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: $0 = \text{no fill}, \ 1 = \text{fill}$
- 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

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

Problems get simpler
Results get complex
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

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

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

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 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 fill 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
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)
Questions

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.
Sweep Fill
How can we reduce the number of recursive calls?

- Use a loop (iteration) instead
- Fill one entire row per function call
- After sweep, call on rows above & below (mix iteration with recursion)

**Pseudocode:**

- Fill pixels to the left
- Fill pixels to the right
- Scan left to right, calling above & below
Sweep Fill

- In what order would sweep fill color the pixels below?

Pseudocode:
- Fill pixels to the left
- Fill pixels to the right
- Scan left to right, calling above & below
Sweep Fill

- In what order would sweep fill color the pixels below?

Pseudocode:
- Fill pixels to the left
- Fill pixels to the right
- Scan left to right, calling above & below
Questions

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

2. In the sweep fill operation, the maximum recursive call depth is most closely proportional to...
   (Same set of options as above.)
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 sweep-based 2D fill algorithm
- Explain the advantages of the sweep-based fill.

Music: [https://www.bensound.com](https://www.bensound.com)