

# CSC 240

# Computer Graphics

Fall 2015  
Smith College

# Outline: 9/16

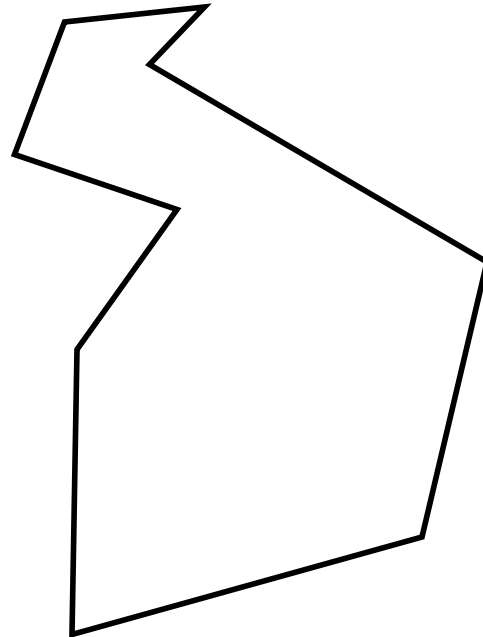
- Polygons
- Flood fill algorithm
- Better fill algorithm

# Polygon

- Chain of line segments that form a closed loop

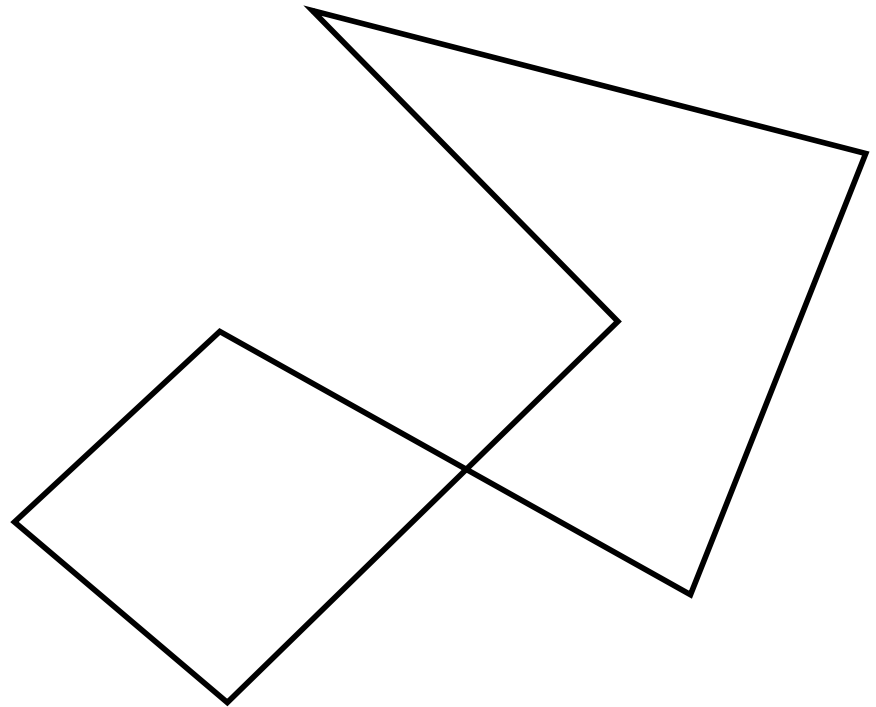
# *Simple* Polygon

- no self intersections



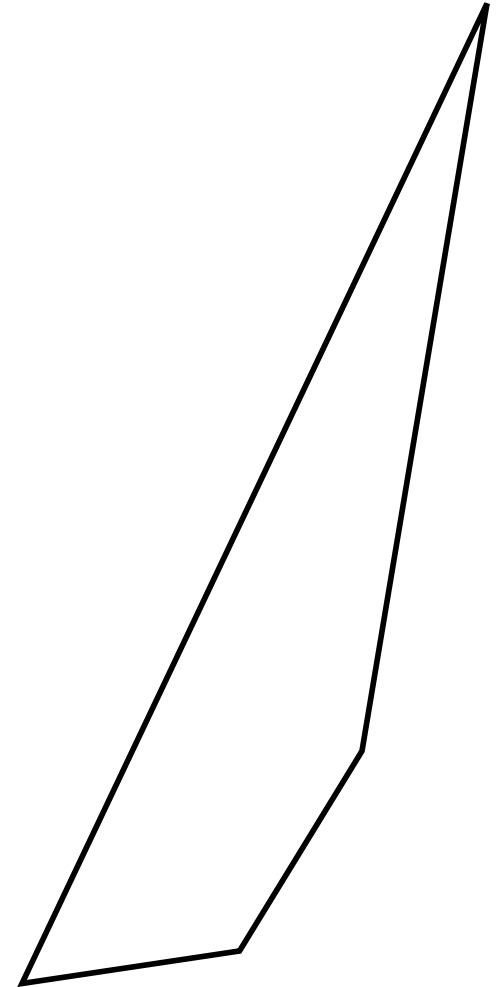
# *Complex Polygon*

- self intersections



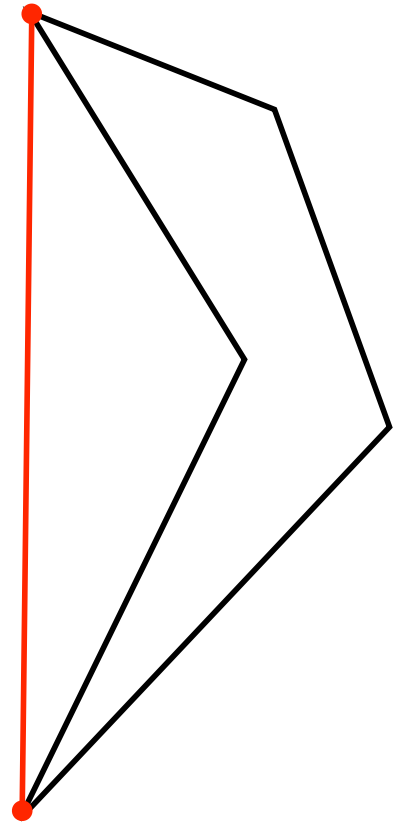
# Convex Polygon

- Every internal angle is less than or equal to 180 degrees.
- (all vertices point outward)
- Every line segment between two vertices remains inside or on the boundary of the polygon.
- (no dents)



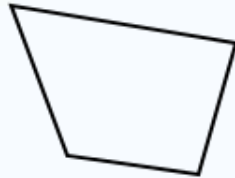
# Concave Polygon

- There exist an internal angle is greater than 180 degrees.
