Tutorial 5
CS3241 Computer Graphics (AY22/23)

September 27, 2022

Wong Pei Xian

✉ e0389023@u.nus.edu

Lecture 6 Wrapup

LECTURE 6
○●○

QUESTION 1
○○○○○○

QUESTION 2
○○○○○○

QUESTION 3
○○○○

QUESTION 4
○○○○○

Hidden Surface Removal Algorithms

1. Painter's Algorithm (Depth sorting)
2. Back-face culling
3. Image space (Ray-tracing, closest polygon per pixel)

Tutorial 5

LECTURE 6
000

QUESTION 1
●00000

QUESTION 2
000000

QUESTION 3
0000

QUESTION 4
00000

Question 1

We want to scan-convert (rasterize) the curve $y = x^2/100$ from the pixel locations $(0, 0)$ to $(200, 400)$. Assume there is a function `write_pixel(x, y, color)` to set the color of a pixel at location (x, y). The curve should be drawn as the thinnest possible but not broken. Write a C program fragment to draw the curve.

You are allowed to use floating-point operations, the `round()` function, and even the square-root function `sqrt()`.

Which algorithm?

Floating Point ops allowed $\rightarrow$ Digital Differential Analyzer

$$m = \frac{dy}{dx} = \frac{y_e - y_o}{x_e - x_o}$$

Obtain $m = \frac{dy}{dx} = \frac{d}{dx}(x^2/100) = 50x$.

- Draw line by stepping from x_0 to x_e, step size 1

```
for (x = x0, y = y0; x <= xe; x++) {
    write_pixel( x, round(y), line_color );
    y += m;
}
```
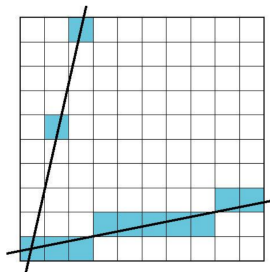
Do we actually need to y += m?

No. We already know the relationship between *y* and *x*.

```
for (int x = 0; x <= xe; x++)
int y = x*x /100;
write_pixel(x, y, color);
```

LECTURE 6
○○○

QUESTION 1
○○○●○○

QUESTION 2
○○○○○○

QUESTION 3
○○○○

QUESTION 4
○○○○○

Inherent problem in vanilla DDA

Problem

- In DDA, for each x, plot pixel at closest y
  - Problem for steep lines

This is where the $m = \frac{dy}{dx}$ comes in handy! We use it to identify when to swop roles of x and y.

LECTURE 6
○○○

QUESTION 1
○○○○●○

QUESTION 2
○○○○○○

QUESTION 3
○○○○

QUESTION 4
○○○○○

Solving this problem

Using Symmetry

- Use only for $0 \leq |m| \leq 1$

- For $|m| > 1$, swap roles of x and y

  - For each y, plot closest x
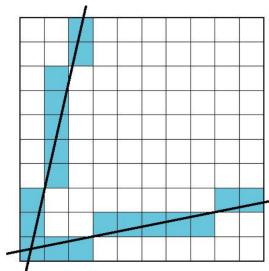
We use it to identify when to swop roles of x and y.

$$m = \frac{x}{50} > 1 \Rightarrow x > 50$$

Lecture 6
○○○

Question 1
○○○○○○●

Question 2
○○○○○○

Question 3
○○○○

Question 4
○○○○○

Complete Solution

LECTURE 6
○○○

QUESTION 1
○○○○○○

QUESTION 2
●○○○○○

QUESTION 3
○○○○

QUESTION 4
○○○○○

Question 2

How do you scan convert a circle without using any floating-point operations? Assume that the circle center has integer pixel coordinates and its radius is an integer. Assume the entire circle is inside the window. Complete the following program fragment.

You are allowed to use a few floating-point operations in the initialization or setup.

