Finger Exercises 10

Rays, Intervals of t, and the Nearest Hit

≈ 55 minutes, plus a 10-minute bonus

CS 453 — Computer Graphics

How to do it: On **paper** first, no computer, squared paper beside you. Sketch the ray *before* you compute. Conventions are the lecture's: x right, y up, bold letters are points and arrows, plain letters are single numbers. If an item takes longer than 5 minutes, skip it, finish the rest, come back. Nothing here is an A3 task — A3 has no ray in it — so this sheet is Lecture 10 and nothing else.

Short on time? Do Parts C, D, F, H and J first (about 25 minutes). They are the spine of the lecture: read a ray, write a ray, know when t is a distance, pick the nearest hit, and run the loop.

Part A • Do You Remember?

RECALL ≈ 6 min

A1. The formula for every line, segment and ray in this course is $\mathbf{p}(t) =$ _____. In words: a _____, plus a _____ times an _____.

A2. Through two points: $\mathbf{p}(t) =$ _____. Then $t = 0$ lands on _____ and $t = 1$ lands on _____.

A3. Same formula, three objects. Line: $t \in$ _____. Segment: $t \in$ _____. Ray: $t \in$ _____.

A4. Going from $t = 0$ to $t = 1$ moves the bead _____ units. So t is a distance only when _____; then $\hat{\mathbf{d}} =$ _____.

A5. A viewing ray has $\mathbf{o} =$ _____ and $\mathbf{d} =$ _____. With that choice, $t = 1$ is exactly _____, and anything at $t < 0$ is _____.

A6. Of all the hits on one viewing ray, the pixel shows the one with the _____. The order the hits were found in matters _____.

A7. The three parts of a basic ray tracer are _____, _____ and _____. The middle one also returns _____, because the last one cannot work without it.

A8. Ray tracing is an _____-order algorithm: the outer loop runs over _____. The other kind loops over _____ instead.

A9. `Ray3` stores exactly _____ things: _____ and _____. Its one method `at(t)` returns _____.

A10. A circle is $(x_c +$ _____, $y_c +$ _____$)$, and an ellipse replaces the two r's by _____ and _____. Why does the book restrict ϕ to a *half-open* interval?

Part B • Arrows First

COMPUTE ≈ 4 min

Everything today is point-plus-arrow arithmetic from week two. Four quick ones to get the hands moving.

B1. $\mathbf{a} = (2, 1)$ and $\mathbf{b} = (6, 7)$. Compute $\mathbf{b} - \mathbf{a}$. Is the result a point or an arrow?

B2. Compute $(2, 1) + \frac{1}{2}(4, 6)$ and $(2, 1) + 2\,(4, 6)$.

B3. Find $|(4, -3)|$ and $|(-6, 8)|$, then write each as a unit arrow.

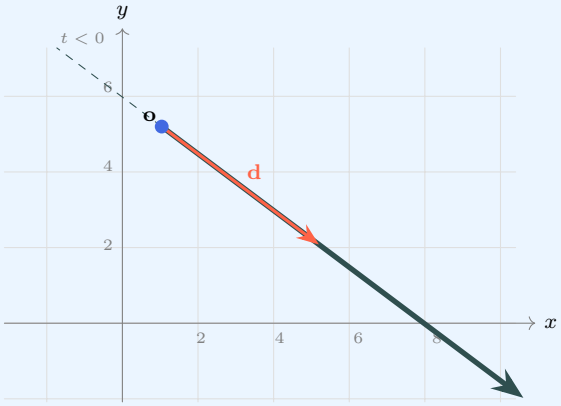
B4. $\mathbf{p}, \mathbf{q}$ are points and $\mathbf{d}$ is an arrow. Which of these are meaningful, and what kind of thing does each give? $\mathbf{p} + \mathbf{d}$, $\mathbf{p} - \mathbf{q}$, $\mathbf{p} + \mathbf{q}$, $3\mathbf{d}$, $3\mathbf{p}$.

Lecture 10 on one card check Part A against it, then keep it open

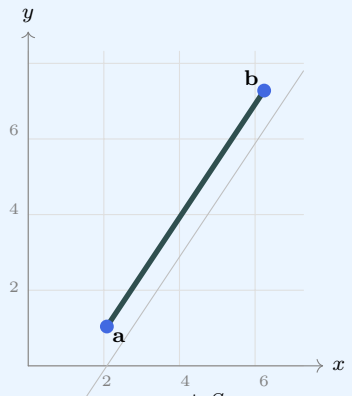
The formula: $\mathbf{p}(t) = \mathbf{o} + t\mathbf{d}$ — a point, plus a number times an arrow. $t = 0$ is $\mathbf{o}$; $t = 1$ is the tip of $\mathbf{d}$.
From two points: $\mathbf{p}(t) = \mathbf{p}_0 + t(\mathbf{p}_1 - \mathbf{p}_0)$, so $\mathbf{o} = \mathbf{p}_0$ and $\mathbf{d} = \mathbf{p}_1 - \mathbf{p}_0$. Between the two, t is the fraction of the way.
Three intervals: line $t \in \mathbb{R}$; segment $t \in [0, 1]$; ray $t \in [0, \infty)$. The book calls all three "rays".
Distance: from $\mathbf{o}$ it is $t\,|\mathbf{d}|$. With $\hat{\mathbf{d}} = \mathbf{d}/|\mathbf{d}|$, t *is* the distance. A line has one speed $|\mathbf{d}|$; a curve speeds up and slows down.
Curves: $\mathbf{p} = \mathbf{f}(t)$, one slot function per coordinate; circle $(x_c + r\cos\phi,\ y_c + r\sin\phi)$; ellipse uses a, b; a domain D cuts a piece out.

Viewing ray: $\mathbf{o}$ = the eye, $\mathbf{d}$ = eye to pixel centre, so $t = 1$ is the pixel and $t < 0$ is behind the eye.
Nearest hit: among the hits, keep the smallest t with $t \geq 0$. Found-order is irrelevant.
Three parts:
generate: camera $\rightarrow \mathbf{o}, \mathbf{d}$
intersect: ray + objects $\rightarrow$ closest hit, $\mathbf{n}$
shade: hit, $\mathbf{n}$, lights $\rightarrow$ colour
The whole program: for each pixel — compute the viewing ray; find the first object hit and its normal $\mathbf{n}$; set the pixel colour from the hit point, the lights and $\mathbf{n}$.
`Ray3`: fields `origin` and `direction` (normalized on the way in); one method, `at(t)`, returning $\mathbf{o} + t\mathbf{d}$.

The two objects on this sheet. Parts C–F and the bonus keep coming back to them. Ray R was chosen so that $|\mathbf{d}| = 5$ exactly; segment S was not.



ray R
$\mathbf{o} = (1, 5)$, $\mathbf{d} = (4, -3)$
$|\mathbf{d}| = 5$, $t \geq 0$



segment S
$\mathbf{a} = (2, 1)$, $\mathbf{b} = (6, 7)$
$\mathbf{b} - \mathbf{a} = (4, 6)$, $t \in [0, 1]$

Part C • Walk the Ray by Counting

TRACE ≈ 5 min

No formula for this part. Put a finger on $\mathbf{o} = (1, 5)$ and count: **one copy of d is 4 right and 3 down**. Half a copy is 2 right and 1.5 down; a negative t walks the arrow backwards.

C1. Count your way to $\mathbf{p}(1)$, $\mathbf{p}(2)$ and $\mathbf{p}(3)$.

C2. Now $\mathbf{p}(0.5)$ and $\mathbf{p}(0.25)$. Are both on the ray?

C3. $\mathbf{p}(-1)$. Is it on the ray? Is it on the line?

C4. Which t puts the bead on $(13, -4)$? And on $(3, 3.5)$? Read them off, then check one slot with arithmetic.

C5. A classmate says $\mathbf{p}(2) = (3, 7)$. What did they do, and what is the one-sentence correction?

Part D • Two Points, One Formula

COMPUTE ≈ 4 min

Segment S: $\mathbf{a} = (2, 1)$, $\mathbf{b} = (6, 7)$.

D1. Write $\mathbf{p}(t)$ for S with the numbers in, then evaluate at $t = 0$ and $t = 1$ to check the ends.

