More Line and Some Circle Algorithms

CS116A

Chris Pollett

Sep1, 2004.

Introduction

- Parallel Line Algorithms
- Frame Buffer Values
- Circles
- Midpoint Circle Algorithm

Parallel Line Algorithms

How to draw lines if we have multiple processor? Adapt Bresenham. Assume $0<m<1$

- If have p processors, we partition line into p segments.
- The change in width of a segment will $\Delta x\_p = (\Delta x+p-1)/p$. The p-1 comes from overlapping 1 pixel at end of segments.
- The change in height of a segment will be $\Delta y\_p=m\Delta x\_p$

More Parallel Line Alg's

- The start position of the *k*th partition will be (x_k,y_k) where $x\_k = x\_0 + k\Delta x\_p$ and $y\_k = y\_0 + \mathrm{round}(k\Delta y\_p)$
- The decision parameter p_k also needs to be determined for each interval. Recall $p\_k = \Delta x(d\_lower-d\_upper)= 2\Delta y*x\_k + 2\Delta x*y\_\{k+1\} +2\Delta y+2\Delta x(2b-1)$.
- Note Δx, Δy, b are constant and the only other variables are x_k, y_k so p_k can be set up without having to calculate p_m for $m<k$

Frame Buffer Values

Final stage for line segment implementation is to set the frame-buffer color values. That is, actually set bits in memory as opposed to 2D arrays. Also can do with incremental operations.

Suppose screen from (0,0) to (x_{max}, y_{max}).

Assume only dealing with black and white.

Then the frame buffer pixel address for (x,y) is:

$$Addr(x,y) = Addr(0,0) + y(x_{max}+1)+x$$

So incrementally, (x+1,y) will be at $Addr(x+1,y) = Addr(x,y)+1$

More Frame Buffer

- And (x+1,y+1) will be at $Addr(x,y)+x_{max}+2$
- So above op's can be done with only adds. To store more than one bit of color depth same idea and can still use adds.

Circles

Recall equation of a circle:

$(x-x_c)^2 + (y-y_c)^2 = r^2.$

So we could calculate (x,y) on circle using

$y = y_c +/- (r^2-(x -x_c)^2)^{1/2}$

This would be slow because would have to do too many calculations per step.

Another way is to use the equations:

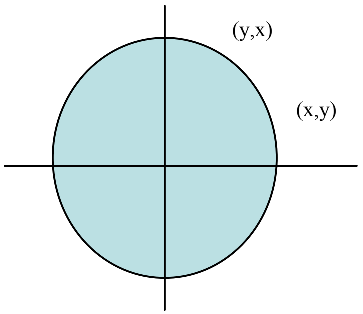
$x = x_c + r \cos\phi$

$y = y_c + r \sin\phi$

Problem is points are determined by changing ϕ and so plot some areas closer together than others

More Circles

- Want to exploit the symmetry of circles.
- First, if angle 0 <= ϕ <= pi/4 then slope of the radius line will be between 0 and 1.
- Notice if can plot circle in this region can extend to whole circle



Midpoint Circle Algorithm

Now we adapt Bresenham to circles:

Let $f_{circ} = x^2+y^2-r^2$

Then $f_{circ} <0$ if (x,y) is within the circle,

Then $f_{circ} =0$ if(x,y) is on the circle,

And then $f_{circ}>0$ if (x,y) is outside the circle

Given we have plotted (x_k,y_k) want to determine which is close to the circle (x_k+1, y_k-1) or (x_k+1, y_k).

More Circle Algorithm

To choose we apply f_{circ} to midway between these two points. So $p_k=f_{circ}(x_k+1, y_k-1/2)$. If $p_k <0$ then the midpoint is inside the circle so should plot (x_k+1, y_k). Otherwise, should plot (x_k+1, y_k-1).

Can compute p_{k+1} from p_k as $p_k+2(x_k+1) +[(y_{k+1})^2 - (y_k)^2]-(y_{k+1}-y_k)+1$.

Here y_{k+1} is either y_k or y_k-1 depending on the sign of p_k

Can show p_0 should be $f_{circ}(1,r-1/2) = 5/4-r$.