LECTURE 6
○○○

QUESTION 1
○○○○○○

QUESTION 2
○●○○○○

QUESTION 3
○○○○

QUESTION 4
○○○○○

Question 2

```
int cx = ...; // center's x
int cy = ...; // center's y
int radius = ...;
int rr = radius * radius;

const float cos45 = 0.70710678; // cos(45 degrees)
int x_max = (int) round(radius * cos45);

int x = 0;
int y = radius;
write_pixel(cx + x, cy + y, color);  // top
write_pixel(cx + x, cy - y, color);  // bottom
write_pixel(cx + y, cy + x, color);  // right
write_pixel(cx - y, cy + x, color);  // left

// Continue here to draw the rest of the circle.
```
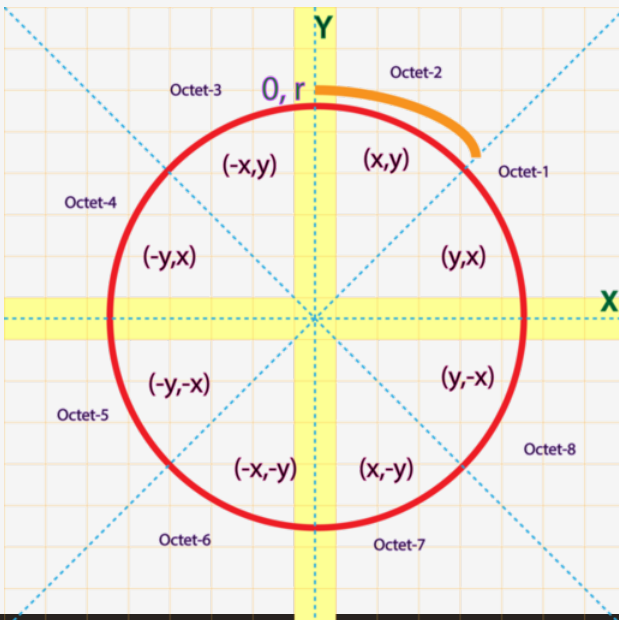
LECTURE 6
000

QUESTION 1
000000

QUESTION 2
000●00

QUESTION 3
0000

QUESTION 4
00000

Which algorithm?

Floating Point ops NOT allowed $\rightarrow$ Bresenham's Algorithm
Hints in the code:

- `x_max = (int) round(radius * cos45:`
- 4 starting pixels: top bottom right left

LECTURE 6
○○○

QUESTION 1
○○○○○○

QUESTION 2
○○○●○○

QUESTION 3
○○○○

QUESTION 4
○○○○○

Pattern

Bresenham's algorithm

With $x = 0$ and $y = r$, we are starting at the **top**.
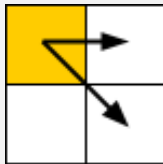
If we have just shaded the yellow pixel (cx + 0, cy + radius), for cx+1 how do we know which pixel to shade?

Don't try differentiating $y^2 + x^2 = r^2$: even with assumptions on the values of y and x it will get clunky and involve sqrt which is prohibited.

LECTURE 6
ooo

QUESTION 1
oooooo

QUESTION 2
ooooo●

QUESTION 3
oooo

QUESTION 4
ooooo

Bresenham's **Circle** algorithm

We know that in a **perfect circle**, $y^2 + x^2 - r^2 = 0$.
Hence for a given y, the closer the value of $| y^2 + x^2 - r^2 |$ to 0, the closer it is to the perfect circle.

```
for (int x = 1; x <= x_max; x++) {
    if ( (x*x + y*y) - rr > rr - (x*x + (y-1)*(y-1))) {
        y = y - 1;
    }
    write_pixel(cx + x, cy + y, color);  // top & right
    write_pixel(cx - x, cy + y, color);  // top & left
    write_pixel(cx + x, cy - y, color);  // bottom & right
    write_pixel(cx - x, cy - y, color);  // bottom & left
    write_pixel(cx + y, cy + x, color);  // right & up
    write_pixel(cx + y, cy - x, color);  // right & down
    write_pixel(cx - y, cy + x, color);  // left & up
    write_pixel(cx - y, cy - x, color);  // left & down
}
```
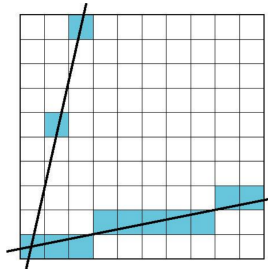
Observe the symmetry.

