Lecture 27: Light on a Fake Sphere

Normals, Diffuse Light and a Highlight

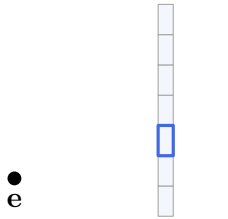
CS 453 — Computer Graphics

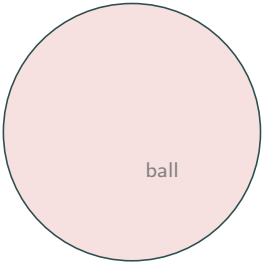
Fakhir Shaheen

Forman Christian University

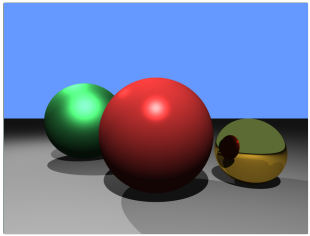
Lecture 17: how the ray tracer coloured one pixel

side view





lecture 17's picture

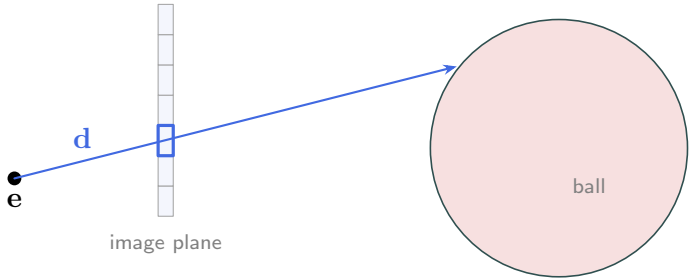


pixel

one at a time

Lecture 17: how the ray tracer coloured one pixel

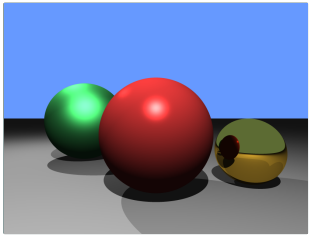
side view





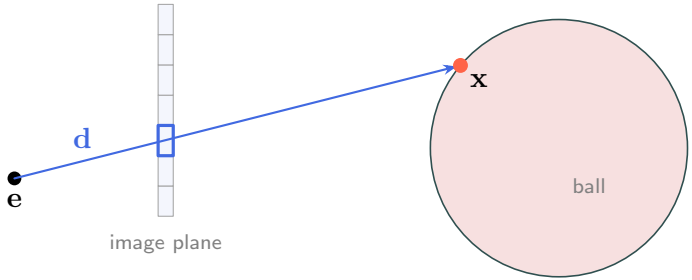
one at a time from e through it

lecture 17's picture



Lecture 17: how the ray tracer coloured one pixel

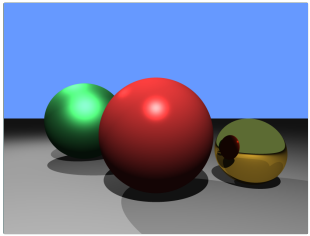
side view





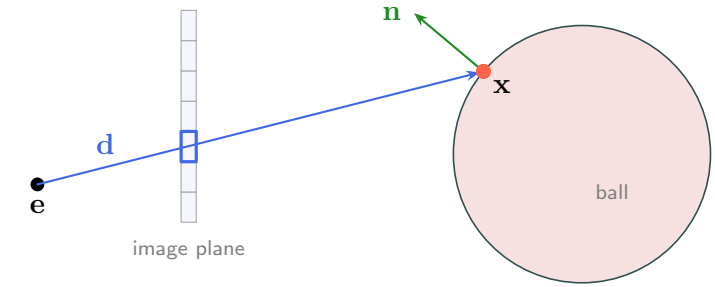
one at a time | from $\mathbf{e}$ through it | solve for t: the point $\mathbf{x}$

lecture 17's picture



Lecture 17: how the ray tracer coloured one pixel

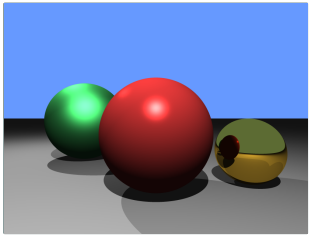
side view



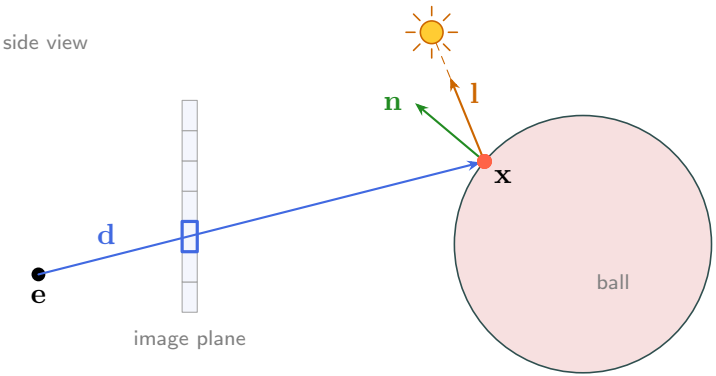


one at a time | from $\mathbf{e}$ through it | solve for t: the point $\mathbf{x}$ | out of the ball at $\mathbf{x}$

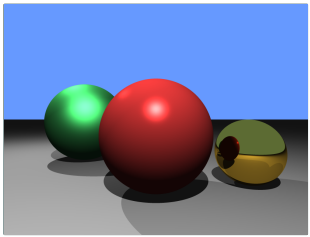
lecture 17's picture

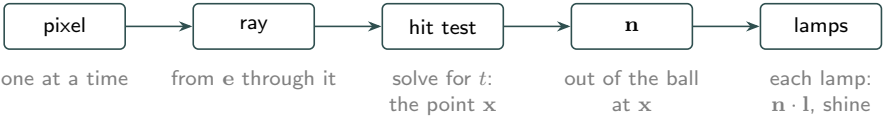


Lecture 17: how the ray tracer coloured one pixel

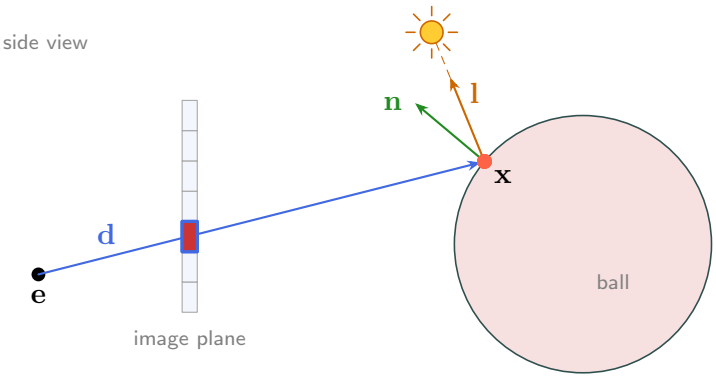


lecture 17's picture

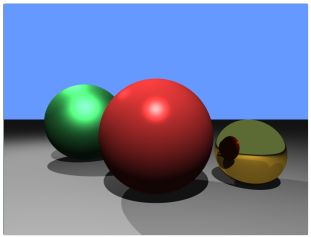




Lecture 17: how the ray tracer coloured one pixel



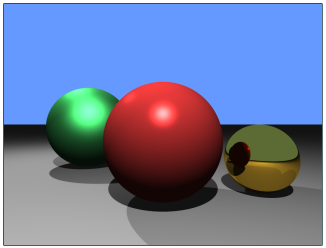
lecture 17's picture



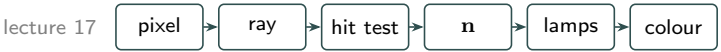
pixel	ray	hit test	$\mathbf{n}$	lamps	colour
one at a time	from $\mathbf{e}$ through it	solve for t: the point $\mathbf{x}$	out of the ball at $\mathbf{x}$	each lamp: $\mathbf{n} \cdot \mathbf{l}$, shine	into the pixel

Lecture 17's ball, drawn by one fragment shader

lecture 17: Python traces rays

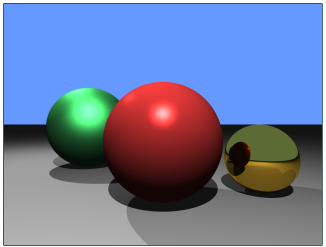


- Lecture 17: a ray for every pixel, then the lamps.

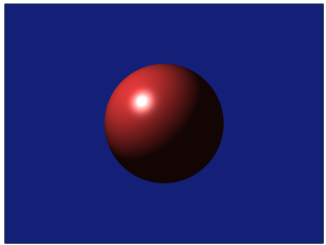


Lecture 17's ball, drawn by one fragment shader

lecture 17: Python traces rays



today: one fragment shader, no ray



- Lecture 17: a ray for every pixel, then the lamps.
- Today: no ray. Each pixel finds its own point and $\mathbf{n}$.

