Finger Exercises 24

The Pipeline: One Triangle's Trip, and the Depth Buffer

≈ 90 minutes, plus a bonus

CS 453 — Computer Graphics

You need a pencil, a ruler, a calculator and, if you have them, a coloured pencil or two. The words are the lecture's. A *vertex* is a corner with the data it carries. A *fragment* is one pixel's share of a triangle. *The box* is where the vertex program puts every corner: x and y from -1 to 1, depth z from 0 (near) to 1 (far). The sheet follows the lecture's four parts. It starts very easy and ends with the cases that catch people out. Nothing here is an assignment task; the sheet is Lecture 24 only.

Short on time? Do Exercises 1, 5, 7, 9, 12, 14 and 15 (about 40 minutes): the trip, the viewport, the edge test, a whole triangle by hand, edge pixels, barycentric coordinates and the depth buffer.

The rules on this sheet (the lecture's). **Pixels:** pixel (i, j) is the square from (i, j) to $(i+1, j+1)$; its centre is $(i+0.5,\ j+0.5)$. y goes up; rows are counted from the bottom. **Corners** are listed counter-clockwise: $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{p}_2$. **Edge function** of the edge from $\mathbf{a}$ to $\mathbf{b}$, at a point $\mathbf{p}$: $f = |\,\mathbf{b}-\mathbf{a}\quad \mathbf{p}-\mathbf{a}\,| = (x_b - x_a)(y - y_a) - (y_b - y_a)(x - x_a)$, positive when $\mathbf{p}$ is on the left as you walk from $\mathbf{a}$ to $\mathbf{b}$. The book's form (equation 9.1) for f_{01}: $(y_0 - y_1)\,x + (x_1 - x_0)\,y + x_0 y_1 - x_1 y_0$. **Covered:** $f_{12} \geq 0$, $f_{20} \geq 0$ and $f_{01} \geq 0$ at the centre (Exercise 12 settles the zeros). **Barycentric:** $\alpha = f_{12}(\mathbf{p})/f_{12}(\mathbf{p}_0)$, $\beta = f_{20}(\mathbf{p})/f_{20}(\mathbf{p}_1)$, $\gamma = f_{01}(\mathbf{p})/f_{01}(\mathbf{p}_2)$; a value v at the pixel is $\alpha v_0 + \beta v_1 + \gamma v_2$. **Depth buffer:** cleared to 1; a fragment is written only if its depth is *less* than the stored one.

Part 1 · One triangle's trip

From three corners to coloured pixels

1 Seven stops, two programs

about 4 min

The order of the stops, what travels between them, and which stops run your own programs.

	depth test and blend	rasterization	framebuffer	application	clipping and viewport	fragment processing	vertex processing
stop no.	☐	☐	☐	☐	☐	☐	☐

(a) Write each card's number, 1 to 7, in its small box, in the order a triangle meets them.

(b) Two stops run programs that *you* write for the GPU's lanes. Which two? Which stop runs on the CPU? The rest are circuits.

(c) What travels along each arrow? Use each phrase once: *fragments* · *corners in a buffer* · *four numbers* (x, y, z, w) *per corner* · *corners in pixels, with their depths* · *one colour per fragment* · *the winning colours.*

stop 1 → 2: _____ stop 4 → 5: _____

stop 2 → 3: _____ stop 5 → 6: _____

stop 3 → 4: _____ stop 6 → 7: _____

(d) Which stop (give its number) (i) runs once for every corner? ____ (ii) runs once for every covered pixel? ____ (iii) can throw a fragment away? ____ (iv) can turn one triangle into two? ____ (v) divides by w? ____ (vi) is the last one that needs to know where the camera is? ____

2 Count the work

about 4 min

The vertex program runs once per corner; the fragment program once per covered pixel, whether or not the fragment is kept later.

Models here are sent as plain triangles: three corners each, no sharing.

what is drawn	corners	vertex program runs	fragment program runs
one triangle covering 15 pixels	3	3	15
a square of 2 triangles, 100×100 pixels	_____	_____	_____
a disc of 20 triangles covering 1,250 pixels	_____	_____	_____
the square and the disc; the square is in front and hides the whole disc	_____	_____	_____
a square of 2 triangles filling a 1920×1080 screen	_____	_____	_____

(a) Fill the table. Careful with the fourth row: where in the trip is the hidden disc thrown away?

(b) In the fourth row, how many fragments are *written* into the picture if the square is drawn first? If the disc is drawn first?

(c) The disc's buffer holds x, y, z and red, green, blue for every corner, 4 bytes each. How many bytes? How often is it copied to the GPU, and what does the CPU send every frame?

(d) The full-screen square at 60 frames a second: how many fragment program runs a second? How many vertex program runs? Which program should you keep short?

3 Divide by w

about 5 min

The vertex program hands back (x, y, z, w); a circuit divides by w. A corner is in the box when $-1 \le x, y \le 1$ and $0 \le z \le 1$ after the divide.

On this sheet's camera, w is the corner's distance in front of the eye. The near plane is at distance 1, the far plane at distance 3. Six corners come out of the vertex program:

corner	(x, y, z, w)	distance in front	x/w	y/w	z/w	in the box?
A	(0.8, 0.5, 1.5, 2)	2	0.4	0.25	0.75	yes
B	(2.4, 0, 1.5, 2)	_____	_____	_____	_____	_____
C	(0, 1, 4.5, 4)	_____	_____	_____	_____	_____
D	(0.4, 0.2, −0.75, 0.5)	_____	_____	_____	_____	_____
E	(1.6, 1, −4.5, −2)	_____	_____	_____	_____	_____
F	(0.5, 0.5, −1.5, 0)	_____	_____	_____	_____	_____

(a) Fill the table.

(b) Which corner is (i) off to one side of the screen? (ii) beyond the far plane? (iii) in front of the eye but nearer than the near plane? (iv) behind the eye? (v) level with the eye?

(c) Before the divide, E's x is $+1.6$: E is on the right. After it, E's x is -0.8. What happened? Its z/w says "far away"; is it?

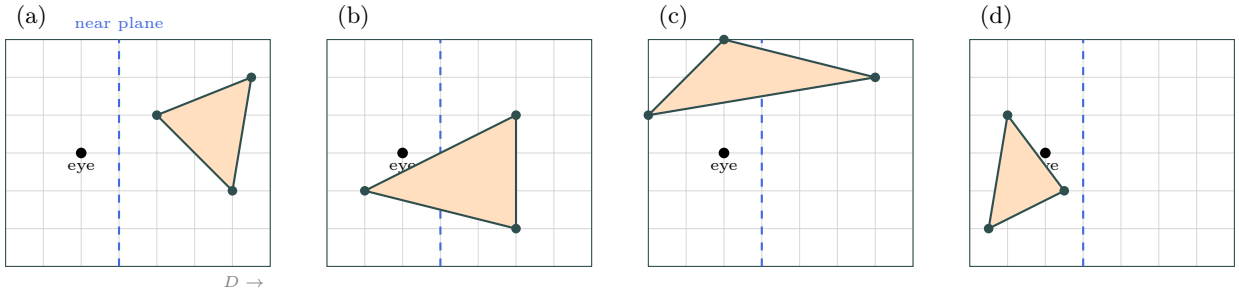
(d) Which corners lie behind the near plane, so that clipping must cut their triangles *before* the divide? Why can the divide not simply come first?

