Lecture 20: Monte Carlo and the Path Tracer

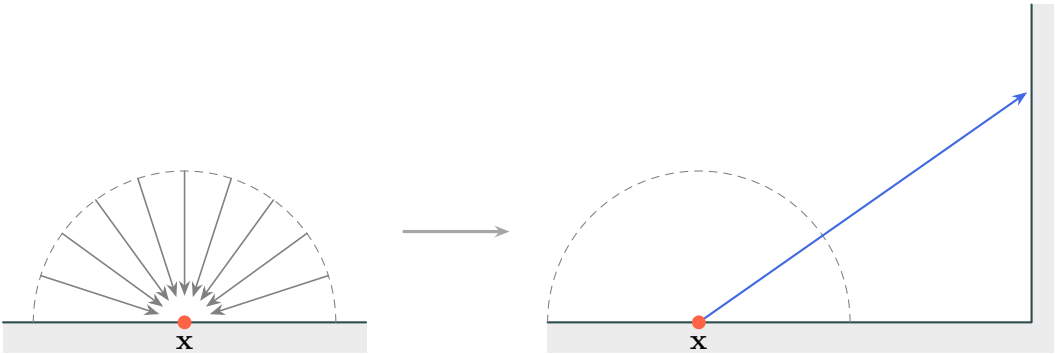
One Random Ray per Bounce

CS 453 — Computer Graphics

Fakhir Shaheen

Forman Christian University

Today: one random ray in place of the whole dome

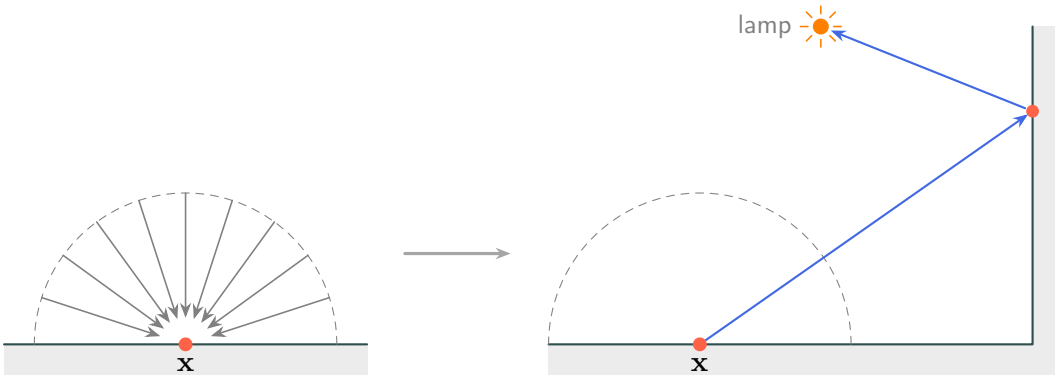


lecture 19: light from every direction

today: one direction, picked at random

- Estimate the integral with random samples.

Today: one random ray in place of the whole dome



lecture 19: light from every direction

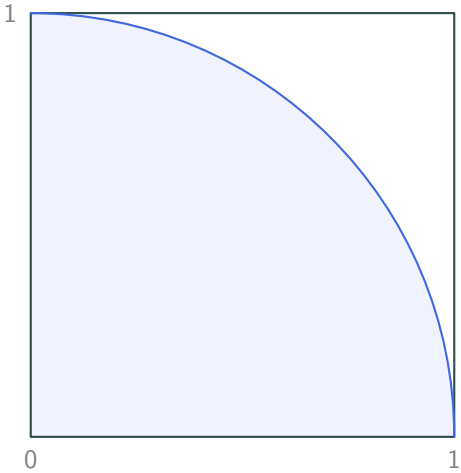
today: one direction, picked at random

- Estimate the integral with random samples.
- Follow each one with `trace`.

Estimating by random sampling

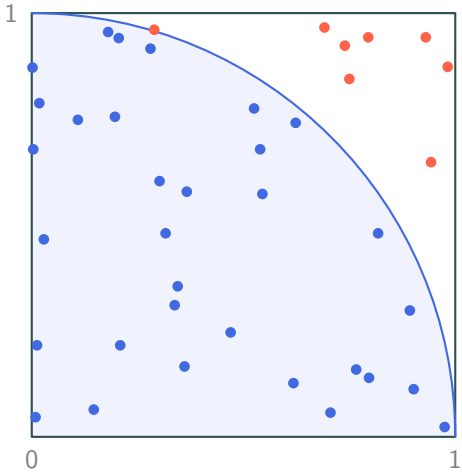
The Monte Carlo method, with darts

Darts measure the area of a quarter circle



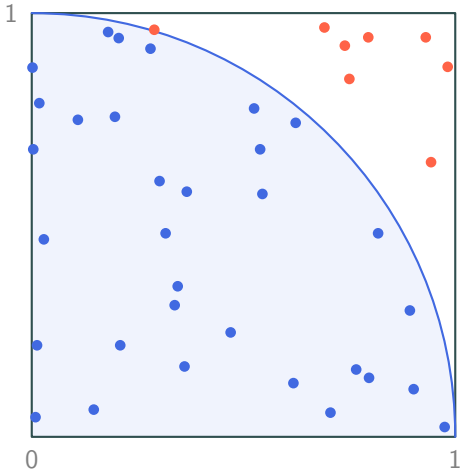
- Goal: the area of the quarter circle, without its formula.

Darts measure the area of a quarter circle



- Goal: the area of the quarter circle, without its formula.
- Throw darts evenly over the square.

Darts measure the area of a quarter circle

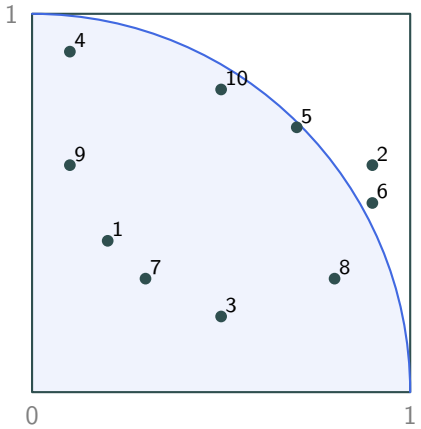


- Goal: the area of the quarter circle, without its formula.
- Throw darts evenly over the square.
- The square's area is 1, so the share of darts inside estimates the area.

$$\frac{32}{40} = 0.8 \qquad \text{exact: } \frac{\pi}{4} \approx 0.785$$

a dart (x, y) is inside when $x^2 + y^2 \leq 1$

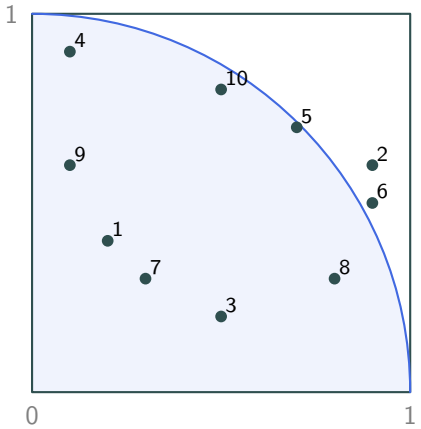
Worked: ten darts, on paper



dart	x	y
1	0.2	0.4
2	0.9	0.6
3	0.5	0.2
4	0.1	0.9
5	0.7	0.7
6	0.9	0.5
7	0.3	0.3
8	0.8	0.3
9	0.1	0.6
10	0.5	0.8

- Try first: which darts land inside? Estimate the area.

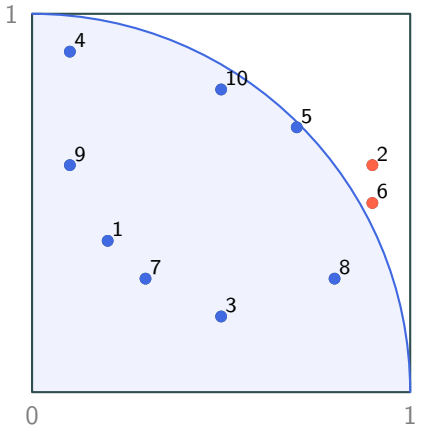
Worked: ten darts, on paper



dart	x	y	$x^2 + y^2$
1	0.2	0.4	0.20
2	0.9	0.6	1.17
3	0.5	0.2	0.29
4	0.1	0.9	0.82
5	0.7	0.7	0.98
6	0.9	0.5	1.06
7	0.3	0.3	0.18
8	0.8	0.3	0.73
9	0.1	0.6	0.37
10	0.5	0.8	0.89

- Try first: which darts land inside? Estimate the area.
- Inside when $x^2 + y^2 \leq 1$: darts 2 and 6 are out.

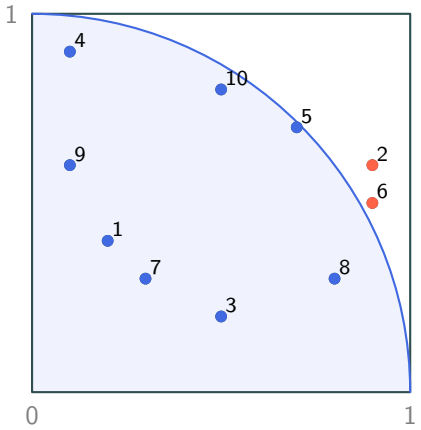
Worked: ten darts, on paper



dart	x	y	$x^2 + y^2$
1	0.2	0.4	0.20
2	0.9	0.6	1.17
3	0.5	0.2	0.29
4	0.1	0.9	0.82
5	0.7	0.7	0.98
6	0.9	0.5	1.06
7	0.3	0.3	0.18
8	0.8	0.3	0.73
9	0.1	0.6	0.37
10	0.5	0.8	0.89

- Try first: which darts land inside? Estimate the area.
- Inside when $x^2 + y^2 \leq 1$: darts 2 and 6 are out.
- 8 of 10 inside: the area is about 0.8.