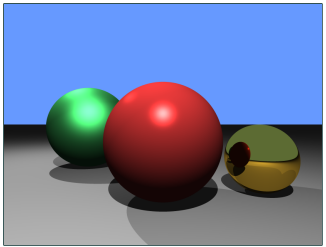
lecture 17: pixel → ray → hit test → $\mathbf{n}$ → lamps → colour

today: pixel → the pixel's point on the ball → $\mathbf{n}$ → lamps → colour

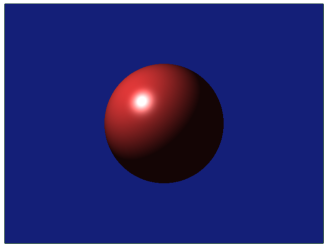
how: soon, slide 8

Lecture 17's ball, drawn by one fragment shader

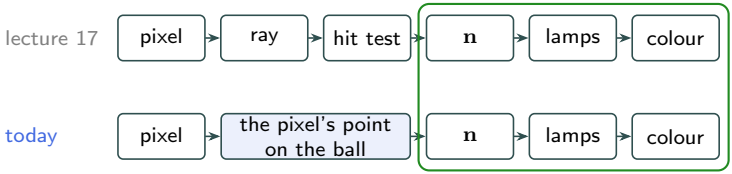
lecture 17: Python traces rays



today: one fragment shader, no ray



- Lecture 17: a ray for every pixel, then the lamps.
- Today: no ray. Each pixel finds its own point and $\mathbf{n}$.
- From $\mathbf{n}$ on, the light is lecture 16 and 17's.



how: soon, slide 8

the same maths: lectures 16 and 17

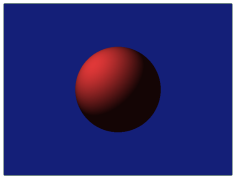
Today: three rounds of slides, then you try

1 **A circle becomes a ball**



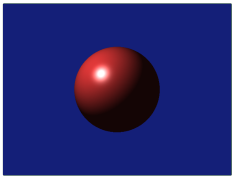
then You Try 1, 2

2 **Light from a lamp**



then You Try 3, 4

3 **A highlight, and the mouse as lamp**



then You Try 5, 6

Open the Shader Lab now. It is on the course calendar, in lecture 27's row.

A laptop or a phone will do. Work in pairs: one screen for two people is enough.

A slide with a blue **Run it in the Shader Lab** box has its code in the lab too.

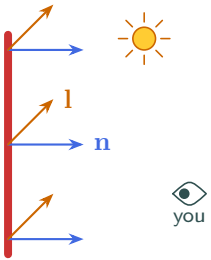
Round 1

A circle becomes a ball

Every pixel asks: which way does my point face?

A flat disc has one normal; a ball has many

a flat disc, from the side





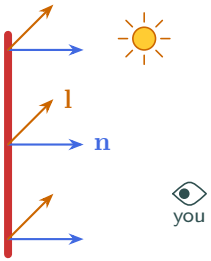
from the front

- Lecture 16: brightness comes from $\mathbf{n} \cdot \mathbf{l}$.
- A flat disc has one $\mathbf{n}$, so one colour.

▶ **Run it in the Shader Lab:**
A flat disc

A flat disc has one normal; a ball has many

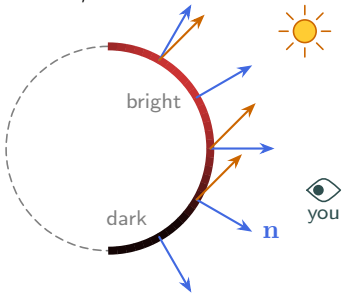
a flat disc, from the side

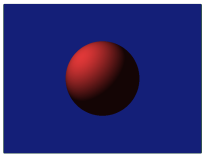




from the front

a ball, from the side





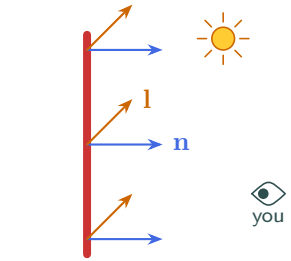
from the front

- Lecture 16: brightness comes from $\mathbf{n} \cdot \mathbf{l}$.
- A flat disc has one $\mathbf{n}$, so one colour.
- A ball's $\mathbf{n}$ turns from point to point.

▶ **Run it in the Shader Lab:**
A flat disc

A flat disc has one normal; a ball has many

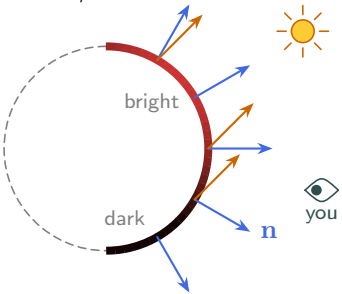
a flat disc, from the side

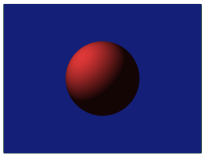




from the front

a ball, from the side





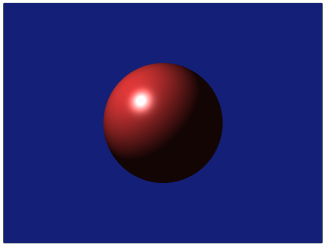
from the front

- Lecture 16: brightness comes from $\mathbf{n} \cdot \mathbf{l}$.
- A flat disc has one $\mathbf{n}$, so one colour.
- A ball's $\mathbf{n}$ turns from point to point.
- Round 1: find $\mathbf{n}$ at every pixel.

▶ **Run it in the Shader Lab:**
A flat disc

The ball is a fake: a flat disc with a ball's normals

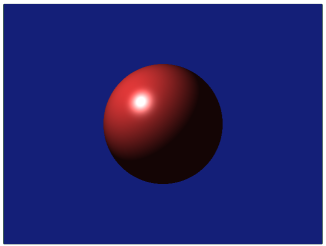
what you see



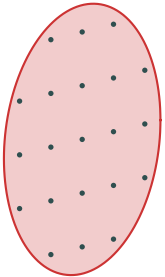
- Looks like a ball.

The ball is a fake: a flat disc with a ball's normals

what you see



the same pixels, turned



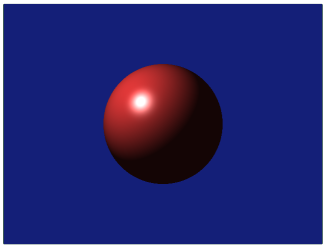


- Looks like a ball.
- Turned: a flat disc.

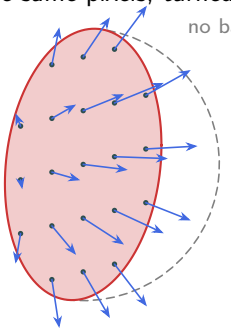
flat: $z = 0$ at every pixel

The ball is a fake: a flat disc with a ball's normals

what you see



the same pixels, turned

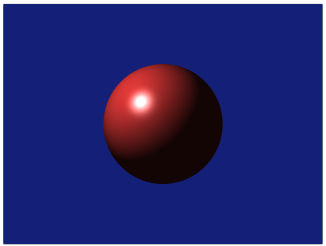


flat: $z = 0$ at every pixel

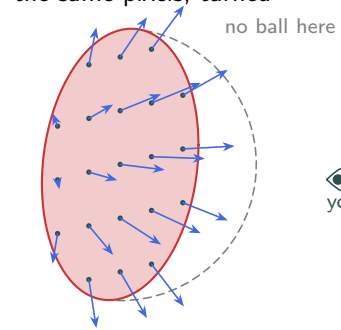
- Looks like a ball.
- Turned: a flat disc.
- Only $\mathbf{n}$ is a ball's.

The ball is a fake: a flat disc with a ball's normals

what you see



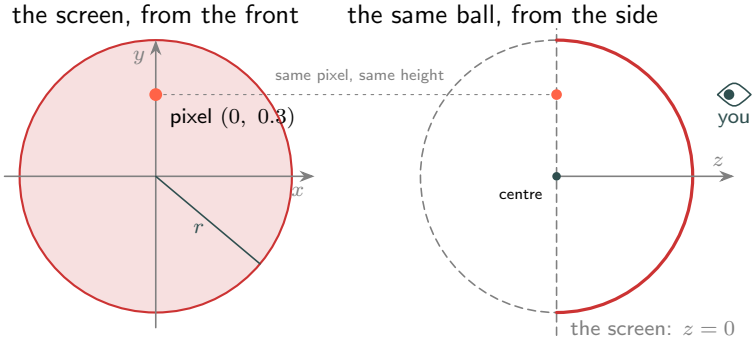
the same pixels, turned



flat: $z = 0$ at every pixel

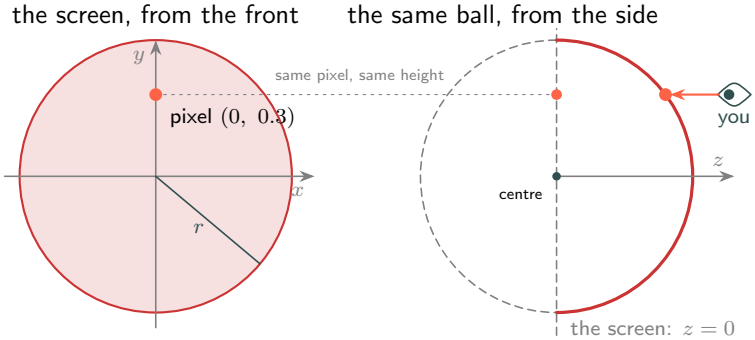
- Looks like a ball.
- Turned: a flat disc.
- Only $\mathbf{n}$ is a ball's.
- The light reads $\mathbf{n}$: you see a ball.

Each pixel meets one point of the ball's front



- z points out of the screen, toward you.