D2. Find $\mathbf{p}(\frac{1}{2})$ and $\mathbf{p}(\frac{1}{4})$. What fraction of the trip is each?

D3. Rewrite S's line in $\mathbf{o} + t\mathbf{d}$ form. What are $\mathbf{o}$ and $\mathbf{d}$, and what is $|\mathbf{d}|$? Is t a distance here?

D4. Which t lands on $(10, 13)$, and is that point on the segment? Is $(5, 6)$ on the line at all?

Part E • Line, Segment, or Ray?

CLASSIFY ≈ 4 min

Still S: $\mathbf{p}(t) = (2, 1) + t(4, 6)$. "The ray from $\mathbf{a}$" means the half-line with $t \geq 0$.

E1. Tick every box that the point $\mathbf{p}(t)$ belongs to.

	on the line	on the segment	on the ray from a
$t = -0.5$	☐	☐	☐
$t = 0$	☐	☐	☐
$t = 0.3$	☐	☐	☐
$t = 1$	☐	☐	☐
$t = 1.4$	☐	☐	☐

E2. The point $(0, -2)$ sits on S's line. Find its t, then say which of the three it belongs to.

E3. Name the interval of t for each: the whole line; the segment; the ray from $\mathbf{a}$; the part of that ray strictly beyond $\mathbf{b}$.

E4. Write the ray that starts at $\mathbf{b}$ and heads toward $\mathbf{a}$. Which t now lands on $\mathbf{a}$, and which t values give the same segment S?

Part F • Is t a Distance?

COMPUTE ≈ 5 min

Back to ray R: $\mathbf{o} = (1, 5)$, $\mathbf{d} = (4, -3)$.

F1. Find $|\mathbf{d}|$. How far from $\mathbf{o}$ is $\mathbf{p}(1)$? And $\mathbf{p}(2)$? And $\mathbf{p}(0.4)$?

F2. Write $\hat{\mathbf{d}}$ and then the same ray with a unit direction, using s for its parameter. Which s is the same place as $t = 1$? Give the rule linking s and t.

F3. Which t puts the bead exactly 3 units from $\mathbf{o}$? Give the point, and check its distance.

F4. A second ray R' has the same origin $(1, 5)$ but $\mathbf{d}' = (8, -6)$. Do R and R' draw the same set of points? At which t does each reach $(5, 2)$? What does that say about comparing t values from two different rays?

Part G • One More Slot: 3D

COMPUTE ≈ 4 min

A 3D line, written slot by slot:

$$x = 4 - 2t, \qquad y = -1 + 3t, \qquad z = 6 + t.$$

G1. Read off $\mathbf{o}$ and $\mathbf{d}$.

G2. Find $\mathbf{p}(3)$, showing the two moves: scale the arrow, then add slot by slot.

G3. Is $(0, 5, 8)$ on this line? Is $(2, 2, 8)$? Use the x slot to get a candidate t, then check the others.

G4. The segment from $\mathbf{a} = (1, 0, 2)$ to $\mathbf{b} = (5, 4, -2)$: write it, give its midpoint, and give the point three quarters of the way from $\mathbf{a}$ to $\mathbf{b}$.

Part H • Nearest Hit Wins

TRACE ≈ 4 min

One viewing ray per item. The intersection code hands back a list of t values, in the order the objects happen to be stored. Say which hit the pixel shows, or that it shows none.

H1. Hits at $t = 6.4,\ 2.5,\ 9.0$. Which is seen, and what is the hit point in terms of $\mathbf{o}$ and $\mathbf{d}$?

H2. Hits at $t = -2.0,\ 4.5,\ 7.1$. Which is seen? Why is the smallest number the wrong answer here?

H3. Hits at $t = -5.0$ and $t = -1.2$, and nothing else. What goes in the pixel?

H4. A classmate's code loops over the objects and returns the first one whose test succeeds. On the item H1 list it returns $t = 6.4$. What appears on the screen, and what is the fix in one line?

H5. A ray starts *inside* a ball. The ball reports hits at $t = -1.5$ and $t = 3.0$. Which is seen, and what surface are you looking at?

Part I • Which Box Does It?

CLASSIFY ≈ 4 min

The three boxes are **generate** (camera → $\mathbf{o}, \mathbf{d}$), **intersect** (ray + objects → closest hit and $\mathbf{n}$) and **shade** (hit, $\mathbf{n}$, lights → colour). Tick *every* box that touches each quantity — two rows need two ticks.

	generate	intersect	shade
where the eye is	☐	☐	☐
how many pixels across the image	☐	☐	☐
the arrow from the eye through a pixel centre	☐	☐	☐
the list of objects in the scene	☐	☐	☐
comparing two t values	☐	☐	☐
deciding that T_1 is hidden behind T_2	☐	☐	☐
the hit point $\mathbf{o} + t\mathbf{d}$	☐	☐	☐
the surface normal $\mathbf{n}$	☐	☐	☐
where the lights are	☐	☐	☐
the colour that goes into the pixel	☐	☐	☐

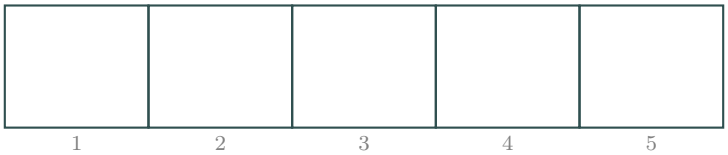
Part J • The Loop on Paper

TRACE ≈ 5 min

A one-row image, five pixels wide. The scene has a red ball, a green wall and a blue triangle; anything not hit is grey. Here is what intersection reports for each pixel's viewing ray:

pixel	hits reported, in found-order
1	wall $t = 6.0$
2	triangle $t = 4.2$, wall $t = 6.5$
3	ball $t = -1.0$, wall $t = 5.8$
4	(none)
5	ball $t = 2.6$, triangle $t = 2.1$, wall $t = 6.9$

Fill in the strip, one word per box:



J1. Fill the five boxes.

J2. How many viewing rays did this row need? How many of them found nothing? Which pixel would change colour if the scene list were reordered under a buggy first-hit-wins loop?

J3. Pixel 4 is grey. Which of the three boxes did the work for it, and which did almost none?

Part K • Find the Bug

FIX ≈ 4 min

Each snippet is one small change away from right. Name the bug and give the fix.

K1. Symptom: every point on the ray comes back the same, and it is always the tip of **d**.

```
def at(self, t: float) -> Vec3:
    return self.origin + self.direction
```

Bug & fix: _____

K2. Symptom: objects behind the camera are drawn, and they are drawn in front of everything else.

```
best_t = min(hits)
```

Bug & fix: _____

K3. Symptom: an object appears or disappears depending on the order the scene was built in.

```
for obj in objects:
    t = obj.hit(ray)
    if t is not None:
        return obj.colour
```

Bug & fix: _____

K4. Symptom: a report of "distance to the nearest object" is off by a constant factor for every pixel.

```
self.direction = direction
...
distance = best_t
```

Bug & fix: _____

K5. Symptom: the render shows whatever is behind the camera, and moving an object nearer makes it vanish.

```
ray = Ray3(pixel_centre, eye - pixel_centre)
```

Bug & fix: _____

Part L • One Sentence Each

EXPLAIN ≈ 4 min

L1. Why is a ray's interval closed at 0 but open at infinity?

L2. Two hits on the same ray. Why does "nearer to the eye" mean "smaller t", with no distance calculation at all?

L3. Why does the book call lines, segments and rays all "rays"?

L4. Why is it safe to normalize $\mathbf{d}$, even though it changes every t?

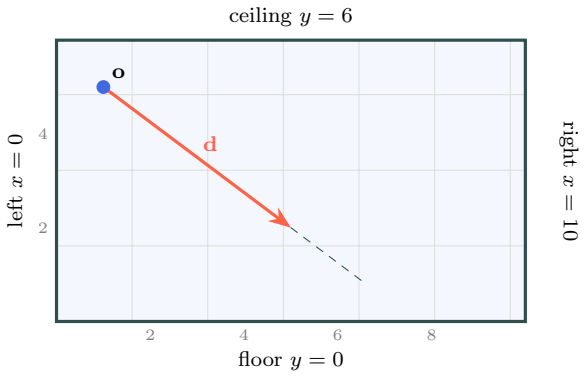
L5. Why is ray tracing so easy to run on many cores at once?

L6. Why does the intersection box return the normal $\mathbf{n}$ rather than leaving shading to look it up?

Part M • Bonus: The First Wall ★

DESIGN ≈ 10 min

For the brave. Wednesday's lecture asks a ray what it hits; here is the simplest possible version of that question, and you already have everything you need. The room is bounded by four lines: the floor $y = 0$, the ceiling $y = 6$, the left wall $x = 0$ and the right wall $x = 10$.



M1. Ray R again: $\mathbf{o} = (1, 5)$, $\mathbf{d} = (4, -3)$. For each of the four walls, set the matching slot of $\mathbf{o} + t\mathbf{d}$ equal to the wall's value and solve for t. You should get four numbers.

M2. Apply the rule from Part H: throw away what cannot be seen, keep the winner, and give the hit point. Which wall does the ray meet first?

M3. Now aim a second ray along $\mathbf{d} = (4, 0)$ from the same $\mathbf{o}$. Redo the four equations. What goes wrong, what does it mean geometrically, and what must the code check before it divides?

M4. Replace $\mathbf{d} = (4, -3)$ by $2\mathbf{d} = (8, -6)$, then by $-\mathbf{d} = (-4, 3)$. In each case, does the *set* of points change? Does the winning t change? Does the wall that is hit change?

Stuck on more than three items? Re-read the Lecture 10 slides — everything above Part M comes straight from them. Start with "t is the fraction of the way", "Drop $\mathbf{p}_1$: an origin and a direction", "Unit $\mathbf{d}$: t measures distance", "Nearest hit wins: the smallest t" and "Three parts: generate, intersect, shade". Most items here are one of those five slides with new numbers.

Answer Key

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

A1. $\mathbf{p}(t) = \mathbf{o} + t\mathbf{d}$. A **point**, plus a **number** times an **arrow**. The answer is a point.

A2. $\mathbf{p}(t) = \mathbf{p}_0 + t(\mathbf{p}_1 - \mathbf{p}_0)$. $t = 0$ gives $\mathbf{p}_0$ (zero times the arrow is the zero arrow); $t = 1$ gives $\mathbf{p}_1$ (one whole copy of the arrow).

A3. Line: $t \in \mathbb{R}$, every number. Segment: $t \in [0, 1]$. Ray: $t \in [0, \infty)$, half-open — 0 is in, infinity is not a number we reach.

A4. $|\mathbf{d}|$ units. t is a distance only when $|\mathbf{d}| = 1$; then $\hat{\mathbf{d}} = \mathbf{d}/|\mathbf{d}|$. In general, distance from $\mathbf{o}$ is $t\,|\mathbf{d}|$.

A5. $\mathbf{o}$ is the **eye**; $\mathbf{d}$ is the arrow **from the eye through the pixel's centre**. Then $t = 1$ is the pixel itself, and $t < 0$ is **behind the eye**, never seen.

A6. The **smallest** t **among the hits with** $t \geq 0$. The order they were found in matters **not at all** — only the t values do.

A7. **Ray generation**, **ray intersection**, **shading**. Intersection also returns the surface normal $\mathbf{n}$ at the hit point; how bright a surface looks depends on the angle between $\mathbf{n}$ and the light.

A8. **Image**-order: the outer loop runs over **pixels**. Object-order rendering loops over **objects** and works out which pixels each covers; that needs projection matrices and comes later in the course.

A9. Two things: `origin` (our $\mathbf{o}$) and `direction` (our $\mathbf{d}$, normalized as it is stored). `at(t)` returns `self.origin + self.direction * t`, that is $\mathbf{o} + t\mathbf{d}$ — one line.

A10. $(x_c + r\cos\phi,\ y_c + r\sin\phi)$; the ellipse uses a in the x slot and b in the y slot. Half-open, such as $[0, 2\pi)$, so that the seam point is counted once: ϕ and $\phi + 2\pi$ name the same place.

B1. $(4, 6)$, an **arrow**: point minus point is the trip from the first to the second. It has no place of its own, only a run and a rise.

B2. $(2, 1) + (2, 3) = (4, 4)$. And $(2, 1) + (8, 12) = (10, 13)$. Scale the arrow first, then add slot by slot.

B3. $|(4, -3)| = \sqrt{16 + 9} = 5$, unit $(0.8, -0.6)$. $|(-6, 8)| = \sqrt{36 + 64} = 10$, unit $(-0.6, 0.8)$. Both are 3-4-5 triangles in disguise.

B4. $\mathbf{p} + \mathbf{d}$: a **point**. $\mathbf{p} - \mathbf{q}$: an **arrow**. $\mathbf{p} + \mathbf{q}$: **meaningless** — move the world origin and the answer changes. $3\mathbf{d}$: an **arrow**, three times as long. $3\mathbf{p}$: **meaningless** as a place, same reason. This is exactly why t multiplies $\mathbf{d}$ and never $\mathbf{o}$.

C1. $\mathbf{p}(1) = (5, 2)$, $\mathbf{p}(2) = (9, -1)$, $\mathbf{p}(3) = (13, -4)$. Each step: 4 right, 3 down.

C2. $\mathbf{p}(0.5) = (3,\ 3.5)$ and $\mathbf{p}(0.25) = (2,\ 4.25)$. Both have $t \geq 0$, so yes, both are on the ray — between $\mathbf{o}$ and the tip of the arrow.

C3. $(1, 5) - (4, -3) = (-3,\ 8)$. **Not** on the ray: $t < 0$ is excluded, and in a renderer it sits behind the eye. It is on the *line*, the dashed part of the picture.

C4. $t = 3$ and $t = 0.5$. Check with the x slot: $1 + 4t = 13$ gives $t = 3$; the y slot must agree, $5 - 3(3) = -4$. It does.

C5. They added t to the point: $(1 + 2,\ 5 + 2)$. t **scales the arrow, never the point** — the correct step is $(1, 5) + 2(4, -3) = (9, -1)$.

D1. $\mathbf{p}(t) = (2, 1) + t\,(4, 6)$. $\mathbf{p}(0) = (2, 1) = \mathbf{a}$; $\mathbf{p}(1) = (2, 1) + (4, 6) = (6, 7) = \mathbf{b}$. Always check the ends first.

D2. $\mathbf{p}(\frac{1}{2}) = (2, 1) + (2, 3) = (4, 4)$, the midpoint. $\mathbf{p}(\frac{1}{4}) = (2, 1) + (1, 1.5) = (3,\ 2.5)$, a quarter of the way. Between the ends, t is the fraction of the trip.

D3. $\mathbf{o} = (2, 1)$, $\mathbf{d} = (4, 6)$; $|\mathbf{d}| = \sqrt{16 + 36} = \sqrt{52} = 2\sqrt{13} \approx 7.21$. **No** — $t = 1$ is about 7.21 units out, not one unit.

D4. $2 + 4t = 10$ gives $t = 2$; check $1 + 6(2) = 13$. On the *line*, but **not on the segment**: $t = 2 > 1$, past $\mathbf{b}$. $(5, 6)$: the x slot needs $t = 0.75$, but then $y = 1 + 4.5 = 5.5 \neq 6$, so it is **not on the line**. Both slots must agree on one t.

E1. Every row is on the **line** — the line is all of t. Segment $(0 \leq t \leq 1)$: $t = 0,\ 0.3,\ 1$ only. Ray $(t \geq 0)$: $t = 0,\ 0.3,\ 1,\ 1.4$. Only $t = -0.5$ fails two of the three.

E2. $2 + 4t = 0$ gives $t = -0.5$; check $1 - 3 = -2$. On the line only: behind $\mathbf{a}$, so off the segment and off the ray.

E3. $\mathbb{R}$; $[0, 1]$; $[0, \infty)$; $(1, \infty)$ — open at 1 because $\mathbf{b}$ itself is excluded by the word "beyond".

E4. $\mathbf{p}(t) = (6, 7) + t\,(-4, -6)$, since $\mathbf{a} - \mathbf{b} = (-4, -6)$. $t = 1$ lands on $\mathbf{a}$, and $t \in [0, 1]$ is the same segment traced the other way. The set of points is identical; the addresses are not.

F1. $|\mathbf{d}| = \sqrt{16 + 9} = 5$. Distance is $t\,|\mathbf{d}|$, so 5, 10 and 2 units. R was built to make this easy; most rays are not so tidy.

