Assignment 1

Zijing Yang

CS6533/CS4533, Spring 2024
Prof. Yi-Jen Chiang, NYU School of Engineering

February 25, 2024

(a)

Based on the Bresenham's scan-conversion algorithm, at the given point $P = (x_p, y_p)$, we have two option either to choose the next pixel in the east (i.e. $E = (x_p + 1, y_p)$) or in the south east (i.e. $SE = (x_p + 1, y_p - 1)$).

The decision variable D is defined by the function $F(x, y) = x^2 + y^2 - r^2$, which determines the position of the midpoint relative to the circle's perimeter. Given the midpoint $M = (x_p + 1, y_p - 1/2)$, since the algorithm begins at point A, which is at $= (0, r)$, the initial decision variable, D_{start}, is computed as follows:

$$
\begin{aligned}
D_{\text{start}} &= F\left(1, r - \frac{1}{2}\right) \\
&= 1 + \left(r - \frac{1}{2}\right)^2 - r^2 \\
&= 1 + \left(r^2 - r + \frac{1}{4}\right) - r^2 \\
&= \frac{5}{4} - r
\end{aligned}
$$

To ensure integer arithmetic and avoid floating-point computations, we simply each side by 4 to keep it in integer form. So we redefine the decision variable D as $D = 4F(M)$, where $F(x, y) = x^2 + y^2 - r^2$. In this case, we have

$$
D_{\text{start}} = 5 - 4r
$$

For each iteration, the algorithm calculates the next point based on the current value of D. If $D < 0$, then M is inside the circle and we will choose E to plot the next pixel on the circle. Otherwise, M is on or outside the circle and we will choose SE. When choosing E (i.e. $D_{old} < 0$),

$$
\begin{aligned}
D_{\text{old}} &= 4F\left(x_p + 1, y_p - \frac{1}{2}\right) \\
&= 4\left((x_p + 1)^2 + \left(y_p - \frac{1}{2}\right)^2 - r^2\right)
\end{aligned}
$$

$$
\begin{aligned}
D_{\text{new}} &= 4F\left(x_p + 1 + 1, y_p - \frac{1}{2}\right) \\
&= 4\left((x_p + 2)^2 + \left(y_p - \frac{1}{2}\right)^2 - r^2\right)
\end{aligned}
$$

The change in D for this move, denoted as ΔD_E, can be calculated as follows.

$$
\begin{aligned}
\Delta D_E &= D_{\text{new}} - D_{\text{old}} \\
&= 4\left((x_p+2)^2 + \left(y_p - \frac{1}{2}\right)^2 - r^2 - \left[(x_p+1)^2 + \left(y_p - \frac{1}{2}\right)^2 - r^2\right]\right) \\
&= 8(x_p+1) + 4
\end{aligned}
$$

When choosing SE (i.e. $D_{old} \geq 0$),

$$
\begin{aligned}
D_{\text{old}} &= 4F\left(x_p+1, y_p - \frac{1}{2}\right) \\
&= 4\left((x_p+1)^2 + \left(y_p - \frac{1}{2}\right)^2 - r^2\right)
\end{aligned}
$$

$$
\begin{aligned}
D_{\text{new}} &= 4F\left(x_p+1+1, y_p - 1 - \frac{1}{2}\right) \\
&= 4\left((x_p+2)^2 + \left(y_p - \frac{3}{2}\right)^2 - r^2\right)
\end{aligned}
$$

The change in D for this move, denoted as ΔD_{SE}, can be calculated as follows.

$$
\begin{aligned}
\Delta D_{SE} &= D_{\text{new}} - D_{\text{old}} \\
&= 4\left((x_p+2)^2 + \left(y_p - \frac{3}{2}\right)^2 - r^2 - \left[(x_p+1)^2 + \left(y_p - \frac{1}{2}\right)^2 - r^2\right]\right) \\
&= 8(x_p+1) - 8(y_p-1) + 4
\end{aligned}
$$

In summary,

$$
\begin{aligned}
D_{\text{start}} &= 5 - 4r \\
D_{\text{new}} &= D_{\text{old}} + \begin{cases} 8(x_p+1) - 8(y_p-1) + 4 & \text{if } D_{\text{old}} \geq 0 \\ 8(x_p+1) + 4 & \text{else.} \end{cases}
\end{aligned}
$$

Due to the circle's symmetry, the algorithm only calculates points for one-eighth of the circle (from A to B), then mirrors those points across the circle's octants: (x, y), (-x, y), (x, -y), (-x, -y), (y, x), (y, -x), (-y, x), (-y, -x).

(b)

The OpenGL coordinate system is a two-dimensional coordinate system where the origin (0,0) is typically at the bottom left of the window, and the x and y coordinates increase to the right and upwards, respectively.

The function call 'gluOrtho2D(0.0, WINDOW_WIDTH, 0.0, WINDOW_HEIGHT)' in the provided code specifies the clipping volume by defining the coordinate system. This particular setup establishes the origin $(0, 0)$ at the bottom-left corner of the window and the point (WINDOW_WIDTH, WINDOW_HEIGHT), which is $(600, 600)$ given the defined constants, at the top-right corner. Thus, this estabilishes a viewport where the positive x-asis extends horizontally to the right and the positive y-axis extends vertically upwards.

Then, the function 'glVertex2i(300, 300)' places a point at the coordinate $(300, 300)$, which is exactly the center of this window, as it is half of the width and height. If I change the call to 'glVertex2i(30, 30)', the point will be drawn 30 pixels from the left edge and 30 pixels from the bottom edge of the window. The point will appear near the left-bottom corner but slightly towards the center of the window.

However, given the current setup, pixels at negative coordinates (x, y) where $x < 0$ or $y < 0$ would not be displayed because they fall outside the defined clipping volume. In this code, only the coordinates from $(0, 0)$ to $(600, 600)$ will be displayed.