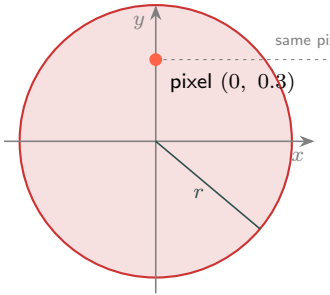
Each pixel meets one point of the ball's front



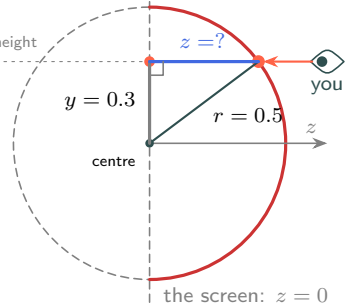
- z points out of the screen, toward you.
- Each pixel looks straight in and meets the ball's front.

Each pixel meets one point of the ball's front

the screen, from the front



the same ball, from the side



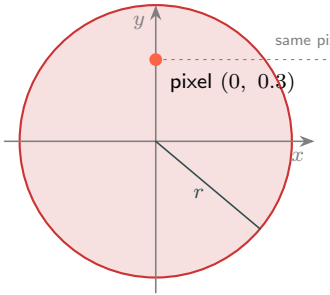
- z points out of the screen, toward you.
- Each pixel looks straight in and meets the ball's front.
- Pythagoras gives how far out: z.

The ball's equation (lecture 14), centre $(0, 0, 0)$: $x^2 + y^2 + z^2 = r^2$

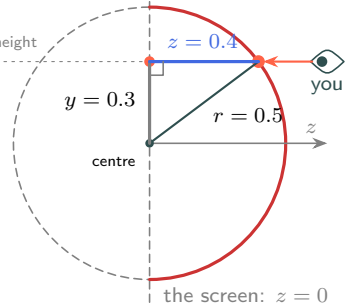
Solve for z: $z = \sqrt{r^2 - x^2 - y^2}$ **Try first:** z at the pixel $(0,\ 0.3)$

Each pixel meets one point of the ball's front

the screen, from the front



the same ball, from the side



- z points out of the screen, toward you.
- Each pixel looks straight in and meets the ball's front.
- Pythagoras gives how far out: z.

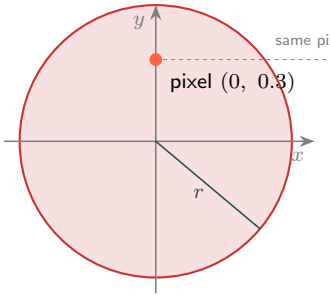
The ball's equation (lecture 14), centre $(0, 0, 0)$: $x^2 + y^2 + z^2 = r^2$

Solve for z: $z = \sqrt{r^2 - x^2 - y^2}$ **Try first:** z at the pixel $(0,\ 0.3)$

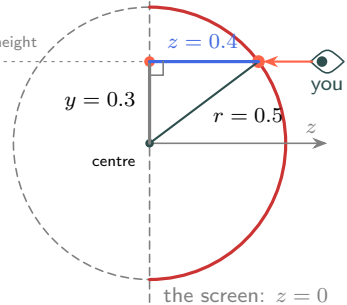
$z = \sqrt{0.25 - 0 - 0.09} = \sqrt{0.16} = 0.4$

Each pixel meets one point of the ball's front

the screen, from the front



the same ball, from the side



- z points out of the screen, toward you.
- Each pixel looks straight in and meets the ball's front.
- Pythagoras gives how far out: z.

The ball's equation (lecture 14), centre $(0, 0, 0)$: $x^2 + y^2 + z^2 = r^2$

Solve for z: $z = \sqrt{r^2 - x^2 - y^2}$ **Try first:** z at the pixel $(0,\ 0.3)$

$z = \sqrt{0.25 - 0 - 0.09} = \sqrt{0.16} = 0.4$

```
let z = sqrt(r * r - x * x - y * y);
```

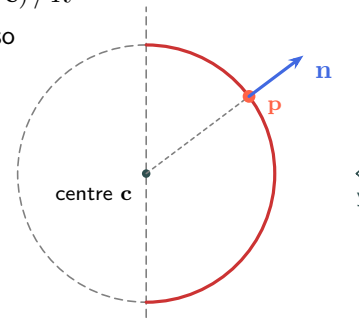
sqrt: built into WGSL

The normal: from the centre out through the point

Lecture 14, slide 26: $\mathbf{n} = (\mathbf{p} - \mathbf{c}) \,/\, R$

Our ball's centre $\mathbf{c}$ is $(0, 0, 0)$, so

$$\mathbf{n} = \frac{(x,\ y,\ z)}{r}$$



seen from the side

- $\mathbf{n}$ starts at the centre and points out through the point.
- Dividing by r makes its length 1.

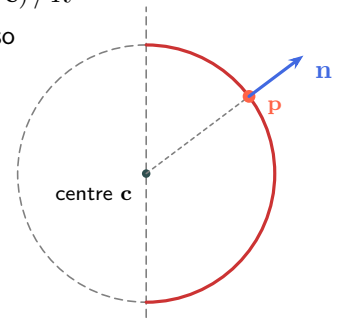
The normal: from the centre out through the point

Lecture 14, slide 26: $\mathbf{n} = (\mathbf{p} - \mathbf{c}) \,/\, R$

Our ball's centre $\mathbf{c}$ is $(0, 0, 0)$, so

$$\mathbf{n} = \frac{(x,\ y,\ z)}{r}$$

Try first: $\mathbf{n}$ at our point $(0,\ 0.3,\ 0.4)$, with $r = 0.5$



seen from the side

- $\mathbf{n}$ starts at the centre and points out through the point.
- Dividing by r makes its length 1.

The normal: from the centre out through the point

Lecture 14, slide 26: $\mathbf{n} = (\mathbf{p} - \mathbf{c}) \,/\, R$

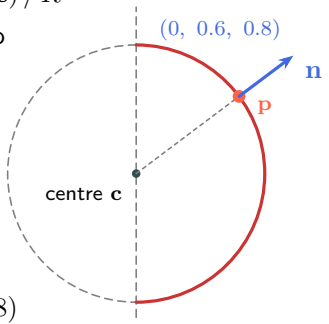
Our ball's centre $\mathbf{c}$ is $(0, 0, 0)$, so

$$\mathbf{n} = \frac{(x,\ y,\ z)}{r}$$

Try first: $\mathbf{n}$ at our point $(0,\ 0.3,\ 0.4)$, with $r = 0.5$

$(0,\ 0.3,\ 0.4) \,/\, 0.5 = (0,\ 0.6,\ 0.8)$

its length: $\sqrt{0^2 + 0.6^2 + 0.8^2} = 1$



seen from the side

- $\mathbf{n}$ starts at the centre and points out through the point.
- Dividing by r makes its length 1.

The normal: from the centre out through the point

Lecture 14, slide 26: $\mathbf{n} = (\mathbf{p} - \mathbf{c}) \,/\, R$

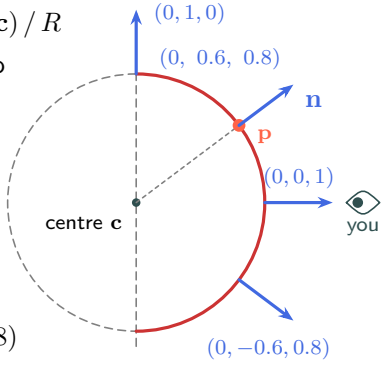
Our ball's centre $\mathbf{c}$ is $(0, 0, 0)$, so

$$\mathbf{n} = \frac{(x,\ y,\ z)}{r}$$

Try first: $\mathbf{n}$ at our point $(0,\ 0.3,\ 0.4)$, with $r = 0.5$

$(0,\ 0.3,\ 0.4) \,/\, 0.5 = (0,\ 0.6,\ 0.8)$

its length: $\sqrt{0^2 + 0.6^2 + 0.8^2} = 1$



seen from the side

- $\mathbf{n}$ starts at the centre and points out through the point.
- Dividing by r makes its length 1.
- At the front $\mathbf{n}$ points at you. At the top it points up.

The normal: from the centre out through the point

Lecture 14, slide 26: $\mathbf{n} = (\mathbf{p} - \mathbf{c}) \,/\, R$

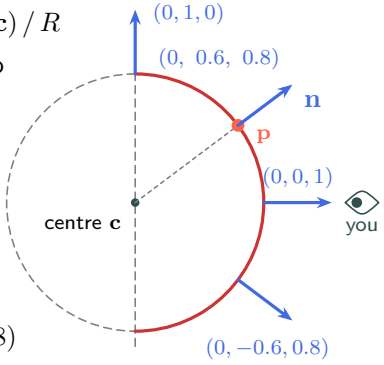
Our ball's centre $\mathbf{c}$ is $(0, 0, 0)$, so

$$\mathbf{n} = \frac{(x,\ y,\ z)}{r}$$

Try first: $\mathbf{n}$ at our point $(0,\ 0.3,\ 0.4)$, with $r = 0.5$

$(0,\ 0.3,\ 0.4) \,/\, 0.5 = (0,\ 0.6,\ 0.8)$

its length: $\sqrt{0^2 + 0.6^2 + 0.8^2} = 1$



seen from the side

- $\mathbf{n}$ starts at the centre and points out through the point.
- Dividing by r makes its length 1.
- At the front $\mathbf{n}$ points at you. At the top it points up.

```
let n = vec3f(x, y, z) / r;
```

vec3f: three numbers in one value (lecture 26)

Colour by the normal: see $\mathbf{n}$

```
let n = vec3f(x, y, z) / r;
let c = 0.5 * n + 0.5;
```

Each part of $\mathbf{n}$ runs from -1 to 1; a colour needs 0 to 1.
So halve it, then add 0.5. Red: $n.x$, green: $n.y$, blue: $n.z$.



