CG141_Lab1

Problem Statement

Design an algorithm using the midpoint method for scan converting the function

$y = x^2 / 20$ in the range $-20 \leq x \leq 20$.

Identify the regions where the algorithm would be invoked and how symmetry could be employed to improve its efficiency. Implement the same using OpenGL.

Scan Conversion

We noted that we cannot use a single scan conversion procedure for the entire range. The scan conversion process depends on the slope of the tangent to the curve being drawn at any point. Curves which can be drawn using a single scan conversion have slope(s) that satisfies either of the following conditions:

1. $s > 1$ - slope strictly greater than one
2. $s < 1$ - slope strictly less than one

In the case of a parabola, we note that the slope varies in both the above mentioned intervals. Thus, we need to follow two separate scan-conversion policies according to the slope at any coordinate.

Procedure:

We start plotting points from the origin (0, 0)

The slope of parabola at origin is 0 i.e. < 1. Implies that we start with scan conversion along x-axis.

1. Scan Converting along x-axis
    a. We keep increasing the x coordinate value by one in each iteration and calculate the y coordinate according to the decision parameter from either of y or y+1.
    b. Given we are at point (x, y) we can either move to (x+1, y) or (x+1, y+1)

2. Scan converting along y-axis
    a. We keep increasing the y coordinate value by one in each iteration and calculate the x coordinate according to the decision parameter from either of x or x+1.
    b. Given we are at point (x, y) we can either move to (x, y+1) or (x+1, y+1)

The Symmetry Used

As can be easily seen from the formula of the parabola, it gives the *same value for x and -x,* implying symmetry about the *y-axis.*

Using this fact, we only plot the right half of the parabola and the left half is automatically drawn by symmetry.

Decision Parameter

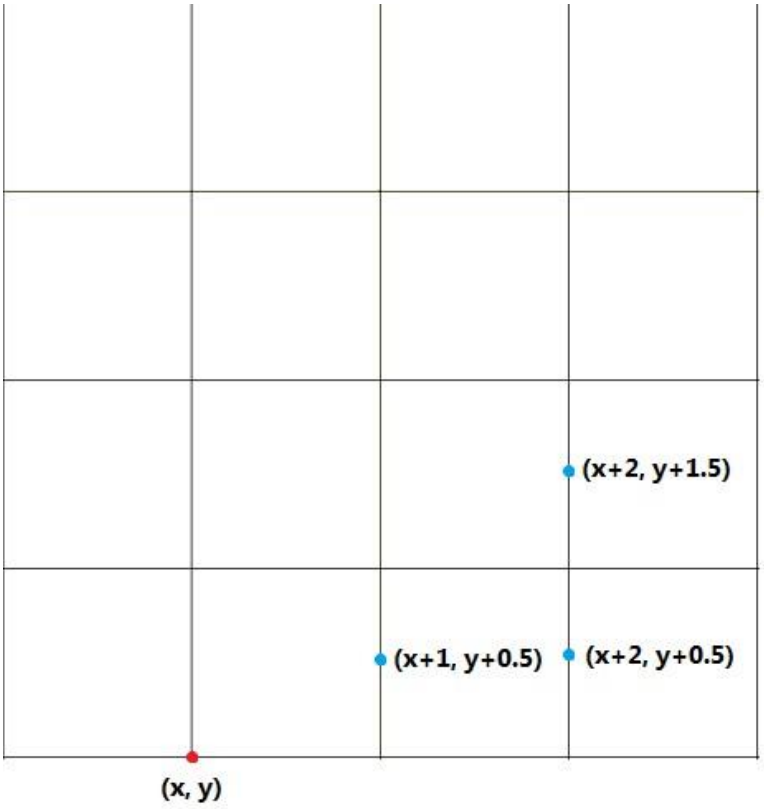
We define a decision parameter d as follows:

d = f(x, y) = x^2 / a - y

Where, a = 20 [Given]

Steps to calculate decision parameter d:

