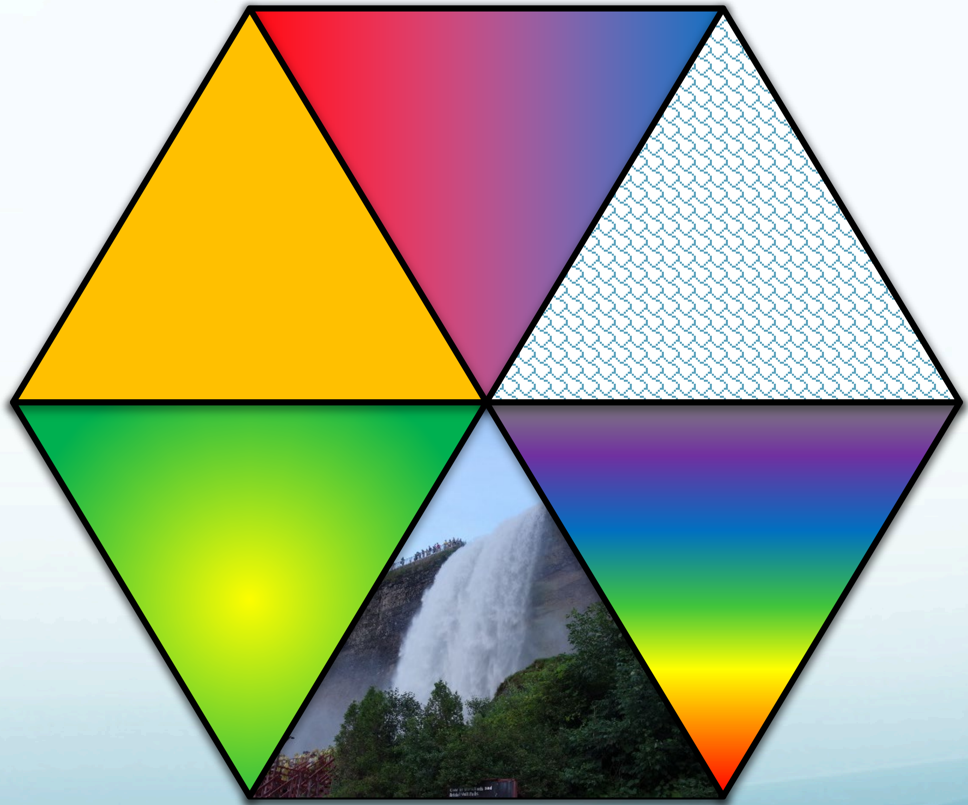
CSC 240 Computer Graphics
Video 4: Flood Fill

Nick Howe
Smith College

Portion based on slides & content courtesy Sara Mathieson

2D Fill Operation

- Replace contents of some region with new
  - Single color
  - Gradient fill
    - Linear
    - Radial
    - Multiple
  - Pattern fill
  - Image fill



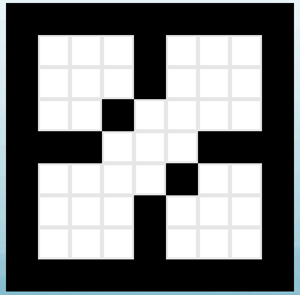
Fill Operation

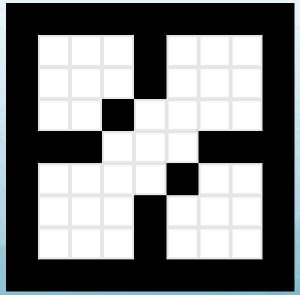
Mask fill: Mask image defines fill region

- Binary mask: 0 = no fill, 1 = fill
- Alpha mask: $\alpha \in [0,1]$ specifies amount; mix colors

Seed Fill: Start with seed pixel

- Region is all connected pixels of same color
  - **4-connected**: just cardinal connections
  - **8-connected**: cardinals+diagonals