Worked: ten darts, on paper



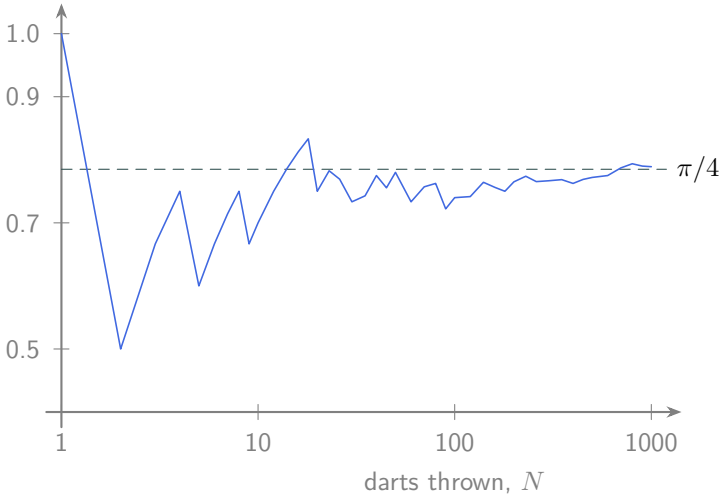
dart	x	y	$x^2 + y^2$
1	0.2	0.4	0.20
2	0.9	0.6	1.17
3	0.5	0.2	0.29
4	0.1	0.9	0.82
5	0.7	0.7	0.98
6	0.9	0.5	1.06
7	0.3	0.3	0.18
8	0.8	0.3	0.73
9	0.1	0.6	0.37
10	0.5	0.8	0.89

- Try first: which darts land inside? Estimate the area.
- Inside when $x^2 + y^2 \leq 1$: darts 2 and 6 are out.
- 8 of 10 inside: the area is about 0.8.

exact: $\pi/4 \approx 0.785$

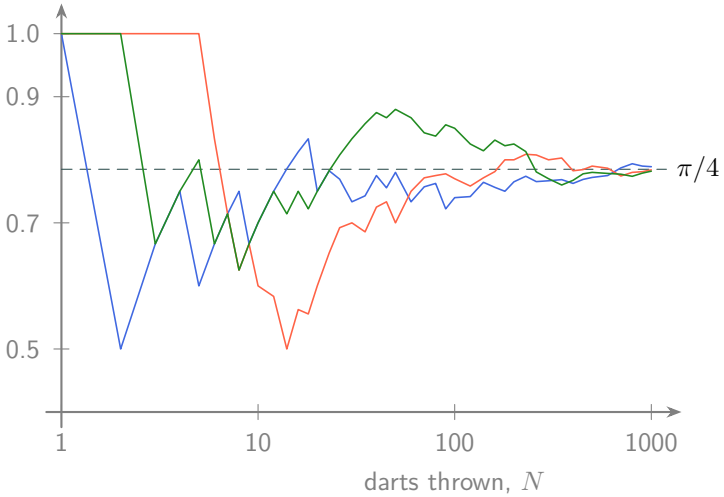
and $\pi \approx 4 \times 0.8 = 3.2$

More darts: closer to the true area, but slowly



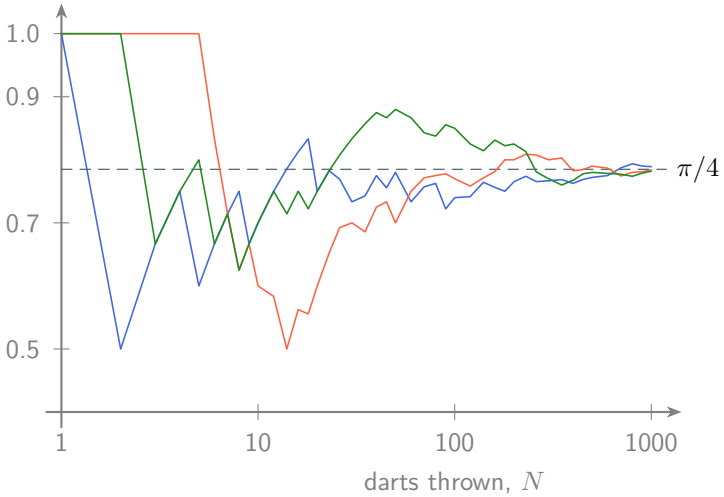
- One run: the estimate after each dart.

More darts: closer to the true area, but slowly



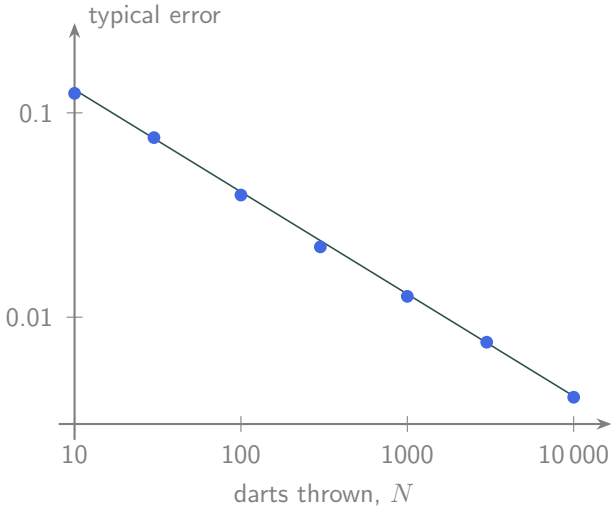
- One run: the estimate after each dart.
- Every run gives a different answer.

More darts: closer to the true area, but slowly



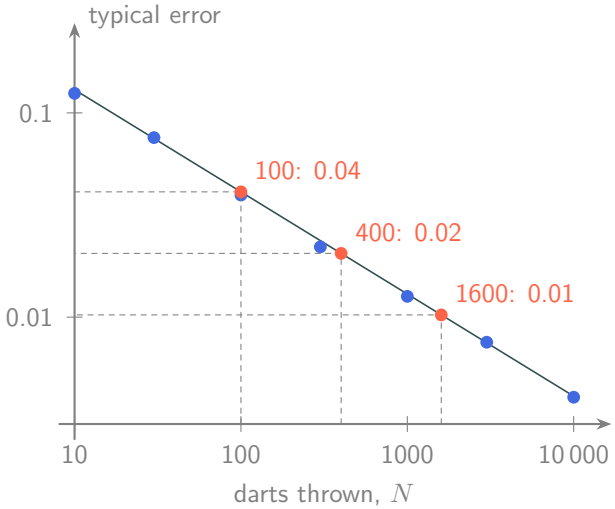
- One run: the estimate after each dart.
- Every run gives a different answer.
- All of them close in on $\pi/4$, slowly.

The error falls as one over $\sqrt{N}$



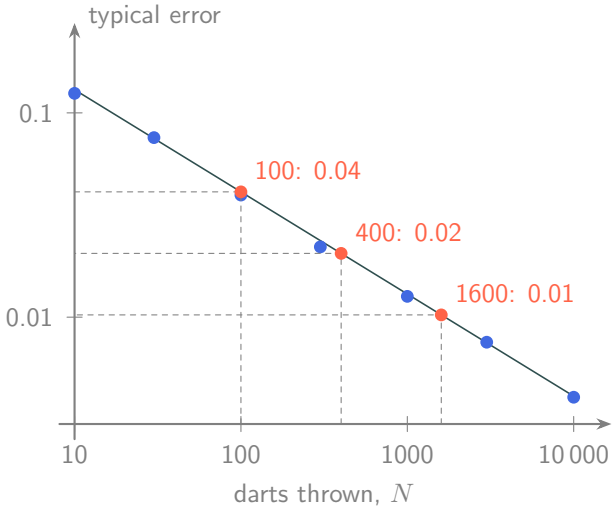
- The typical error is about $0.41/\sqrt{N}$.

The error falls as one over $\sqrt{N}$



- The typical error is about $0.41/\sqrt{N}$.
- Four times the darts give half the error.

The error falls as one over $\sqrt{N}$

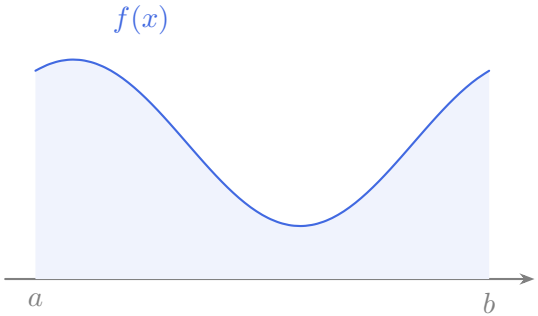


- The typical error is about $0.41/\sqrt{N}$.
- Four times the darts give half the error.
- One more correct digit costs 100 times the darts.

true of every Monte Carlo estimate

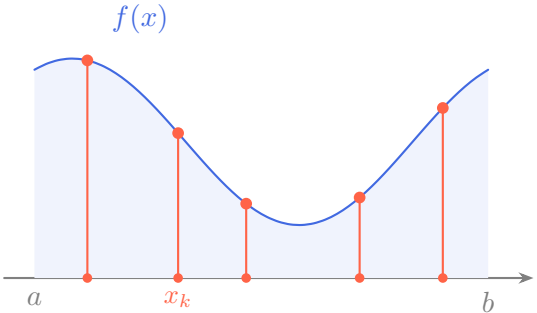
The Monte Carlo estimate of any integral

$$\int_a^b f(x)\,dx \;\approx\; \frac{1}{N}\sum_{k=1}^{N}\frac{f(x_k)}{p(x_k)}$$



the area under f is the integral

The Monte Carlo estimate of any integral

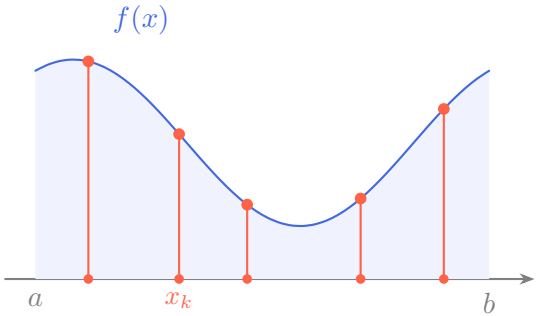


the area under f is the integral

$$\int_a^b f(x)\,dx \;\approx\; \frac{1}{N}\sum_{k=1}^{N}\frac{f(x_k)}{p(x_k)}$$

- Pick N points x_k at random.

The Monte Carlo estimate of any integral

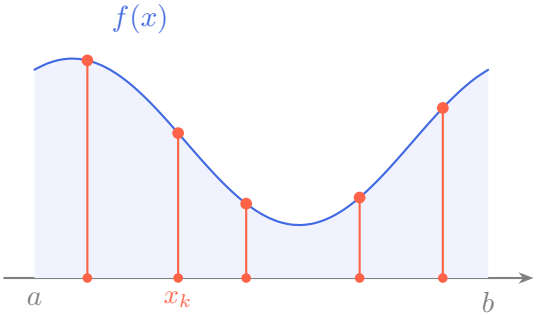


the area under f is the integral

$$\int_a^b f(x)\,dx \;\approx\; \frac{1}{N}\sum_{k=1}^{N}\frac{f(x_k)}{p(x_k)}$$

- Pick N points x_k at random.
- $p(x)$ says how often we pick near x.

