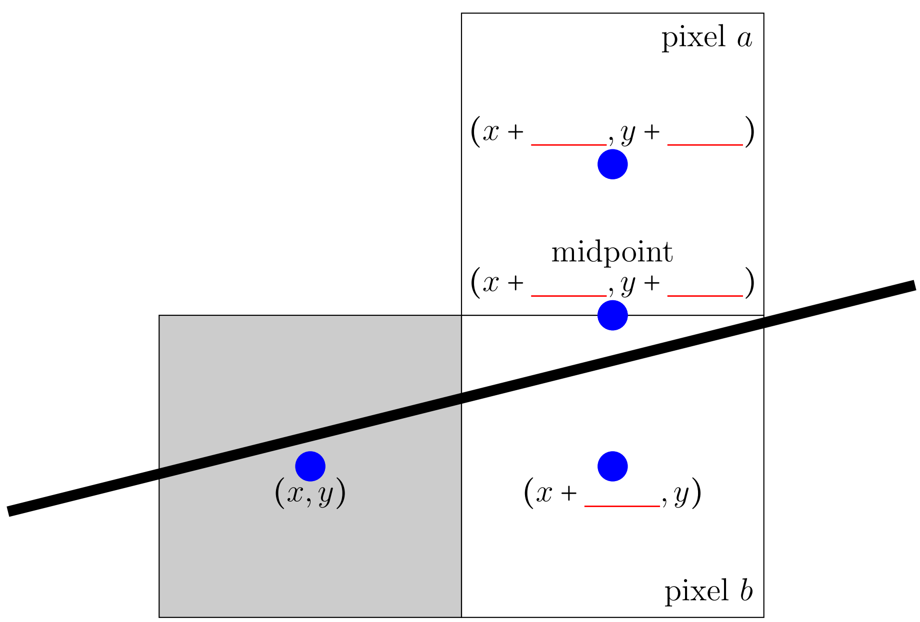
CS130 - LAB - Bresenham's line algorithm / midpoint algorithm

Name: ____________________________ SID: ________________

1 Midpoint algorithm - case 1: $0 \le m \le 1$

This Lab consists of implementing the midpoint algorithm to draw continuous lines using only integer operations. Recall the line equation is $y = mx + n$, where m is the slope and n is the y intercept. Given two points $p_0 = (x_0, y_0)$ and $p_1 = (x_1, y_1)$, the slope is calculated as $m = \frac{y_1 - y_0}{x_1 - x_0} = \frac{dy}{dx}$. Consider $0 \le m \le 1$ (line in angle between 0 and 45 degrees). The idea is to determine which of the two pixels (a or b) we should draw. Complete the missing coordinates of the points below.



In particular, we can evaluate the midpoint between a and b using a function $f(x, y)$ that returns a positive number if the point (x, y) lies above the line, negative if it lies below, and zero if the point is on the line. This function should have the form

$$f(x,y) = Ax + By + C, \tag{1}$$

where A, B, and C depend on x_0, y_0, x_1, and y_1. You may assume that $x_0 \le x_1$, $y_0 \le y_1$, and that the slope satisfies $0 \le m \le 1$. (If the endpoints are in the opposite order, you can swap them before you rasterize.) Note that scaling A, B, C by a positive number does not change the sign of $f(x,y)$, so we can require that A, B, C be polynomials. (Hint: use $f(x_0,y_0) = f(x_1,y_1) = 0$ to solve for B and C in terms of A and the endpoints. Choose A so that the fractions go away and so that A, B, C share no common factors. At this point, your $f(x,y)$ should be correct up to sign. Since points above the line should give you a positive value, you can check $f(x_0, y_0+1) > 0$, since the point (x_0, y_0+1) should be above the line. If the sign is wrong, negate A, B, C.) A = ____________________, B = ____________________, C = ____________________.

In the diagram above, we have just finished setting the pixel (x,y), and now we must decide whether we should increment y, leading us to either pixel a or b. Let $g(x,y)$ be a function that is negative if we want to increment y. You can use $f(x,y)$ to help you define $g(x,y)$. (Note that $f(x,y) \ne g(x,y)$. If x,y are integers, $g(x,y)$ should an integer, too. Note that you can scale it to clear fractions, since only the sign matters.) $g(x,y)$ = __________________________.