Images: Wikipedia

Recursion Review

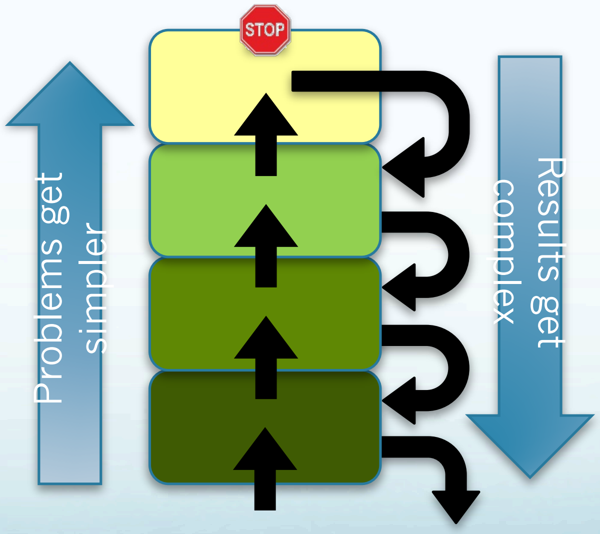
Recursion Review

Recursion = Solving problems via simplification

- Stop condition: problem so simple the solution is obvious
- Otherwise, use the same approach on a simpler problem

Pseudocode:

```
if (at stop condition) then
    return w/solution
else
    find solution based on
        simplified recursive call
endif
```



Example: WriteVertical

Given positive integer x, print out digits vertically

- Stop condition?

  (x<10) is a single digit – just print it!

- Simplification?

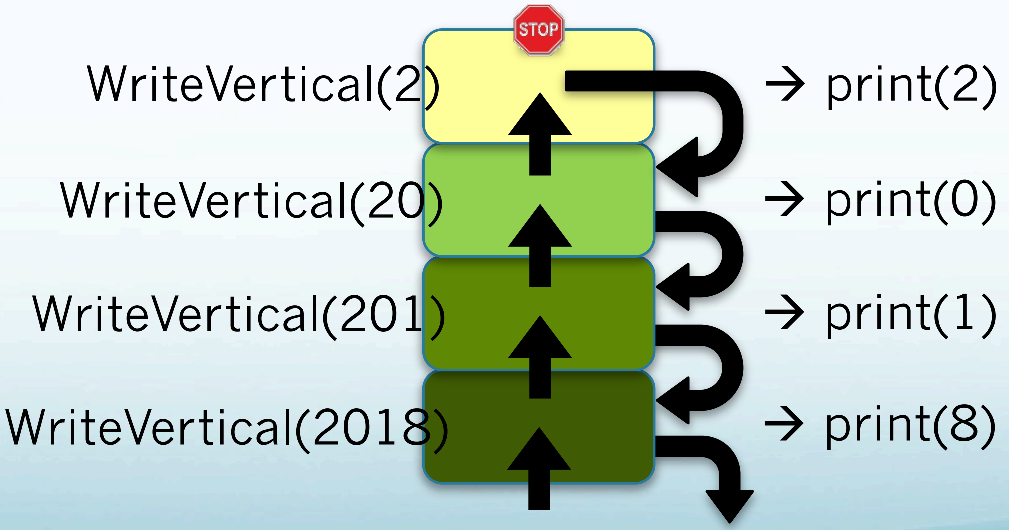
  (x%10) is last digit – make the recursive call first

```
WriteVertical(x):
    if (x < 10) then
        print x
    else
        WriteVertical(floor(x/10))
        print x%10
    endif
```

2
0
1
8

Example: WriteVertical

```
WriteVertical(x):
    if (x < 10) then
        print x
    else
        WriteVertical(floor(x/10))
        print x%10
    endif
```



WriteVertical(2) → print(2)

WriteVertical(20) → print(0)

WriteVertical(201) → print(1)

WriteVertical(2018) → print(8)

2

0

1

8

Questions

PAUSE NOW & ANSWER

1. What happens if you have a recursive function that doesn't have a stop condition?

   It will run forever.

2. What happens if the recursive call is not made on a simplified version of the problem?

   It will run forever.