The Monte Carlo estimate of any integral

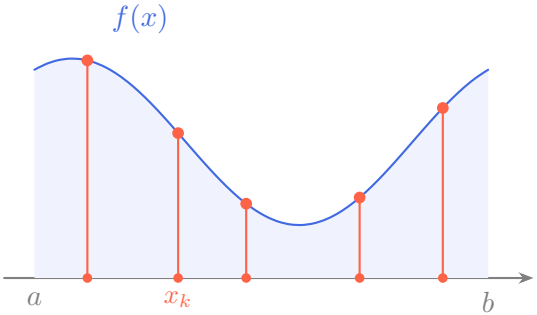


the area under f is the integral

$$\int_a^b f(x)\,dx \;\approx\; \frac{1}{N}\sum_{k=1}^{N}\frac{f(x_k)}{p(x_k)}$$

- Pick N points x_k at random.
- $p(x)$ says how often we pick near x.
- Even picks: $p = \dfrac{1}{b-a}$, so the estimate is $(b-a)$ times the average f.

The Monte Carlo estimate of any integral



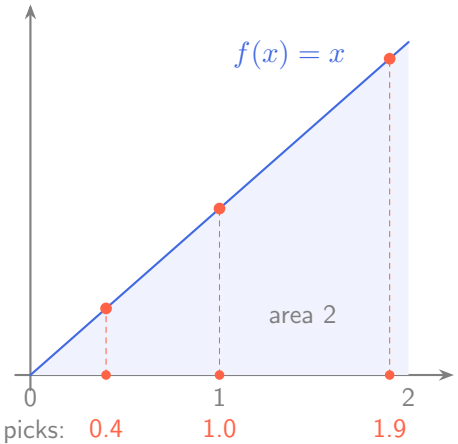
the area under f is the integral

$$\int_a^b f(x)\,dx \;\approx\; \frac{1}{N}\sum_{k=1}^{N}\frac{f(x_k)}{p(x_k)}$$

- Pick N points x_k at random.
- $p(x)$ says how often we pick near x.
- Even picks: $p = \dfrac{1}{b-a}$, so the estimate is $(b-a)$ times the average f.

darts: f is 1 inside, 0 outside; $p = 1$

Worked: one integral, two ways to pick

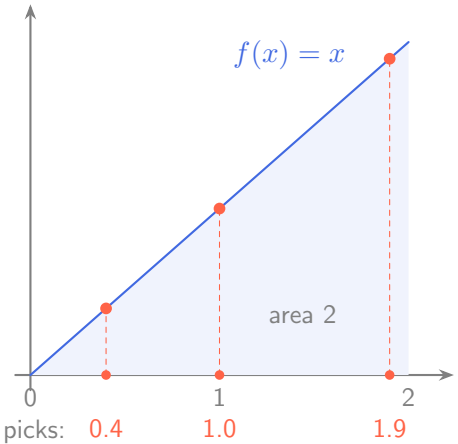


$\int_0^2 x\,dx = 2$. Estimate it from three picks.

x_k	even: $p = 1/2$	shaped: $p = x/2$
0.4		
1.0		
1.9		
average		

- Try first: $f(x_k)/p(x_k)$ for each pick, both ways.

Worked: one integral, two ways to pick

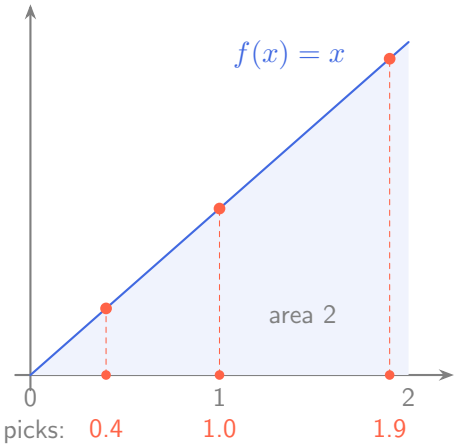


$\int_0^2 x\,dx = 2$. Estimate it from three picks.

x_k	even: $p = 1/2$	shaped: $p = x/2$
0.4	$0.4/0.5 = 0.8$	
1.0	2.0	
1.9	3.8	
average	2.2	

- Try first: $f(x_k)/p(x_k)$ for each pick, both ways.
- Even picks: right on average, each one off.

Worked: one integral, two ways to pick

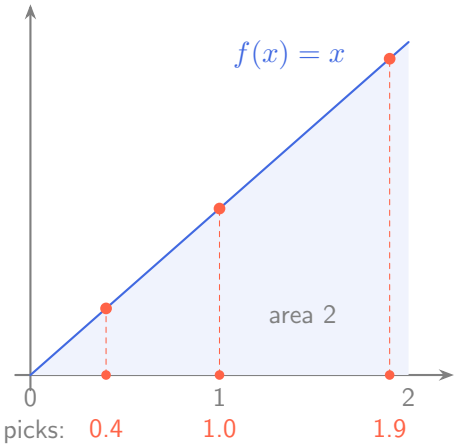


$\int_0^2 x\,dx = 2$. Estimate it from three picks.

x_k	even: $p = 1/2$	shaped: $p = x/2$
0.4	$0.4/0.5 = 0.8$	$0.4/0.2 = 2$
1.0	2.0	2
1.9	3.8	2
average	2.2	2

- Try first: $f(x_k)/p(x_k)$ for each pick, both ways.
- Even picks: right on average, each one off.

Worked: one integral, two ways to pick



$\int_0^2 x\,dx = 2$. Estimate it from three picks.

x_k	even: $p = 1/2$	shaped: $p = x/2$
0.4	$0.4/0.5 = 0.8$	$0.4/0.2 = 2$
1.0	2.0	2
1.9	3.8	2
average	2.2	2

- Try first: $f(x_k)/p(x_k)$ for each pick, both ways.
- Even picks: right on average, each one off.
- When p has the shape of f, every pick is exact.

  $p = x/2$ adds up to 1 over $[0, 2]$

Quick check 1: how many darts?

With 100 darts the typical error is about 0.04.

1. About how many darts give an error of 0.02?
2. About how many darts give 0.004?
3. What error do 200 darts give, about?
4. Does the error depend on the order of the darts?

- The error falls as $1/\sqrt{N}$.

Quick check 1: how many darts?

With 100 darts the typical error is about 0.04.

1. About how many darts give an error of 0.02?	400
2. About how many darts give 0.004?	10 000
3. What error do 200 darts give, about?	about 0.028
4. Does the error depend on the order of the darts?	no

- The error falls as $1/\sqrt{N}$.

Recap: estimating by random sampling

$$\int f \approx \frac{1}{N} \sum \frac{f(x_k)}{p(x_k)}$$

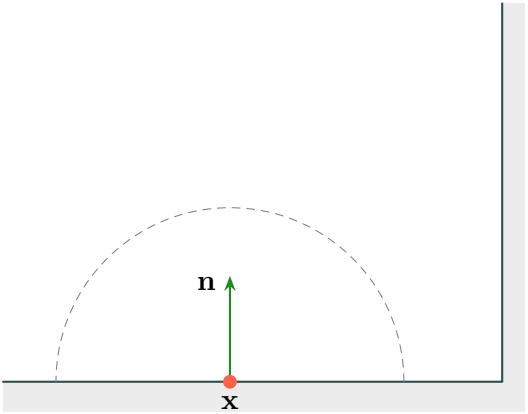
right on average; error $\propto 1/\sqrt{N}$

pick more often where f is large: less noise

Monte Carlo on the dome

One random direction, and how to pick it

One random direction stands in for the whole dome

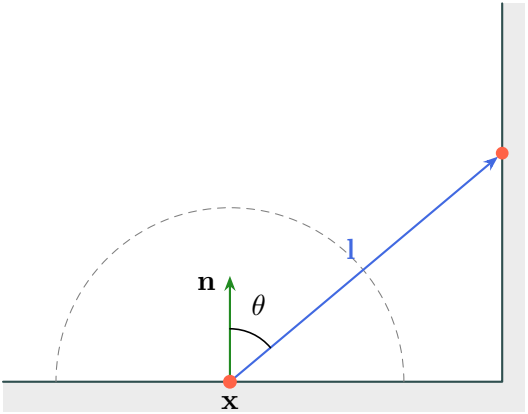


lecture 19:

$$L_o = L_e + \int_{\text{dome}} \rho\, L_i(\mathbf{l}) \cos\theta \, d\sigma$$

ρ: the BRDF; $L_i(\mathbf{l})$: the light arriving from $\mathbf{l}$; θ: from $\mathbf{n}$ to $\mathbf{l}$

One random direction stands in for the whole dome



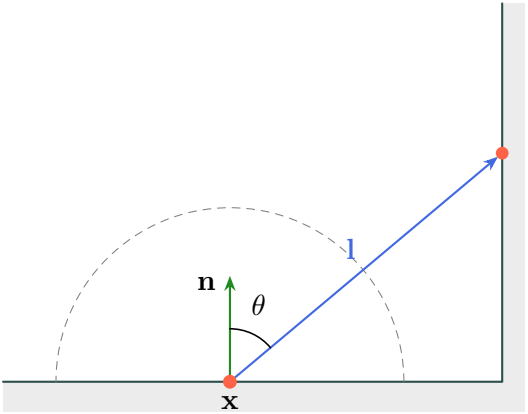
lecture 19:

$$L_o = L_e + \int_{\text{dome}} \rho\, L_i(\mathbf{l}) \cos\theta \, d\sigma$$

- Pick one direction $\mathbf{l}$ at random.

ρ: the BRDF; $L_i(\mathbf{l})$: the light arriving from $\mathbf{l}$; θ: from $\mathbf{n}$ to $\mathbf{l}$

One random direction stands in for the whole dome



lecture 19:

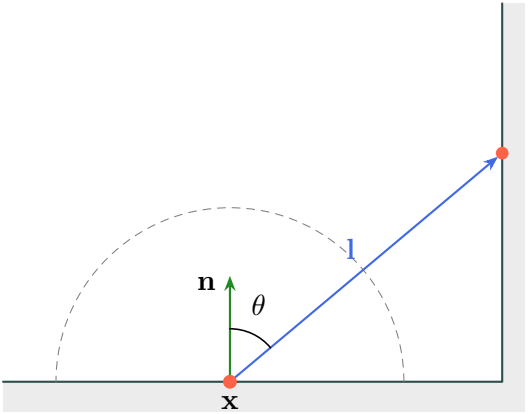
$$L_o = L_e + \int_{\text{dome}} \rho\, L_i(\mathbf{l}) \cos\theta \, d\sigma$$

$$L_o \approx L_e + \frac{\rho\, L_i(\mathbf{l}) \cos\theta}{p(\mathbf{l})}$$

- Pick one direction $\mathbf{l}$ at random.
- Divide by $p(\mathbf{l})$, as in part one.

