Lecture 31: The Shadow Ray Returns

A Ray from Every Pixel, against Suzanne's Own Triangles

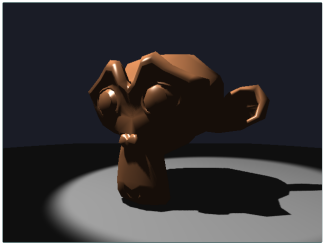
CS 453 — Computer Graphics

Fakhir Shaheen

Forman Christian University

Lecture 30 read a picture; today we cast the ray

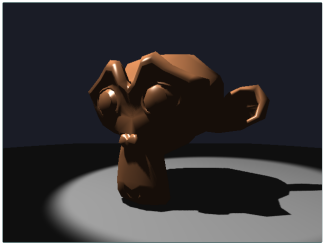
lecture 30: the shadow map



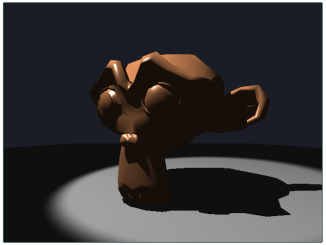
- Lecture 30: each point reads one texel of the spot's picture.

Lecture 30 read a picture; today we cast the ray

lecture 30: the shadow map



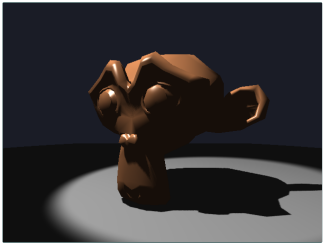
today: a shadow ray per pixel



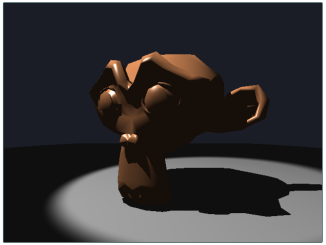
- Lecture 30: each point reads one texel of the spot's picture.
- Today: each point sends lecture 17's ray to the spot.

Lecture 30 read a picture; today we cast the ray

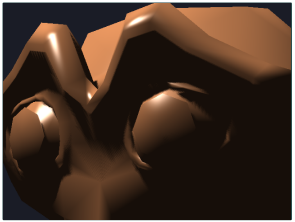
lecture 30: the shadow map

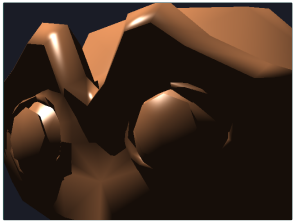


today: a shadow ray per pixel



close up: her brow

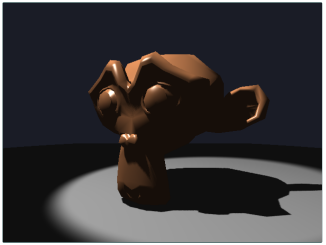




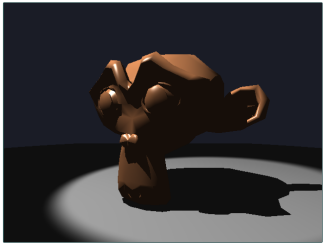
- Lecture 30: each point reads one texel of the spot's picture.
- Today: each point sends lecture 17's ray to the spot.
- Close up: the map stripes and blurs. The ray is sharp.

Lecture 30 read a picture; today we cast the ray

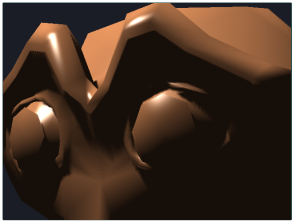
lecture 30: the shadow map

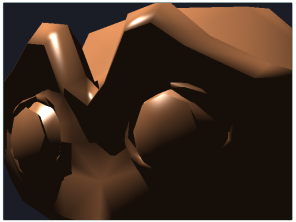


today: a shadow ray per pixel



close up: her brow

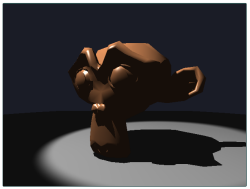




- Lecture 30: each point reads one texel of the spot's picture.
- Today: each point sends lecture 17's ray to the spot.
- Close up: the map stripes and blurs. The ray is sharp.
- The price: about 78 million triangle tests a frame.

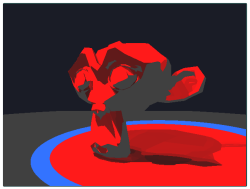
Today: three rounds of slides, then you try

1 **A shadow ray in fs_main**



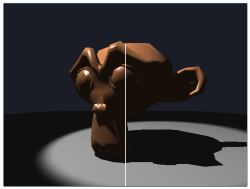
then You Try 1, 2

2 **Trace only what you must**



then You Try 3, 4

3 **The map against the ray**



then You Try 5

Open the Ray Lab now. It is on the course calendar, in lecture 31's row.

It is lecture 30's Shadow Lab, plus Suzanne's triangles for fs_main to read.

It shows how many milliseconds your GPU takes a frame: watch it.

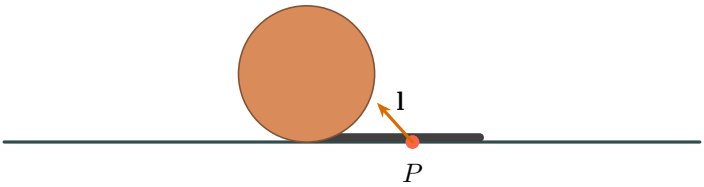
Round 1

A shadow ray in fs_main

Lecture 17's ray, sent by every pixel

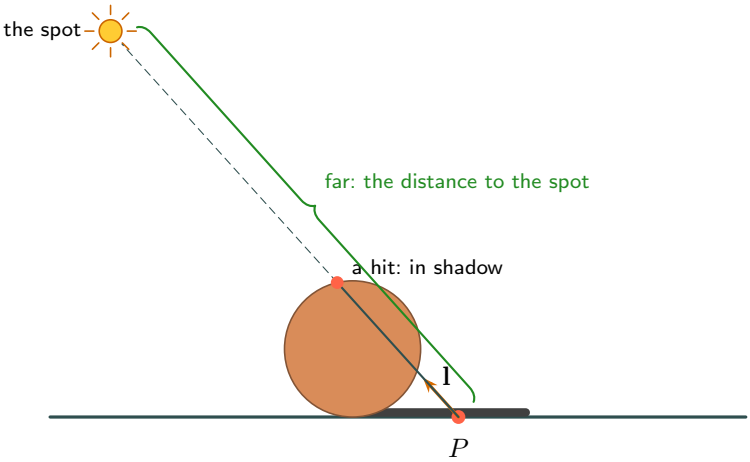
Lecture 17's ray, now in fs_main





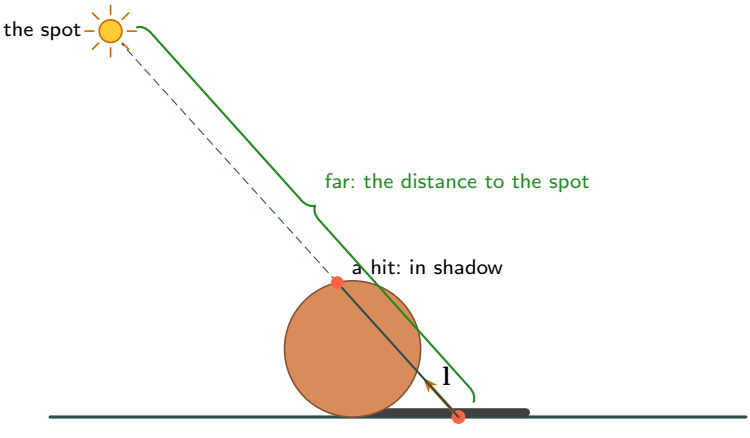
- fs_main already knows P and $\mathbf{l}$, toward the spot (lecture 29).

Lecture 17's ray, now in fs_main



- fs_main already knows P and $\mathbf{l}$, toward the spot (lecture 29).
- Lecture 17 (slides 7, 12): go from P along $\mathbf{l}$; any hit before the lamp means shadow.

Lecture 17's ray, now in fs_main

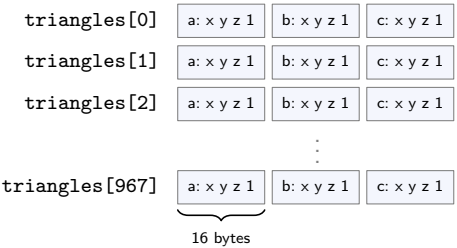


- fs_main already knows P and $\mathbf{l}$, toward the spot (lecture 29).
- Lecture 17 (slides 7, 12): go from P along $\mathbf{l}$; any hit before the lamp means shadow.
- Missing: the scene. fs_main has never seen her triangles.

```
lecture 17:  srec = scene.hit(Ray(x, l), 0.00001, r)
today:       hits_triangle(P, l, eps, far, a, b, c)  // for each of her triangles
```

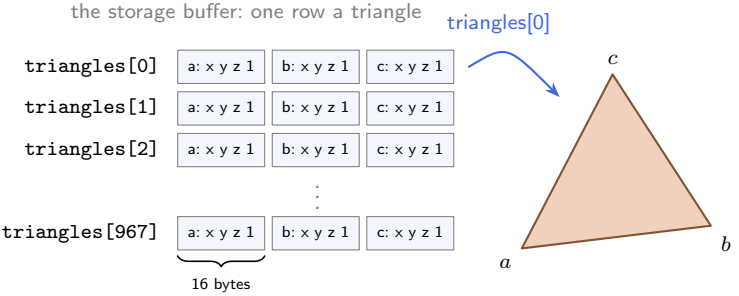
Her triangles go to the GPU once: a storage buffer

the storage buffer: one row a triangle



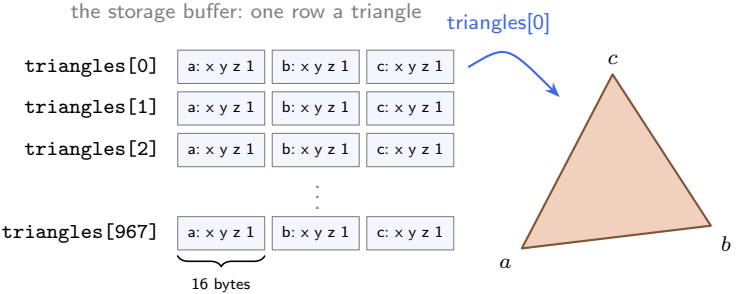
- rays.py copies all 968 triangles into one buffer, once.

