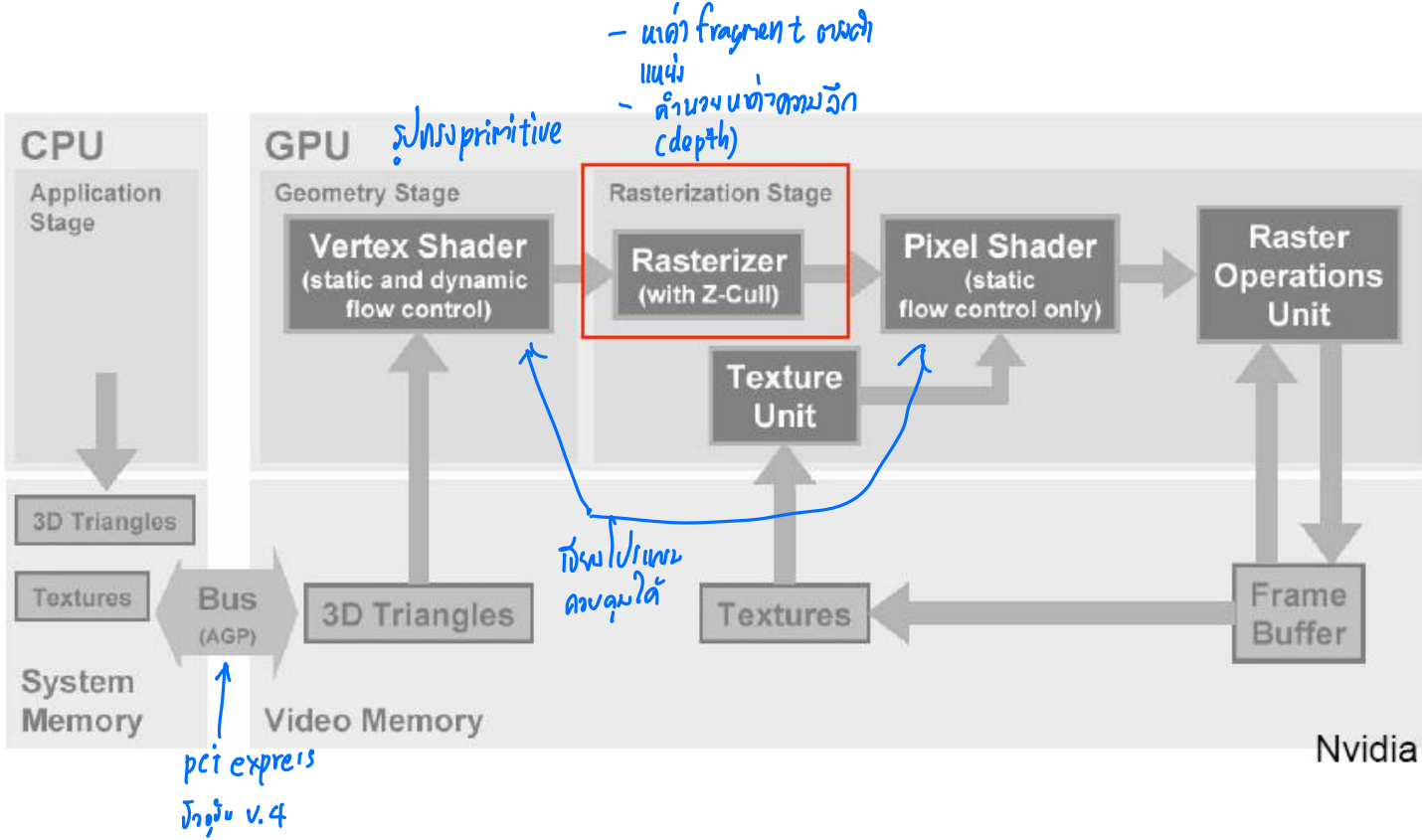
- แปลงค่าพิกัดจุด vertex ในช่วงที่เป็น Raster เป็นช่วงของ fragment

