Assignment 1

Name: Xucheng Tang NetID: xt544

A. Based on Bresenham's scan-conversion algorithm for a line segment, we can derive this algorithm for a circle. Here is the step to implement it.

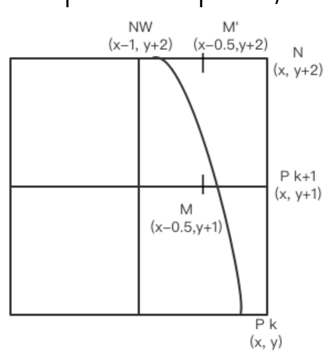
- Split the circle to **eight** parts with two axes and two lines x = y, x = −y. Thus we can only consider the region from A(r, 0) to B(r/√2, r/√2) which can be drawn to other seven regions with corresponding symmetric points.
- Part I

  When we start from the right border of the circle, what we know is that the coordinate of the start point A is (r, 0). Let's take a look for the **first selection** where the next block will be. As we can known from the assignment, we are supposed to increase the y value by 1 and judge whether the midpoint of N(r, 1) and NW(r-1, 1), so the midpoint M's coordinate is (r-0.5, 1).

  To judge which candidate will we pick, we apply M to the function $F(x, y) = x^2 + y^2 - r^2$. $D_{first} = F(M_{start}) = (r - \frac{1}{2})^2 + 1^2 - r^2 = -r + \frac{5}{4}$.
- Part II

  Now we are going to compute for a formula for D_{New} from D_{old} there are two situations to be considered. The first one is when we choose point N as previous point. The second one is when we choose point NW. I will discuss them as two cases.

① When we choose **point N** as previous point, then the situation will be:



What we can know from this situation is: $D_{old} = (x - \frac{1}{2})^2 + (y+1)^2 - r^2 > 0$

So for the new selection, the formula is:

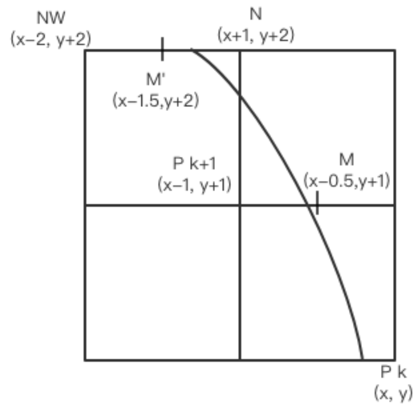
$D_{new} - D_{old} = [(x - \frac{1}{2})^2 + (y+2)^2 - r^2] - [(x - \frac{1}{2})^2 + (y+1)^2 - r^2] = 2y + 3$.

Because we take P(k) (x, y) as old point, but the next selection will based on P(k+1) (x, y+1), so we can simply use the new point as induction as below:

$$D_{new} = D_{old} + 2(y-1) + 3 = D_{old} + 2y + 1$$

If D_{new} is greater than 0, then M' is outside the circle, we will choose NW as new block.
If D_{new} is smaller than 0, then M' is inside the circle, we will choose W as new block.
Since y is a integer, so this calculation will not involve with non-integer term.

② When we choose **point NW** as previous point, then the situation will be:



We can tell from this case that: $D_{old} = (x - \dfrac{1}{2})^2 + (y+1)^2 - r^2 < 0$.

So the formula is:

$$D_{new} - D_{old} = [(x - \tfrac{3}{2})^2 + (y+2)^2 - r^2] - [(x - \tfrac{1}{2})^2 + (y+1)^2 - r^2]$$
$$= -3x + \tfrac{9}{4} + 4y + 4 + x - \tfrac{1}{4} - 2y - 1 = -2x + 2y + 5$$

Because we take P(k) (x, y) as old point, but the next selection will based on P(k+1) (x-1, y+1), so we can simply use the new point as induction as below:

$$D_{new} = D_{old} + 2(y - 1 - (x+1)) + 5 = D_{old} + 2(y - x) + 1$$

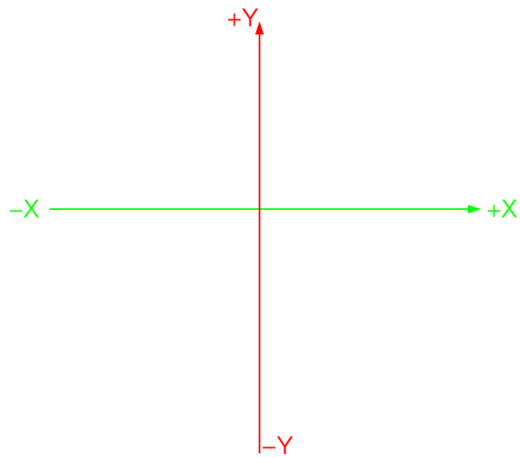
If D_{new} is greater than 0, then M' is outside the circle, we will choose NW as new block.
If D_{new} is smaller than 0, then M' is inside the circle, we will choose W as new block.
Since x and y are integers, so this calculation will not involve with non-integer term.

- **When finished the computation of D_new, we can figure out that the computation will not involve with non-integer term. But the initial D will have 1/4 add to all the results, to eliminate the fraction from the equation, we simply round off the fraction and let the D_{start}= 1-r.**

So the computational formula will be:
Start from A(r, 0). Increase y by 1 and decrease x according to the circle.
While x≥y: (to ensure all points are in arc AB)

$$D_{new} = \begin{cases} 1 - r & when\ D_{start} \\ D_{old} + 2y + 1 & when\ choose\ N \quad (D_{old} < 0) \\ D_{old} + 2(y - x) + 1 & when\ choose\ NW\ (D_{old} \geq 0) \end{cases}$$

B. After running this code, I figure out OpenGL coordinate system works as follows.
The origin is lower-left corner of the window, where coordinate is (0, 0).
The X orientation is towards right when x value increasing.
The Y orientation is towards left when y value increasing.
So the screen coordinate system is as below.



For pixels at (x, y) where either x<0 or y<0, **I can not display it** even if I expand the window size. Because the origin of the window is set to (0, 0), if I set one or two arguments of coordinate of a pixel to negative, the point will always left or below the window, which will not be displayed on the screen.

C. D. E. Implementation of the instruction is in the **main.cpp** file.
Here are some instructions for running the code.

Part C

When run the main.cpp file, three options will list as below:

```
please input the select fucntion:
1 for part(c) 2 for part(d) and 3 for part(e):
```

So you need to choose one part to run, part 1 is for static circle based on the input. You need to input three integers to indicate the x coordinate, y coordinate and the radius of the circle. The input should separated by one space in between.
For example, I type in three integers like this:

```
1 for part(c) 2 for part(d) and 3 for part(e):
1
The window size is (600, 600)
please input the value of x, y and radius r (separate by space):
300 300 200
```

Then the circle will be displayed and (x, y) = (300, 300) and radius is 200 pixels, as Image 1.

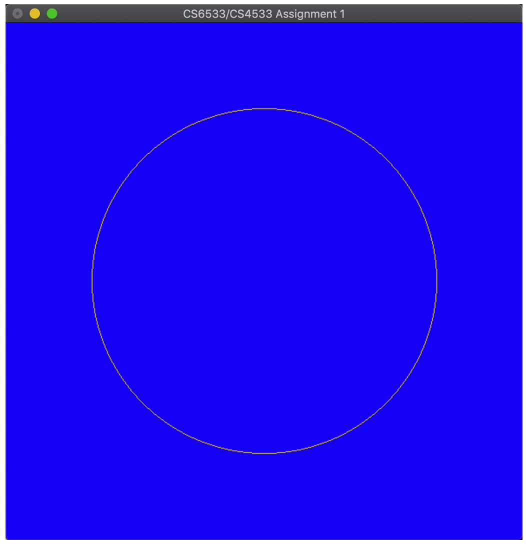


Image 1

Part D

When we choose the part D, the input value is 2.
Then we will get the output line and the picture like this:

```
1 for part(c) 2 for part(d) and 3 for part(e):
2
The max window size is 755.
Circles are displayed.
```

The output max window size is indicating the original maximum size of input which will be used in transforming the coordinate system.
The image of result will be like Image 2.

