# Lecture 10: Rays

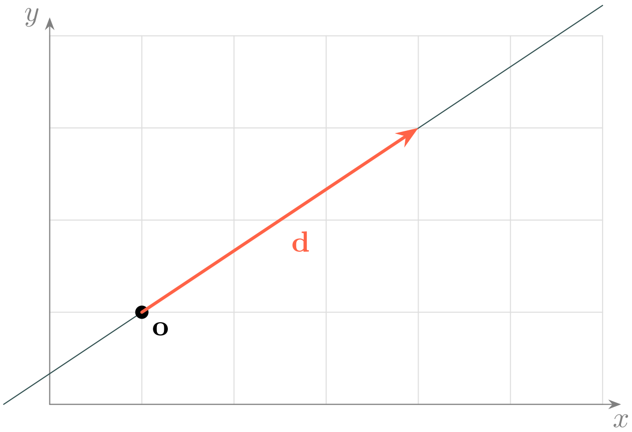
## $\mathbf{p}(t) = \mathbf{o} + t\mathbf{d}$ and the Nearest Hit

**CS 453 — Computer Graphics**


Fakhir Shaheen

Forman Christian University

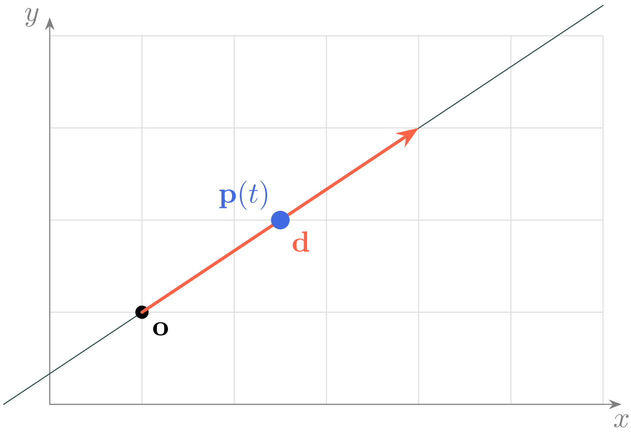
# One formula draws every line




- A start point and an arrow

$$\mathbf{p}(t) = \mathbf{o} + t\mathbf{d}$$

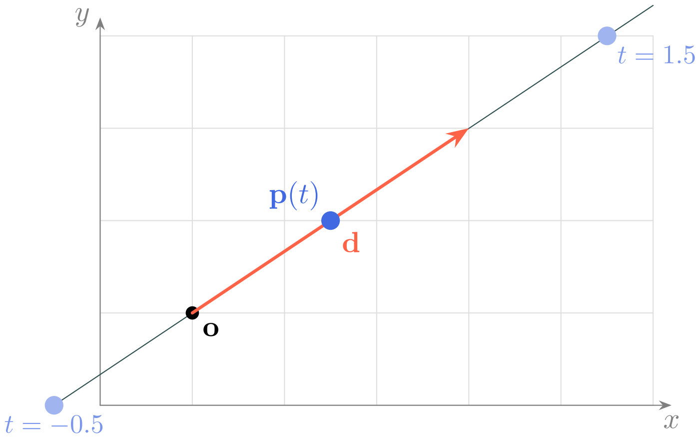
# One formula draws every line




- A start point and an arrow
- One number $t$ slides the point

$$\mathbf{p}(t) = \mathbf{o} + t\mathbf{d}$$

# One formula draws every line




- A start point and an arrow
- One number $t$ slides the point
- Lines, segments, rays: all this formula

$$\mathbf{p}(t) = \mathbf{o} + t\mathbf{d}$$

# Our conventions in one picture

$$\mathbf{p} = \begin{bmatrix} x \\ y \end{bmatrix}$$

column vector

$\mathbf{o},\ \mathbf{d},\ \mathbf{p}$

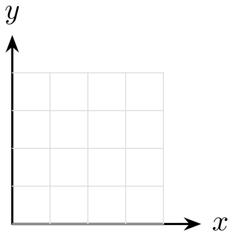
bold: point or arrow

$t,\ x,\ \phi$

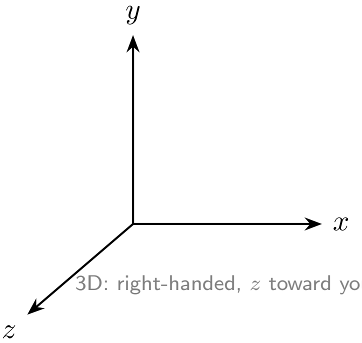
plain: one number

$$\mathbf{p}(t) = \mathbf{o} + t\,\mathbf{d}$$

point = point + number × arrow




2D: $y$ up

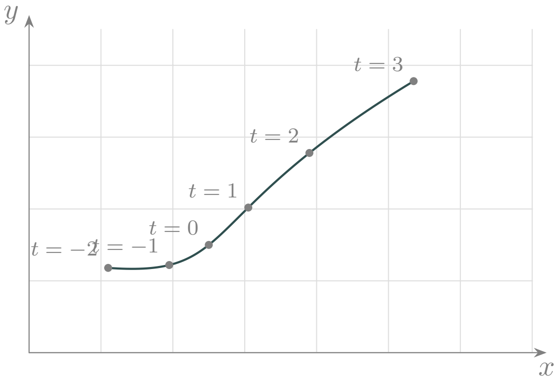



3D: right-handed, $z$ toward you

# Curves as moving points

One number $t$, one point on the curve

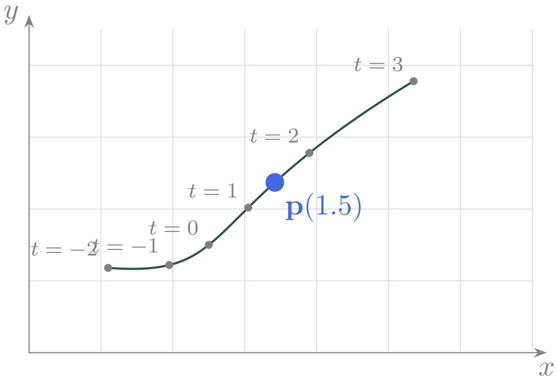
# A curve is a moving point




- $t$ is like a clock reading

$$\mathbf{p} = \mathbf{f}(t), \qquad \mathbf{f} : \mathbb{R} \to \mathbb{R}^2$$

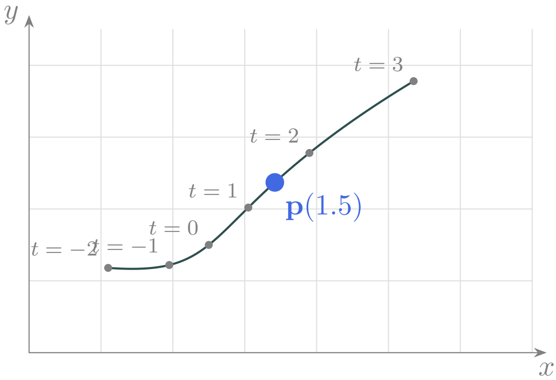
# A curve is a moving point




- $t$ is like a clock reading
- Read the clock, get the point

$$\mathbf{p} = \mathbf{f}(t), \qquad \mathbf{f} : \mathbb{R} \to \mathbb{R}^2$$

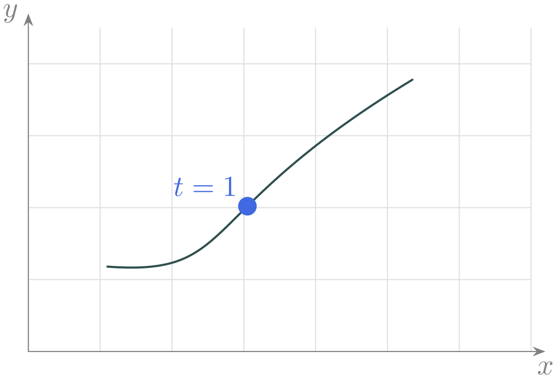
# A curve is a moving point




- $t$ is like a clock reading
- Read the clock, get the point
- Small change in $t$, small move

$$\mathbf{p} = \mathbf{f}(t), \qquad \mathbf{f} : \mathbb{R} \to \mathbb{R}^2$$

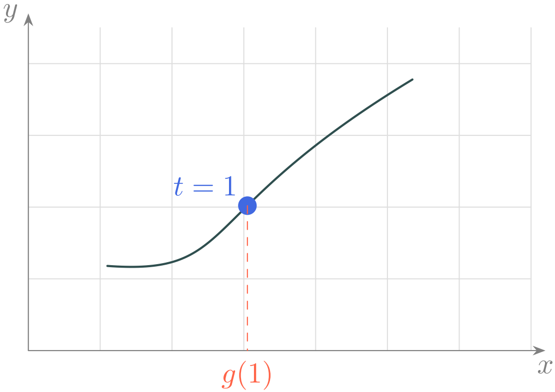
# Each $t$ picks one point




- One function for each slot

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} g(t) \\ h(t) \end{bmatrix}$$

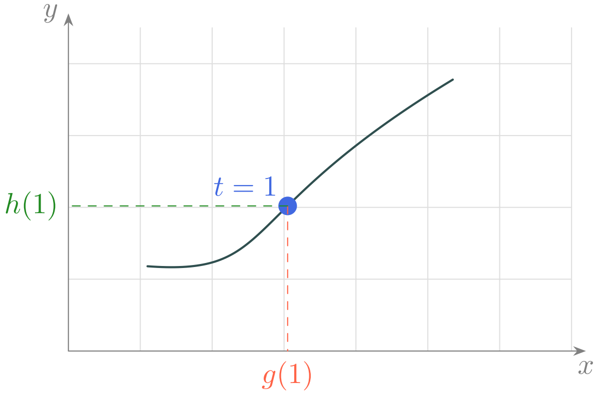
# Each $t$ picks one point




- One function for each slot
- $g$ gives the $x$ value

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} g(t) \\ h(t) \end{bmatrix}$$

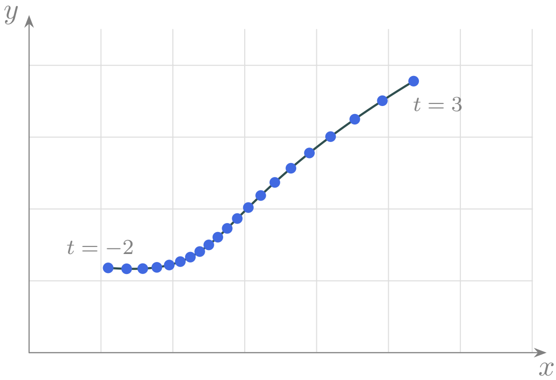
# Each $t$ picks one point




- One function for each slot
- $g$ gives the $x$ value
- $h$ gives the $y$ value

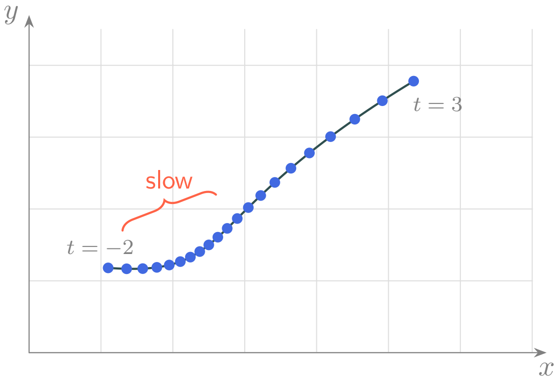
$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} g(t) \\ h(t) \end{bmatrix}$$

# Slow here, fast there



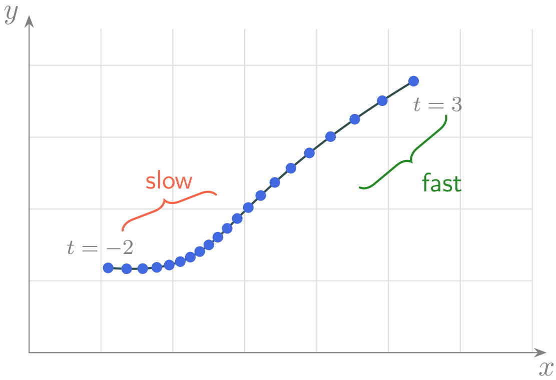

- Equal steps of $t$

# Slow here, fast there



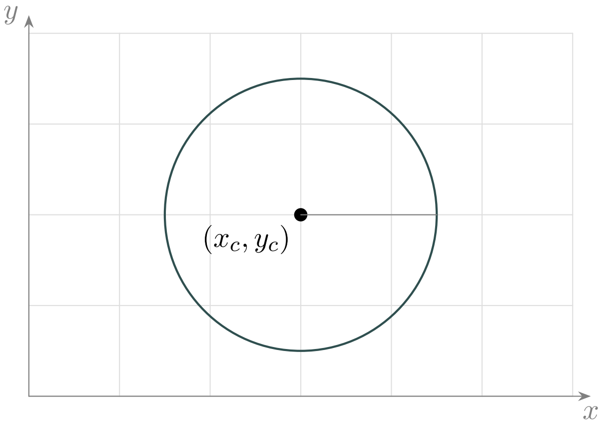

- Equal steps of $t$
- Beads bunch up: slow

# Slow here, fast there




- Equal steps of $t$
- Beads bunch up: slow
- Beads spread out: fast

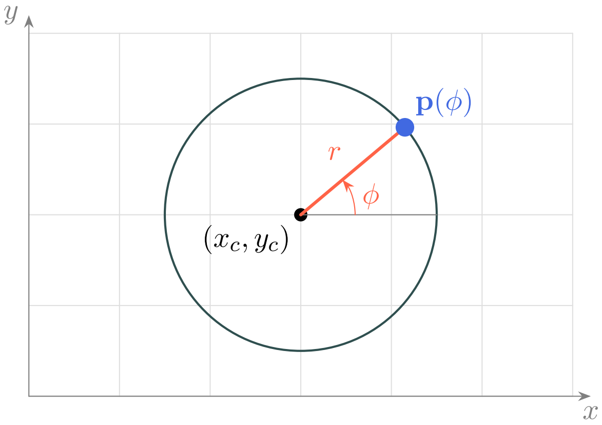
# A circle: the parameter is an angle




- Centre, radius, and an angle

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_c + r\cos\phi \\ y_c + r\sin\phi \end{bmatrix}$$