Rasterization

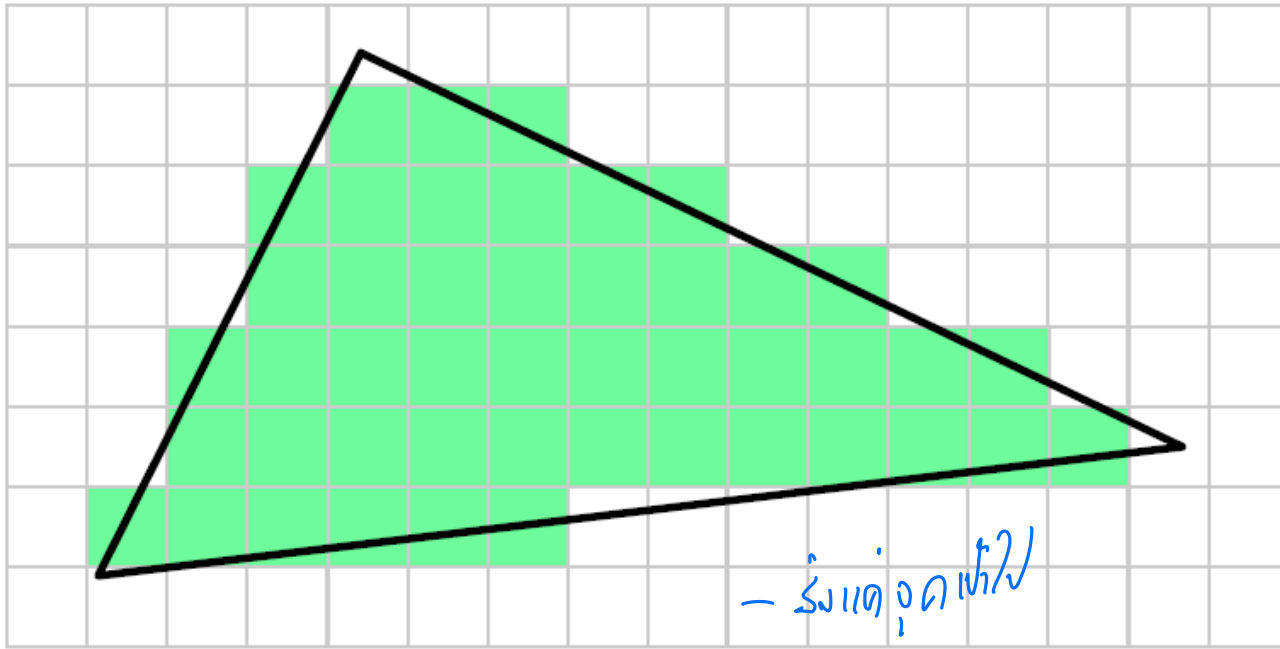
3D Graphics Pipeline

- The rasterization step scan converts the object into pixels



Rasterization (scan conversion)

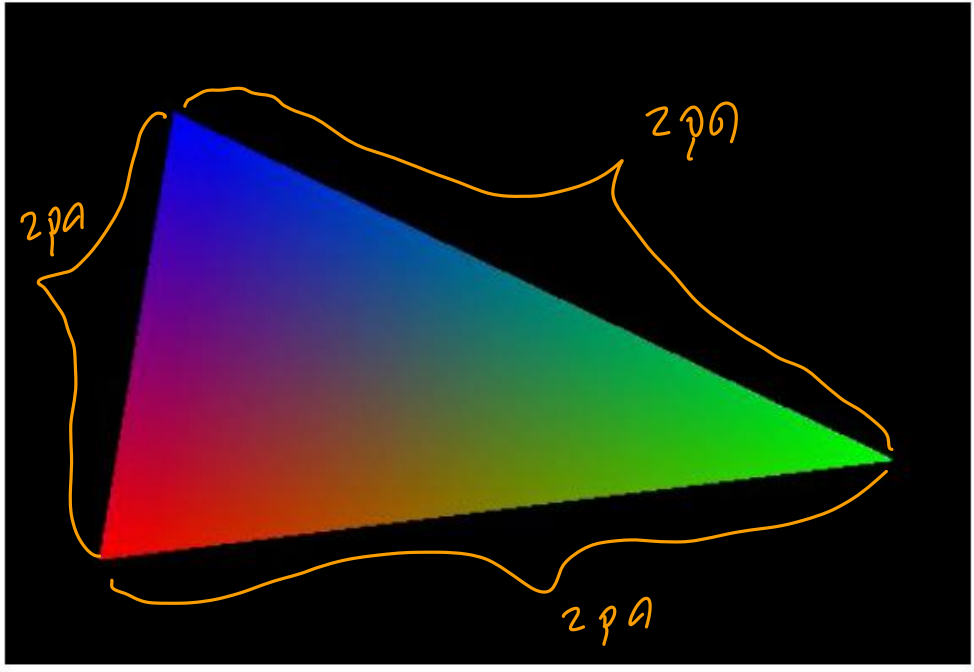
- Determine which fragments get generated
- Interpolate parameters (colors, texture coordinates, etc.)



