Computer Graphics HW01

소프트웨어학부
소프트웨어학과
2017012197
여채린

문영식 교수님

1. 알고리즘 유도 및 설명

For drawing parabola, we should use midpoint algorithm. Summary of midpoint algorithm is as follows.

First, find proper assumptions/range so that the next point is a binary decision.

Then, derive the curve equation $f_{para}(x, y) = 0$. At this time, our parabola equation is $y = \frac{1}{100} x^2$, so $f_{para}(x, y) = 100y - x^2 = 0$.

Next, set up the decision function p_k.

At this time, $p_k = f_{para}(x_k + 1, y_k - 0.5) = \frac{1}{100}(x_k + 1)(x_k + 1) - (y_k - 0.5)$. Decision function is very important.

Then, test the sign of p_k.

Then, select the next point out of 2 candidates, depending on the sign of p_k. For example, next point can be E(East point) or SE (South East point)

Selections are as follows : $p_k = 0$, if (x, y) is on the parabola boundary. $p_k < 0$, if (x, y)is inside the parabola boundary. $p_k > 0$, if (x, y) is outside the parabola boundary.

Next, for each selection(If $p_k > 0$, else if $p_k <= 0$.), update p_k -1 using p_k. (The update equation should be derived.)

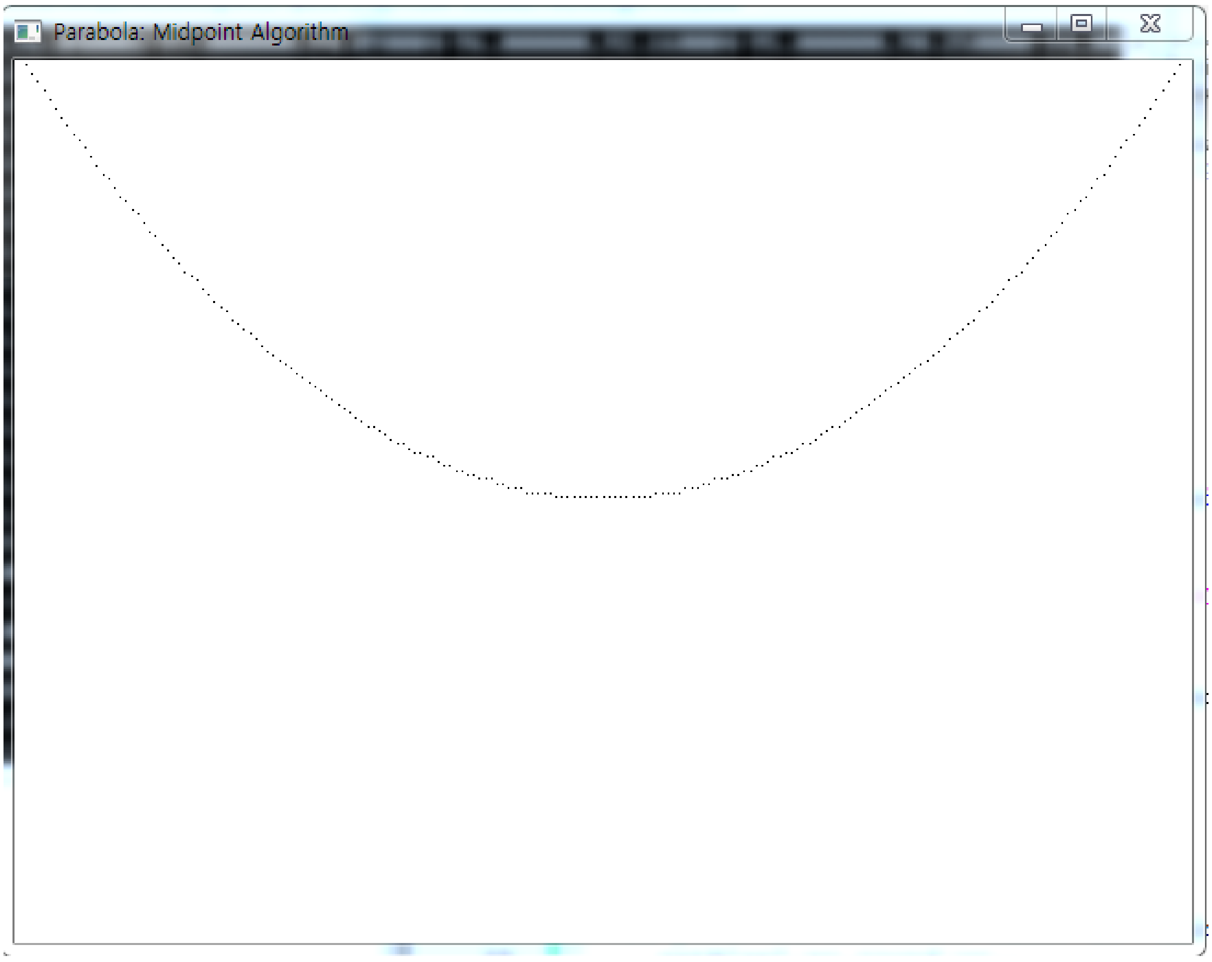
Then, compute the initial condition for the decision function and modify the whole algorithm.