ρ: the BRDF; $L_i(\mathbf{l})$: the light arriving from $\mathbf{l}$; θ: from $\mathbf{n}$ to $\mathbf{l}$

One random direction stands in for the whole dome



lecture 19:

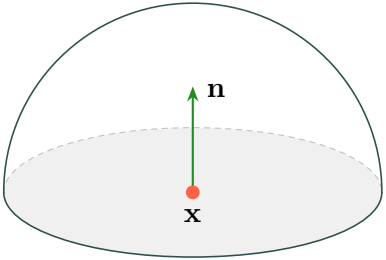
$$L_o = L_e + \int_{\text{dome}} \rho\, L_i(\mathbf{l}) \cos\theta \, d\sigma$$

$$L_o \approx L_e + \frac{\rho\, L_i(\mathbf{l}) \cos\theta}{p(\mathbf{l})}$$

- Pick one direction $\mathbf{l}$ at random.
- Divide by $p(\mathbf{l})$, as in part one.
- One sample: right on average, noisy.

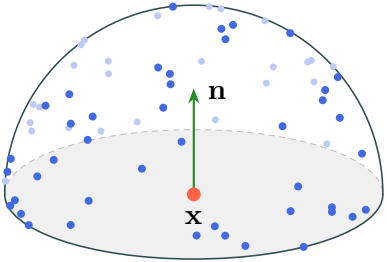
ρ: the BRDF; $L_i(\mathbf{l})$: the light arriving from $\mathbf{l}$; θ: from $\mathbf{n}$ to $\mathbf{l}$

Even picks over the dome: $p = 1/(2\pi)$

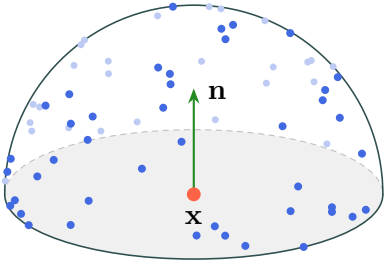


Even picks over the dome: $p = 1/(2\pi)$

- Every direction is equally likely.

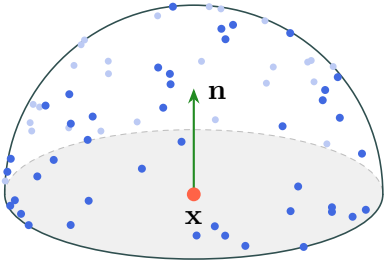


Even picks over the dome: $p = 1/(2\pi)$



- Every direction is equally likely.
- The dome is 2π in all, so $p = \dfrac{1}{2\pi}$.

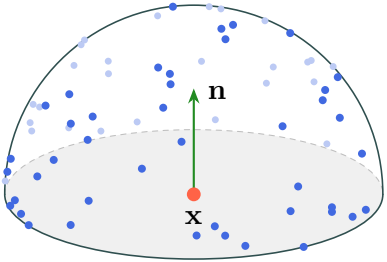
Even picks over the dome: $p = 1/(2\pi)$



- Every direction is equally likely.
- The dome is 2π in all, so $p = \dfrac{1}{2\pi}$.
- For chalk, $\rho = R/\pi$:

$$\frac{\rho\, L_i \cos\theta}{p} = \frac{R}{\pi}\, L_i \cos\theta \cdot 2\pi$$

Even picks over the dome: $p = 1/(2\pi)$

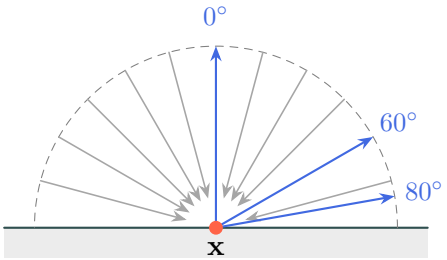


- Every direction is equally likely.
- The dome is 2π in all, so $p = \dfrac{1}{2\pi}$.
- For chalk, $\rho = R/\pi$:

$$\frac{\rho\, L_i \cos\theta}{p} = \frac{R}{\pi}\, L_i \cos\theta \cdot 2\pi$$

$$= 2R\, L_i \cos\theta \qquad \text{one direction's estimate}$$

Worked: an even sky, three even picks

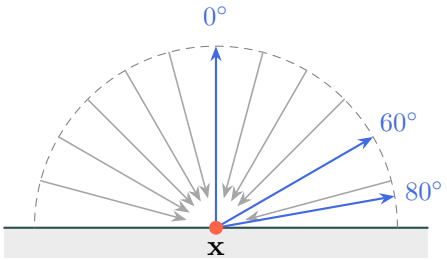


chalk $R = 0.5$; sky $L = 1$ everywhere

θ	$\cos\theta$	$2R\,L_i\cos\theta$
$0°$	1.0	
$60°$	0.5	
$80°$	0.17	

- Try first: each pick's estimate of L_o.

Worked: an even sky, three even picks

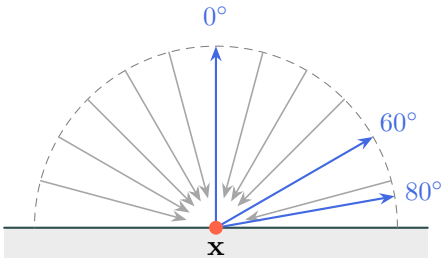


chalk $R = 0.5$; sky $L = 1$ everywhere

θ	$\cos\theta$	$2R\,L_i\cos\theta$
$0°$	1.0	1.0
$60°$	0.5	0.5
$80°$	0.17	0.17

- Try first: each pick's estimate of L_o.

Worked: an even sky, three even picks

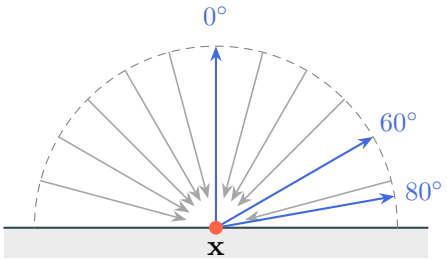


chalk $R = 0.5$; sky $L = 1$ everywhere

θ	$\cos\theta$	$2R\,L_i\cos\theta$
$0°$	1.0	1.0
$60°$	0.5	0.5
$80°$	0.17	0.17

- Try first: each pick's estimate of L_o.
- Exact (lecture 19): $L_o = R\,L = 0.5$.

Worked: an even sky, three even picks

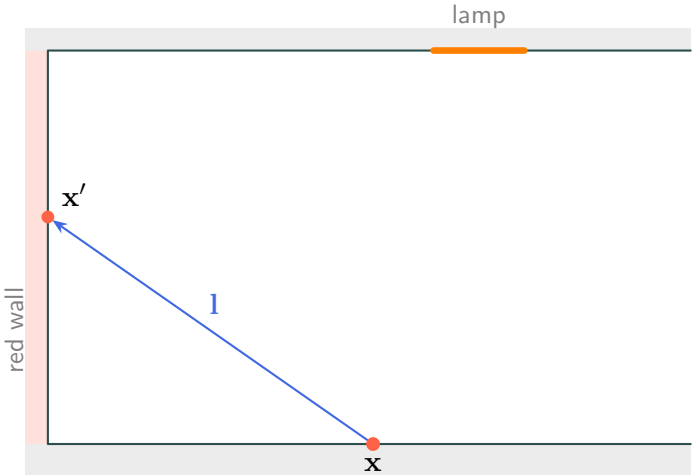


chalk $R = 0.5$; sky $L = 1$ everywhere

θ	$\cos\theta$	$2R\,L_i\cos\theta$
$0°$	1.0	1.0
$60°$	0.5	0.5
$80°$	0.17	0.17

- Try first: each pick's estimate of L_o.
- Exact (lecture 19): $L_o = R\,L = 0.5$.
- Right on average, but each pick is off.

The light from $\mathbf{l}$ is found by `trace`

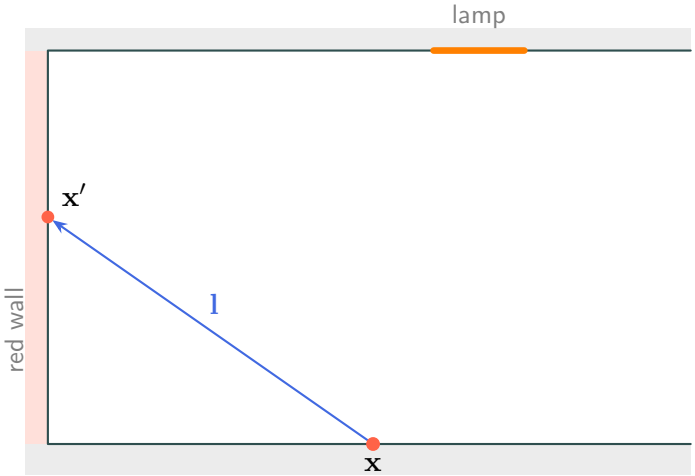


$$L_i(\mathbf{x}, \mathbf{l}) = L_o(\mathbf{x}', -\mathbf{l})$$

lecture 19

- The light from $\mathbf{l}$ left the surface that $\mathbf{l}$ meets.

The light from $\mathbf{l}$ is found by `trace`



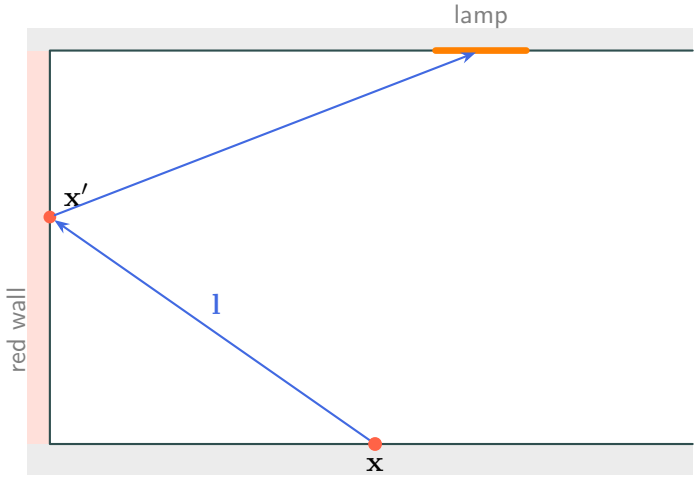
$$L_i(\mathbf{x}, \mathbf{l}) = L_o(\mathbf{x}', -\mathbf{l})$$

lecture 19

$$= \texttt{trace(Ray(x, l))}$$

- The light from $\mathbf{l}$ left the surface that $\mathbf{l}$ meets.
- A ray along $\mathbf{l}$ finds that surface.