Parameter interpolation

- What does “interpolation” mean?
- Example: colors

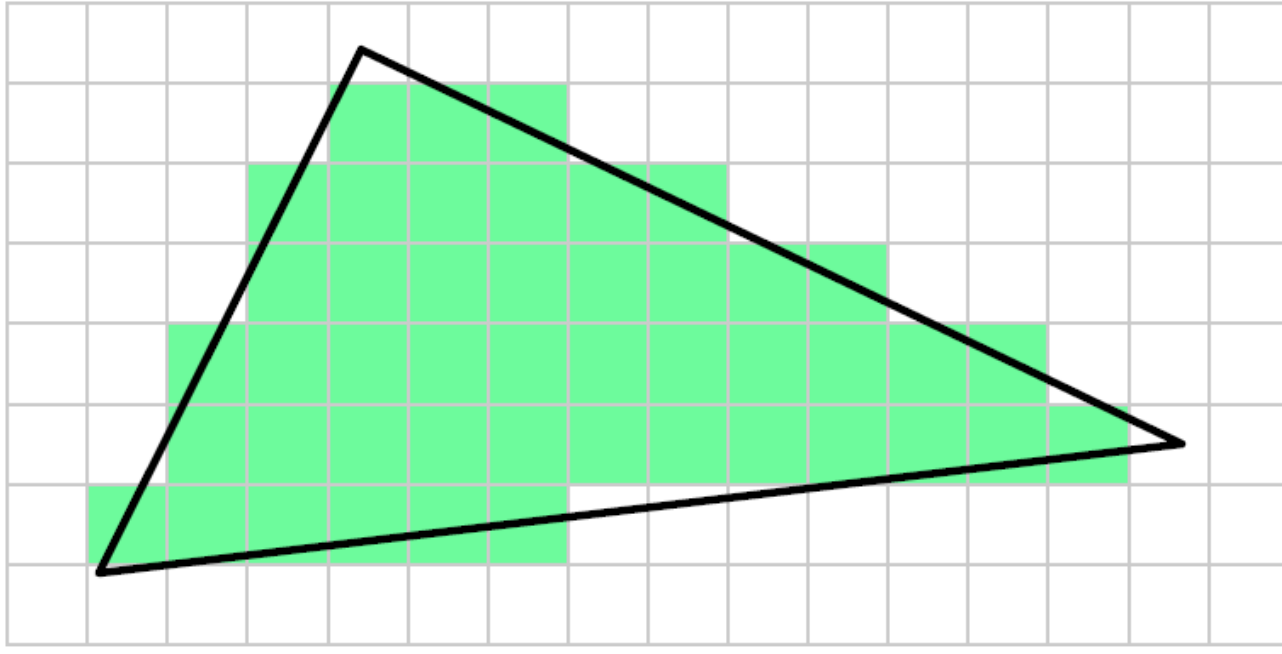
↳ เฉลี่ยแบบ linear interpolation



One Triangle

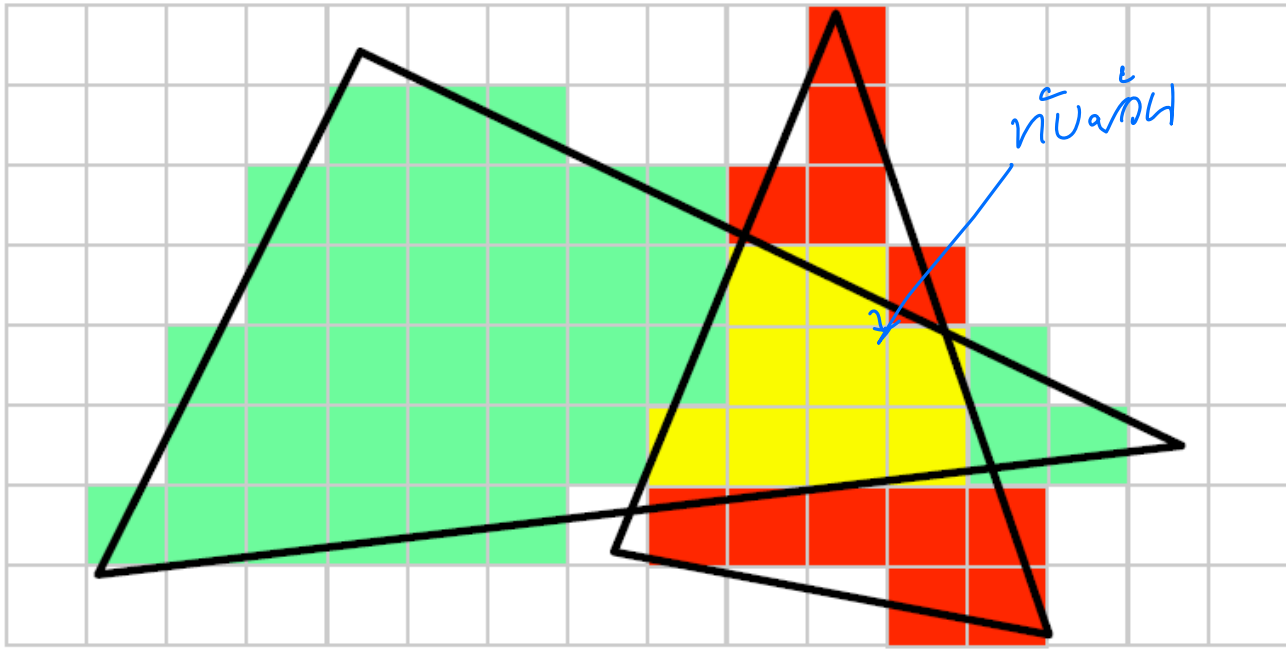
- With one triangle, things are simple
- Fragments never overlap!

fram buffer



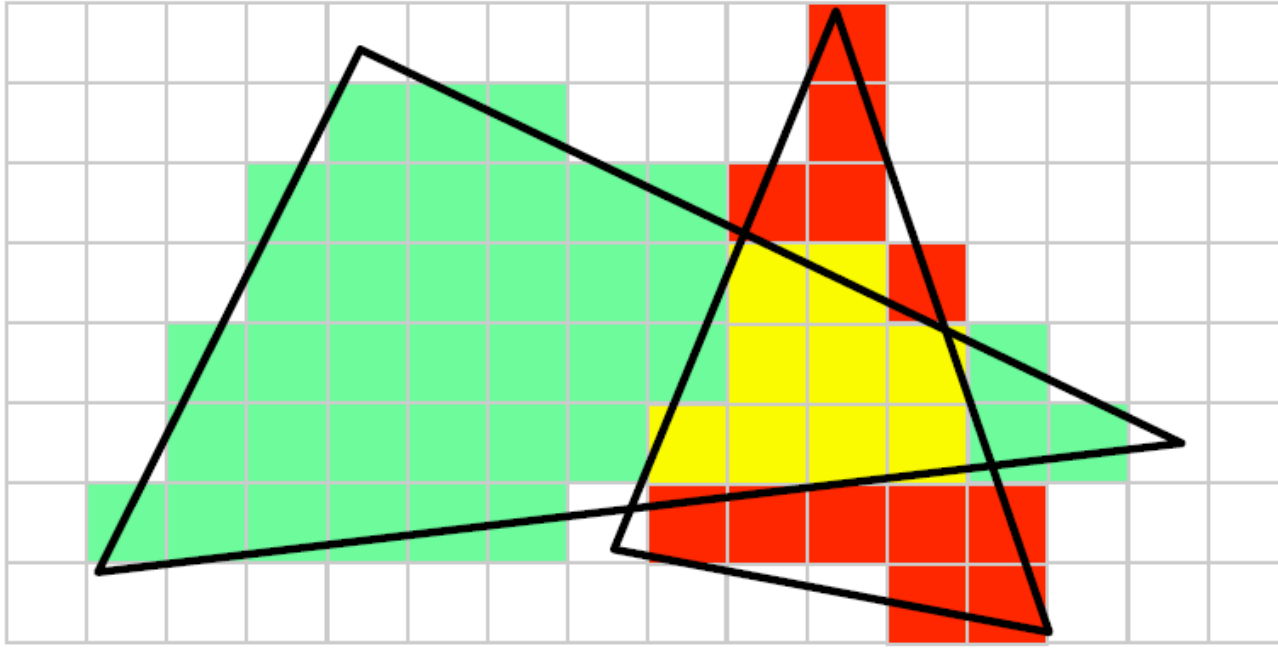
Two Triangles

- Things get more complicated with multiple triangles
- Fragments might overlap in screen space!



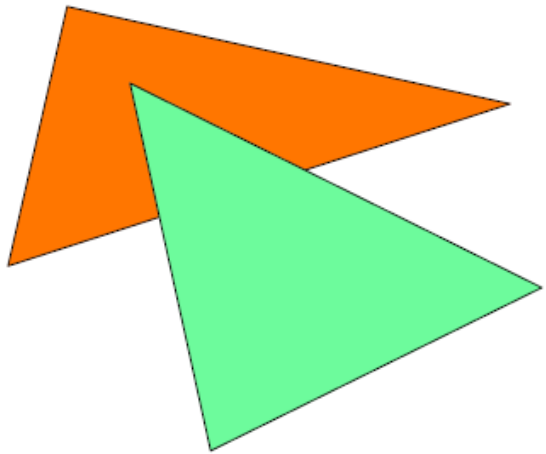
Fragments vs. Pixels

- Each pixel has a unique framebuffer (image) location
- But multiple fragments may end up at same address

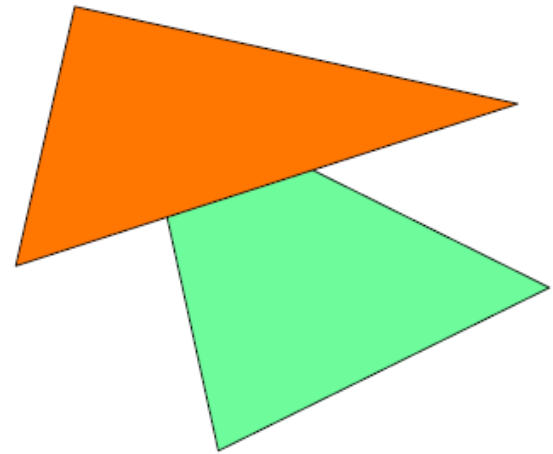


Which triangle wins?

- Two possible cases:



green triangle on top

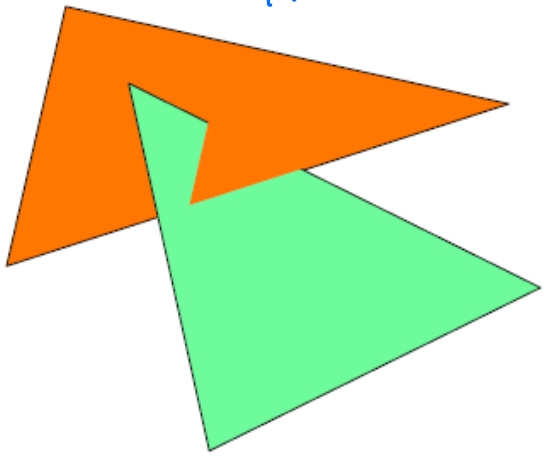


orange triangle on top

Which (partial) triangle wins?

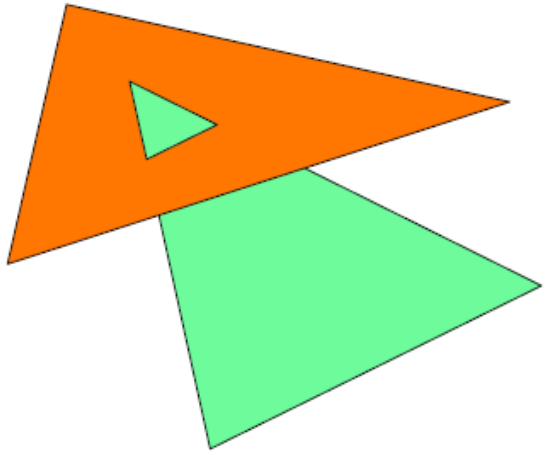
- Many other cases possible!