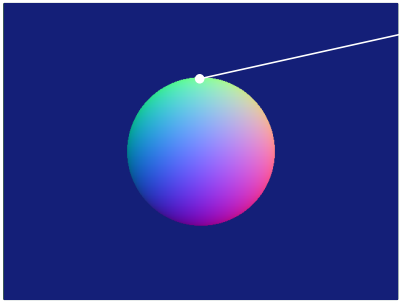
- A colour shows $\mathbf{n}$'s three parts at once.

▶ **Run it in the Shader Lab:**
The normal as a colour

Colour by the normal: see $\mathbf{n}$

```
let n = vec3f(x, y, z) / r;
let c = 0.5 * n + 0.5;
```

Each part of $\mathbf{n}$ runs from -1 to 1; a colour needs 0 to 1.
So halve it, then add 0.5. Red: $n.x$, green: $n.y$, blue: $n.z$.



top: $\mathbf{n} = (0, 1, 0)$
colour $(0.5, 1, 0.5)$: green

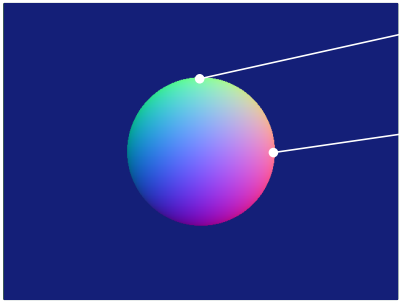
- A colour shows $\mathbf{n}$'s three parts at once.
- Lecture 14 (slide 33) coloured its sphere this way.

▶ **Run it in the Shader Lab:**
The normal as a colour

Colour by the normal: see $\mathbf{n}$

```
let n = vec3f(x, y, z) / r;
let c = 0.5 * n + 0.5;
```

Each part of $\mathbf{n}$ runs from -1 to 1; a colour needs 0 to 1.
So halve it, then add 0.5. Red: $n.x$, green: $n.y$, blue: $n.z$.



top: $\mathbf{n} = (0, 1, 0)$
colour $(0.5, 1, 0.5)$: green

right edge: $\mathbf{n} = (1, 0, 0)$
colour $(1, 0.5, 0.5)$: pink

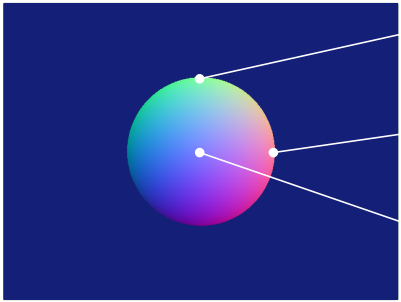
- A colour shows $\mathbf{n}$'s three parts at once.
- Lecture 14 (slide 33) coloured its sphere this way.

▶ **Run it in the Shader Lab:**
The normal as a colour

Colour by the normal: see $\mathbf{n}$

```
let n = vec3f(x, y, z) / r;
let c = 0.5 * n + 0.5;
```

Each part of $\mathbf{n}$ runs from -1 to 1; a colour needs 0 to 1.
So halve it, then add 0.5. Red: $n.x$, green: $n.y$, blue: $n.z$.



top: $\mathbf{n} = (0, 1, 0)$
colour $(0.5, 1, 0.5)$: green

right edge: $\mathbf{n} = (1, 0, 0)$
colour $(1, 0.5, 0.5)$: pink

middle: $\mathbf{n} = (0, 0, 1)$
colour $(0.5, 0.5, 1)$: blue

- A colour shows $\mathbf{n}$'s three parts at once.
- Lecture 14 (slide 33) coloured its sphere this way.
- If the colours look right, $\mathbf{n}$ is right.

▶ **Run it in the Shader Lab:**
The normal as a colour

You Try 1: grey by $n.z$

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

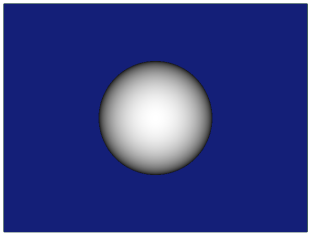
```
let n = vec3f(x, y, z) / r;
let c = 0.5 * n + 0.5;
return vec4f(c.r, c.g, c.b, 1.0);
```

change this line

now



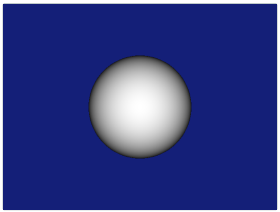
make this



- **Paint the ball grey by $n.z$: how much each point faces you.**
- Guess first: where will the ball be brightest?

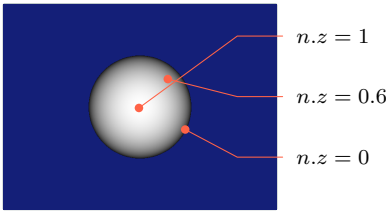
You Try 1, the answer: a brightness becomes a grey

1. How much a point faces you: $\mathbf{n} \cdot \mathbf{l}$ with $\mathbf{l} = (0, 0, 1)$



You Try 1, the answer: a brightness becomes a grey

1. How much a point faces you: $\mathbf{n} \cdot \mathbf{l}$ with $\mathbf{l} = (0, 0, 1)$
2. $\mathbf{n} \cdot \mathbf{l} = n.x \cdot 0 + n.y \cdot 0 + n.z \cdot 1 = n.z$: one number



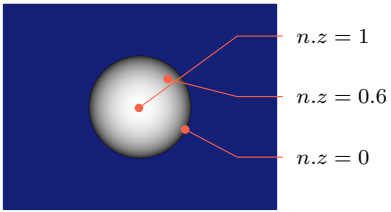
brightest in the middle:
there $\mathbf{n}$ points at you

You Try 1, the answer: a brightness becomes a grey

1. How much a point faces you: $\mathbf{n} \cdot \mathbf{l}$ with $\mathbf{l} = (0, 0, 1)$
2. $\mathbf{n} \cdot \mathbf{l} = n.x \cdot 0 + n.y \cdot 0 + n.z \cdot 1 = n.z$: one number
3. One number for three channels: a grey is (b, b, b)



b from 0 to 1: black to white



brightest in the middle:
there $\mathbf{n}$ points at you

You Try 1, the answer: a brightness becomes a grey

1. How much a point faces you: $\mathbf{n} \cdot \mathbf{l}$ with $\mathbf{l} = (0, 0, 1)$
2. $\mathbf{n} \cdot \mathbf{l} = n.x \cdot 0 + n.y \cdot 0 + n.z \cdot 1 = n.z$: one number
3. One number for three channels: a grey is (b, b, b)

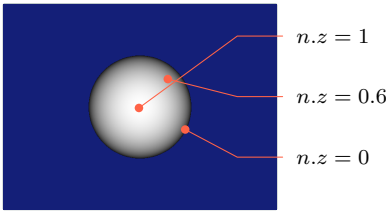


0 0.25 0.5 0.75 1

b from 0 to 1: black to white

4. So paint each pixel the grey of its $n.z$:

```
let c = vec3f(n.z, n.z, n.z);
```



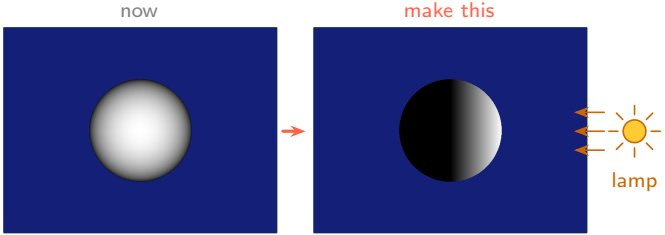
brightest in the middle:
there $\mathbf{n}$ points at you

You Try 2: light from the right

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

```
let n = vec3f(x, y, z) / r;
let c = vec3f(n.z, n.z, n.z);
return vec4f(c.r, c.g, c.b, 1.0);
```

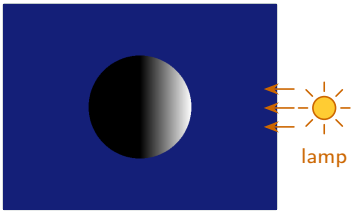
change this line



- **Make the ball look lit from the right.**
- Guess first: which part of $\mathbf{n}$ points right?

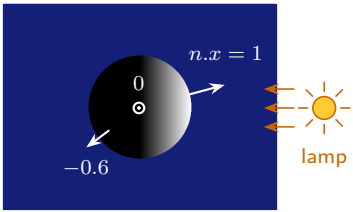
You Try 2, the answer: the lamp on the right

1. The lamp is to the right: $\mathbf{l} = (1, 0, 0)$



You Try 2, the answer: the lamp on the right

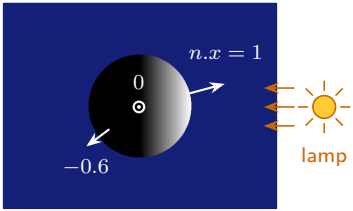
1. The lamp is to the right: $\mathbf{l} = (1, 0, 0)$
2. $\mathbf{n} \cdot \mathbf{l} = n.x \cdot 1 + n.y \cdot 0 + n.z \cdot 0 = n.x$: one number



You Try 2, the answer: the lamp on the right

1. The lamp is to the right: $\mathbf{l} = (1, 0, 0)$
2. $\mathbf{n} \cdot \mathbf{l} = n.x \cdot 1 + n.y \cdot 0 + n.z \cdot 0 = n.x$: one number
3. A grey of that brightness, as in You Try 1:

```
let c = vec3f(n.x, n.x, n.x);
```

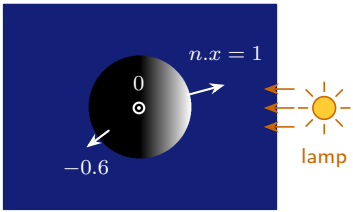


You Try 2, the answer: the lamp on the right

1. The lamp is to the right: $\mathbf{l} = (1, 0, 0)$
2. $\mathbf{n} \cdot \mathbf{l} = n.x \cdot 1 + n.y \cdot 0 + n.z \cdot 0 = n.x$: one number
3. A grey of that brightness, as in You Try 1:

```
let c = vec3f(n.x, n.x, n.x);
```

4. Left half: $n.x$ is negative, so $(-0.6, -0.6, -0.6)$

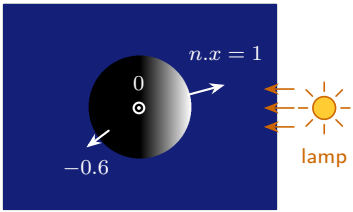


You Try 2, the answer: the lamp on the right

1. The lamp is to the right: $\mathbf{l} = (1, 0, 0)$
2. $\mathbf{n} \cdot \mathbf{l} = n.x \cdot 1 + n.y \cdot 0 + n.z \cdot 0 = n.x$: one number
3. A grey of that brightness, as in You Try 1:

```
let c = vec3f(n.x, n.x, n.x);
```

4. Left half: $n.x$ is negative, so $(-0.6, -0.6, -0.6)$
5. The GPU stores below 0 as 0: pure black



After fs_main: the GPU clamps

not in your code: bytes, 0 to 255



Round 2

Light from a lamp

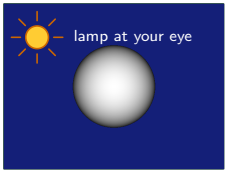
$$\text{brightness} = \mathbf{n} \cdot \mathbf{l}$$

Brightness is $\mathbf{n} \cdot \mathbf{l}$: you have done it twice

Lecture 16, slide 18: brightness goes with $\cos\theta = \mathbf{n} \cdot \mathbf{l}$, $\mathbf{l}$ toward the lamp.

- You Try 1 was $\mathbf{n} \cdot \mathbf{l}$ with $\mathbf{l}$ toward you.

You Try 1



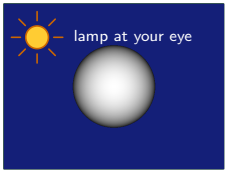
$\mathbf{l} = (0,\ 0,\ 1)$:
toward you

$\mathbf{n} \cdot \mathbf{l} = n.z$

Brightness is $\mathbf{n} \cdot \mathbf{l}$: you have done it twice

Lecture 16, slide 18: brightness goes with $\cos\theta = \mathbf{n} \cdot \mathbf{l}$, $\mathbf{l}$ toward the lamp.

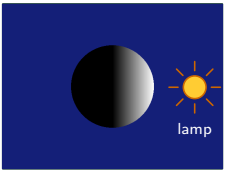
You Try 1



$\mathbf{l} = (0,\ 0,\ 1)$:
toward you

$\mathbf{n} \cdot \mathbf{l} = n.z$

You Try 2



$\mathbf{l} = (1,\ 0,\ 0)$:
to the right

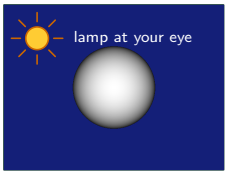
$\mathbf{n} \cdot \mathbf{l} = n.x$

- You Try 1 was $\mathbf{n} \cdot \mathbf{l}$ with $\mathbf{l}$ toward you.
- You Try 2 was $\mathbf{n} \cdot \mathbf{l}$ with $\mathbf{l}$ to the right.

Brightness is $\mathbf{n} \cdot \mathbf{l}$: you have done it twice

Lecture 16, slide 18: brightness goes with $\cos\theta = \mathbf{n} \cdot \mathbf{l}$, $\mathbf{l}$ toward the lamp.

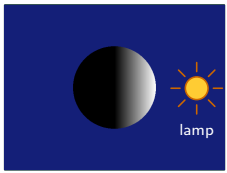
You Try 1



$\mathbf{l} = (0,\ 0,\ 1)$: toward you

$\mathbf{n} \cdot \mathbf{l} = n.z$

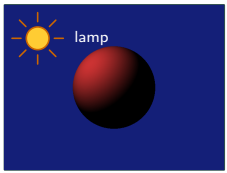
You Try 2



$\mathbf{l} = (1,\ 0,\ 0)$: to the right

$\mathbf{n} \cdot \mathbf{l} = n.x$

Any lamp



$\mathbf{l}$: toward a lamp up on the left

`dot(n, l)`

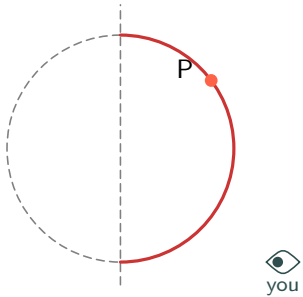
- You Try 1 was $\mathbf{n} \cdot \mathbf{l}$ with $\mathbf{l}$ toward you.
- You Try 2 was $\mathbf{n} \cdot \mathbf{l}$ with $\mathbf{l}$ to the right.
- Any lamp: dot(n, l). Only $\mathbf{l}$ changes.

```
let diffuse = dot(n, l);
```

dot: built into WGSL

1: from the point toward the lamp

seen from the side





- The lamp is a point in front of the screen.

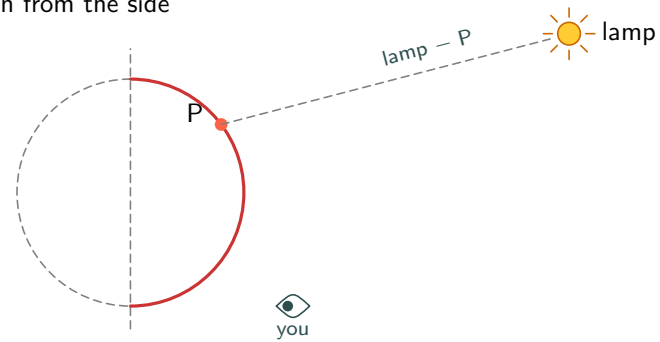
```
let lamp = vec3f(-1.0, 1.0, 1.0);
let P = vec3f(x, y, z);
```

up, left, in front of the screen

the point this pixel sees

l: from the point toward the lamp

seen from the side



- The lamp is a point in front of the screen.
- lamp − P runs from P to the lamp.

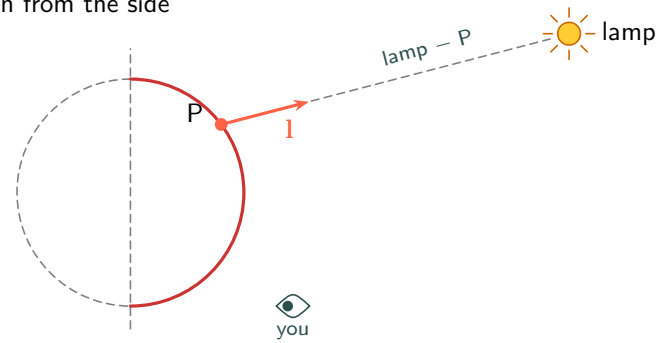
```
let lamp = vec3f(-1.0, 1.0, 1.0);
let P = vec3f(x, y, z);
```

up, left, in front of the screen

the point this pixel sees

l: from the point toward the lamp

seen from the side



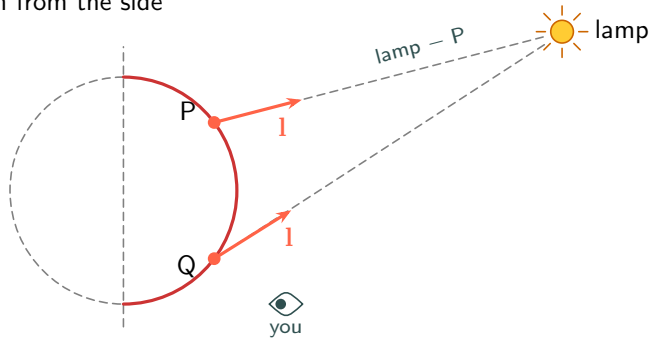
- The lamp is a point in front of the screen.
- lamp − P runs from P to the lamp.
- normalize makes it length 1: that is $\mathbf{l}$.

```
let lamp = vec3f(-1.0, 1.0, 1.0);      up, left, in front of the screen
let P = vec3f(x, y, z);                the point this pixel sees
let l = normalize(lamp - P);           normalize: built into WGSL
```

normalize(a) is a divided by its length: lecture 17 made $\mathbf{h}$ the same way.

l: from the point toward the lamp

seen from the side



- The lamp is a point in front of the screen.
- lamp − P runs from P to the lamp.
- normalize makes it length 1: that is $\mathbf{l}$.
- Every point gets its own $\mathbf{l}$.

```
let lamp = vec3f(-1.0, 1.0, 1.0);      up, left, in front of the screen
let P = vec3f(x, y, z);                the point this pixel sees
let l = normalize(lamp - P);           normalize: built into WGSL
```

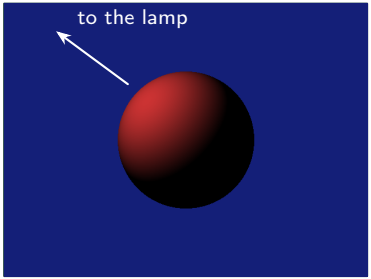
normalize(a) is a divided by its length: lecture 17 made $\mathbf{h}$ the same way.