Her triangles go to the GPU once: a storage buffer



- rays.py copies all 968 triangles into one buffer, once.
- 968 $\times$ 3 corners $\times$ 16 bytes $=$ 46,464 bytes.

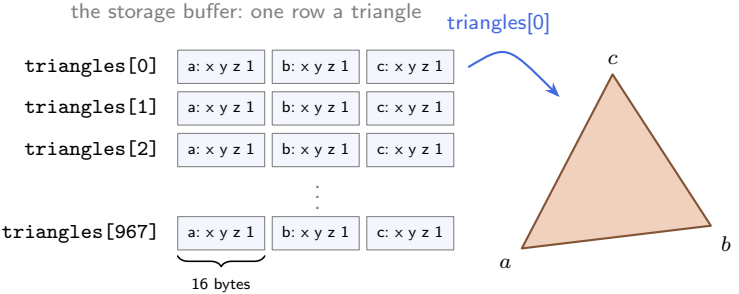
Her triangles go to the GPU once: a storage buffer




```
struct Triangle { a: vec4f, b: vec4f, c: vec4f };  // given
@group(0) @binding(2) var<storage, read>
    triangles: array<Triangle>;
```

- rays.py copies all 968 triangles into one buffer, once.
- 968 $\times$ 3 corners $\times$ 16 bytes $=$ 46,464 bytes.
- Binding 2, beside u and the shadow map.

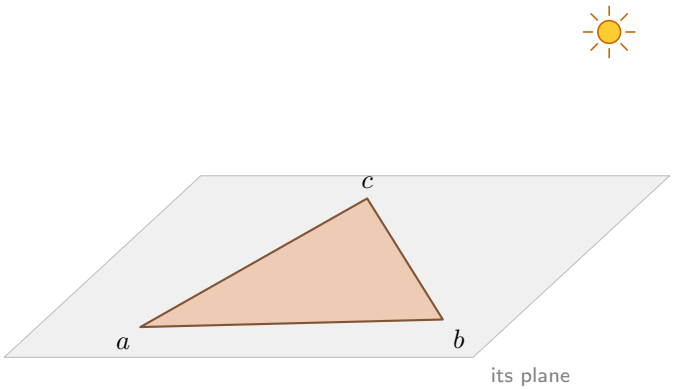
Her triangles go to the GPU once: a storage buffer




```
struct Triangle { a: vec4f, b: vec4f, c: vec4f };  // given
@group(0) @binding(2) var<storage, read>
    triangles: array<Triangle>;
fn world(corner: vec4f) -> vec3f {  // new
    let q = u.model * corner;  // as in vs_main
    return vec3f(q.x, q.y, q.z);
}
```

- rays.py copies all 968 triangles into one buffer, once.
- 968 $\times$ 3 corners $\times$ 16 bytes $=$ 46,464 bytes.
- Binding 2, beside `u` and the shadow map.
- The corners are Suzanne's own: `world` turns each by M.

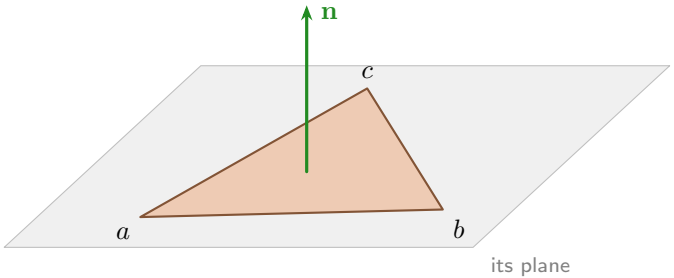
Step 1: where does the ray meet the triangle's plane?



- The ray leaves P toward the spot. Does it pass through this triangle?

Step 1: where does the ray meet the triangle's plane?

$$\mathbf{n} = (b - a) \times (c - a)$$

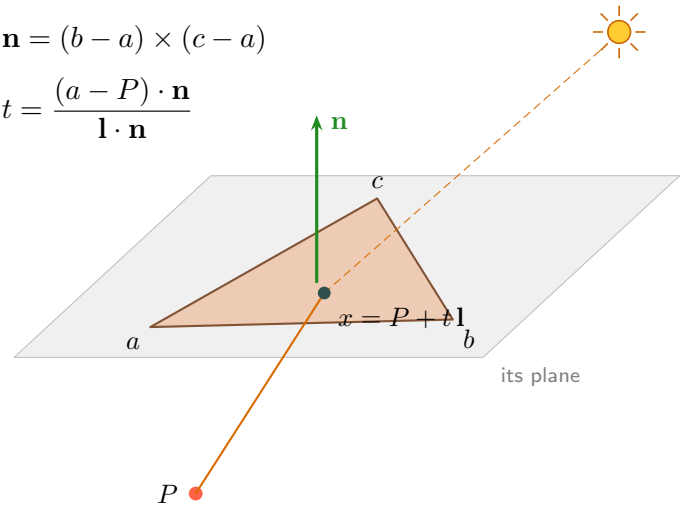


- The ray leaves P toward the spot. Does it pass through this triangle?
- $\mathbf{n}$: perpendicular to the triangle, from two of its edges.

Step 1: where does the ray meet the triangle's plane?

$$\mathbf{n} = (b - a) \times (c - a)$$

$$t = \frac{(a - P) \cdot \mathbf{n}}{\mathbf{l} \cdot \mathbf{n}}$$

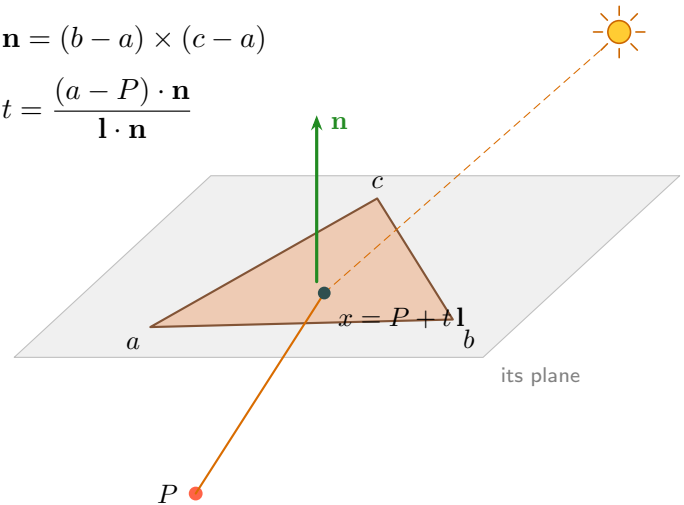


- The ray leaves P toward the spot. Does it pass through this triangle?
- $\mathbf{n}$: perpendicular to the triangle, from two of its edges.
- Lecture 14, slide 38: the ray meets a plane. Its $\mathbf{e}$ is our P, its $\mathbf{d}$ our $\mathbf{l}$.

Step 1: where does the ray meet the triangle's plane?

$$\mathbf{n} = (b - a) \times (c - a)$$

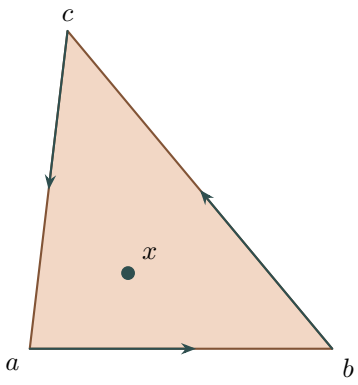
$$t = \frac{(a - P) \cdot \mathbf{n}}{\mathbf{l} \cdot \mathbf{n}}$$



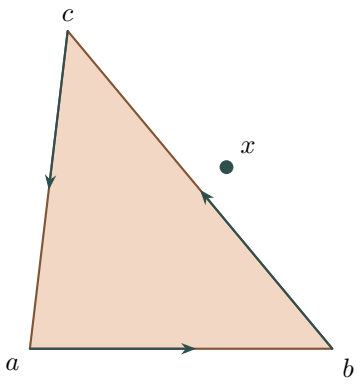
- The ray leaves P toward the spot. Does it pass through this triangle?
- $\mathbf{n}$: perpendicular to the triangle, from two of its edges.
- Lecture 14, slide 38: the ray meets a plane. Its $\mathbf{e}$ is our P, its $\mathbf{d}$ our $\mathbf{l}$.
- Keep only t between eps and far. $\mathbf{l} \cdot \mathbf{n} = 0$: no hit.

Step 2: is x inside? All three edges must agree

seen from above: $\mathbf{n}$ points at you



inside

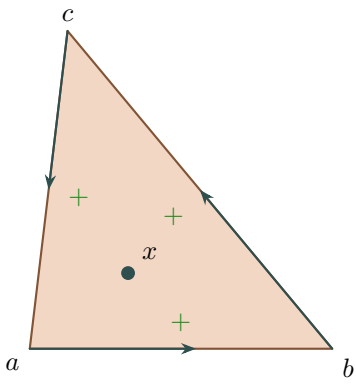


outside

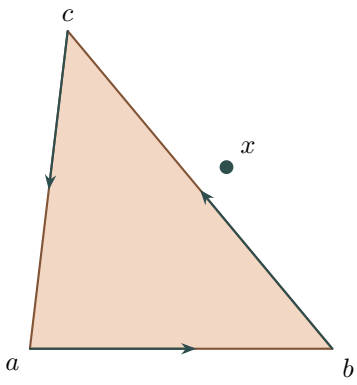
- x is on the plane. Is it inside the triangle?

Step 2: is x inside? All three edges must agree

seen from above: $\mathbf{n}$ points at you



inside

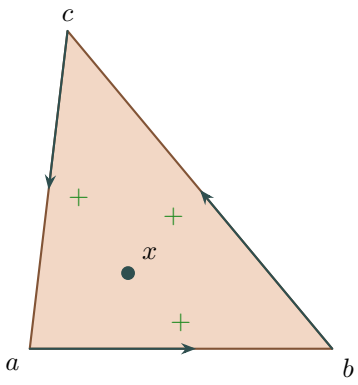


outside

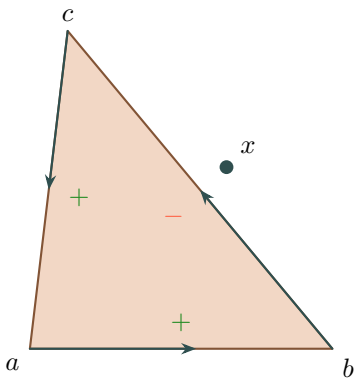
- x is on the plane. Is it inside the triangle?
- Lecture 24, slide 26: inside when all three edges give the same sign.

Step 2: is x inside? All three edges must agree

seen from above: $\mathbf{n}$ points at you



inside

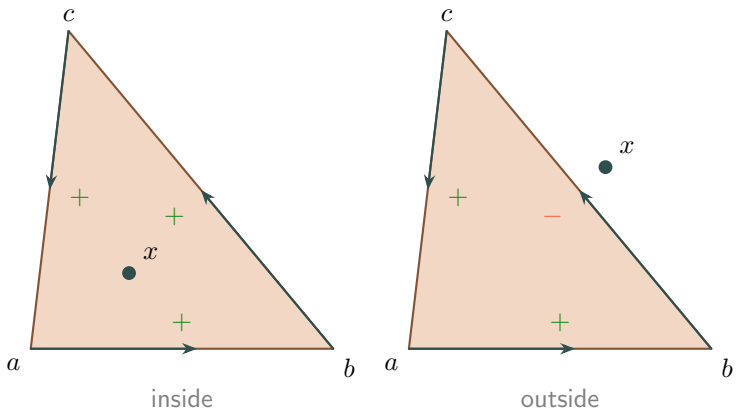


outside

- x is on the plane. Is it inside the triangle?
- Lecture 24, slide 26: inside when all three edges give the same sign.
- One edge disagrees: x is outside. No hit.

Step 2: is x inside? All three edges must agree

seen from above: $\mathbf{n}$ points at you



- x is on the plane. Is it inside the triangle?
- Lecture 24, slide 26: inside when all three edges give the same sign.
- One edge disagrees: x is outside. No hit.
- In 3D, the sign: the cross product along $\mathbf{n}$ or against it.

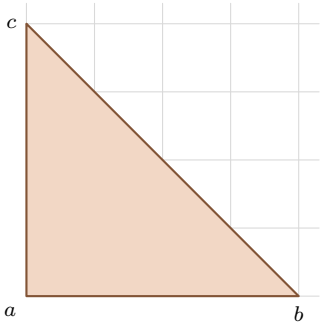
```
if (dot(cross(b - a, x - a), n) < 0.0) { return false; }  // and two more
```

Worked: does each ray meet this triangle?

$a = (0, 0, 0)$, $b = (4, 0, 0)$, $c = (0, 4, 0)$; the spot along $-z$: $\mathbf{l} = (0, 0, -1)$

Two rays: from $P = (1, 1, 3)$ and from $Q = (3, 3, 3)$.

Try first: find t and x, then the three edge signs.



seen from the spot

Worked: does each ray meet this triangle?

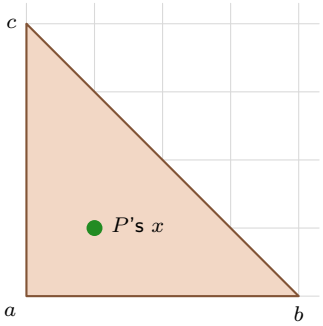
$a = (0,0,0)$, $b = (4,0,0)$, $c = (0,4,0)$; the spot along $-z$: $\mathbf{l} = (0,0,-1)$

Two rays: from $P = (1,1,3)$ and from $Q = (3,3,3)$.

Try first: find t and x, then the three edge signs.

$\mathbf{n} = (4,0,0) \times (0,4,0) = (0,0,16)$

P: $t = \dfrac{(-1,-1,-3)\cdot\mathbf{n}}{-16} = \dfrac{-48}{-16} = 3$, $x = (1,1,0)$



seen from the spot

Worked: does each ray meet this triangle?

$a = (0,0,0)$, $b = (4,0,0)$, $c = (0,4,0)$; the spot along $-z$: $\mathbf{l} = (0,0,-1)$

Two rays: from $P = (1,1,3)$ and from $Q = (3,3,3)$.

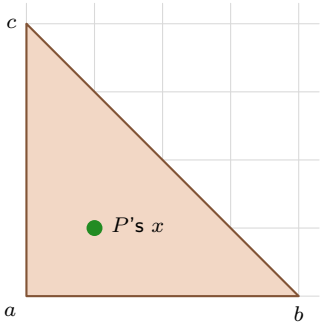
Try first: find t and x, then the three edge signs.

$\mathbf{n} = (4,0,0) \times (0,4,0) = (0,0,16)$

P: $t = \dfrac{(-1,-1,-3)\cdot\mathbf{n}}{-16} = \dfrac{-48}{-16} = 3$, $x = (1,1,0)$

edges: $(0,0,4)\cdot\mathbf{n} = 64$, $(0,0,8)\cdot\mathbf{n} = 128$, $(0,0,4)\cdot\mathbf{n} = 64$: **hit**

P is in this triangle's shadow.



seen from the spot

Worked: does each ray meet this triangle?

$a = (0,0,0)$, $b = (4,0,0)$, $c = (0,4,0)$; the spot along $-z$: $\mathbf{l} = (0,0,-1)$

Two rays: from $P = (1,1,3)$ and from $Q = (3,3,3)$.

Try first: find t and x, then the three edge signs.

$\mathbf{n} = (4,0,0) \times (0,4,0) = (0,0,16)$

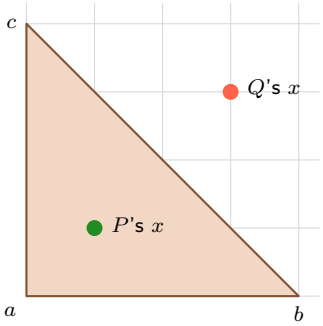
P: $t = \dfrac{(-1,-1,-3)\cdot\mathbf{n}}{-16} = \dfrac{-48}{-16} = 3$, $x = (1,1,0)$

edges: $(0,0,4)\cdot\mathbf{n} = 64$, $(0,0,8)\cdot\mathbf{n} = 128$, $(0,0,4)\cdot\mathbf{n} = 64$: **hit**

P is in this triangle's shadow.

Q: $t = 3$, $x = (3,3,0)$; edge $b \to c$: $(0,0,-8)\cdot\mathbf{n} = -128$

One edge disagrees: **no hit**. Q's ray passes beside it.



seen from the spot

The two steps, in WGSL: hits_triangle

```
fn hits_triangle(start: vec3f, dir: vec3f, eps: f32, far: f32,
                 a: vec3f, b: vec3f, c: vec3f) -> bool {




}
```

- A function of our own, as `turn_y` was in lecture 28.

The two steps, in WGSL: hits_triangle

```
fn hits_triangle(start: vec3f, dir: vec3f, eps: f32, far: f32,
                 a: vec3f, b: vec3f, c: vec3f) -> bool {
    // Step 1: where does the ray meet the plane? (lecture 14)
    let n = cross(b - a, c - a);
    let d_dot_n = dot(dir, n);
    if (d_dot_n == 0.0) { return false; }  // along the plane
    let t = dot(a - start, n) / d_dot_n;
    if (t < eps || t > far) { return false; }
    let x = start + t * dir;




}
```

- A function of our own, as `turn_y` was in lecture 28.
- Slide 7, line by line.

The two steps, in WGSL: hits_triangle

```
fn hits_triangle(start: vec3f, dir: vec3f, eps: f32, far: f32,
                 a: vec3f, b: vec3f, c: vec3f) -> bool {
    // Step 1: where does the ray meet the plane? (lecture 14)
    let n = cross(b - a, c - a);
    let d_dot_n = dot(dir, n);
    if (d_dot_n == 0.0) { return false; }  // along the plane
    let t = dot(a - start, n) / d_dot_n;
    if (t < eps || t > far) { return false; }
    let x = start + t * dir;
    // Step 2: is x inside? All three edges agree (lecture 24).
    if (dot(cross(b - a, x - a), n) < 0.0) { return false; }
    if (dot(cross(c - b, x - b), n) < 0.0) { return false; }
    if (dot(cross(a - c, x - c), n) < 0.0) { return false; }
    return true;
}
```

- A function of our own, as `turn_y` was in lecture 28.
- Slide 7, line by line.
- Slide 8, line by line.

The two steps, in WGSL: hits_triangle

```
fn hits_triangle(start: vec3f, dir: vec3f, eps: f32, far: f32,
                 a: vec3f, b: vec3f, c: vec3f) -> bool {
    // Step 1: where does the ray meet the plane? (lecture 14)
    let n = cross(b - a, c - a);
    let d_dot_n = dot(dir, n);
    if (d_dot_n == 0.0) { return false; }  // along the plane
    let t = dot(a - start, n) / d_dot_n;
    if (t < eps || t > far) { return false; }
    let x = start + t * dir;
    // Step 2: is x inside? All three edges agree (lecture 24).
    if (dot(cross(b - a, x - a), n) < 0.0) { return false; }
    if (dot(cross(c - b, x - b), n) < 0.0) { return false; }
    if (dot(cross(a - c, x - c), n) < 0.0) { return false; }
    return true;
}
```

cross, dot: built into WGSL. || means "or".
Every test that fails returns at once: most triangles stop after a line or two.

- A function of our own, as `turn_y` was in lecture 28.
- Slide 7, line by line.
- Slide 8, line by line.
- true: the ray meets this triangle between eps and far.

The loop: try all 968, stop at the first hit

```
let eps = 0.001;                // start a little way along (lecture 17, slide 10)
let far = length(u.spot - P);   // stop at the spot
var lit = 1.0;
```

- eps and far: lecture 17's interval $[\epsilon, r]$.

```
let c = base * (ambient + lit * k * diffuse) + lit * k * shine;
```

▶ **In the Ray Lab:**
The shadow ray from every point

The loop: try all 968, stop at the first hit

```
let eps = 0.001;                // start a little way along (lecture 17, slide 10)
let far = length(u.spot - P);  // stop at the spot
var lit = 1.0;
for (var i = 0; i < 968; i++) {
    let tri = triangles[i];
    if (hits_triangle(P, l, eps, far,
                  world(tri.a), world(tri.b), world(tri.c))) {



    }
}
let c = base * (ambient + lit * k * diffuse) + lit * k * shine;
```

- eps and far: lecture 17's interval $[\epsilon, r]$.
- Try every triangle, turned into the world.

▶ **In the Ray Lab:**
The shadow ray from every point

The loop: try all 968, stop at the first hit

```
let eps = 0.001;                // start a little way along (lecture 17, slide 10)
let far = length(u.spot - P);  // stop at the spot
var lit = 1.0;
for (var i = 0; i < 968; i++) {
    let tri = triangles[i];
    if (hits_triangle(P, l, eps, far,
                  world(tri.a), world(tri.b), world(tri.c))) {
        lit = 0.0;
        break;  // one is enough: stop looking
    }
}
let c = base * (ambient + lit * k * diffuse) + lit * k * shine;
```

- eps and far: lecture 17's interval $[\epsilon, r]$.
- Try every triangle, turned into the world.
- A shadow needs any hit, not the nearest: `break`.