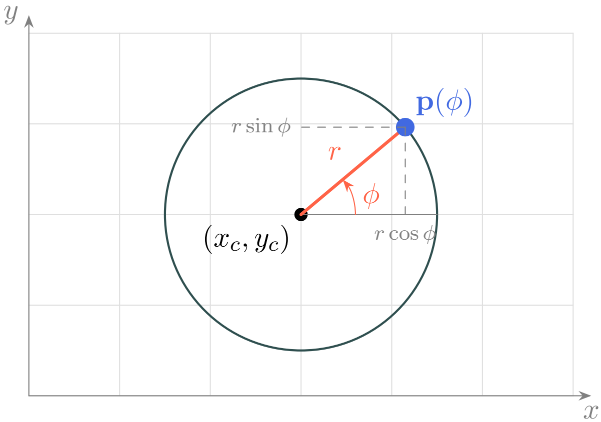
# A circle: the parameter is an angle




- Centre, radius, and an angle
- Turn $\phi$: the bead orbits

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_c + r\cos\phi \\ y_c + r\sin\phi \end{bmatrix}$$

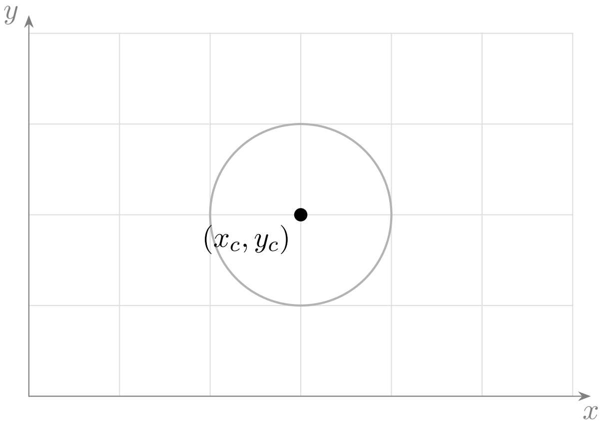
# A circle: the parameter is an angle




- Centre, radius, and an angle
- Turn $\phi$: the bead orbits
- One angle per point: half-open range

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_c + r\cos\phi \\ y_c + r\sin\phi \end{bmatrix}$$

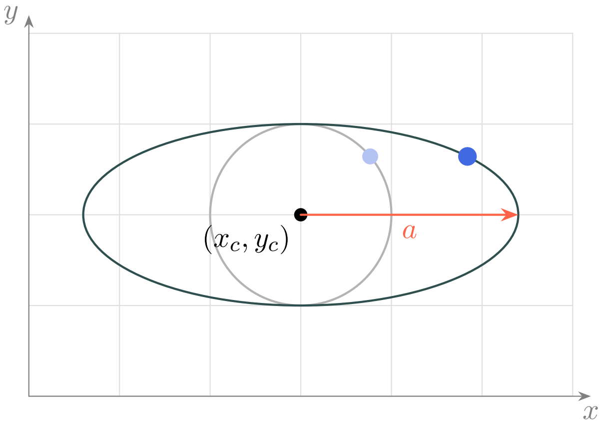
# Stretch each axis: an ellipse




- Start from the circle

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_c + a\cos\phi \\ y_c + b\sin\phi \end{bmatrix}$$

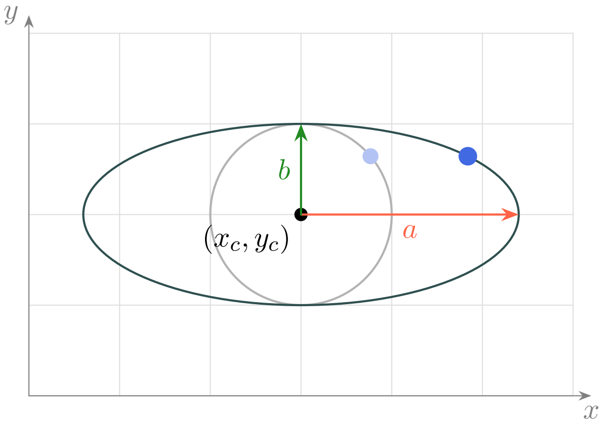
# Stretch each axis: an ellipse




- Start from the circle
- Scale $x$ by $a$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_c + a\cos\phi \\ y_c + b\sin\phi \end{bmatrix}$$

# Stretch each axis: an ellipse

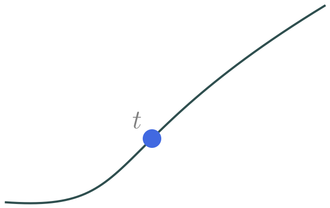



- Start from the circle
- Scale $x$ by $a$
- Scale $y$ by $b$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_c + a\cos\phi \\ y_c + b\sin\phi \end{bmatrix}$$

# Recap: one number in, one point out

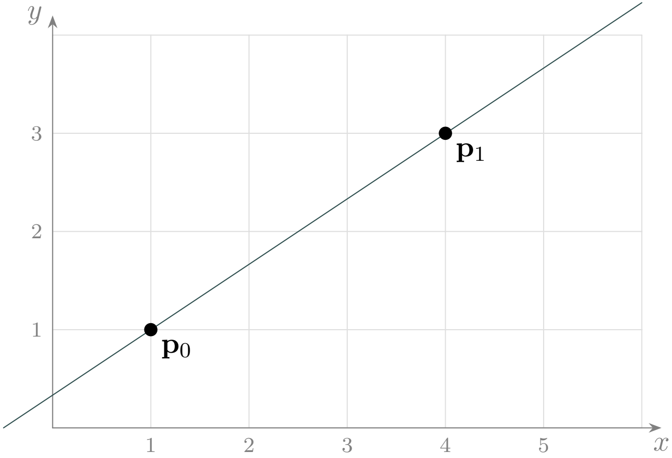
$$\mathbf{p} = \mathbf{f}(t)$$



# Lines in 2D

Start at a point, walk toward another

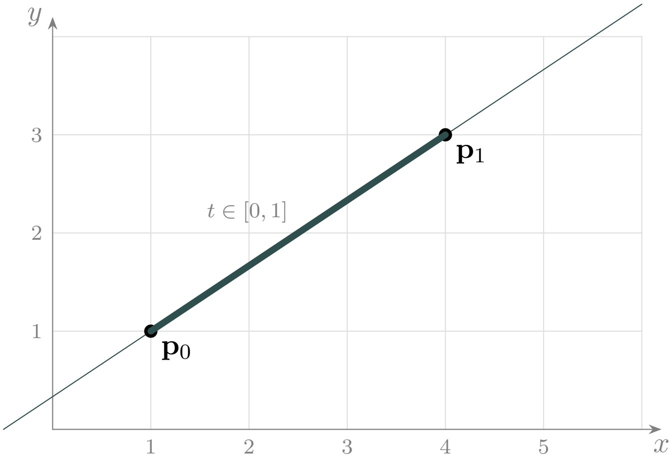
# A line through two points




- Two points fix the line

$$\mathbf{p}(t) = \mathbf{p}_0 + t(\mathbf{p}_1 - \mathbf{p}_0)$$

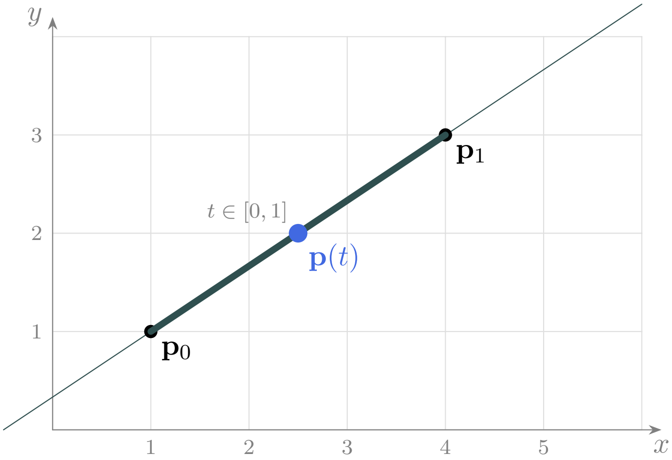
# A line through two points




- Two points fix the line
- Bold part: the segment between

$$\mathbf{p}(t) = \mathbf{p}_0 + t(\mathbf{p}_1 - \mathbf{p}_0)$$

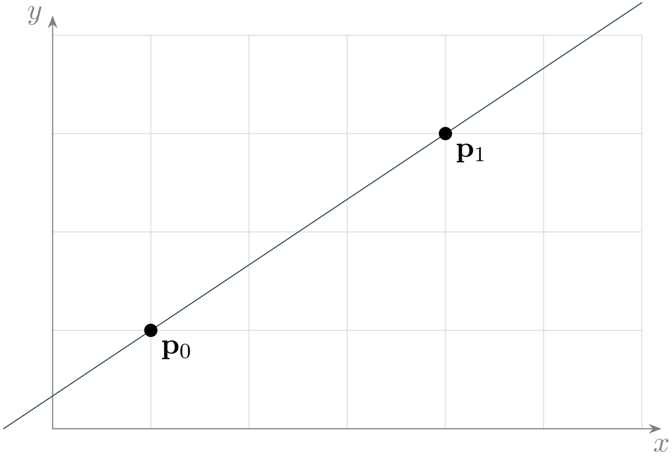
# A line through two points




- Two points fix the line
- Bold part: the segment between
- The bead can slide anywhere

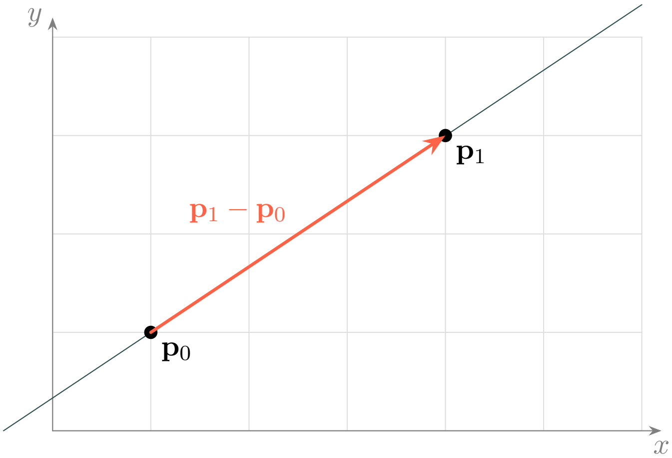
$$\mathbf{p}(t) = \mathbf{p}_0 + t(\mathbf{p}_1 - \mathbf{p}_0)$$

# Read it: start at $\mathbf{p}_0$, head toward $\mathbf{p}_1$



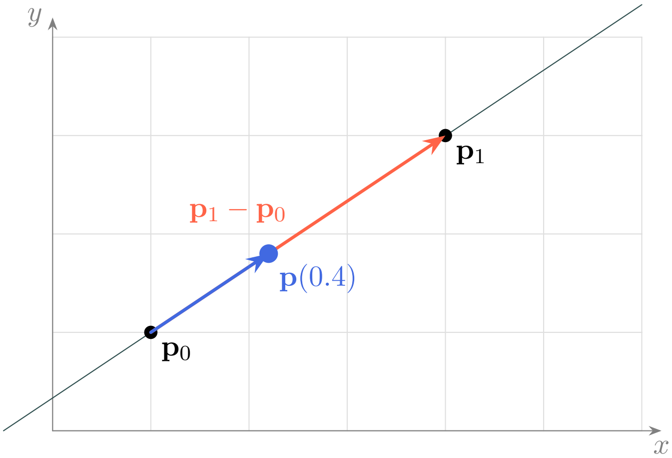

- Stand at $\mathbf{p}_0$

# Read it: start at $\mathbf{p}_0$, head toward $\mathbf{p}_1$



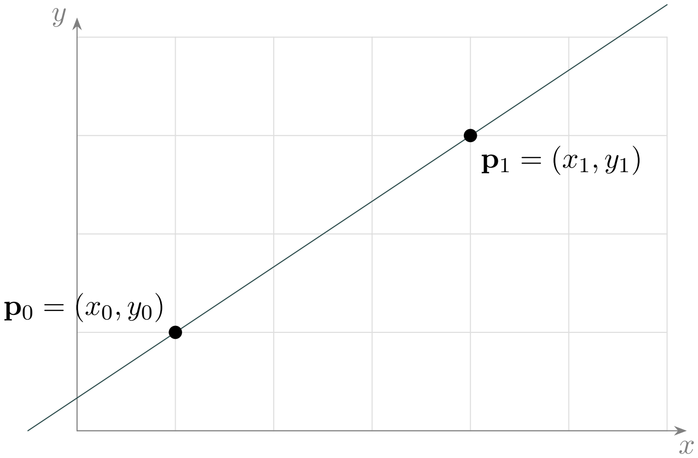

- Stand at $\mathbf{p}_0$
- Face $\mathbf{p}_1$: that arrow

# Read it: start at $\mathbf{p}_0$, head toward $\mathbf{p}_1$




- Stand at $\mathbf{p}_0$
- Face $\mathbf{p}_1$: that arrow
- Walk $t$ of the way

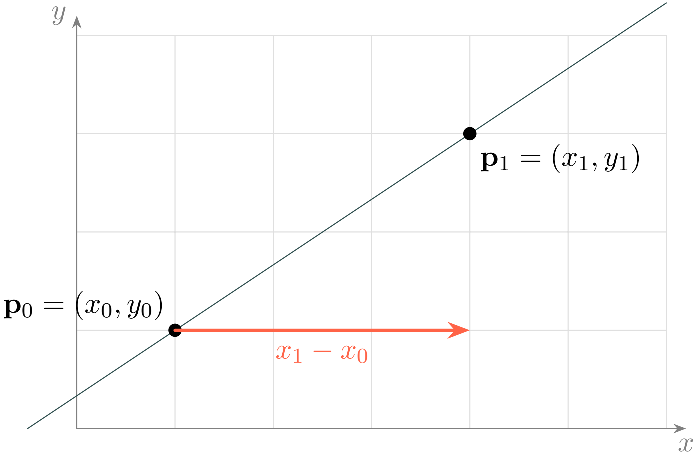
# Same formula, one slot at a time




- Two slots, same shape

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_0 + t(x_1 - x_0) \\ y_0 + t(y_1 - y_0) \end{bmatrix}$$

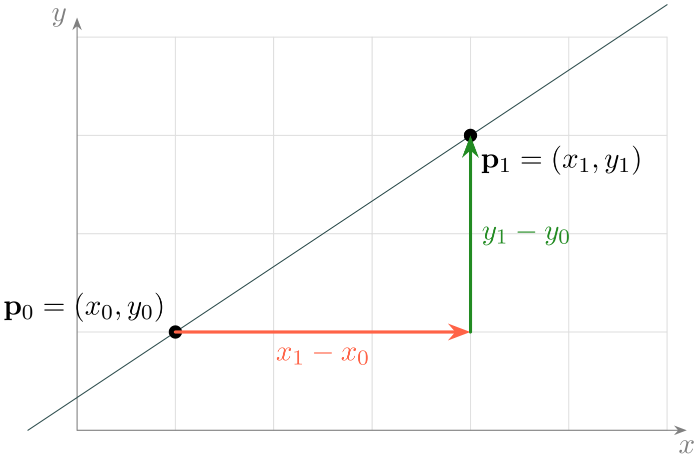
# Same formula, one slot at a time




- Two slots, same shape
- $x$ slot: run

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_0 + t(x_1 - x_0) \\ y_0 + t(y_1 - y_0) \end{bmatrix}$$

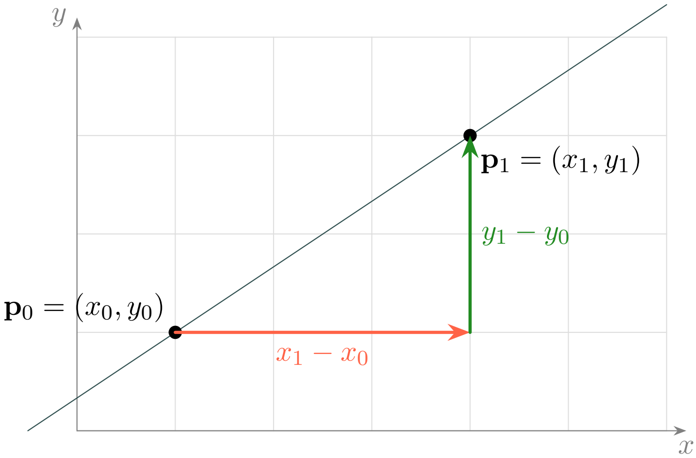
# Same formula, one slot at a time




- Two slots, same shape
- $x$ slot: run
- $y$ slot: rise

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_0 + t(x_1 - x_0) \\ y_0 + t(y_1 - y_0) \end{bmatrix}$$

# Same formula, one slot at a time

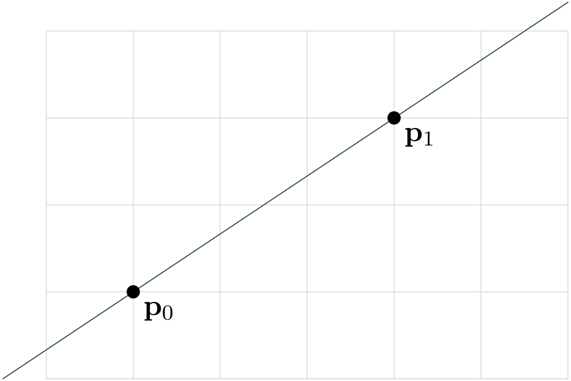



- Two slots, same shape
- $x$ slot: run
- $y$ slot: rise
- So we write the vector form

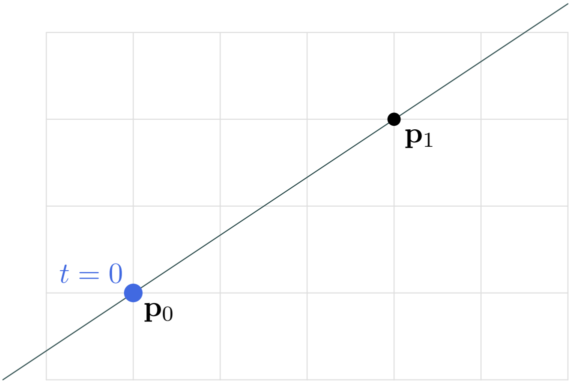
$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_0 + t(x_1 - x_0) \\ y_0 + t(y_1 - y_0) \end{bmatrix}$$

$$\mathbf{p}(t) = \mathbf{p}_0 + t(\mathbf{p}_1 - \mathbf{p}_0)$$

# $t = 0$ lands on $\mathbf{p}_0$, $t = 1$ on $\mathbf{p}_1$

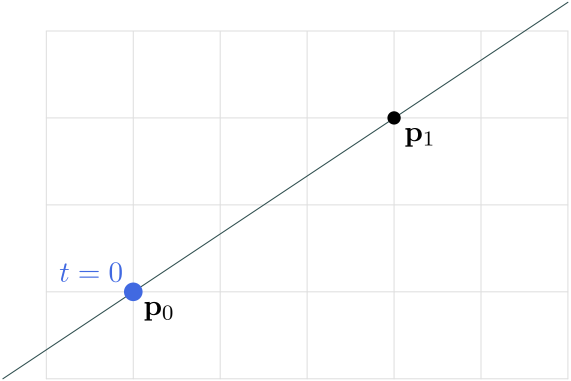


# $t = 0$ lands on $\mathbf{p}_0$, $t = 1$ on $\mathbf{p}_1$



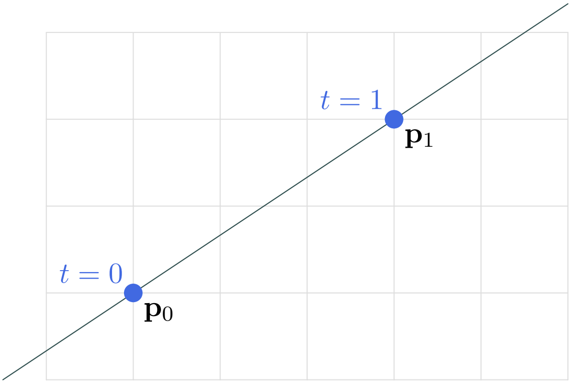

$$\mathbf{p}(0) = \mathbf{p}_0 + 0\,(\mathbf{p}_1 - \mathbf{p}_0) \qquad \text{put in } t = 0$$

# $t = 0$ lands on $\mathbf{p}_0$, $t = 1$ on $\mathbf{p}_1$




$$
\begin{aligned}
\mathbf{p}(0) &= \mathbf{p}_0 + 0\,(\mathbf{p}_1 - \mathbf{p}_0) && \text{put in } t = 0 \\
&= \mathbf{p}_0 + \mathbf{0} && \text{zero times arrow} \\
&= \mathbf{p}_0
\end{aligned}
$$

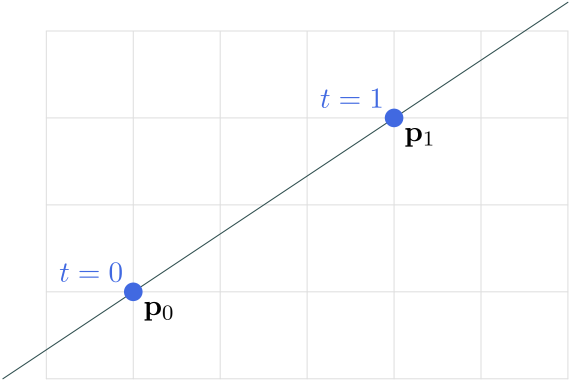
# $t = 0$ lands on $\mathbf{p}_0$, $t = 1$ on $\mathbf{p}_1$




$$\begin{aligned}
\mathbf{p}(0) &= \mathbf{p}_0 + 0\,(\mathbf{p}_1 - \mathbf{p}_0) && \text{put in } t = 0 \\
&= \mathbf{p}_0 + \mathbf{0} && \text{zero times arrow} \\
&= \mathbf{p}_0
\end{aligned}$$

$$\mathbf{p}(1) = \mathbf{p}_0 + 1\,(\mathbf{p}_1 - \mathbf{p}_0) \qquad \text{put in } t = 1$$

# $t = 0$ lands on $\mathbf{p}_0$, $t = 1$ on $\mathbf{p}_1$

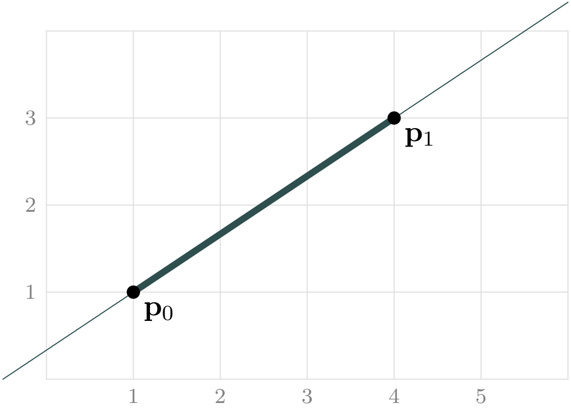



$$
\begin{aligned}
\mathbf{p}(0) &= \mathbf{p}_0 + 0\,(\mathbf{p}_1 - \mathbf{p}_0) && \text{put in } t = 0 \\
&= \mathbf{p}_0 + \mathbf{0} && \text{zero times arrow} \\
&= \mathbf{p}_0
\end{aligned}
$$

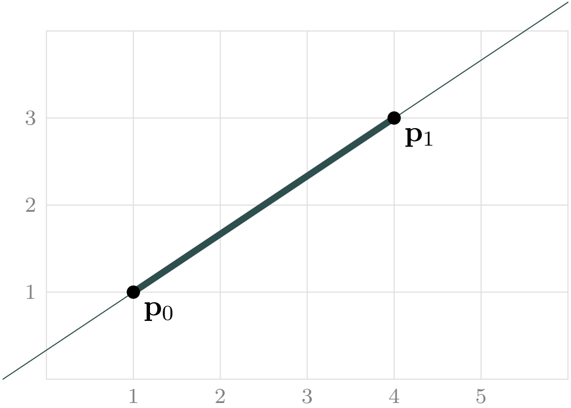
$$
\begin{aligned}
\mathbf{p}(1) &= \mathbf{p}_0 + 1\,(\mathbf{p}_1 - \mathbf{p}_0) && \text{put in } t = 1 \\
&= \mathbf{p}_0 + \mathbf{p}_1 - \mathbf{p}_0 && \text{one times arrow} \\
&= \mathbf{p}_1
\end{aligned}
$$