It is very similar to circle generating algorithm. I'll write it out with my own handwriting.

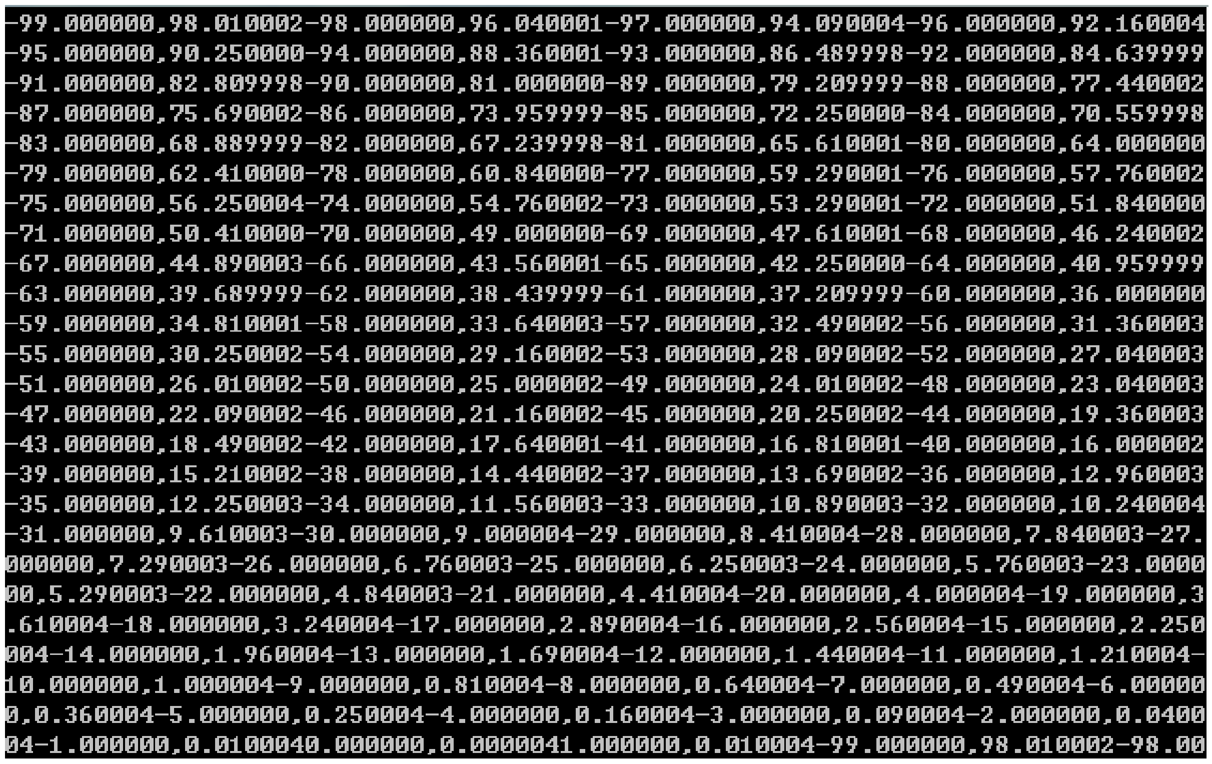
2. 프로그램의 주요 부분에 대한 소스 코드

```
void midpoint_parabola(int x0, int y0, int xEnd, int yEnd)
{
    int k;
    float p, xIncrement, yIncrement, x=x0,y=y0, xx, yy;
    //for (k=0;k<=100;k++)
    for (k=x0;k<=0;k++)
    {
        xIncrement=1.0;
        //x+=xIncrement;
        yIncrement=(2*x+1)/100;
        //y+=yIncrement;
        //p=((x+1)*(x+1))-100*(y+0.5); //decision function, y=(1/100)x^2이므로
        p=((x+1)*(x+1))/100-(y-0.5);
        xx=x+1;
        yy=y+1;
        if (p<=0)
        {
            x+=xIncrement;
            y+=yIncrement;
            //p=p+(2*x+2)+1;
            //p=p+2*x+1;
            //p=p+2*x+3;
            p=p-(2*x)-1;
            parabolaPlotPoints(xx,yy);
            //setPixel(xx,round(y));
            //setPixel(-(xx),round(y));
        }
        else
        {
            x+=xIncrement;
            y+=yIncrement;
            //p=p+2*x-100;
            //p=p+(2*x)-97;
            //p=p+(2*x + 2)+1-(2*y - 2);
            //p=p+(2*x)+1-(2*y);
            p=p+(2*x)-101;
            //setPixel(xx,round(yy));
            //setPixel(-(xx),round(yy));
            parabolaPlotPoints(xx,yy);
        }
        printf("%f,%f",x,y);
    }
}
void parabolaPlotPoints(GLint xc, GLint yc){
    setPixel(xc,round(yc));
    setPixel(-(xc),round(yc));
}
```

