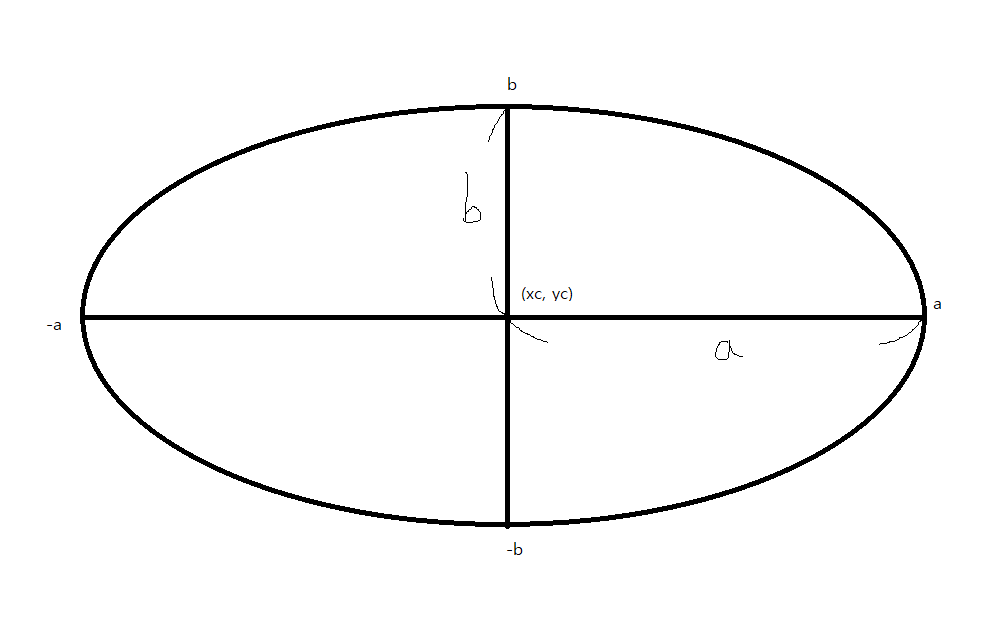
Ellipse Drawing

(Assumption)



Ellipse has center (xc, yc),

semi-major axis length a,

semi-minor axis length b

Implicit equation of ellipse

F(x, y) = (x – x1)^2 / a^2 + (y – y1)^2 / b^2 – 1 = 0

F(x, y) = b^2(x-x1)^2 + a^2(y-y1)^2 – a^2\*b^2 = 0

Introduction :

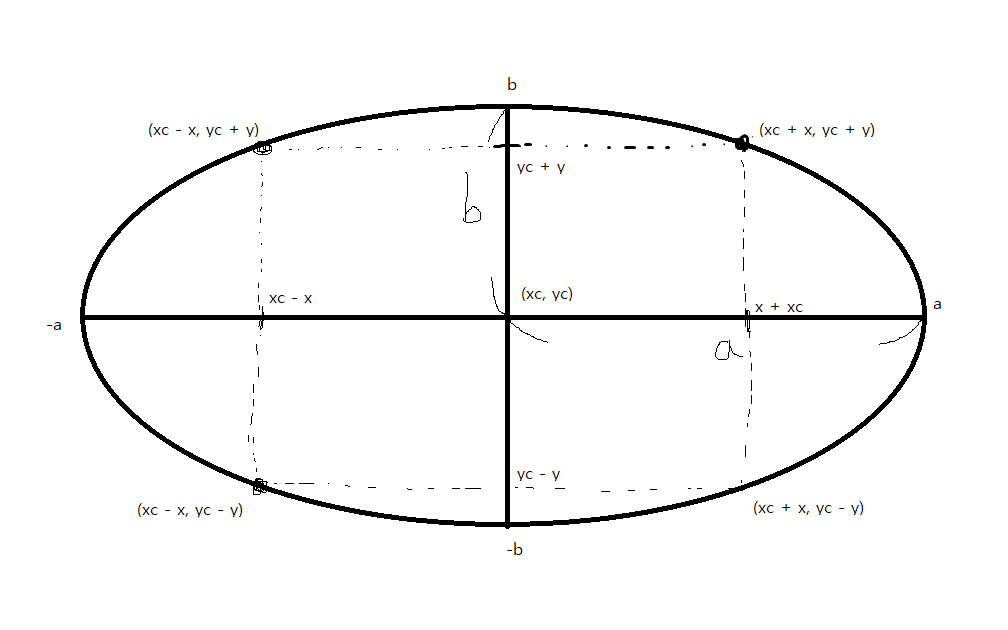
To draw ellipse by four – way symmetry,

I only calculate (x, y)