- The ends come out right

# $t$ is the fraction of the way



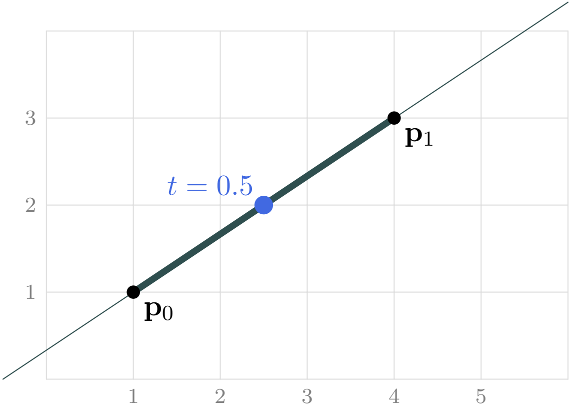
# $t$ is the fraction of the way




$$\begin{aligned}\mathbf{p}(0.5) &= (1,1) + 0.5\left((4,3) - (1,1)\right) \\ &= (1,1) + 0.5\,(3,2)\end{aligned}$$

arrow first

# $t$ is the fraction of the way

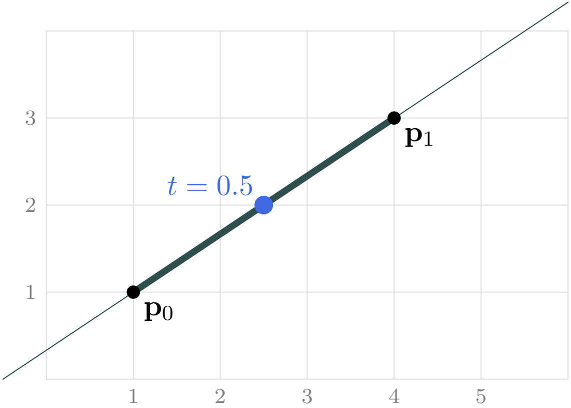



$$
\begin{aligned}
\mathbf{p}(0.5) &= (1,1) + 0.5\left((4,3) - (1,1)\right) \\
&= (1,1) + 0.5\,(3,2) \\
&= (1,1) + (1.5,1) = (2.5,\ 2)
\end{aligned}
$$

arrow first

- Halfway in $t$: halfway in space

# $t$ is the fraction of the way



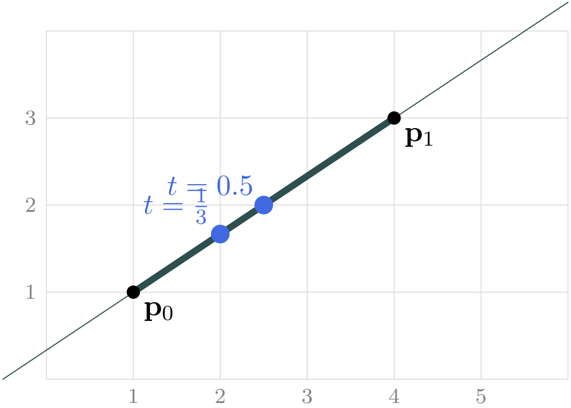

$$\begin{aligned}\mathbf{p}(0.5) &= (1,1) + 0.5\,\big((4,3) - (1,1)\big)\\ &= (1,1) + 0.5\,(3,2)\\ &= (1,1) + (1.5,1) = (2.5,\ 2)\end{aligned}$$

arrow first

$$\mathbf{p}(\tfrac{1}{3}) = (1,1) + \tfrac{1}{3}\,(3,2)$$

- Halfway in $t$: halfway in space

# $t$ is the fraction of the way



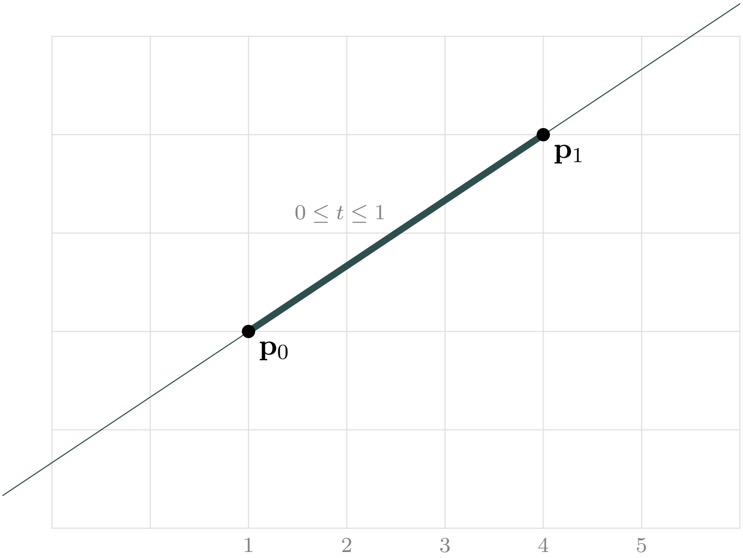

$$\begin{aligned}\mathbf{p}(0.5) &= (1,1) + 0.5\,\big((4,3) - (1,1)\big)\\ &= (1,1) + 0.5\,(3,2)\\ &= (1,1) + (1.5,1) = (2.5,\ 2)\end{aligned}$$

arrow first

$$\begin{aligned}\mathbf{p}(\tfrac{1}{3}) &= (1,1) + \tfrac{1}{3}\,(3,2)\\ &= (1,1) + (1,\tfrac{2}{3}) = (2,\ \tfrac{5}{3})\end{aligned}$$

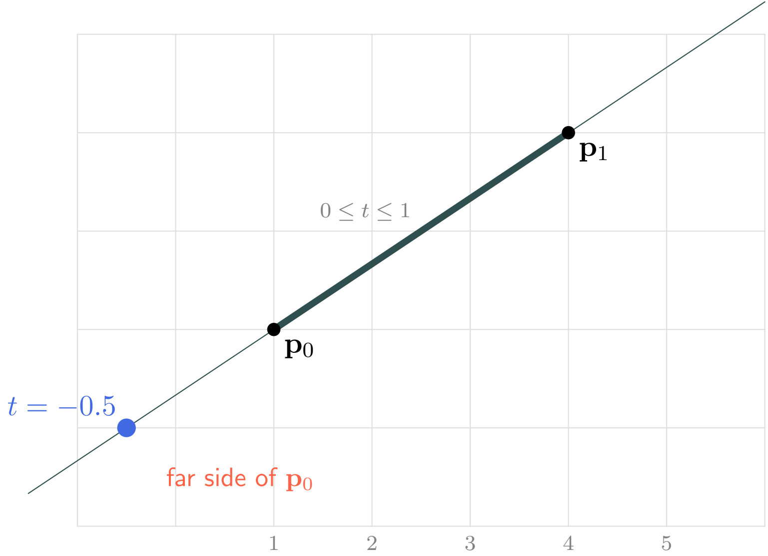
- Halfway in $t$: halfway in space
- A third of $t$: a third of the trip

# Past the ends: $t < 0$ and $t > 1$



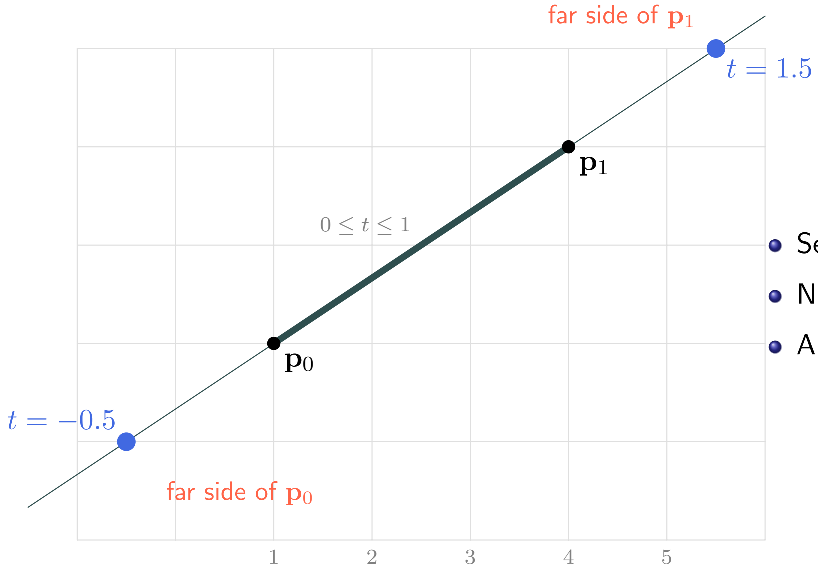

- Segment: $t$ between 0 and 1

# Past the ends: $t < 0$ and $t > 1$



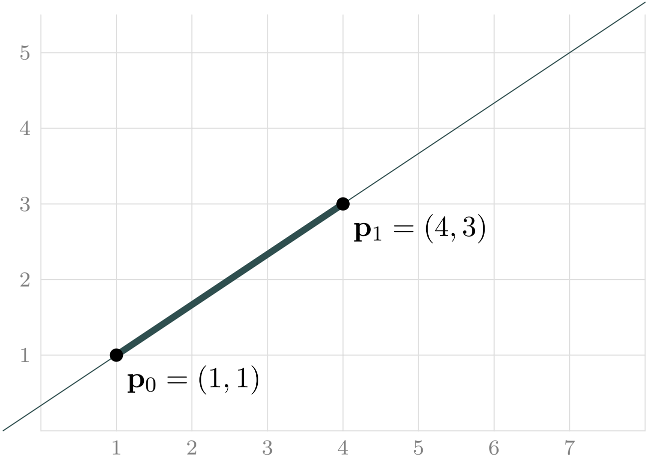

- Segment: $t$ between 0 and 1
- Negative $t$: behind the start

# Past the ends: $t < 0$ and $t > 1$



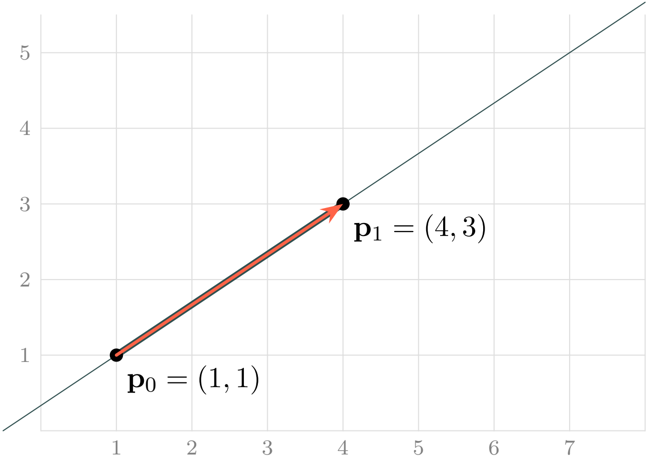

- Segment: $t$ between 0 and 1
- Negative $t$: behind the start
- Above 1: past the end

# Quick check 1: where is $t = 2$?



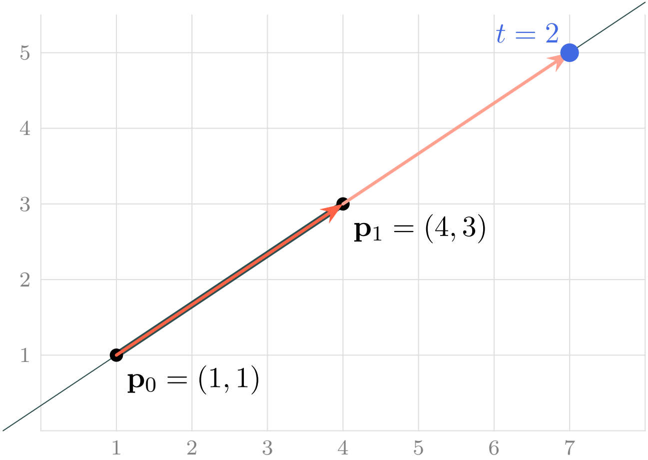

- Find $\mathbf{p}(2)$ by hand

# Quick check 1: where is $t = 2$?




- Find $\mathbf{p}(2)$ by hand
- Arrow: $(4, 3) - (1, 1) = (3, 2)$

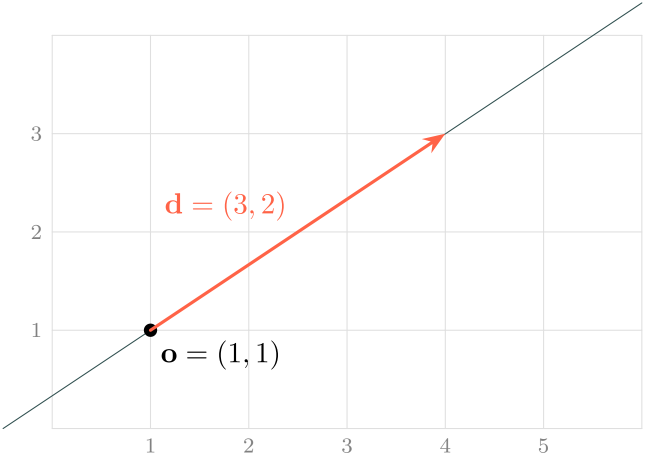
# Quick check 1: where is $t = 2$?