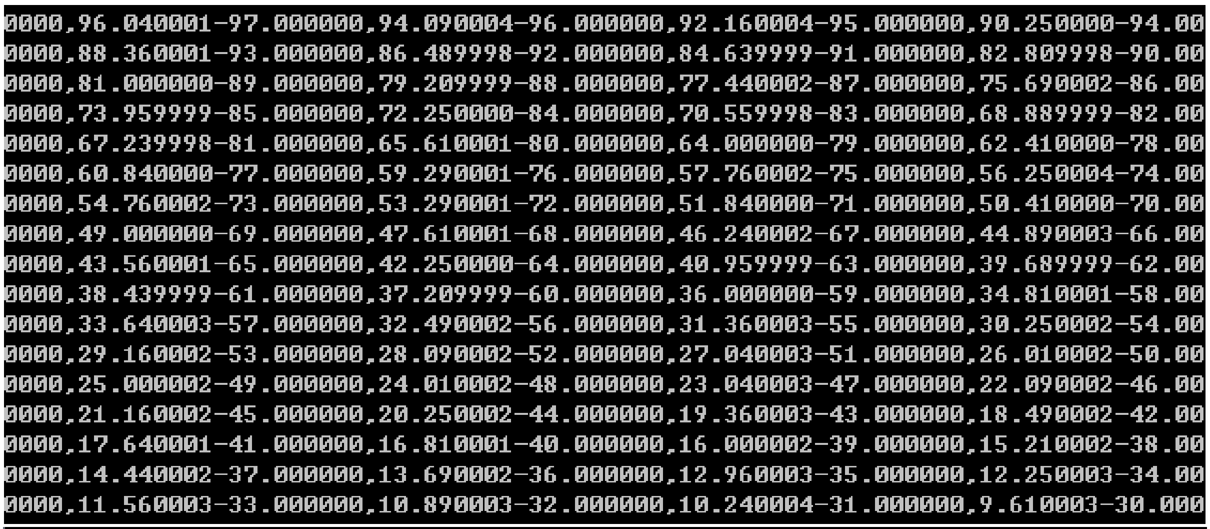
3. 실행 결과



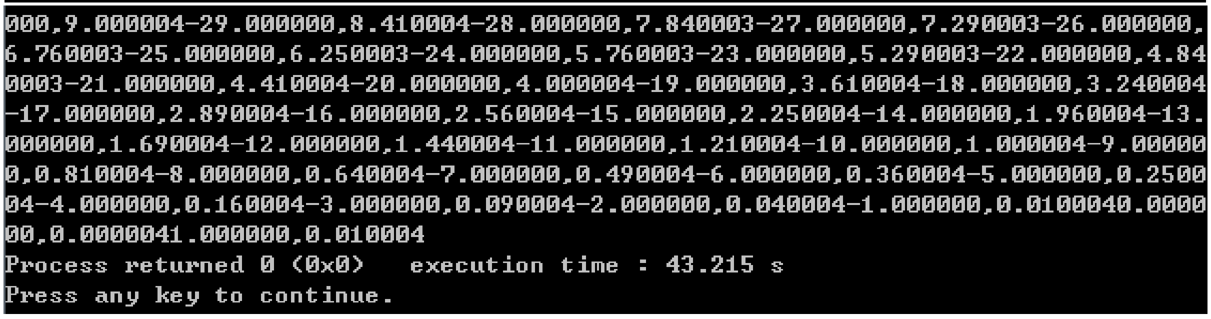
➔ print parabola x, y result



```
-99.000000,98.010002-98.000000,96.040001-97.000000,94.090004-96.000000,92.160004
-95.000000,90.250000-94.000000,88.360001-93.000000,86.489998-92.000000,84.639999
-91.000000,82.809998-90.000000,81.000000-89.000000,79.209999-88.000000,77.440002
-87.000000,75.690002-86.000000,73.959999-85.000000,72.250000-84.000000,70.559998
-83.000000,68.889999-82.000000,67.239998-81.000000,65.610001-80.000000,64.000000
-79.000000,62.410000-78.000000,60.840000-77.000000,59.290001-76.000000,57.760002
-75.000000,56.250004-74.000000,54.760002-73.000000,53.290001-72.000000,51.840000
-71.000000,50.410000-70.000000,49.000000-69.000000,47.610001-68.000000,46.240002
-67.000000,44.890003-66.000000,43.560001-65.000000,42.250000-64.000000,40.959999
-63.000000,39.689999-62.000000,38.439999-61.000000,37.209999-60.000000,36.000000
-59.000000,34.810001-58.000000,33.640003-57.000000,32.490002-56.000000,31.360003
-55.000000,30.250002-54.000000,29.160002-53.000000,28.090002-52.000000,27.040003
-51.000000,26.010002-50.000000,25.000002-49.000000,24.010002-48.000000,23.040003
-47.000000,22.090002-46.000000,21.160002-45.000000,20.250002-44.000000,19.360003
-43.000000,18.490002-42.000000,17.640001-41.000000,16.810001-40.000000,16.000002
-39.000000,15.210002-38.000000,14.440002-37.000000,13.690002-36.000000,12.960003
-35.000000,12.250003-34.000000,11.560003-33.000000,10.890003-32.000000,10.240004
-31.000000,9.610003-30.000000,9.000004-29.000000,8.410004-28.000000,7.840003-27.
000000,7.290003-26.000000,6.760003-25.000000,6.250003-24.000000,5.760003-23.0000
00,5.290003-22.000000,4.840003-21.000000,4.410004-20.000000,4.000004-19.000000,3
.610004-18.000000,3.240004-17.000000,2.890004-16.000000,2.560004-15.000000,2.250
004-14.000000,1.960004-13.000000,1.690004-12.000000,1.440004-11.000000,1.210004-
10.000000,1.000004-9.000000,0.810004-8.000000,0.640004-7.000000,0.490004-6.00000
0,0.360004-5.000000,0.250004-4.000000,0.160004-3.000000,0.090004-2.000000,0.0400
04-1.000000,0.0100040.000000,0.0000041.000000,0.010004-99.000000,98.010002-98.00
```



```
0000,96.040001-97.000000,94.090004-96.000000,92.160004-95.000000,90.250000-94.00
0000,88.360001-93.000000,86.489998-92.000000,84.639999-91.000000,82.809998-90.00
0000,81.000000-89.000000,79.209999-88.000000,77.440002-87.000000,75.690002-86.00
0000,73.959999-85.000000,72.250000-84.000000,70.559998-83.000000,68.889999-82.00
0000,67.239998-81.000000,65.610001-80.000000,64.000000-79.000000,62.410000-78.00
0000,60.840000-77.000000,59.290001-76.000000,57.760002-75.000000,56.250004-74.00
0000,54.760002-73.000000,53.290001-72.000000,51.840000-71.000000,50.410000-70.00
0000,49.000000-69.000000,47.610001-68.000000,46.240002-67.000000,44.890003-66.00
0000,43.560001-65.000000,42.250000-64.000000,40.959999-63.000000,39.689999-62.00
0000,38.439999-61.000000,37.209999-60.000000,36.000000-59.000000,34.810001-58.00
0000,33.640003-57.000000,32.490002-56.000000,31.360003-55.000000,30.250002-54.00
0000,29.160002-53.000000,28.090002-52.000000,27.040003-51.000000,26.010002-50.00
0000,25.000002-49.000000,24.010002-48.000000,23.040003-47.000000,22.090002-46.00
0000,21.160002-45.000000,20.250002-44.000000,19.360003-43.000000,18.490002-42.00
0000,17.640001-41.000000,16.810001-40.000000,16.000002-39.000000,15.210002-38.00
0000,14.440002-37.000000,13.690002-36.000000,12.960003-35.000000,12.250003-34.00
0000,11.560003-33.000000,10.890003-32.000000,10.240004-31.000000,9.610003-30.000
```



```
000,9.000004-29.000000,8.410004-28.000000,7.840003-27.000000,7.290003-26.000000,
6.760003-25.000000,6.250003-24.000000,5.760003-23.000000,5.290003-22.000000,4.84
0003-21.000000,4.410004-20.000000,4.000004-19.000000,3.610004-18.000000,3.240004
-17.000000,2.890004-16.000000,2.560004-15.000000,2.250004-14.000000,1.960004-13.
000000,1.690004-12.000000,1.440004-11.000000,1.210004-10.000000,1.000004-9.00000
0,0.810004-8.000000,0.640004-7.000000,0.490004-6.000000,0.360004-5.000000,0.2500
04-4.000000,0.160004-3.000000,0.090004-2.000000,0.040004-1.000000,0.0100040.0000
00,0.0000041.000000,0.010004
Process returned 0 (0x0)   execution time : 43.215 s
Press any key to continue.
```

