Computer Graphics

P. Healy

CS1-08
Computer Science Bldg.
tel: 202727
patrick.healy@ul.ie

Spring 2021–2022

Outline

1. Drawing Algorithms
   - Circle Drawing Algorithms; §6-4

Outline

1. Drawing Algorithms
   - Circle Drawing Algorithms; §6-4

Equation of a circle

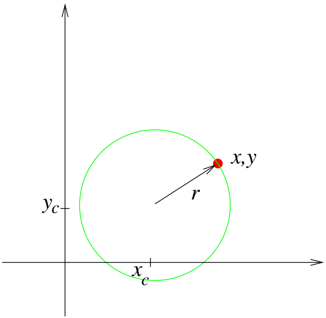
- With respect to their centre the points of a circle must satisfy the Pythagorean theorem: $A^2 + B^2 = C^2$
- For a circle of radius r centred at (x_c, y_c) this means

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

- If we're sampling in terms of x then

$$y = y_c \pm \sqrt{r^2 - (x - x_c)^2}$$

- This leads to "uneven" sampling and is expensive



Parametric equation of a circle

- We want the **perimeter** of the circle to look as even as possible so it would be better to sample along the perimeter, rather than using x coordinates for sampling
- For circle of radius r centred at (x_c, y_c) circle comprises

$$x = x_c + r\cos\theta$$
$$y = y_c + r\sin\theta$$

- By sampling at increments of θ we can get a smoother looking curve
- What is an appropriate value of θ to use?
- The curvature of a circle is given by $\frac{1}{r}$ and this is an appropriate value for θ

Using symmetry

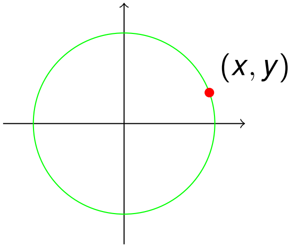
- Computing trigonometrical functions in the θ approach previously is still expensive
- Do we have to compute pixels over entire circle ($360°$)?
- From (x, y) we can get $(\pm x, \pm y)$ for free
- But also (y, x) and ...
- So we only need to compute points in one **octant**
- To keep slope $|m| < 1$ and for increasing x, the octant we choose is second
- We have reduced computations to , but still requires functions

Using symmetry

- Computing trigonometrical functions in the θ approach previously is still expensive
- Do we have to compute pixels over entire circle (360°)?
- From (x, y) we can get $(\pm x, \pm y)$ for free
- But also (y, x) and ...
- So we only need to compute points in one **octant**
- To keep slope $|m| < 1$ and for increasing x, the octant we choose is second
- We have reduced computations to , but still requires functions

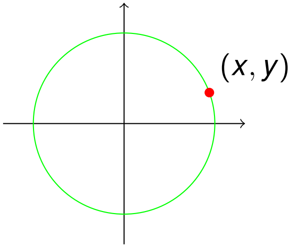
Using symmetry

- Computing trigonometrical functions in the θ approach previously is still expensive
- Do we have to compute pixels over entire circle (360°)?
- From (x, y) we can get $(\pm x, \pm y)$ for free
- But also (y, x) and ...
- So we only need to compute points in one **octant**
- To keep slope $|m| < 1$ and for increasing x, the octant we choose is second
- We have reduced computations to , but still requires functions



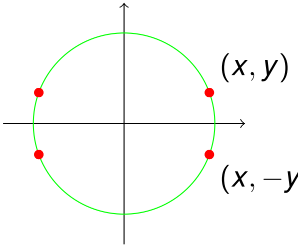
Using symmetry

- Computing trigonometrical functions in the θ approach previously is still expensive
- Do we have to compute pixels over entire circle (360°)?
- From (x, y) we can get $(\pm x, \pm y)$ for free
- But also (y, x) and ...
- So we only need to compute points in one **octant**
- To keep slope $|m| < 1$ and for increasing x, the octant we choose is second
- We have reduced computations to , but still requires functions



