Yingjie Lian
CS 4600
October 4, 2018

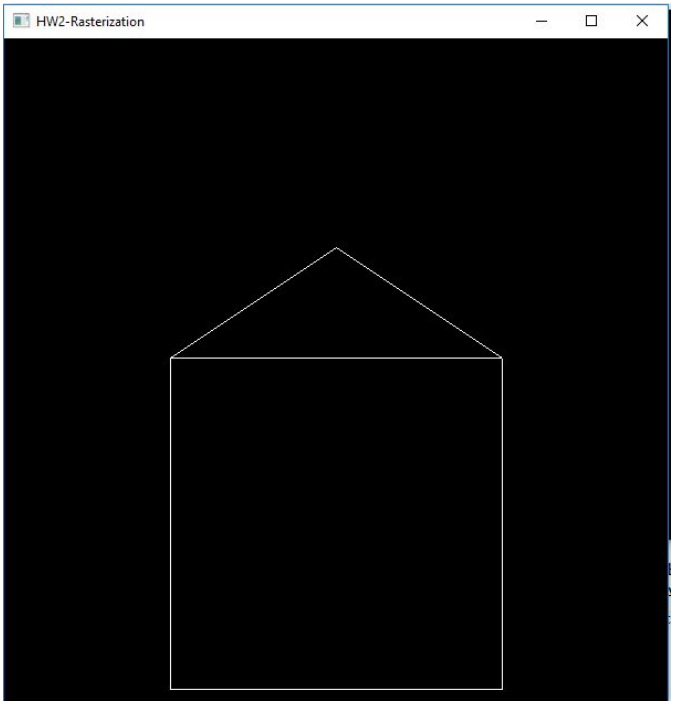
HW2 - Rasterization

Line rasterization

For this part, I followed the equations and directions in the pdf. Also, I have looked through the given lecture and discussion board talked about the Bresenham's algorithm. Here are the screenshots for the codes I wrote.

```
void drawLine(int x1, int y1, int x2, int y2)
{
    // Task 1
    // This function should draw a line from pixel (x1, y1) to pixel (x2, y2)
    // Task 1

    bool incline = abs(x2 - x1) < abs(y2 - y1);

    if (incline)
    {
        std::swap(x1, y1);
        std::swap(x2, y2);
    }

    if (x1 > x2)
    {
        std::swap(x1, x2);
        std::swap(y1, y2);
    }

    int dx = x2 - x1;
    int dy = abs(y2 - y1);

    int error = dx / 2;

    int step = 0;

    if (y1 < y2) {
        step = 1;
    }
    else {
        step = -1;
    }

    while (x1 < x2)
    {
        if (incline)
        {
            putPixel(y1, x1++);
        }
        else
        {
            putPixel(x1++, y1);
        }

        error -= dy;

        if (error < 0)
        {
            y1 += step;
            error += dx;
        }
    }

}
```

By following the Bresenham's algorithm, first I declared the incline because I need to draw a line from pixel (x1, y1) to pixel (x2, y2). As a result, determine which direction and how far the line should is important. Then, call the putPixel() method to draw pixels and then draw the line.

Then, it looks like this: 

Circle rasterization

Had some hard time for thinking how to deal with the radius of the circle until I saw the source code in lecture06-lineCircle from lecture. I used a while loop to call the putPixel() method in order to draw pixels and then draw the line. Also, I used if-else statement to determine where the

```
void drawCircle(int x0, int y0, int R)
{
    // Task 2
    // This function should draw a circle,
    // where (x0, y0) is the center of the circle and R is the radius
    int x = 0;
    int y = R;
    int d = 1 - R;

    while (y >= x)
    {
        putPixel(x0 + x, y0 + y);
        putPixel(x0 - x, y0 + y);
        putPixel(x0 + x, y0 - y);
        putPixel(x0 - x, y0 - y);
        putPixel(x0 + y, y0 + x);
        putPixel(x0 - y, y0 + x);
        putPixel(x0 + y, y0 - x);
        putPixel(x0 - y, y0 - x);

        if (d < 0)
        {
            d = d + 2 * x + 3;

        }
        else {
            d = d + 2 * (x - y) + 5;
            y--;
        }

        x++;

        putPixel(x0 + x, y0 + y);
        putPixel(x0 - x, y0 + y);
        putPixel(x0 + x, y0 - y);
        putPixel(x0 - x, y0 - y);
        putPixel(x0 + y, y0 + x);
        putPixel(x0 - y, y0 + x);
        putPixel(x0 + y, y0 - x);
        putPixel(x0 - y, y0 - x);

    }
```

angles need to change. Here are the screenshots for the codes I wrote.

Then, it looks like this, a circle along with the picture from DrawLine: