**计算机图形学实验**

**姓 名：白文强**

**学 号：20191060064**

**专 业：计算机科学与技术**

**教 师：钱文华**

实验四 中点Bresenham画圆和椭圆算法

时间：2020年10月23日

地点：信息学院2202机房

1、实验内容

　　使用opengl，用中点bresenham算法画圆和椭圆

2、实验目的

　　使用中点坐标求得判别式D来画圆，根据判别式的正负来取得下一个点的坐标来画圆。将bresenham算法用代码的形式实现，参考课本伪代码

3、实验代码

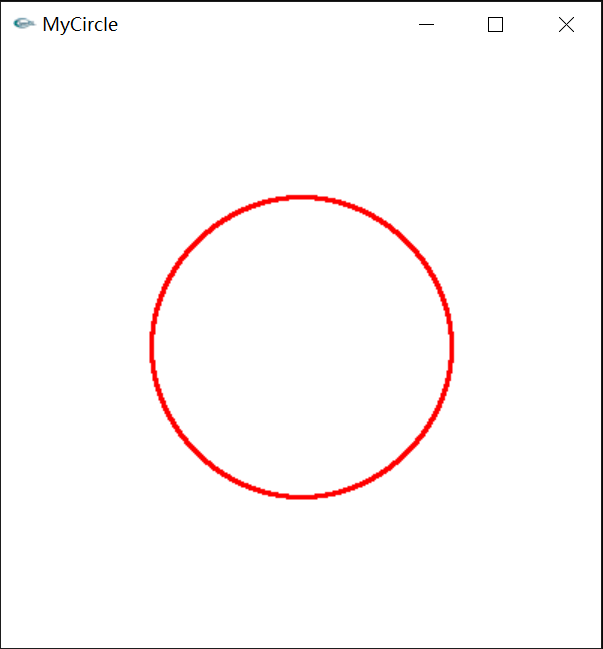
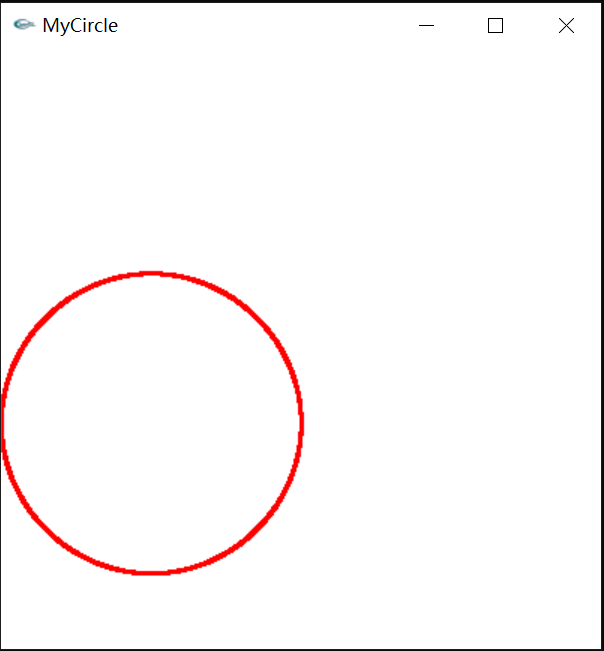
**画圆：**

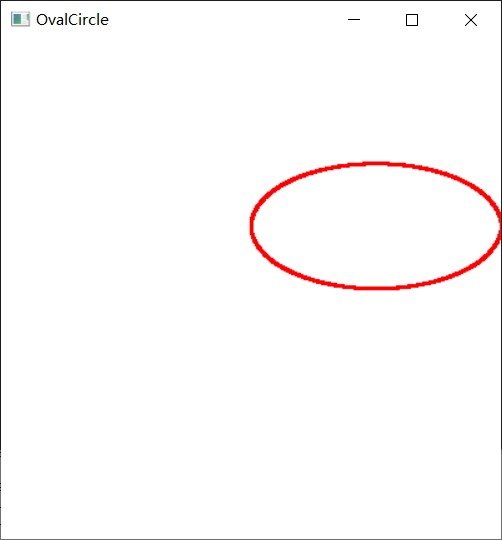
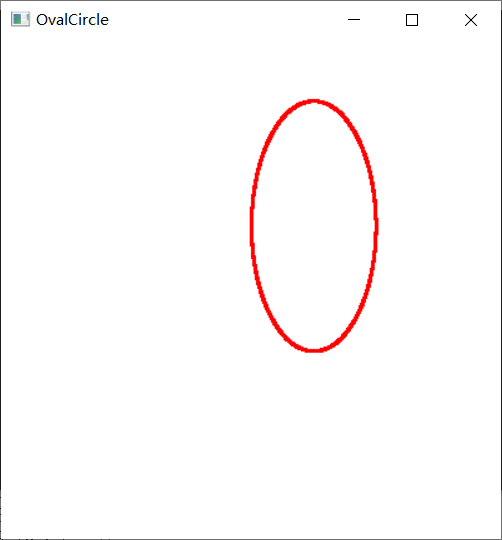
1. #include <windows.h>
2. #include <GL/glut.h>
3. #include <stdlib.h>
4. #include <math.h>
5. #include <stdio.h>
6. **void** init(**void**)
7. {
8. glClearColor(1.0, 1.0, 1.0, 1.0);
9. gluOrtho2D(-200, 200.0, -200, 200.0);
10. }
11. **void** DrawCirclePoint(**int** x0, **int** y0, **int** x, **int** y)
12. {
13. glBegin(GL\_POINTS);
14. glVertex2i(x+x0, y+y0);
15. glEnd();
16. }
17. **void** CircleBresenHam(**int** x0, **int** y0, **int** R)
18. {
19. **float** d = 1.25 - R;
20. **int** x = 0;
21. **int** y = R;
22. DrawCirclePoint(x0,y0,x, y);    //坐标根据(x0,y0)平移
23. **for** (**int** i = 0; i < sqrt(2)\*R / 2; i++)
24. {
25. x++;
26. **if** (d > 0)
27. {
28. y--;
29. d += 2 \* (x - y) + 5;
30. }
31. **else**
32. d += 2 \* x + 3;
33. DrawCirclePoint(x0,  y0,  x, y);
34. DrawCirclePoint(x0,  y0,  x, -y);
35. DrawCirclePoint(x0,  y0, -x, y);
36. DrawCirclePoint(x0,  y0, -x, -y);
37. DrawCirclePoint(x0,  y0,  y, x);
38. DrawCirclePoint(x0,  y0, -y, x);
39. DrawCirclePoint(x0,  y0,  y, -x);
40. DrawCirclePoint(x0,  y0, -y, -x);
41. }
42. }
43. **void** CircleSegement(**void**)
44. {
45. glClear(GL\_COLOR\_BUFFER\_BIT);   // 清空显示窗口
46. glColor3f(1.0, 0.0, 0.0);        // 指定前景色
47. glPointSize(3.0);                //点的大小
48. CircleBresenHam(0, 0, 100);      //画圆
49. glFlush();
50. }
51. **int** main(**int** argc, **char**\*\* argv)
52. {
53. glutInit(&argc, argv);
54. glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);
55. glutInitWindowPosition(100, 100);
56. glutInitWindowSize(400, 400);
57. glutCreateWindow("MyCircle");
58. init();
59. glutDisplayFunc(CircleSegement);
60. glutMainLoop();
61. **return** 0;
62. }