- Find $\mathbf{p}(2)$ by hand
- Arrow: $(4, 3) - (1, 1) = (3, 2)$
- Twice the arrow: $(6, 4)$

$$\mathbf{p}(2) = (1, 1) + (6, 4) = (7,\ 5)$$

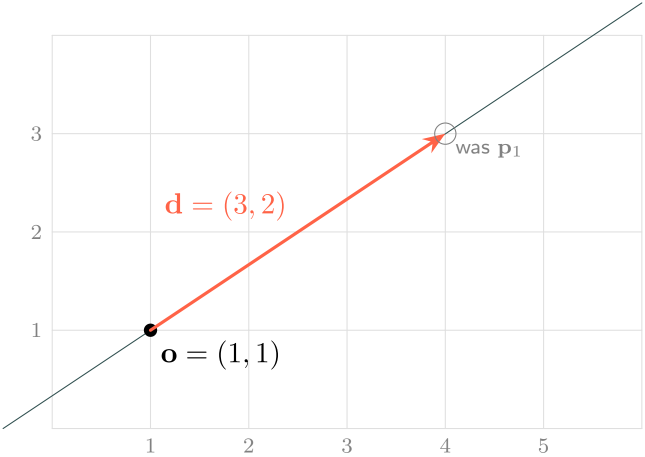
# Drop $\mathbf{p}_1$: an origin and a direction




- Keep the start, keep the arrow

$$\mathbf{p}(t) = \mathbf{o} + t\,\mathbf{d}$$

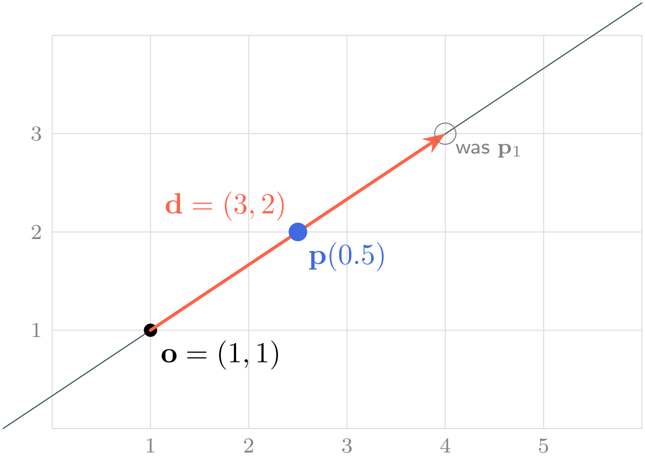
# Drop $\mathbf{p}_1$: an origin and a direction




- Keep the start, keep the arrow
- The second point is not needed

$$\mathbf{p}(t) = \mathbf{o} + t\,\mathbf{d}$$

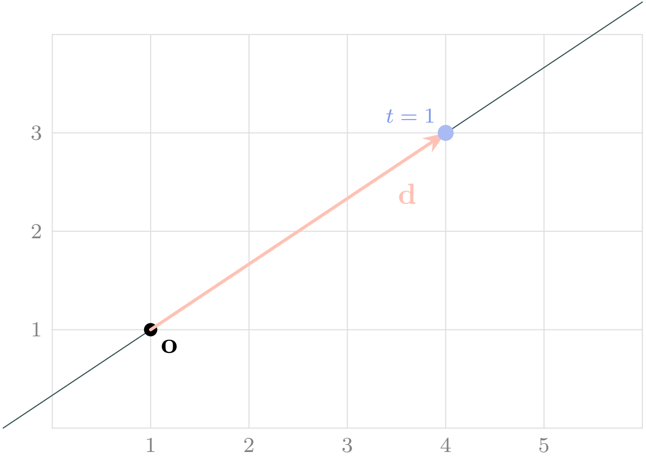
# Drop $\mathbf{p}_1$: an origin and a direction




- Keep the start, keep the arrow
- The second point is not needed
- Same bead, shorter formula

$$\mathbf{p}(t) = \mathbf{o} + t\,\mathbf{d}$$

# Unit $\mathbf{d}$: $t$ measures distance

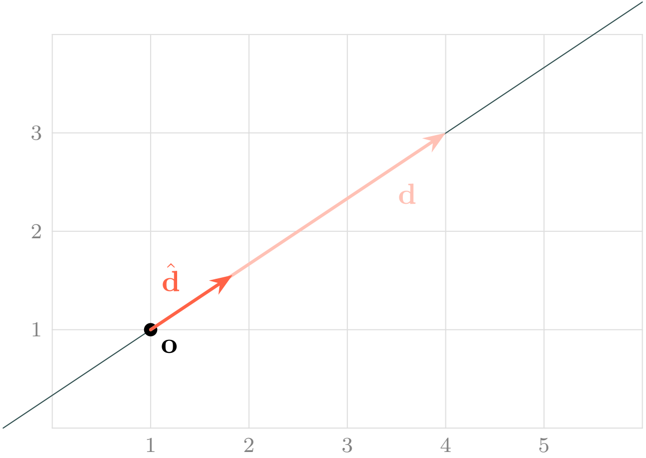



distance $= t\,|\mathbf{d}|$

$$|\mathbf{d}| = \sqrt{3^2 + 2^2} = \sqrt{13} \approx 3.6$$

- Long $\mathbf{d}$: $t = 1$ is $3.6$ units out

# Unit $\mathbf{d}$: $t$ measures distance



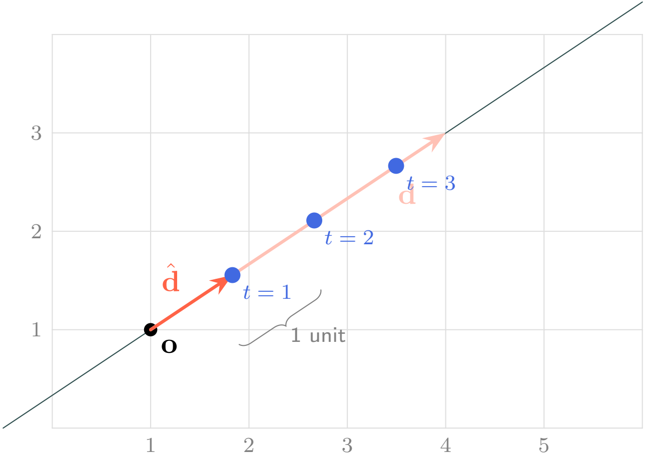

distance $= t\,|\mathbf{d}|$

$$|\mathbf{d}| = \sqrt{3^2 + 2^2} = \sqrt{13} \approx 3.6$$

$$\hat{\mathbf{d}} = \frac{1}{\sqrt{13}}(3, 2) \approx (0.83,\ 0.55)$$

- Long $\mathbf{d}$: $t = 1$ is $3.6$ units out
- Shrink $\mathbf{d}$ to length 1

# Unit $\mathbf{d}$: $t$ measures distance




distance $= t\,|\mathbf{d}|$

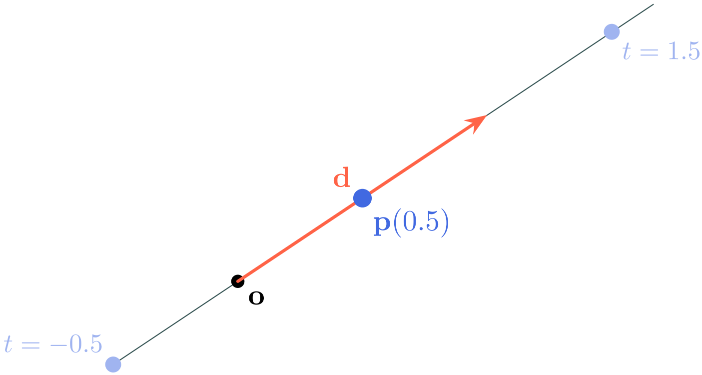
$$|\mathbf{d}| = \sqrt{3^2 + 2^2} = \sqrt{13} \approx 3.6$$

$$\hat{\mathbf{d}} = \frac{1}{\sqrt{13}}(3, 2) \approx (0.83,\ 0.55)$$

- Long $\mathbf{d}$: $t = 1$ is $3.6$ units out
- Shrink $\mathbf{d}$ to length 1
- Factor is 1: $t$ *is* the distance
- A line has one speed $|\mathbf{d}|$; a curve speeds up and slows down

# Recap: start point plus $t$ copies of the arrow

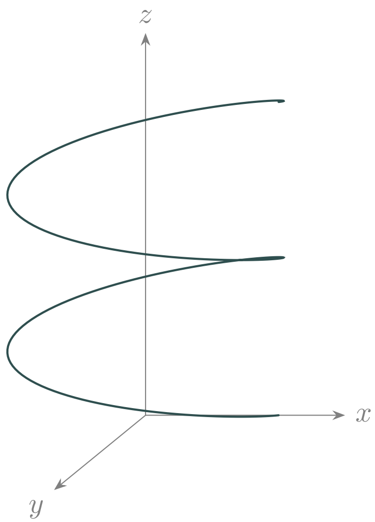
$$\mathbf{p}(t) = \mathbf{o} + t\,\mathbf{d}$$



# Lines in 3D

Add a third slot; nothing else changes

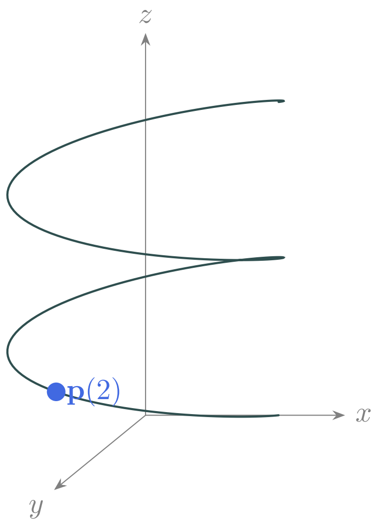
# 3D curves: three functions of $t$




- One function per slot, now three

$$x = f(t) \qquad x = \cos t$$
$$y = g(t) \qquad y = \sin t$$
$$z = h(t) \qquad z = t$$

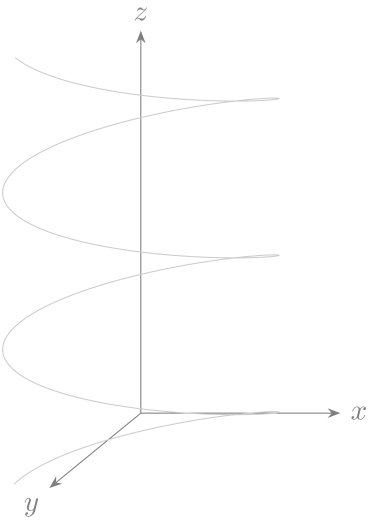
# 3D curves: three functions of $t$




- One function per slot, now three
- A spiral climbing the $z$-axis

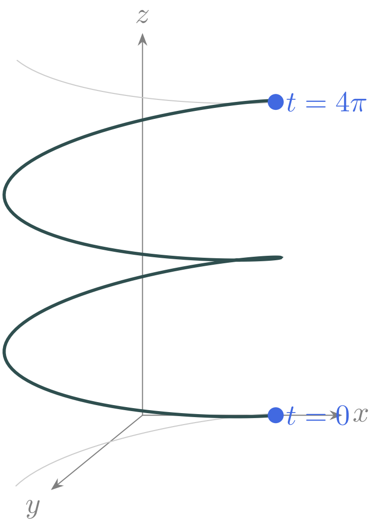
$$\begin{aligned} x &= f(t) & x &= \cos t \\ y &= g(t) & y &= \sin t \\ z &= h(t) & z &= t \end{aligned}$$

# Domain $D$: where the curve starts and ends




- Grey: the curve for every $t$

# Domain $D$: where the curve starts and ends




- Grey: the curve for every $t$
- Dark: $t$ restricted to $D$
- Two turns, then stop

$$D = [0,\ 4\pi] \subset \mathbb{R}$$

# A 3D line, one slot at a time

$$\begin{aligned} x &= 2 \quad + 7t \\ y &= 1 \quad + 2t \\ z &= 3 \quad - 5t \end{aligned}$$

- Same shape in each slot

# A 3D line, one slot at a time

$$\begin{aligned} x &= \boxed{2} + 7t \\ y &= 1 + 2t \\ z &= 3 - 5t \end{aligned}$$

start

- Same shape in each slot
- A start number

# A 3D line, one slot at a time

$$\begin{aligned} x &= \boxed{2} \; \boxed{+7t} \\ y &= 1 \; +2t \\ z &= 3 \; -5t \end{aligned}$$

$$\underbrace{\begin{matrix}2\\1\\3\end{matrix}}_{\text{start}} \qquad \underbrace{\begin{matrix}+7t\\+2t\\-5t\end{matrix}}_{t \times \text{step}}$$

- Same shape in each slot
- A start number
- Plus $t$ times a step

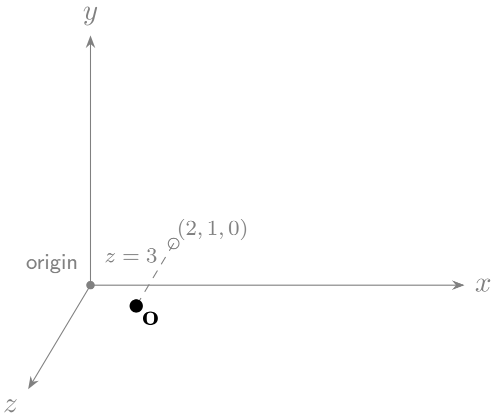
# A 3D line, one slot at a time

$$\begin{aligned} x &= \boxed{\begin{matrix} 2 \\ 1 \\ 3 \end{matrix}} \; \boxed{\begin{matrix} +\,7t \\ +\,2t \\ -\,5t \end{matrix}} \end{aligned}$$

$$\begin{array}{rcc} x = & 2 & +\,7t \\ y = & 1 & +\,2t \\ z = & 3 & -\,5t \\ & \text{start} & t \times \text{step} \end{array}$$

- Same shape in each slot
- A start number
- Plus $t$ times a step
- Clumsy in code: three variables