3. Which is more powerful, recursion or iteration (looping)?

   They are equally powerful.
   (Anything recursion can be expressed using iteration, and vice versa.)

Recursive Flood Fill

Recursive Fill

recFill(x,y,c_0): fill all color c_0 pixels connected to (x,y)

- Stop condition?

  color(x,y) $\neq c_0$ – empty region

- Simplification?

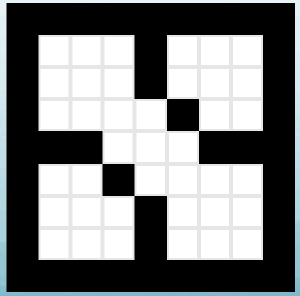
  ink(x,y); then try filling all neighbors:

  recFill(x+1,y,c_0); // east

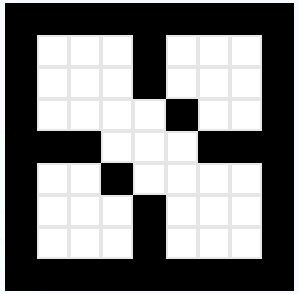
  recFill(x-1,y,c_0); // west

  recFill(x,y+1,c_0); // south

  recFill(x,y-1,c_0); // north



Art: doer.site



Recursive Fill

How many recursive calls are made?

- 4 per pixel that gets colored
- Most hit stop condition and return immediately

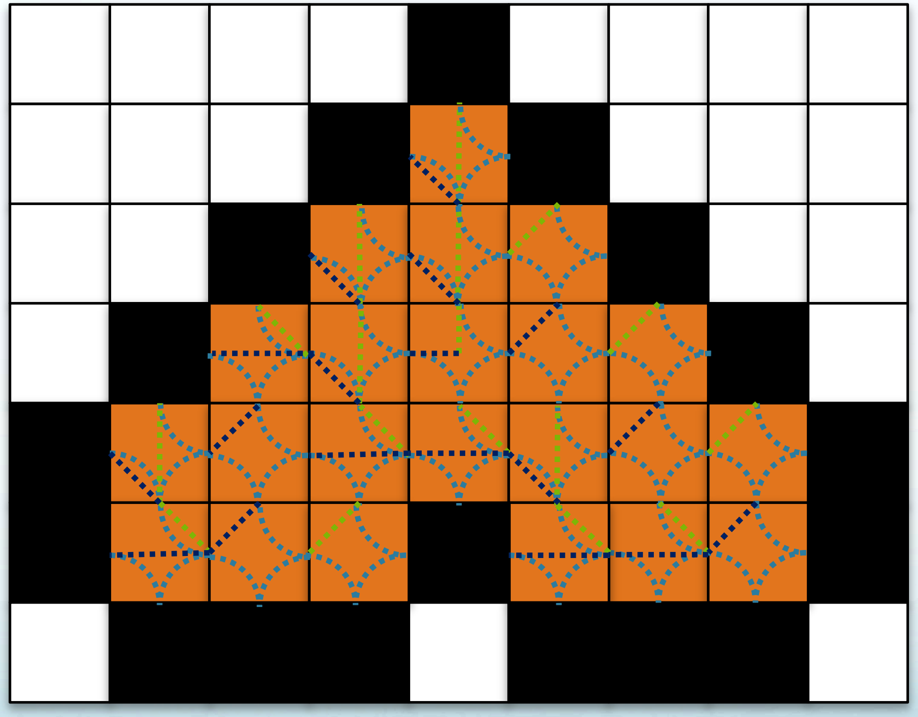
Does the order of the recursive calls matter?

- It changes the coloring order but not the final result
- Normally the order is not visible to a user

Stack overflow: too many recursive calls in a row

Recursive Fill

- In what order would this fill color the pixels below? (assume recursion is N-E-S-W)



DONE!

Questions

PAUSE NOW & ANSWER

1. If we change the order of the recursive calls, does it change the final result (in terms of which pixels are filled)?

   No.