F2. $\hat{\mathbf{d}} = \frac{1}{5}(4, -3) = (0.8, -0.6)$, so $\mathbf{p}(s) = (1, 5) + s\,(0.8, -0.6)$. $t = 1$ is the point $(5, 2)$, which is $s = 5$. The rule is $s = t\,|\mathbf{d}| = 5t$: shrinking the arrow stretches the parameter by the same factor.

F3. $5t = 3$, so $t = 0.6$, and the point is $(1, 5) + (2.4, -1.8) = (3.4,\ 3.2)$. Check: the arrow from $\mathbf{o}$ is $(2.4, -1.8)$,

whose length is $\sqrt{5.76 + 3.24} = \sqrt{9} = 3$.

F4. **Same set of points**: $\mathbf{d}'$ is $2\mathbf{d}$, so the half-line is identical. R reaches $(5, 2)$ at $t = 1$; R' reaches it at $t = 0.5$. So a t is only meaningful *for its own ray.* Two rays can be compared directly only if both directions are unit length — which is exactly why `Ray3` normalizes in the constructor.

G1. $\mathbf{o} = (4, -1, 6)$, the three start numbers; $\mathbf{d} = (-2, 3, 1)$, the three steps. The minus sign belongs to the step, not to the start.

G2. $3\mathbf{d} = (-6, 9, 3)$; then $(4 - 6,\ -1 + 9,\ 6 + 3) = (-2,\ 8,\ 9)$.

G3. $(0, 5, 8)$: $4 - 2t = 0$ gives $t = 2$; then $y = -1 + 6 = 5$ and $z = 6 + 2 = 8$. **Yes**, at $t = 2$. $(2, 2, 8)$: $t = 1$ from x, and $y = 2$ agrees, but $z = 7 \neq 8$. **No** — all three slots must agree on one t.

G4. $\mathbf{p}(t) = (1, 0, 2) + t\,(4, 4, -4)$, $t \in [0, 1]$. Midpoint $\mathbf{p}(0.5) = (3, 2, 0)$, which is also $(\mathbf{a} + \mathbf{b})/2$. Three quarters: $\mathbf{p}(0.75) = (1 + 3,\ 0 + 3,\ 2 - 3) = (4, 3, -1)$.

H1. $t = 2.5$, the smallest. The hit point is $\mathbf{o} + 2.5\,\mathbf{d}$, which is exactly what `ray.at(2.5)` returns. Found-order is irrelevant.

H2. $t = 4.5$. The rule is the smallest t *among* $t \geq 0$: -2.0 is behind the eye, and a ray is half-open at 0, so it is not on the ray at all.

H3. Nothing in front of the eye was hit, so the pixel gets the **background colour**. "No hit" is a perfectly ordinary answer for a viewing ray and the loop must handle it.

H4. The far object is painted over the near one — objects appear or disappear depending on the order they were added to the scene. Fix: do not return early; keep a running best t and return the smallest non-negative one after testing every object.

H5. $t = 3.0$: the other one is behind the eye. You are looking at the **inside of the far wall** of the ball, because the eye is between the two crossings.

I. Generate: the eye, the pixel count, the arrow through the pixel centre. Intersect: the object list, comparing t values, deciding T_1 is hidden. Shade: the lights, the pixel colour. **Two ticks each**: the hit point and the normal $\mathbf{n}$ — intersection produces them, shading consumes them. That pair is the whole interface between the two boxes, which is why the book's pseudocode names the normal explicitly in line 3.

J1. green (wall, 6.0) · blue (triangle, $4.2 < 6.5$) · green (wall: the ball's -1.0 is behind the eye) · grey (no hit) · blue (triangle, 2.1 is smaller than the ball's 2.6).

J2. Five rays, one per pixel — generate runs once per pixel, not once per object. One found nothing, pixel 4. Pixels 2, 3 and 5 each have more than one candidate, so all three could change under a first-hit-wins loop; pixel 5 is the worst, since the ball is listed before the nearer triangle.

J3. Generate ran in full, intersect ran in full and reported no hit, and shade did almost nothing — there is no hit point and no normal to shade, so the loop just writes the background colour. Every pixel pays for generate and intersect whether or not it sees anything.

K1. The `t` is missing, so the method always computes $\mathbf{o} + \mathbf{d}$, which is $\mathbf{p}(1)$. Fix: `return self.origin + self.direction * t`.

K2. It takes the smallest t of all, so a hit at $t = -2$ beats every hit in front (H2). Fix: filter first — `min((t for t in hits if t >= 0), default=None)` — and handle the empty case as "no hit".

K3. First hit in list order wins, not nearest (H4, J2). Fix: do not return inside the loop; keep `best_t` and `best_obj`, update them when `t` is non-negative and smaller, and return after the loop.

K4. The direction was stored without normalizing, so t counts copies of $\mathbf{d}$, not units (F1, F4). Fix either `self.direction = direction.normalized()` in the constructor, or leave it and report `best_t * self.direction.length()`. Pick one and stay with it.

K5. Origin and direction are both wrong: the ray starts at the screen and points back at the eye. Fix: `Ray3(eye, pixel_centre - eye)` — origin is the eye, and the direction is eye-to-pixel, so $t = 1$ is the pixel and t grows into the scene.

L1. $t = 0$ is a real place, the origin $\mathbf{o}$, so it is included; infinity is not a number the parameter ever reaches, so there is no endpoint to include.

L2. Both hits lie on the same ray, so both are $\mathbf{o}$ plus some number of copies of the same arrow; more copies means further along, whatever $|\mathbf{d}|$ happens to be.

L3. They are the same formula and differ only in which interval of t is allowed, so one word covers all three — but when you read the word, check which interval is meant.

L4. Normalizing rescales the parameter without moving a single point of the ray, so the picture is unchanged and t gains a useful meaning: distance in world units.

L5. Each pixel's ray is answered without reference to any other pixel, so the pixels can be computed in any order, or all at the same time.

L6. Only the intersection test knows *which* object was hit and *where*, and the normal depends on both; handing it back costs nothing and saves shading from repeating the whole search.

M1. Floor: $5 - 3t = 0$, $t = \frac{5}{3} \approx 1.67$. Ceiling: $5 - 3t = 6$, $t = -\frac{1}{3}$. Right: $1 + 4t = 10$, $t = \frac{9}{4} = 2.25$. Left: $1 + 4t = 0$, $t = -\frac{1}{4}$. One linear equation per wall — this is the entire intersection test for an axis-aligned line.

M2. Drop $-\frac{1}{3}$ and $-\frac{1}{4}$: both are behind $\mathbf{o}$. Of $\frac{5}{3}$ and $\frac{9}{4}$, the smaller is $\frac{5}{3}$, so the **floor** is hit first, at $\mathbf{o} + \frac{5}{3}\mathbf{d} = (1 + \frac{20}{3},\ 0) = (\frac{23}{3},\ 0) \approx (7.67,\ 0)$, comfortably between $x = 0$ and $x = 10$. The right-wall solution at $t = 2.25$ is $(10, -1.75)$, which is below the floor — the ray had already left the room.

M3. Floor: $5 = 0$ has no solution; ceiling: $5 = 6$, likewise. Solving would divide by $d_y = 0$. Geometrically the ray is **parallel** to the floor and the ceiling, so it meets neither, ever. The side walls still work: right gives $t = 2.25$ at $(10, 5)$, which is inside $[0, 6]$, and left gives $t = -0.25$. So the ray hits the **right wall**. The code must test whether the relevant slot of $\mathbf{d}$ is zero (in practice, near zero) before dividing, and report "no hit" if it is.

M4. $2\mathbf{d}$: same set of points, every t halved, so the floor still wins, now at $t = \frac{5}{6}$ — same wall, same hit point, a different address for it. $-\mathbf{d}$: a **different ray**, the other half-line. The two t values that were negative become positive and the two that were positive become negative, so the winner is now the **left wall** at $t = \frac{1}{4}$, hit point $(0,\ \frac{23}{4}) = (0,\ 5.75)$ — and 5.75 is between 0 and 6, so that one is a genuine hit. Scaling $\mathbf{d}$ by a positive number renames the points; scaling by a negative number turns the ray around.