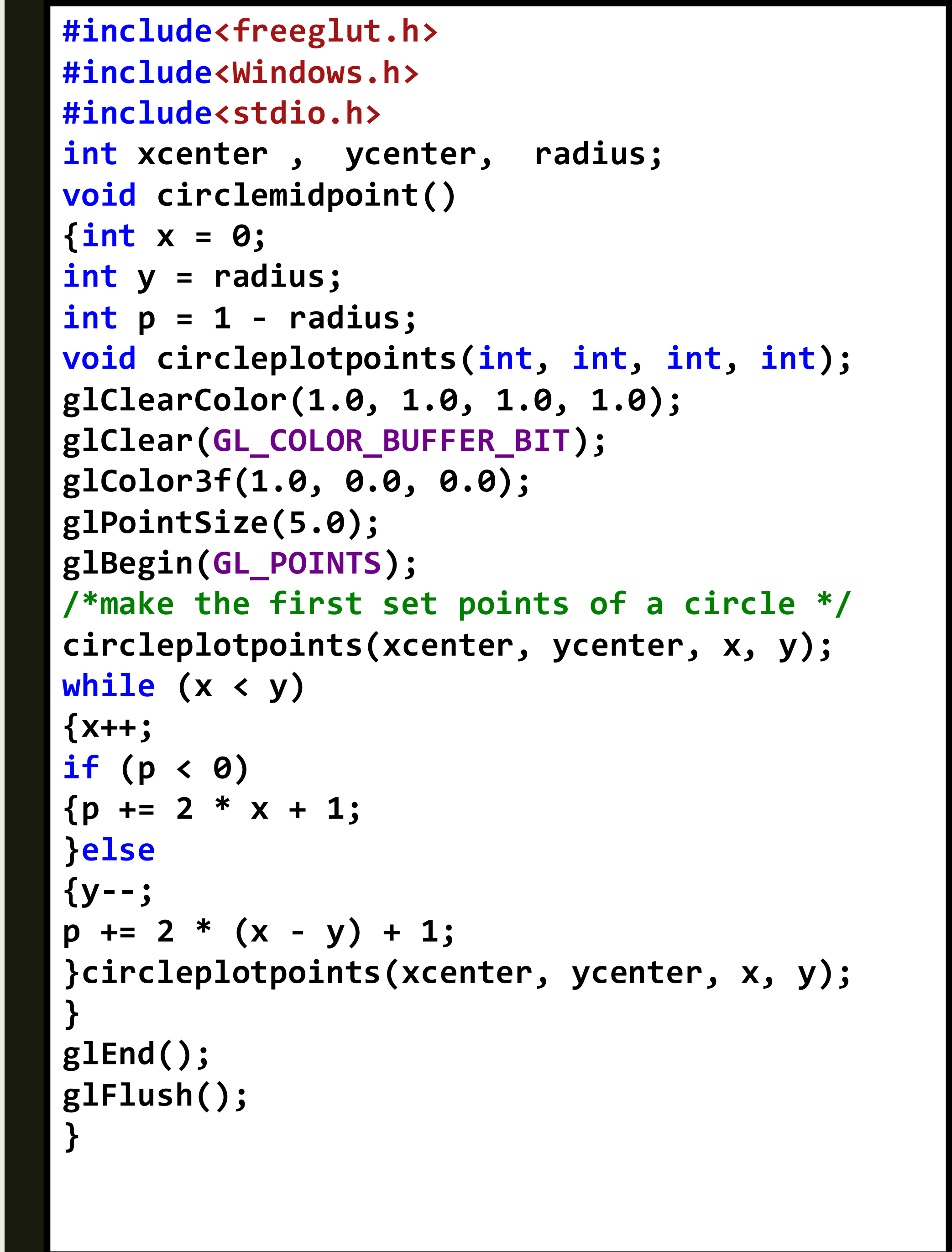
$$p_{k+1} = p_k + 2x_{k+1} - 2y_{k+1} + 1$$

- ❑ *Determine the symmetry points in the other seven points.*
- ❑ *Move each calculated position (x,y) onto the circle path centere on* $(\boldsymbol{x_c}, \boldsymbol{y_c})$ *and plot the coordinates values :*