2. If we change the order of the recursive calls, does it change the order in which pixels are filled?

   Yes, in most cases.

3. Why is recursion a good way to implement the fill operation?

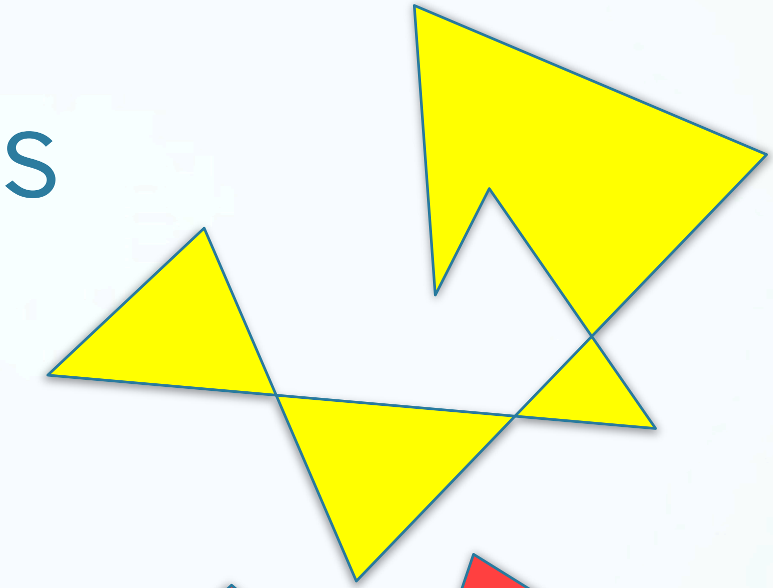
   The stored sequence of recursive calls serves to remember all the places we still have to check. If we didn't use recursion, we would need to create a data structure (stack or queue) to keep track of this.

Scanline Fill

Complex Polygons

Flood fill won’t work on complex polygons.

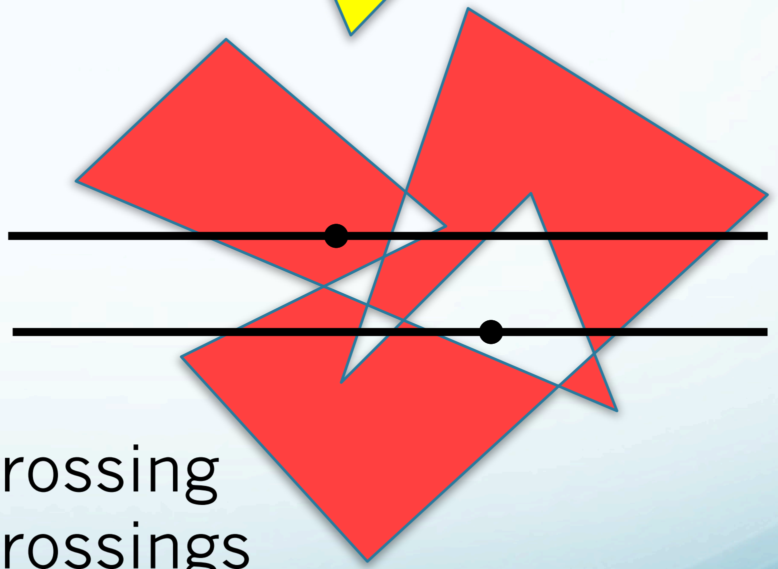
- Disconnected interiors
- What is interior when lines overlap?



Solution: count edge crossings from exterior

- Odd # crossings = interior point
- Even # crossing = exterior point

Doesn’t matter which way you draw the line!

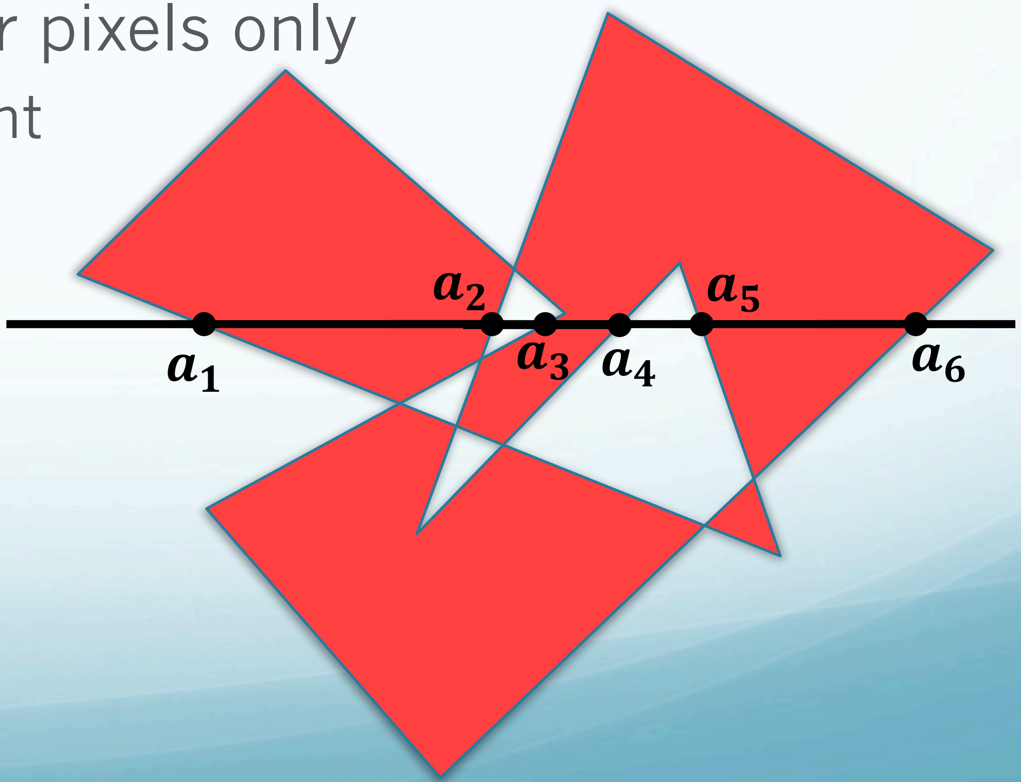


1 crossing
5 crossings
4 crossings
2 crossings

Scanline Fill

Scanline fill isn't based on drawn lines like flood fill

- Need a mathematical description: endpoints of all edge segments
- Concept: sweep across rows, filling interior pixels only
  - First thought: count crossings for each point
  - Second thought:
    - Compute all intersections for each scanline
    - Sort them left to right
    - Fill between every other pair
      E.g., $a_1 \rightarrow a_2$, $a_3 \rightarrow a_4$, $a_5 \rightarrow a_6$

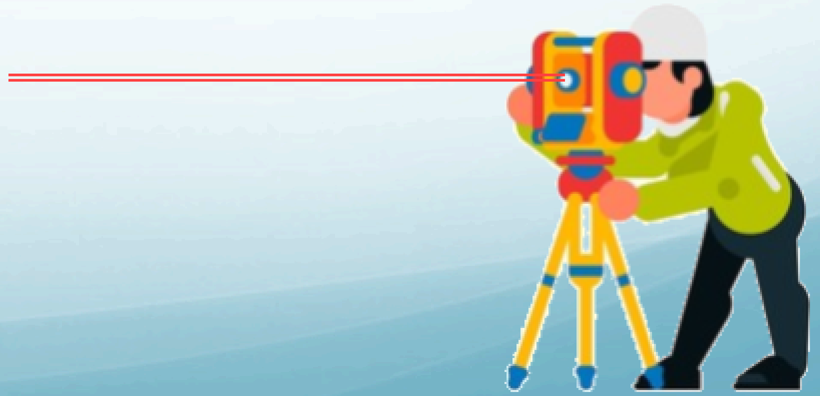


Scanline Algorithm

Input: Sequential vertices $(x_0, y_0), (x_1, y_1), \ldots (x_n, y_n)$, where $(x_0, y_0) = (x_n, y_n)$