- (at least one vertex points inward)
- There exists at least one line segment between two vertices that exits the boundary of the polygon.
- There is a “dent” or “cave”



# More polygons

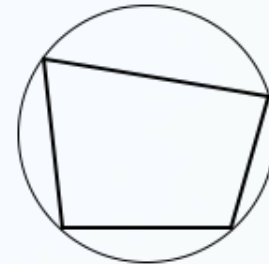
**Simple**



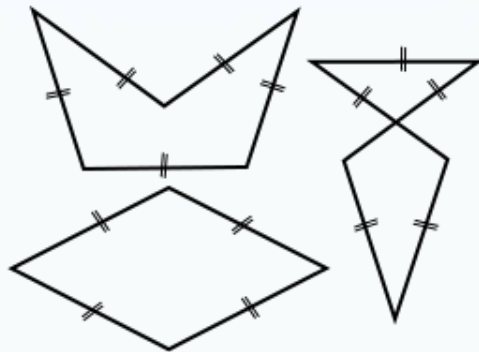
**Convex**



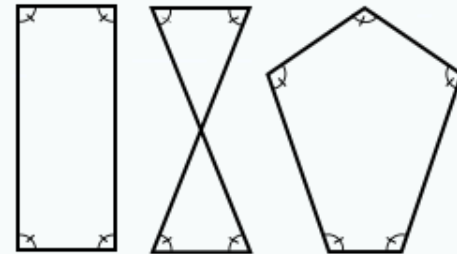
**Concave**



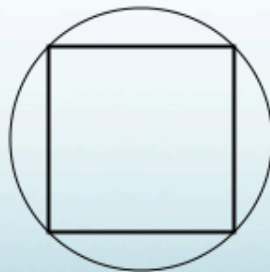
**Cyclic**



**Equilateral**



**Equiangular**



**Regular convex**

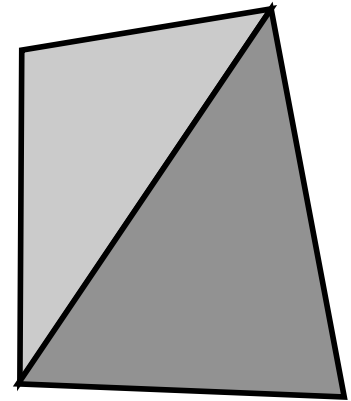


**Regular star**



# Qualities of a good Polygon Algorithm

- good line
  - approximates line
  - constant weight
  - fast
- No cracks between adjacent polygons



# Flood Fill

- Pick a pixel inside the fill area
  - Fill the pixel
  - Repeat with neighboring non-edge pixels

