COSC 4370 - Homework 1

Rizwan Lokhandwala PSID: 1887820

September 2022

1 Problem

The assignment requires the rasterization of the arcs of an ellipse which is defined as:

$(\frac{x}{12})^2 + (\frac{y}{6})^2 = 64^2 | x \geq 0$

The dimensions of the image will be 1600x1600 pixels^2 and represent a graph ranging from x:[-800, 800] and y:[-800, 800].

2 Method

Only main.cpp is to be modified. I created two structs which were utilized by three functions.

Structs:
- `Coordinate`
- `ellipse_info`

Functions:
- `light_pixel_within_boundaries()`
- `light_ellipse_quadrants()`
- `draw_ellipse()`

`light_pixel_within_boundaries()` receives integers `x, y, START_X, END_X.`
`x,y` represents a graphical coordinate and `START_X, END_X` represent the pixel coordinate boundaries.
`x,y` is translated to pixel coordinate and if it within the boundaries and also within the size of the image window, it will be set to white (RGB = [255,255,255])

`light_ellipse_quadrants()` receives Coordinates `center` and `graphical_pixel`
The graphical coordinate we want to light will be translated according to where the user describes the center of the ellipse.
We then light the pixel along with each mirror point on the ellipse.

`draw_ellipse()` receives `(class ellipse_info) my_ellipse`
Given the information of the ellipse, this function utilizes the midpoint algorithm to draw the ellipse.

3 Implementation

We are given the function:

$(\frac{x}{12})^2 + (\frac{y}{6})^2 = 64^2 | x \geq 0$

However, my algorithm works on the ellipse parent formula:

$b^2x^2 + a^2y^2 = a^2b^2$

To formulate this we undergo the following procedures:

$(\frac{x}{12})^2 + (\frac{y}{6})^2 = 64^2 => (\frac{x}{12*64})^2 + (\frac{y}{6*64})^2 = 1$

$=> (\frac{x}{768})^2 + (\frac{y}{384})^2 = 1 => 384^2x^2 + 768^2y^2 = 384^2 * 768^2$
Therefore, b = 384 and a = 768
and the ellipse is denoted by $f(x, y) = b^2x^2 + a^2y^2 - a^2 * b^2 | f(x, y) = 0$.

`light_ellipse_quadrants`
An ellipse can be broken up into four quadrants, therefore we only need to design an algorithm to draw
one quadrant and we can then mirror the other 3. In this algorithm, we draw the top-right quadrant.
As we determine that pixel(x,y) we will light: (x,y), (x,-y), (-x,y), (-x,-y).

`draw_ellipse()`
A check is first performed to determine if the desired boundaries `(START_X, END_X)` are within the boundaries of the image window.
If the check is passed, four Coordinates will be assigned:
`pixel_current, pixel_east, pixel_south_east, and pixel_south.`
The y-intercept of an ellipse is (0, b), so we begin with `current_pixel` assigned here.
we then call `light_ellipse_quadrants(pixel_current)` This will light (0,b) and (0,-b).
The midpoint algorithm follows this line of thought:
As the ellipse passes through two pixels, whichever pixel the line is closer to gets lit.
We denote two regions (region 1 and region 2). Region 1 is such that the slope, m $\geq$ -1 and Region 2 is such that the slope, m $\leq$ -1.
We denote `pixel_east` as the pixel directly right of `pixel_current`, and `pixel_south_east` as the pixel down right of `pixel_current`.
While in Region 1, the ellipse will pass through `pixel_east` and `pixel_south_east`. The midpoint would then be (`pixel_current.x + 1, pixel_current.y - 0.5`)
since our first point is (0,b): At first this distance will be $f(midpoint) = f(0 + 1, b - 0.5) = b^2(0 + 1) + a^2(b - .5)^2 - a^2 * b^2 = b^2 + a^2(b - 0.5)^2 - a^2 * b^2$.
let xp, yp denote the coordinates of the last chosen pixel. $if distance < 0, (aka the ellipse is above the midpoint) we choose e$ distance next = f(xp + 1, yp)
then we increment pixel east.x + 1 and pixel south east x + 1.
else if distance ¿= 0, we choose south east pixel and distance next = f(xp+1, yp-0.5)
then we increment pixel east and southeast down right 1 pixel.
The same is done in region 2, except we choose between south pixel and south east pixel.