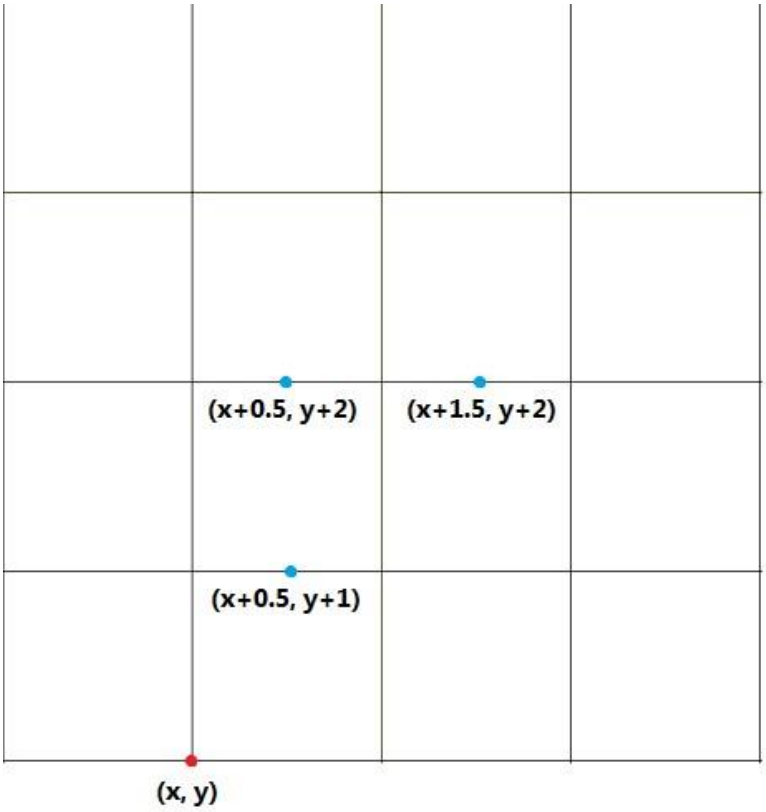
A. Along the x-axis



1. Let current position be (x, y)
2. The decision parameter d0 is calculated at (x+1, y+0.5) to decide the next point.
3. Depending on whether the decision parameter is greater than 0 or less than 0, we can have 2 choices for the decision parameter: f(x+2, y+0.5) or f(x+2, y+1.5)
4. Now,
    a. f(x+2, y+0.5) - f(x+1, y+0.5) = (2*x + 3) / a [d <= 0]

    b. f(x+2, y+1.5) - f(x+1, y+0.5) = (2*x + 3 - a) / a [d > 0]

B. Along the y-axis



1. Let current position be (x, y)
2. The decision parameter d0 is calculated at (x+0.5, y+1) to decide the next point.
3. Depending on whether the decision parameter is greater than 0 or less than 0, we can have 2 choices for the decision parameter: f(x+0.5, y+2) or f(x+1.5, y+2)
4. Now,
   a. f(x+1.5, y+2) - f(x+0.5, y+1) = (2*x + 2 - a) / a [d <= 0]

   b. f(x+0.5, y+2) - f(x+0.5, y+1) = -1 [d > 0]

For simplifying the calculation, we have taken d as f(x,y)*a.

A. Scan conversion along x axis:

Initialising the decision parameter:
d = f(1, 0.5)
= (2 - a) / 2a = -9

```
if (d <= 0)
        d += 2*x + 3;
        next point is (x+1, y)
if (d > 0)
        d += 2*x + 3 - a;
        next point is (x+1, y+1)
```

B. Scan conversion along y axis:

Initialising the decision parameter:

```
d = f(10.5, 6)
  = -9
```

```
if (d <= 0)
        d += 2*x + 2 - a;
        next point is (x+1, y+1)
if (d > 0)
        d += -a;
        next point is (x, y+1)
```

Optimizations done

1. Avoiding floating point calculations

- To make the plotting process faster and less memory intensive, we have used integer calculations instead of floating point calculations, without the output being affected.
- We also compared the parabolas plotted using floating point calculations and the one using the above algorithm and they came out exactly same.