(I assume that the center of ellipse is origin (0,0) for simply calculating)

and then draw (xc + x, yc + y), (xc + x, yc – y), (xc – x, yc + y), (xc – x, yc – y)

( It means that I move the assumed center of ellipse (0,0) to real center of ellipse (xc, yc) )



Therefore, to draw one ellipse, I only need to calculate (x, y) on Quadrant1 (x > 0, y > 0)

In calculating part ( x and y on Quadrant1), I need to divide into two sections,

one with tangent line’s slope is bigger than 1(m > 1),

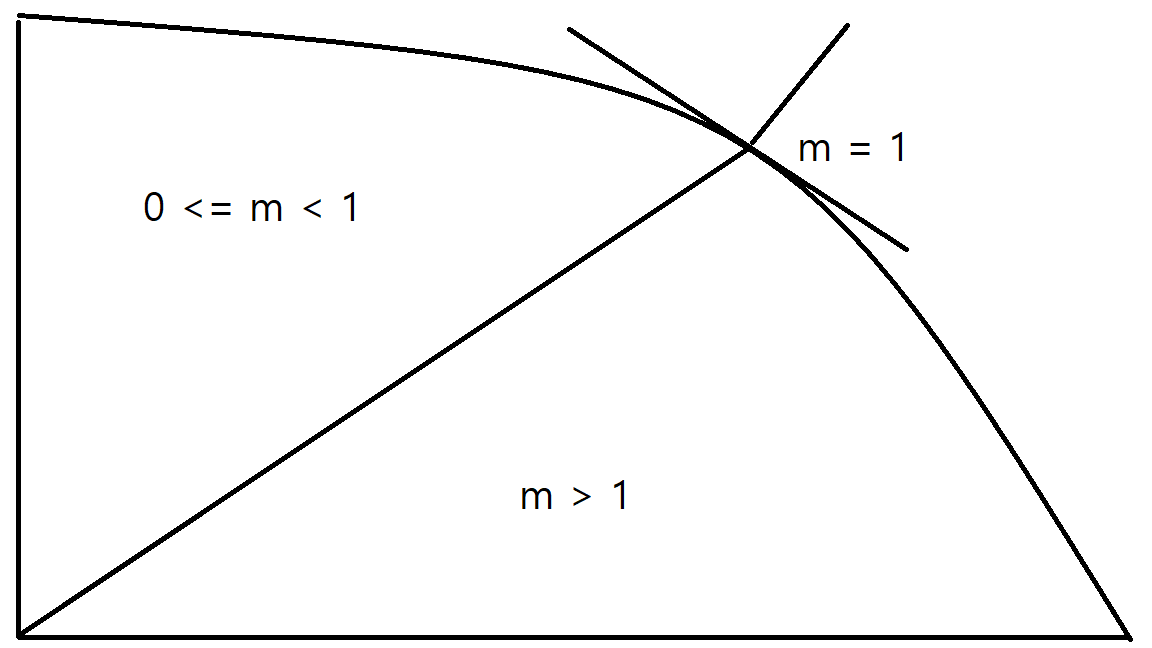
the other with tangent line’s slope is less than 1(0<=m<1).

( because if slope is bigger than 1, I need to draw by y-axis while the rest of part is drawn by x-axis )

F(x, y) = b^2\*x^2 + a^2\*y^2 – a^2\*b^2 = 0

F’(x,y) = 2(b^2)(x) + 2(a^2)(y) \* dy/dx = 0

dy / dx = -b^2\*x / a^2 \* y

So I need to calculate them differently 

Part 1 :

The picture above shows that x is increased and y is decreased. So to use midpoint algorithm, we need to find out the determinant1 d1 = F(x + 1, y – 1/2) on the section that satisfies 0 <= m < 1