# Vector form: $\mathbf{o} = (2, 1, 3)$, $\mathbf{d} = (7, 2, -5)$

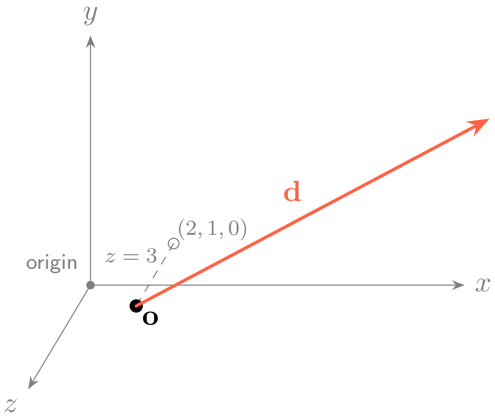



- The three starts: a point

$$\mathbf{p} = \mathbf{o} + t\mathbf{d}$$

$$\mathbf{o} = (2, 1, 3), \quad \mathbf{d} = (7, 2, -5)$$

# Vector form: $\mathbf{o} = (2, 1, 3)$, $\mathbf{d} = (7, 2, -5)$

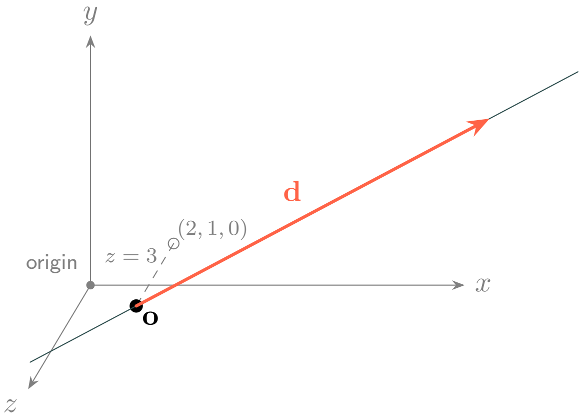



- The three starts: a point
- The three steps: an arrow

$$\mathbf{p} = \mathbf{o} + t\mathbf{d}$$

$$\mathbf{o} = (2, 1, 3), \quad \mathbf{d} = (7, 2, -5)$$

# Vector form: $\mathbf{o} = (2, 1, 3)$, $\mathbf{d} = (7, 2, -5)$

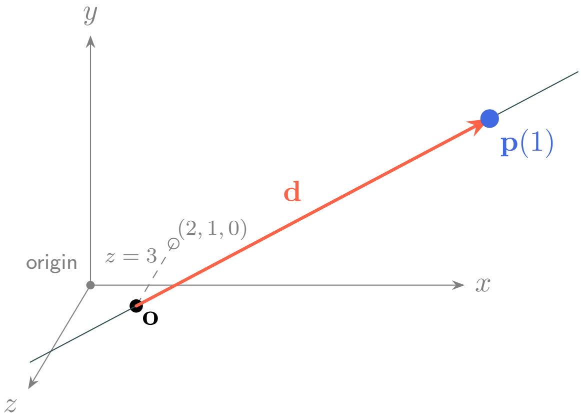



- The three starts: a point
- The three steps: an arrow
- Line through $\mathbf{o}$, parallel to $\mathbf{d}$

$$\mathbf{p} = \mathbf{o} + t\mathbf{d}$$

$$\mathbf{o} = (2, 1, 3), \quad \mathbf{d} = (7, 2, -5)$$

# Vector form: $\mathbf{o} = (2, 1, 3)$, $\mathbf{d} = (7, 2, -5)$

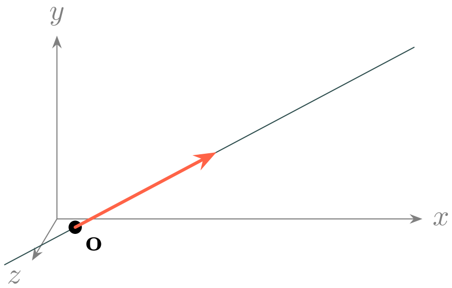



- The three starts: a point
- The three steps: an arrow
- Line through $\mathbf{o}$, parallel to $\mathbf{d}$
- $t = 1$ is the tip: $(9, 3, -2)$

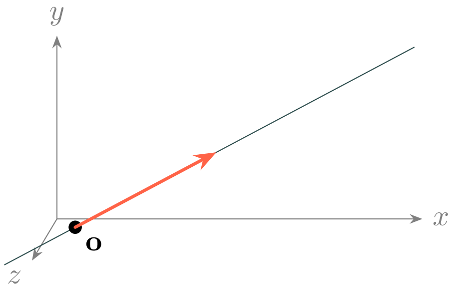
$$\mathbf{p} = \mathbf{o} + t\mathbf{d}$$

$$\mathbf{o} = (2, 1, 3), \quad \mathbf{d} = (7, 2, -5)$$

# Worked: $t = 2$ lands at $(16, 5, -7)$

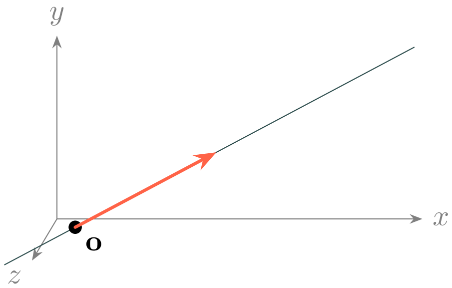


# Worked: $t = 2$ lands at $(16, 5, -7)$




$$\mathbf{p}(2) = (2, 1, 3) + 2\,(7, 2, -5)$$

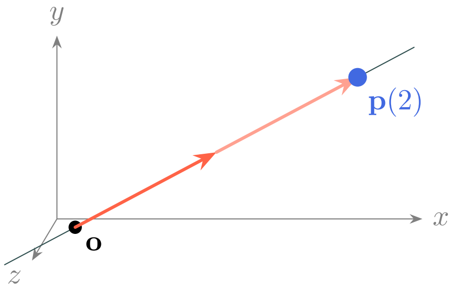
# Worked: $t = 2$ lands at $(16, 5, -7)$




$$\begin{aligned}\mathbf{p}(2) &= (2, 1, 3) + 2\,(7, 2, -5) \\ &= (2, 1, 3) + (14, 4, -10)\end{aligned}$$

scale the arrow

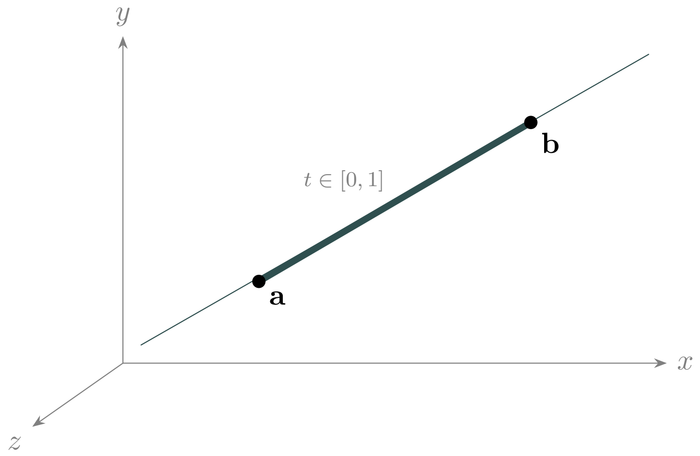
# Worked: $t = 2$ lands at $(16, 5, -7)$




$$
\begin{aligned}
\mathbf{p}(2) &= (2,1,3) + 2\,(7,2,-5) \\
&= (2,1,3) + (14,4,-10) && \text{scale the arrow} \\
&= (2+14,\ 1+4,\ 3-10) && \text{add slot by slot} \\
&= (16,\ 5,\ -7)
\end{aligned}
$$

- Two arrows from $\mathbf{o}$

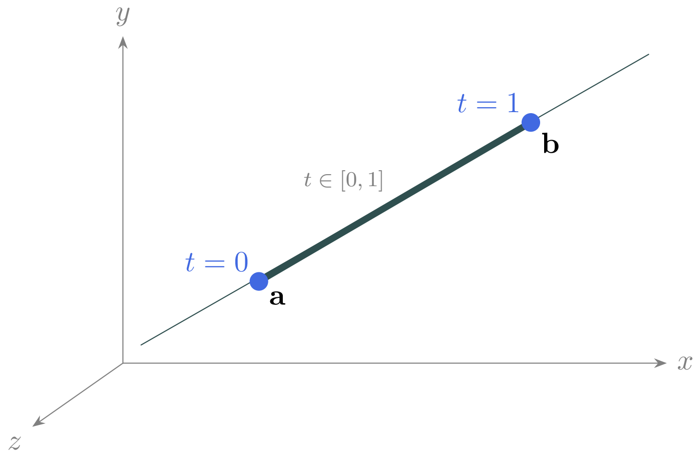
# Segment: $\mathbf{a}$, $\mathbf{b}$, and $t$ in $[0, 1]$




- A piece of the line

$$\mathbf{p}(t) = \mathbf{a} + t(\mathbf{b} - \mathbf{a})$$

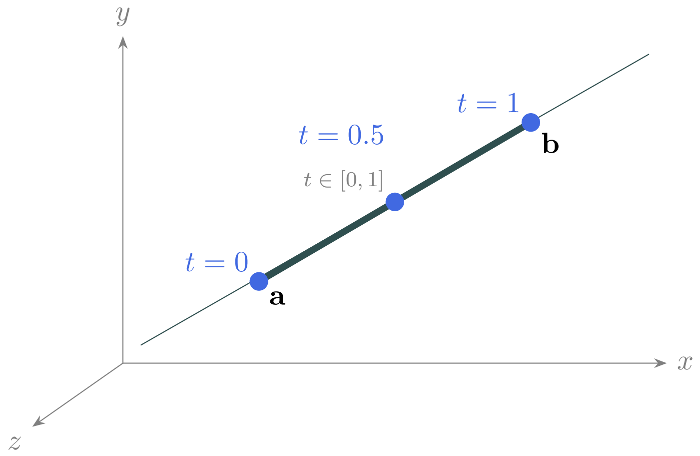
# Segment: $\mathbf{a}$, $\mathbf{b}$, and $t$ in $[0, 1]$




- A piece of the line
- $t = 0$ at $\mathbf{a}$, $t = 1$ at $\mathbf{b}$

$$\mathbf{p}(t) = \mathbf{a} + t(\mathbf{b} - \mathbf{a})$$

# Segment: $\mathbf{a}$, $\mathbf{b}$, and $t$ in $[0, 1]$

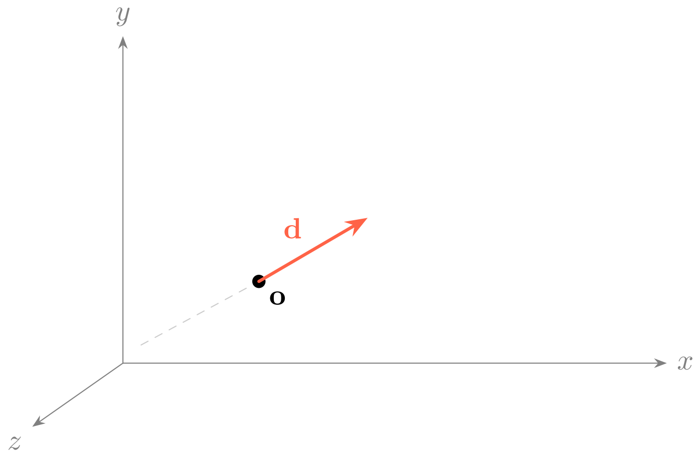



- A piece of the line
- $t = 0$ at $\mathbf{a}$, $t = 1$ at $\mathbf{b}$
- $t = 0.5$: the midpoint

$$\mathbf{p}(t) = \mathbf{a} + t(\mathbf{b} - \mathbf{a})$$

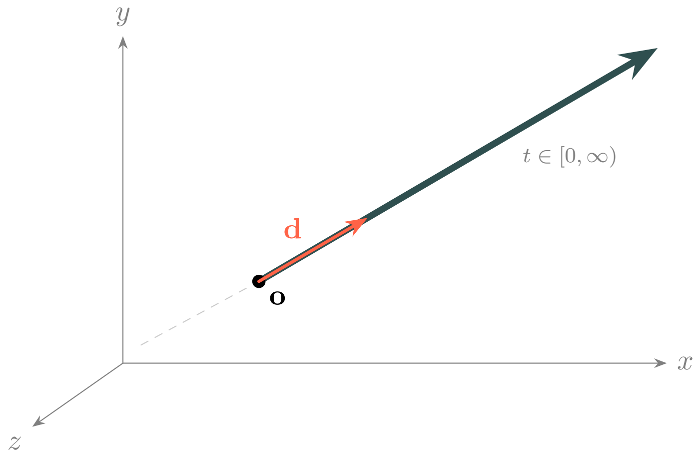
$$\mathbf{p}(0.5) = \frac{\mathbf{a} + \mathbf{b}}{2}$$

# Ray: $t$ from 0 to infinity




- A half-line: one end only

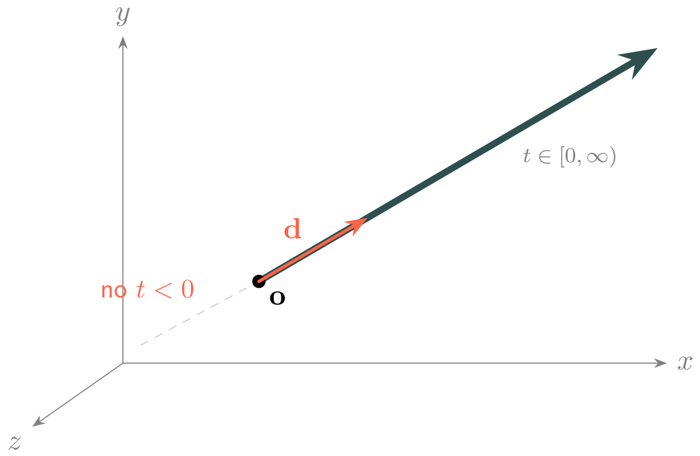
# Ray: $t$ from 0 to infinity




- A half-line: one end only
- Starts at $\mathbf{o}$, never stops

$$\mathbf{p}(t) = \mathbf{o} + t\mathbf{d}, \quad t \geq 0$$

# Ray: $t$ from 0 to infinity




- A half-line: one end only
- Starts at $\mathbf{o}$, never stops
- Behind $\mathbf{o}$ does not count

$$\mathbf{p}(t) = \mathbf{o} + t\mathbf{d}, \quad t \geq 0$$

# The Ray3 class: `at(t)` returns $\mathbf{o} + t\mathbf{d}$

```python
class Ray3:
    """origin + t * direction."""

    def __init__(self, origin: Vec3,
                 direction: Vec3) -> None:
        self.origin = origin
        self.direction = direction.normalized()

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

- Stores the origin and direction

# The Ray3 class: `at(t)` returns $\mathbf{o} + t\mathbf{d}$