The light from $\mathbf{l}$ is found by `trace`



$$L_i(\mathbf{x}, \mathbf{l}) = L_o(\mathbf{x}', -\mathbf{l})$$

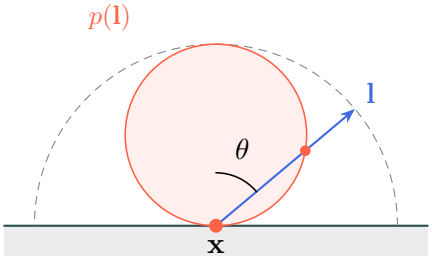
lecture 19

$$= \texttt{trace(Ray(x, l))}$$

- The light from $\mathbf{l}$ left the surface that $\mathbf{l}$ meets.
- A ray along $\mathbf{l}$ finds that surface.
- There, one random ray again.

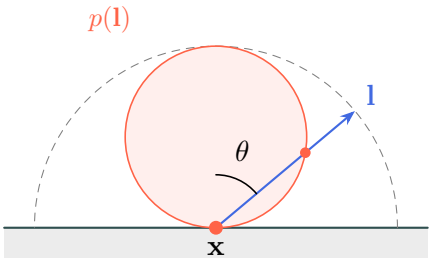
Pick by the cosine: $p = \cos\theta/\pi$

- Pick near $\mathbf{n}$ more often: $p(\mathbf{l}) = \dfrac{\cos\theta}{\pi}$.



the density, as a length in each direction

Pick by the cosine: $p = \cos\theta/\pi$

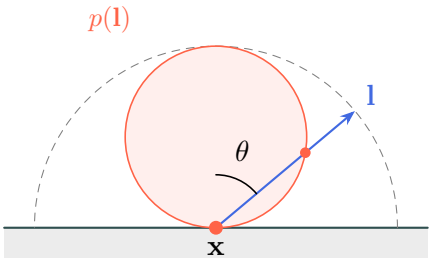


the density, as a length in each direction

- Pick near $\mathbf{n}$ more often: $p(\mathbf{l}) = \dfrac{\cos\theta}{\pi}$.

  it adds up to 1: lecture 19's disc has area π

Pick by the cosine: $p = \cos\theta/\pi$



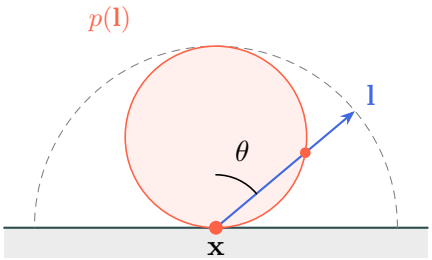
the density, as a length in each direction

- Pick near $\mathbf{n}$ more often: $p(\mathbf{l}) = \dfrac{\cos\theta}{\pi}$.

  it adds up to 1: lecture 19's disc has area π

$$\frac{\rho\, L_i \cos\theta}{p} = \frac{R}{\pi}\, L_i \cos\theta \cdot \frac{\pi}{\cos\theta}$$

Pick by the cosine: $p = \cos\theta/\pi$



the density, as a length in each direction

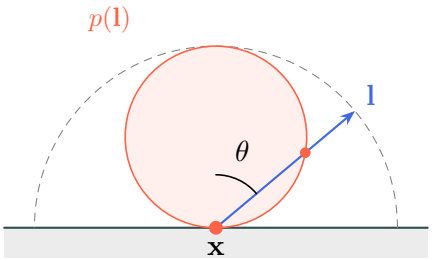
- Pick near $\mathbf{n}$ more often: $p(\mathbf{l}) = \dfrac{\cos\theta}{\pi}$.

  it adds up to 1: lecture 19's disc has area π

$$\frac{\rho\, L_i \cos\theta}{p} = \frac{R}{\pi}\, L_i \cos\theta \cdot \frac{\pi}{\cos\theta}$$

$$= R\, L_i$$

Pick by the cosine: $p = \cos\theta/\pi$



the density, as a length in each direction

- Pick near $\mathbf{n}$ more often: $p(\mathbf{l}) = \dfrac{\cos\theta}{\pi}$.

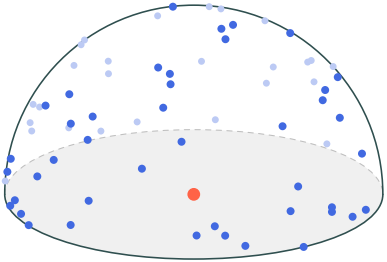
  it adds up to 1: lecture 19's disc has area π

$$\frac{\rho\, L_i \cos\theta}{p} = \frac{R}{\pi}\, L_i \cos\theta \cdot \frac{\pi}{\cos\theta}$$

$$= R\, L_i$$

- The cosine and the π cancel.
- One pick's estimate: R times L_i.

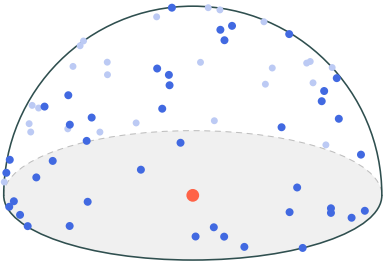
Two ways to pick, side by side



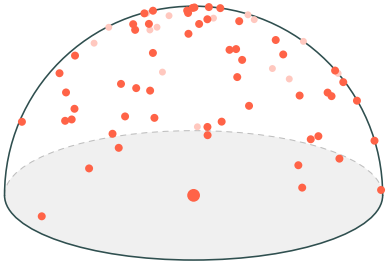
even: $p = 1/(2\pi)$

- Even: as many picks near the rim as on top.

Two ways to pick, side by side



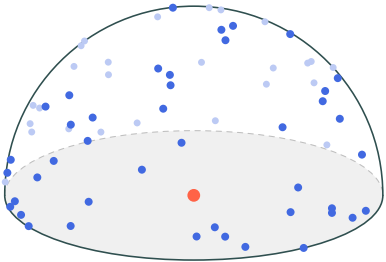
even: $p = 1/(2\pi)$



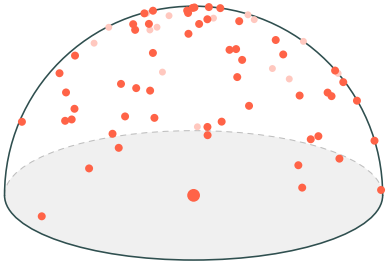
by the cosine: $p = \cos\theta/\pi$

- Even: as many picks near the rim as on top.
- By the cosine: crowded near $\mathbf{n}$, few near the rim.

Two ways to pick, side by side



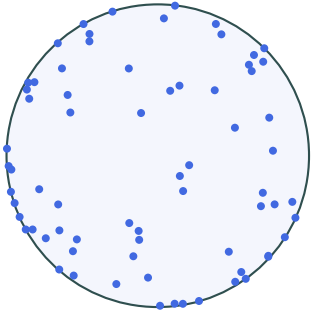
even: $p = 1/(2\pi)$



by the cosine: $p = \cos\theta/\pi$

- Even: as many picks near the rim as on top.
- By the cosine: crowded near $\mathbf{n}$, few near the rim.
- Picks near the rim count little, so few are wasted.

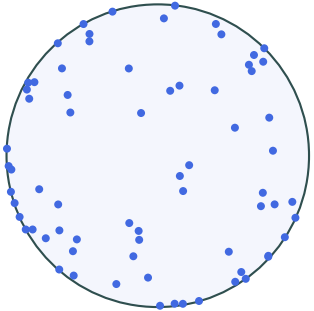
Seen from above: cosine picks fill lecture 19's disc evenly



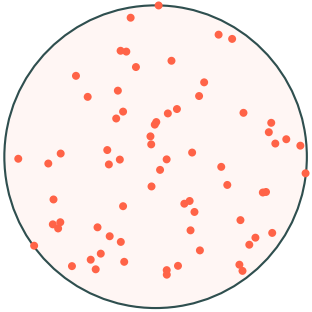
even picks

- Seen from above, even picks crowd the rim.

Seen from above: cosine picks fill lecture 19's disc evenly



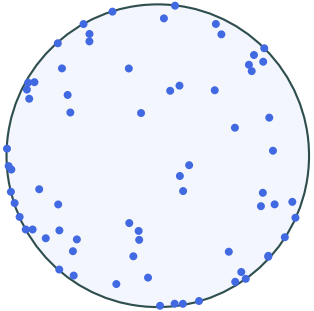
even picks



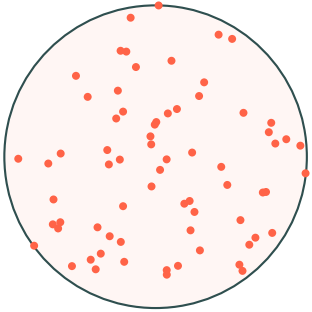
cosine picks

- Seen from above, even picks crowd the rim.
- Cosine picks are even over the disc.

Seen from above: cosine picks fill lecture 19's disc evenly



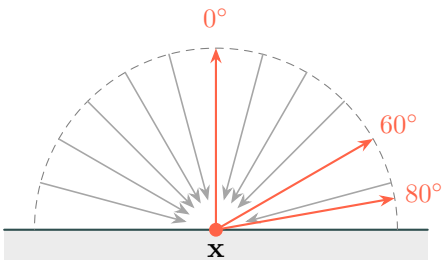
even picks



cosine picks

- Seen from above, even picks crowd the rim.
- Cosine picks are even over the disc.
- The disc's area is π: hence $p = \cos\theta/\pi$.

Worked: the even sky again, with cosine picks

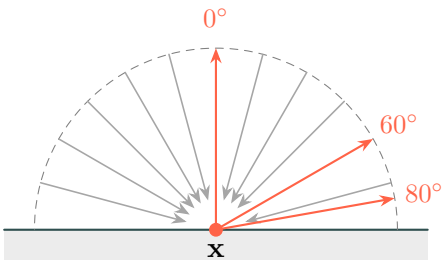


chalk $R = 0.5$; sky $L = 1$ everywhere

θ	even: $2R\,L_i \cos\theta$	cosine: $R\,L_i$
$0°$	1.0	
$60°$	0.5	
$80°$	0.17	

- Try first: each cosine pick's estimate.

Worked: the even sky again, with cosine picks

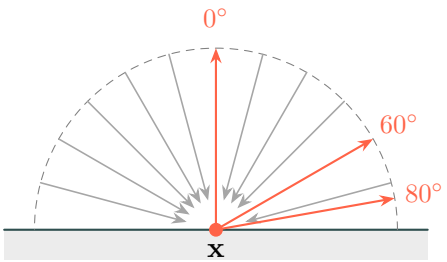


chalk $R = 0.5$; sky $L = 1$ everywhere

θ	even: $2R\,L_i \cos\theta$	cosine: $R\,L_i$
$0°$	1.0	0.5
$60°$	0.5	0.5
$80°$	0.17	0.5

- Try first: each cosine pick's estimate.
- Every pick gives 0.5, the exact answer.