The lamp lights the ball

```
let diffuse = max(dot(n, l), 0.0);
let c = base * diffuse;
```

max(a, b), built into WGSL: the bigger of the two.

base: the ball's own colour, lecture 17's red $(0.8,\ 0.2,\ 0.2)$.



- diffuse is $\mathbf{n} \cdot \mathbf{l}$, never below 0.

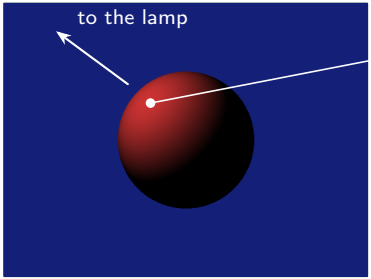
▶ **Run it in the Shader Lab:**
The lamp: diffuse light

The lamp lights the ball

```
let diffuse = max(dot(n, l), 0.0);
let c = base * diffuse;
```

max(a, b), built into WGSL: the bigger of the two.

base: the ball's own colour, lecture 17's red $(0.8,\ 0.2,\ 0.2)$.



faces the lamp:
$\mathbf{n} \cdot \mathbf{l}$ near 1, bright

- diffuse is $\mathbf{n} \cdot \mathbf{l}$, never below 0.
- Where the ball faces the lamp, it is bright.

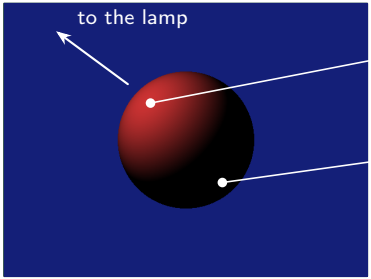
▶ **Run it in the Shader Lab:**
The lamp: diffuse light

The lamp lights the ball

```
let diffuse = max(dot(n, l), 0.0);
let c = base * diffuse;
```

max(a, b), built into WGSL: the bigger of the two.

base: the ball's own colour, lecture 17's red $(0.8,\ 0.2,\ 0.2)$.



faces the lamp:
$\mathbf{n} \cdot \mathbf{l}$ near 1, bright

faces away: $\mathbf{n} \cdot \mathbf{l} < 0$,
0: black

Left out today: lecture 16's $1/r^2$. The ball is small, so r hardly changes over it.

- diffuse is $\mathbf{n} \cdot \mathbf{l}$, never below 0.
- Where the ball faces the lamp, it is bright.
- Facing away: 0, black (lecture 16, slide 20).

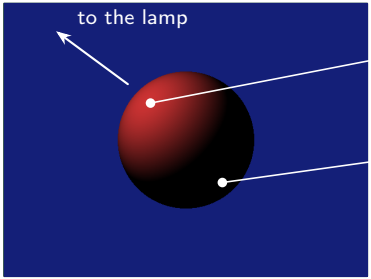
▶ **Run it in the Shader Lab:**
The lamp: diffuse light

The lamp lights the ball

```
let diffuse = max(dot(n, l), 0.0);
let c = base * diffuse;
```

max(a, b), built into WGSL: the bigger of the two.

base: the ball's own colour, lecture 17's red $(0.8,\ 0.2,\ 0.2)$.



faces the lamp:
$\mathbf{n} \cdot \mathbf{l}$ near 1, bright

faces away: $\mathbf{n} \cdot \mathbf{l} < 0$,
0: black

Left out today: lecture 16's $1/r^2$. The ball is small, so r hardly changes over it.

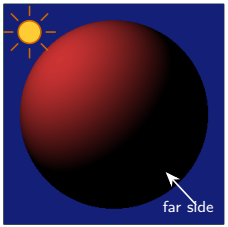
- diffuse is $\mathbf{n} \cdot \mathbf{l}$, never below 0.
- Where the ball faces the lamp, it is bright.
- Facing away: 0, black (lecture 16, slide 20).
- Here the GPU's clamp would do the same. max matters next slide.

▶ **Run it in the Shader Lab:**
The lamp: diffuse light

Ambient light: the far side is not black

```
let ambient = 0.1;
let c = base * (ambient + diffuse);
```

Lecture 16, slide 42: ambient light, a little light from every side.



ambient $= 0$
the far side is black

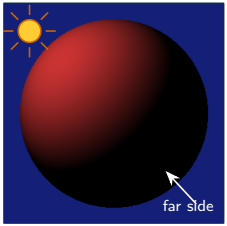
- With the lamp alone, the far side is black.

▶ **Run it in the Shader Lab:**
Lamp and ambient light

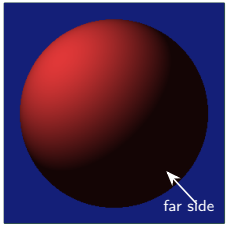
Ambient light: the far side is not black

```
let ambient = 0.1;
let c = base * (ambient + diffuse);
```

Lecture 16, slide 42: ambient light, a little light from every side.



ambient $= 0$
the far side is black



ambient $= 0.1$
the far side shows red

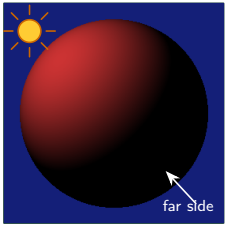
- With the lamp alone, the far side is black.
- Ambient adds 0.1 everywhere: the far side shows the ball.

▶ **Run it in the Shader Lab:**
Lamp and ambient light

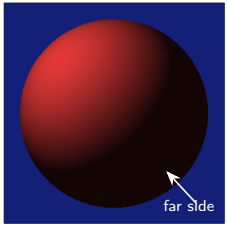
Ambient light: the far side is not black

```
let ambient = 0.1;
let c = base * (ambient + diffuse);
```

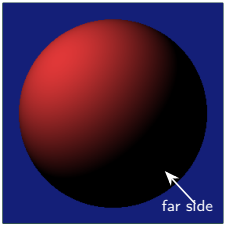
Lecture 16, slide 42: ambient light, a little light from every side.



ambient $= 0$
the far side is black



ambient $= 0.1$
the far side shows red



no max: $0.1+$ a negative
black again

```
diffuse = dot(n, l);
```

- With the lamp alone, the far side is black.
- Ambient adds 0.1 everywhere: the far side shows the ball.
- No max: $0.1 + (-0.5) = -0.4$, black. The GPU clamps only the final colour.

▶ **Run it in the Shader Lab:**
Lamp and ambient light

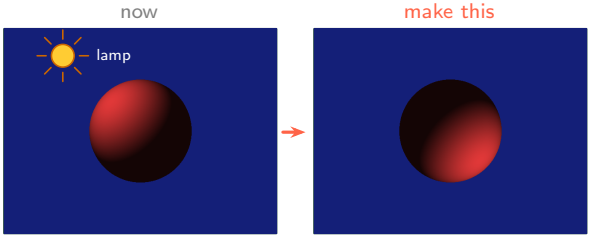
You Try 3: the lamp to the lower right

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

```
let lamp = vec3f(-1.0, 1.0, 1.0);
let P = vec3f(x, y, z);
let l = normalize(lamp - P);
```

change this line

lamp: x right, y up, z toward you (in front of the screen).



- **Move the lamp to the lower right.**
- Guess first: which two numbers change, and to what sign?

You Try 3: the lamp to the lower right

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

```
let lamp = vec3f(-1.0, 1.0, 1.0);
let P = vec3f(x, y, z);
let l = normalize(lamp - P);
```

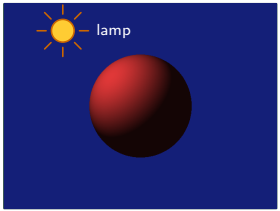
change this line

lamp: x right, y up, z toward you (in front of the screen).

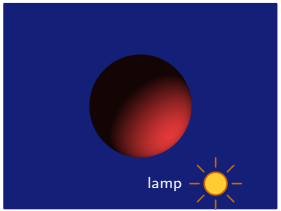
answer: `let lamp = vec3f(1.0, -1.0, 1.0);`

x: $-1 \to +1$, to the right; y: $+1 \to -1$, down; z stays

now



make this



- **Move the lamp to the lower right.**
- Guess first: which two numbers change, and to what sign?
- x becomes $+1$ (right), y becomes -1 (down). z stays: in front.

You Try 4: a blue lamp

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

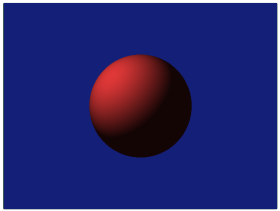
```
let base = vec3f(0.8, 0.2, 0.2);
let light = vec3f(1.0, 1.0, 1.0);
let c = base * (ambient + light * diffuse);
```

change this line

light: the lamp's colour. * on two vec3f: red times red, green times green, blue times blue.

- **Make the lamp blue: (0.3, 0.5, 1.0).**
- Before you run it: what colour will the red ball look?

now



after your change



You Try 4: a blue lamp

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

```
let base = vec3f(0.8, 0.2, 0.2);
let light = vec3f(1.0, 1.0, 1.0);
let c = base * (ambient + light * diffuse);
```

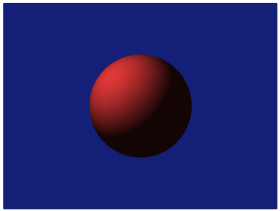
change this line

light: the lamp's colour. * on two vec3f: red times red, green times green, blue times blue.

