# Lecture 24: The Pipeline

## One Triangle's Trip, and the Depth Buffer

**CS 453 — Computer Graphics**

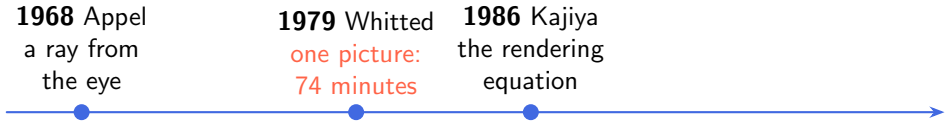
Fakhir Shaheen

Forman Christian University

# Rays for the best picture, triangles for the next frame

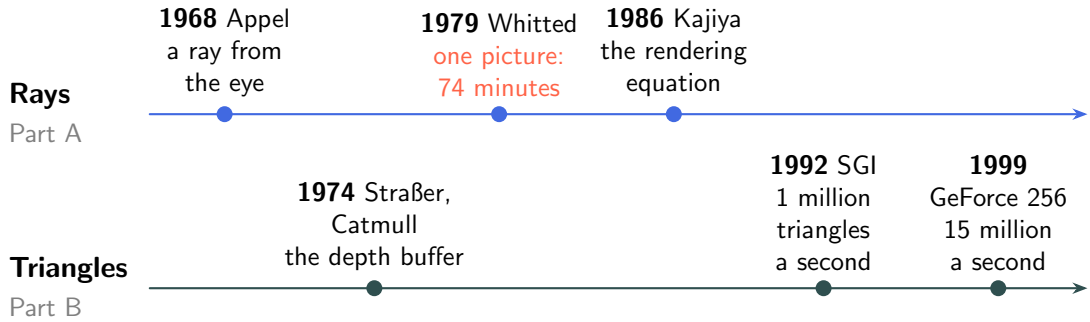
**Rays**

Part A



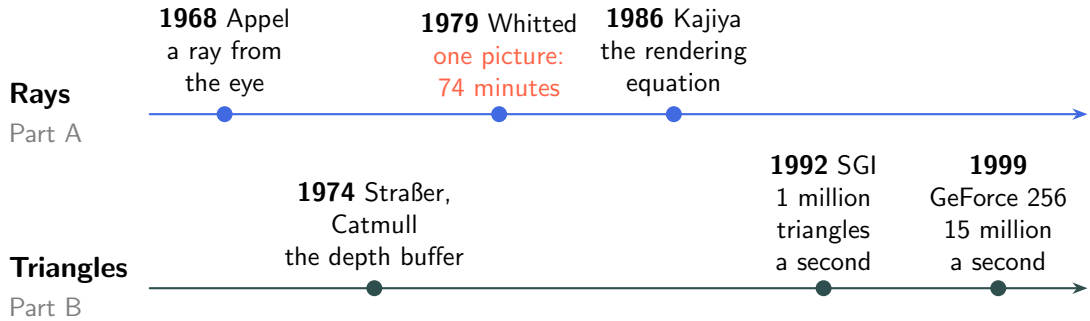

- Rays: quality first, however long it takes

# Rays for the best picture, triangles for the next frame




- Rays: quality first, however long it takes
- Triangles: one frame in 16.7 ms, and fake what doesn't fit

# Rays for the best picture, triangles for the next frame




- Rays: quality first, however long it takes
- Triangles: one frame in 16.7 ms, and fake what doesn't fit
- Part A walked the top line; from today, Part B walks the bottom one

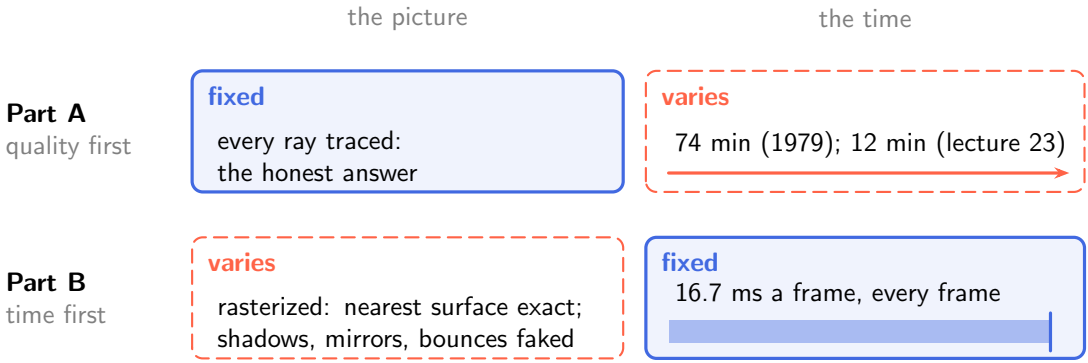
# The big idea: Part A fixes the picture, Part B fixes the time

| | the picture | the time |
|---|---|---|
| **Part A**<br>quality first | **fixed**<br>every ray traced:<br>the honest answer | **varies**<br>74 min (1979); 12 min (lecture 23) → |

# The big idea: Part A fixes the picture, Part B fixes the time

| | the picture | the time |
|---|---|---|
| **Part A** quality first | **fixed** every ray traced: the honest answer | **varies** 74 min (1979); 12 min (lecture 23) → |
| **Part B** time first | **varies** rasterized: nearest surface exact; shadows, mirrors, bounces faked | **fixed** 16.7 ms a frame, every frame |

# The big idea: Part A fixes the picture, Part B fixes the time

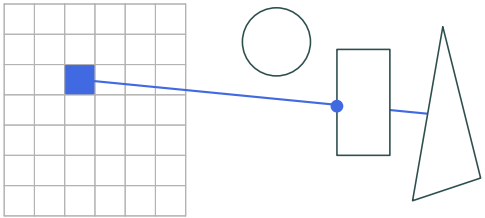



- Every Part B lecture: what fits in 16.7 ms, and what it costs the picture

# The two loops swap places

**Part A**: image order (lecture 15, slide 17)

```
for each pixel:
    for each object:
        keep the nearest hit
```



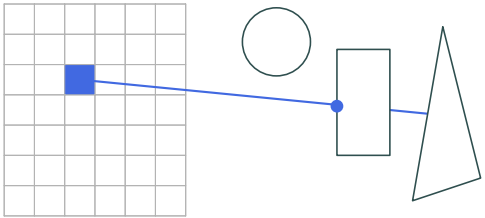
one pixel asks every object

- Same two loops

# The two loops swap places

**Part A**: image order (lecture 15, slide 17)

```
for each pixel:
    for each object:
        keep the nearest hit
```

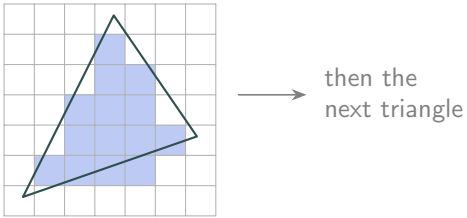


one pixel asks every object

- Same two loops

**Part B**: object order (today)

```
for each triangle:
    for each pixel it covers:
        keep the nearest depth
```



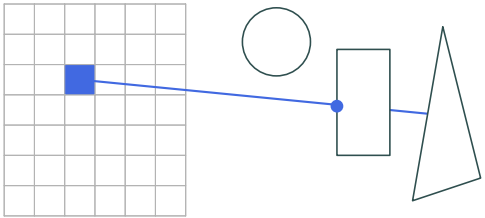

one triangle visits its pixels

- Swapped: the triangle is outside

# The two loops swap places

**Part A**: image order (lecture 15, slide 17)

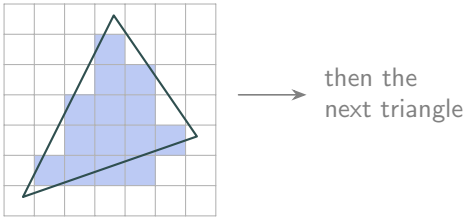
```
for each pixel:
    for each object:
        keep the nearest hit
```



one pixel asks every object

**Part B**: object order (today)

```
for each triangle:
    for each pixel it covers:
        keep the nearest depth
```

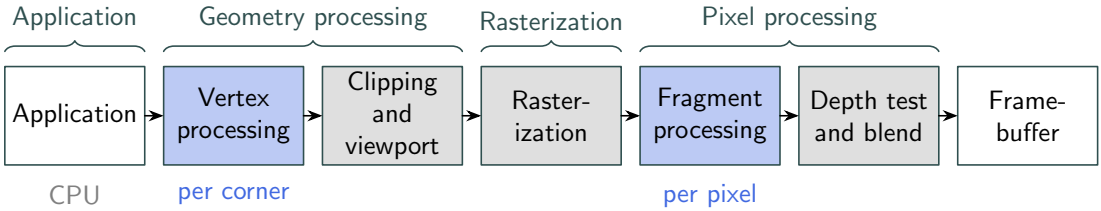



one triangle visits its pixels

- Same two loops
- Swapped: the triangle is outside
- Each triangle read once

# Seven stops from three corners to a picture

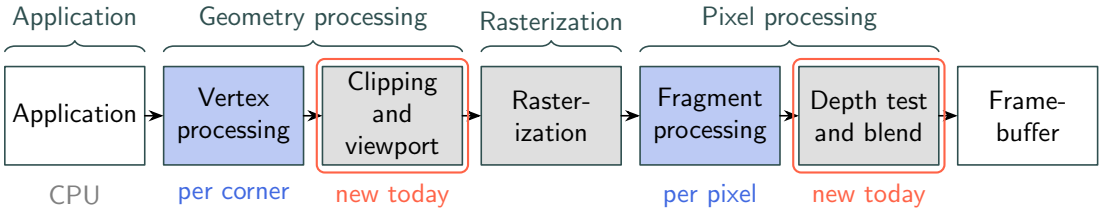
lecture 23, slide 40's boxes, with two stops named




- Blue: your program, run many times
- Grey: a circuit with settings

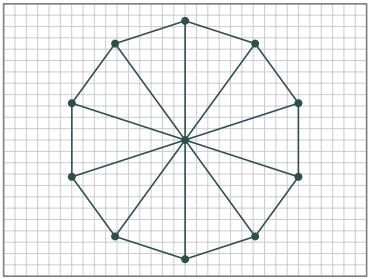
# Seven stops from three corners to a picture

lecture 23, slide 40's boxes, with two stops named




- Blue: your program, run many times
- Two stops the row hid: clipping, depth
- Grey: a circuit with settings

# Ten triangles: you send 30 corners and one command



a disc from 10 triangles

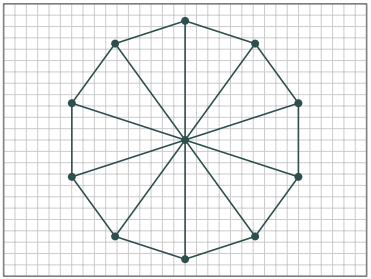
a buffer: 30 vertices



3 corners per triangle

- You send corners, not pixels

# Ten triangles: you send 30 corners and one command



a disc from 10 triangles

a buffer: 30 vertices



3 corners per triangle

two small programs

vertex program

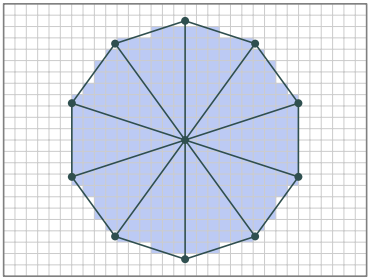
fragment program

one command

`draw(30)`

- You send corners, not pixels
- Plus two programs and one draw

# Ten triangles: you send 30 corners and one command



a disc from 10 triangles

324 pixels covered

a buffer: 30 vertices



3 corners per triangle

two small programs

vertex program

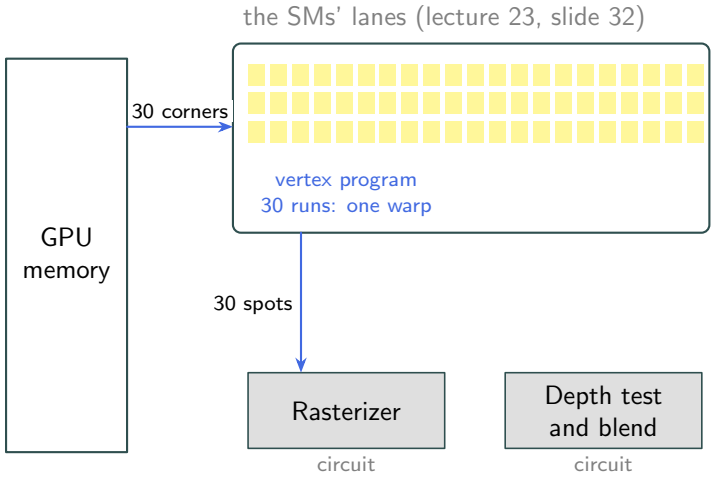
fragment program

one command

`draw(30)`

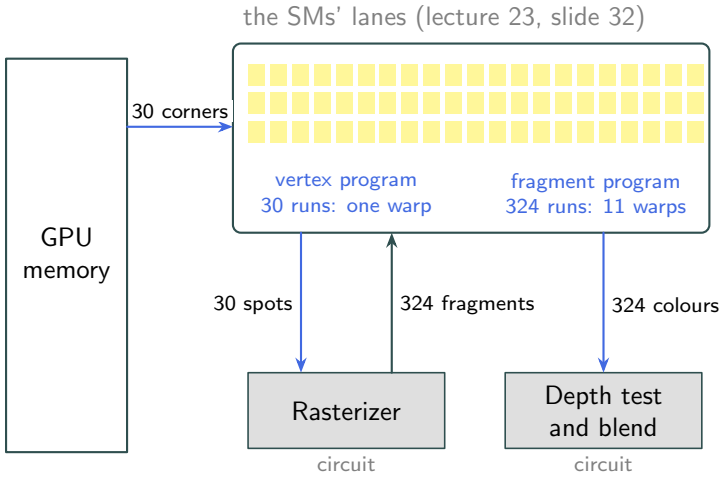
- You send corners, not pixels
- Plus two programs and one draw
- The GPU finds the 324 pixels

# Your two programs run on the lanes; the rest are circuits



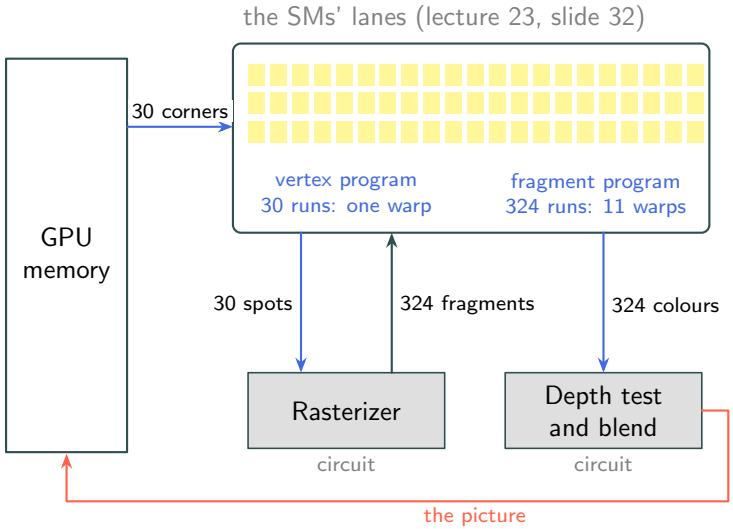

- Your programs: on the lanes

# Your two programs run on the lanes; the rest are circuits




- Your programs: on the lanes
- Fixed steps: circuits beside them

# Your two programs run on the lanes; the rest are circuits

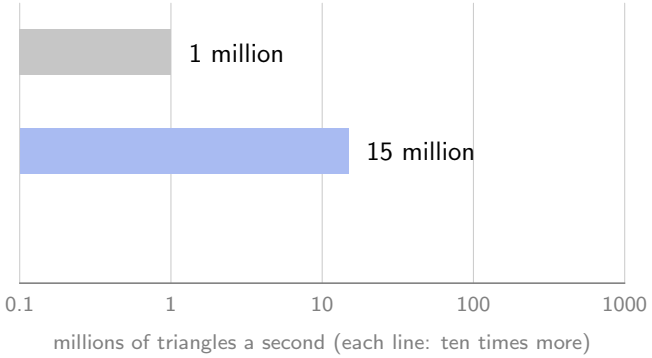



- Your programs: on the lanes
- Fixed steps: circuits beside them
- The same lanes run CUDA

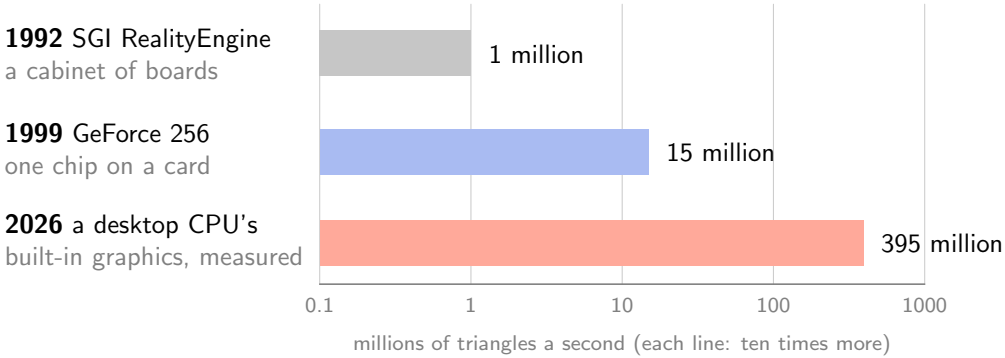
# Why we still draw triangles: hundreds of millions a second

**1992** SGI RealityEngine
a cabinet of boards

**1999** GeForce 256
one chip on a card

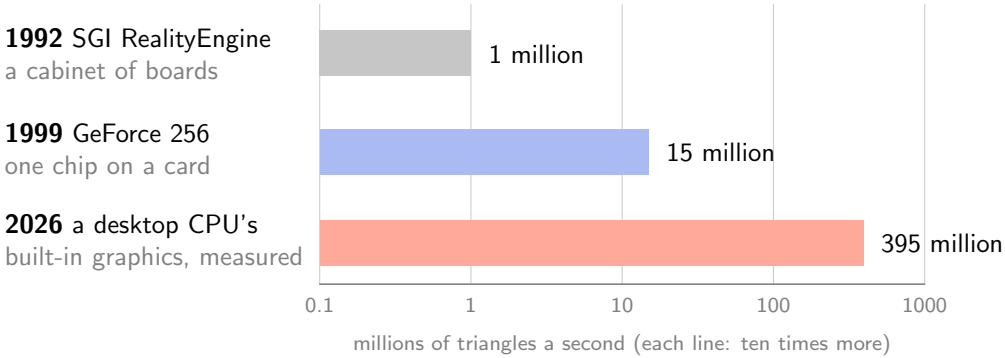


# Why we still draw triangles: hundreds of millions a second




- 6.6 million triangles in every frame at 60 Hz, on a small GPU

# Why we still draw triangles: hundreds of millions a second

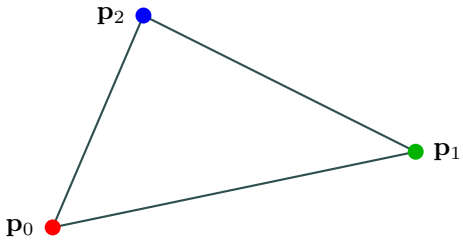



- 6.6 million triangles in every frame at 60 Hz, on a small GPU
- And the nearest surface stays exact: part four

# One triangle's trip

From three corners to coloured pixels

# Application: three corners in one buffer



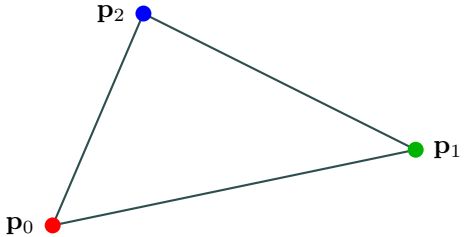

a vertex: a corner
and what it carries

- The program sends corners, not pixels

now: application
stage: application

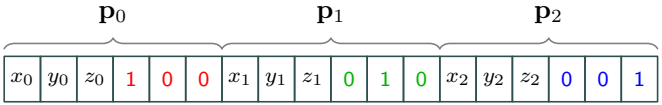


# Application: three corners in one buffer

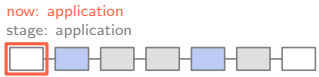



a vertex: a corner and what it carries

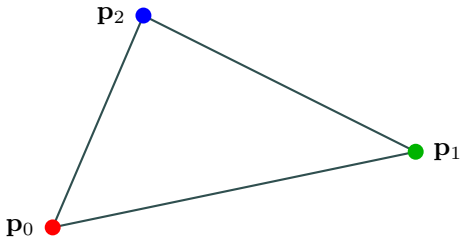
- The program sends corners, not pixels
- A buffer: numbers in one row



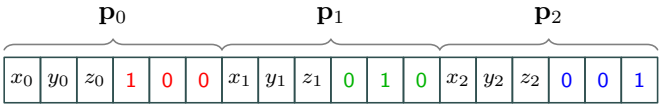

position in the scene  colour (red, green, blue)



# Application: three corners in one buffer



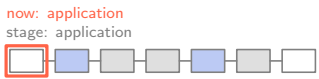

a vertex: a corner and what it carries



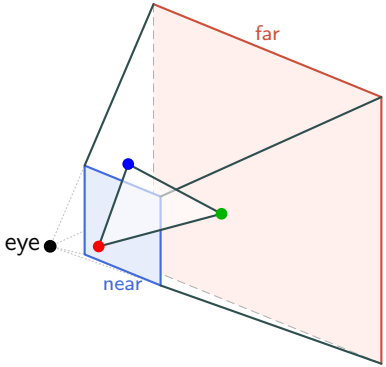

position in the scene  colour (red, green, blue)

copied once to the GPU's memory (lecture 23, slide 44)

- The program sends corners, not pixels
- A buffer: numbers in one row
- Sent once, drawn every frame



# The vertex program squeezes the frustum into a box



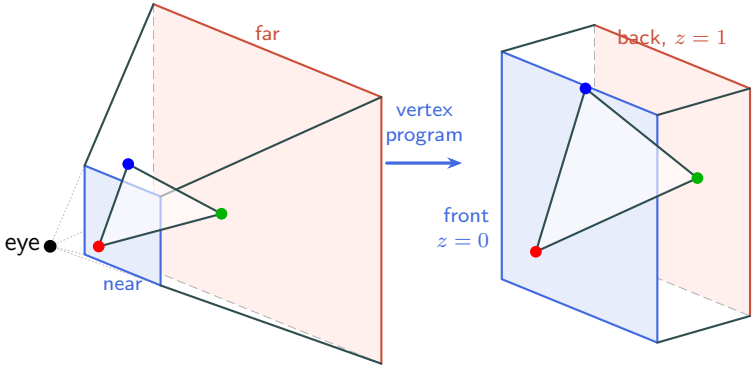

the frustum:
what the camera sees

- The camera sees a frustum: a pyramid cut at near and far

now: vertex processing
stage: geometry processing



# The vertex program squeezes the frustum into a box




the frustum:
what the camera sees

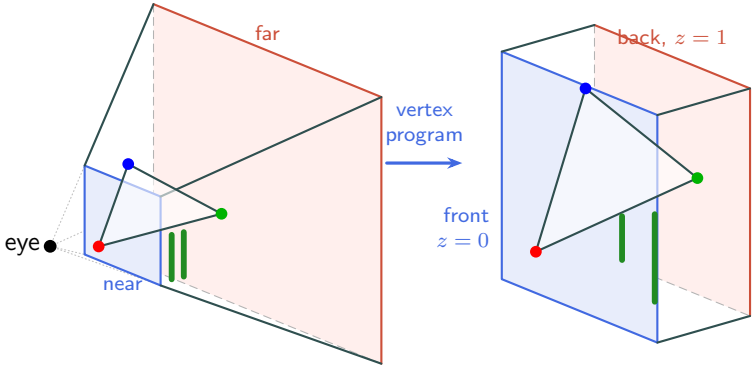
the box: $x, y$ from $-1$ to $1$,
$z$ from 0 to 1

- The camera sees a frustum: a pyramid cut at near and far
- The vertex program squeezes it into the box

now: vertex processing
stage: geometry processing



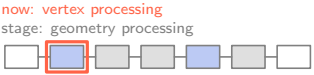
# The vertex program squeezes the frustum into a box



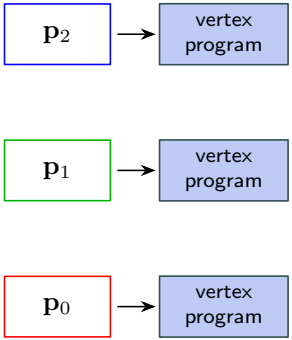

the frustum:
what the camera sees

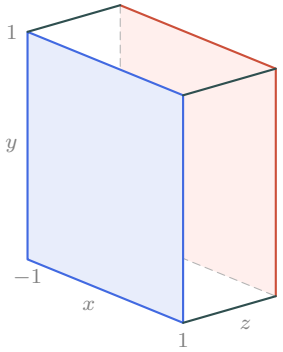
the box: $x, y$ from $-1$ to $1$,
$z$ from 0 to 1

- The camera sees a frustum: a pyramid cut at near and far
- The vertex program squeezes it into the box
- Far end squeezed most: perspective



# Vertex processing: one corner at a time, into the box



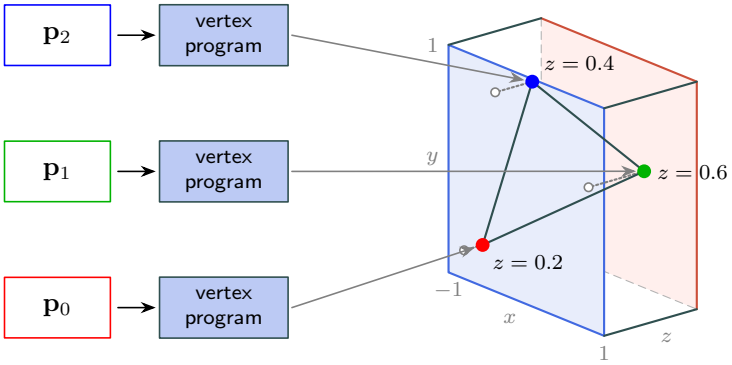

- One small program per corner

slide 11's box: $z = 0$ at the front, 1 at the back

now: vertex processing
stage: geometry processing



# Vertex processing: one corner at a time, into the box




slide 11's box: $z = 0$ at the front, 1 at the back

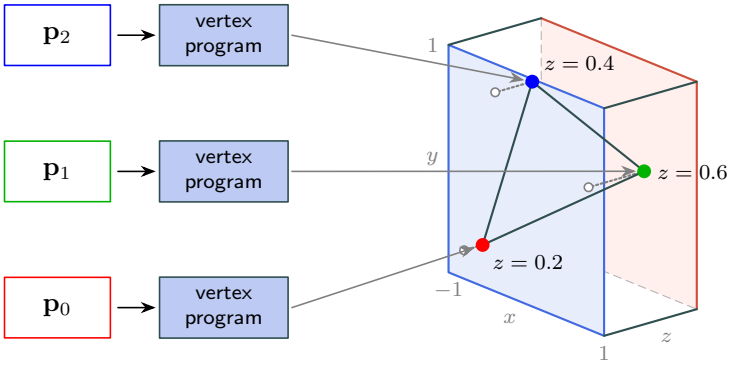
- One small program per corner
- $x, y$: where on the screen; $z$: the depth

now: vertex processing
stage: geometry processing



# Vertex processing: one corner at a time, into the box

out of the program: $(x, y, z, w)$
a circuit divides by $w$ to land here




slide 11's box: $z = 0$ at the front, 1 at the back

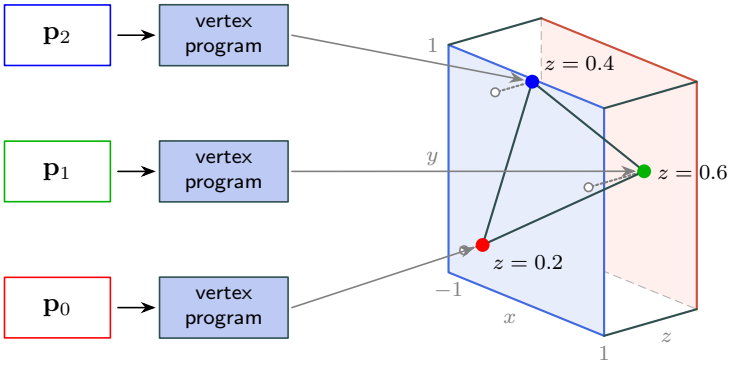
- One small program per corner
- $x, y$: where on the screen; $z$: the depth
- Really four numbers; $\div w$ gives the box

now: vertex processing
stage: geometry processing



# Vertex processing: one corner at a time, into the box

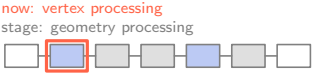
out of the program: $(x, y, z, w)$
a circuit divides by $w$ to land here



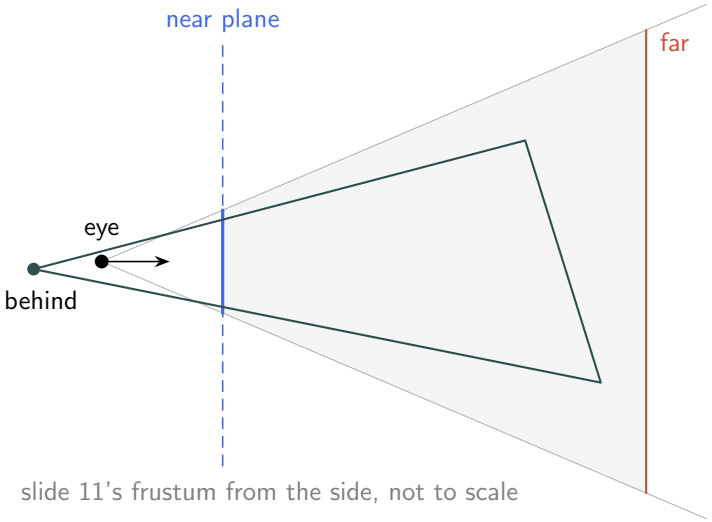

slide 11's box: $z = 0$ at the front, 1 at the back

- One small program per corner
- $x, y$: where on the screen; $z$: the depth
- Really four numbers; $\div w$ gives the box
- The matrices: lecture 28

now: vertex processing
stage: geometry processing



# Clipping: cut off what is behind the eye



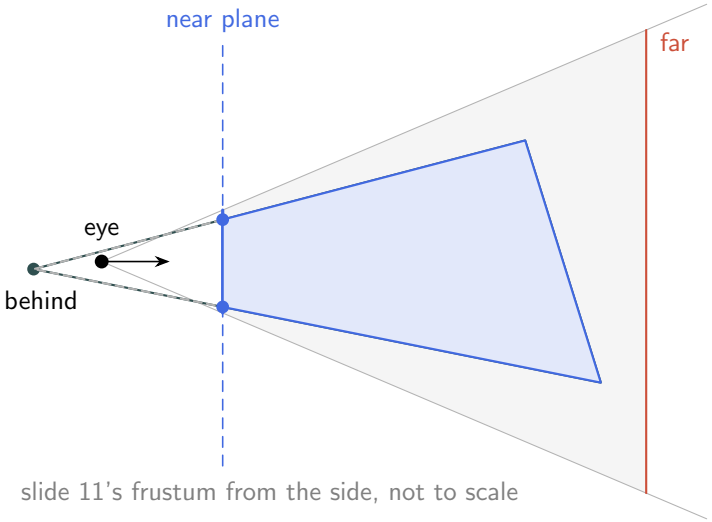

slide 11's frustum from the side, not to scale

- Behind the eye, the divide breaks

now: clipping and viewport
stage: geometry processing



# Clipping: cut off what is behind the eye



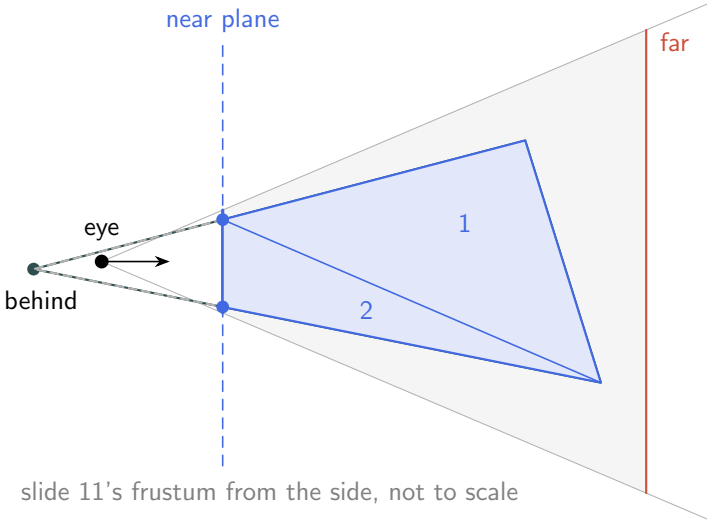

slide 11's frustum from the side, not to scale

- Behind the eye, the divide breaks
- Cut at the near plane first

now: clipping and viewport
stage: geometry processing



# Clipping: cut off what is behind the eye



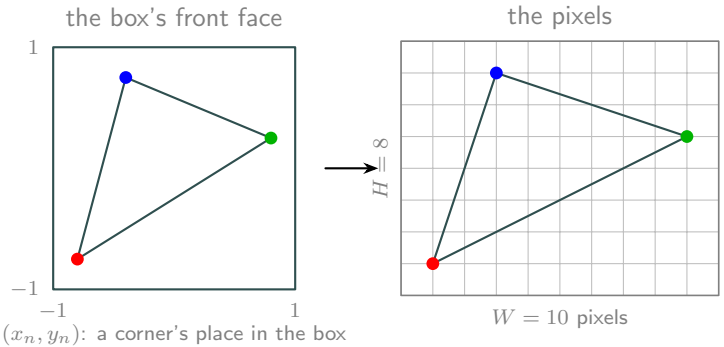

slide 11's frustum from the side, not to scale

- Behind the eye, the divide breaks
- Cut at the near plane first
- Four corners left: two triangles

now: clipping and viewport
stage: geometry processing



# Viewport: from the box to pixels

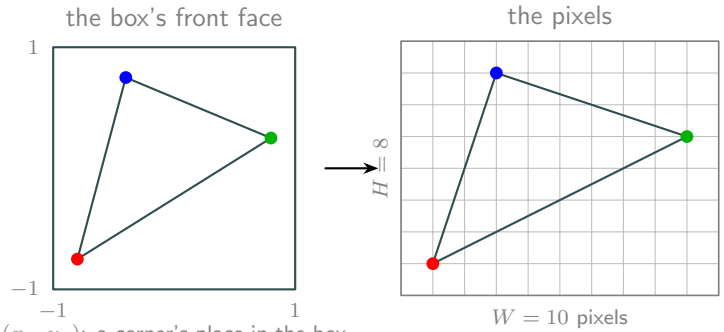



- Stretch the box over the pixels

now: clipping and viewport
stage: geometry processing



# Viewport: from the box to pixels



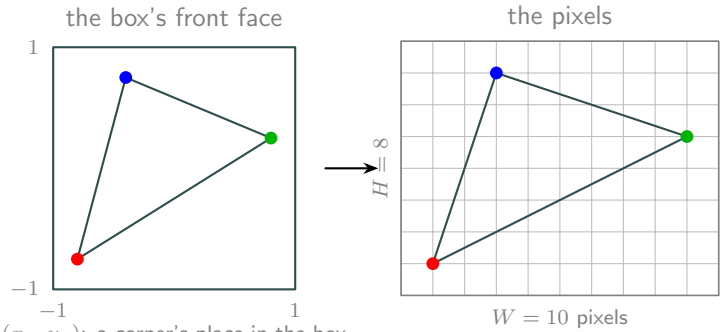

$$x = (x_n + 1)\,\underbrace{}_{\text{shift: 0 to 2}}\,\underbrace{\frac{W}{2}}_{\text{stretch: 0 to } W} \qquad y = (y_n + 1)\,\frac{H}{2}$$

- Stretch the box over the pixels
- Lecture 8, slide 28: move, scale

now: clipping and viewport
stage: geometry processing



# Viewport: from the box to pixels




$$x = (x_n + 1)\,\underbrace{\frac{W}{2}}$$

shift: 0 to 2 stretch: 0 to $W$

$$y = (y_n + 1)\,\frac{H}{2}$$

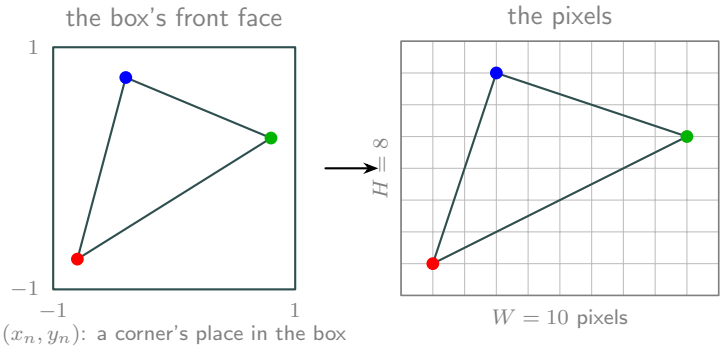
$\mathbf{p}_1 = (0.8,\ 0.25)$: $(0.8 + 1) \cdot 5 = 9$, $(0.25 + 1) \cdot 4 = 5$

- Stretch the box over the pixels
- Lecture 8, slide 28: move, scale
- Depth rides along unchanged

now: clipping and viewport
stage: geometry processing



# Viewport: from the box to pixels




$$x = (x_n + 1)\,\underbrace{}_{\text{shift: 0 to 2}}\,\underbrace{\frac{W}{2}}_{\text{stretch: 0 to } W} \qquad y = (y_n + 1)\,\frac{H}{2}$$

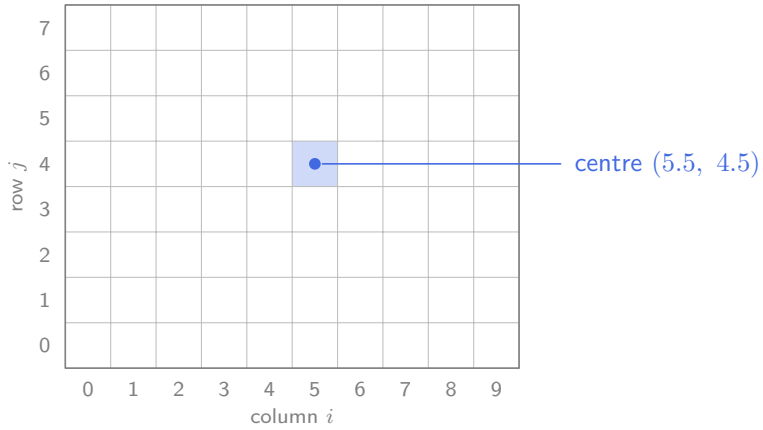
$\mathbf{p}_1 = (0.8,\ 0.25)$: $(0.8 + 1) \cdot 5 = 9,\ \ (0.25 + 1) \cdot 4 = 5$

- Stretch the box over the pixels
- Lecture 8, slide 28: move, scale
- Depth rides along unchanged
- Rows from the bottom here; WebGPU uses $1 - y_n$

now: clipping and viewport
stage: geometry processing

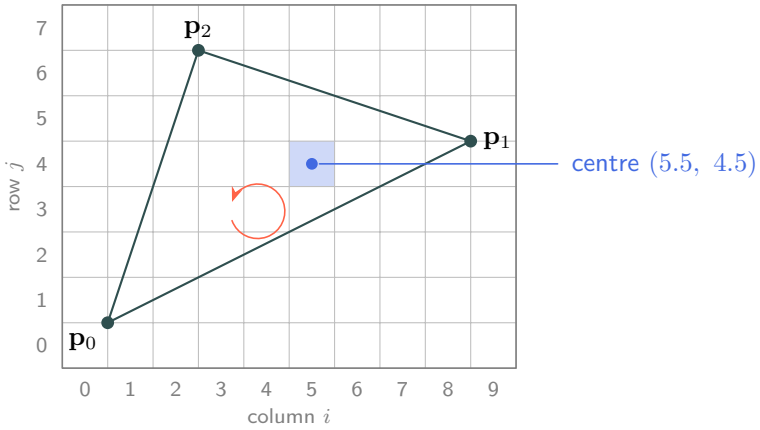


# On the pixels: centres at the halves, corners counter-clockwise



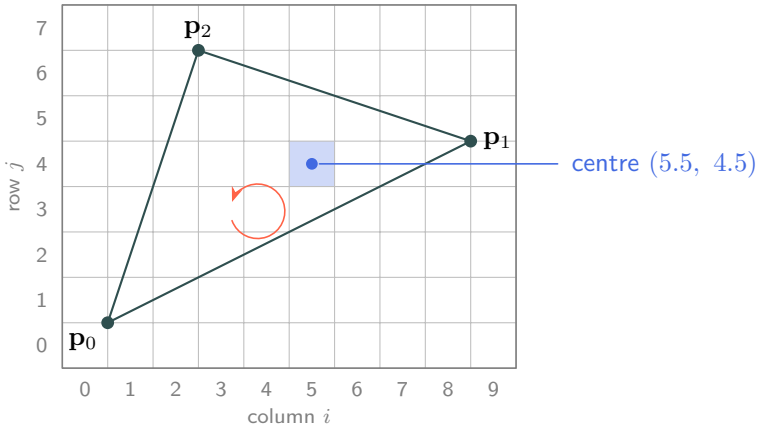

- Pixel $(i, j)$: centre $(i + 0.5,\ j + 0.5)$

# On the pixels: centres at the halves, corners counter-clockwise



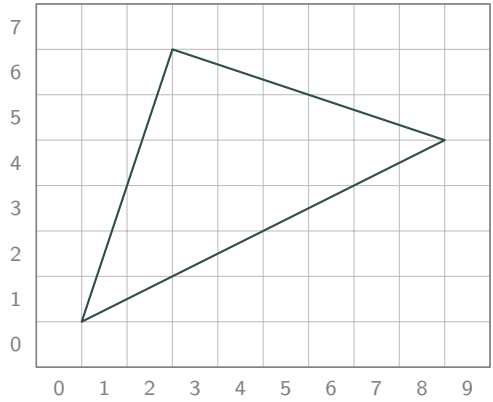

- Pixel $(i, j)$: centre $(i + 0.5,\ j + 0.5)$
- Corners go counter-clockwise

# On the pixels: centres at the halves, corners counter-clockwise

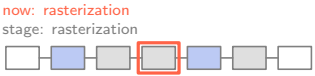



- Pixel $(i, j)$: centre $(i + 0.5,\ j + 0.5)$
- Corners go counter-clockwise
- Drawn with $y$ up, rows from the bottom

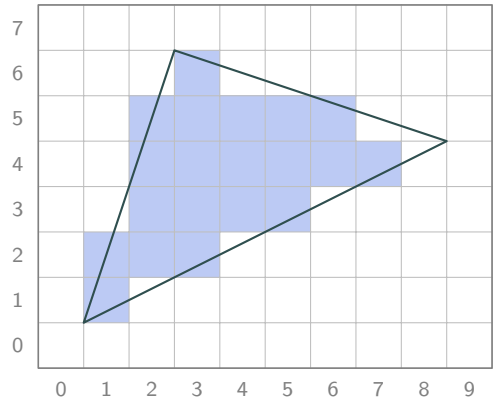
# Rasterization: which pixels does it cover?




- Covered: its centre is inside



# Rasterization: which pixels does it cover?



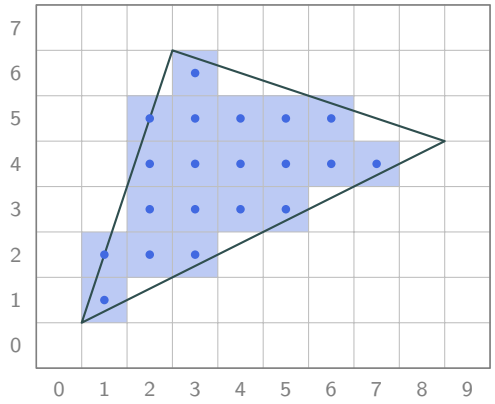

20 of 80 pixels

- Covered: its centre is inside
- Fragment: one pixel's share of it



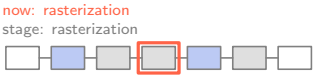


# Rasterization: which pixels does it cover?

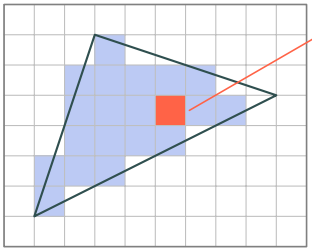



20 of 80 pixels

- Covered: its centre is inside
- Fragment: one pixel's share of it
- How it decides: part two



# Fragment processing: a colour for each fragment

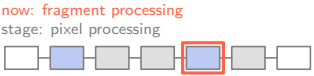


fragment at pixel $(5, 4)$

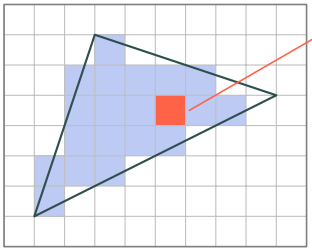
depth $0.45$

the corners' colours, blended

- Each fragment brings its data



# Fragment processing: a colour for each fragment



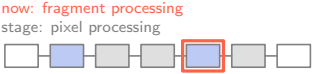
fragment at pixel $(5, 4)$

depth $0.45$

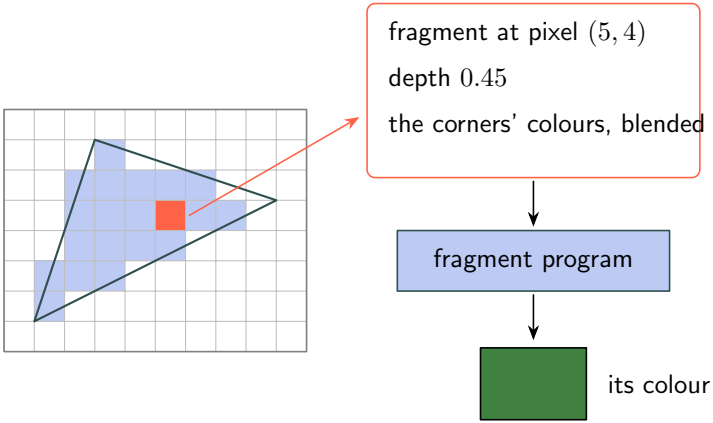
the corners' colours, blended

fragment program

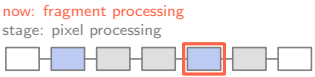
- Each fragment brings its data
- Your program runs once for each



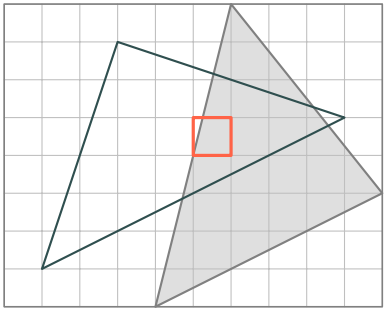
# Fragment processing: a colour for each fragment




- Each fragment brings its data
- Your program runs once for each
- Out: one colour (lectures 26, 30)



# Depth test and blend: the nearer one stays



two fragments, pixel $(5, 4)$

ours: depth $0.45$

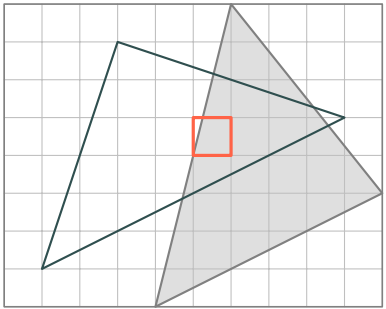
grey: depth $0.7$

- One pixel, many fragments

now: depth test and blend
stage: pixel processing



# Depth test and blend: the nearer one stays



two fragments, pixel $(5, 4)$

ours: depth $0.45$

grey: depth $0.7$

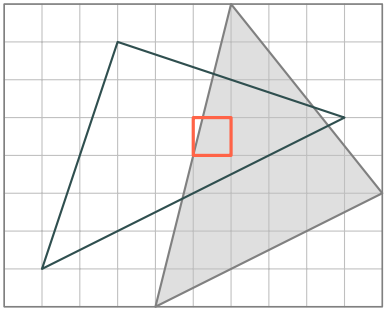
$0.45 < 0.7$: keep ours

- One pixel, many fragments
- The nearer one wins

now: depth test and blend
stage: pixel processing



# Depth test and blend: the nearer one stays



two fragments, pixel $(5, 4)$

ours: depth $0.45$

grey: depth $0.7$

$0.45 < 0.7$: keep ours

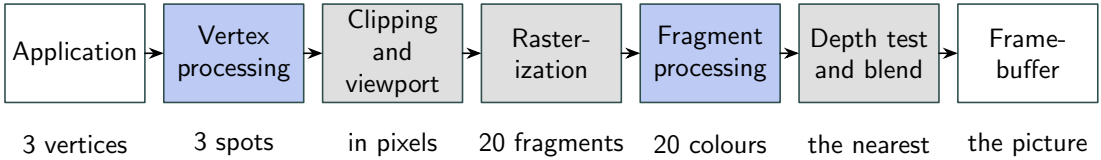
- One pixel, many fragments
- The nearer one wins
- Then written into the picture

blend: mix with what is there
(lecture 3, slide 8: alpha)

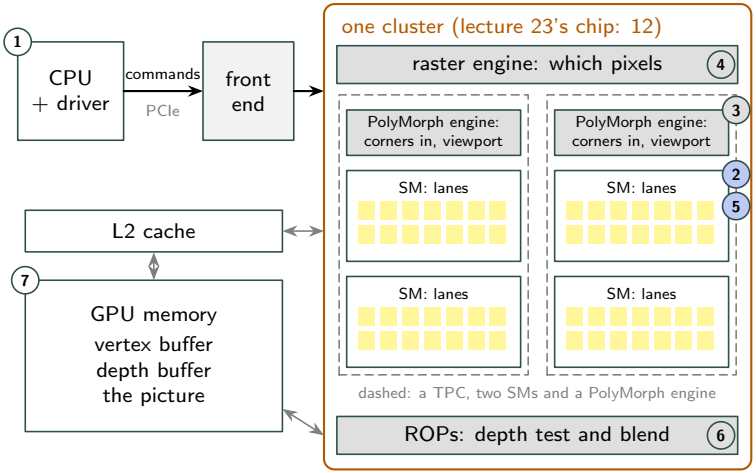
now: depth test and blend
stage: pixel processing



# Recap: corners in, coloured pixels out



# Where each stop runs on the chip

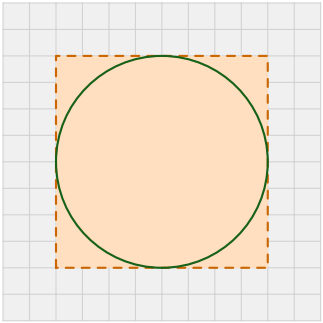



1. Application
   CPU and driver
2. Vertex program
   SM lanes
3. Clipping, viewport
   around the PolyMorph engine
4. Rasterization
   raster engine
5. Fragment program
   SM lanes, again
6. Depth test, blend
   ROPs
7. Framebuffer
   GPU memory

# Inside the rasterizer

Three signs for every pixel

# Lecture 3's disc: ask each pixel in the box

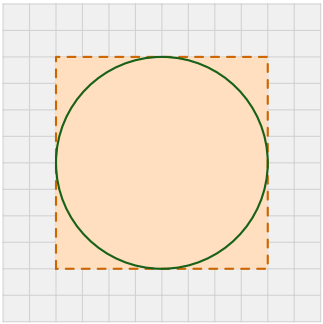


lecture 3, slides 27–29

$(x - c_x)^2 + (y - c_y)^2 \leq r^2$?

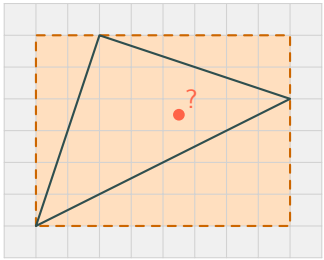
- Ask each pixel: am I inside?
- Only the pixels in the box

# Lecture 3's disc: ask each pixel in the box



lecture 3, slides 27–29

$(x - c_x)^2 + (y - c_y)^2 \leq r^2$?




our triangle, in its box

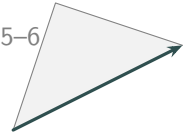
inside? a test we need

- Ask each pixel: am I inside?
- Only the pixels in the box
- Same plan, a new inside test

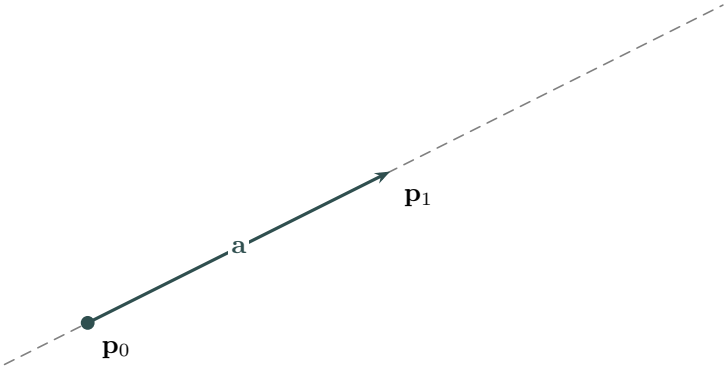
# Which side of one edge is a pixel on?