```python
class Ray3:
    """origin + t * direction."""

    def __init__(self, origin: Vec3,
                 direction: Vec3) -> None:
        self.origin = origin
        self.direction = direction.normalized()

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

- Stores the origin and direction
- `at(t)` is the whole formula

$$\mathbf{P}(t) = \mathbf{A} + t\,\mathbf{b}$$

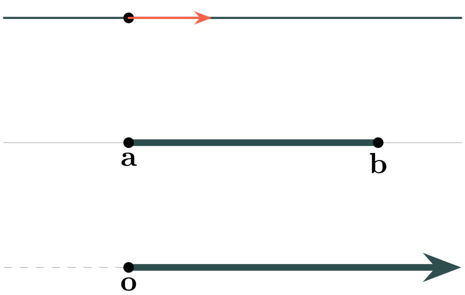
# The Ray3 class: `at(t)` returns $\mathbf{o} + t\mathbf{d}$

```python
class Ray3:
    """origin + t * direction."""

    def __init__(self, origin: Vec3,
                 direction: Vec3) -> None:
        self.origin = origin
        self.direction = direction.normalized()

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

- Stores the origin and direction
- `at(t)` is the whole formula
- Every ray tracer has this class

$$\mathbf{P}(t) = \mathbf{A} + t\,\mathbf{b}$$

# Recap: one formula, three intervals




line: $t \in \mathbb{R}$

segment: $t \in [0, 1]$

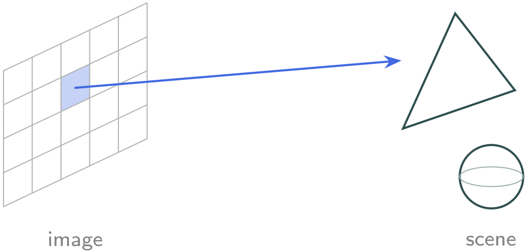
ray: $t \in [0, \infty)$

$$\mathbf{p}(t) = \mathbf{o} + t\,\mathbf{d}$$

# The basic ray-tracing algorithm

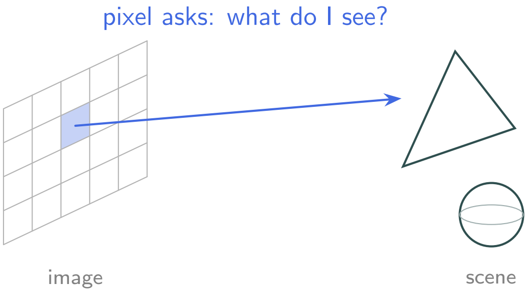
One ray per pixel, nearest hit wins

# Start here: no heavy machinery needed



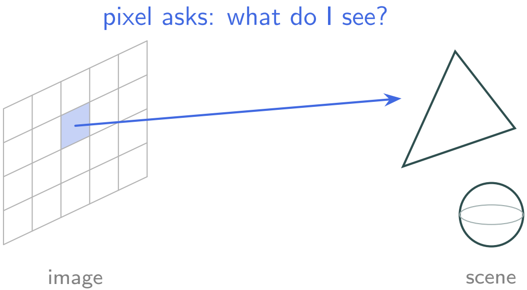

- Loop over pixels, not objects

# Start here: no heavy machinery needed



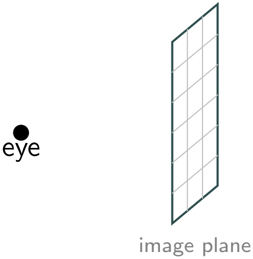

- Loop over pixels, not objects
- Each pixel asks its own question

# Start here: no heavy machinery needed



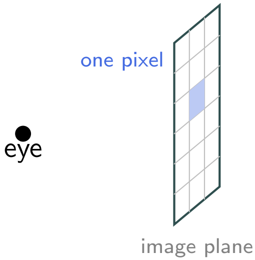

- Loop over pixels, not objects
- Each pixel asks its own question
- Works with just the ray formula

# Each pixel looks in one direction



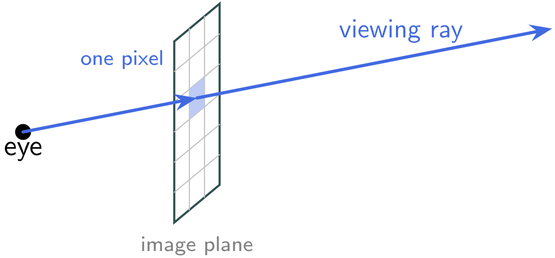

- Eye behind an image plane

# Each pixel looks in one direction



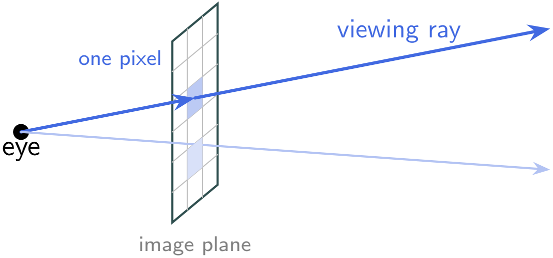

- Eye behind an image plane
- Pick one pixel

# Each pixel looks in one direction



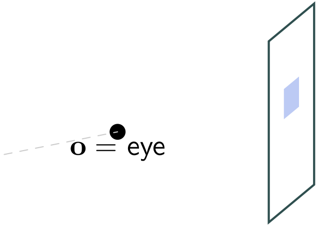

- Eye behind an image plane
- Pick one pixel
- Eye through pixel: a ray

# Each pixel looks in one direction



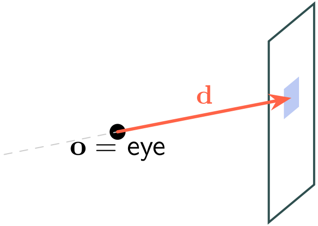

- Eye behind an image plane
- Pick one pixel
- Eye through pixel: a ray
- Other pixel, other direction

# The viewing ray starts at the eye



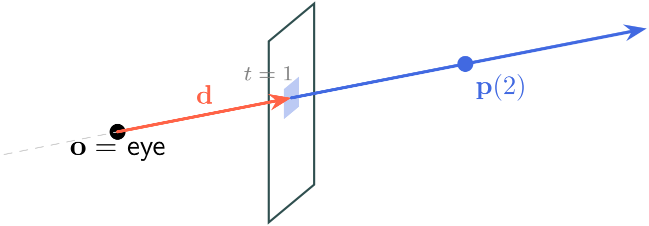

- Origin: the eye

# The viewing ray starts at the eye




- Origin: the eye
- Direction: eye to pixel

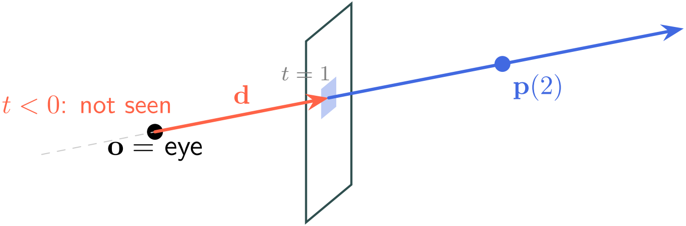
# The viewing ray starts at the eye




- Origin: the eye
- Direction: eye to pixel
- $t$ grows into the scene

$$\mathbf{p}(t) = \mathbf{o} + t\mathbf{d}, \quad t \geq 0$$

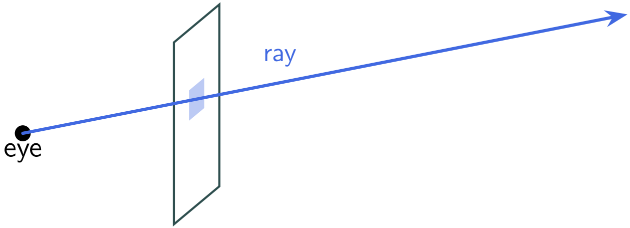
# The viewing ray starts at the eye




- Origin: the eye
- Direction: eye to pixel
- $t$ grows into the scene
- Behind the eye: negative $t$

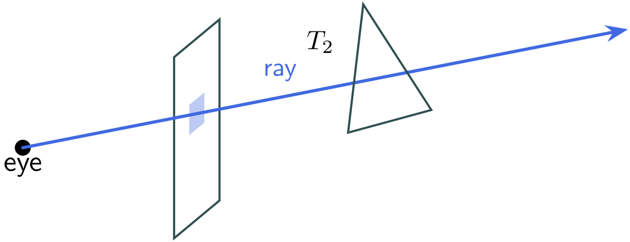
$$\mathbf{p}(t) = \mathbf{o} + t\mathbf{d}, \quad t \geq 0$$

# The ray meets two triangles



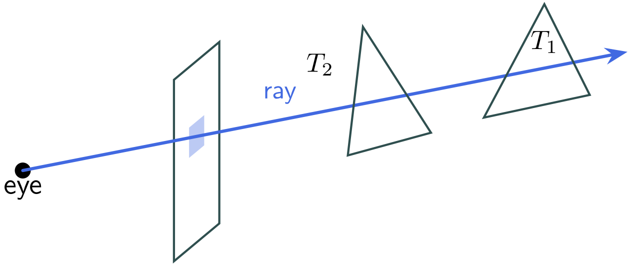

- Send the ray into the scene

# The ray meets two triangles



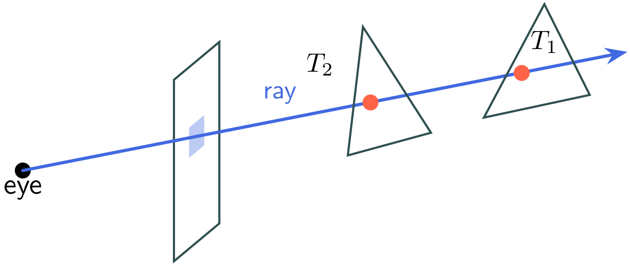

- Send the ray into the scene
- It crosses one triangle

# The ray meets two triangles



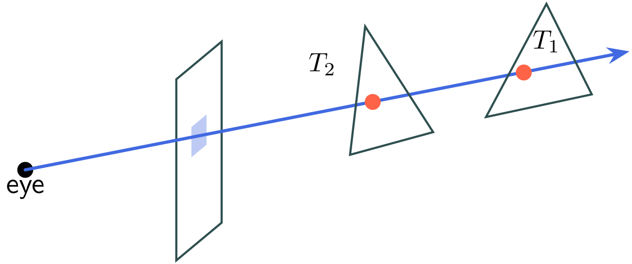

- Send the ray into the scene
- It crosses one triangle
- And another, further on

# The ray meets two triangles



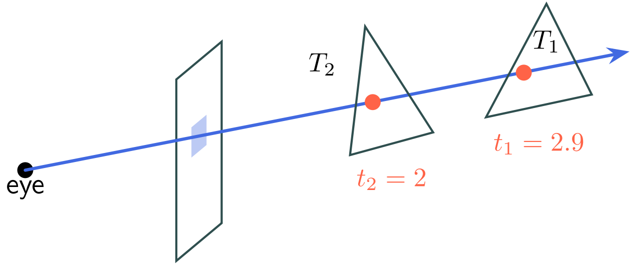

- Send the ray into the scene
- It crosses one triangle
- And another, further on
- Two hits on one ray

# Nearest hit wins: the smallest $t$



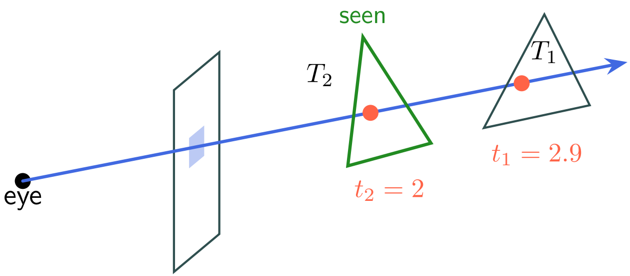

- Each hit has its own $t$

# Nearest hit wins: the smallest $t$



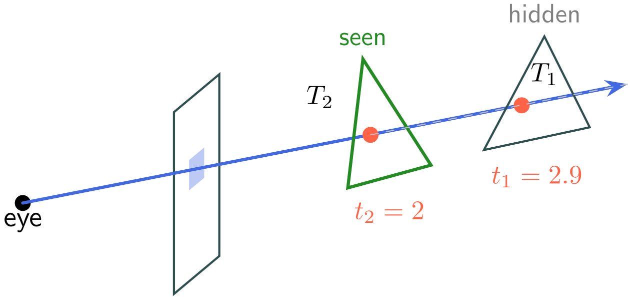

- Each hit has its own $t$
- Smaller $t$: closer to the eye

# Nearest hit wins: the smallest $t$



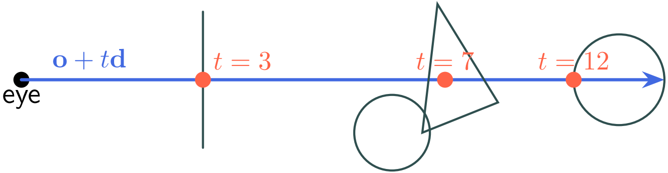

- Each hit has its own $t$
- Smaller $t$: closer to the eye
- $T_2$ is what the pixel shows

# Nearest hit wins: the smallest $t$



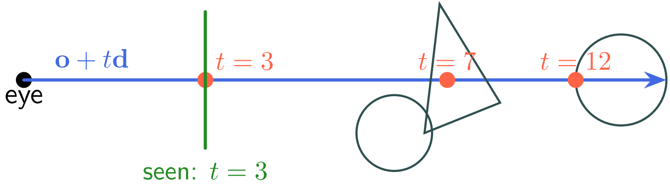

- Each hit has its own $t$
- Smaller $t$: closer to the eye
- $T_2$ is what the pixel shows
- $T_1$ is blocked behind it

# Quick check 2: three hits, which is seen?



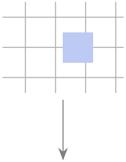

- Hits at $t = 7, 3, 12$
- Which object does the pixel show?

# Quick check 2: three hits, which is seen?




- Hits at $t = 7, 3, 12$
- Which object does the pixel show?
- Smallest $t$: the wall at 3

# Three parts: generate, intersect, shade

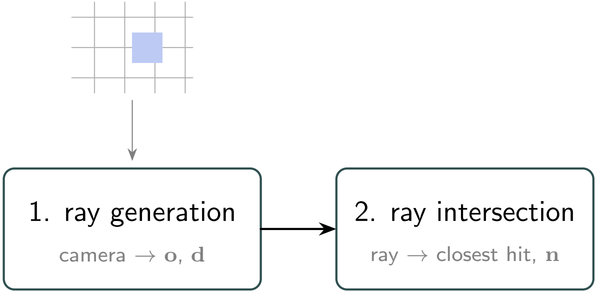


1. ray generation

camera $\rightarrow \mathbf{o}, \mathbf{d}$

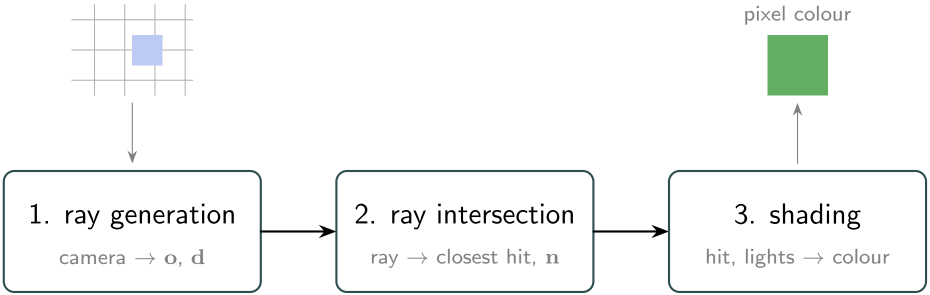
- From the pixel, make its ray

# Three parts: generate, intersect, shade



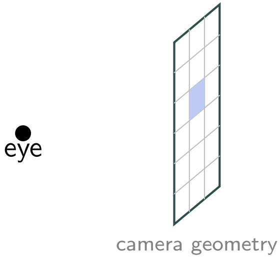

- From the pixel, make its ray
- Find the first thing it hits

# Three parts: generate, intersect, shade



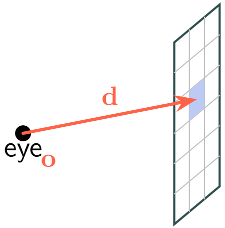

- From the pixel, make its ray
- Find the first thing it hits
- Turn that hit into a colour

# Part 1: ray generation from the camera




- In: eye and image plane

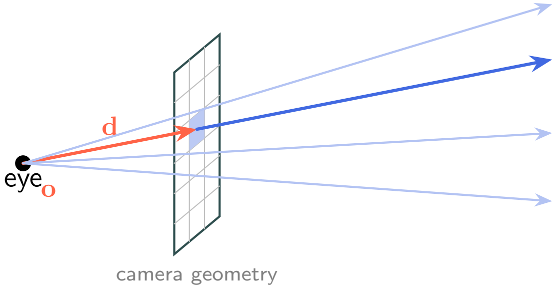
# Part 1: ray generation from the camera




camera geometry

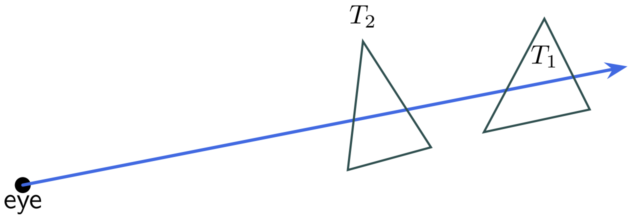
- In: eye and image plane
- Out: $\mathbf{o}$ and $\mathbf{d}$ for this pixel

# Part 1: ray generation from the camera



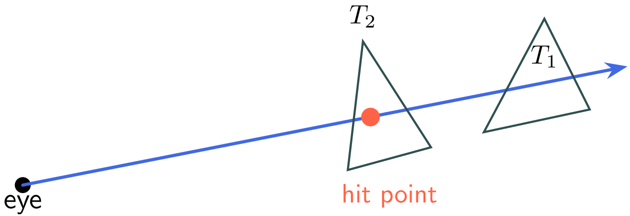

- In: eye and image plane
- Out: $\mathbf{o}$ and $\mathbf{d}$ for this pixel
- Repeat for every pixel

# Part 2: intersection returns the hit and $\mathbf{n}$



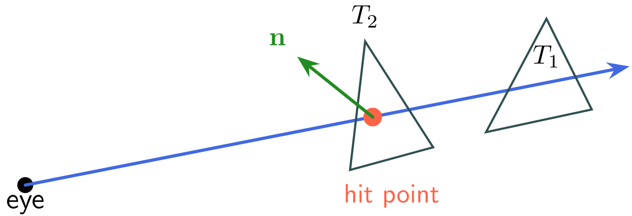

- In: one ray, all objects

# Part 2: intersection returns the hit and $\mathbf{n}$



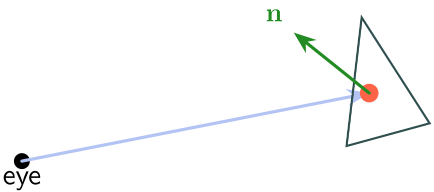

- In: one ray, all objects
- Out: the closest hit point

# Part 2: intersection returns the hit and $\mathbf{n}$



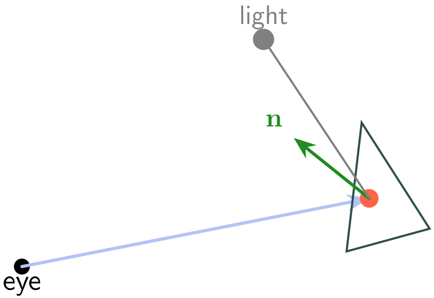

- In: one ray, all objects
- Out: the closest hit point
- Plus the surface normal there

# Part 3: shading turns the hit into colour



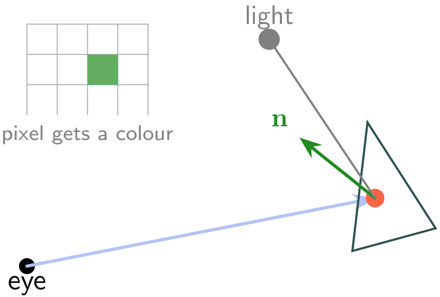

- In: hit point and $\mathbf{n}$

# Part 3: shading turns the hit into colour




- In: hit point and $\mathbf{n}$
- Plus where the lights are

# Part 3: shading turns the hit into colour




- In: hit point and $\mathbf{n}$
- Plus where the lights are
- Out: one colour for the pixel

# The whole program: a loop over pixels