Worked: the even sky again, with cosine picks

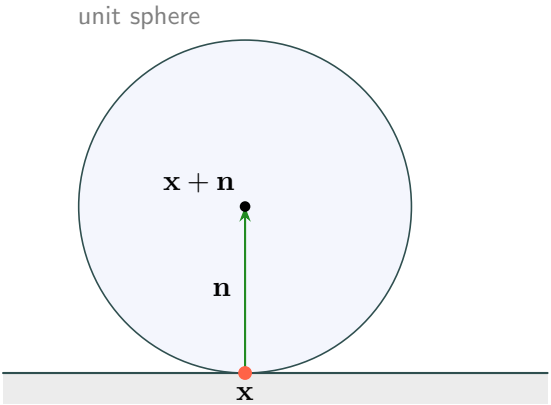


chalk $R = 0.5$; sky $L = 1$ everywhere

θ	even: $2R\,L_i \cos\theta$	cosine: $R\,L_i$
$0°$	1.0	0.5
$60°$	0.5	0.5
$80°$	0.17	0.5

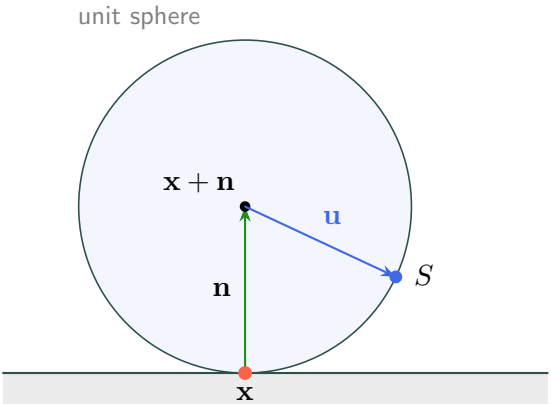
- Try first: each cosine pick's estimate.
- Every pick gives 0.5, the exact answer.
- With uneven light, some noise is left.

RTIOW's trick: the normal plus a random unit vector



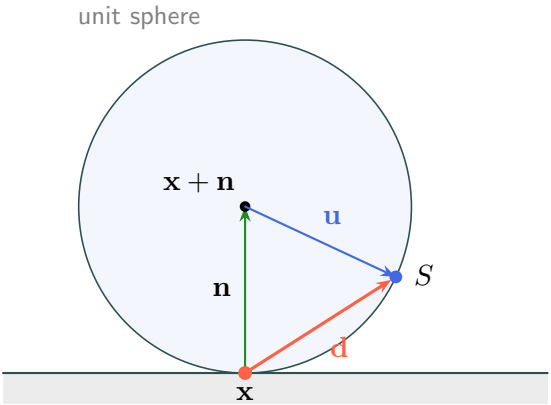
- A unit sphere sits on the surface at $\mathbf{x}$, centre $\mathbf{x} + \mathbf{n}$.

RTIOW's trick: the normal plus a random unit vector



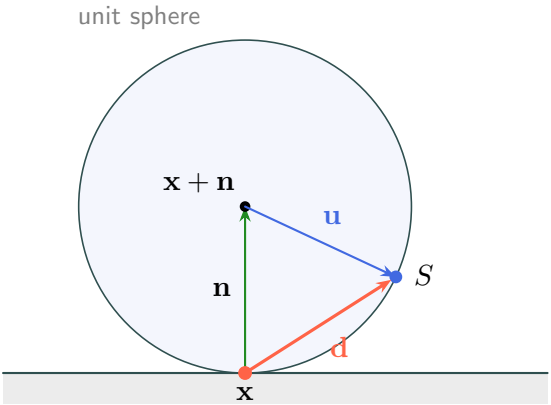
- A unit sphere sits on the surface at $\mathbf{x}$, centre $\mathbf{x}+\mathbf{n}$.
- Pick a random point S on it: $S = \mathbf{x}+\mathbf{n}+\mathbf{u}$, with $\mathbf{u}$ a random unit vector.

RTIOW's trick: the normal plus a random unit vector



- A unit sphere sits on the surface at $\mathbf{x}$, centre $\mathbf{x}+\mathbf{n}$.
- Pick a random point S on it: $S = \mathbf{x}+\mathbf{n}+\mathbf{u}$, with $\mathbf{u}$ a random unit vector.
- Go from $\mathbf{x}$ to S: $\mathbf{d} = \mathbf{n}+\mathbf{u}$.

RTIOW's trick: the normal plus a random unit vector

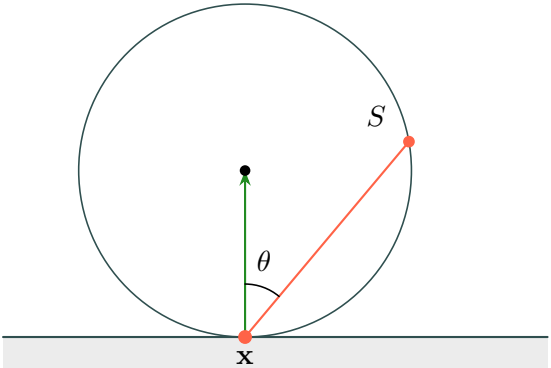


- A unit sphere sits on the surface at $\mathbf{x}$, centre $\mathbf{x}+\mathbf{n}$.
- Pick a random point S on it: $S = \mathbf{x}+\mathbf{n}+\mathbf{u}$, with $\mathbf{u}$ a random unit vector.
- Go from $\mathbf{x}$ to S: $\mathbf{d} = \mathbf{n}+\mathbf{u}$.

$\mathbf{d}$ is picked with density $\cos\theta/\pi$.

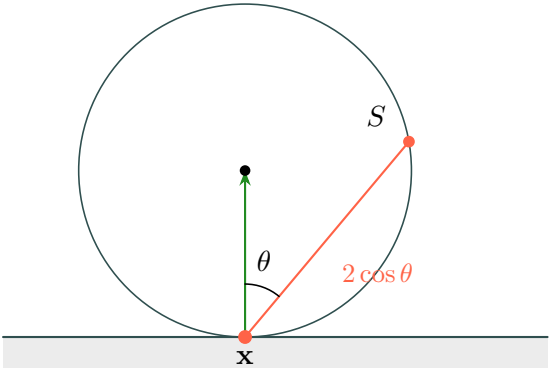
RTIOW: "True Lambertian Reflection"

Why the trick gives $\cos\theta/\pi$



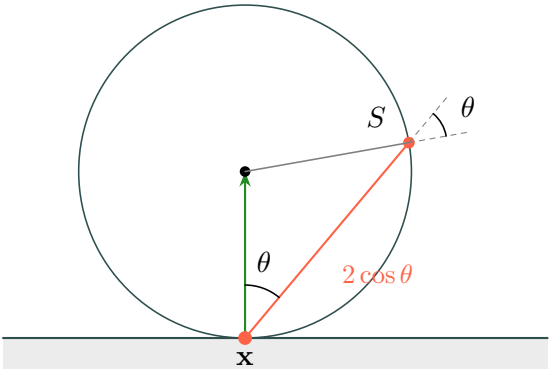
1. The ray meets the sphere $2\cos\theta$ away.

Why the trick gives $\cos\theta/\pi$



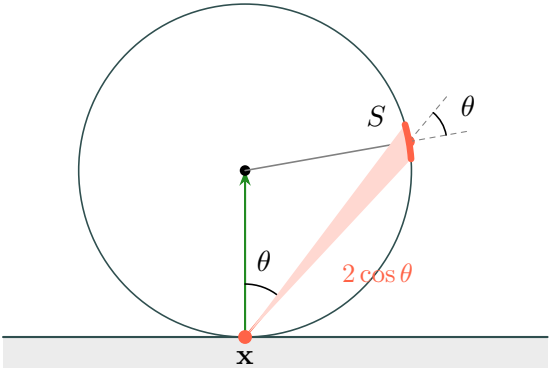
1. The ray meets the sphere $2\cos\theta$ away.

Why the trick gives $\cos\theta/\pi$



1. The ray meets the sphere $2\cos\theta$ away.
2. It meets the surface tilted by θ.

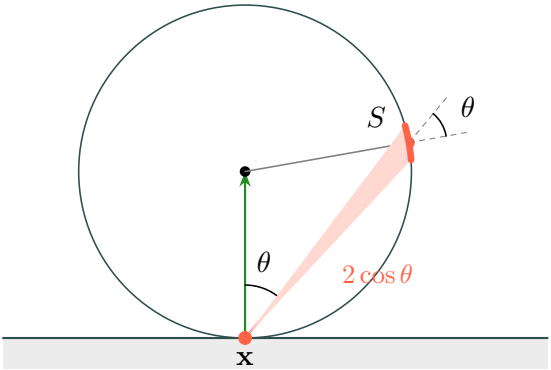
Why the trick gives $\cos\theta/\pi$



1. The ray meets the sphere $2\cos\theta$ away.
2. It meets the surface tilted by θ.
3. A patch $d\sigma$ of directions covers

$$dA = \frac{(2\cos\theta)^2\, d\sigma}{\cos\theta} = 4\cos\theta\, d\sigma$$

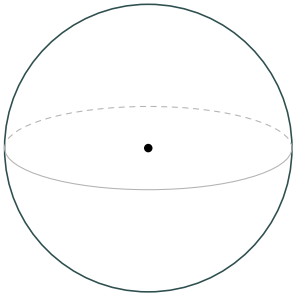
Why the trick gives $\cos\theta/\pi$



1. The ray meets the sphere $2\cos\theta$ away.
2. It meets the surface tilted by θ.
3. A patch $d\sigma$ of directions covers
$$dA = \frac{(2\cos\theta)^2\, d\sigma}{\cos\theta} = 4\cos\theta\, d\sigma$$
4. S is even over the area 4π:
$$p = \frac{4\cos\theta}{4\pi} = \frac{\cos\theta}{\pi}$$

A random unit vector, in Python

```
def random_unit_vector():
    a = random.uniform(0, 2 * math.pi)
```

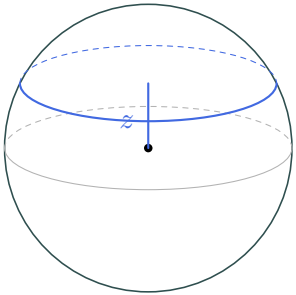


the unit sphere

A random unit vector, in Python

```
def random_unit_vector():
    a = random.uniform(0, 2 * math.pi)
    z = random.uniform(-1, 1)
```

- The height z is even between -1 and 1.