$f_{01}(\mathbf{p}) = |\mathbf{a}\,\mathbf{b}|, \;\; \mathbf{a} = \mathbf{p}_1 - \mathbf{p}_0, \;\; \mathbf{b} = \mathbf{p} - \mathbf{p}_0$

lecture 7, slides 5–6



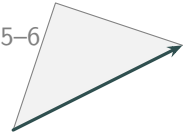
Slide 22's triangle: one edge first



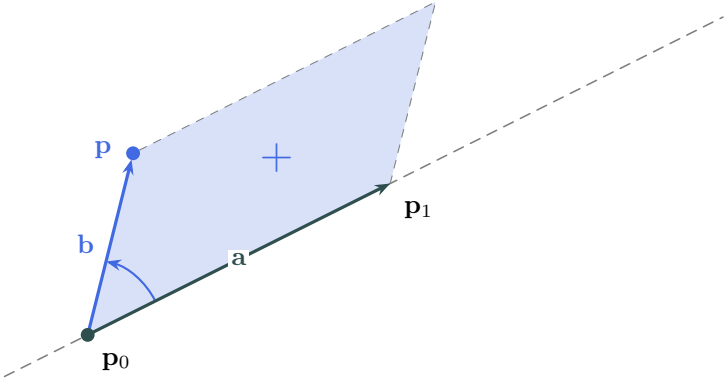
# Which side of one edge is a pixel on?

$f_{01}(\mathbf{p}) = |\mathbf{a}\,\mathbf{b}|, \;\; \mathbf{a} = \mathbf{p}_1 - \mathbf{p}_0, \;\; \mathbf{b} = \mathbf{p} - \mathbf{p}_0$

lecture 7, slides 5–6



Slide 22's triangle: one edge first

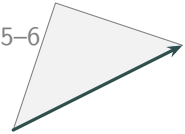



- Pixel on the left: $\mathbf{a}$ turns left to $\mathbf{b}$, $f_{01} > 0$

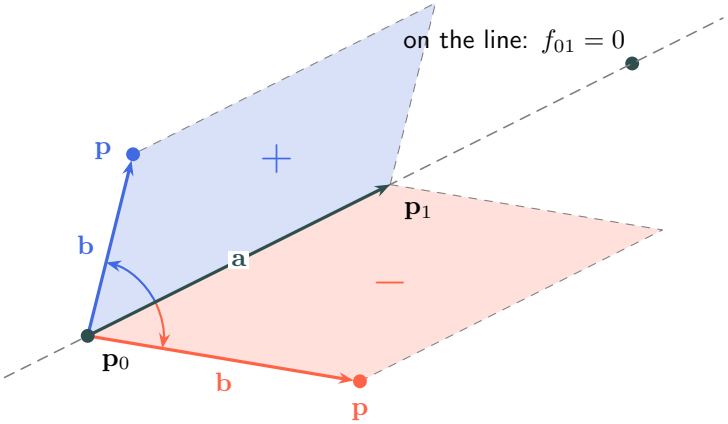
# Which side of one edge is a pixel on?

$f_{01}(\mathbf{p}) = |\mathbf{a}\,\mathbf{b}|, \;\; \mathbf{a} = \mathbf{p}_1 - \mathbf{p}_0, \;\; \mathbf{b} = \mathbf{p} - \mathbf{p}_0$

lecture 7, slides 5–6



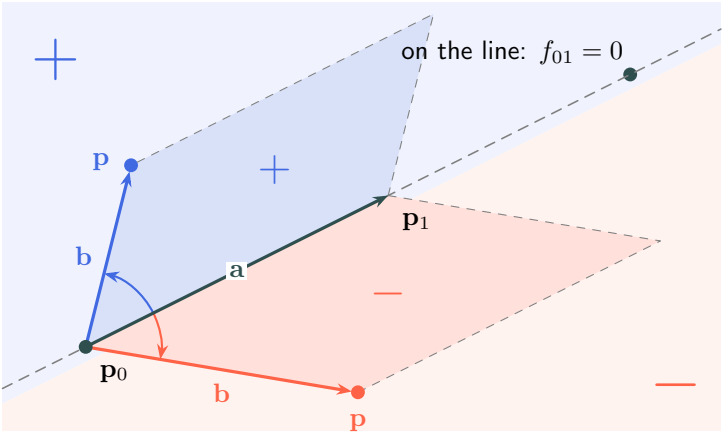
Slide 22's triangle: one edge first

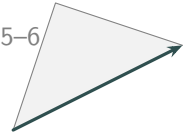



- Pixel on the left: $\mathbf{a}$ turns left to $\mathbf{b}$, $f_{01} > 0$
- Pixel on the right: $\mathbf{a}$ turns right, $f_{01} < 0$

# Which side of one edge is a pixel on?

$f_{01}(\mathbf{p}) = |\mathbf{a}\,\mathbf{b}|, \;\; \mathbf{a} = \mathbf{p}_1 - \mathbf{p}_0, \;\; \mathbf{b} = \mathbf{p} - \mathbf{p}_0$ lecture 7, slides 5–6



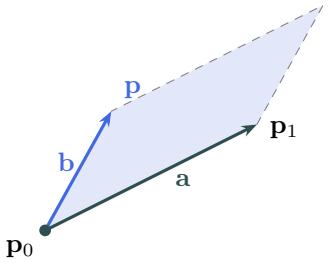
Slide 22's triangle: one edge first

- Pixel on the left: $\mathbf{a}$ turns left to $\mathbf{b}$, $f_{01} > 0$
- Pixel on the right: $\mathbf{a}$ turns right, $f_{01} < 0$
- The sign of $f_{01}$ tells the pixel's side

# The edge test, term by term

$$f_{01}(x, y) = |\mathbf{a}\,\mathbf{b}|, \quad \mathbf{a} = \mathbf{p}_1 - \mathbf{p}_0,\ \mathbf{b} = \mathbf{p} - \mathbf{p}_0$$

signed area, not a length



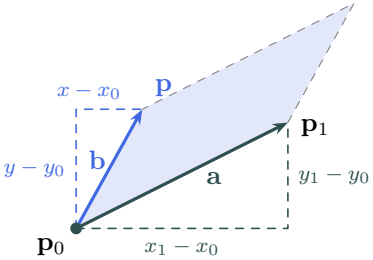
# The edge test, term by term

$$f_{01}(x, y) = |\mathbf{a}\,\mathbf{b}|, \quad \mathbf{a} = \mathbf{p}_1 - \mathbf{p}_0,\ \mathbf{b} = \mathbf{p} - \mathbf{p}_0$$

signed area, not a length

$$= (x_1 - x_0)(y - y_0) - (y_1 - y_0)(x - x_0)$$

lecture 7, slide 12

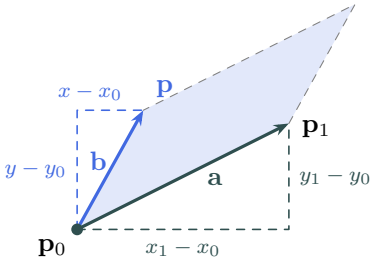


# The edge test, term by term

$$f_{01}(x,y) = |\mathbf{a}\,\mathbf{b}|, \quad \mathbf{a} = \mathbf{p}_1 - \mathbf{p}_0,\ \mathbf{b} = \mathbf{p} - \mathbf{p}_0$$ signed area, not a length

$$= (x_1 - x_0)(y - y_0) - (y_1 - y_0)(x - x_0)$$ lecture 7, slide 12

$$= (x_1 - x_0)\,y - (x_1 - x_0)\,y_0 - (y_1 - y_0)\,x + (y_1 - y_0)\,x_0$$ multiply out



# The edge test, term by term

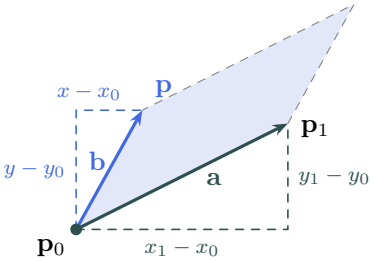
$$f_{01}(x,y) = |\mathbf{a}\,\mathbf{b}|, \quad \mathbf{a} = \mathbf{p}_1 - \mathbf{p}_0,\ \mathbf{b} = \mathbf{p} - \mathbf{p}_0$$ signed area, not a length

$$= (x_1 - x_0)(y - y_0) - (y_1 - y_0)(x - x_0)$$ lecture 7, slide 12

$$= (x_1 - x_0)\,y - (x_1 - x_0)\,y_0 - (y_1 - y_0)\,x + (y_1 - y_0)\,x_0$$ multiply out

$$= (y_0 - y_1)\,x + (x_1 - x_0)\,y + x_0 y_1 - x_1 y_0$$ $x_0 y_0$ cancels

the book's equation 9.1:
$f_{01}(x,y) = (y_0 - y_1)\,x + (x_1 - x_0)\,y + x_0 y_1 - x_1 y_0$



# The edge test, term by term

$$f_{01}(x,y) = |\mathbf{a}\,\mathbf{b}|, \quad \mathbf{a} = \mathbf{p}_1 - \mathbf{p}_0,\ \mathbf{b} = \mathbf{p} - \mathbf{p}_0$$ signed area, not a length

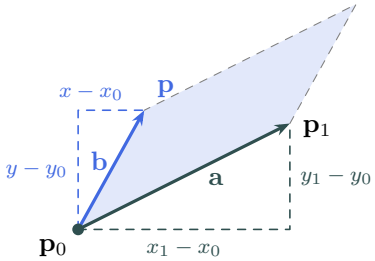
$$= (x_1 - x_0)(y - y_0) - (y_1 - y_0)(x - x_0)$$ lecture 7, slide 12

$$= (x_1 - x_0)\,y - (x_1 - x_0)\,y_0 - (y_1 - y_0)\,x + (y_1 - y_0)\,x_0$$ multiply out

$$= (y_0 - y_1)\,x + (x_1 - x_0)\,y + x_0 y_1 - x_1 y_0$$ $x_0 y_0$ cancels

the book's equation 9.1:
$f_{01}(x,y) = (y_0 - y_1)\,x + (x_1 - x_0)\,y + x_0 y_1 - x_1 y_0$

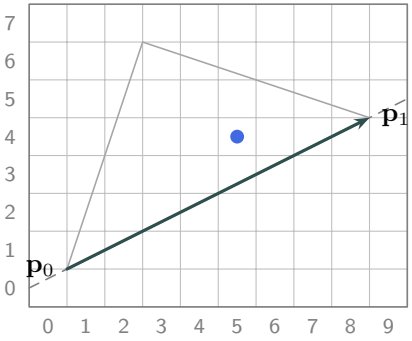
ours, $\mathbf{p}_0 = (1,1)$, $\mathbf{p}_1 = (9,5)$:
$f_{01} = (1-5)\,x + (9-1)\,y + 1 \cdot 5 - 9 \cdot 1 = -4x + 8y - 4$



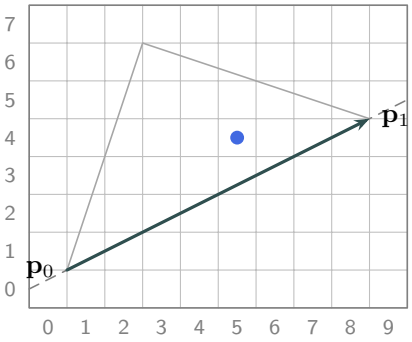
# Worked: $f_{01}$ is 10 at pixel (5, 4)

centre $(5.5,\ 4.5)$:

- Plug in the pixel's centre



# Worked: $f_{01}$ is 10 at pixel (5, 4)

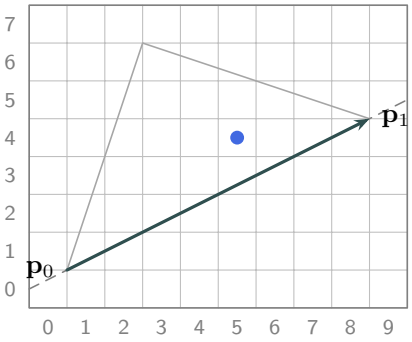



centre $(5.5,\ 4.5)$:

- Plug in the pixel's centre

$$f_{01} = -4(5.5) + 8(4.5) - 4$$

# Worked: $f_{01}$ is 10 at pixel (5, 4)



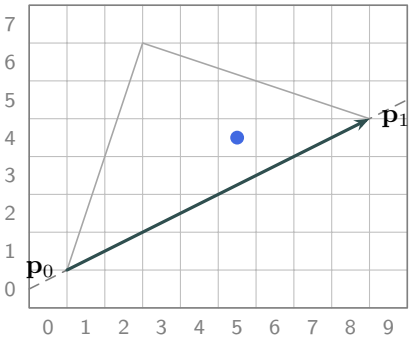

centre $(5.5,\ 4.5)$:

- Plug in the pixel's centre

$$f_{01} = -4(5.5) + 8(4.5) - 4$$

$$= -22 + 36 - 4$$

# Worked: $f_{01}$ is 10 at pixel (5, 4)




centre $(5.5,\ 4.5)$:

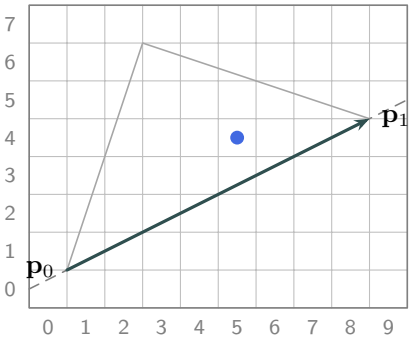
$$f_{01} = -4(5.5) + 8(4.5) - 4$$

$$= -22 + 36 - 4$$

$$= 10 > 0$$

- Plug in the pixel's centre
- Positive: left of $\mathbf{p}_0 \to \mathbf{p}_1$

# Worked: $f_{01}$ is 10 at pixel (5, 4)




centre $(5.5,\ 4.5)$:

$$f_{01} = -4(5.5) + 8(4.5) - 4$$

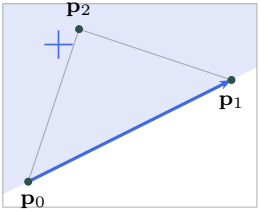
$$= -22 + 36 - 4$$

$$= 10 > 0$$

check: $8 \cdot 3.5 - 4 \cdot 4.5 = 28 - 18 = 10$

- Plug in the pixel's centre
- Positive: left of $\mathbf{p}_0 \to \mathbf{p}_1$

# Inside: all three edges agree

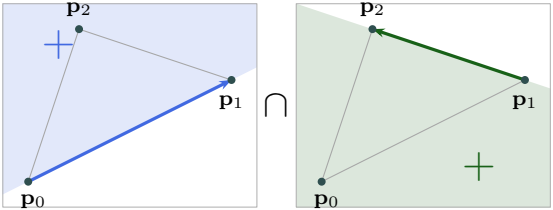



$f_{01} > 0$

$f_{01} = -4x + 8y - 4$

- Walk the edges $\mathbf{p}_0 \to \mathbf{p}_1 \to \mathbf{p}_2 \to \mathbf{p}_0$: the inside is always on the left

# Inside: all three edges agree




$f_{01} > 0$

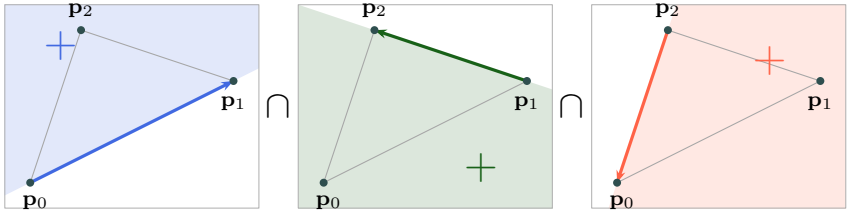
$f_{01} = -4x + 8y - 4$

$f_{12} > 0$

$f_{12} = -2x - 6y + 48$

- Walk the edges $\mathbf{p}_0 \to \mathbf{p}_1 \to \mathbf{p}_2 \to \mathbf{p}_0$: the inside is always on the left

# Inside: all three edges agree




$f_{01} > 0$

$f_{01} = -4x + 8y - 4$

$f_{12} > 0$

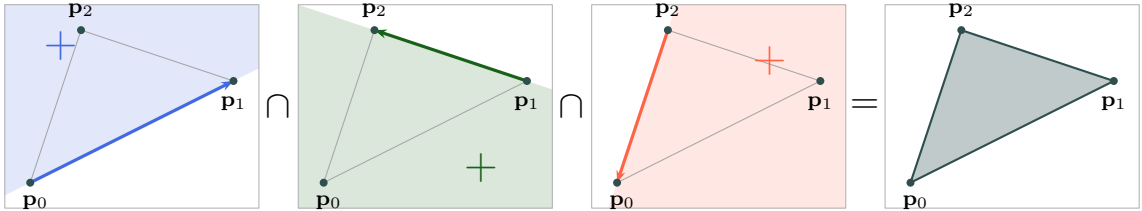
$f_{12} = -2x - 6y + 48$

$f_{20} > 0$

$f_{20} = 6x - 2y - 4$

- Walk the edges $\mathbf{p}_0 \to \mathbf{p}_1 \to \mathbf{p}_2 \to \mathbf{p}_0$: the inside is always on the left