▶ **In the Ray Lab:**
The shadow ray from every point

The loop: try all 968, stop at the first hit

```
let eps = 0.001;                // start a little way along (lecture 17, slide 10)
let far = length(u.spot - P);   // stop at the spot
var lit = 1.0;
for (var i = 0; i < 968; i++) {
    let tri = triangles[i];
    if (hits_triangle(P, l, eps, far,
                  world(tri.a), world(tri.b), world(tri.c))) {
        lit = 0.0;
        break;  // one is enough: stop looking
    }
}
let c = base * (ambient + lit * k * diffuse) + lit * k * shine;
```

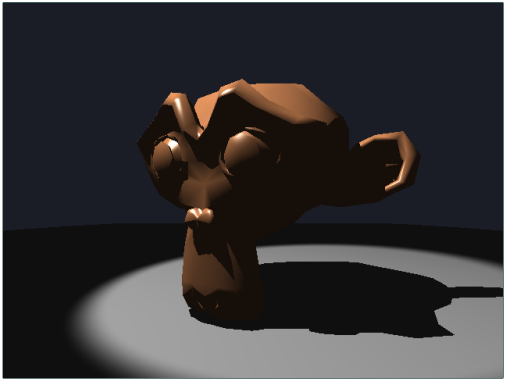
- eps and far: lecture 17's interval $[\epsilon, r]$.
- Try every triangle, turned into the world.
- A shadow needs any hit, not the nearest: `break`.

Lecture 30's $P = (1, -1, -1)$ (its slide 11): the 968 triangles, in order



▶ **In the Ray Lab:**
The shadow ray from every point

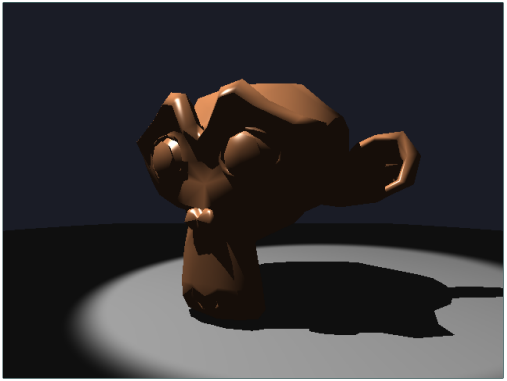
Suzanne shadows herself



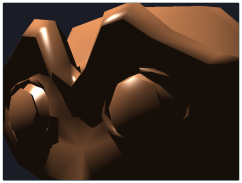
ray_every.wgsl: every point tries all 968 triangles

- Her shadow on the floor, as in lecture 30.

Suzanne shadows herself



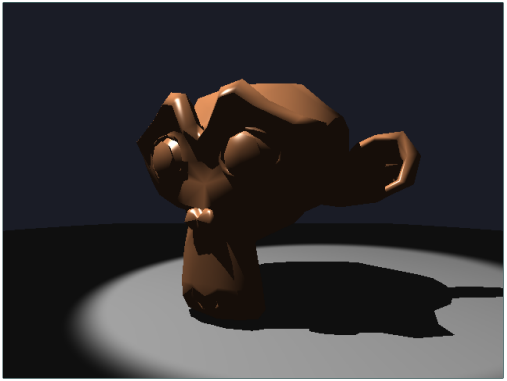
ray_every.wgsl: every point tries all 968 triangles



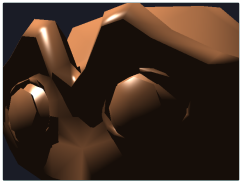
her brow, close up

- Her shadow on the floor, as in lecture 30.
- Her brow shadows her eyes; her nose, her mouth. The edges are exact.

Suzanne shadows herself



ray_every.wgsl: every point tries all 968 triangles



her brow, close up

- Her shadow on the floor, as in lecture 30.
- Her brow shadows her eyes; her nose, her mouth. The edges are exact.
- No map, no texels, no bias: one ray per pixel.

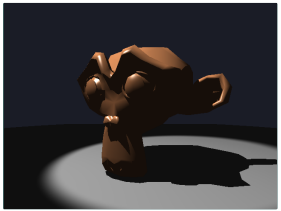
You Try 1: start the ray at the point itself

In the Ray Lab, choose **You Try 1**. Guess first, then change the code.

```
let eps = 0.001;
let far = length(u.spot - P);
```

change this line

now



- **Set eps to 0.0: the ray starts at P itself.**
- Guess first: what happens to her lit face? To the floor?

You Try 1: start the ray at the point itself

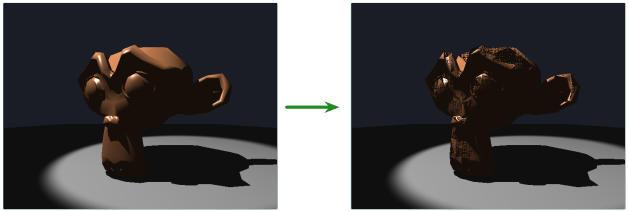
In the Ray Lab, choose **You Try 1**. Guess first, then change the code.

```
let eps = 0.001;
let far = length(u.spot - P);
```

change this line

answer: `let eps = 0.0;`

now



- **Set eps to 0.0: the ray starts at P itself.**
- Guess first: what happens to her lit face? To the floor?
- Speckles on her: rays meet her own triangle at $t \approx 0$ (lecture 17, slide 10). The floor has no triangles: clean.

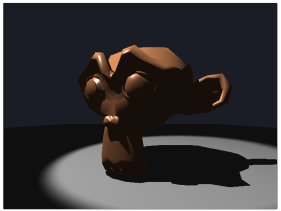
You Try 2: a short ray

In the Ray Lab, choose **You Try 2**. Guess first, then change the code.

```
let eps = 0.001;
let far = length(u.spot - P);
```

change this line

now



- **Set far to 1.0: the ray looks only 1.0 along.**
- Guess first: which shadows stay, and which go?

You Try 2: a short ray

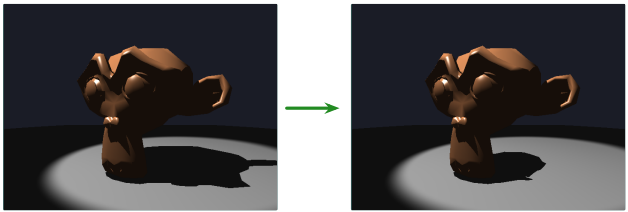
In the Ray Lab, choose **You Try 2**. Guess first, then change the code.

```
let eps = 0.001;
let far = length(u.spot - P);
```

change this line

answer: `let far = 1.0;`

now



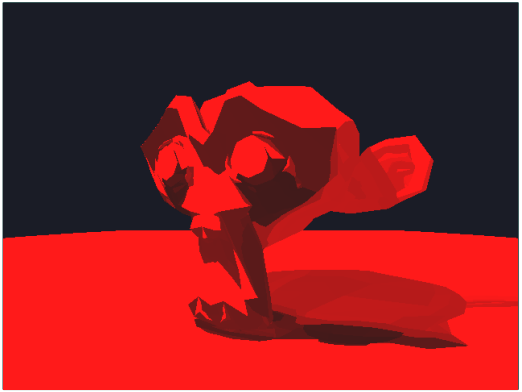
- **Set far to 1.0: the ray looks only 1.0 along.**
- Guess first: which shadows stay, and which go?
- Only near ones stay: under her chin, in her creases. A far floor point's blocker is more than 1.0 away.

Round 2

Trace only what you must

Send a ray only where it can change the picture

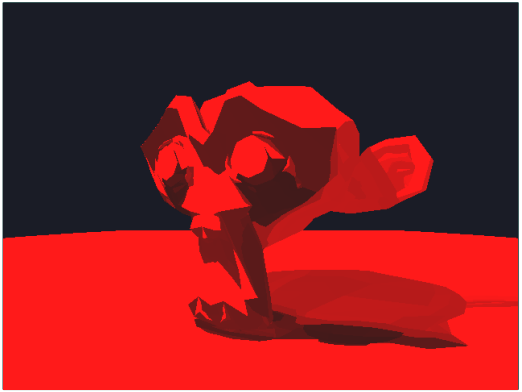
How much work? Every point tries up to 968 triangles



each pixel painted by its own work: brighter red = more triangles tried
darker red: the ray found a hit early and stopped

- Every point Suzanne or the floor covers sends a ray.

How much work? Every point tries up to 968 triangles



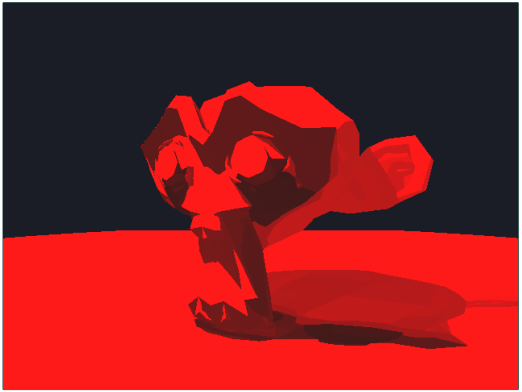
each pixel painted by its own work: brighter red = more triangles tried
darker red: the ray found a hit early and stopped

172,073

rays a frame

- Every point Suzanne or the floor covers sends a ray.
- One ray a pixel, read back from the GPU.

How much work? Every point tries up to 968 triangles



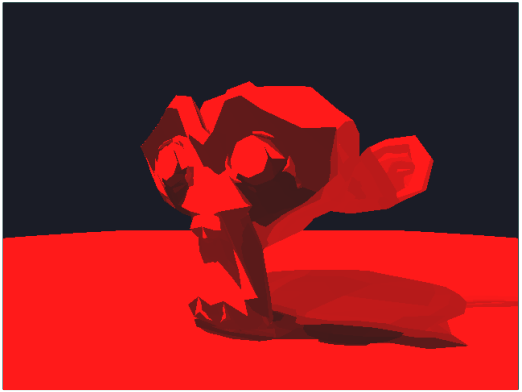
each pixel painted by its own work: brighter red = more triangles tried
darker red: the ray found a hit early and stopped

172,073
rays a frame

131 M
triangle tests a frame

- Every point Suzanne or the floor covers sends a ray.
- One ray a pixel, read back from the GPU.
- Most rays try all 968 triangles. 60 frames a second: 8 billion tests.

