UTM
UNIVERSITI TEKNOLOGI MALAYSIA

SCHOOL OF COMPUTING

SESSION 2020/2021 SEMESTER 2

SECV2213 - FUNDAMENTAL OF COMPUTER GRAPHICS

SECTION 01

ASSIGNMENT 2 - OUTPUT PRIMITIVES

GROUP MEMBERS:

1. CHONG HONG LEI (A19EC0035)
2. KOH XIN YI (A19EC0064)
3. NUR HIDAYAH BINTI HAMRI (B19EC0046)

LECTURER'S NAME: DR GOH EG SU

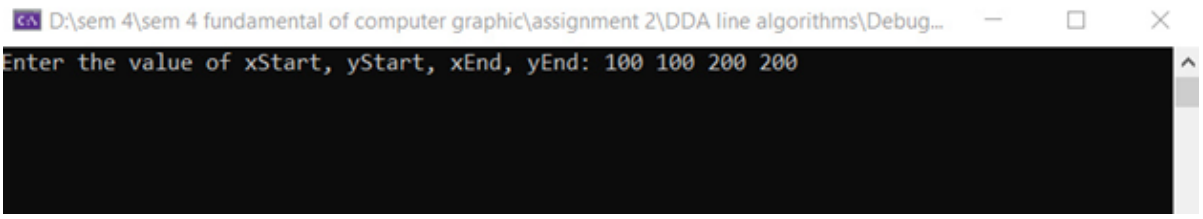
LINE (DDA line algorithm.cpp)

Adjustment

To get user input, *iostream*, a standard C++ library header file is added in the program. Users can enter the value of starting pixel *(xStart, yStart)* and ending pixel *(xEnd, yEnd)* from the keyboard. The code is written as follow:

```
// get input of starting point and end point from user
     cout << "Enter the value of xStart, yStart, xEnd, yEnd: ";
     cin >> xStart >> yStart >> xEnd >> yEnd;
```

The output screen executed after successfully run the program:



For the algorithm part, first, two integer points (starting point and ending point) are declared at global to accept the input value from users. Horizontal and vertical differences between the endpoint positions are assigned to parameters dx and dy. Without considering the sign, if (dx > dy) the number of steps will be equal to dx, else it will equal to dy. The value of x and y increments is then calculated in order to generate the next pixel position at each step along the line path. To print the points, a for loop is used. The value of the pixel is getting by adding the value of increment to the previous pixel.

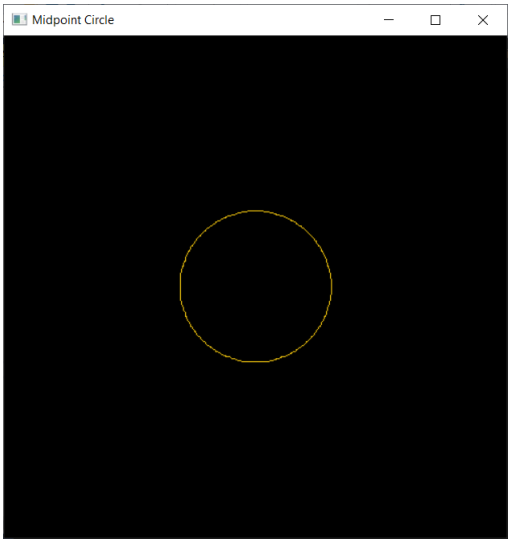
Achievement

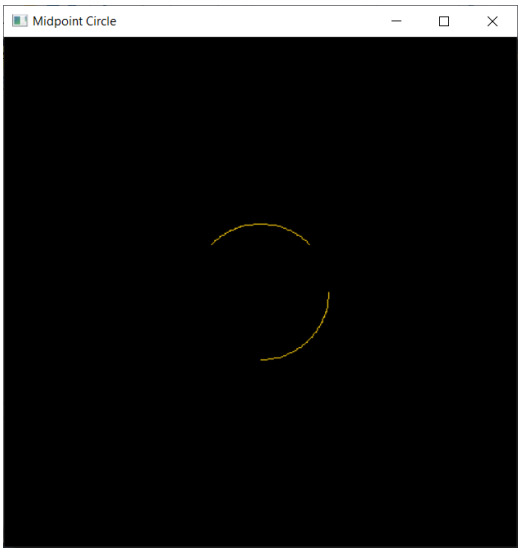
Based on the DDA line algorithm code, it accepts four integer screen positions for starting point *(xStart, yStart)* and ending point *(xEnd, yEnd)* of line segment. The points then pass to *lineDDA* function to generate the next pixel position along the path. The line will draw from the starting point until the end point after *drawMyLine* function is called. Through the activity, I had learned to implement a DDA line algorithm to display line segments. Besides, I had also managed to add user input using keyboard in openGL code.

MIDPOINT CIRCLE ALGORITHM (Midpoint Circle Algorithm.cpp)

Adjustment

For the adjustment, I changed 4 way symmetry to 8 way symmetry because it makes the circle form faster and look full compared to 4 way symmetry. At the left is the 8 way symmetry and at the right is 4 way symmetry





First I declare the x=0, y, p, dx=0, dy=0 and r then we calculate the p to get the value pk+1. After we get the pk+1, then we can decide which plot to go next if the condition at $(p < 0)$ then run the calculation $p = p + (2 * x) + 1$ (which p is the calculation for the radius) same also when the condition at $(p>0)$ then run the calculation $p = p + (2 * x) + 1 - (2 * y)$.

The important thing that we need to alert during calculating the pk+1 is when $(p < 0)$ then the value x needs to add 1 but for the value y stay at y (x+1 , y). If the $(p > 0)$, the value x needs to add 1 and this time the value y subtract 1 (x+1 , y-1).