2. Calculating decision parameter using the previous decision parameter

To avoid the calculations of decision parameter at every iteration, we have calculated it by adding some delta to the previously calculated decision parameter, according to whether the decision parameter is positive or negative.

Brief Explanation of Code

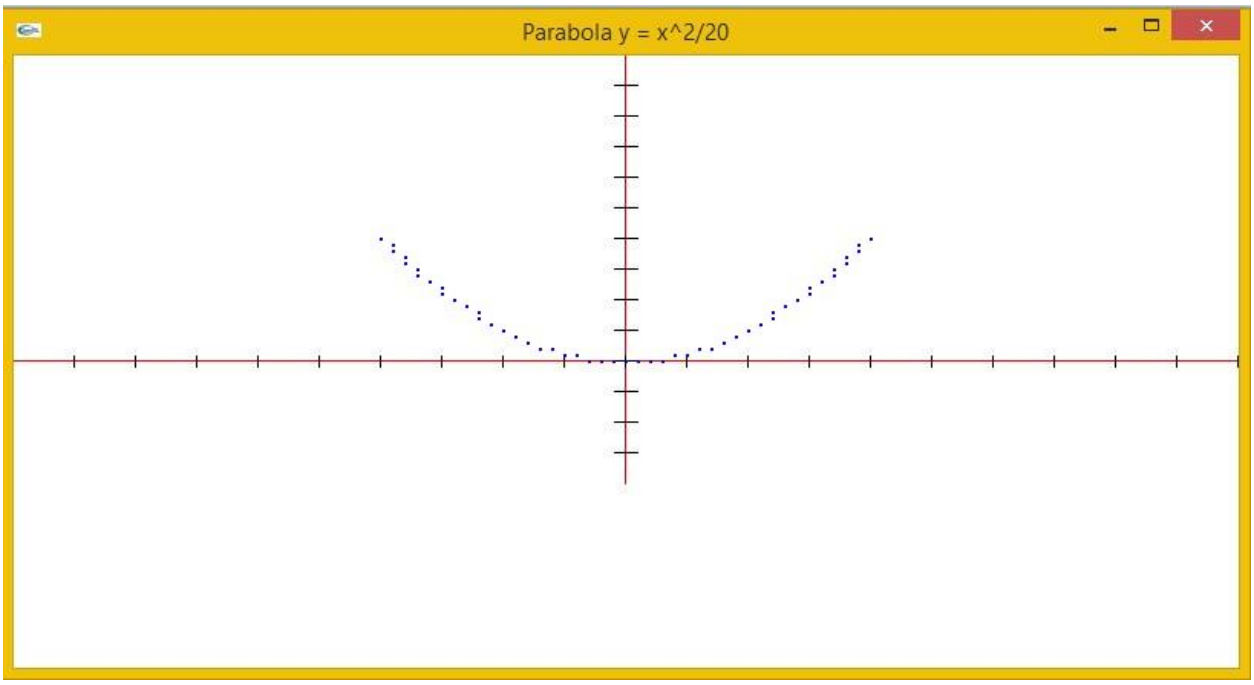
1. parabola ()

The function responsible for scan converting the parabola, calculating the decision parameter and ultimately plotting the points.

2. drawPoints()
Function for plotting a pair of points symmetric about the y-axis.
Given (x, y) it plots (x, y) as well as (-x, y).

3. drawAxes()
Function to draw X and Y axes on the window.

The Output
The figure below shows the output parabola as plotted by our algorithm.



Range of x values: -20 < x < 20

Point of slope = 1: (10, 5)
To make the parabola plotted at the center of the window for better view and to avoid any part of parabola being cut, we have added a delta, i.e. a constant x and y offset to all the points on the parabola.

Assumption Taken:
We have assumed the left-bottom vertex of grid points as the coordinate points.

Submitted by :
Siddhant Tuli : 2012A7PS077P
Shalaka Somani : 2012C6PS718P
Ronil Pancholia : 2012C6PS629P