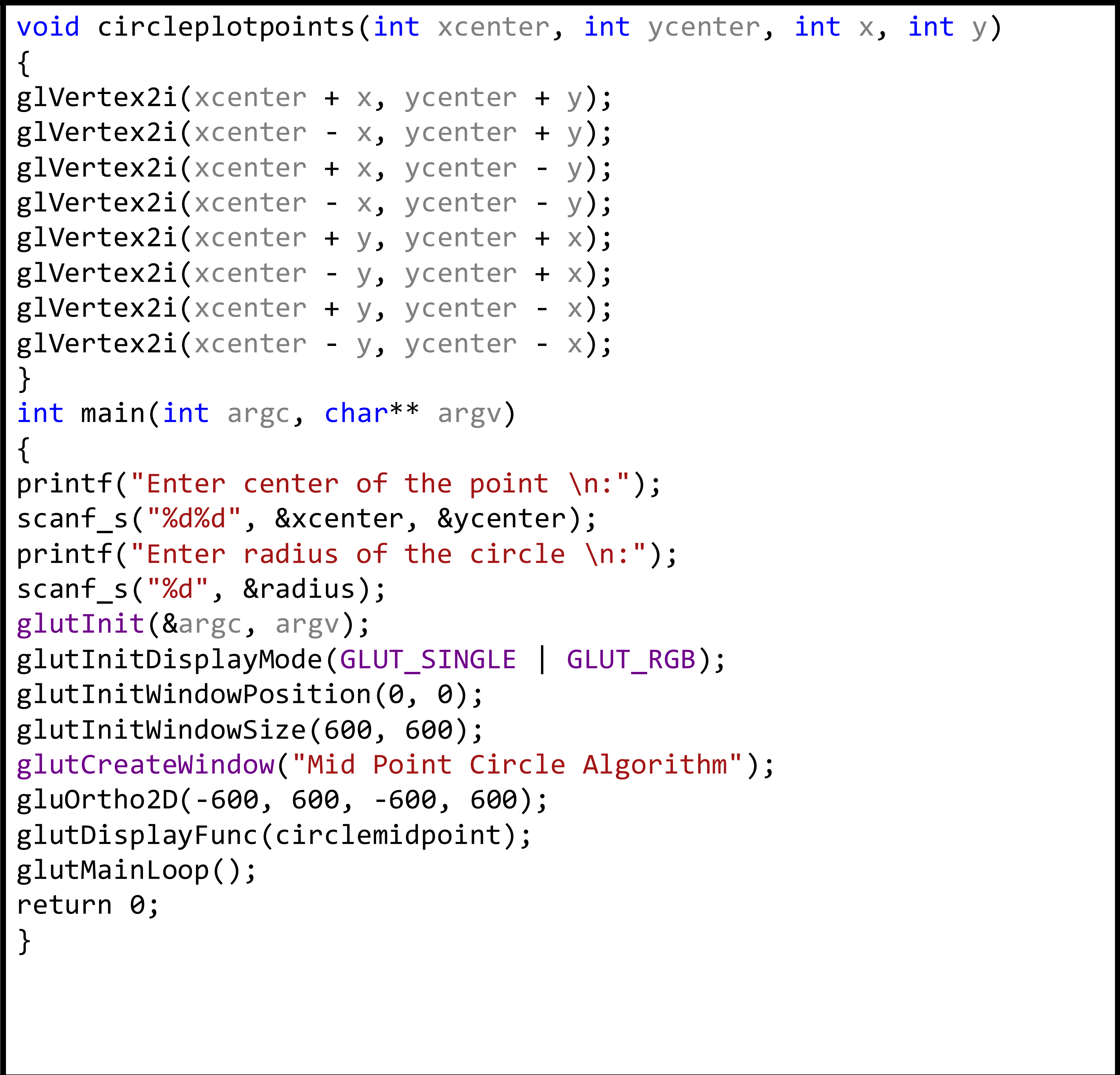
$$\boldsymbol{x} = \boldsymbol{x} + \mathbf{x_c} \ , \ \boldsymbol{y} = \boldsymbol{y} + \boldsymbol{y_c}$$

- ❑ *Repeat step 3 to step 5 until x>= y*

Mid Point Circle Code



```
#include<freeglut.h>
#include<Windows.h>
#include<stdio.h>
int xcenter ,  ycenter,  radius;
void circlemidpoint()
{int x = 0;
int y = radius;
int p = 1 - radius;
void circleplotpoints(int, int, int, int);
glClearColor(1.0, 1.0, 1.0, 1.0);
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1.0, 0.0, 0.0);
glPointSize(5.0);
glBegin(GL_POINTS);
/*make the first set points of a circle */
circleplotpoints(xcenter, ycenter, x, y);
while (x < y)
{x++;
if (p < 0)
{p += 2 * x + 1;
}else
{y--;
p += 2 * (x - y) + 1;
}circleplotpoints(xcenter, ycenter, x, y);
}
glEnd();
glFlush();
}
```



```
void circleplotpoints(int xcenter, int ycenter, int x, int y)
{
glVertex2i(xcenter + x, ycenter + y);
glVertex2i(xcenter - x, ycenter + y);
glVertex2i(xcenter + x, ycenter - y);
glVertex2i(xcenter - x, ycenter - y);
glVertex2i(xcenter + y, ycenter + x);
glVertex2i(xcenter - y, ycenter + x);
glVertex2i(xcenter + y, ycenter - x);
glVertex2i(xcenter - y, ycenter - x);
}
int main(int argc, char** argv)
{
printf("Enter center of the point \n:");
scanf_s("%d%d", &xcenter, &ycenter);
printf("Enter radius of the circle \n:");
scanf_s("%d", &radius);
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
glutInitWindowPosition(0, 0);
glutInitWindowSize(600, 600);
glutCreateWindow("Mid Point Circle Algorithm");
gluOrtho2D(-600, 600, -600, 600);
glutDisplayFunc(circlemidpoint);
glutMainLoop();
return 0;
}
```

The End