# Inside: all three edges agree




$f_{01} > 0$

$f_{01} = -4x + 8y - 4$

$f_{12} > 0$

$f_{12} = -2x - 6y + 48$

$f_{20} > 0$

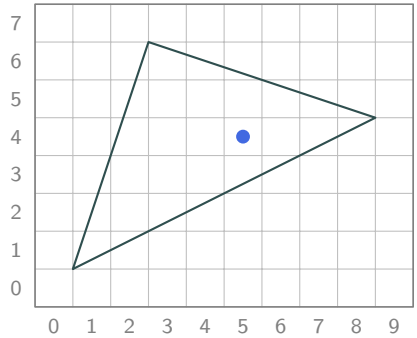
$f_{20} = 6x - 2y - 4$

inside

all three $> 0$

- Walk the edges $\mathbf{p}_0 \to \mathbf{p}_1 \to \mathbf{p}_2 \to \mathbf{p}_0$: the inside is always on the left
- A pixel is inside when all three edge functions are positive

# Worked: pixel (5, 4) is in, (7, 2) is out

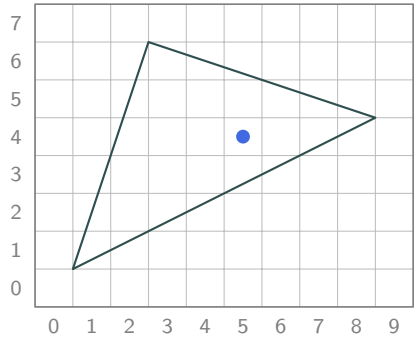



$(5.5,\ 4.5)$:

$$f_{01} = -22 + 36 - 4 = 10$$
$$f_{12} = -11 - 27 + 48 = 10$$
$$f_{20} = 33 - 9 - 4 = 20$$

- Three numbers per pixel

# Worked: pixel (5, 4) is in, (7, 2) is out




$(5.5,\ 4.5)$:

$f_{01} = -22 + 36 - 4 = 10$

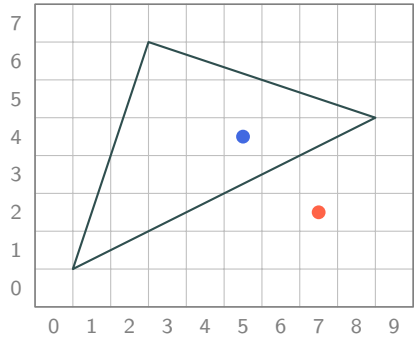
$f_{12} = -11 - 27 + 48 = 10$

$f_{20} = 33 - 9 - 4 = 20$

all positive: in

- Three numbers per pixel

# Worked: pixel (5, 4) is in, (7, 2) is out




$(5.5,\ 4.5)$:

$f_{01} = -22 + 36 - 4 = 10$

$f_{12} = -11 - 27 + 48 = 10$

$f_{20} = 33 - 9 - 4 = 20$

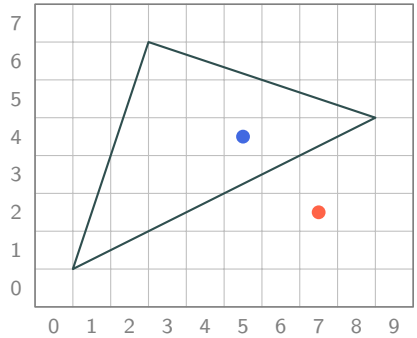
all positive: in

$(7.5,\ 2.5)$:

$f_{01} = -30 + 20 - 4 = -14$

- Three numbers per pixel

# Worked: pixel (5, 4) is in, (7, 2) is out




$(5.5,\ 4.5)$:

$f_{01} = -22 + 36 - 4 = 10$

$f_{12} = -11 - 27 + 48 = 10$

$f_{20} = 33 - 9 - 4 = 20$

all positive: in

$(7.5,\ 2.5)$:

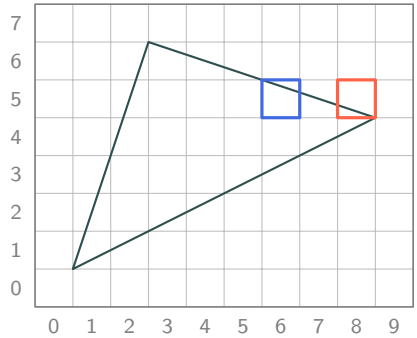
$f_{01} = -30 + 20 - 4 = -14$

one negative: out

($f_{12} = 18$, $f_{20} = 36$: no help)

- Three numbers per pixel
- One negative is enough: out

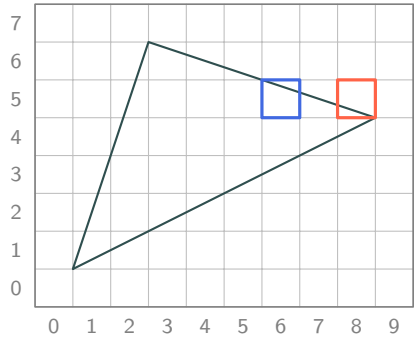
# Quick check 1: covered or not?




$$f_{01} = -4x + 8y - 4$$
$$f_{12} = -2x - 6y + 48$$
$$f_{20} = 6x - 2y - 4$$

(a) pixel $(6, 5)$? (b) pixel $(8, 5)$?

# Quick check 1: covered or not?




$$f_{01} = -4x + 8y - 4$$
$$f_{12} = -2x - 6y + 48$$
$$f_{20} = 6x - 2y - 4$$

(a) pixel $(6, 5)$? (b) pixel $(8, 5)$?

(a) $(6.5, 5.5)$: $14,\ 2,\ 24$: in

(b) $(8.5, 5.5)$: $6,\ -2,\ 36$: out

$f_{12}(8.5, 5.5) = -17 - 33 + 48 = -2$

- Close to an edge: trust the sign

# edge(): the determinant, in Python

`edge(start, end, p)` returns

```
(end.x - start.x) * (p.y - start.y) - (end.y - start.y) * (p.x - start.x)
```

# edge(): the determinant, in Python

`edge(start, end, p)` returns

```
(end.x - start.x) * (p.y - start.y) - (end.y - start.y) * (p.x - start.x)
```

$$x_a \times y_b - y_a \times x_b$$

$\mathbf{a} = \texttt{end} - \texttt{start} = (x_a,\ y_a)$ $\mathbf{b} = \texttt{p} - \texttt{start} = (x_b,\ y_b)$

# edge(): the determinant, in Python

`edge(start, end, p)` returns

```
(end.x - start.x) * (p.y - start.y) - (end.y - start.y) * (p.x - start.x)
```

$$x_a \times y_b - y_a \times x_b$$

$\mathbf{a} = \texttt{end} - \texttt{start} = (x_a,\ y_a)$ $\qquad$ $\mathbf{b} = \texttt{p} - \texttt{start} = (x_b,\ y_b)$

$$x_a\, y_b - y_a\, x_b \;=\; \begin{vmatrix} x_a & x_b \\ y_a & y_b \end{vmatrix} \;=\; |\mathbf{a}\,\mathbf{b}|,$$ the determinant of $\mathbf{a}$ and $\mathbf{b}$: slide 23's signed area

# edge(): the determinant, in Python

`edge(start, end, p)` returns

```
(end.x - start.x) * (p.y - start.y) - (end.y - start.y) * (p.x - start.x)
```

$$x_a \times y_b - y_a \times x_b$$

$$8 \times 3.5 - 4 \times 4.5$$

$$= 28 - 18 = 10$$

$\mathbf{a} = \texttt{end} - \texttt{start} = (x_a,\ y_a) = (8, 4)$ $\qquad$ $\mathbf{b} = \texttt{p} - \texttt{start} = (x_b,\ y_b) = (4.5, 3.5)$

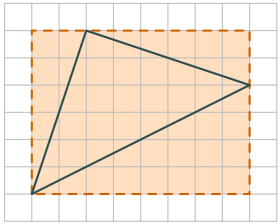
$x_a\, y_b - y_a\, x_b \ = \ \begin{vmatrix} x_a & x_b \\ y_a & y_b \end{vmatrix} \ = \ |\mathbf{a}\,\mathbf{b}|$, the determinant of $\mathbf{a}$ and $\mathbf{b}$: slide 23's signed area

# The whole loop, over the box

```
def edge(start, end, p):
    return ((end.x - start.x) * (p.y - start.y)
            - (end.y - start.y) * (p.x - start.x))
```

# The whole loop, over the box

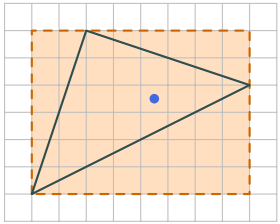
```
def edge(start, end, p):
    return ((end.x - start.x) * (p.y - start.y)
            - (end.y - start.y) * (p.x - start.x))

def rasterize(p0, p1, p2):
    xs = [p0.x, p1.x, p2.x]
    ys = [p0.y, p1.y, p2.y]
    for j in range(floor(min(ys)), ceil(max(ys))):
        for i in range(floor(min(xs)), ceil(max(xs))):
```



$i$: 1 to 8, $j$: 1 to 6: 48 tests

# The whole loop, over the box

```
def edge(start, end, p):
    return ((end.x - start.x) * (p.y - start.y)
            - (end.y - start.y) * (p.x - start.x))

def rasterize(p0, p1, p2):
    xs = [p0.x, p1.x, p2.x]
    ys = [p0.y, p1.y, p2.y]
    for j in range(floor(min(ys)), ceil(max(ys))):
        for i in range(floor(min(xs)), ceil(max(xs))):
            p = Vec2(i + 0.5, j + 0.5)
            f12 = edge(p1, p2, p)
            f20 = edge(p2, p0, p)
            f01 = edge(p0, p1, p)
```

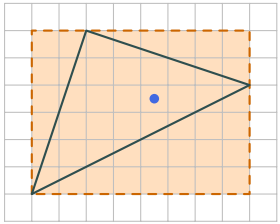


$i$: 1 to 8, $j$: 1 to 6: 48 tests

- Three values per pixel

# The whole loop, over the box

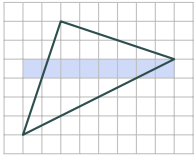
```
def edge(start, end, p):
    return ((end.x - start.x) * (p.y - start.y)
            - (end.y - start.y) * (p.x - start.x))

def rasterize(p0, p1, p2):
    xs = [p0.x, p1.x, p2.x]
    ys = [p0.y, p1.y, p2.y]
    for j in range(floor(min(ys)), ceil(max(ys))):
        for i in range(floor(min(xs)), ceil(max(xs))):
            p = Vec2(i + 0.5, j + 0.5)
            f12 = edge(p1, p2, p)
            f20 = edge(p2, p0, p)
            f01 = edge(p0, p1, p)
            if f12 >= 0 and f20 >= 0 and f01 >= 0:
                fragment(i, j, f12, f20, f01)
```



$i$: 1 to 8, $j$: 1 to 6: 48 tests

- Three values per pixel
- Why $>=$? Slide 35

# One step right adds the same number



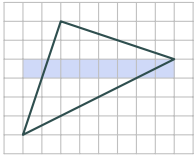
$f_{01}$ at the centres of row 4

row 4

| $i=1$ | $i=2$ | $i=3$ | $i=4$ | $i=5$ | $i=6$ | $i=7$ | $i=8$ |
|---|---|---|---|---|---|---|---|
| 26 | 22 | 18 | 14 | 10 | 6 | 2 | $-2$ |

- Neighbours' values differ by a constant

# One step right adds the same number



$f_{01}$ at the centres of row 4

row 4

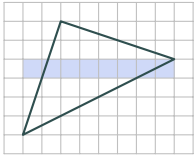
| $i=1$ | $i=2$ | $i=3$ | $i=4$ | $i=5$ | $i=6$ | $i=7$ | $i=8$ |
|---|---|---|---|---|---|---|---|
| 26 | 22 | 18 | 14 | 10 | 6 | 2 | $-2$ |

$-4$ $-4$ $-4$ $-4$ $-4$ $-4$ $-4$

$f_{01}(x+1,\ y) = f_{01}(x,y) + (y_0 - y_1)$, ours: $1 - 5 = -4$

- Neighbours' values differ by a constant
- Step right: add $y_0 - y_1$

# One step right adds the same number



$f_{01}$ at the centres of row 4

row 4

| $i=1$ | $i=2$ | $i=3$ | $i=4$ | $i=5$ | $i=6$ | $i=7$ | $i=8$ |
|---|---|---|---|---|---|---|---|
| 26 | 22 | 18 | 14 | 10 | 6 | 2 | $-2$ |

$-4$ $-4$ $-4$ $-4$ $-4$ $-4$ $-4$

$$f_{01}(x+1,\ y) = f_{01}(x,y) + (y_0 - y_1), \quad \text{ours: } 1-5=-4$$

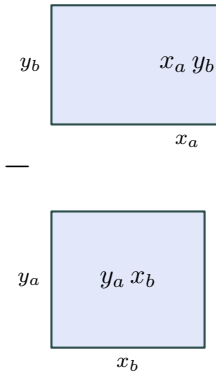
$$f_{01}(x,\ y+1) = f_{01}(x,y) + (x_1 - x_0), \quad \text{ours: } 9-1=8$$

- Neighbours' values differ by a constant
- Step right: add $y_0 - y_1$
- Step up: add $x_1 - x_0$

# Why a step right always adds the same number

slide 29: $f_{01} = x_a\, y_b - y_a\, x_b$, with $\mathbf{b} = \mathbf{p} - \texttt{start} = (x - x_0,\ y - y_0)$

$$f_{01}(x, y) = x_a\,(y - y_0) - y_a\,(x - x_0)$$

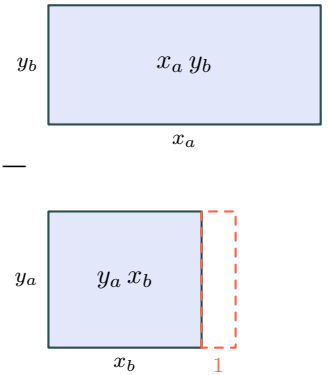


# Why a step right always adds the same number

slide 29: $f_{01} = x_a\, y_b - y_a\, x_b$, with $\mathbf{b} = \mathbf{p} - \texttt{start} = (x - x_0,\ y - y_0)$

$$f_{01}(x, y) = x_a\,(y - y_0) - y_a\,(x - x_0)$$

$$f_{01}(x + 1, y) = x_a\,(y - y_0) - y_a\,(x + 1 - x_0)$$



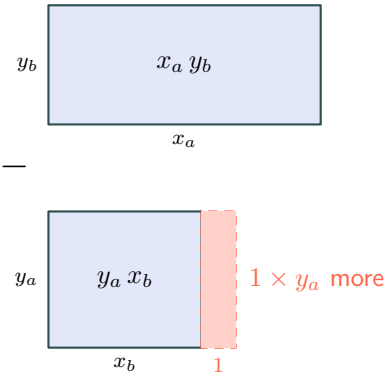
# Why a step right always adds the same number

slide 29: $f_{01} = x_a\, y_b - y_a\, x_b$, with $\mathbf{b} = \mathbf{p} - \texttt{start} = (x - x_0,\ y - y_0)$

$$f_{01}(x, y) = x_a\,(y - y_0) - y_a\,(x - x_0)$$

$$f_{01}(x + 1, y) = x_a\,(y - y_0) - y_a\,(x + 1 - x_0)$$

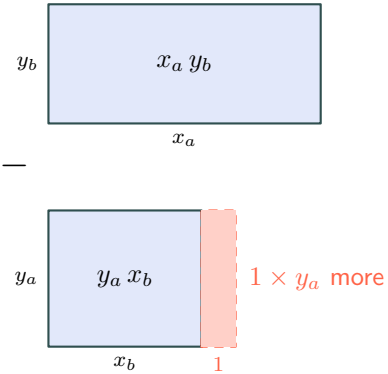
$$= \underbrace{x_a\,(y - y_0) - y_a\,(x - x_0)}_{f_{01}(x,y)} - y_a$$



# Why a step right always adds the same number

slide 29: $f_{01} = x_a\, y_b - y_a\, x_b$, with $\mathbf{b} = \mathbf{p} - \texttt{start} = (x - x_0,\ y - y_0)$

$$
\begin{aligned}
f_{01}(x, y) &= x_a\,(y - y_0) - y_a\,(x - x_0) \\
f_{01}(x+1, y) &= x_a\,(y - y_0) - y_a\,(x + 1 - x_0) \\
&= \underbrace{x_a\,(y - y_0) - y_a\,(x - x_0)}_{f_{01}(x,y)} - y_a \\
&= f_{01}(x, y) - y_a
\end{aligned}
$$




Step right: add $-y_a = y_0 - y_1$; ours $1 - 5 = -4$

# Why a step right always adds the same number

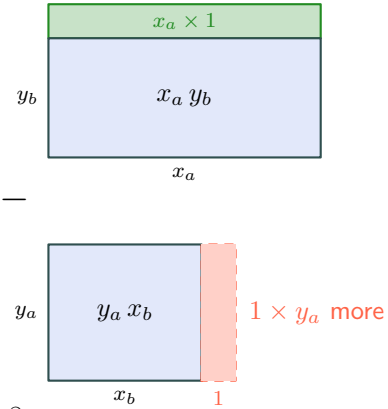
slide 29: $f_{01} = x_a\, y_b - y_a\, x_b$, with $\mathbf{b} = \mathbf{p} - \texttt{start} = (x - x_0,\ y - y_0)$

$$f_{01}(x, y) = x_a\,(y - y_0) - y_a\,(x - x_0)$$

$$f_{01}(x+1, y) = x_a\,(y - y_0) - y_a\,(x + 1 - x_0)$$

$$= \underbrace{x_a\,(y - y_0) - y_a\,(x - x_0)}_{f_{01}(x,y)} - y_a$$

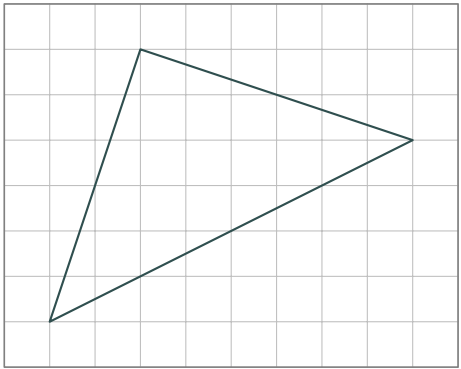
$$= f_{01}(x, y) - y_a$$




Step right: add $-y_a = y_0 - y_1$; ours $1 - 5 = -4$

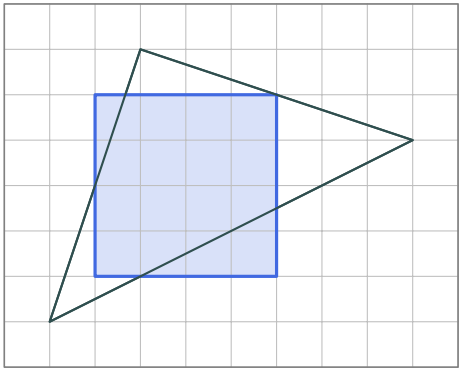
Step up, the same way: add $+x_a = x_1 - x_0$; ours $9 - 1 = 8$

