

Exercise Computer graphics – (till October 12, 2007)

Anti-Aliasing of primitives using the mid-point algorithm

Exercise 8: Finish your test application for drawing anti-aliased lines.

Solution: See code on homepage.

Exercise 9: a) Find a solution for drawing anti-aliased circles based on the mid-point version of the circle routine from the last exercise.

Solution:

ANTI-ALIASED
CIRCLES

midpoint circle (x_m, y_m)
Let $(x_m, y_m) = (0, 0)$
for simplicity

The distance between a point (x_p, y_p) and the circle is simply defined as:

$$D = \sqrt{x_p^2 + y_p^2} - r \quad ; \text{subst} = x_p^2 + y_p^2$$

Distance to point $(x_p, y_p + 1)$:

$$D_{0,+1} = \sqrt{x_p^2 + (y_p + 1)^2} - r$$

$$= \sqrt{x_p^2 + y_p^2 + 2y_p + 1} - r$$

(we can save on a few computations by substituting subst.)

$$= \sqrt{\text{subst} + 2y_p + 1} - r$$

Distance to point $(x_p, y_p - 1)$:

$$D = \sqrt{\text{subst} - 2y_p + 1} - r$$

Exercise Computer graphics – (till October 12, 2007)

Anti-Aliasing of primitives using the mid-point algorithm

Solution:
(continued)

Another simplification can be done by calculating the new subst from the previous one:

Case: We chose E:

$$\text{subst}_{\text{new}} = (x_p+1)^2 + y_p^2 = x^2 + 2x_p + 1 + y_p^2 = \text{subst}_{\text{old}} + 2x_p + 1$$

Case: We chose SE:

$$\text{subst}_{\text{new}} = (x_p+1)^2 + (y_p-1)^2 = x^2 + 2x_p + 1 + y_p^2 - 2y_p + 1 = \text{subst}_{\text{old}} + 2(x_p - y_p) + 1$$

also note that $2x_p$ and $2(x_p - y_p)$ have already been calculated in your midpoint implementation of the circle.

However: We still need to calculate $\sqrt{\dots} - r$ for each pixel
 ↑ expensive operation

[note that +2 can be calculated as a left-shift]

How to speed up the calculation of the square root? => Don't calculate at all but look up the gray-value based on subst_new in a look up table (r is only const.)