```
1. for each pixel do
2.    compute viewing ray
3.    find first object hit by ray
         and its surface normal n
4.    set pixel color to value computed
         from hit point, lights, and n
```

- Four lines, whole algorithm

# The whole program: a loop over pixels

```
1. for each pixel do
2.    compute viewing ray
3.    find first object hit by ray
         and its surface normal n
4.    set pixel color to value computed
         from hit point, lights, and n
```

- Four lines, whole algorithm
- Line 2: generate

# The whole program: a loop over pixels

```
1. for each pixel do
2.    compute viewing ray
3.    find first object hit by ray
         and its surface normal n
4.    set pixel color to value computed
         from hit point, lights, and n
```

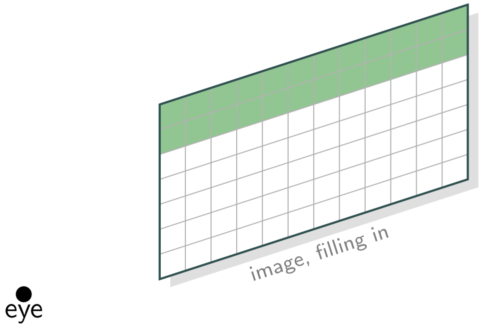
- Four lines, whole algorithm
- Line 2: generate
- Line 3: intersect

# The whole program: a loop over pixels

```
1. for each pixel do
2.    compute viewing ray
3.    find first object hit by ray
         and its surface normal n
4.    set pixel color to value computed
         from hit point, lights, and n
```

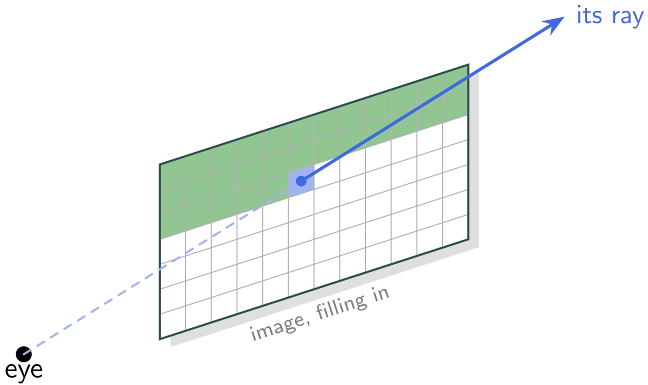
- Four lines, whole algorithm
- Line 2: generate
- Line 3: intersect
- Line 4: shade

# One pixel at a time, left to right



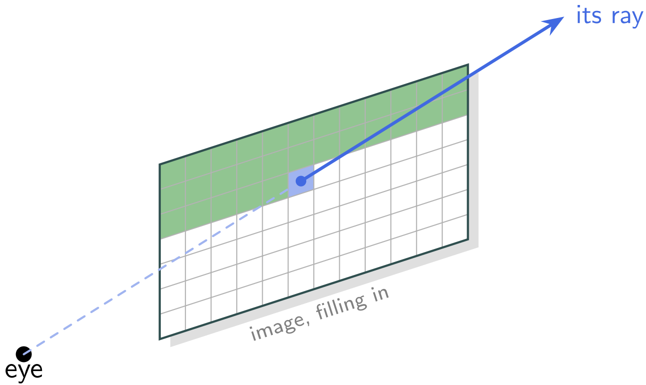

- Done pixels are green

# One pixel at a time, left to right




- Done pixels are green
- The current pixel sends its ray

# One pixel at a time, left to right

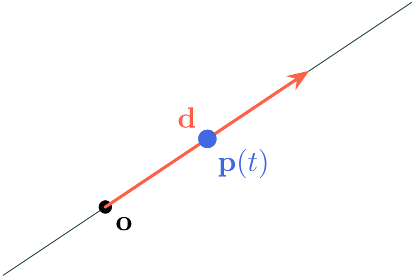



- Done pixels are green
- The current pixel sends its ray
- The rest wait their turn

# Recap: for each pixel, a ray, a hit, a colour


smallest $t \geq 0$ wins

# Summary




$$\mathbf{p}(t) = \mathbf{o} + t\mathbf{d}$$

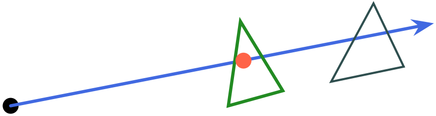
a point, a number, an arrow

line: all $t$

segment: $[0, 1]$

ray: $[0, \infty)$

same formula, three ranges of $t$

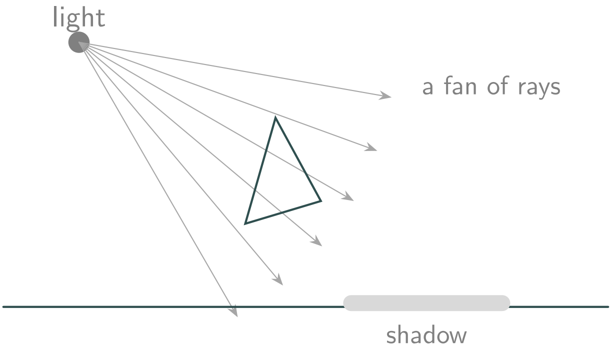


nearest hit: smallest $t \geq 0$

generate → intersect → shade

for each pixel, three parts

# Next lecture: light and shadow




Point lights, and rays that tell us who is in shadow

Marschner & Shirley §4.5.1, §4.5.3