4. 실행 결과에 대한 분석 및 Discussion

I used midpoint algorithm for drawing y = $\frac{1}{100}x^2$ parabola within (-100 <= x <= 100) range. Origin is (0, 0), window size is 640 x 480, coordinate is (-100, 100, -100, 100).

I used midpoint to decide upon which of the two pixels is to be selected. For this, I have to derive decision function of parabola. Given parabola equation is y = $\frac{1}{100}x^2$, so it is easy to find out decision function p_k. p_k is (x_k + 1)^2/100 - (y_k - 0.5) (same as 100(y_k – 0.5) - (x_k + 1)^2).

Then, I derive the equation of y increment. (I think "amount of change" is a more appropriate expression than "increment") Amount of y change equation can be derived by differentiating given parabola equation y = $\frac{1}{100}x^2$. The differentiating result is y = (2*x + 1)/100. Amount of x change is 1.

If $p_k < 0$, than select east position pixel. At this time, $x_{k+1} = x_k + 1$, $y_{k+1} = y_k$. Then put this in the equation p_k, then the result is p_{k+1} = 100(y_k – 0.5) - (x_k + 2)^2 = $p_k - 2x_k - 3 = p_k - 2x_{k+1} - 1$. If $p_k > 0$, than select south east position pixel. At this time, $x_{k+1} = x_k + 1$, $y_{k+1} = y_k - 1$. Then put this in the equation p_k, then the result is p_{k+1} = 100(y_k – 3/2) - (x_k + 2)^2 = $p_k - 2x_k - 103 = p_k - 2x_{k+1} - 101$. If $p_k = 0$, than pixel position is on the parabola boundary. In this way, parabola is stamped with pixels on the other side symmetrically, depending on the point on which it is stamped.

Now, how can I draw a parabola if the given equation is $y = ax^2 + bx + c$? (without $-100 <= x <= 100$ range limits.) Which part should be changed? Depending on the value of x, the value of y changes by the derivative of the parabola equation. So, Amount of y change equation should be y'. Now, we have to derive the generalization of the decision function. I'll write it out with my own handwriting.