And I have also changed the way to get the radius by asking the user to enter a radius value

```
cout <<"Enter radius: ";  //user enter radius
cin >> r;
y = r;
```

Achievement

My achievement during completing this Assignment 2 is it makes me more understand the function of Midpoint Circle and how to calculate the pk+1 and also make me more understand where I need to find the number and put it in the table like previous exercise in the class. I can also distinguish between condition (p <0) and (p> 0) more easily after finishing this Assignment 2.

ELLIPSE (Ellipse.cpp)

Adjustment

In assignment 2, I learned to write a program of an ellipse, which includes the ellipseMidpoint method, ellipsePlotPoints method and displayEllipse method. The four variables, xc (x coordinate of the ellipse center), yc (y coordinate of the ellipse center), rx (x radius of the ellipse) and ry (y radius of the ellipse) are declared in line 12 to allow the program reads the values entered by the user. Before calculating each point of the ellipse, I declare some int variables, dx, dy, p, x and y. p represents the decision parameter, $p1_k$ at region 1 and the decision parameter, $p2_k$ at region 2. dx and dy represent $2r_y^2r_x$ and $2r_x^2r_y$ respectively as defined in line 39 and 40. I insert two more variables which are rx2 and ry2 as the calculation involves a lot of r_x^2 and r_y^2. This allows the equation shorter and clearer to understand. At the starting point, the value of x is initialized to 0 and the value of y is initialized to ry. The void method name ellipsePlotPoint to calculate the coordinate of the point in each quadrant is defined so that the ellipseMidpoint can call the function to display the points.

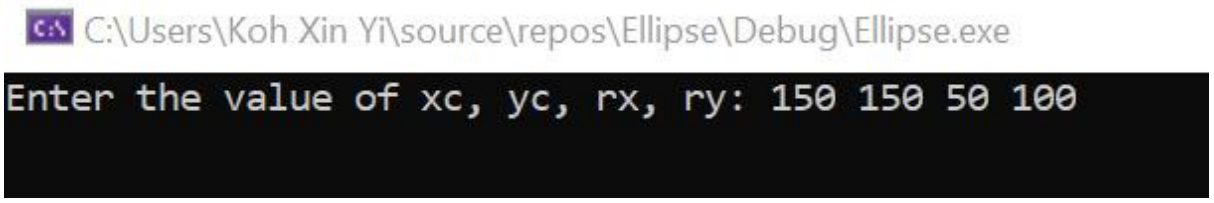
The ellipse point calculation is divided into two parts which are region 1 and region 2. In region 1, the initial decision parameter is calculated in line 49. The x coordinate of the next point is always an increment of the x coordinate of the current point (x++) as written in line 51. The y coordinate of the next point is determined by the value of p, if the p is negative, the y coordinate of the next point remains unchanged, else the y coordinate of the next point is the decrement of the y coordinate of the current point (y--) as written in line 56. Region 1 ends when the dx ($2r_y^2r_x$) is greater than dy ($2r_x^2r_y$) (while dx < dy). The ellipsePlotPoints function is called every time in the while loop to generate the point of the ellipse in region 1 for every quadrant.

In region 2, the initial decision parameter of region 2 is calculated in line 63. The y coordinate of the next point is always a decrement of the y coordinate of the current point (y--) as written in line 65. The x coordinate of the next point is determined by the value of p, if the p is positive, the x coordinate of the next point remains unchanged, else the x coordinate of the next point is the increment of the x coordinate of the current point (x++) as written in line 70. Region 2 ends when the y is smaller than 0 (while y > 0). The ellipsePlotPoints function is also called every time in the while loop to generate the point of

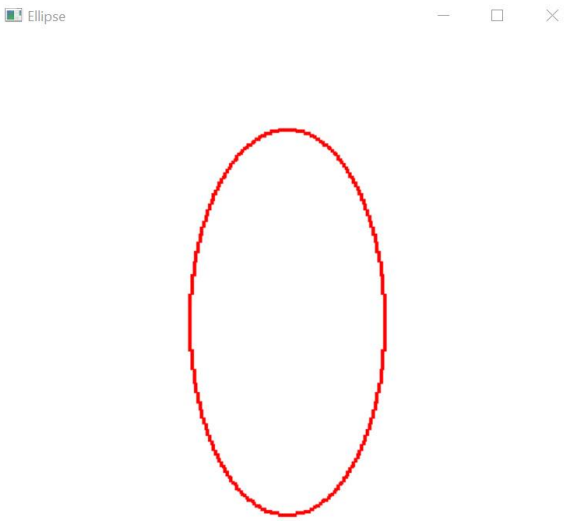
the ellipse in region 2 for every quadrant. In the void ellipsePlotPoint function, start from line 77 to line 83, receives the value passed by the ellipseMidpoint function, and calculates the corresponding coordinate of the point in each quadrant. For the void displayEllipse function, it allows the programmer to set the value of point colour, point size, coordinates of ellipse center and the radii of the ellipse. At the main function, the program asks the user to enter the value of x and y coordinate of the ellipse center and the x and y radii of the ellipse.

Output

The user enters the value xc = 150, yc = 150, rx = 50 and ry = 100.



An ellipse of center point (150, 150), rx = 50, ry = 100 is displayed. The point size I use is 3.



Achievement

After completing assignment 2, I had learnt how to convert the equation of the ellipse I learned in class into program code. The while loop can be used to determine whether the region is ended. I believe the knowledge I learned from this assignment will be useful for my future attempts to draw objects.