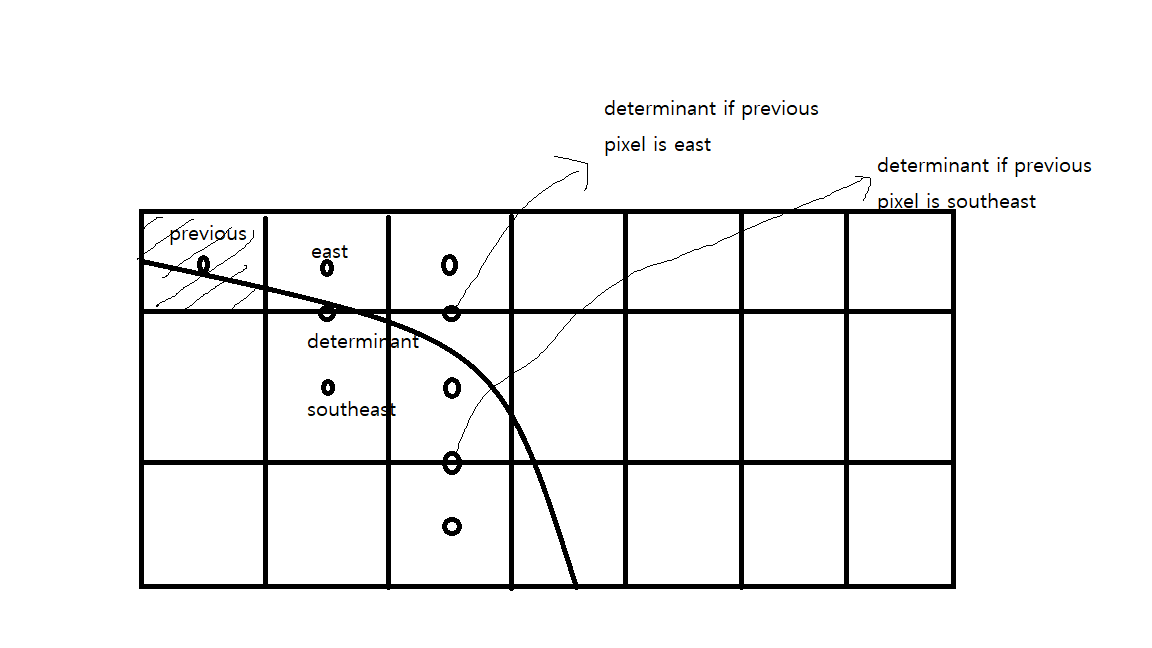
and the determinant2 d2 = F(x + 1/2, y – 1) on the section to satisfies m > 1

F(x + 1, y – 1/2) = -

F(x + 1/2, y – 1) = -

If d >= 0, I will draw east pixel of previous one,

If d < 0, I will draw southeast pixel of previous one.

Therefore, the determinant of next one depends on where I draw. 

If previous pixel is east,

Case 1: 0 <= m < 1

F(x + 1, y – 1/2) = -

F(x + 2, y – 1/2) = -

d +=

Case 2: m > 1

F(x + 1/2, y – 1) = –

F(x + 1/2, y – 2) = -

d += (-2y + 3)

If previous pixel is southeast

Case 1: 0 <= m < 1

F(x + 1, y – 1/2) = -

F(x + 2, y – 3/2) = -

d += (-2y + 2)

Case 2: m > 1

F(x + 1/2, y – 1) = –

F(x + 3/2, y – 2) = -

d += (-2y + 3) (2x + 2)

Since the starting point is (0, b), first d1 = F(0 + 1, b – 1/2)

F(0 + 1,b) = + (-b + 1/4)

So, to use all of these in condition, pseudo code is like

Void DrawEllipse() {

// semi-major axis length a, semi-minor axis length b

int a,b;

int x = 0, y = b;

//d1 is the determinant on section1 (0 <= m < 1)

int d1 = b\*b – a\*a\*(-b + 1/4);

// To draw first point of ellipse

Draw(x,y,color);

//in the section that satisfies 0 <= m < 1

While(a\*a\*(y-0.5) > b\*b(x+1)) {

//Draw East pixel

If(d1 < 0)

//Calculate next determinant

d1 += b\*b\*(2x + 3);

//Draw SouthEast pixel

else {

d1 += b\*b\*(2x+3) + a\*a\*(-2y+2);

y—

}

x++;

DrawFourParts();

}

//d2 is determinant on the section2 ( m > 1 )

Int d2 = b\*b\*(x+0.5)^2 + a\*a\*(y-1)^2 – a\*a\*b\*b;

//In the section that satisfies m > 1

while(y > 0) {

//Draw southeast pixel

if(d2 < 0) {  
 d2 += b\*b\*(2x+2) + a\*a\*(-2y + 3);

x++

}

Else

d2 += a\*a(-2y+3);

y—

DrawFourParts();

