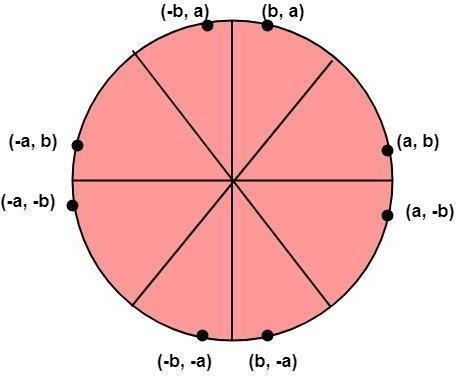
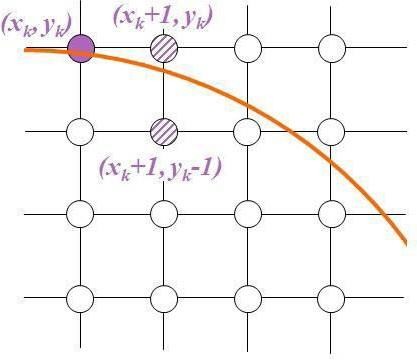
|  |  |
| --- | --- |
| **Name:** | Yash Ravindra Kerkar |
| **Roll No:** |  |
| **Class/Sem:** | SE/III |
| **Experiment No.:** | 3 |
| **Title:** | Midpoint Circle Drawing Algorithm |
| **Date of Performance:** |  |
| **Date of Submission:** |  |
| **Marks:** |  |
| **Sign of Faculty:** |  |

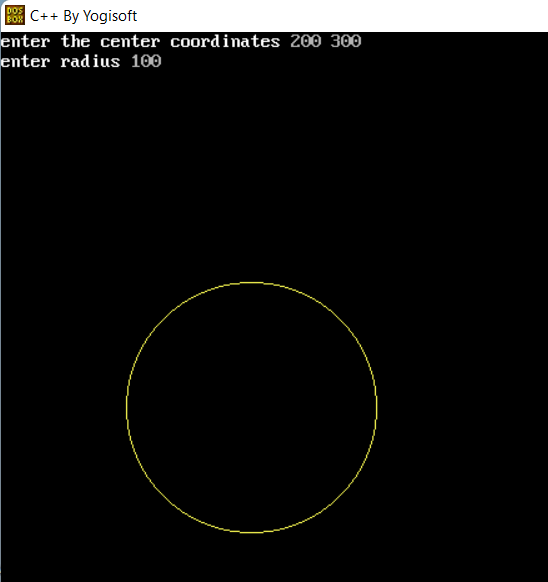
**Experiment no.3**

|  |  |
| --- | --- |
| **Aim** | Write a program to implement Midpoint Circle Drawing Algorithm in C. |
| **Objective** | To implement midpoint circle drawing algorithm for drawing a circlewith radius (R) and centre (Xc, Yc) |
| **Theory** | **Midpoint Circle Algorithm**  Circles have the property of being highly symmetrical, which is handy when it comes to drawing them on a display screen.   * We know that there are 360 degrees in a circle. First we see that a circle is symmetrical about the x axis, so only the ﬁrst 180 degrees   need to be calculated.   * Next we see that it's also symmetrical about the y axis, so now we only need to calculate the ﬁrst 90 degrees. * Finally we see that the circle is also symmetrical about the 45 degree diagonal axis, so we only need to calculate the ﬁrst 45 degrees. * We only need to calculate the values on the border of the circle in the ﬁrst octant. The other values may be determined by symmetry.   Midpoint circle algorithm calculates the locations of the pixels in theﬁrst 45 degrees. It assumes that the circle is centered on the origin. So for every pixel (x, y) it calculates, we draw a pixel in each of the eight octants of the circle. This is done till when the value of the y coordinate equals the x coordinate. The pixel positions for determining symmetry are given in the below algorithm.  It is based on the following function for testing the spatial relationship between the arbitrary point (x, y) and a circle of radius r centered at the origin: |





|  |
| --- |
|  |
| **Algorithm** |
| 1. Plot the initial (xk , yk ) i.e. xk = 0 and yk =r 2. Find the Initial Decision Parameter 3. If Pk < 0, then choose yk+1 = yk    * xk+1 = xk + 1    * yk+1 = yk   ○   1. If Pk > = 0, then choose yk+1 = yk    * xk+1 = xk + 1    * yk+1 = yk – 1      * Repeat 3 and 4 until x becomes greater than or equal to y * To plot the entire circle, use the 8-way   Symmetry  **Code:**  #include<stdio.h>  #include<conio.h>  #include<graphics.h>  void pixel(int x,int y,int xc,int yc)  {  putpixel(x+xc,y+yc,BLUE);  putpixel(x+xc,-y+yc,BLUE);  putpixel(-x+xc,y+yc,BLUE);  putpixel(-x+xc,-y+yc,BLUE);  putpixel(y+xc,x+yc,BLUE);  putpixel(y+xc,-x+yc,BLUE);  putpixel(-y+xc,x+yc,BLUE);    putpixel(-y+xc,-x+yc,BLUE);  }  main()  {  int gd=DETECT,gm=0,r,xc,yc,x,y;  float p;  //detectgraph(&gd,&gm);  initgraph(&gd,&gm," ");  printf("\n Enter the radius of the circle:");  scanf("%d",&r);  printf("\n Enter the center of the circle:");  scanf("%d %d",&xc,&yc);  y=r;  x=0;  p=(5/4)-r;  while(x<y)  {  if(p<0)  {  x=x+1;  y=y;  p=p+2\*x+3;  }  else  {  x=x+1;  y=y-1;  p=p+2\*x-2\*y+5;  }  pixel(x,y,xc,yc);  }  getch();  closegraph();  }  **Output:** |



**Conclusion:**

In this programming exercise, we implemented a C program to draw a circle using the Midpoint Circle Drawing Algorithm. This algorithm is an efficient and widely used method for drawing circles on a computer screen.

The key components of the program include:

1. Initialization of graphics mode using the **initgraph** function from the **graphics.h** library.
2. Accepting the radius (**r**) and center coordinates (**xc** and **yc**) of the circle from the user.
3. Initialization of variables **x** and **y** to start the circle drawing process.
4. Calculation of the decision parameter **p**, which determines the next pixel position based on the midpoint of the circle.