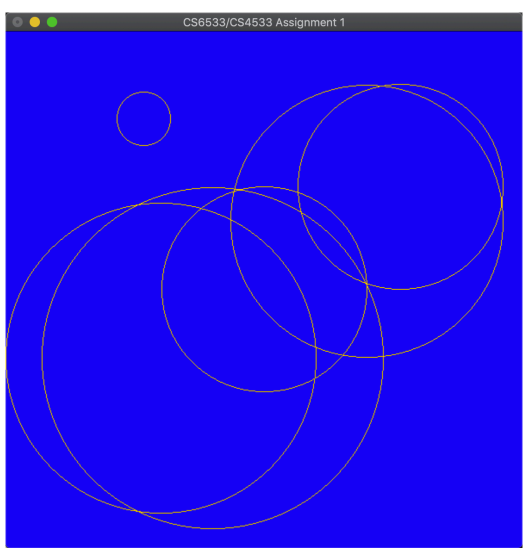


Image 2

Part E

When we choose the part E, the input value is 3.

Then we will get the output line and the picture like this:

```
please input the select fucntion:
1 for part(c) 2 for part(d) and 3 for part(e):
3
K value = 1000.
Circles are displayed.
```

I set the K value as 1000 manually. It is smooth and not create motion blur.

The screenshots of result will be like Image 3 and Image 4.

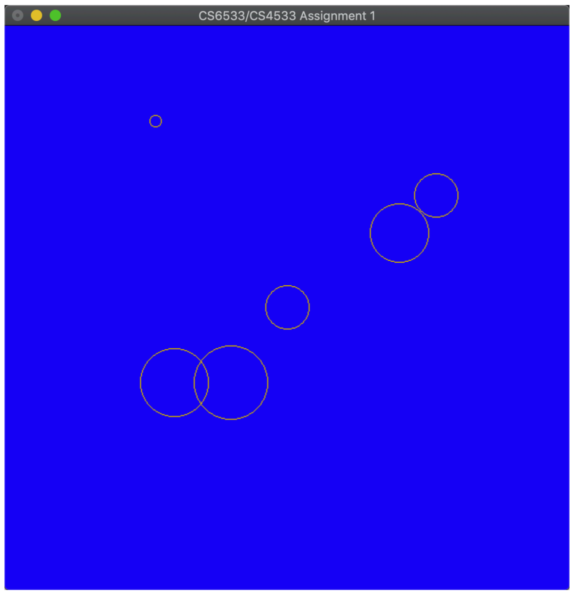


Image 3

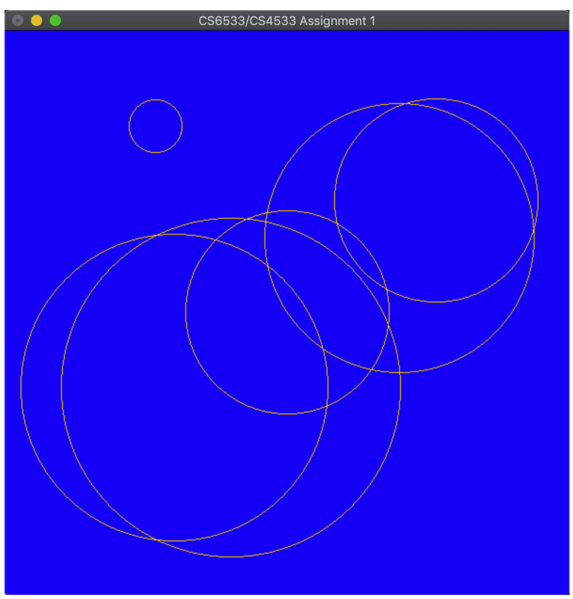


Image 4