How much work? Every point tries up to 968 triangles



each pixel painted by its own work: brighter red = more triangles tried
darker red: the ray found a hit early and stopped

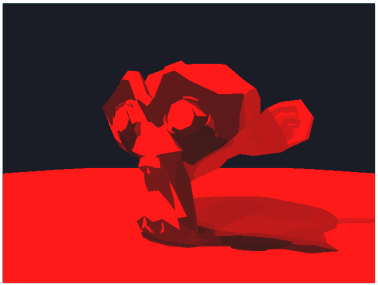
172,073
rays a frame

131 M
triangle tests a frame

- Every point Suzanne or the floor covers sends a ray.
- One ray a pixel, read back from the GPU.
- Most rays try all 968 triangles. 60 frames a second: 8 billion tests.
- Which of these rays can change the picture?

Skip 1: a point the spot does not light needs no ray

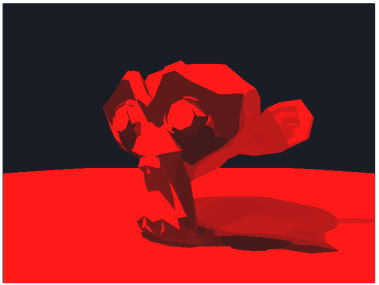
every point: 131 M tests



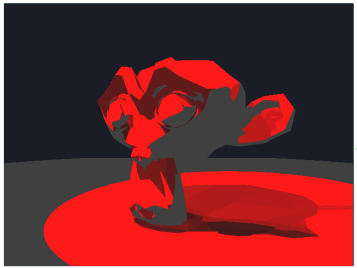
- Outside the cone, or facing away: the spot adds nothing. It is dark anyway.

Skip 1: a point the spot does not light needs no ray

every point: 131 M tests



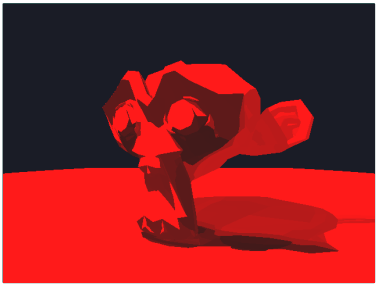
skip 1: 90 M tests



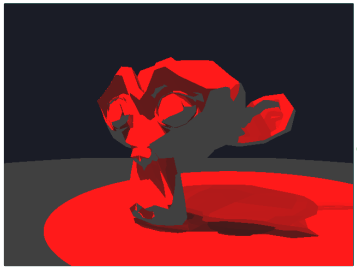
- Outside the cone, or facing away: the spot adds nothing. It is dark anyway.
- Grey: 56,989 points now send no ray. A third of the work, gone.

Skip 1: a point the spot does not light needs no ray

every point: 131 M tests



skip 1: 90 M tests



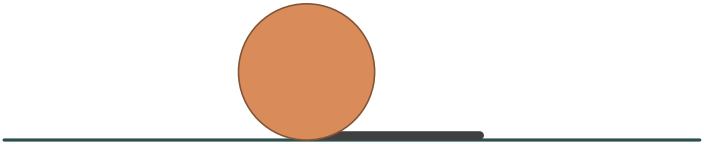
- Outside the cone, or facing away: the spot adds nothing. It is dark anyway.
- Grey: 56,989 points now send no ray. A third of the work, gone.
- The picture does not change at all.

```
if (k > 0.0 && dot(n, l) > 0.0) {  // only points the spot lights
    ... the ray and its loop, as on slide 11 ...
}
```

&& means "and". k: lecture 29's cone; $\mathbf{n} \cdot \mathbf{l}$: does it face the spot?

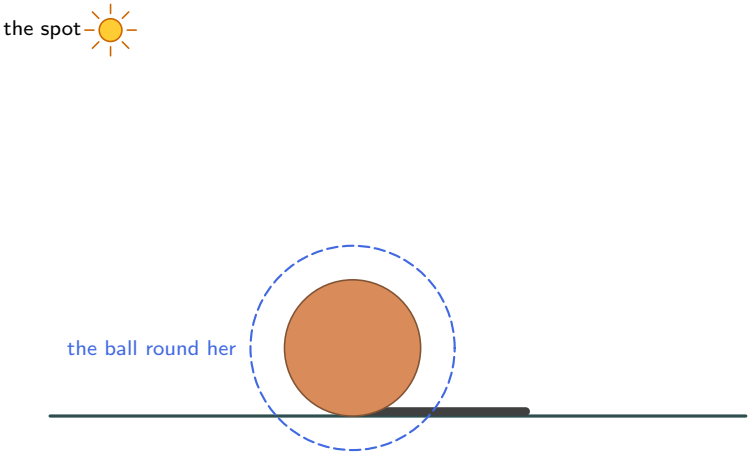
Skip 2: first ask a ball round her





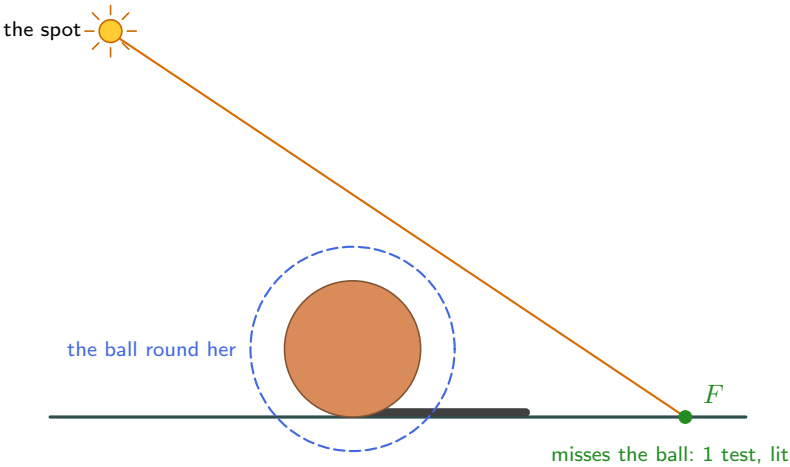
- Her farthest corner is 1.49 from her middle.

Skip 2: first ask a ball round her



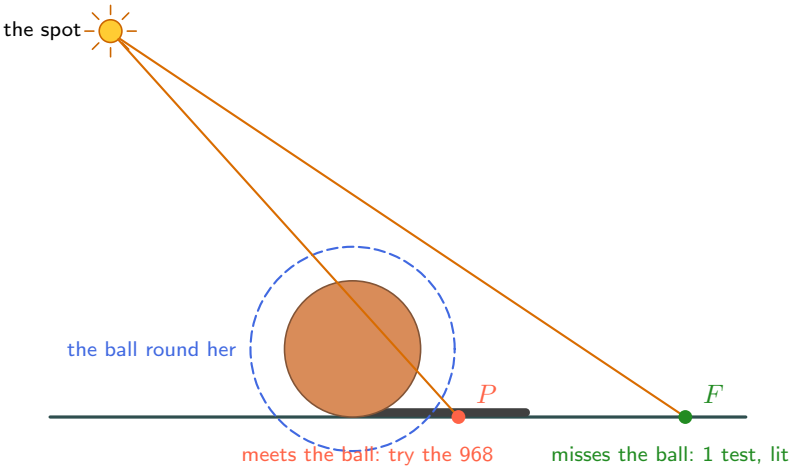
- Her farthest corner is 1.49 from her middle.
- A ball of radius 1.5 holds every one of her triangles.

Skip 2: first ask a ball round her



- Her farthest corner is 1.49 from her middle.
- A ball of radius 1.5 holds every one of her triangles.
- A ray that misses the ball misses all 968: one test, not 968.

Skip 2: first ask a ball round her



- Her farthest corner is 1.49 from her middle.
- A ball of radius 1.5 holds every one of her triangles.
- A ray that misses the ball misses all 968: one test, not 968.
- A ray that meets it still tries them all.

Worked: the ball test for two floor points

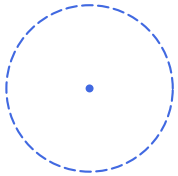
Ball: centre $(0, 0, 0)$, $R = 1.5$. Lecture 14, slide 15, with $A = 1$:

$B = 2\mathbf{l} \cdot P$, $C = P \cdot P - R^2$, $D = B^2 - 4C$: $D < 0$ misses

$P = (1, -1, -1)$, $\mathbf{l} = (-0.474,\ 0.813,\ 0.339)$ lecture 30's P

$F = (3, -1, 0)$, $\mathbf{l} = (-0.665,\ 0.725,\ 0.181)$

Try first: which ray meets the ball?



Worked: the ball test for two floor points

Ball: centre $(0,0,0)$, $R = 1.5$. Lecture 14, slide 15, with $A = 1$:

$B = 2\mathbf{l} \cdot P$, $C = P \cdot P - R^2$, $D = B^2 - 4C$: $D < 0$ misses

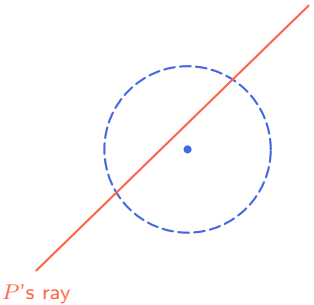
$P = (1, -1, -1)$, $\mathbf{l} = (-0.474,\ 0.813,\ 0.339)$ lecture 30's P

$F = (3, -1, 0)$, $\mathbf{l} = (-0.665,\ 0.725,\ 0.181)$

Try first: which ray meets the ball?

P: $B = -3.25$, $C = 0.75$, $D = 10.57 - 3 = 7.57 \geq 0$: **meets it**

and ahead of P ($t_+ = 3.0 > 0$): try the triangles; 571 is hit



Worked: the ball test for two floor points

Ball: centre $(0,0,0)$, $R = 1.5$. Lecture 14, slide 15, with $A = 1$:

$B = 2\mathbf{l}\cdot P$, $C = P\cdot P - R^2$, $D = B^2 - 4C$: $D < 0$ misses

$P = (1,-1,-1)$, $\mathbf{l} = (-0.474,\ 0.813,\ 0.339)$ lecture 30's P

$F = (3,-1,0)$, $\mathbf{l} = (-0.665,\ 0.725,\ 0.181)$

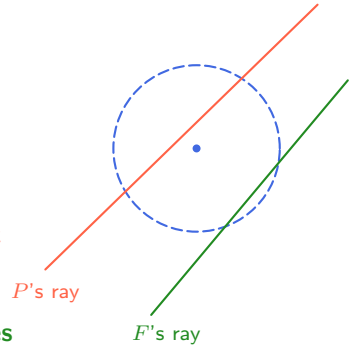
Try first: which ray meets the ball?