# Recursion!

## Lab 3

# Flood Fill: example code

```
# flood_fill takes a pixel location (x,y), an old_color (what we want
# to replace), and a new_color (replacement color)
def flood_fill(img, x, y, old_color, new_color):

    curr_color = img.getPixel(x, y)

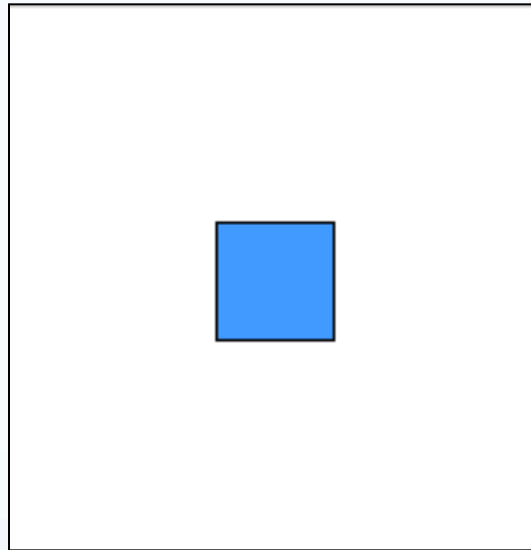
    if curr_color == new_color:
        return # nothing to do, finished
    if curr_color != old_color:
        return # hit an edge, finished

    img.setPixel(x, y, new_color) # curr_color must have been old_color, "fill" it

    # recurse!
    flood_fill(img, x-1, y, old_color, new_color) # east
    flood_fill(img, x+1, y, old_color, new_color) # west
    flood_fill(img, x, y-1, old_color, new_color) # south
    flood_fill(img, x, y+1, old_color, new_color) # north

    return
```

# Flood Fill result



# Better Fill Algorithm

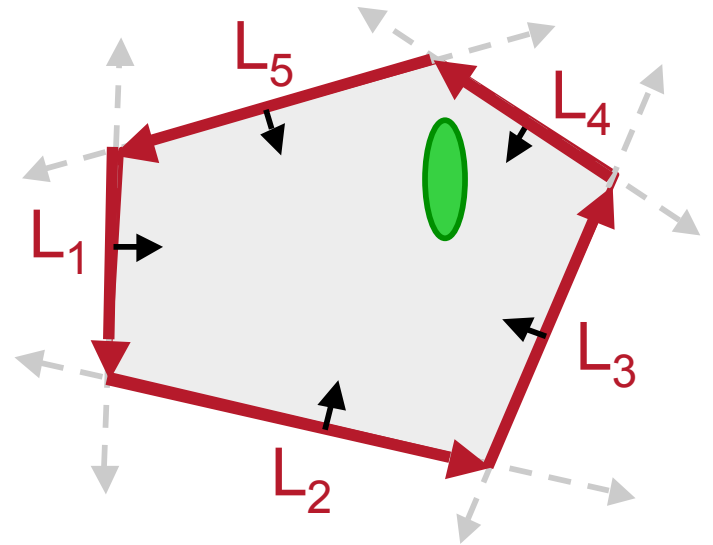
- Triangle sweep

# Polygons

- How do we know if we are inside?

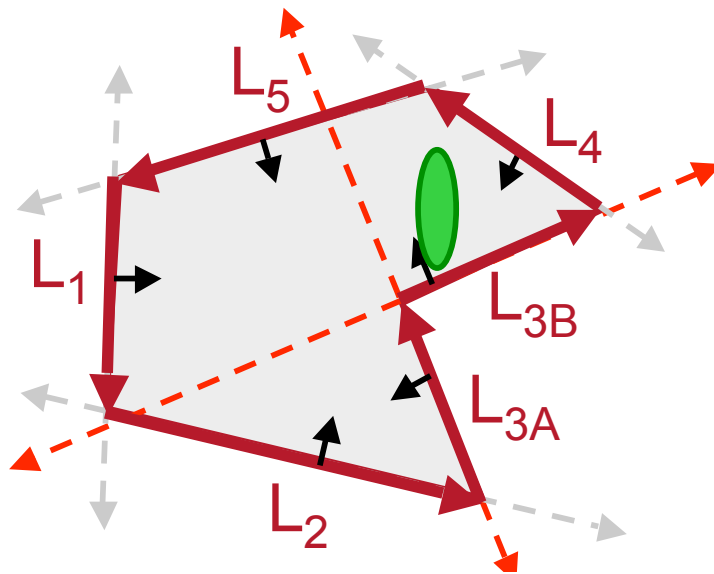
# Convex Polygons

- Triangle Method





# Concave



# Polygon Sweep Algorithm