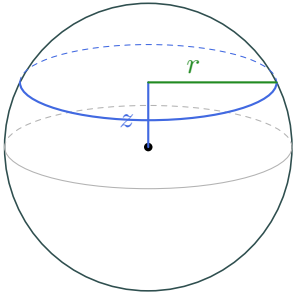


the unit sphere

A random unit vector, in Python

```
def random_unit_vector():
    a = random.uniform(0, 2 * math.pi)
    z = random.uniform(-1, 1)
    r = math.sqrt(1 - z**2)
```

- The height z is even between -1 and 1.
- At that height the sphere is a circle of radius r.

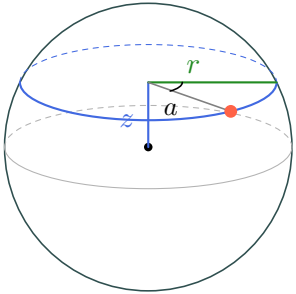


the unit sphere

A random unit vector, in Python

```
def random_unit_vector():
    a = random.uniform(0, 2 * math.pi)
    z = random.uniform(-1, 1)
    r = math.sqrt(1 - z**2)
    return Vec3(r * math.cos(a), r * math.sin(a), z)
```

- The height z is even between -1 and 1.
- At that height the sphere is a circle of radius r.
- The angle a is even around that circle.

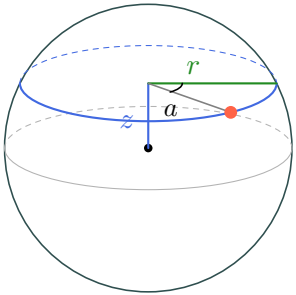


the unit sphere

A random unit vector, in Python

```
def random_unit_vector():
    a = random.uniform(0, 2 * math.pi)
    z = random.uniform(-1, 1)
    r = math.sqrt(1 - z**2)
    return Vec3(r * math.cos(a), r * math.sin(a), z)
```

- The height z is even between -1 and 1.
- At that height the sphere is a circle of radius r.
- The angle a is even around that circle.
- Even heights give even area (Archimedes).



the unit sphere

Recap: Monte Carlo on the dome

one random direction: $L_o \approx L_e + \dfrac{\rho\, L_i \cos\theta}{p}$

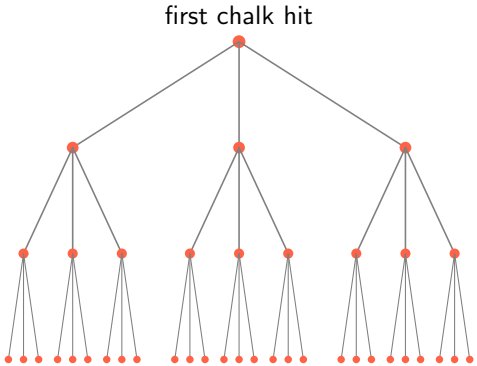
chalk, cosine picks: $L_o \approx L_e + R\, L_i$

$L_i = \texttt{trace}$ along $\mathbf{d} = \mathbf{n} + \mathbf{u}$

The path tracer

Chalk bounces; everything else stays

One ray per bounce: a path, not a tree

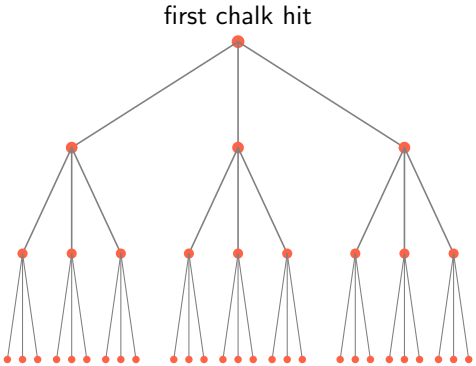


$3 + 9 + 27 = 39$ rays

three random rays at every chalk hit

- Several rays per bounce grow a tree.

One ray per bounce: a path, not a tree



$3 + 9 + 27 = 39$ rays

three random rays at every chalk hit

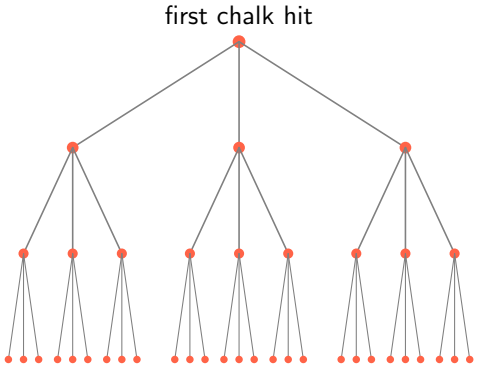


3 rays

one random ray at every hit

- Several rays per bounce grow a tree.
- One ray per bounce makes a path.

One ray per bounce: a path, not a tree



$3 + 9 + 27 = 39$ rays

three random rays at every chalk hit

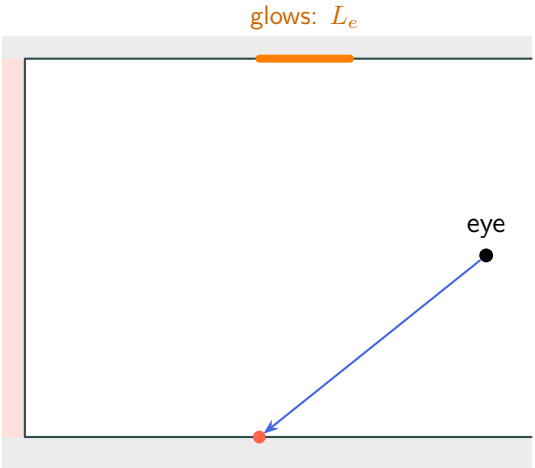


3 rays

one random ray at every hit

- Several rays per bounce grow a tree.
- One ray per bounce makes a path.
- Spend the saved rays on more paths per pixel.

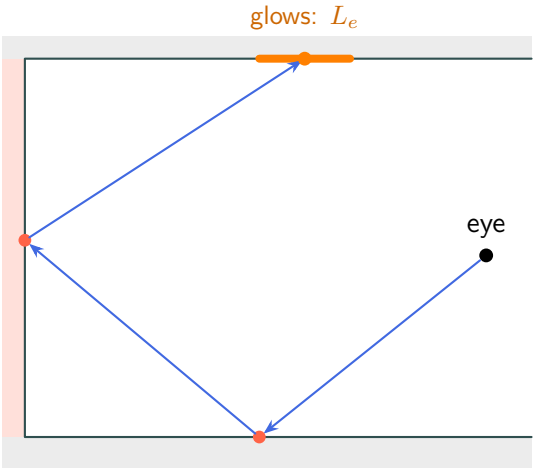
A glowing surface ends the path



chalk: one random ray on

- A lamp is now a surface that glows. It emits L_e.

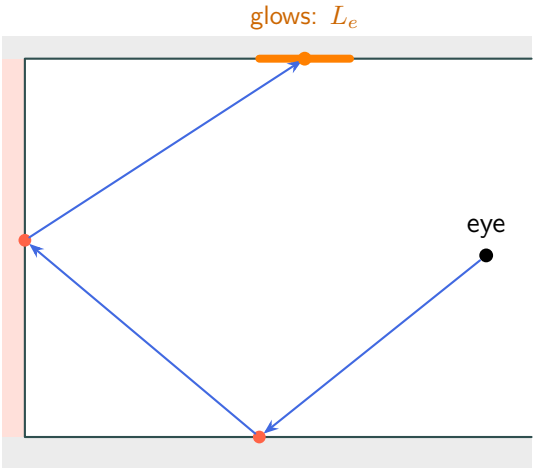
A glowing surface ends the path



chalk: one random ray on

- A lamp is now a surface that glows. It emits L_e.
- A ray that hits it brings back L_e. The path ends there.

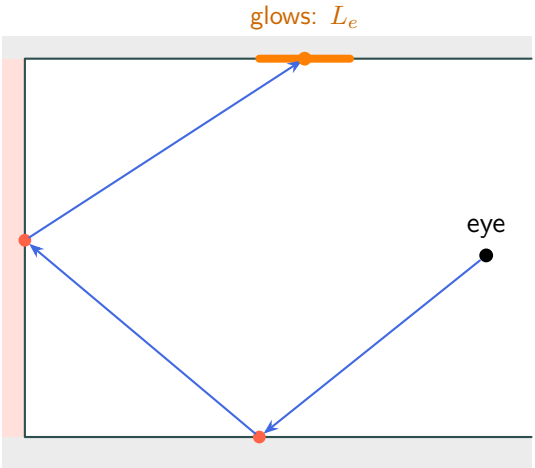
A glowing surface ends the path



chalk: one random ray on

- A lamp is now a surface that glows. It emits L_e.
- A ray that hits it brings back L_e. The path ends there.
- A ray that misses everything brings back the sky. It ends too.

A glowing surface ends the path

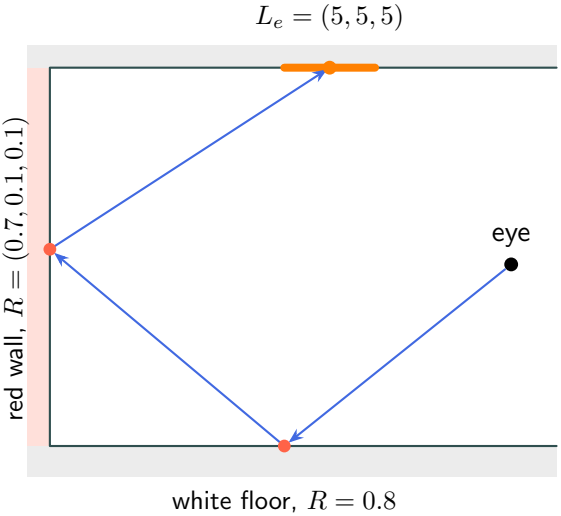


chalk: one random ray on

- A lamp is now a surface that glows. It emits L_e.
- A ray that hits it brings back L_e. The path ends there.
- A ray that misses everything brings back the sky. It ends too.

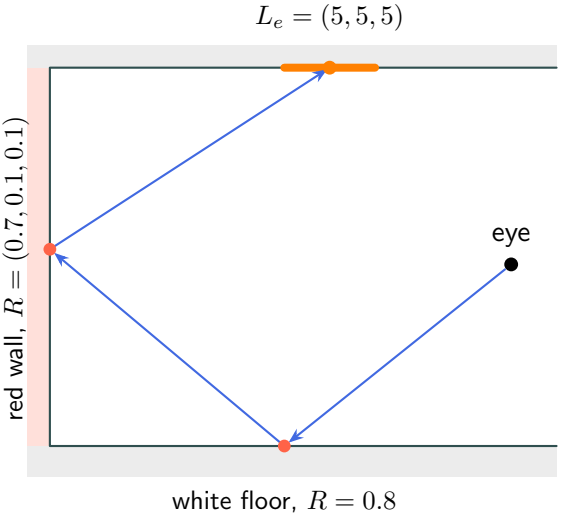
in code: one more kind of material, `DiffuseLight(emission)`

Worked: what one path brings back



- Try first: the colour this path brings to the eye.