P: $B = -3.25$, $C = 0.75$, $D = 10.57 - 3 = 7.57 \geq 0$: **meets it**

and ahead of P ($t_+ = 3.0 > 0$): try the triangles; 571 is hit

F: $B = -5.44$, $C = 7.75$, $D = 29.56 - 31 = -1.44 < 0$: **misses**

lit after one test, with 968 triangle tests saved



The ball test, in WGSL: meets_ball

```
fn meets_ball(start: vec3f, dir: vec3f) -> bool {
    let centre = vec3f(0.0, 0.0, 0.0);  // the middle of her box
    let radius = 1.5;                   // holds all of her




}
```

- Her ball: one centre, one radius.

▶ **In the Ray Lab:**
Trace only what you must: two skips

The ball test, in WGSL: meets_ball

```
fn meets_ball(start: vec3f, dir: vec3f) -> bool {
    let centre = vec3f(0.0, 0.0, 0.0);  // the middle of her box
    let radius = 1.5;                   // holds all of her
    let B = 2.0 * dot(dir, start - centre);
    let C = dot(start - centre, start - centre) - radius * radius;
    let D = B * B - 4.0 * C;          // A = 1: dir has length 1
    if (D < 0.0) { return false; }  // misses

}
```

- Her ball: one centre, one radius.
- Lecture 14's B, C, D, with P for $\mathbf{e}$.

▶ **In the Ray Lab:**
Trace only what you must: two skips

The ball test, in WGSL: meets_ball

```
fn meets_ball(start: vec3f, dir: vec3f) -> bool {
    let centre = vec3f(0.0, 0.0, 0.0);  // the middle of her box
    let radius = 1.5;                   // holds all of her
    let B = 2.0 * dot(dir, start - centre);
    let C = dot(start - centre, start - centre) - radius * radius;
    let D = B * B - 4.0 * C;          // A = 1: dir has length 1
    if (D < 0.0) { return false; }  // misses
    return (-B + sqrt(D)) / 2.0 > 0.0;  // t+ is ahead
}
```

- Her ball: one centre, one radius.
- Lecture 14's B, C, D, with P for $\mathbf{e}$.
- Ahead only: $t_+ > 0$.

Lecture 14, slide 25: both hits behind the start mean no hit. A floor point whose ray leaves her behind should not try her triangles.

▶ **In the Ray Lab:**
Trace only what you must: two skips

The ball test, in WGSL: meets_ball

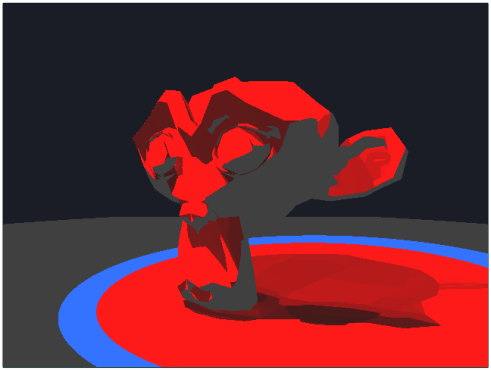
```
fn meets_ball(start: vec3f, dir: vec3f) -> bool {
    let centre = vec3f(0.0, 0.0, 0.0);  // the middle of her box
    let radius = 1.5;                   // holds all of her
    let B = 2.0 * dot(dir, start - centre);
    let C = dot(start - centre, start - centre) - radius * radius;
    let D = B * B - 4.0 * C;          // A = 1: dir has length 1
    if (D < 0.0) { return false; }  // misses
    return (-B + sqrt(D)) / 2.0 > 0.0;  // t+ is ahead
}

if (k > 0.0 && dot(n, l) > 0.0) {   // skip 1 (slide 17)
    if (meets_ball(P, l)) {             // skip 2
        for (var i = 0; i < 968; i++) { ... }  // slide 11's loop
    }
}
```

Lecture 14, slide 25: both hits behind the start mean no hit. A floor point whose ray leaves her behind should not try her triangles.

- Her ball: one centre, one radius.
- Lecture 14's B, C, D, with P for $\mathbf{e}$.
- Ahead only: $t_+ > 0$.
- Two `if`s round slide 11's loop. Same picture.

▶ **In the Ray Lab:**
Trace only what you must: two skips

The cost picture: grey, blue, red

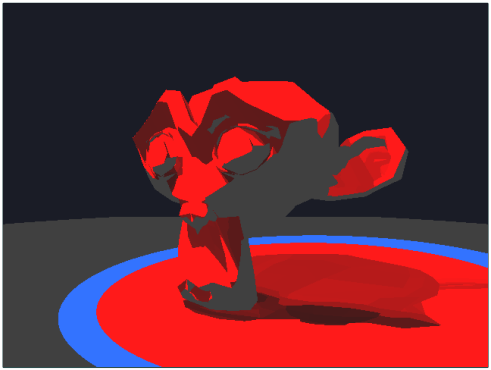


- no ray (skip 1)
- one test: missed the ball (skip 2)
- the loop: brighter = more triangles

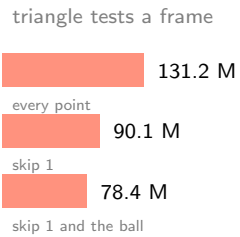
- Each pixel painted by the work its ray did.

▶ **In the Ray Lab:**
The cost picture: grey, blue, red

The cost picture: grey, blue, red



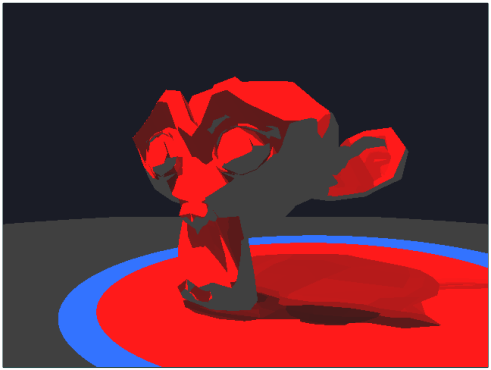
- no ray (skip 1)
- one test: missed the ball (skip 2)
- the loop: brighter = more triangles



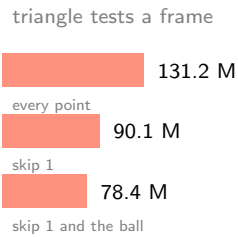
- Each pixel painted by the work its ray did.
- The ball saves only 13%.

▶ **In the Ray Lab:**
The cost picture: grey, blue, red

The cost picture: grey, blue, red



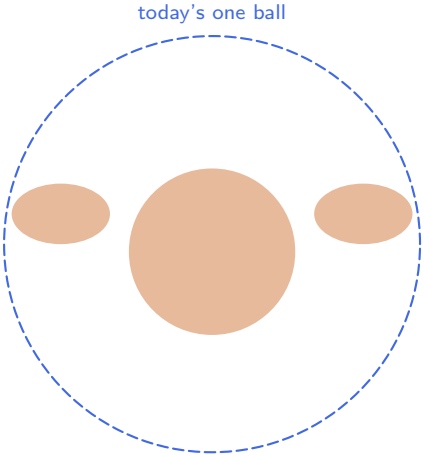
no ray (skip 1)
one test: missed the ball (skip 2)
the loop: brighter = more triangles



- Each pixel painted by the work its ray did.
- The ball saves only 13%.
- Why: seen from the spot, her lit floor lies right behind her.

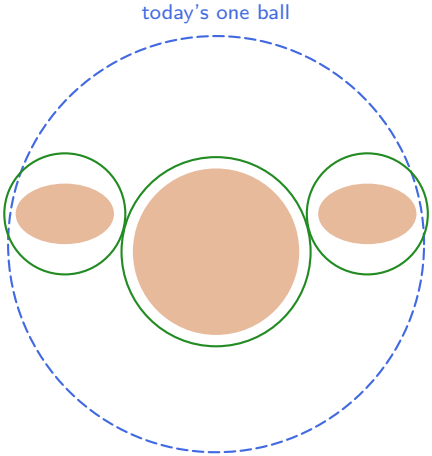
▶ **In the Ray Lab:**
The cost picture: grey, blue, red

Why a BVH exists: balls inside the ball



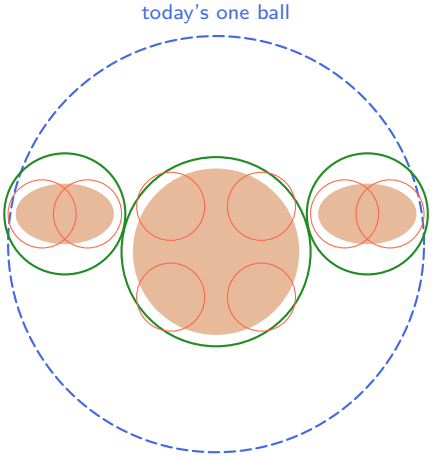
- A ray that misses a ball skips everything inside it.

Why a BVH exists: balls inside the ball



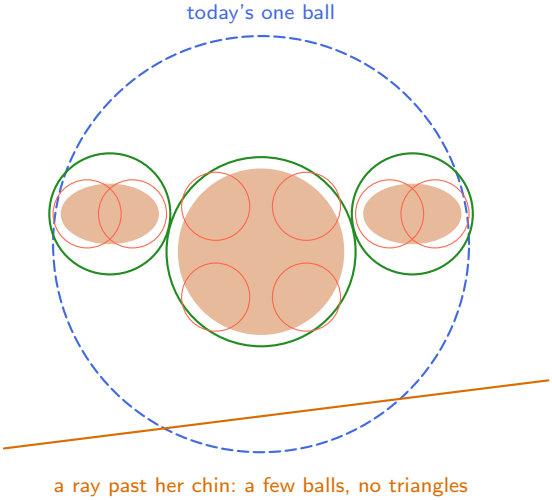
- A ray that misses a ball skips everything inside it.
- Inside it, smaller balls, and smaller again.