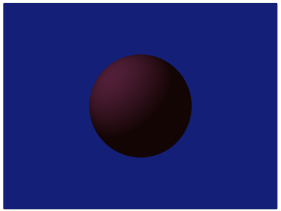
answer: `let light = vec3f(0.3, 0.5, 1.0);`

ball $(0.8, 0.2, 0.2)$ $\times$ lamp $(0.3, 0.5, 1.0)$ $=$ $(0.24, 0.1, 0.2)$

now



after your change



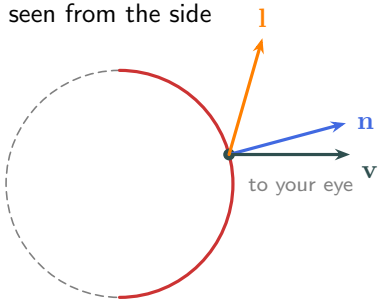
- **Make the lamp blue: (0.3, 0.5, 1.0).**
- Before you run it: what colour will the red ball look?
- Dark purple: each channel times the lamp's (lecture 16, slide 35).

Round 3

A highlight, and the mouse

lecture 17's half vector, in one more line

The highlight: lecture 17's half vector

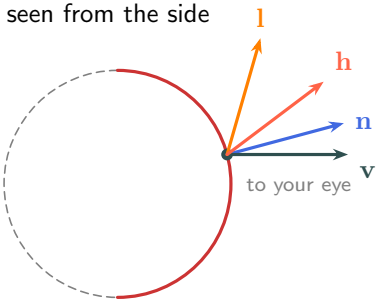


- $\mathbf{v}$ points to your eye: $(0, 0, 1)$ for every pixel.

```
let v = vec3f(0.0, 0.0, 1.0);
```

▶ **Run it in the Shader Lab:**
Lamp, ambient and a highlight

The highlight: lecture 17's half vector

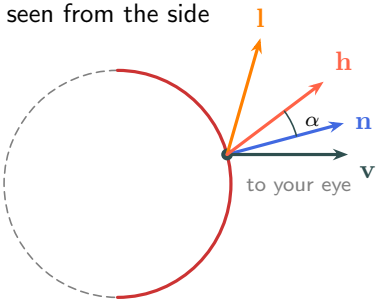



```
let v = vec3f(0.0, 0.0, 1.0);
let h = normalize(l + v);
```

- $\mathbf{v}$ points to your eye: $(0, 0, 1)$ for every pixel.
- $\mathbf{h}$ is halfway between $\mathbf{l}$ and $\mathbf{v}$ (lecture 17, slide 35).

▶ **Run it in the Shader Lab:**
Lamp, ambient and a highlight

The highlight: lecture 17's half vector



- $\mathbf{v}$ points to your eye: $(0, 0, 1)$ for every pixel.
- $\mathbf{h}$ is halfway between $\mathbf{l}$ and $\mathbf{v}$ (lecture 17, slide 35).
- Brightest where $\mathbf{h}$ lies on $\mathbf{n}$, so α is 0.

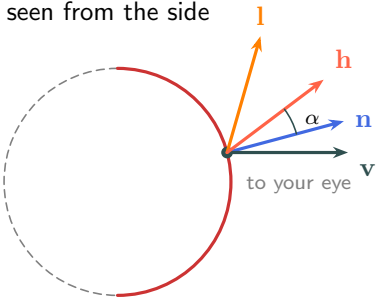
```
let v = vec3f(0.0, 0.0, 1.0);
let h = normalize(l + v);
let shine = pow(max(dot(n, h), 0.0), 50.0);
```

pow(a, 50.0) is a^{50}, built into WGSL (lecture 17, slide 38).

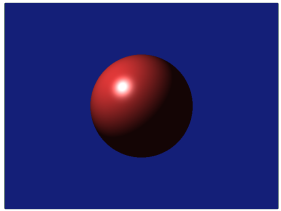
▶ **Run it in the Shader Lab:**
Lamp, ambient and a highlight

The highlight: lecture 17's half vector

seen from the side



The shine is white: the clear top layer reflects every colour alike (lecture 16, slide 26).



- $\mathbf{v}$ points to your eye: $(0, 0, 1)$ for every pixel.
- $\mathbf{h}$ is halfway between $\mathbf{l}$ and $\mathbf{v}$ (lecture 17, slide 35).
- Brightest where $\mathbf{h}$ lies on $\mathbf{n}$, so α is 0.

```
let v = vec3f(0.0, 0.0, 1.0);
let h = normalize(l + v);
let shine = pow(max(dot(n, h), 0.0), 50.0);
let c = base * (ambient + diffuse) + shine;
```

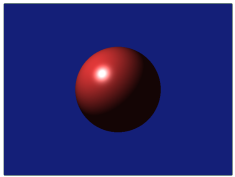
pow(a, 50.0) is a^{50}, built into WGSL (lecture 17, slide 38).

▶ **Run it in the Shader Lab:**
Lamp, ambient and a highlight

The power sets the highlight's size

```
let shine = pow(max(dot(n, h), 0.0), 50.0);
```

Lecture 17, slide 38 drew the curves $(\mathbf{n} \cdot \mathbf{h})^p$. These are the pictures.



50.0

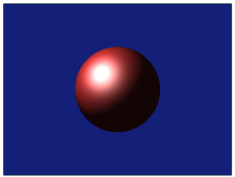
the one above

- The power sets the size of the spot.

The power sets the highlight's size

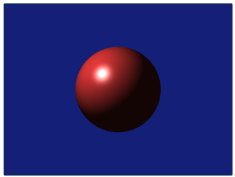
```
let shine = pow(max(dot(n, h), 0.0), 50.0);
```

Lecture 17, slide 38 drew the curves $(\mathbf{n} \cdot \mathbf{h})^p$. These are the pictures.



10.0

a wide, soft spot



50.0

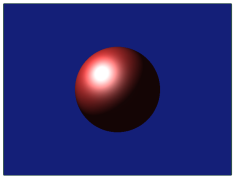
the one above

- The power sets the size of the spot.
- A small power gives a wide, soft spot.

The power sets the highlight's size

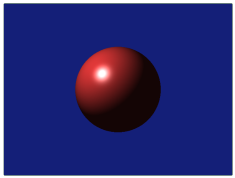
```
let shine = pow(max(dot(n, h), 0.0), 50.0);
```

Lecture 17, slide 38 drew the curves $(\mathbf{n} \cdot \mathbf{h})^p$. These are the pictures.



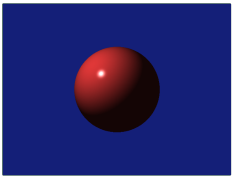
10.0

a wide, soft spot



50.0

the one above



250.0

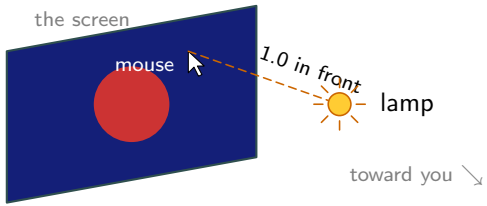
small and sharp

- The power sets the size of the spot.
- A small power gives a wide, soft spot.
- A big power gives a small, sharp spot, like polished plastic.

The mouse is the lamp

```
let lamp = vec3f(mx, my, 1.0);
```

mx, my: the mouse in centred coordinates, like x and y (lecture 26).



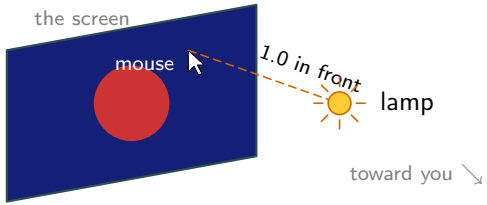
- The lamp floats 1.0 in front of the screen, over the mouse.

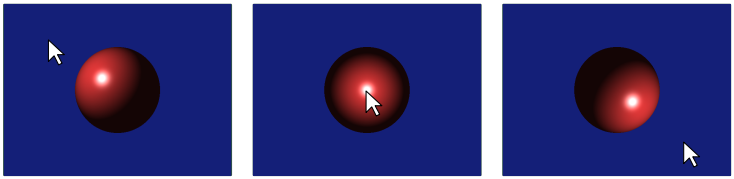
▶ **Run it in the Shader Lab:**
The mouse is the lamp

The mouse is the lamp

```
let lamp = vec3f(mx, my, 1.0);
```

mx, my: the mouse in centred coordinates, like x and y (lecture 26).





- The lamp floats 1.0 in front of the screen, over the mouse.
- Move the mouse: the light and the highlight follow.