เฉือน



intersection #1

ทิ่ม



intersection #2

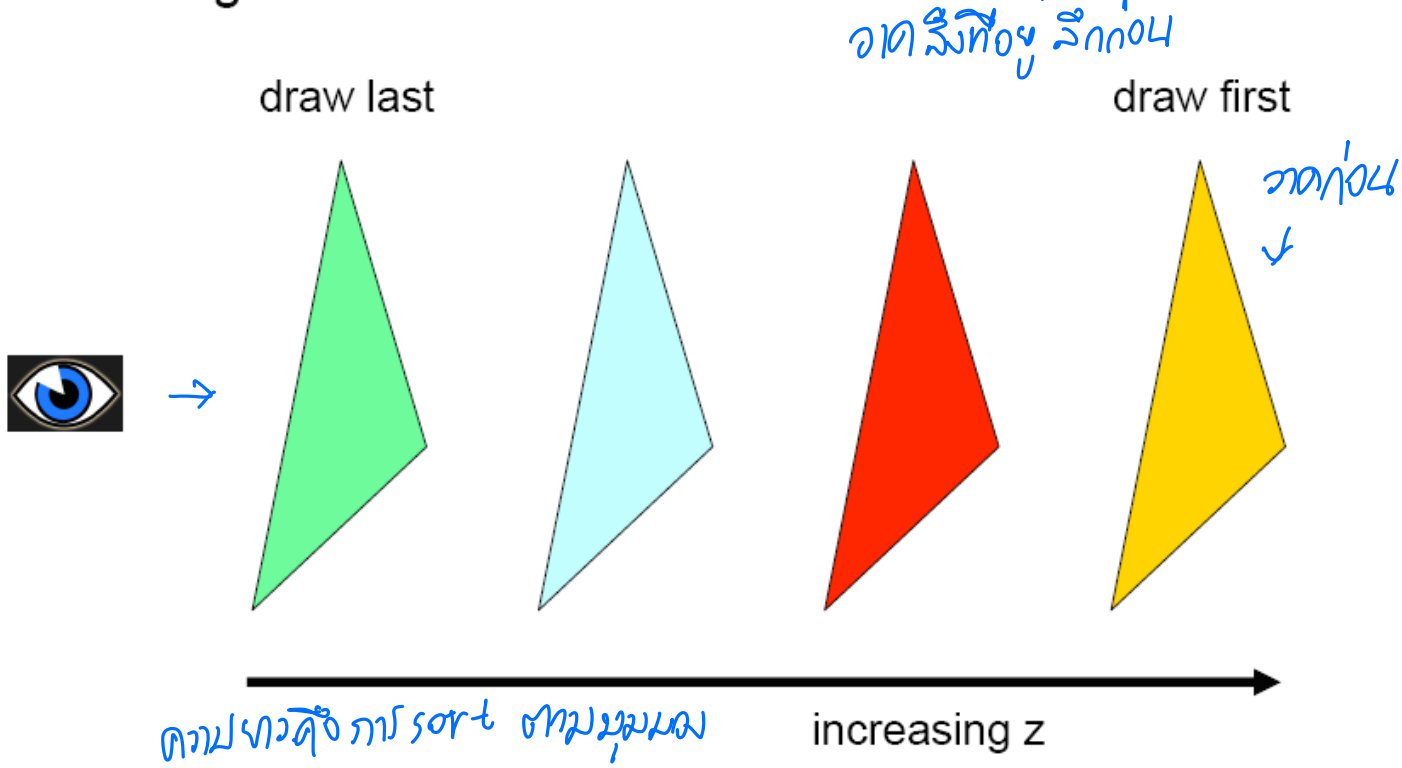
Hidden Surface Removal

วิธีการวาดภาพที่ไม่ขึ้นกับลำดับของการวาด

- Idea: keep track of visible surfaces
- Typically, we see only the front-most surface
- Exception: transparency

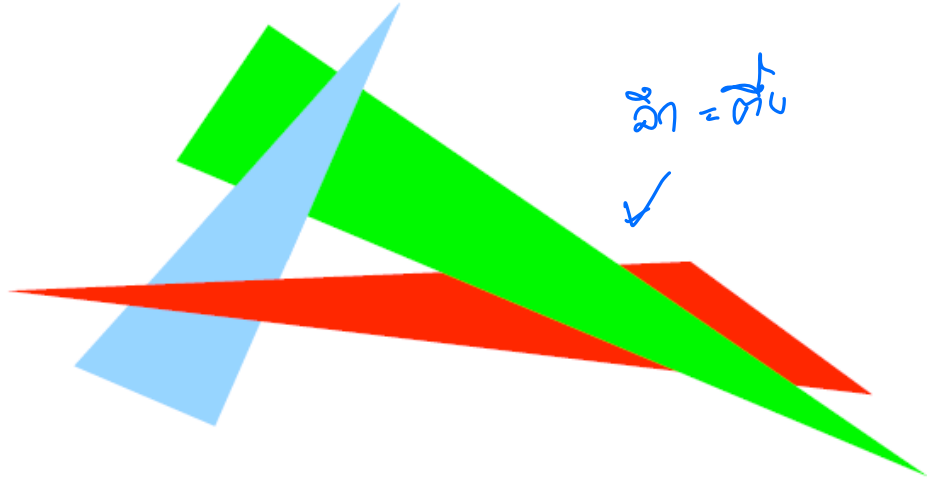
First Attempt: Painter's Algorithm

- Sort triangles (using z values in eye space)
- Draw triangles from back to front



Problems? ปัญหาที่พบกับ

- Correctness issues:
  - Intersections
  - Cycles
  - Solve by splitting triangles, but ugly and expensive
- Efficiency (sorting)



The Depth Buffer (Z-buffer)

Buffer ใช้ตัดรูปตามลำดับ

- Perform hidden surface removal per-fragment
- Idea:
  - Each fragment gets a z value in screen space
  - Keep only the fragment with the smallest z value

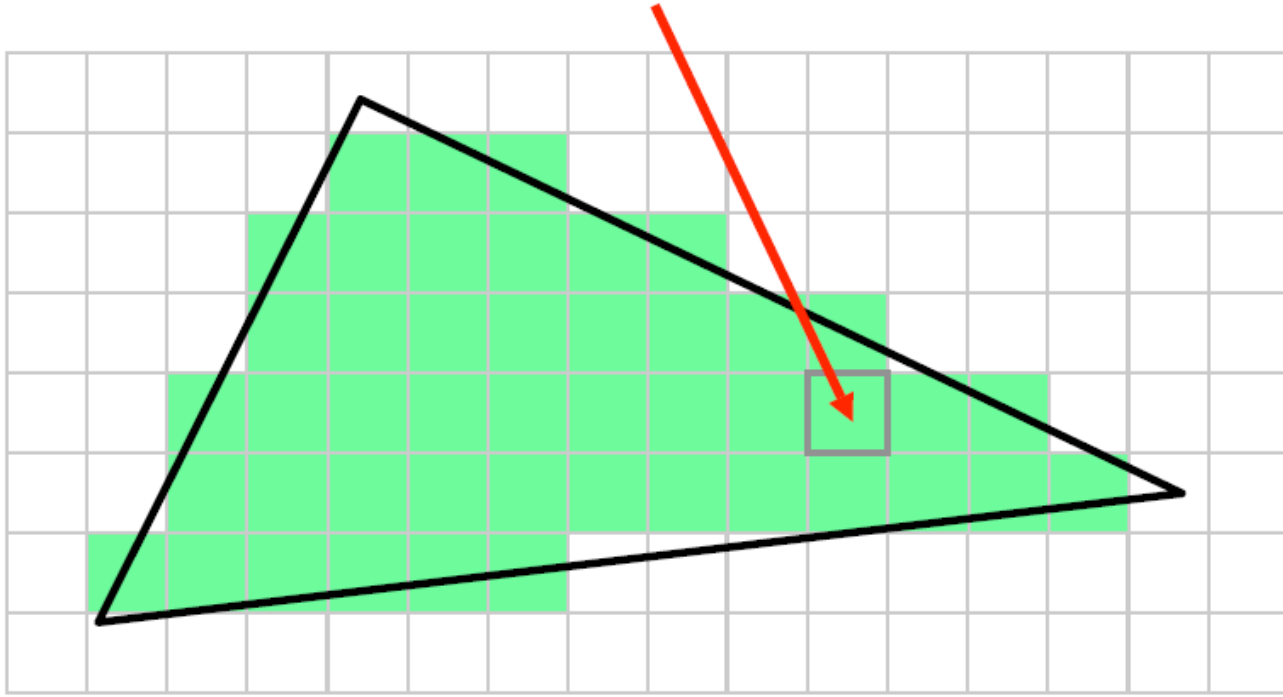
ตัดรูปทีระดับ fragment Rasterization รูปเรขาคณิต → fragment

ในแต่ละ fragment จะมีการเก็บค่าตำแหน่งบนหน้าจอ และค่า z ที่ลึกตามลำดับของ fragment

depth buffer เก็บค่า z ของ fragment ที่มีค่าน้อยที่สุด

The Depth Buffer (Z-buffer)

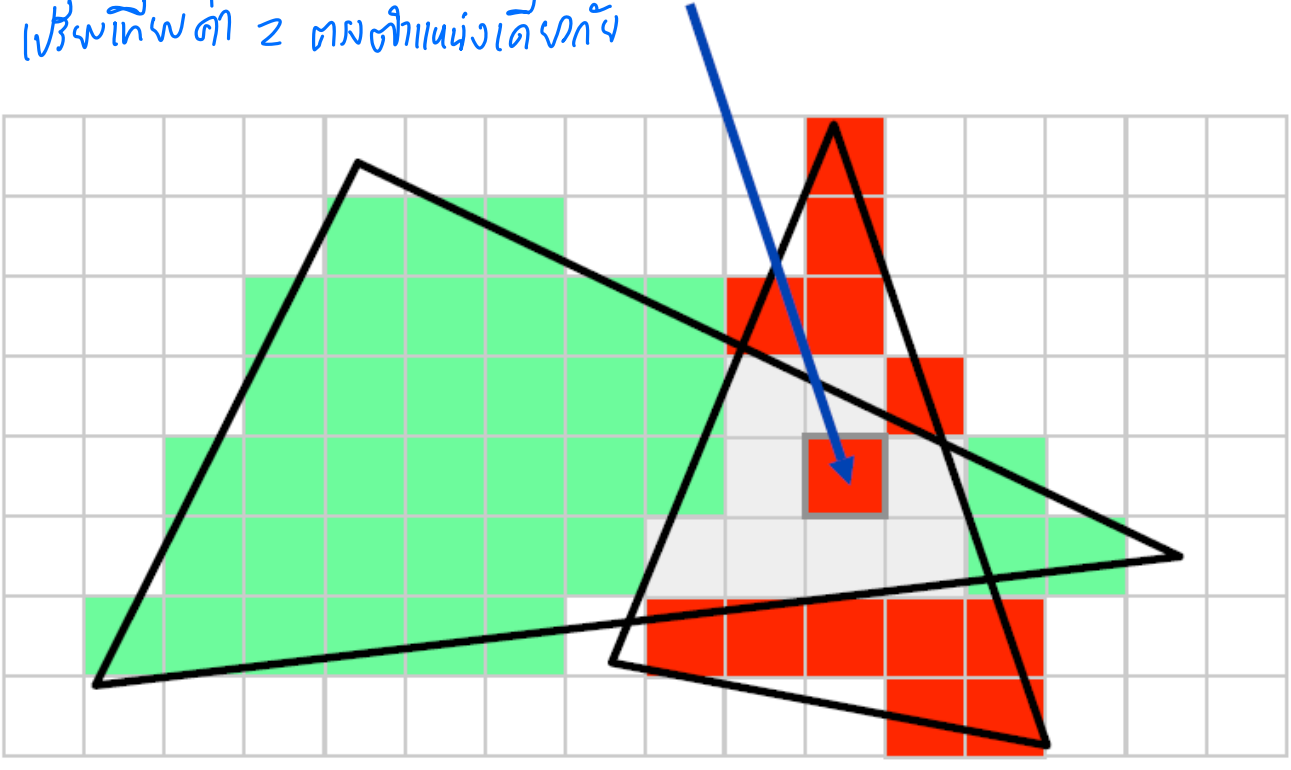
- Example:
  - fragment from green triangle has z value of 0.7



The Depth Buffer (Z-buffer)

- Since 0.3 < 0.7, the red fragment wins

เปรียบเทียบค่า z ตามตำแหน่งเดียวกัน



The Z-buffer

- Lots of fragments might map to the same pixel location
- How to track their z-values? gl clear Depth Buffer Bit
- Solution: z-buffer (2D buffer, same size as image)

1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.1	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.1	0.1	1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.2	0.2	0.3	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.3	0.3	0.4	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.3	0.4	0.4	0.5	1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.4	0.4	0.5	0.5	0.5	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.4	0.5	1.0	1.0	1.0

- เริ่มก็ต้องมาทำการ clear ค่า โดยให้แต่ละ buffer เป็น ค่าที่มากที่สุดที่จะเป็นไปได้ เอาไปเปรียบเทียบกับค่าที่เข้ามา

Z-buffer Algorithm

color buffer

- Let CB be color buffer, ZB be z-buffer
- Initialize z-buffer contents to 1.0 (far away) ไกลต่นู้ดูที่รุด

Z-buffer Algorithm

- Let CB be color buffer, ZB be z-buffer
- Initialize z-buffer contents to 1.0 (far away)
- For each triangle T
  - Rasterize T to generate fragments
    - For each fragment F with screen position (x,y,z) and color value C
      - If (z < ZB[x,y]) then
        - Update color: CB[x,y] = C
        - Update depth: ZB[x,y] = z

Z-buffer Algorithm Properties

- What makes this method nice? อัลกอริทึม
  - simple (faciliates hardware implementation)
  - handles intersections polygon มีการซ้อนทับ
  - handles cycles
  - draw opaque polygons in any order

    วาด polygon โดยไม่ต้องคำนึงถึงลำดับ