Question 3a

What could be the problems with the rasterization of a very thin triangle?

LECTURE 6
○○○

QUESTION 1
○○○○○○

QUESTION 2
○○○○○○

QUESTION 3
○●○○

QUESTION 4
○○○○○

In context of rasterization...

The main problem we've seen is this one:

LECTURE 6
○○○

QUESTION 1
○○○○○○

QUESTION 2
○○○○○○

QUESTION 3
○○●○

QUESTION 4
○○○○○

In context of rasterization…

Notice this would happen with thin triangles too.

LECTURE 6
○○○

QUESTION 1
○○○○○○

QUESTION 2
○○○○○○

QUESTION 3
○○○●
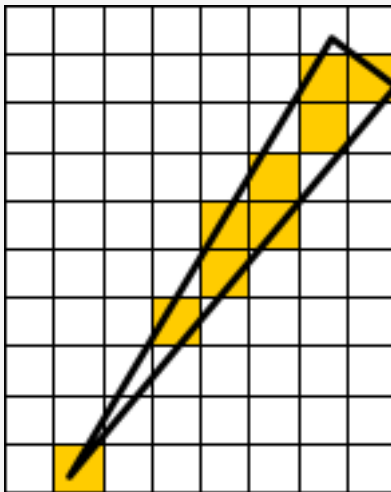
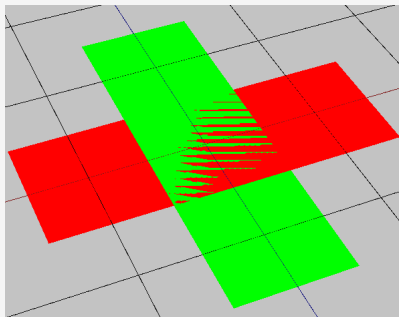QUESTION 4
○○○○○

Question 3b

Would it still be a problem if the thin triangle is part of a triangle mesh?

No, all the spaces will be filled up.

LECTURE 6
○○○

QUESTION 1
○○○○○○

QUESTION 2
○○○○○○

QUESTION 3
○○○○
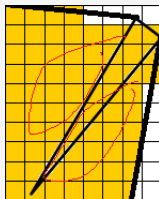
QUESTION 4
●○○○○

Question 4a

The following image shows the rendering of a green rectangle and a red rectangle in 3D space. In the overlap area of the two rectangles, there is some rendering artifact caused by z-fighting. What do you think z-fighting is? What is the cause of z-fighting in this example?

z-fighting

- 2 or more primitives have **very similar distances to camera**
- Near similar or **identical** values in the z-buffer.
- The fragment to rasterize is chosen **randomly** between the fighting fragments.

LECTURE 6
○○○

QUESTION 1
○○○○○○

QUESTION 2
○○○○○○

QUESTION 3
○○○○

QUESTION 4
○○●○○

Question 4b

When you are setting up a view volume, what can you do to minimize the chance of z-fighting?

*i.e. how to **reduce the likelihood of the primitives having identical values***

Minimizing z-fighting

1. Increase the precision (number of bits) of z-buffer value.
2. Decrease the distance between the near and far plane (how is z-buffer value calculated?)
   ○ $\frac{z-near}{near-far}$
   ○ By decreasing this distance the z-buffer distance has less significant values.

Attendance taking

Thanks! Get the slides here after the tutorial.

https://trxe.github.io/cs3241-notes