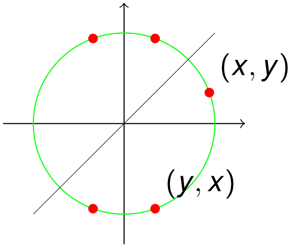
Using symmetry

- Computing trigonometrical functions in the θ approach previously is still expensive
- Do we have to compute pixels over entire circle (360°)?
- From (x, y) we can get $(\pm x, \pm y)$ for free
- But also (y, x) and ...
- So we only need to compute points in one **octant**
- To keep slope $|m| < 1$ and for increasing x, the octant we choose is second
- We have reduced computations to , but still requires functions



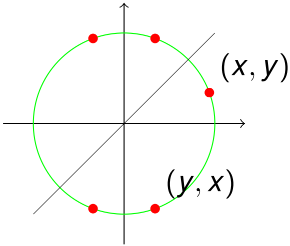
Using symmetry

- Computing trigonometrical functions in the θ approach previously is still expensive
- Do we have to compute pixels over entire circle (360°)?
- From (x, y) we can get $(\pm x, \pm y)$ for free
- But also (y, x) and ...
- So we only need to compute points in one **octant**
- To keep slope $|m| < 1$ and for increasing x, the octant we choose is second
- We have reduced computations to , but still requires functions



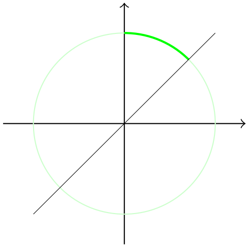
Using symmetry

- Computing trigonometrical functions in the θ approach previously is still expensive
- Do we have to compute pixels over entire circle (360°)?
- From (x, y) we can get $(\pm x, \pm y)$ for free
- But also (y, x) and ...
- So we only need to compute points in one **octant**
- To keep slope $|m| < 1$ and for increasing x, the octant we choose is second
- We have reduced computations to , but still requires functions



Using symmetry

- Computing trigonometrical functions in the θ approach previously is still expensive
- Do we have to compute pixels over entire circle (360°)?
- From (x, y) we can get $(\pm x, \pm y)$ for free
- But also (y, x) and ...
- So we only need to compute points in one **octant**
- To keep slope $|m| < 1$ and for increasing x, the octant we choose is second
- We have reduced computations to , but still requires functions



Using symmetry

- Computing trigonometrical functions in the θ approach previously is still expensive
- Do we have to compute pixels over entire circle (360°)?
- From (x, y) we can get $(\pm x, \pm y)$ for free
- But also (y, x) and ...
- So we only need to compute points in one **octant**
- To keep slope $|m| < 1$ and for increasing x, the octant we choose is second
- We have reduced computations to , but still requires functions

Midpoint Circle Algorithm

- Bresenham's **circle algorithm** avoids this by finding closest pixel to circumference of circle
- It avoids the calculation by comparing **squares** of distances: if $d_l > d_u$ then $d_l^2 > d_u^2$
- We can avoid even working with squares of distances and this is what the Midpoint Circle Algorithm does
- As before, sample at unit intervals and find closest pixel to circle path
- For a circle of radius r, centered at (x_c, y_c) we **translate** it to origin and add back x_c and y_c to all computed values later
- We need a function that tests where a given point is relative to our circle

$$f_{\text{circ}}(x, y) = x^2 + y^2 - r^2$$

Midpoint Circle Algorithm (contd.)

- Given a point (x, y)

$$f_{\text{circ}}(x, y) \begin{cases} < 0, & \text{if } (x, y) \text{ is inside the circle} \\ = 0, & \text{if } (x, y) \text{ is on the circle} \\ > 0, & \text{if } (x, y) \text{ is outside the circle} \end{cases}$$

- If we have coloured (x_k, y_k), for $x_{k+1} = x_k + 1$ what should y_{k+1} be?
- Should we stay with (and colour) $y_{k+1} = y_k$ or drop one to ?
- We test the point $(x_k + 1, y_k - \frac{1}{2})$ and make this be our decision parameter

$$\begin{aligned} p_k &= f_{\text{circ}}(x_k + 1, y_k - \frac{1}{2}) \\ &= (x_k + 1)^2 + (y_k - \frac{1}{2})^2 - r^2 \end{aligned}$$

Midpoint Circle Algorithm (contd.)