**画椭圆：**

1. #include <windows.h>
2. #include <GL/glut.h>
3. #include <stdio.h>
4. #include <stdlib.h>
5. #include <math.h>
6. **void** init(**void**)
7. {
8. glClearColor(1.0, 1.0, 1.0, 1.0);
9. gluOrtho2D(-400, 400.0, -400, 400.0);
10. }
11. **int** Myround(**double** x)
12. {
13. **return** (**int**)(x+0.5);
14. }
15. **void** DrawCirclePoint(**int** x0, **int** y0, **int** x, **int** y)
16. {
17. glBegin(GL\_POINTS);
18. glVertex2i(x+x0, y+y0);
19. glEnd();
20. }
21. **void** BresenhamOval(**int** xc, **int** yc, **int** a, **int** b)
22. {
23. **int** sqa = a \* a;
24. **int** sqb = b \* b;
26. **int** x = 0;
27. **int** y = b;
28. **int** d = 2 \* sqb - 2 \* b \* sqa + sqa;
29. DrawCirclePoint(xc, yc, x, y);
30. **int** P\_x = Myround( (**double**)sqa/sqrt((**double**)(sqa+sqb)) );
31. **while**(x <= P\_x)
32. {
33. **if**(d < 0)
34. {
35. d += 2 \* sqb \* (2 \* x + 3);
36. }
37. **else**
38. {
39. d += 2 \* sqb \* (2 \* x + 3) - 4 \* sqa \* (y - 1);
40. y--;
41. }
42. x++;
43. DrawCirclePoint(xc, yc, x, y);
44. DrawCirclePoint(xc, yc, x, -y);
45. DrawCirclePoint(xc, yc, -x, y);
46. DrawCirclePoint(xc, yc, -x, -y);
47. }
48. d = sqb \* (x \* x + x) + sqa \* (y \* y - y) - sqa \* sqb;
49. **while**(y >= 0)
50. {
51. DrawCirclePoint(xc, yc, x, y);
52. DrawCirclePoint(xc, yc, x, -y);
53. DrawCirclePoint(xc, yc, -x, y);
54. DrawCirclePoint(xc, yc, -x, -y);
55. y--;
56. **if**(d < 0)
57. {
58. x++;
59. d = d - 2 \* sqa \* y - sqa + 2 \* sqb \* x + 2 \* sqb;
60. }
61. **else**
62. {
63. d = d - 2 \* sqa \* y - sqa;
64. }
65. }
66. }
67. **void** CircleSegement(**void**)
68. {
69. glClear(GL\_COLOR\_BUFFER\_BIT);   // 清空显示窗口
70. glColor3f(1.0, 0.0, 0.0);       // 指定前景色
71. glPointSize(3.0);               //点的大小
72. BresenhamOval(200, 100, 200,100); //画圆
73. glFlush();
74. }
75. **int** main(**int** argc, **char**\*\* argv)
76. {
77. glutInit(&argc, argv);
78. glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);
79. glutInitWindowPosition(100, 100);
80. glutInitWindowSize(400, 400);
81. glutCreateWindow("MyCircle");
82. init();
83. glutDisplayFunc(CircleSegement);
84. glutMainLoop();
85. **return** 0;
86. }

4、实验结果

5、实验总结

1/8法画圆，画出八分之一的圆弧，然后可通过点的对称画出剩余的7/8的圆，从而合成整个圆。对于圆心不在原点的圆，可以根据原点的位置，通过把圆心在原点的圆平移的方式画出。

由于圆心在原点的椭圆不具有根据y=x对称的性质，所以画椭圆时，不可采用1/8法，只可采用1/4法画椭圆。在第一象限，以斜率绝对值1为分界线，斜率绝对值大于1时，以y坐标为主动，每次减一计算x的取值，斜率绝对值小于1时，以x坐标为主动，每次加一计算y的取值。其他象限的点，可以根据对称的性质画出来。