Why a BVH exists: balls inside the ball



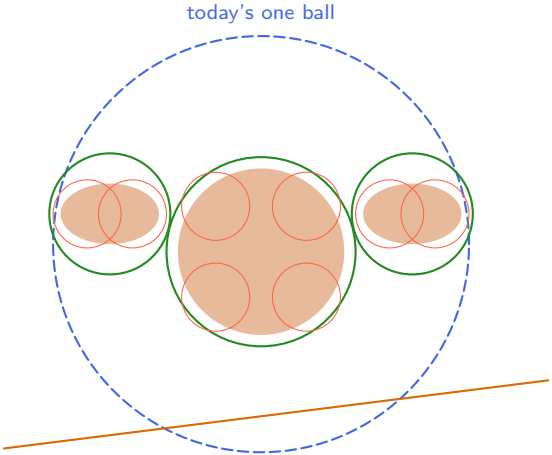
- A ray that misses a ball skips everything inside it.
- Inside it, smaller balls, and smaller again.

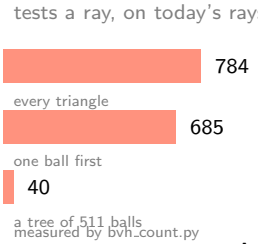
Why a BVH exists: balls inside the ball



- A ray that misses a ball skips everything inside it.
- Inside it, smaller balls, and smaller again.

Why a BVH exists: balls inside the ball





- A ray that misses a ball skips everything inside it.
- Inside it, smaller balls, and smaller again.
- 40 tests a ray, not 685: a **BVH**. RTX cards walk one in hardware.

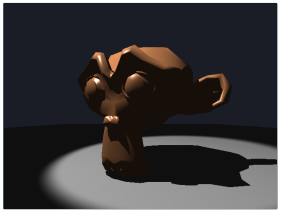
You Try 3: a ball too small

In the Ray Lab, choose **You Try 3**. Guess first, then change the code.

```
fn meets_ball(start: vec3f, dir: vec3f) -> bool {
    let centre = vec3f(0.0, 0.0, 0.0);
    let radius = 1.5;
```

change this line

now



- **Set the radius to 0.8. Her ears and chin now stick out of the ball.**
- Guess first: what happens to her shadow on the floor?

You Try 3: a ball too small

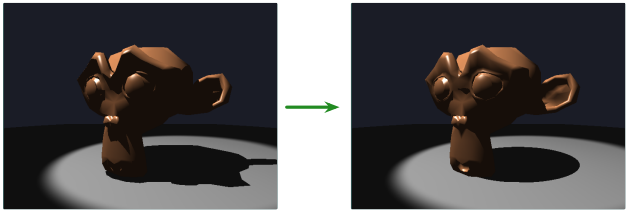
In the Ray Lab, choose **You Try 3**. Guess first, then change the code.

```
fn meets_ball(start: vec3f, dir: vec3f) -> bool {
    let centre = vec3f(0.0, 0.0, 0.0);
    let radius = 1.5;
```

change this line

answer: `let radius = 0.8;`

now



- **Set the radius to 0.8. Her ears and chin now stick out of the ball.**
- Guess first: what happens to her shadow on the floor?
- It turns into the small ball's round shadow. The ball must hold all of her.

You Try 4: stop at the first hit

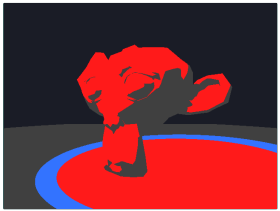
In the Ray Lab, choose **You Try 4**. Guess first, then change the code.

```
if (hits_triangle(P, l, eps, far, ...)) {
    lit = 0.0;
    // your line here
}
```

add a line here

- **The cost picture, with no break. Put it back.**
- Guess first: which red parts get darker?

now: the cost picture, no break



You Try 4: stop at the first hit

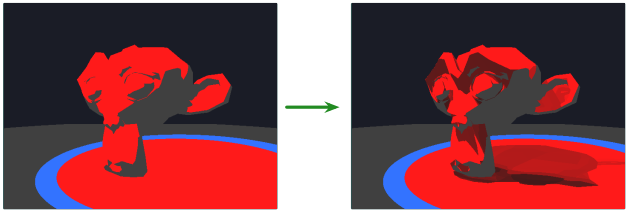
In the Ray Lab, choose **You Try 4**. Guess first, then change the code.

```
if (hits_triangle(P, l, eps, far, ...)) {
    lit = 0.0;
    // your line here
}
```

add a line here

answer: `break;`

now: the cost picture, no break



- **The cost picture, with no break. Put it back.**
- Guess first: which red parts get darker?
- The shadowed ones: their rays stop at a hit. Lit rays still try all 968.

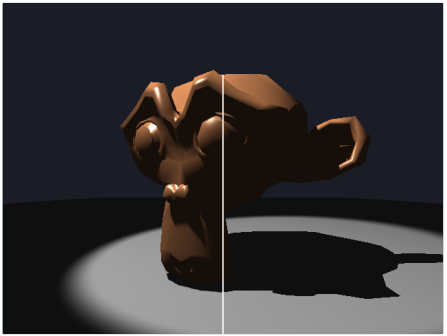
Round 3

The map against the ray

One scene, two answers to one question

Side by side: the map on the left, the ray on the right

lecture 30's map today's ray

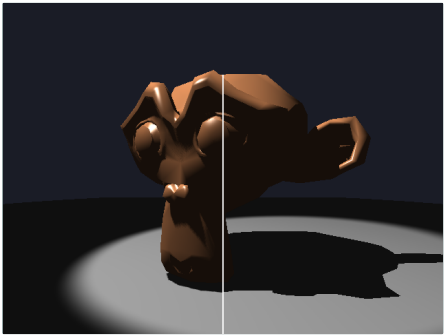


- Both answers in one shader: lecture 30's PCF map, and today's ray.

▶ **In the Ray Lab:**
Map on the left, ray on the right

Side by side: the map on the left, the ray on the right

lecture 30's map today's ray



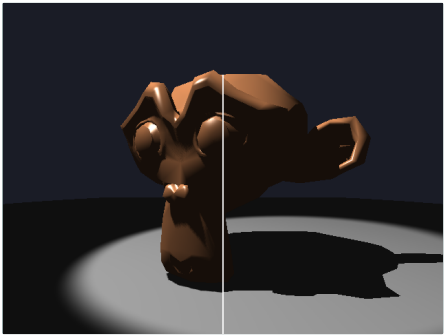
```
var lit = ray_lit;                    // the ray's answer
if (pos.x < u.resolution.x / 2.0) {  // the left half:
    lit = map_lit;                   // the map's answer
}
pos: the pixel, as @builtin(position) (lecture 26)
```

- Both answers in one shader: lecture 30's PCF map, and today's ray.
- Each pixel keeps one of the two.

▶ **In the Ray Lab:**
Map on the left, ray on the right

Side by side: the map on the left, the ray on the right

lecture 30's map today's ray



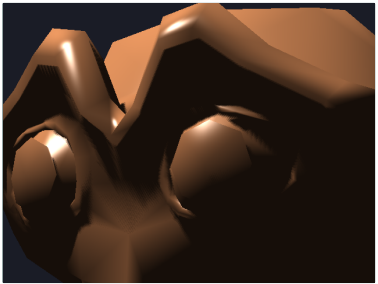
```
var lit = ray_lit;                    // the ray's answer
if (pos.x < u.resolution.x / 2.0) {  // the left half:
    lit = map_lit;                   // the map's answer
}
pos: the pixel, as @builtin(position) (lecture 26)
```

- Both answers in one shader: lecture 30's PCF map, and today's ray.
- Each pixel keeps one of the two.
- From afar they look alike. Close up, they do not.

▶ **In the Ray Lab:**
Map on the left, ray on the right

Close up: her brow

the map (1024, PCF)

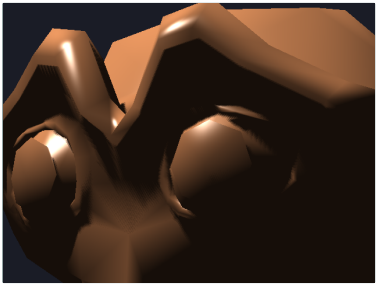


stripes between her eyes: acne
the bias did not cure; soft, blurred edges

- Lecture 30, slide 33: acne, and PCF's blur.

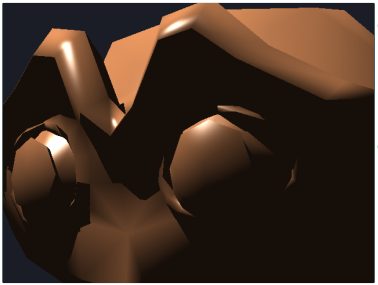
Close up: her brow

the map (1024, PCF)



stripes between her eyes: acne the bias did not cure; soft, blurred edges

the ray

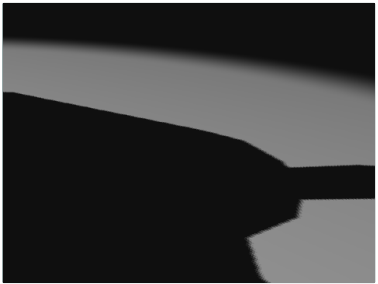


sharp edges where her brow really blocks the spot

- Lecture 30, slide 33: acne, and PCF's blur.
- The ray asks the triangles themselves: no texels, no bias.

Close up: her shadow's edge on the floor

the map (1024, PCF)

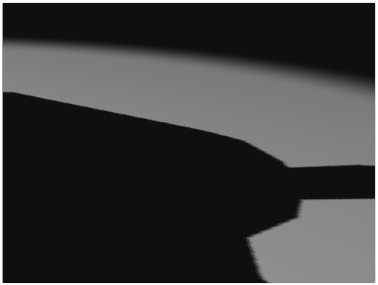


texel steps, blurred by PCF

- A texel covers many pixels (lecture 30, slide 27).

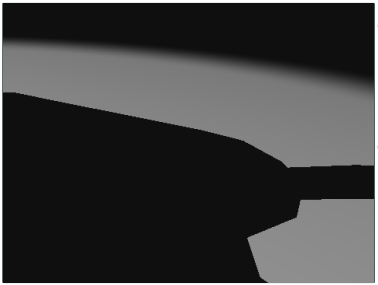
Close up: her shadow's edge on the floor

the map (1024, PCF)



texel steps, blurred by PCF

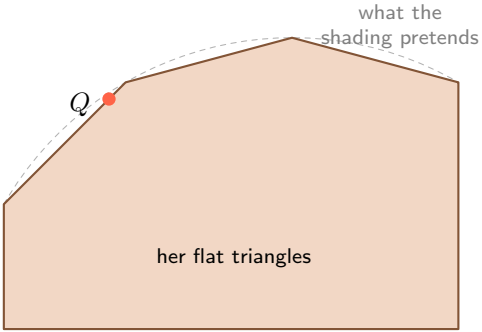
the ray



the true outline: her ear's flat triangles

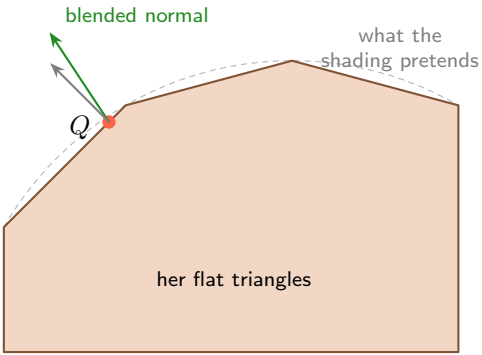
- A texel covers many pixels (lecture 30, slide 27).
- A ray per pixel: the edge is exact at any size.