- Given a point (x, y)

$$f_{\text{circ}}(x,y)\begin{cases} <0, & \text{if } (x,y) \text{ is inside the circle} \\ =0, & \text{if } (x,y) \text{ is on the circle} \\ >0, & \text{if } (x,y) \text{ is outside the circle} \end{cases}$$

- If we have coloured (x_k, y_k), for $x_{k+1} = x_k + 1$ what should y_{k+1} be?
- Should we stay with (and colour) $y_{k+1} = y_k$ or drop one to $\boxed{\phantom{y_{k+1}=y_k-1}}$?
- We test the point $(x_k + 1, y_k - \frac{1}{2})$ and make this be our decision parameter

$$\begin{aligned} p_k &= f_{\text{circ}}(x_k + 1, y_k - \frac{1}{2}) \\ &= (x_k + 1)^2 + (y_k - \frac{1}{2})^2 - r^2 \end{aligned}$$

Midpoint Circle Algorithm (contd.)

- Given a point (x, y)

$$f_{\mathrm{circ}}(x, y) \begin{cases} < 0, & \text{if } (x, y) \text{ is inside the circle} \\ = 0, & \text{if } (x, y) \text{ is on the circle} \\ > 0, & \text{if } (x, y) \text{ is outside the circle} \end{cases}$$

- If we have coloured (x_k, y_k), for $x_{k+1} = x_k + 1$ what should y_{k+1} be?
- Should we stay with (and colour) $y_{k+1} = y_k$ or drop one to ?
- We test the point $(x_k + 1, y_k - \frac{1}{2})$ and make this be our decision parameter

$$\begin{aligned} p_k &= f_{\mathrm{circ}}(x_k + 1, y_k - \frac{1}{2}) \\ &= (x_k + 1)^2 + (y_k - \frac{1}{2})^2 - r^2 \end{aligned}$$

Midpoint Circle Algorithm (contd.)

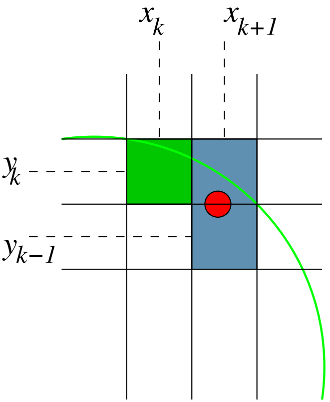
- Given a point (x, y)

$$f_{\text{circ}}(x, y) \begin{cases} < 0, & \text{if } (x, y) \text{ is inside the circle} \\ = 0, & \text{if } (x, y) \text{ is on the circle} \\ > 0, & \text{if } (x, y) \text{ is outside the circle} \end{cases}$$

- If we have coloured (x_k, y_k), for $x_{k+1} = x_k + 1$ what should y_{k+1} be?
- Should we stay with (and colour) $y_{k+1} = y_k$ or drop one to ?
- We test the point $(x_k + 1, y_k - \frac{1}{2})$ and make this be our decision parameter

$$\begin{aligned} p_k &= f_{\text{circ}}(x_k + 1, y_k - \frac{1}{2}) \\ &= (x_k + 1)^2 + (y_k - \frac{1}{2})^2 - r^2 \end{aligned}$$

Midpoint Circle Algorithm (contd.)



If $p_k < 0$ the “half-way” y value is already inside so we should stay with $y_{k+1} = y_k$

o/w if the halfway point is outside, $p_k > 0$, we should definitely compensate by dropping down to a lower y

or, finally, if $p_k = 0$ then either one is equally good (bad) so we could drop, too.

Midpoint Circle Algorithm (contd.)

- From p_k we can express p_{k+1} in terms of it, and after simplifying

$$
\begin{aligned}
p_k &= (x_k + 1)^2 + (y_k - \frac{1}{2})^2 - r^2 \\
p_{k+1} &= (x_k + 1 + 1)^2 + (y_{k+1} - \frac{1}{2})^2 - r^2 \\
&= p_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1
\end{aligned}
$$

- Since y_{k+1} can be either y_k ($p_k < 0$) or $y_k - 1$ ($p_k \geq 0$)

$$
p_{k+1} = p_k + 2x_{k+1} + \begin{cases} 1, & p_k < 0 \\ 1 - 2y_{k+1}, & \text{otherwise} \end{cases}
$$

- : $(x_0, y_0) = (0, r)$ and

$$
p_0 = f_{\text{circ}}(1, r - \frac{1}{2}) = \frac{5}{4} - r
$$