▶ **Run it in the Shader Lab:**
The mouse is the lamp

You Try 5: a sharper highlight

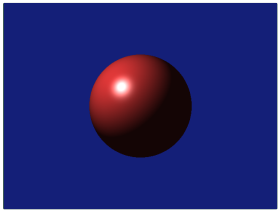
In the Shader Lab, choose **You Try 5**. Guess first, then change the code.

```
let h = normalize(l + v);
let shine = pow(max(dot(n, h), 0.0), 50.0);
let c = base * (ambient + diffuse) + shine;
```

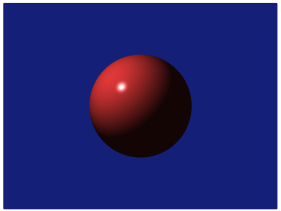
change this line

- **Make the highlight smaller and sharper, like a billiard ball's.**
- Guess first: a bigger power, or a smaller one?

now



make this



You Try 5: a sharper highlight

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

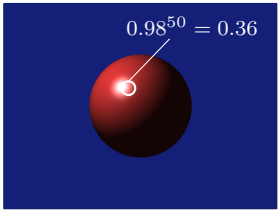
```
let h = normalize(l + v);
let shine = pow(max(dot(n, h), 0.0), 50.0);
let c = base * (ambient + diffuse) + shine;
```

change this line

answer: `let shine = pow(max(dot(n, h), 0.0), 250.0);`

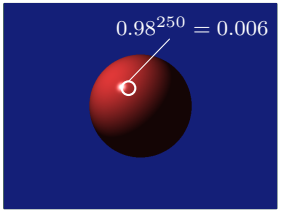
a pixel near the spot, $\mathbf{n} \cdot \mathbf{h} = 0.98$:

now



still bright

make this



dark: the spot shrinks

- **Make the highlight smaller and sharper, like a billiard ball's.**
- Guess first: a bigger power, or a smaller one?
- Bigger: 250.0 makes a small, sharp spot.

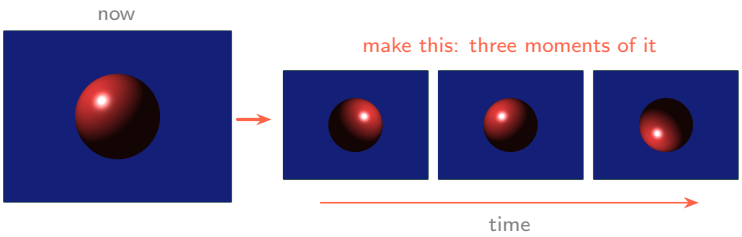
You Try 6: the lamp goes round

In the Shader Lab, choose **You Try 6**. Guess first, then change the code.

```
let lamp = vec3f(-1.0, 1.0, 1.0);
let P = vec3f(x, y, z);
let l = normalize(lamp - P);
```

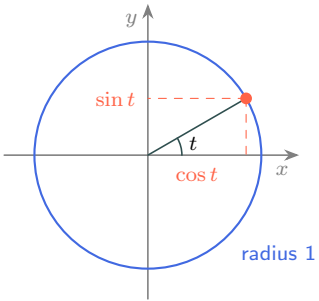
change this line

Hint: u.time grows every frame (lecture 26). And $(\cos t,\ \sin t)$?



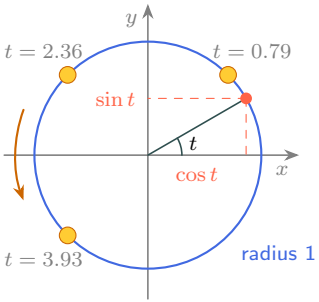
- **Make the lamp go round the ball, on its own.**
- Guess first: which line must use u.time?

You Try 6, the answer: a lamp on a circle



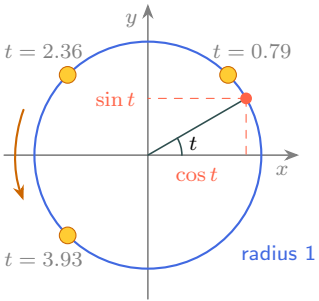
1. A point at angle t on a circle of radius 1: $(\cos t,\ \sin t)$

You Try 6, the answer: a lamp on a circle



1. A point at angle t on a circle of radius 1: $(\cos t,\ \sin t)$
2. t = u.time grows each frame: the point goes round

You Try 6, the answer: a lamp on a circle

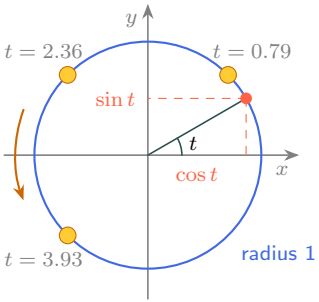


1. A point at angle t on a circle of radius 1: $(\cos t,\ \sin t)$
2. t = u.time grows each frame: the point goes round
3. The lamp's x and y ride on it; $z = 1$ keeps it in front:

```
let lamp = vec3f(cos(u.time), sin(u.time), 1.0);
```

cos, like sin, is built into WGSL. One turn takes $2\pi \approx 6.3$ seconds.

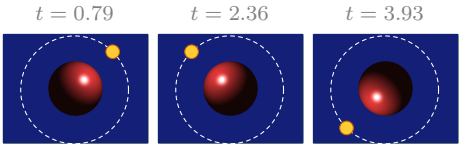
You Try 6, the answer: a lamp on a circle



1. A point at angle t on a circle of radius 1: $(\cos t,\ \sin t)$
2. t = u.time grows each frame: the point goes round
3. The lamp's x and y ride on it; $z = 1$ keeps it in front:

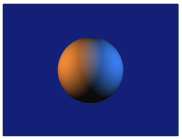
```
let lamp = vec3f(cos(u.time), sin(u.time), 1.0);
```

cos, like sin, is built into WGSL. One turn takes $2\pi \approx 6.3$ seconds.



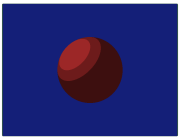
the highlight follows the lamp

Now make your own picture



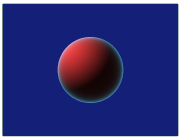
Two coloured lamps

`warm * d1 + cool * d2`



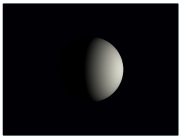
A cartoon ball

`floor(diffuse * 3.0) / 3.0`



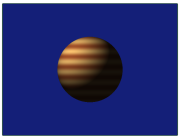
A glowing rim

`pow(1.0 - n.z, 3.0)`



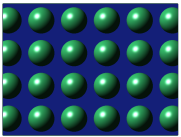
The moon's phases

`the lamp goes behind the ball`



A striped planet

`sin(20.0 * n.y)`



A tray of marbles

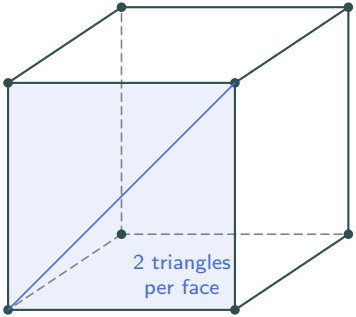
`fract tiles, a ball in each`

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

Summary

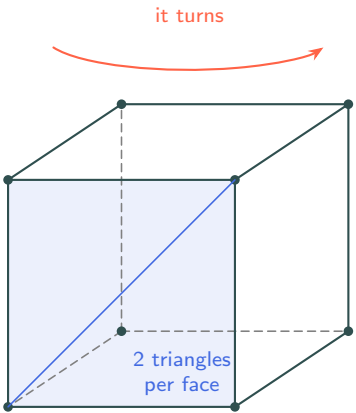
1 A circle becomes a ball	$z = \sqrt{r^2 - x^2 - y^2}$ each pixel looks straight in and meets the ball's front
2 The normal	$\mathbf{n} = (x,\ y,\ z)\,/\,r$ from the centre out through the point; check it as a colour
3 Light from a lamp	`max(dot(n, l), 0.0)` $\mathbf{l}$ = normalize(lamp − P), plus a little ambient
4 A highlight	`pow(max(dot(n, h), 0.0), p)` $\mathbf{h}$ = normalize(l + v); a bigger p, a smaller spot

Next lecture: into 3D



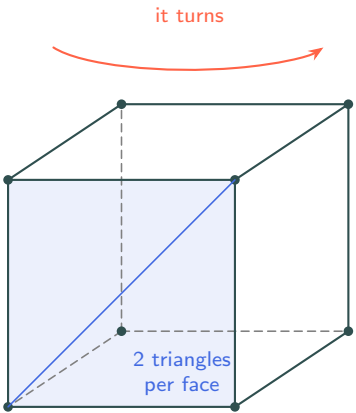
- A cube: 8 corners, 12 triangles, in lecture 25's buffers.

Next lecture: into 3D



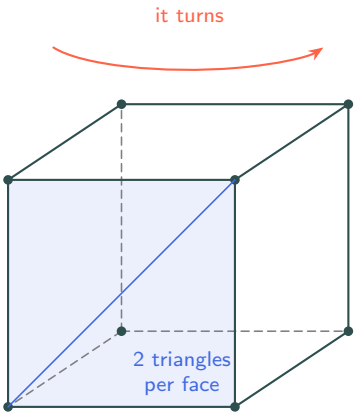
- A cube: 8 corners, 12 triangles, in lecture 25's buffers.
- The vertex shader moves every corner: model, view, projection.

Next lecture: into 3D



- A cube: 8 corners, 12 triangles, in lecture 25's buffers.
- The vertex shader moves every corner: model, view, projection.
- The depth buffer keeps the nearest (lecture 24).

Next lecture: into 3D



- A cube: 8 corners, 12 triangles, in lecture 25's buffers.
- The vertex shader moves every corner: model, view, projection.
- The depth buffer keeps the nearest (lecture 24).
- Lecture 29: today's light on a real model.

Exit Ticket

Today's question *(2 minutes)*

(a) A fake ball has centre $(0, 0, 0)$ and radius $r = 0.5$. The pixel is $(x, y) = (0.4,\ 0)$. Find z, then the normal $\mathbf{n} = (x, y, z)/r$.

(b) The lamp is straight in front, so $\mathbf{l} = (0, 0, 1)$. How bright is this pixel: what is $\mathbf{n} \cdot \mathbf{l}$?

CS 453 — Exit Ticket #27

Name: _____

Roll #: _____ Date: _____

Answer:

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