This would be a good time to implement a version of `draw_line` to make sure that all of the work that you have done so far is correct. Your code should work correctly when $0 \le m \le 1$. (In other cases, it may do strange things - we will handle the other cases later.) When it works, continue on to the next step. At this stage, your code should look like this:

```
void draw_line(int x0, int y0, int x1, int y1, float col[3])
{
    /* TODO: swap the points? */
    for(int x=x0,y=y0;x<=x1;x++)
    {
        set_pixel(x,y,col);
        int g=/* TODO */;
        if(g<0) y++;
    }
}
```

2 Case 2: $-1 \le m \le 0$.

Next, we will extend our algorithm to handle slopes $-1 \le m \le 0$. Instead of updating x, y using `x++;` and `y++;`, we will instead use `x++;` and `y+=dy;`, where $\Delta y = \pm 1$. The points (x_0, y_0) and (x_1, y_1) should be swapped if ____________________.Next, we need to compute Δy = ____________________ (from x_0, x_1, y_0, y_1).

Now, we must update our definition of $g(x,y)$ so that it works correctly when $\Delta y = 1$ or $\Delta y = -1$. We must make sure that $g(x,y) < 0$ when we want to change y and $g(x,y) \geq 0$ when we want to leave y unchanged. $g(x,y) =$ ______________________________________.

This is a good time to test your modifications. At this stage, your code should look like this:

```
void draw_line(int x0, int y0, int x1, int y1, float col[3])
{
    /* TODO: swap the points? */
    int dy=/* TODO: this should be +1 or -1. */;
    for(int x=x0,y=y0;x<=x1;x++)
    {
        set_pixel(x,y,col);
        int g=/* TODO */;
        if(g<0) y+=dy;
    }
}
```

3 Incremental updates.

Instead of recomputing g each iteration, we instead like to update it incrementally. If $g < 0$, then we will update `x++;y+=dy;g+=dg0;`. Otherwise, we will update `x++;g+=dg1;`. The initial value of g should be $g =$ __________________________. The update increments are $\Delta g_0 =$ __________________________ and $\Delta g_1 =$ __________________________. (You can compute the increments by writing down the difference between what g is currently and what it should be in the next iteration assuming the corresponding update has been applied. This difference should be the same for every loop iteration.)

This is a good time to test your modifications. At this stage, your code should look like this:

```
void draw_line(int x0, int y0, int x1, int y1, float col[3])
{
    /* TODO */
    int dy=/* TODO: this should be +1 or -1. */;
    int g=/* TODO */;
    int dg0=/* TODO */;
    int dg1=/* TODO */;
    for(int x=x0,y=y0;x<=x1;x++)
    {
        set_pixel(x,y,col);
        if(g<0)
        {
            y+=dy;
```

```
            g+=dg0;
        }
        else g+=dg1;
    }
}
```

4 Cases 3 & 4: $|m| > 1$

The remaining cases can be handled by swapping the roles of x and y in your existing code. (You do not need to repeat all of your derivations. This step should be very easy.)

Your final code should look like this:

```
void draw_line(int x0, int y0, int x1, int y1, float col[3])
{
    if(/* TODO */)
    {
        /* TODO */
        int dy=/* TODO: this should be +1 or -1. */;
        int g=/* TODO */;
        int dg0=/* TODO */;
        int dg1=/* TODO */;
        for(int x=x0,y=y0;x<=x1;x++)
        {
            set_pixel(x,y,col);
            if(g<0)
            {
                y+=dy;
                g+=dg0;
            }
            else g+=dg1;
        }
    }
    else
    {
        /* TODO */
        int dx=/* TODO: this should be +1 or -1. */;
        int g=/* TODO */;
        int dg0=/* TODO */;
        int dg1=/* TODO */;
        for(int y=y0,x=x0;y<=y1;y++)
        {
```

```
            set_pixel(x,y,col);
            if(g<0)
            {
                x+=dx;
                g+=dg0;
            }
            else g+=dg1;
        }
    }
}
```