4 Cut at the near plane

about 5 min

Clipping keeps the part of a triangle in front of the near plane. Cut points come from a lerp, as in week 2.

Each panel is a view from above: distance D in front of the eye to the right, sideways X up, one square per unit. The eye is at $D = 0$; the near plane is the dashed line $D = 1$.



Corners: (a) $(2, 1)$, $(4, -1)$, $(4.5, 2)$; (b) A $= (3, 1)$, B $= (3, -2)$, C $= (-1, -1)$; (c) $(4, 2)$, $(0, 3)$, $(-2, 1)$; (d) $(0.5, -1)$, $(-1.5, -2)$, $(-1, 1)$, written as (D, X).

panel	(a)	(b)	(c)	(d)
corners behind the near plane	____	____	____	____
corners after the cut	____	____	____	____
triangles sent on	____	____	____	____

(a) In each panel, mark where the edges cross the near plane and shade the part that survives. Then fill the table.

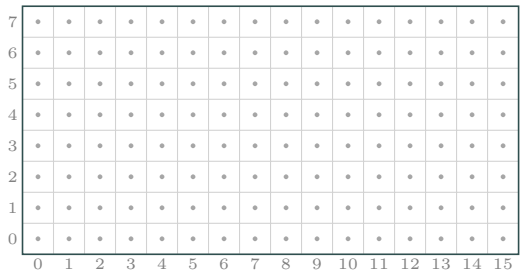
(b) Panel (b): the edge from A to C crosses $D = 1$ at $\mathbf{A} + t(\mathbf{C} - \mathbf{A})$. Find t and the cut point. Do the same for the edge from B to C, and draw the two triangles the rasterizer gets. Then find panel (c)'s two cut points the same way.

(c) Complete the rule: 0, 1, 2, 3 corners behind the plane give ___, ___, ___, ___ triangles. And one corner exactly *on* the plane, two in front?

5 From the box to pixels

about 5 min

The viewport: $x = (x_n + 1)\,W/2$, $y = (y_n + 1)\,H/2$. WebGPU counts rows from the top: row $= (1 - y_n)\,H/2$.



a screen of $W \times H = 16 \times 8$ pixels

(x_n, y_n)	x	y	row	where it lands
$(0.3125,\ -0.375)$	10.5	2.5	5.5	centre of $(10, 2)$
$(-1,\ -1)$	____	____	____	_____
$(1,\ 1)$	____	____	____	_____
$(0,\ 0)$	____	____	____	_____
$(-0.4375,\ 0.625)$	____	____	____	_____
$(1.25,\ 0)$	____	____	____	_____

"row": WebGPU's row, counted from the top.

(a) Fill the table and mark each point on the grid.

(b) $(1, 1)$ lands at $(16, 8)$: which pixel is that? And $(0, 0)$ on a 1920×1080 screen: where does it land? Is it a pixel centre?

(c) Backwards: which point of the box lands on the centre of pixel $(11, 2)$?

(d) The last point is outside the box. Does the rasterizer make fragments for it? (Lecture: what happens to pixels off the screen?)

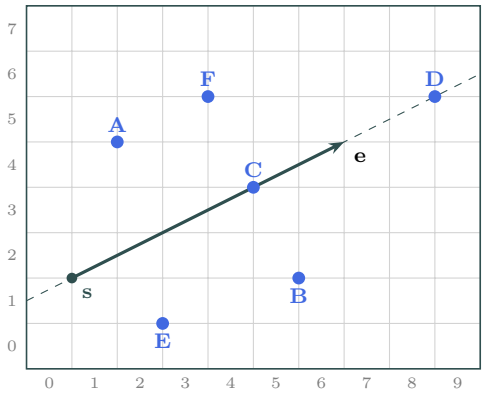
Part 2 · Inside the rasterizer

Three signs for every pixel

6 Left, right or on? Guess first

about 4 min

The sign of the edge function is the turn from the edge's arrow to the point's arrow: left +, right −, straight 0.



Edge from $\mathbf{s} = (1, 2)$ to $\mathbf{e} = (7, 5)$, so $\mathbf{a} = \mathbf{e} - \mathbf{s} = (6, 3)$. For each point, $\mathbf{b} = \mathbf{p} - \mathbf{s}$ and $f = x_a y_b - y_a x_b$.

point	$\mathbf{p}$	guess	$\mathbf{b}$	f
A	$(2, 5)$	+	$(1, 3)$	$18 - 3 = 15$
B	$(6, 2)$	___	_____	_____
C	$(5, 4)$	___	_____	_____
D	$(9, 6)$	___	_____	_____
E	$(3, 1)$	___	_____	_____
F	$(4, 6)$	___	_____	_____

(a) Guess every sign from the picture first. Then compute f.

(b) D gives 0, but D is not on the edge from **s** to **e**. What does f really test: a side of the edge, or a side of something bigger?

(c) A and F give the same value. Draw the line through them. How does it sit against the edge? Where are all points with $f = 15$?

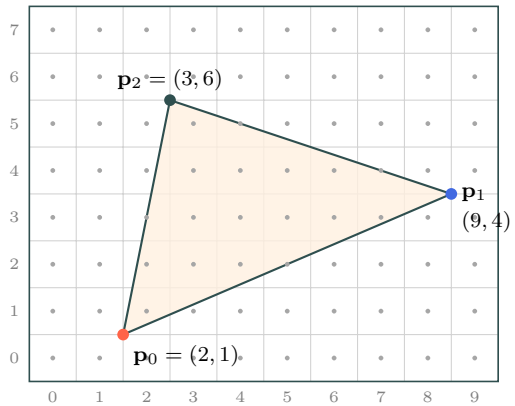
(d) Write f in the book's form, $(y_s - y_e)\,x + (x_e - x_s)\,y + x_s y_e - x_e y_s$, and check it at E.

(e) If the edge is walked from **e** to **s** instead, what happens to the six values?

7 The sheet's triangle

about 5 min

One edge function per edge, each worked out once per triangle, before any pixel is looked at.



T, used again in Exercises 8, 9, 12 and 14.

edge	$y_a - y_b$	$x_b - x_a$	$x_a y_b - x_b y_a$	$f(x, y)$
f_{12}: $\mathbf{p}_1 \to \mathbf{p}_2$	-2	-6	42	$-2x - 6y + 42$
f_{20}: $\mathbf{p}_2 \to \mathbf{p}_0$	_____	_____	_____	_____
f_{01}: $\mathbf{p}_0 \to \mathbf{p}_1$	_____	_____	_____	_____

Here **a** is the edge's start and **b** its end.

(a) Fill the table.

(b) Each edge function must be 0 at its own two corners. Check all six.

(c) Work out $f_{12}(\mathbf{p}_0)$, $f_{20}(\mathbf{p}_1)$ and $f_{01}(\mathbf{p}_2)$. They should be equal. Why? (What is the number?) Which sign would clockwise corners give?

(d) Find T's area another way: its box is 7×5; cut off the three right-angled triangles between T and the box's sides. Does it agree with (c)?

8 Four pixels, one box

about 4 min

Covered means all three values are at least 0 at the centre. One negative value is enough to reject.

pixel	centre	f_{12}	f_{20}	f_{01}	covered?
$(4,3)$	$(4.5,\ 3.5)$	12	10	10	yes
$(7,2)$					
$(2,3)$					
$(1,6)$					

(a) Fill the table with T's edge functions from Exercise 7.

(b) Pixel $(2,3)$ has a zero. Where exactly is its centre? Mark it on Exercise 7's picture.

(c) Pixel $(1,6)$ has a zero too. Is its centre on T's edge? Why is it not covered?

(d) The lecture's loop visits the box: `range(floor(min(xs)), ceil(max(xs)))` for the columns, the same for the rows. Which columns and rows does T's box have? How many centres are tested? Draw the box on Exercise 7's picture.

9 A whole triangle, by adding

about 7 min

A step right adds $y_0 - y_1$; a step up adds $x_1 - x_0$. One full value per edge, then only additions.

The four grids are T's box: columns 2 to 8, rows 1 to 5. The first two are filled for you. Each number is the edge value at that pixel's centre.

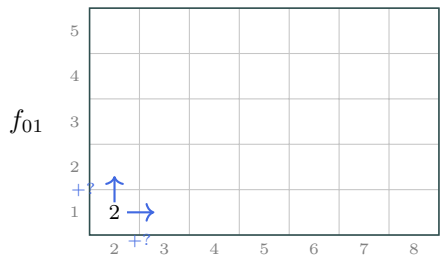
f_{12}

	2	3	4	5	6	7	8
5	4	2	0	−2	−4	−6	−8
4	10	8	6	4	2	0	−2
3	16	14	12	10	8	6	4
2	22	20	18	16	14	12	10
1	28	26	24	22	20	18	16

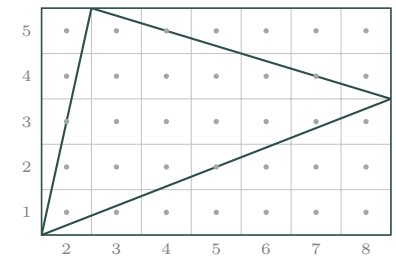
f_{20}

	2	3	4	5	6	7	8
5	−2	3	8	13	18	23	28
4	−1	4	9	14	19	24	29
3	0	5	10	15	20	25	30
2	1	6	11	16	21	26	31
1	2	7	12	17	22	27	32

f_{01}



covered



(a) From $f_{01} = -3x + 7y - 1$: what does one step right add? One step up? Write them by the arrows.

(b) Fill the f_{01} grid from the 2 in the corner, using only additions. Check one far cell, $(8,5)$, with the formula.

(c) Read the steps of the two filled grids. Do they match f_{12}'s and f_{20}'s formulas?

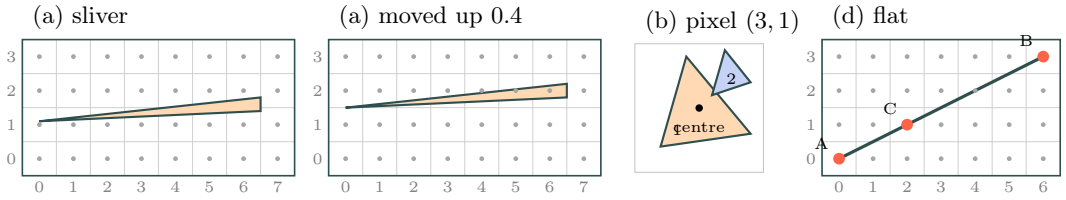
(d) On the last grid, shade every pixel whose three values are all at least 0. How many? Circle the ones with a 0 among their values.

(e) How many of the 35 tests were wasted on pixels outside T?

10 Strange triangles

about 6 min

Only centres count. Slivers, specks, backwards and flat triangles, and triangles that hang off the screen.



(a) Sliver. How many fragments does each sliver make? A long thin triangle drifts slowly upward on the screen, one frame at a time. What does the viewer see?

(b) Specks. Two tiny triangles inside pixel $(3, 1)$. How many fragments does each make? Does it matter how much of the pixel's square a triangle covers?

(c) Backwards. Someone sends T as $\mathbf{p}_0, \mathbf{p}_2, \mathbf{p}_1$. What are the three edge values at pixel $(4, 3)$ now? How many fragments does the lecture's loop make?

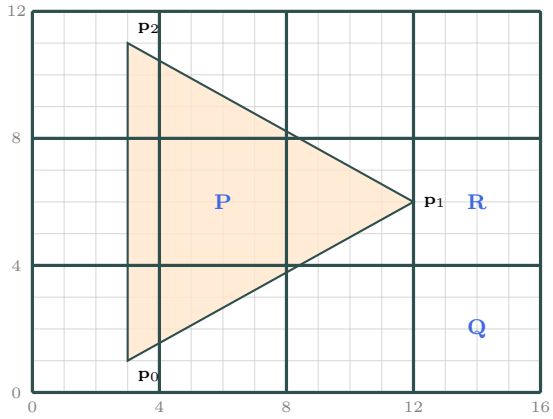
(d) Flat. Triangle A $(0.5, 0.5)$, B $(6.5, 3.5)$, C $(2.5, 1.5)$ has all three corners on one line. What is its $2A$? With the test ≥ 0, which pixels get a fragment? What goes wrong in Part 3, where we divide by $2A$? What should a rasterizer do with such a triangle?

(e) Off the screen. T is moved 4 pixels to the right on its 10×8 screen: $(6, 1)$, $(13, 4)$, $(7, 6)$. How many covered centres fall in columns 10 and beyond? (Use Exercise 9.) What happens if the loop writes pixel $(11, 3)$ into a 10×8 picture? Fix the box so this cannot happen. How many tests are left?

11 Whole blocks at once

about 5 min

Test a 4×4 block by its four corners: all four negative for one edge, skip it; all four ≥ 0 for all three edges, take it whole; otherwise test its 16 pixels.



Triangle $(3, 1)$, $(12, 6)$, $(3, 11)$: $f_{12} = -5x - 9y + 114$, $f_{20} = 10x - 30$, $f_{01} = -5x + 9y + 6$. Block corners in the order bottom-left, bottom-right, top-left, top-right:

block	f_{12}	f_{20}	f_{01}	decide
P	$58, 38, 22, 2$	$10, 50, 10, 50$	$22, 2, 58, 38$	whole
Q	$54, 34, 18, -2$	$90, 130, 90, 130$	$-54, -74, -18, -38$	_____
R	_____	_____	_____	_____

(a) By eye first: write in each of the 12 blocks *out* (no part of the triangle), *in* (all triangle) or *part*.

(b) Decide Q. Then work out R's twelve corner values and decide R.

(c) R holds no part of the triangle (the tip only touches its side), yet its corner test says "test all 16". Look at R's corners: which edge is each one outside? Why can no single edge throw R out?

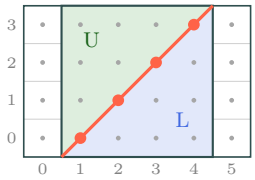
(d) Which test from the lecture's loop, done before any block, throws R out at once?

12 One owner for an edge pixel

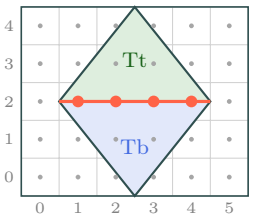
about 6 min

A centre exactly on a shared edge has $f = 0$ for both triangles. Rule: walk the row left to right; the triangle the row enters *there draws it. A flat edge, which no row enters, counts when it is the triangle's top edge.*

(1) a square



(2) a flat edge



square (1)	L	U	total	you see
≥ 0 on every edge	___	___	___	_____
> 0 on every edge	___	___	___	_____
the rule	___	___	___	_____

L, U: each triangle's fragments.

(1) $L = (1,0), (5,0), (5,4)$ and $U = (1,0), (5,4), (1,4)$, both counter-clockwise; 16 pixels in the square, 4 centres on the diagonal. (2) $Tt = (1,2.5), (5,2.5), (3,5)$ and $Tb = (1,2.5), (3,0), (5,2.5)$.

(a) In square (1), walk the row through each red centre from left to right. Which triangle does the row enter at the diagonal? Fill the table.

(b) Panel (2): does any row enter Tt or Tb through the red edge? Whose *top* edge is it? Who draws the four red centres, and how many fragments does each triangle make?

(c) Back to T (Exercise 9): which of its four zero pixels does T keep? For each of the other three, say which edge it is on and why T leaves it to a neighbour. How many fragments does T really make?

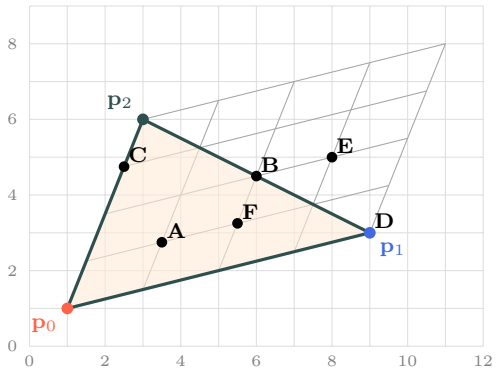
Part 3 · Barycentric coordinates

The three numbers we already have

13 Addresses on a slanted grid

about 5 min

A point is $\mathbf{p}_0 + \beta(\mathbf{p}_1 - \mathbf{p}_0) + \gamma(\mathbf{p}_2 - \mathbf{p}_0) = \alpha\mathbf{p}_0 + \beta\mathbf{p}_1 + \gamma\mathbf{p}_2$ *with* $\alpha = 1 - \beta - \gamma$.



$\mathbf{p}_0 = (1,1)$, $\mathbf{p}_1 = (9,3)$, $\mathbf{p}_2 = (3,6)$. The grey lines are quarter steps along $\mathbf{p}_1 - \mathbf{p}_0$ and along $\mathbf{p}_2 - \mathbf{p}_0$.

point	β	γ	α	inside, edge, corner, outside?
A	0.25	0.25	0.5	inside
B	___	___	___	_____
C	___	___	___	_____
D	___	___	___	_____
E	___	___	___	_____
F	___	___	___	_____

(a) Count the steps to each point and fill the table. Tick how a point's three numbers tell you where it is.

(b) Find F's (x, y) from $\mathbf{p}_0 + \beta(\mathbf{p}_1 - \mathbf{p}_0) + \gamma(\mathbf{p}_2 - \mathbf{p}_0)$, then again from $\alpha\mathbf{p}_0 + \beta\mathbf{p}_1 + \gamma\mathbf{p}_2$.

(c) The balance point of the triangle has $\alpha = \beta = \gamma$. What are they, and where is the point?

(d) Draw all points with $\beta = 0$. Then all points with $\beta = 0.5$. How do the two lines sit against the edge $\mathbf{p}_2\mathbf{p}_0$?

14 From edge values to depth and colour

about 7 min

Divide each edge value by twice the area; the three results weight the corners' depths and colours.

Back to T ($2A = 32$). The corners carry: $\mathbf{p}_0$: depth 0.8, red $(1, 0, 0)$; $\mathbf{p}_1$: depth 0.4, blue $(0, 0, 1)$; $\mathbf{p}_2$: depth 0.6, white $(1, 1, 1)$.

pixel	$f_{12},\ f_{20},\ f_{01}$	$\alpha,\ \beta,\ \gamma$	sum	depth	colour
$(6, 3)$	8, 20, 4	0.25, 0.625, 0.125	1	0.525	(0.375, 0.125, 0.75)
$(3, 4)$					
$(4, 3)$	12, 10, 10				
$(2, 3)$	16, 0, 16				
$(7, 2)$	12, 26, −6				
the point $\mathbf{p}_1$					

(a) Fill the table. For $(3, 4)$, work out the edge values first.

(b) With these three colours the blend is always $(\alpha + \gamma,\ \gamma,\ \beta + \gamma)$. Why?

(c) Pixel $(7, 2)$ is not covered (Exercise 8). What gives it away in its barycentric coordinates? And in its "colour"?

(d) A classmate takes α from f_{01} and γ from f_{12} by mistake. What depth do they get at $(6, 3)$? Which edge must α, the share of $\mathbf{p}_0$, come from?

(e) Every covered pixel's depth lies between 0.4 and 0.8. Why?

(f) One step right changes f_{12}, f_{20}, f_{01} by -2, $+5$, -3. By how much does the depth change? Check with pixel $(7, 3)$.

Part 4 · The depth buffer

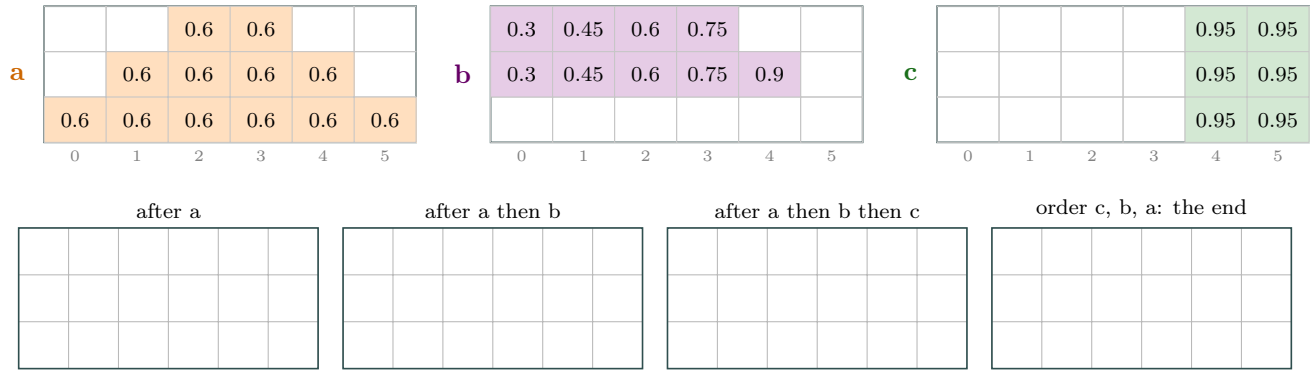
Keep the nearest depth per pixel

15 Be the depth circuit

about 8 min

For each fragment: if its depth is less than the stored one, write its depth and its colour; otherwise throw it away.

Three triangles a, b and c on a 6×3 screen. Each grid shows the depth of every fragment the triangle makes (empty: not covered).



(a) The buffer starts cleared to 1. Draw a, then b, then c. In each empty grid write the stored depth and the letter of the triangle that now owns the pixel (a dot for the background).

(b) Now draw c, then b, then a, and fill the last grid. Which cells differ from the first order? Why there? Who would own them in the order a, b, c if the test were "less or equal"?

(c) For each order: how many times did the fragment program run? How many fragments were written? Which order wrote less, and what drawing order would make the GPU write the least?

16 Settings gone wrong

about 4 min

Predict the picture from the setting. Some symptoms fit more than one setting.

Each row changes one thing in a working program that draws a room with a ball, every frame. Write the letter of what you would see. A letter may be used more than once.

	change	you see
1	The depth buffer is cleared to 0 instead of 1.	____
2	The depth buffer is never cleared; the picture is cleared every frame.	____
3	The test is "greater" instead of "less"; the buffer is still cleared to 1.	____
4	The test is "greater" and the buffer is cleared to 0.	____
5	A big sky square is drawn at depth exactly 1, with "less" and a clear to 1.	____
6	The depth test is switched off.	____

P Nothing at all is drawn, only the clear colour.

Q Whatever is drawn last is on top, near or far.

R The farthest surface in each pixel shows: you see the back wall through the ball.

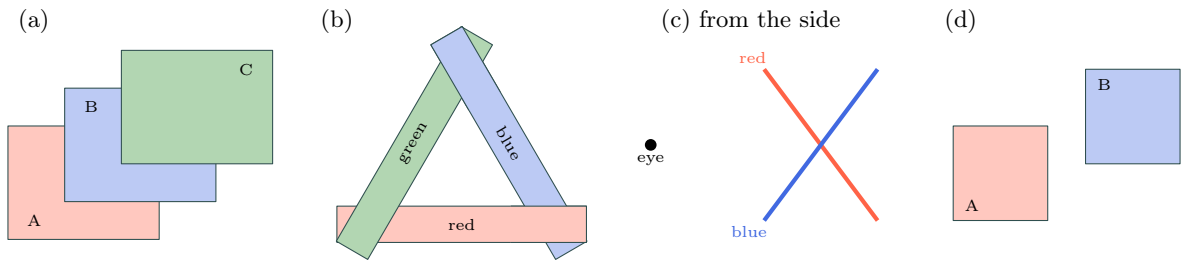
S Everything is right except that one surface never appears.

T The first frame is right; after it, almost nothing appears.

17 When the order matters

about 6 min

Far to near works only when an order exists; see-through surfaces need one even with a depth buffer.



(a) For each scene, write an order that paints far to near correctly (the painter's algorithm), or "none". For (b), say who covers whom.

(b) Which of the four scenes does the depth buffer draw correctly, in any order?

Now glass. A grey wall $(0.4, 0.4, 0.4)$ at depth 0.9; a blue pane $(0.2, 0.4, 1.0)$ at depth 0.5 in front of it; the background is white $(1, 1, 1)$. A pane is blended half and half: new $= 0.5\times$ pane $+0.5\times$ what is already in the pixel. Panes write their depth, like everything else.

(c) One pixel behind the pane: wall first, then the pane. What colour?

(d) Pane first, then the wall. What colour, and what does the viewer see?

(e) A red pane $(1, 0.2, 0.2)$ is added at depth 0.3. Draw solids first, then panes far to near: what colour? And if the red pane comes before the blue one?

18 Depth in steps

about 4 min

A depth buffer keeps only so many different values. Two depths in the same step look equal, and ties go to whoever was drawn first.

A toy buffer with 16 steps stores $\lfloor 16z \rfloor$ (so 0.50 and 0.55 are both step 8). A poster sits on a slanted wall, 0.02 nearer. The wall is drawn first; the test is "less" on the stored steps.

pixel	1	2	3	4	5	6
wall depth	0.50	0.54	0.58	0.62	0.66	0.70
wall step	8	___	___	___	___	___
poster depth	0.48	0.52	0.56	0.60	0.64	0.68
poster step	7	___	___	___	___	___
shows (W or P)	P	___	___	___	___	___

(a) Fill the table. The poster is nearer everywhere; what does the viewer see along the row?

(b) Lift the poster: make every poster depth $1/16 = 0.0625$ smaller. Fill its steps again. What shows now?

(c) A real buffer has 24 bits per depth. How many steps is that? How many bytes does a 1920×1080 depth buffer need at 2, 3 and 4 bytes per pixel?

19 The same answer, loops swapped

about 3 min

The depth buffer is lecture 15's "closest so far", kept for every pixel at once.

```
for each pixel:                    # lecture 15
    closest = inf
    for each triangle:
        t = hit(ray_through(pixel), triangle)
        if t < closest:
            closest = t
            colour = shade(...)
    picture[pixel] = colour
```

```
depth = [[1] * W for j in range(H)]   # today
for each triangle:
    for each covered pixel (i, j):
        if z < depth[j][i]:
            depth[j][i] = z
            picture[j][i] = c
```

(a) Which variable on the left does `depth[j][i]` stand in for? Which value on the left plays the part of the clear to 1?

(b) 1000 triangles, each with a box of 100 pixels, on a 1920×1080 screen. How many inside tests does each loop make (no speed-up tricks on the left)?

(c) How many "closest so far" numbers does each loop keep at one time?

(d) Both loops find the same nearest surface at every pixel centre. Name one kind of surface for which the right loop alone is not enough.

20 ★ Bonus for the brave: the top-left rule in code

about 10 min

Turn Exercise 12's rule into one small function, then check it on every case of this sheet.

The lecture's loop tests `f >= 0` for every edge. The rule wants `f > 0`, or `f == 0` on an edge the triangle *owns*: an edge where rows enter, or a flat top edge.

(a) Corners go counter-clockwise and y is up, so the inside is on the left of every edge's walk. Sketch an edge walked straight down, and one walked straight up. On which side is the inside of each? Through which one does a row, walking left to right, *enter*?

(b) So which sign of `end.y - start.y` marks an edge where rows enter? Which way is a flat *top* edge walked: towards larger or smaller x?

(c) Write `owns(start, end)` that returns `True` for exactly those edges, and the new test for one edge value `f`.

(d) Check it: T keeps only $(2, 3)$ of its four zeros; L owns the diagonal of square (1); Tb owns the flat edge of panel (2). For each, write which of the triangle's edges `owns` accepts.

(e) A friend writes `end.y <= start.y` for the first part, to catch flat edges too. Which case of Exercise 12 breaks, and how?

Try it tonight: a rasterizer in thirty lines. Type the lecture's `edge` and `rasterize` into one file. Instead of `fragment`, put `#` into a 10×8 grid of dots, and print the rows from the top. Draw T: you should count 18 `#` (Exercise 9). Add your `owns` from Exercise 20: now 15. Then add a depth buffer and the barycentric depth, draw the lecture's worked pair, red and blue, that cross in depth, and print each pixel's owner in both orders. Run it with `pypy3` (or `python3`).

Answer Key

Check your work *after* attempting everything. If an answer surprises you, that item is worth re-reading in the slides.

1a. 1 application, 2 vertex processing, 3 clipping and viewport, 4 rasterization, 5 fragment processing, 6 depth test and blend, 7 framebuffer. The cards from left to right: 6, 4, 7, 1, 3, 5, 2.

1b. Vertex processing (the vertex program) and fragment processing (the fragment program). The application runs on the CPU.

1c. 1→2 corners in a buffer; 2→3 four numbers (x, y, z, w) per corner; 3→4 corners in pixels, with their depths; 4→5 fragments; 5→6 one colour per fragment; 6→7 the winning colours.

1d. (i) 2; (ii) 5; (iii) 6; (iv) 3 (clipping); (v) 3 (the divide is a circuit after the program, after clipping); (vi) 2: after the vertex program every stop works in the box.

2a. Square: 6, 6, 10,000. Disc: 60, 60, 1,250. Both: 66, 66, 11,250: the disc's fragments are thrown away only at stop 6, *after* its fragment program has run. Full screen: 6, 6, 2,073,600.

2b. Square first: 10,000 written (the disc's 1,250 all lose the test). Disc first: 11,250 (the disc is written, then overwritten by the square).

2c. $60 \times 6 \times 4 = 1{,}440$ bytes, copied once; each frame the CPU only says "draw these 60 vertices".

2d. $2{,}073{,}600 \times 60 = 124{,}416{,}000$ fragment runs against 360 vertex runs a second. Keep the fragment program short.

3a. B: 2; 1.2, 0, 0.75; no ($x > 1$). C: 4; 0, 0.25, 1.125; no ($z > 1$). D: 0.5; 0.8, 0.4, -1.5; no ($z < 0$). E: -2; -0.8, -0.5, 2.25; no (and the numbers are nonsense). F: 0; cannot divide by 0.

3b. (i) B, right of the screen; (ii) C; (iii) D; (iv) E; (v) F.

3c. Dividing by a negative w flips the signs of x and y (E jumps to the left and down) and sends z past the far plane (2.25), although E is behind the eye, not far away.

3d. D, E and F (distance 0.5, -2, 0, all less than 1). A divide first would flip E and fail at F, and the triangle would smear across the screen. Clipping cuts at $D = 1$, where $w = 1$, so every corner left has $w \geq 1$. B is only off the screen: the rasterizer skips those pixels.

4a. (a) 0, 3, 1; (b) 1, 4, 2; (c) 2, 3, 1; (d) 3, 0, 0.

4b. A to C: $D = 3 + t(-1 - 3) = 1$ gives $t = 0.5$, the cut point $(1, 0)$. B to C: $t = 0.5$, $(1, -1.5)$. The four corners A, B, $(1, -1.5)$, $(1, 0)$ split along a diagonal into two triangles, for example A, B, $(1, -1.5)$ and A, $(1, -1.5)$, $(1, 0)$. Panel (c): from $(4, 2)$ to $(0, 3)$: $t = 3/4$, $(1, 2.75)$. From $(4, 2)$ to $(-2, 1)$: $t = 3/6 = 0.5$, $(1, 1.5)$. One triangle: $(4, 2)$, $(1, 2.75)$, $(1, 1.5)$.

4c. 1, 2, 1, 0. On the plane: nothing to cut: 3 corners, 1 triangle, unchanged. Its corner on the plane has $w = 1$, which divides safely.

5a. $(-1, -1)$: $(0, 0)$, row 8, the screen's bottom-left corner. $(1, 1)$: $(16, 8)$, row 0, the top-right corner. $(0, 0)$: $(8, 4)$, row 4, the corner shared by pixels $(7, 3)$, $(8, 3)$, $(7, 4)$, $(8, 4)$. $(-0.4375, 0.625)$: $(4.5, 6.5)$, row 1.5, the centre of pixel $(4, 6)$. $(1.25, 0)$: $(18, 4)$, off the screen.

5b. None: $(16, 8)$ is a corner of the screen, and the columns are 0 to 15. The box's edge is the screen's edge, never a centre. $(0, 0)$ on 1920×1080: $(960, 540)$: the corner of four pixels, not a centre (true on every screen of even size).

5c. $x_n = 11.5/8 - 1 = 0.4375$, $y_n = 2.5/4 - 1 = -0.375$.

5d. No: only on-screen centres are tested. A triangle reaching off the screen is not cut there, its off-screen pixels are simply skipped.

6a. B: $(5, 0)$, $0 - 15 = -15$. C: $(4, 2)$, $12 - 12 = 0$. D: $(8, 4)$, $24 - 24 = 0$. E: $(2, -1)$, $-6 - 6 = -12$. F: $(3, 4)$, $24 - 9 = 15$.

6b. The side of the whole *line* through **s** and **e**: D lies on that line, past **e**. That is why one edge alone never decides; the other two edges reject D.

6c. The line AF is parallel to the edge. f changes evenly as you move away from the line, so every point on a parallel line has the same value: all points with $f = 15$ are on line AF.

6d. $f = -3x + 6y - 9$. At E: $-9 + 6 - 9 = -12$.

6e. Every sign flips: $-15, 15, 0, 0, 12, -15$. The left of $\mathbf{e} \to \mathbf{s}$ is the right of $\mathbf{s} \to \mathbf{e}$.

7a. f_{20}: $6 - 1 = 5$, $2 - 3 = -1$, $3 \cdot 1 - 2 \cdot 6 = -9$: $5x - y - 9$. f_{01}: $1 - 4 = -3$, $9 - 2 = 7$, $2 \cdot 4 - 9 \cdot 1 = -1$: $-3x + 7y - 1$.

7b. For example $f_{12}(9, 4) = -18 - 24 + 42 = 0$ and $f_{12}(3, 6) = -6 - 36 + 42 = 0$; the same for the other four.

7c. $f_{12}(2, 1) = -4 - 6 + 42 = 32$; $f_{20}(9, 4) = 45 - 4 - 9 = 32$; $f_{01}(3, 6) = -9 + 42 - 1 = 32$. Each is the determinant of two sides of T, twice its area: $2A = 32$. Positive: the corners are counter-clockwise; clockwise ones would give -32.

7d. $35 - (7 \cdot 3/2 + 6 \cdot 2/2 + 1 \cdot 5/2) = 35 - (10.5 + 6 + 2.5) = 16$, and $2 \times 16 = 32$.

8a. $(7, 2)$: $(7.5, 2.5)$; 12, 26, -6; no. $(2, 3)$: $(2.5, 3.5)$; 16, 0, 16; yes (≥ 0). $(1, 6)$: $(1.5, 6.5)$; 0, -8, 40; no.

8b. Exactly on the edge $\mathbf{p}_2\mathbf{p}_0$ ($f_{20} = 0$): that edge passes through $(2.5, 3.5)$.

8c. No. It is on the *line* through $\mathbf{p}_1$ and $\mathbf{p}_2$, beyond $\mathbf{p}_2$ (Exercise 6(b) again), and $f_{20} = -8$ puts it outside.

8d. `range(2, 9)`: columns 2 to 8; `range(1, 6)`: rows 1 to 5. $7 \times 5 = 35$ tests.

9a. Right: $y_0 - y_1 = -3$. Up: $x_1 - x_0 = 7$.

9b. Rows from the top: row 5: $30, 27, 24, 21, 18, 15, 12$ Check: $f_{01}(8.5, 5.5) = -25.5 + 38.5 - 1 = 12$.
row 4: $23, 20, 17, 14, 11, 8, 5$
row 3: $16, 13, 10, 7, 4, 1, -2$
row 2: $9, 6, 3, 0, -3, -6, -9$
row 1: $2, -1, -4, -7, -10, -13, -16$

9c. f_{12}: -2 right, -6 up; f_{20}: $+5$ right, -1 up. Both match $y_a - y_b$ and $x_b - x_a$.

9d. 18 pixels: row 1: column 2; row 2: columns 2–5; row 3: 2–7; row 4: 3–7; row 5: 3–4. Zeros at $(5, 2)$ (f_{01}), $(2, 3)$ (f_{20}), $(7, 4)$ and $(4, 5)$ (f_{12}).

9e. $35 - 18 = 17$, about half.

10a. 0 fragments, then 2 (pixels $(5, 2)$ and $(6, 2)$). The sliver lies between two rows of centres, so it vanishes; as it drifts it flickers on and off and breaks into dashes.

10b. Speck 1 contains the centre: 1 fragment, coloured for the whole pixel. Speck 2 misses it: 0. Only the centre counts, not the share of the square: a big share can give nothing, a tiny one a whole pixel.

10c. -12, -10, -10 in some order: every inside value is negative and $2A = -32$. No pixel passes: 0 fragments. (A triangle seen from behind looks clockwise; lecture 29 uses this.)

10d. $2A = 0$. Every edge function is 0 on the line and the three agree in sign nowhere else, so the centres on the line pass: $(0, 0)$, $(2, 1)$, $(4, 2)$, $(6, 3)$, four fragments of a triangle with no area. Part 3 divides by 0. Drop zero-area triangles before the loop.

10e. 4: T's covered pixels in columns 6 and 7, now $(10, 3)$, $(11, 3)$, $(10, 4)$, $(11, 4)$. Writing $(11, 3)$ indexes past the row's end: an error, or (in a flat array) a pixel of the next row. Clamp: columns `range(max(0, floor(min(xs))), min(W, ceil(max(xs))))`, rows the same with H: columns 6 to 9, $4 \times 5 = 20$ tests.

11a. Bottom row: part, part, part, out. Middle row: part, in (P), part, out (R). Top row: part, part, part, out. (The third block of the top and bottom rows is "part" with no centre covered: it must still be tested.)

11b. Q: all four f_{01} negative: skip. R: f_{12} $18, -2, -18, -38$; f_{20} $90, 130, 90, 130$; f_{01} $-18, -38, 18, -2$. No edge has four negatives: test all 16 (and find none).

11c. Bottom-left $(12, 4)$ is outside f_{01} only; top-left $(12, 8)$ outside f_{12} only; the right corners are outside both. Each edge has a corner on its inside, so no edge alone rejects R: the corner test is safe (never skips a covered pixel) but not perfect.

11d. The triangle's box: x runs only up to 12, and R starts at $x = 12$. Real rasterizers only visit blocks inside the box.

12a. Each row starts in U (left of the diagonal) and enters L at the diagonal. ≥ 0: 10, 10, 20; the diagonal is drawn twice (a darker seam on see-through surfaces). > 0: 6, 6, 12; a crack of 4 holes. Rule: 10, 6, 16; every pixel once.

12b. No row enters there: the row $y = 2.5$ runs along the edge. It is Tb's top edge (Tb lies below it), so Tb draws all four: Tt 2 fragments, Tb 6.

12c. Only $(2, 3)$, on $\mathbf{p}_2\mathbf{p}_0$, where rows enter T. $(5, 2)$ is on $\mathbf{p}_0\mathbf{p}_1$, the bottom edge, which rows never enter; $(7, 4)$ and $(4, 5)$ are on $\mathbf{p}_1\mathbf{p}_2$, where rows *leave* T. The triangles on the other sides draw those three. T makes 15 fragments.

13a. B: 0.5, 0.5, 0, on the edge $\mathbf{p}_1\mathbf{p}_2$. C: 0, 0.75, 0.25, on the edge $\mathbf{p}_0\mathbf{p}_2$. D: 1, 0, 0, the corner $\mathbf{p}_1$. E: 0.75, 0.5, -0.25, outside (one negative). F: 0.5, 0.25, 0.25, inside. All positive: inside; one zero: edge; two zeros: corner; any negative: outside.

13b. $(1, 1) + 0.5(8, 2) + 0.25(2, 5) = (1, 1) + (4, 1) + (0.5, 1.25) = (5.5, 3.25)$; $0.25(1, 1) + 0.5(9, 3) + 0.25(3, 6) = (0.25 + 4.5 + 0.75,\ 0.25 + 1.5 + 1.5) = (5.5, 3.25)$.

13c. 1/3 each: $\big((1 + 9 + 3)/3,\ (1 + 3 + 6)/3\big) = (4.33, 3.33)$.

13d. $\beta = 0$: the line through $\mathbf{p}_0$ and $\mathbf{p}_2$. $\beta = 0.5$: the parallel line half way to $\mathbf{p}_1$ (through B and F). Lines of equal β are parallel to the edge facing $\mathbf{p}_1$.

14a.

pixel	f	α, β, γ	depth	colour
$(3, 4)$	$8, 4, 20$	$0.25, 0.125, 0.625$	0.625	$(0.875, 0.625, 0.75)$
$(4, 3)$	$12, 10, 10$	$0.375, 0.3125, 0.3125$	0.6125	$(0.6875, 0.3125, 0.625)$
$(2, 3)$	$16, 0, 16$	$0.5, 0, 0.5$	0.7	$(1, 0.5, 0.5)$
$(7, 2)$	$12, 26, -6$	$0.375, 0.8125, -0.1875$	0.5125	$(0.1875, -0.1875, 0.625)$
$\mathbf{p}_1$	$0, 32, 0$	$0, 1, 0$	0.4	$(0, 0, 1)$

Every sum is 1. $(3, 4)$, centre $(3.5, 4.5)$: $f_{12} = -7 - 27 + 42 = 8$, $f_{20} = 17.5 - 4.5 - 9 = 4$, $f_{01} = -10.5 + 31.5 - 1 = 20$.

14b. Red: $\alpha \cdot 1 + \beta \cdot 0 + \gamma \cdot 1$; green: only white has green, γ; blue: $\beta + \gamma$.

14c. γ is negative (the point is past the edge facing $\mathbf{p}_2$); the sum is still 1. Its green is -0.1875, not a colour at all. The GPU never makes this fragment.

14d. $0.125 \times 0.8 + 0.625 \times 0.4 + 0.25 \times 0.6 = 0.5$ instead of 0.525: wrong, and nothing warns you. α comes from f_{12}, the edge that does not touch $\mathbf{p}_0$.

14e. Inside, the three weights are at least 0 and add to 1, so the blend is an average of 0.8, 0.4, 0.6: never below the smallest or above the largest.

14f. $(-2\times0.8+5\times0.4-3\times0.6)/32 = -1.4/32 = -0.04375$. $(7, 3)$: $6, 25, 1$, depth $0.15+0.3125+0.01875 = 0.48125 = 0.525 - 0.04375$.

15a. The clear value 1 is the far end of the box, "nothing drawn here yet", like lecture 15's $t = \infty$. Rows from the top, owner and depth:

after a:	row 2: · · a a · ·	$(1, 1, 0.6, 0.6, 1, 1)$
	row 1: · a a a a ·	$(1, 0.6, 0.6, 0.6, 0.6, 1)$
	row 0: a a a a a a	(all 0.6)
after b:	row 2: b b a a · ·	$(0.3, 0.45, 0.6, 0.6, 1, 1)$
	row 1: b b a a a ·	$(0.3, 0.45, 0.6, 0.6, 0.6, 1)$
after c:	row 2: b b a a c c	$(0.3, 0.45, 0.6, 0.6, 0.95, 0.95)$
	row 1: b b a a a c	$(0.3, 0.45, 0.6, 0.6, 0.6, 0.95)$

Row 0 stays all a, 0.6. b's 0.75 and 0.9 lose to a's 0.6; c wins only where nobody else is.

15b. Rows 2 and 1 become b b b a c c and b b b a a c. Only $(2, 1)$ and $(2, 2)$ differ: there a and b both have depth 0.6, a tie, and "less" keeps whoever came first. With "less or equal" the later of two equal depths wins: b.

15c. The fragment program runs $12 + 9 + 6 = 27$ times in both orders. Writes: a, b, c $= 12 + 4 + 3 = 19$; c, b, a $= 6 + 9 + 9 = 24$. Drawing near to far writes least: farther fragments then fail the test.

16. 1 P (no depth is less than 0). 2 T (in frame 2 every still surface has exactly the depth already stored, and "less" fails on a tie; only things that come nearer than ever before show). 3 P (no depth is greater than 1). 4 R. 5 S ($1 < 1$ is false: the sky never wins; draw it with "less or equal" or just inside the far plane). 6 Q.

17a. (a) A, B, C. (b) None: red covers blue, blue covers green, green covers red, a cycle. (c) None: red is nearer at the top, blue at the bottom; whole triangles cannot be ordered. (d) Any order.

17b. All four: it decides pixel by pixel.

17c. $0.5(0.2, 0.4, 1.0) + 0.5(0.4, 0.4, 0.4) = (0.3, 0.4, 0.7)$.

17d. The pane blends with white: $(0.6, 0.7, 1.0)$, and writes depth 0.5. The wall's 0.9 then fails the test: a hole in the wall, a window onto nothing.

17e. Wall, blue, red: $(0.3, 0.4, 0.7)$, then $0.5(1, 0.2, 0.2) + 0.5(0.3, 0.4, 0.7) = (0.65, 0.3, 0.45)$. Red before blue: red gives $(0.7, 0.3, 0.3)$ and writes 0.3; blue's 0.5 fails: the blue pane is missing.

18a. Wall steps $8, 8, 9, 9, 10, 11$; poster $7, 8, 8, 9, 10, 10$. Shows P W P W W P: the poster breaks into speckles and bands, which crawl when the camera moves (z-fighting).

18b. Depths $0.4175, 0.4575, 0.4975, 0.5375, 0.5775, 0.6175$: steps $6, 7, 7, 8, 9, 9$, always below the wall's. P everywhere.

18c. $2^{24} = 16{,}777{,}216$ steps. 2,073,600 pixels: 4,147,200, 6,220,800 and 8,294,400 bytes (about 4, 6 and 8 MB).

19a. `closest`; the start value `inf`.

19b. Left: $2{,}073{,}600 \times 1000 = 2{,}073{,}600{,}000$. Right: $1000 \times 100 = 100{,}000$.

19c. Left: one at a time. Right: 2,073,600, one per pixel, for the whole frame.

19d. See-through surfaces (glass, Exercise 17): the buffer keeps one surface per pixel, but a pane needs what is behind it too.

20a. Walking down, your left hand points to $+x$: the inside is to the right of the edge, so a row walking right enters through it. Walking up, the inside is to the left: rows leave through it.

20b. Rows enter where `end.y < start.y`. A flat edge has the inside on the walker's left; for the inside to be below, the edge is walked towards smaller x: `end.x < start.x`.

20c. `owns = end.y < start.y or (end.y == start.y and end.x < start.x)`; the test for one edge: `f > 0 or (f == 0 and owns(start, end))`.

20d. T: only $\mathbf{p}_2 \to \mathbf{p}_0$ ($6 \to 1$, down) is owned, so only $(2, 3)$ is kept: 15. L: $(5, 4) \to (1, 0)$, the diagonal walked down, is owned. U walks it up, $(1, 0) \to (5, 4)$: not owned. Tb: $(5, 2.5) \to (1, 2.5)$, flat towards smaller x: owned (top edge). Tt walks it $(1, 2.5) \to (5, 2.5)$: not owned.

20e. With `<=`, every flat edge is owned, bottom ones too: Tt would also own the shared flat edge of panel (2), and those four pixels would be drawn twice.