Goal: Fill complex polygon delineated by these vertices

```
Loop for y from $\mathbf{min}(\{\boldsymbol{y_0}, \ldots \boldsymbol{y_n}\})$ to $\mathbf{max}(\{\boldsymbol{y_0}, \ldots \boldsymbol{y_n}\})$
  C = [ ]
  Loop for i from 0 to n-1
    If edge $(x_i, y_i)$ to $(x_{i+1}, y_{i+1})$ intersects scanline y at $(x_c, y_c)$:
       C.append($(x_c, y_c)$)
  Sort C by the x coordinates
  Loop for j from 0 to Size(C) step by 2
    Loop for x from C[j] to C[j+1]
       Fill(x,y)
```



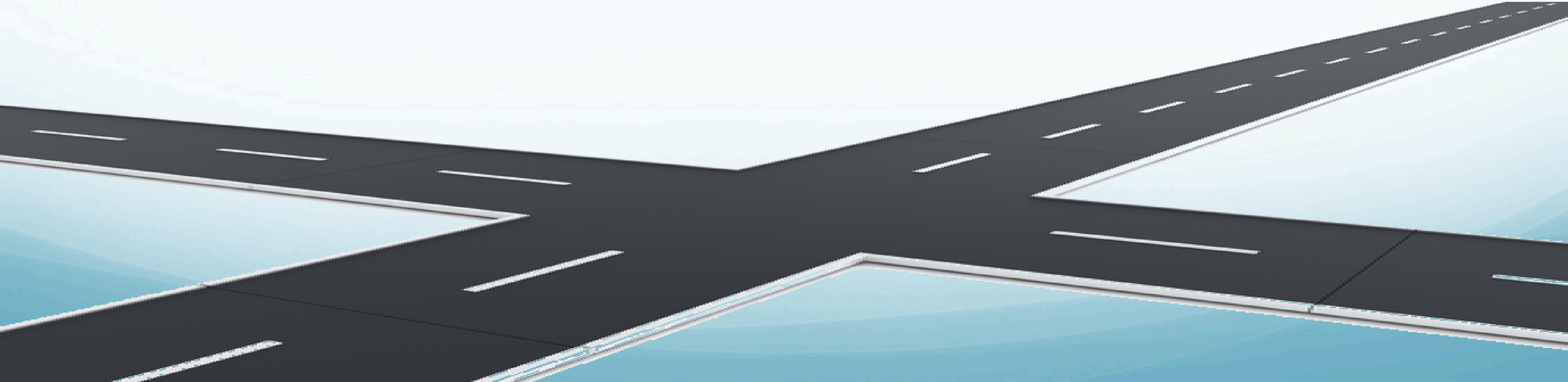
Scanline Details

Intersection of edges with scanlines:

- One endpoint **below y**, the other **on or above y**

$$[(y_i < y) \land (y_{i+1} \geq y)] \lor [(y_i \geq y) \land (y_{i+1} < y)]$$

- Use line equation to find intersection point: $x = \frac{x_i - x_{i+1}}{y_i - y_{i+1}} y + \frac{x_{i+1} y_i - x_i y_{i+1}}{y_i - y_{i+1}}$



Questions

PAUSE NOW & ANSWER

1. In the recursive fill operation, the maximum recursive call depth is most closely proportional to...
   a. The number of pixels to be filled
   b. The number of rows to be filled
   c. The number of columns to be filled
   d. The number of rows times the number of columns
   e. A constant

a. The number of pixels to be filled

2. In the scanline fill operation, the number of lines that must be checked for edge intersections is most closely proportional to...

(Same set of options as above.)

b. The number of rows to be filled

Review

After watching this video, you should be able to...

- Define the 2D fill operation
- Determine 4-connected and 8-connected regions
- Design a recursive function with stop condition & simplification
- Implement a recursive 2D fill algorithm
- Pseudocode & simulate a scanline 2D fill algorithm
- Explain the advantages of the scanline fill.

Music: https://www.bensound.com