# The hardware tests a block at once



- Every pixel's test stands alone

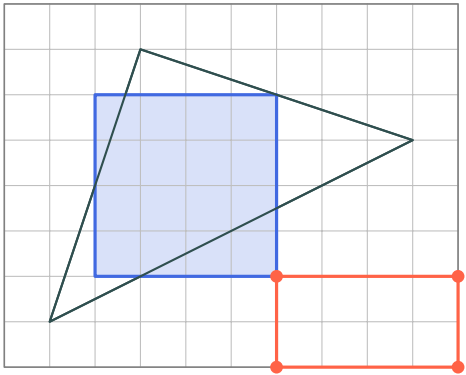
# The hardware tests a block at once



16 pixels,
16 lanes, one step

- Every pixel's test stands alone
- A whole block per step

# The hardware tests a block at once

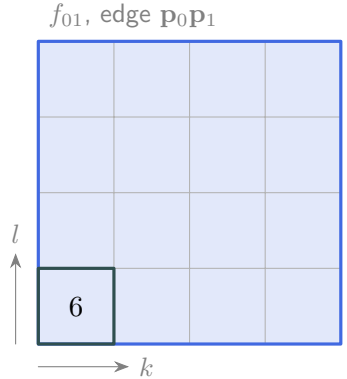


16 pixels,
16 lanes, one step

4 corners below
$f_{01}$'s line: skip

- Every pixel's test stands alone
- A whole block per step
- Block outside one edge: skip it

# Inside a block: one full test per edge, then steps

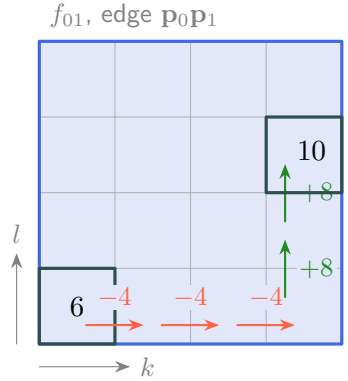



$f_{\text{first}} = f_{01}(2.5,\ 2.5) = -10 + 20 - 4 = 6$

- Full $f_{01}$: once per block

# Inside a block: one full test per edge, then steps

$$f_{01} = f_{\text{first}} + k\,(-4) + l\,(+8)$$

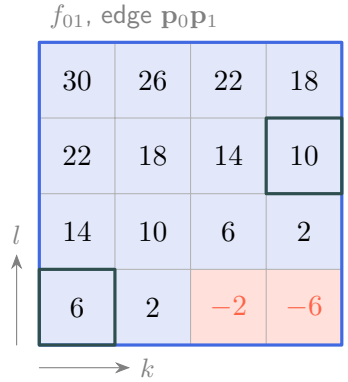



$$6 - 12 + 16 = 10$$

- Full $f_{01}$: once per block
- Slide 32's steps

# Inside a block: one full test per edge, then steps

$$f_{01} = f_{\text{first}} + k\,(-4) + l\,(+8)$$




- Full $f_{01}$: once per block
- Slide 32's steps
- 16 lanes, one add each

# Inside a block: one full test per edge, then steps

$$f_{01} = f_{\text{first}} + k\,(-4) + l\,(+8)$$

$f_{01}$, edge $\mathbf{p}_0\mathbf{p}_1$

| | | | |
|---|---|---|---|
| 30 | 26 | 22 | 18 |
| 22 | 18 | 14 | 10 |
| 14 | 10 | 6 | 2 |
| 6 | 2 | −2 | −6 |

$l$ ↑ , $k$ →

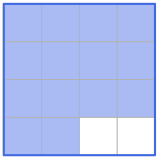
$f_{12}$

| | | | |
|---|---|---|---|
| 10 | 8 | 6 | 4 |
| 16 | 14 | 12 | 10 |
| 22 | 20 | 18 | 16 |
| 28 | 26 | 24 | 22 |

$f_{20}$

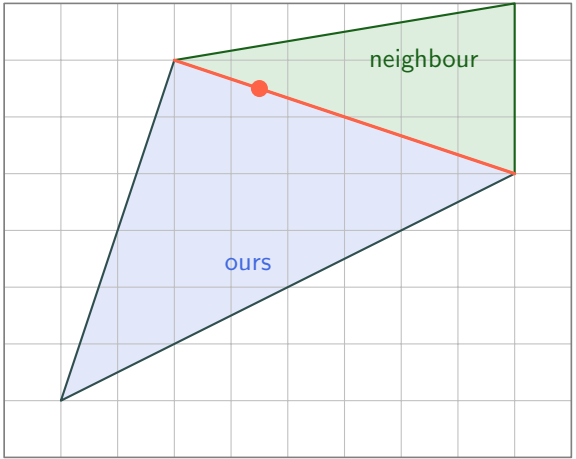
| | | | |
|---|---|---|---|
| 0 | 6 | 12 | 18 |
| 2 | 8 | 14 | 20 |
| 4 | 10 | 16 | 22 |
| 6 | 12 | 18 | 24 |

in: all three $\geq 0$



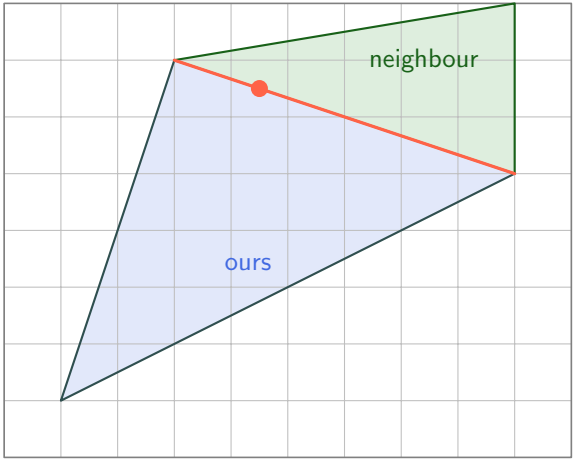
- Full $f_{01}$: once per block
- Slide 32's steps
- 16 lanes, one add each
- Same for $f_{12}$, $f_{20}$

# A pixel on a shared edge: who draws it?



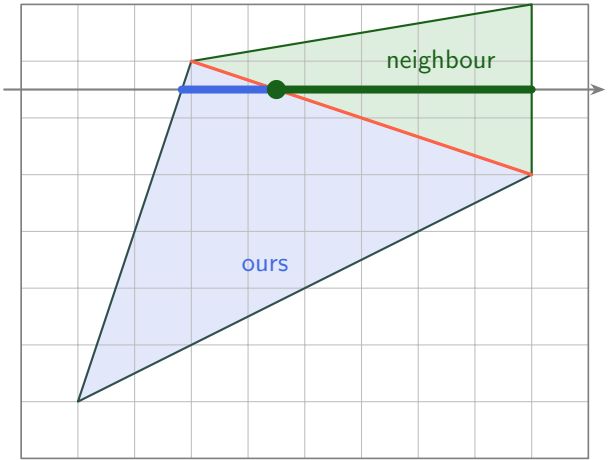

- Centre on the red edge: $f = 0$ for both triangles

# A pixel on a shared edge: who draws it?




- Centre on the red edge: $f = 0$ for both triangles
- Both draw it: twice
- Neither draws it: a hole

# A pixel on a shared edge: who draws it?




- Centre on the red edge: $f = 0$ for both triangles
- Both draw it: twice
- Neither draws it: a hole
- Rule: walk the row left to right; the triangle it *enters* at the pixel draws it

# Recap: three signs decide each pixel

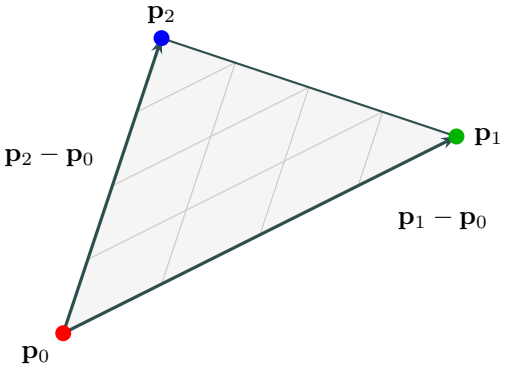
$$f_{01}(x, y) = (y_0 - y_1)\, x + (x_1 - x_0)\, y + x_0 y_1 - x_1 y_0$$

covered: $f_{12} \geq 0$, $f_{20} \geq 0$, $f_{01} \geq 0$ at the centre ($= 0$: top-left rule)

# Barycentric coordinates

The three numbers we already have

# Barycentric coordinates: an address from the corners

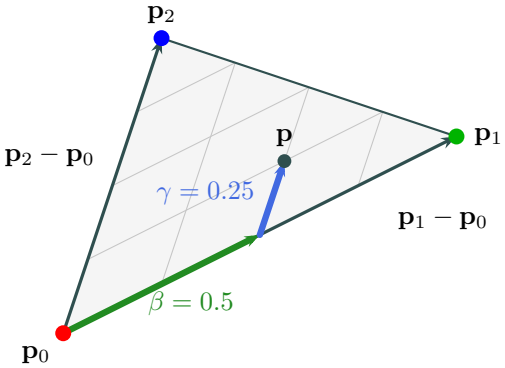



$$\mathbf{p} = \mathbf{p}_0 + \beta\,(\mathbf{p}_1 - \mathbf{p}_0) + \gamma\,(\mathbf{p}_2 - \mathbf{p}_0)$$

$\beta$ steps along one side, $\gamma$ along the other

the book, §2.9.1

# Barycentric coordinates: an address from the corners

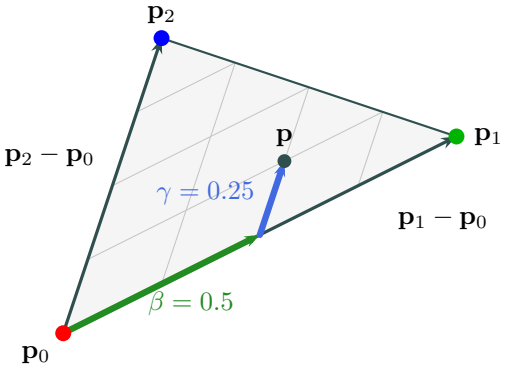



$$\mathbf{p} = \mathbf{p}_0 + \beta\,(\mathbf{p}_1 - \mathbf{p}_0) + \gamma\,(\mathbf{p}_2 - \mathbf{p}_0)$$

$\beta$ steps along one side, $\gamma$ along the other

the book, §2.9.1

# Barycentric coordinates: an address from the corners




$$\mathbf{p} = \mathbf{p}_0 + \beta\,(\mathbf{p}_1 - \mathbf{p}_0) + \gamma\,(\mathbf{p}_2 - \mathbf{p}_0)$$

$\beta$ steps along one side, $\gamma$ along the other

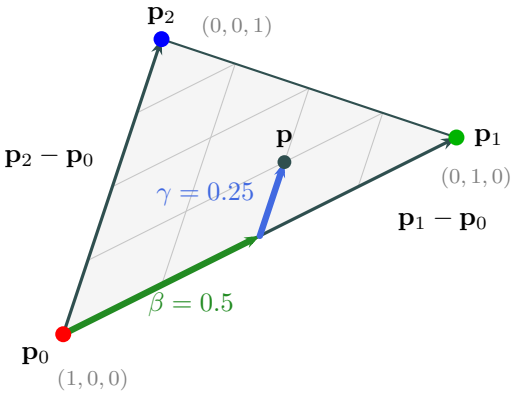
$$\mathbf{p} = \alpha\,\mathbf{p}_0 + \beta\,\mathbf{p}_1 + \gamma\,\mathbf{p}_2, \qquad \alpha = 1 - \beta - \gamma$$

$\alpha + \beta + \gamma = 1$: the **barycentric coordinates**

ours: $(\alpha, \beta, \gamma) = (0.25,\ 0.5,\ 0.25)$

the book, §2.9.1

# Barycentric coordinates: an address from the corners




$$\mathbf{p} = \mathbf{p}_0 + \beta\,(\mathbf{p}_1 - \mathbf{p}_0) + \gamma\,(\mathbf{p}_2 - \mathbf{p}_0)$$

$\beta$ steps along one side, $\gamma$ along the other

$$\mathbf{p} = \alpha\,\mathbf{p}_0 + \beta\,\mathbf{p}_1 + \gamma\,\mathbf{p}_2, \qquad \alpha = 1 - \beta - \gamma$$

$\alpha + \beta + \gamma = 1$: the **barycentric coordinates**

ours: $(\alpha, \beta, \gamma) = (0.25,\ 0.5,\ 0.25)$

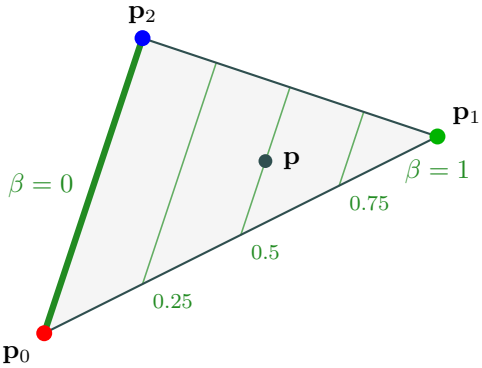
inside: all three $> 0$ on an edge: one is $0$

at a corner: one is $1$, the others $0$

the book, §2.9.1

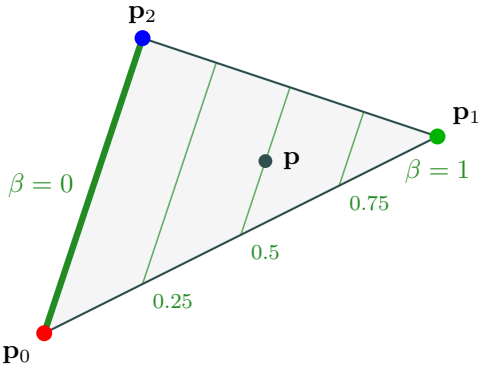
# Barycentric coordinates from the edge values

$\beta$: 0 on edge $\mathbf{p}_2\mathbf{p}_0$, 1 at $\mathbf{p}_1$, even steps




the book, eq. 2.32, 2.33, §9.1.2

# Barycentric coordinates from the edge values



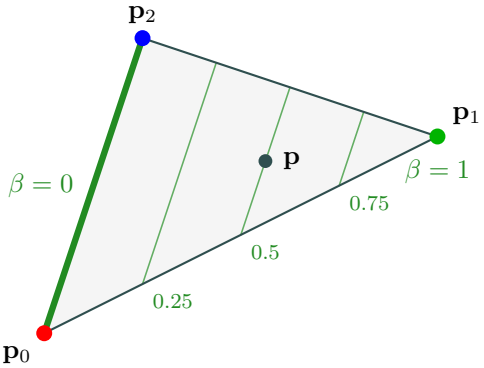

$\beta$: 0 on edge $\mathbf{p}_2\mathbf{p}_0$, 1 at $\mathbf{p}_1$, even steps

$$\beta = \frac{f_{20}(\mathbf{p})}{f_{20}(\mathbf{p}_1)}$$

$f_{20}$: also 0 on that edge, even steps; divide to get 1 at $\mathbf{p}_1$

the book, eq. 2.32, 2.33, §9.1.2

# Barycentric coordinates from the edge values




$\beta$: 0 on edge $\mathbf{p}_2\mathbf{p}_0$, 1 at $\mathbf{p}_1$, even steps

$$\beta = \frac{f_{20}(\mathbf{p})}{f_{20}(\mathbf{p}_1)}$$

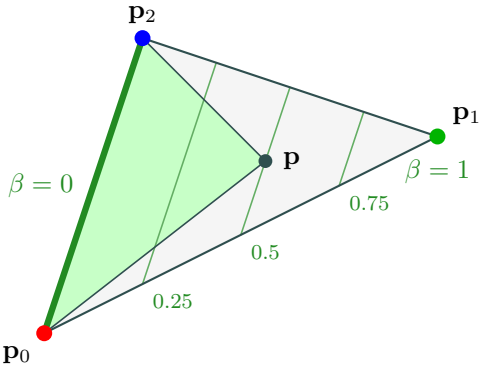
$f_{20}$: also 0 on that edge, even steps; divide to get 1 at $\mathbf{p}_1$

$$\alpha = \frac{f_{12}(\mathbf{p})}{f_{12}(\mathbf{p}_0)} \qquad \gamma = \frac{f_{01}(\mathbf{p})}{f_{01}(\mathbf{p}_2)}$$

each bottom is 40: twice the triangle's area

the book, eq. 2.32, 2.33, §9.1.2

# Barycentric coordinates from the edge values




$\beta$: 0 on edge $\mathbf{p}_2\mathbf{p}_0$, 1 at $\mathbf{p}_1$, even steps

$$\beta = \frac{f_{20}(\mathbf{p})}{f_{20}(\mathbf{p}_1)}$$

$f_{20}$: also 0 on that edge, even steps; divide to get 1 at $\mathbf{p}_1$

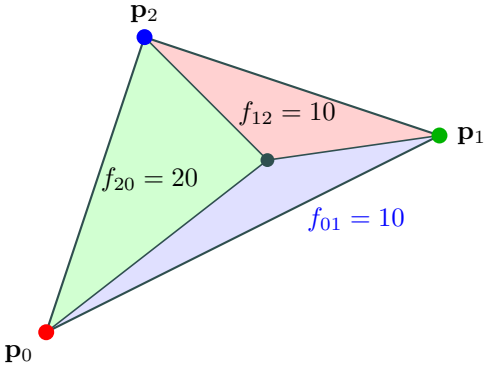
$$\alpha = \frac{f_{12}(\mathbf{p})}{f_{12}(\mathbf{p}_0)} \qquad \gamma = \frac{f_{01}(\mathbf{p})}{f_{01}(\mathbf{p}_2)}$$

each bottom is 40: twice the triangle's area

$$\beta = \frac{\text{green piece}}{\text{whole triangle}}$$ (edge value $= 2\times$ area)

the book, eq. 2.32, 2.33, §9.1.2

# Worked: pixel (5, 4)'s barycentric coordinates

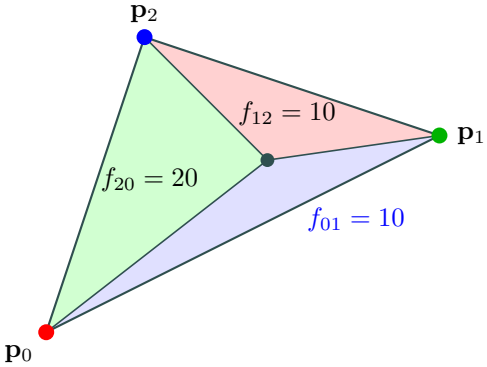



$\mathbf{p} = (5.5, 4.5)$: $f_{12} = 10$, $f_{20} = 20$, $f_{01} = 10$

every bottom: 40 (slide 39)

- Try first: $\alpha$, $\beta$, $\gamma$

# Worked: pixel (5, 4)'s barycentric coordinates



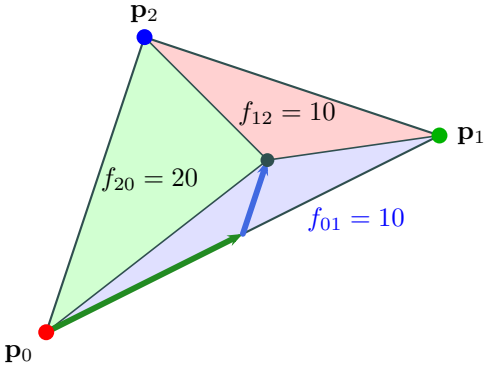

$\mathbf{p} = (5.5, 4.5)$: $f_{12} = 10$, $f_{20} = 20$, $f_{01} = 10$

every bottom: 40 (slide 39)

- Try first: $\alpha$, $\beta$, $\gamma$

$$\alpha = \frac{10}{40} = 0.25 \quad \beta = \frac{20}{40} = 0.5 \quad \gamma = \frac{10}{40} = 0.25$$

# Worked: pixel (5, 4)'s barycentric coordinates




$\mathbf{p} = (5.5, 4.5)$: $f_{12} = 10$, $f_{20} = 20$, $f_{01} = 10$

every bottom: 40 (slide 39)

- Try first: $\alpha$, $\beta$, $\gamma$

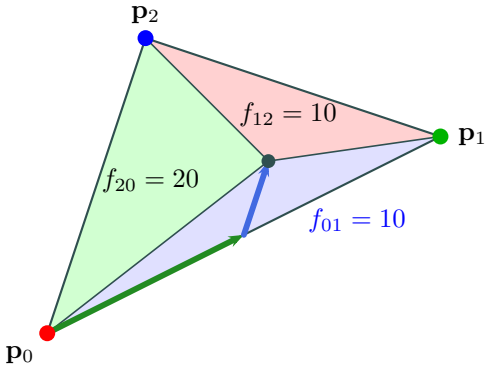
$$\alpha = \frac{10}{40} = 0.25 \quad \beta = \frac{20}{40} = 0.5 \quad \gamma = \frac{10}{40} = 0.25$$

check: $0.25 + 0.5 + 0.25 = 1$

check: $\mathbf{p}_0 + 0.5\,(8, 4) + 0.25\,(2, 6) = (5.5, 4.5)$

slide 38's address lands on $\mathbf{p}$

# Worked: pixel (5, 4)'s barycentric coordinates




$\mathbf{p} = (5.5, 4.5)$: $f_{12} = 10$, $f_{20} = 20$, $f_{01} = 10$

every bottom: 40 (slide 39)

- Try first: $\alpha$, $\beta$, $\gamma$

$$\alpha = \frac{10}{40} = 0.25 \quad \beta = \frac{20}{40} = 0.5 \quad \gamma = \frac{10}{40} = 0.25$$

check: $0.25 + 0.5 + 0.25 = 1$

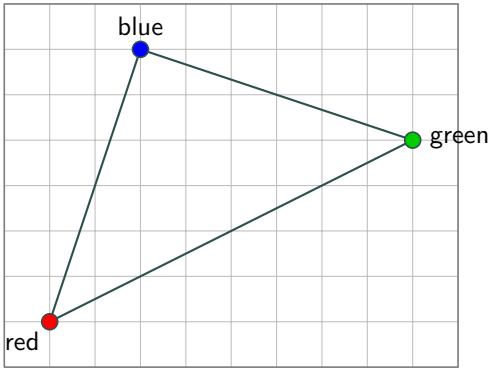
check: $\mathbf{p}_0 + 0.5\,(8, 4) + 0.25\,(2, 6) = (5.5, 4.5)$

slide 38's address lands on $\mathbf{p}$

- Half $\mathbf{p}_1$, a quarter each of $\mathbf{p}_0$, $\mathbf{p}_2$

# Barycentric coordinates blend whatever the corners carry

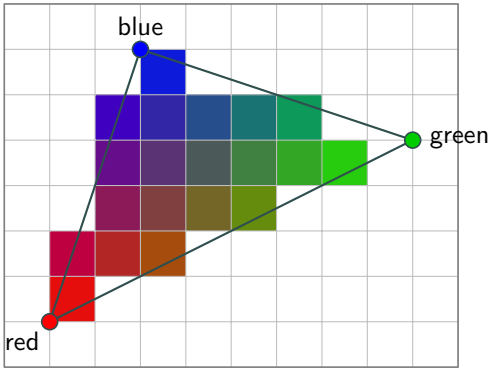
$$\mathbf{c} = \alpha\,\mathbf{c}_0 + \beta\,\mathbf{c}_1 + \gamma\,\mathbf{c}_2$$




- Corner data, blended by $\alpha$, $\beta$, $\gamma$

# Barycentric coordinates blend whatever the corners carry

$$\mathbf{c} = \alpha\,\mathbf{c}_0 + \beta\,\mathbf{c}_1 + \gamma\,\mathbf{c}_2$$

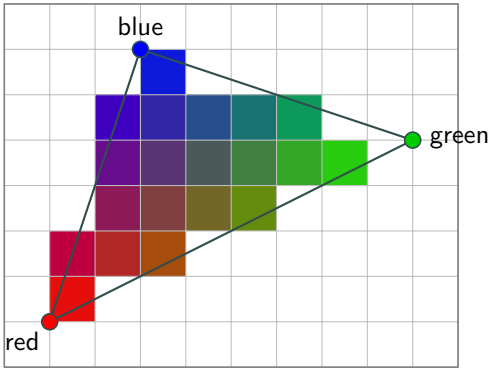



one flat colour per pixel

- Corner data, blended by $\alpha$, $\beta$, $\gamma$
- Even steps along every row

# Barycentric coordinates blend whatever the corners carry

$$\mathbf{c} = \alpha\,\mathbf{c}_0 + \beta\,\mathbf{c}_1 + \gamma\,\mathbf{c}_2$$




one flat colour per pixel

- Corner data, blended by $\alpha$, $\beta$, $\gamma$
- Even steps along every row
- Same for depth, normals, anything

# Worked: depth and colour at pixel (5, 4)

barycentric $(0.25,\ 0.5,\ 0.25)$; depths $0.2,\ 0.6,\ 0.4$

$$z = 0.25 \cdot 0.2 + 0.5 \cdot 0.6 + 0.25 \cdot 0.4$$

- Same $\alpha$, $\beta$, $\gamma$, every value

# Worked: depth and colour at pixel (5, 4)

barycentric $(0.25,\ 0.5,\ 0.25)$; depths $0.2,\ 0.6,\ 0.4$

$$z = 0.25 \cdot 0.2 + 0.5 \cdot 0.6 + 0.25 \cdot 0.4$$

$$= 0.05 + 0.3 + 0.1 = 0.45$$

- Same $\alpha$, $\beta$, $\gamma$, every value
- The depth the test will use

# Worked: depth and colour at pixel (5, 4)

barycentric $(0.25,\ 0.5,\ 0.25)$; depths $0.2,\ 0.6,\ 0.4$

$$z = 0.25 \cdot 0.2 + 0.5 \cdot 0.6 + 0.25 \cdot 0.4$$

$$= 0.05 + 0.3 + 0.1 = 0.45$$

$$\mathbf{c} = 0.25\,(1,0,0) + 0.5\,(0,1,0) + 0.25\,(0,0,1)$$

- Same $\alpha$, $\beta$, $\gamma$, every value
- The depth the test will use

# Worked: depth and colour at pixel (5, 4)

barycentric $(0.25,\ 0.5,\ 0.25)$; depths $0.2,\ 0.6,\ 0.4$

$$z = 0.25 \cdot 0.2 + 0.5 \cdot 0.6 + 0.25 \cdot 0.4$$

$$= 0.05 + 0.3 + 0.1 = 0.45$$

$$\mathbf{c} = 0.25\,(1,0,0) + 0.5\,(0,1,0) + 0.25\,(0,0,1)$$

$$= (0.25,\ 0.5,\ 0.25)$$



- Same $\alpha$, $\beta$, $\gamma$, every value
- The depth the test will use

# Worked: depth and colour at pixel (5, 4)

barycentric $(0.25,\ 0.5,\ 0.25)$; depths $0.2,\ 0.6,\ 0.4$

$$z = 0.25 \cdot 0.2 + 0.5 \cdot 0.6 + 0.25 \cdot 0.4$$
$$= 0.05 + 0.3 + 0.1 = 0.45$$

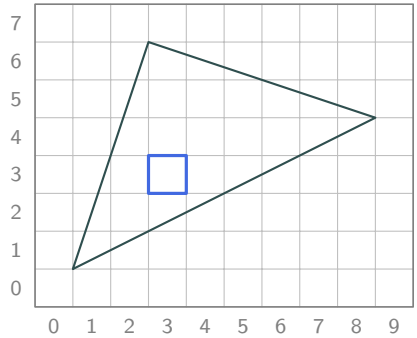
$$\mathbf{c} = 0.25\,(1,0,0) + 0.5\,(0,1,0) + 0.25\,(0,0,1)$$
$$= (0.25,\ 0.5,\ 0.25)$$



under perspective, colours need a fix: lecture 31 (§9.1.3)

- Same $\alpha$, $\beta$, $\gamma$, every value
- The depth the test will use
- Depth: right as it is

# Quick check 2: pixel (3, 3)

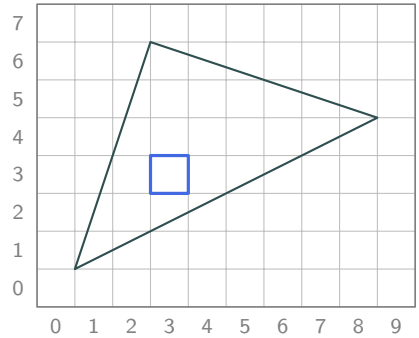



at $(3.5,\ 3.5)$: $f_{12} = 20,\ f_{20} = 10,\ f_{01} = 10$

corners: depths $0.2$, $0.6$, $0.4$; red, green, blue

(a) $\alpha$, $\beta$, $\gamma$? (b) depth? (c) colour?

# Quick check 2: pixel (3, 3)




at $(3.5,\ 3.5)$: $f_{12} = 20,\ f_{20} = 10,\ f_{01} = 10$

corners: depths $0.2$, $0.6$, $0.4$; red, green, blue

(a) $\alpha$, $\beta$, $\gamma$? (b) depth? (c) colour?

(a) $0.5,\ 0.25,\ 0.25$

(b) $0.1 + 0.15 + 0.1 = 0.35$

(c) $(0.5,\ 0.25,\ 0.25)$

# Recap: an edge value, divided, is a barycentric coordinate

$$\alpha = \frac{f_{12}(\mathbf{p})}{f_{12}(\mathbf{p}_0)} \qquad \beta = \frac{f_{20}(\mathbf{p})}{f_{20}(\mathbf{p}_1)} \qquad \gamma = \frac{f_{01}(\mathbf{p})}{f_{01}(\mathbf{p}_2)}$$

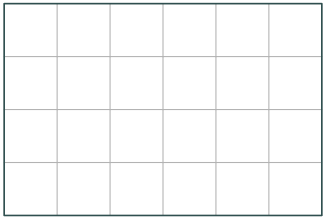
$$v = \alpha\, v_0 + \beta\, v_1 + \gamma\, v_2$$

# The depth buffer

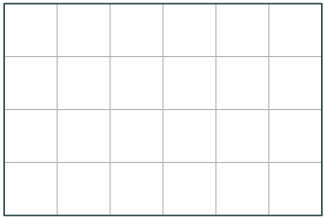
Lecture 15's nearest hit, the other way round

# A second grid: one depth per pixel

colour buffer: the picture



depth buffer: nearest so far

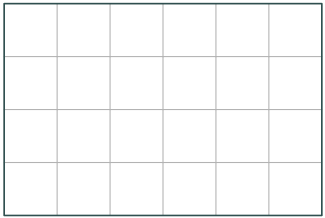


cleared to the background

- One more number per pixel

# A second grid: one depth per pixel

colour buffer: the picture



cleared to the background

depth buffer: nearest so far

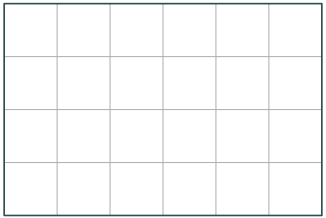
| | | | | | |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |

cleared to 1: the far end

- One more number per pixel
- Start: everything is far

# A second grid: one depth per pixel

colour buffer: the picture



cleared to the background

depth buffer: nearest so far

| | | | | | |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |

cleared to 1: the far end

- One more number per pixel
- Start: everything is far
- Cleared at the start of every frame

# Nearer than stored: write it. Farther: discard

```
for (i, j, z, c) in fragments:
    if z < depth[j][i]:
        image[j][i] = c
        depth[j][i] = z
    # else: thrown away
```

- Compare with what is stored

# Nearer than stored: write it. Farther: discard

```
for (i, j, z, c) in fragments:
    if z < depth[j][i]:
        image[j][i] = c
        depth[j][i] = z
    # else: thrown away
```

fragment $z = 0.45$
stored $0.7$: written

- Compare with what is stored
- Nearer: overwrite colour and depth

# Nearer than stored: write it. Farther: discard

```
for (i, j, z, c) in fragments:
    if z < depth[j][i]:
        image[j][i] = c
        depth[j][i] = z
    # else: thrown away
```

fragment $z = 0.45$
stored $0.7$: written

fragment $z = 0.7$
stored $0.45$: discarded

- Compare with what is stored
- Nearer: overwrite colour and depth
- Farther: drop it

# Worked: draw red, then blue

cleared

| | | | | | |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |

# Worked: draw red, then blue

cleared

| | | | | | |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |

after red

| | | | | | |
|---|---|---|---|---|---|
| 0.45 | 0.45 | 0.45 | 0.45 | 0.45 | 1 |
| 0.45 | 0.45 | 0.45 | 0.45 | 1 | 1 |
| 1 | 0.45 | 0.45 | 1 | 1 | 1 |
| 1 | 0.45 | 1 | 1 | 1 | 1 |

red: $0.45$ everywhere

- Red: every fragment beats 1

# Worked: draw red, then blue

cleared

| | | | | | |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |

after red

| | | | | | |
|---|---|---|---|---|---|
| 0.45 | 0.45 | 0.45 | 0.45 | 0.45 | 1 |
| 0.45 | 0.45 | 0.45 | 0.45 | 1 | 1 |
| 1 | 0.45 | 0.45 | 1 | 1 | 1 |
| 1 | 0.45 | 1 | 1 | 1 | 1 |

red: $0.45$ everywhere

after blue

| | | | | | |
|---|---|---|---|---|---|
| 0.2 | 0.3 | 0.4 | 0.45 | 0.45 | 1 |
| 0.2 | 0.3 | 0.4 | 0.45 | 1 | 1 |
| 0.2 | 0.3 | 0.45 | 1 | 1 | 1 |
| 0.2 | 0.45 | 1 | 1 | 1 | 1 |

blue: $0.2$, $0.3$, $\ldots$ by column

- Red: every fragment beats 1
- Blue wins left of 0.45, loses right

# Swap the order: the same picture

blue first

| | | | | | |
|---|---|---|---|---|---|
| 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 1 |
| 0.2 | 0.3 | 0.4 | 0.5 | 1 | 1 |
| 0.2 | 0.3 | 1 | 1 | 1 | 1 |
| 0.2 | 1 | 1 | 1 | 1 | 1 |

- Either order, same picture

# Swap the order: the same picture

blue first

| | | | | | |
|---|---|---|---|---|---|
| 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 1 |
| 0.2 | 0.3 | 0.4 | 0.5 | 1 | 1 |
| 0.2 | 0.3 | 1 | 1 | 1 | 1 |
| 0.2 | 1 | 1 | 1 | 1 | 1 |

then red

| | | | | | |
|---|---|---|---|---|---|
| 0.2 | 0.3 | 0.4 | 0.45 | 0.45 | 1 |
| 0.2 | 0.3 | 0.4 | 0.45 | 1 | 1 |
| 0.2 | 0.3 | 0.45 | 1 | 1 | 1 |
| 0.2 | 0.45 | 1 | 1 | 1 | 1 |

- Either order, same picture

# Swap the order: the same picture

blue first

| | | | | | |
|---|---|---|---|---|---|
| 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 1 |
| 0.2 | 0.3 | 0.4 | 0.5 | 1 | 1 |
| 0.2 | 0.3 | 1 | 1 | 1 | 1 |
| 0.2 | 1 | 1 | 1 | 1 | 1 |

then red

| | | | | | |
|---|---|---|---|---|---|
| 0.2 | 0.3 | 0.4 | 0.45 | 0.45 | 1 |
| 0.2 | 0.3 | 0.4 | 0.45 | 1 | 1 |
| 0.2 | 0.3 | 0.45 | 1 | 1 | 1 |
| 0.2 | 0.45 | 1 | 1 | 1 | 1 |

=

red first (slide 48)

| | | | | | |
|---|---|---|---|---|---|
| 0.2 | 0.3 | 0.4 | 0.45 | 0.45 | 1 |
| 0.2 | 0.3 | 0.4 | 0.45 | 1 | 1 |
| 0.2 | 0.3 | 0.45 | 1 | 1 | 1 |
| 0.2 | 0.45 | 1 | 1 | 1 | 1 |

- Either order, same picture
- Lecture 15, slide 21: same idea

# Lecture 15's nearest hit, the other way round

**Part A**: lecture 15, slide 17 (inside the pixel loop)

```
for surf in self.surface_list:
    rec = surf.hit(ray, t0, t1)
    if rec.t < math.inf:
        closest_hit = rec
        t1 = rec.t
```

- Same question: what is nearest?

# Lecture 15's nearest hit, the other way round

**Part A**: lecture 15, slide 17 (inside the pixel loop)

```
for surf in self.surface_list:
    rec = surf.hit(ray, t0, t1)
    if rec.t < math.inf:
        closest_hit = rec
        t1 = rec.t
```

**Part B**: the depth buffer

```
for tri in triangles:
  for (i, j, z, c) in frags(tri):
    if z < depth[j][i]:
      image[j][i] = c
      depth[j][i] = z
```

- Same question: what is nearest?

# Lecture 15's nearest hit, the other way round

**Part A**: lecture 15, slide 17 (inside the pixel loop)

```
for surf in self.surface_list:
    rec = surf.hit(ray, t0, t1)
    if rec.t < math.inf:
        closest_hit = rec
        t1 = rec.t
```

**Part B**: the depth buffer

```
for tri in triangles:
  for (i, j, z, c) in frags(tri):
    if z < depth[j][i]:
      image[j][i] = c
      depth[j][i] = z
```



the best so far, made smaller

- Same question: what is nearest?
- $t_1$ per ray; depth per pixel

# Lecture 15's nearest hit, the other way round

**Part A**: lecture 15, slide 17 (inside the pixel loop)

```
for surf in self.surface_list:
    rec = surf.hit(ray, t0, t1)
    if rec.t < math.inf:
        closest_hit = rec
        t1 = rec.t
```

**Part B**: the depth buffer

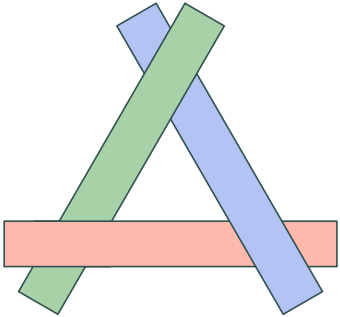
```
for tri in triangles:
  for (i, j, z, c) in frags(tri):
    if z < depth[j][i]:
      image[j][i] = c
      depth[j][i] = z
```

the best so far, made smaller

- Same question: what is nearest?
- $t_1$ per ray; depth per pixel
- Nothing faked: the first ray's answer, with no ray

  no ray skipped: the answer is exact, wherever both sample the pixel's centre

# Why not paint from far to near?

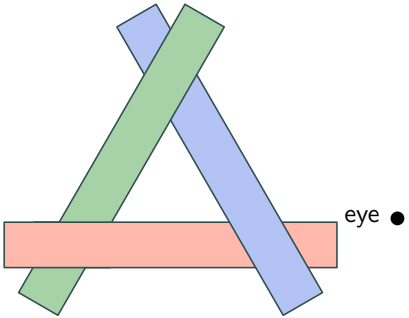


each bar is over the next

no far-to-near order exists

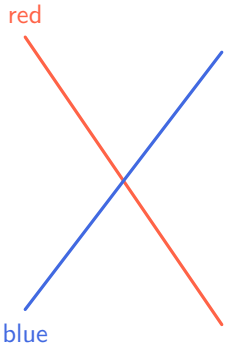
- Painter: draw far things first

# Why not paint from far to near?




each bar is over the next

no far-to-near order exists

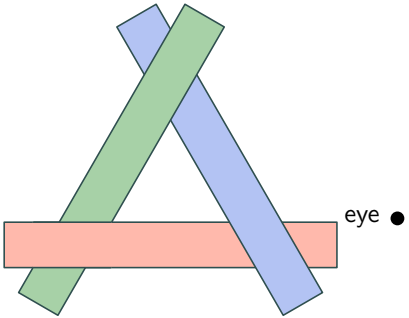



seen from the side

crossing: no order

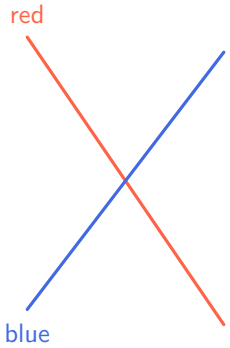
- Painter: draw far things first
- Cycles and crossings: no order

# Why not paint from far to near?




each bar is over the next

no far-to-near order exists



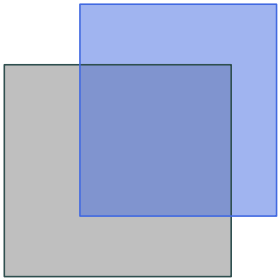

seen from the side

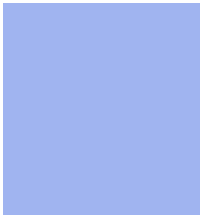
crossing: no order

- Painter: draw far things first
- Cycles and crossings: no order
- Sorting every frame is slow

# Mistake 1: see-through surfaces need an order

wall, then glass





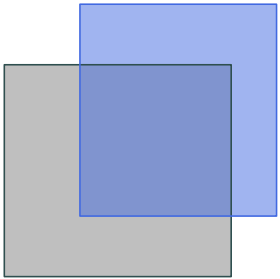
right

glass: lecture 3, slide 8's alpha $= 0.5$

- One depth per pixel: one surface

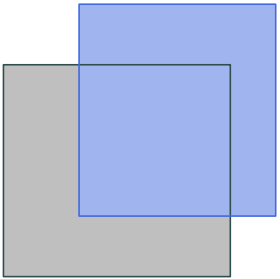
# Mistake 1: see-through surfaces need an order

wall, then glass



right

glass, then wall



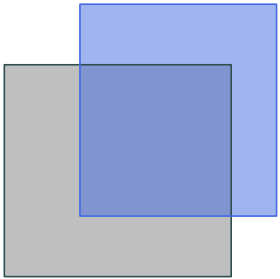
wall missing behind the glass

glass: lecture 3, slide 8's alpha $= 0.5$

- One depth per pixel: one surface
- Glass first: the wall is discarded

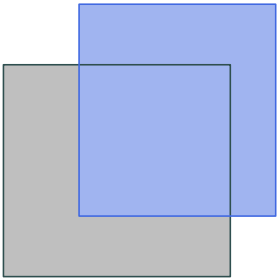
# Mistake 1: see-through surfaces need an order

wall, then glass



right

glass, then wall

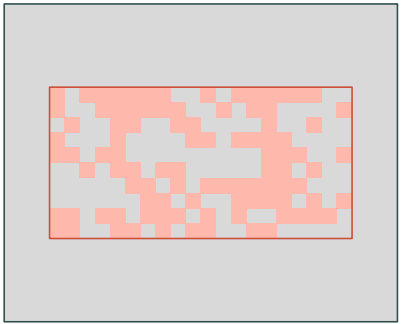


wall missing behind the glass

glass: lecture 3, slide 8's alpha $= 0.5$

- One depth per pixel: one surface
- Glass first: the wall is discarded
- Fix: solids first, then glass far to near

# Mistake 2: equal depths fight

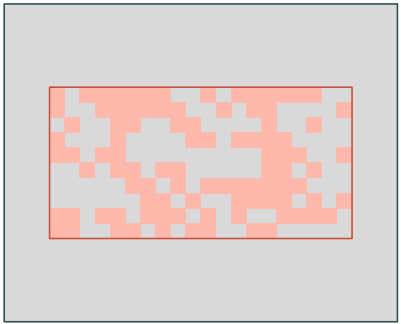


a poster on a wall

(a sketch of the pattern)

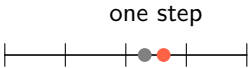
- Two surfaces at one depth

# Mistake 2: equal depths fight



a poster on a wall

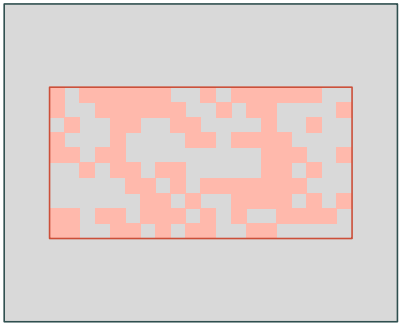
(a sketch of the pattern)




wall and poster:
the same step

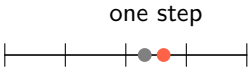
- Two surfaces at one depth
- Depths are kept in steps

# Mistake 2: equal depths fight



a poster on a wall

(a sketch of the pattern)

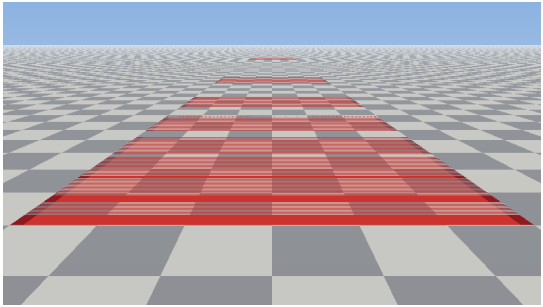



wall and poster:
the same step

who wins: rounding,
pixel by pixel

- Two surfaces at one depth
- Depths are kept in steps
- Far away the steps grow: lecture 28

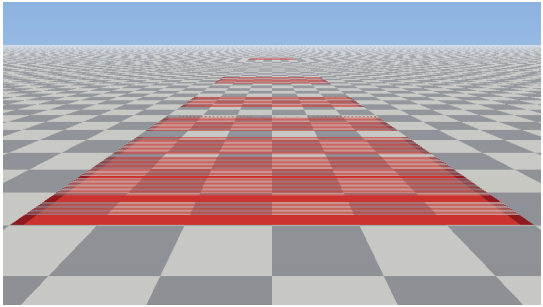
# What z-fighting really looks like



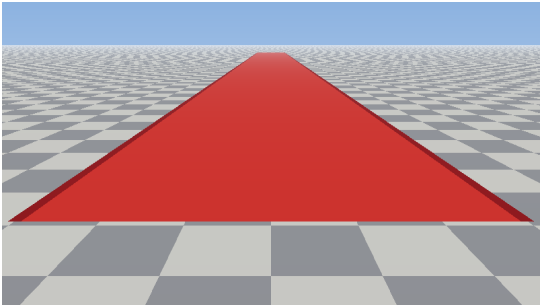
rug 1 mm above the floor: they fight

drawn with this lecture's rasterizer and a 16-bit depth buffer

# What z-fighting really looks like



rug 1 mm above the floor: they fight



lifted to 5 cm: clean

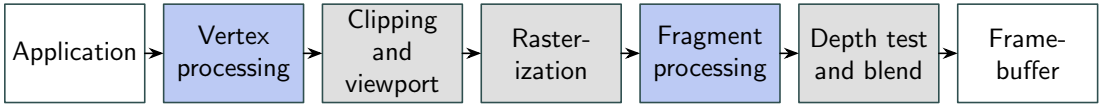
drawn with this lecture's rasterizer and a 16-bit depth buffer

# Recap: keep the nearest depth per pixel

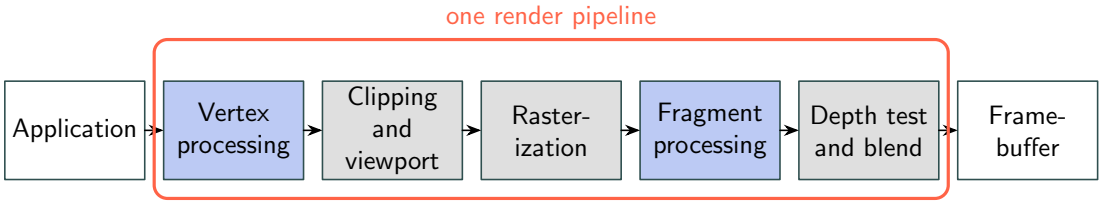
`if z < depth[j][i]:` write colour and depth

exact for solid surfaces  glass: sort it  equal depths: fight

# Lecture 25: the stops become one object

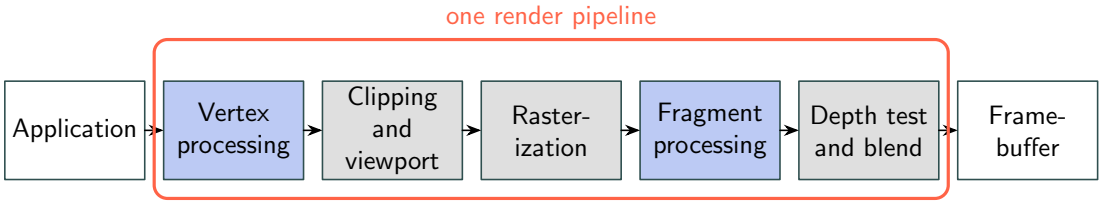


# Lecture 25: the stops become one object




- Programs and settings, fixed together

# Lecture 25: the stops become one object

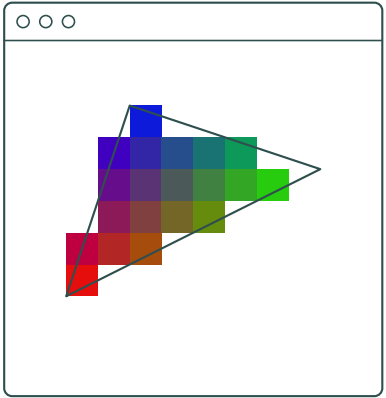



- Programs and settings, fixed together
- Built once, used every frame
- Friday: built in Python

# Summary

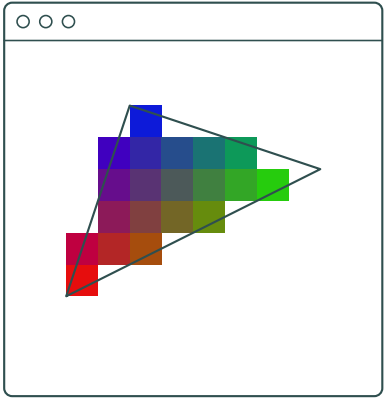
| | |
|---|---|
| 1 One triangle's trip | corners → box → pixels → fragments → colour → depth<br>your programs: per vertex, per fragment; the rest are circuits |
| 2 Inside the rasterizer | $f_{01} = (y_0 - y_1)\,x + (x_1 - x_0)\,y + x_0 y_1 - x_1 y_0$<br>inside: three signs agree; one add per step; top-left rule |
| 3 Barycentric coordinates | $\alpha = f_{12}(\mathbf{p})/f_{12}(\mathbf{p}_0), \;\; v = \alpha v_0 + \beta v_1 + \gamma v_2$<br>the edge values, as shares of the area; they add to one |
| 4 The depth buffer | `if z < depth[j][i]`: lecture 15's $t_1$, per pixel<br>exact for solids, any order; glass needs sorting; equal depths fight |

# Next lecture: the API and the first triangle



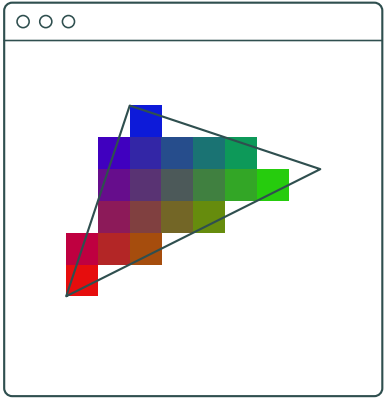
- WebGPU's objects, in the order you make them

# Next lecture: the API and the first triangle



- WebGPU's objects, in the order you make them
- This triangle in Python and WGSL

# Next lecture: the API and the first triangle



- WebGPU's objects, in the order you make them
- This triangle in Python and WGSL
- Assignment 8 released on Friday

# Exit Ticket

## Today's question *(2 minutes)*

A triangle has corners $\mathbf{p}_0 = (0, 0)$, $\mathbf{p}_1 = (4, 0)$, $\mathbf{p}_2 = (0, 4)$. At pixel $(1, 0)$'s centre: $f_{12} = 8$, $f_{20} = 6$, $f_{01} = 2$.

**(a)** Is the pixel covered? Give its barycentric coordinates.

**(b)** The corners' depths are $0.4$, $0.8$ and $0.2$; the depth buffer holds $0.5$ there. Is the fragment kept?

**CS 453 — Exit Ticket #24**

Name: _____

Roll #: _____ Date: _____

**Answer:**

*Hand it to me on your way out.*
*No name, no attendance.*