The ray's own artefact: triangle steps



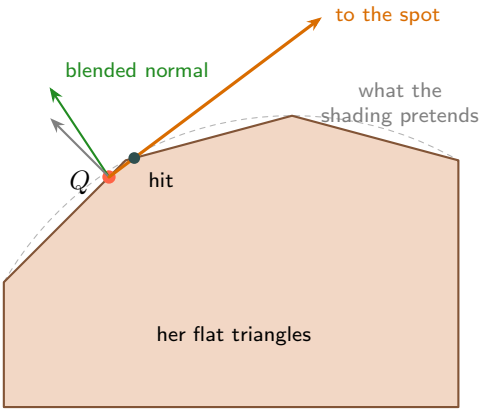
- The shading blends the corners' normals: she looks round (lecture 29).

The ray's own artefact: triangle steps



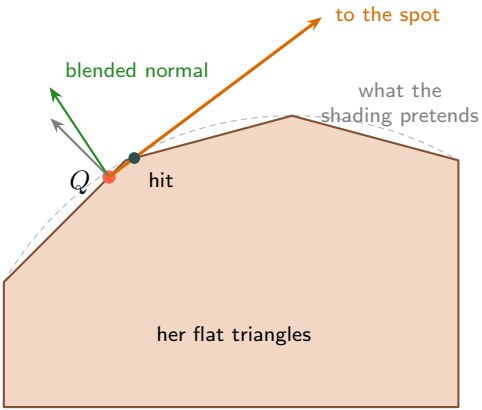
- The shading blends the corners' normals: she looks round (lecture 29).
- Near her dark side, Q's blended normal still faces the spot.

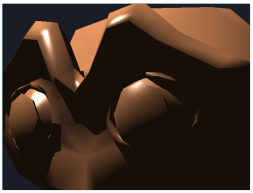
The ray's own artefact: triangle steps



- The shading blends the corners' normals: she looks round (lecture 29).
- Near her dark side, Q's blended normal still faces the spot.
- Its ray meets the next flat triangle: shadow, in a triangle's shape.

The ray's own artefact: triangle steps



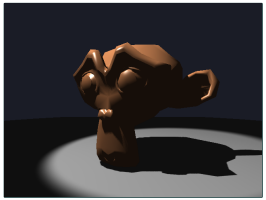


close up, slide 27

- The shading blends the corners' normals: she looks round (lecture 29).
- Near her dark side, Q's blended normal still faces the spot.
- Its ray meets the next flat triangle: shadow, in a triangle's shape.
- The map had acne; the ray has steps. More triangles make them smaller.

What each one costs

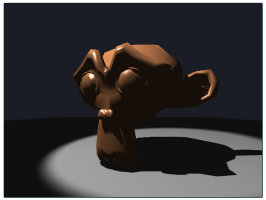
The shadow map



work	one extra drawing: 3,048 corners
memory	1024×1024 depths: 4 MB
artefacts	acne, Peter Panning, blocky edges
cost	grows with the map's size

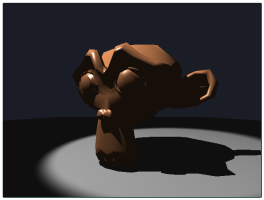
What each one costs

The shadow map



work	one extra drawing: 3,048 corners
memory	1024×1024 depths: 4 MB
artefacts	acne, Peter Panning, blocky edges
cost	grows with the map's size

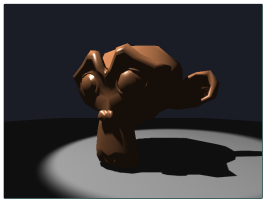
The shadow ray



work	about 78 million triangle tests
memory	968 triangles: 46 KB
artefacts	triangle steps at her dark side
cost	grows with pixels $\times$ triangles

What each one costs

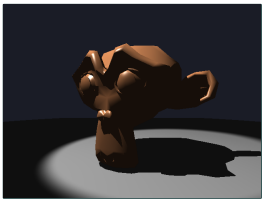
The shadow map



Cheap, and the same speed for any mesh.

work	one extra drawing: 3,048 corners
memory	1024 × 1024 depths: 4 MB
artefacts	acne, Peter Panning, blocky edges
cost	grows with the map's size

The shadow ray



A BVH (slide 22) cuts the ray's cost about 17 times.

work	about 78 million triangle tests
memory	968 triangles: 46 KB
artefacts	triangle steps at her dark side
cost	grows with pixels × triangles

You Try 5: the map for the floor, the ray for her

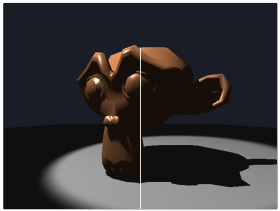
In the Ray Lab, choose **You Try 5**. Guess first, then change the code.

```
var lit = ray_lit;
if (pos.x < u.resolution.x / 2.0) {
    lit = map_lit;
}
```

change this line

- **Let the floor use the map and Suzanne use the ray.**
- Guess first: where does the picture change?

now: left half map, right half ray



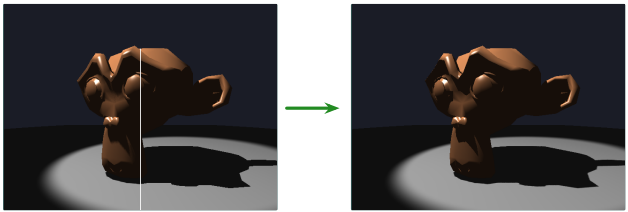
You Try 5: the map for the floor, the ray for her

In the Ray Lab, choose **You Try 5**. Guess first, then change the code.

```
var lit = ray_lit;
if (pos.x < u.resolution.x / 2.0) {
    lit = map_lit;
}
answer:  if (P.y < -0.99) {  // the floor
```

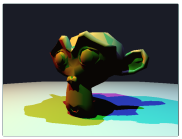
change this line

now: left half map, right half ray



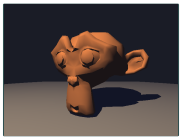
- **Let the floor use the map and Suzanne use the ray.**
- Guess first: where does the picture change?
- Only on her: sharp self-shadows. The floor keeps the map's shadow.
- Choosing rays by what a pixel shows: lecture 32.

Now make your own picture



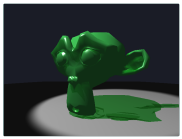
Coloured shadows

`three lamps, three rays`



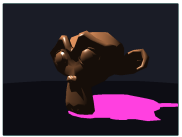
A sundial

`the sun moves on a circle`



A jelly Suzanne

`through *= green`



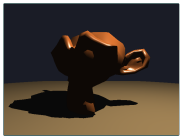
A shadow made of light

`k * (1.0 - lit)`



A searchlight

`aim moves with u.time`



The mouse is the lamp

`bulb = (5 mx, 3, 4 mz)`

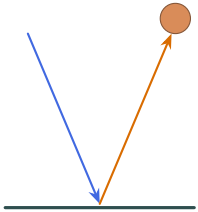
All six are in the Ray Lab, under **Ideas**. Change one. Mix two. Show your neighbour.

Summary

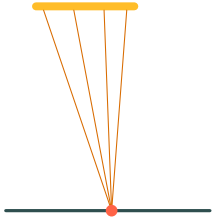
1	The shadow ray in fs_main	lecture 17's ray, from P toward the spot her triangles in a storage buffer; any hit in $[\epsilon, \text{far}]$: shadow
2	Ray meets triangle	the plane (lecture 14), then three edge signs $t = (a - P) \cdot \mathbf{n} / (\mathbf{l} \cdot \mathbf{n})$; inside when all three agree
3	Trace only what you must	skip unlit points; ask a ball first; stop at a hit 131 M $\rightarrow$ 78 M tests a frame; a tree of balls: 17 times fewer
4	The map against the ray	the map: cheap, acne and blur the ray: exact edges, triangle steps, costly
5	The hybrid	rasterize what you can, trace only what you must the floor reads the map; her face sends rays

Next lecture: the rays return, part 2

a mirror floor



soft shadows



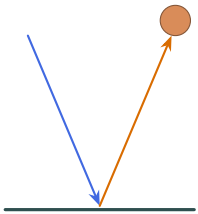
rays per pixel



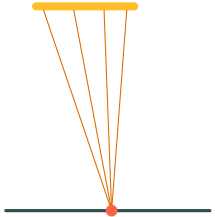
- Lecture 32: a mirror ray, sent only by floor pixels.

Next lecture: the rays return, part 2

a mirror floor



soft shadows



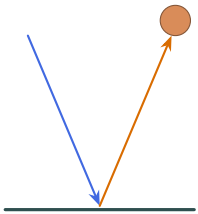
rays per pixel



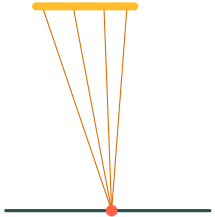
- Lecture 32: a mirror ray, sent only by floor pixels.
- Soft shadows: a few random rays to a lamp with a size, averaged over frames (A7).

Next lecture: the rays return, part 2

a mirror floor



soft shadows



rays per pixel



- Lecture 32: a mirror ray, sent only by floor pixels.
- Soft shadows: a few random rays to a lamp with a size, averaged over frames (A7).
- The frame coloured by its rays per pixel: 0, 1 or 2.

Exit Ticket

Today's question *(2 minutes)*

Triangle $a = (0, 0, 0)$, $b = (4, 0, 0)$, $c = (0, 4, 0)$. A shadow ray starts at $P = (1, 2, 5)$ with $\mathbf{l} = (0, 0, -1)$.

(a) Find t and x. Is x inside the triangle?

(b) The spot is only 4 away (far $= 4$). Is P lit or in shadow?

CS 453 — Exit Ticket #31

Name: _____

Roll #: _____ Date: _____

Answer:

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