}

}

Part2 : Second-order difference

Unlike primary polynomial like lines, secondary polynomial like circles and ellipses remains a variable if I get the determinant. It can be expressed in += Ax + B form, which is slower than += A. So, I want to use second order difference to change += Ax + B form into += SOD and SOD += A

Assume that

f(x) = A

Then its difference is f(x + -f(x)

(x) = (A ) + C )

- (A)

Let is 1 ( like calculating determinant by x-axis )

(x) = 2Ax + A + B

Now, I can see difference express +=Ax + b form

I use difference of difference(second order difference) to make it += SOD and SOD += A form

(x) = (x + ) - (x)

(x) = 2A ( I let )

So I can change the form

f(x) += (x)

(x) += (x)

Part 3 : extend Part1 using Part2

Part1’s difference(determinant is using difference) is += Ax + b form, so I can change this into += SOD and SOD += A form.

If previous pixel is east,

Case 1: 0 <= m < 1

F(x + 1, y – 1/2) = -

F(x + 2, y – 1/2) = -

(first point’s)dE +=

d1 += dE

dE += 2

Case 2: m > 1

F(x + 1/2, y – 1) = –

F(x + 1/2, y – 2) = -

(first point’s)dE += (-2y + 3)

d2 += dE

dE += 2

If previous pixel is southeast

Case 1: 0 <= m < 1

F(x + 1, y – 1/2) = -

F(x + 2, y – 3/2) = -

(first point’s)dSE += 3(-2b + 2)

d1 +=

dSE += 2+ 2

Case 2: m > 1

F(x + 1/2, y – 1) = –

F(x + 3/2, y – 2) = -

(first point’s)dSE += (-2y + 3)

d2 +=

dSE += 2+ 2

Part4 : write the pseudo-code

Void DrawEllipse() {

// semi-major axis length a, semi-minor axis length b

int a,b;

int x = 0, y = b;

//d1 is the determinant on section1 (0 <= m < 1)

int d1 = b\*b – a\*a\*(-b + 1/4);

//Set the first difference

int dE +=

int dSE += 3(-2b + 2)

// To draw first point of ellipse

Draw(x,y,color);

//in the section that satisfies 0 <= m < 1

While(a\*a\*(y-0.5) > b\*b(x+1)) {

//Draw East pixel

If(d1 < 0) {

//Calculate next determinant

d1 += dE;

dE += 2\*b\*b;

dSE += 2\*b\*b;

}

//Draw SouthEast pixel

else {

d1 += dSE;

dSE += 2\*b\*b+ 2\*a\*a;

dE += 2\*b\*b;

y—

}

x++;

DrawFourParts();

}

//d2 is determinant on the section2 ( m > 1 )

Int d2 = b\*b\*(x+0.5)^2 + a\*a\*(y-1)^2 – a\*a\*b\*b;

//Set the first difference next to (m = 1)

int dE = (a\*a)\*(-2y + 3);

int dSE = (2y + 3);

//In the section that satisfies m > 1

while(y > 0) {

//Draw southeast pixel

if(d2 < 0) {  
 d2 += dSE;

dE += 2\*a\*a;

dSE += 2\*b\*b+ 2\*a\*a;

x++

}

Else {

d2 += dE;

dE += 2\*a\*a;

dSE += 2\*a\*a;

}

y—

DrawFourParts();

}}

**Declaration: CS200**

**Name: Hwang Chan Il\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Login Name: c.hwang\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**SRS Student ID: 5415279\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# Non-Programming Portion: Worth 50 points

\_o\_1: Correct information was handed in: declaration page, derivation pages, and readme page. **If not, penalty is** -**20 points.**

\_o\_2: Derivation of *integer-only second-order differences* based Midpoint ellipse algorithm from first principles. **Worth 50 points.**

# Programming Portion: Worth 50 points

\_o\_1: Student program compiles, links and executes. **If not, penalty is** -**50 points.**

\_o\_2: Filled ellipse scan conversion using second-order differences integer Midpoint ellipse algorithm. **Worth 50 points.**

**DECLARATION: Hwang Chan IL**

I have read the statements regarding cheating in both the CS200 course handout and DigiPen student handbook. I affirm with my signature that this is my own solution to A3 and represents my own work.

**Signature: Hwang Chan IL**