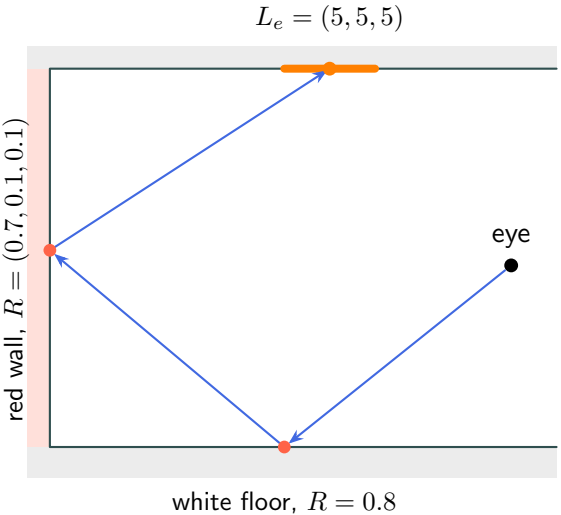
Worked: what one path brings back



- Try first: the colour this path brings to the eye.

  lamp: $(5, 5, 5)$

Worked: what one path brings back

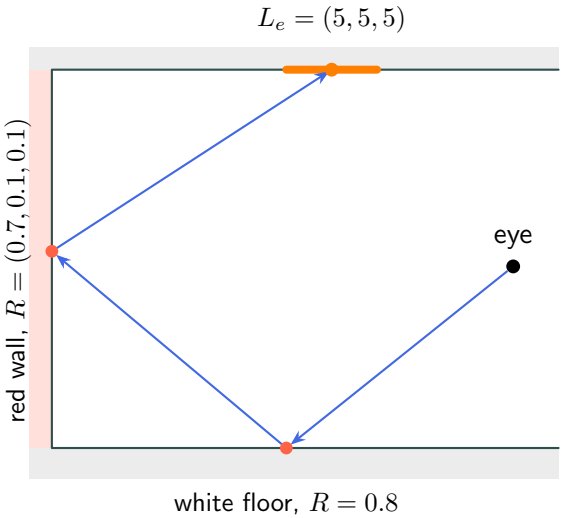


- Try first: the colour this path brings to the eye.

  lamp: $(5, 5, 5)$

  wall: $R \times (5, 5, 5) = (3.5, 0.5, 0.5)$

Worked: what one path brings back



- Try first: the colour this path brings to the eye.

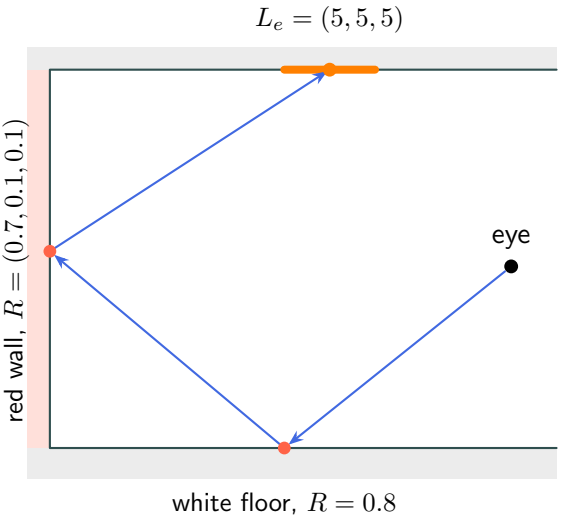
  lamp: $(5, 5, 5)$

  wall: $R \times (5, 5, 5) = (3.5, 0.5, 0.5)$

  floor: $0.8 \times (3.5, 0.5, 0.5) = (2.8, 0.4, 0.4)$

- Work back from the lamp: times each R.

Worked: what one path brings back



- Try first: the colour this path brings to the eye.

  lamp: $(5, 5, 5)$

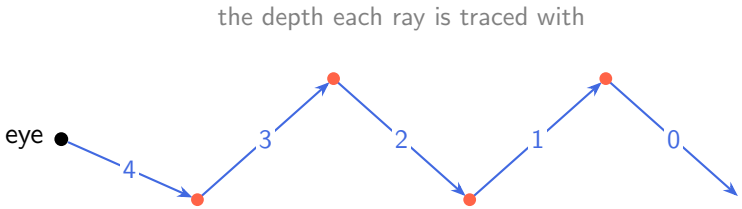
  wall: $R \times (5, 5, 5) = (3.5, 0.5, 0.5)$

  floor: $0.8 \times (3.5, 0.5, 0.5) = (2.8, 0.4, 0.4)$

- Work back from the lamp: times each R.
- The floor looks red here: colour bleeding.

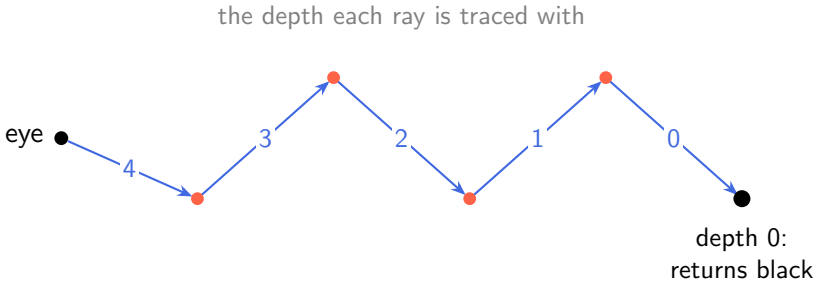
  one path of many; the pixel averages them

Stopping, way one: a fixed depth



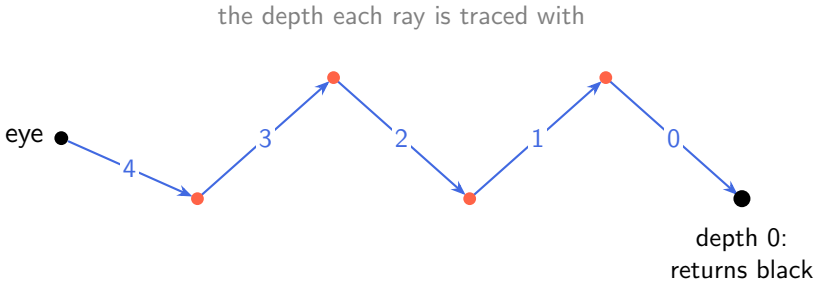
- Each call passes on one less depth.

Stopping, way one: a fixed depth



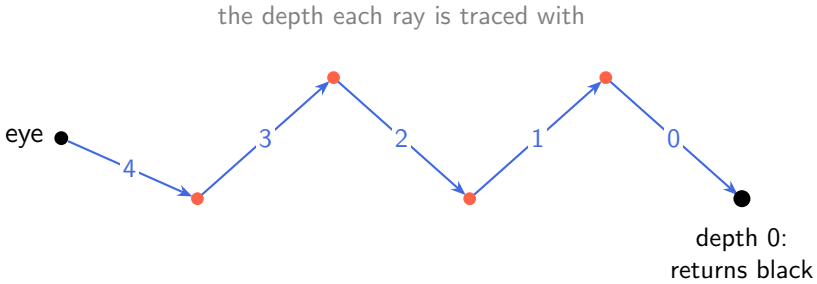
- Each call passes on one less depth.
- At depth 0, `trace` returns black.

Stopping, way one: a fixed depth



- Each call passes on one less depth.
- At depth 0, `trace` returns black.
- The light of longer paths is lost.

Stopping, way one: a fixed depth

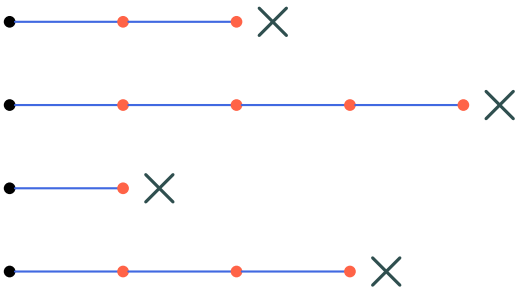


- Each call passes on one less depth.
- At depth 0, `trace` returns black.
- The light of longer paths is lost.

after k hits a path carries R^k: $0.5^5 \approx 0.03$, but $0.8^5 \approx 0.33$

a dark room loses little; a white room loses more

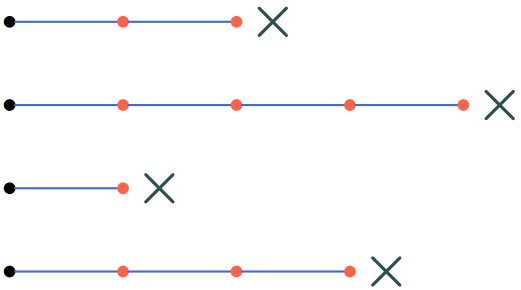
Stopping, way two: Russian roulette



each path stops at a random hit

- At each hit, go on with chance q.

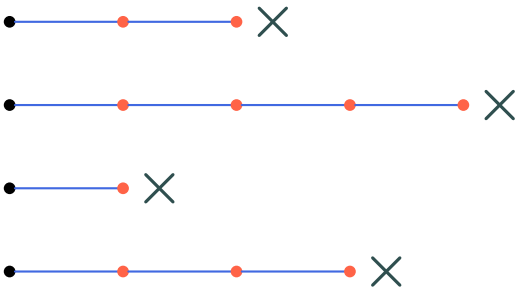
Stopping, way two: Russian roulette



each path stops at a random hit

- At each hit, go on with chance q.
- If it stops, return black.
- If it goes on, divide by q.

Stopping, way two: Russian roulette

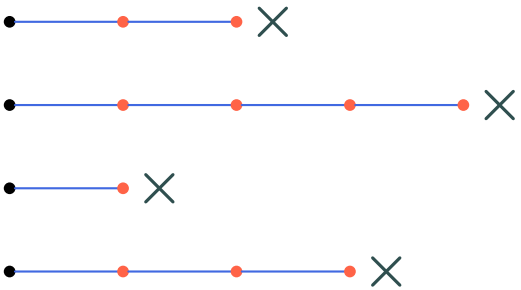


each path stops at a random hit

- At each hit, go on with chance q.
- If it stops, return black.
- If it goes on, divide by q.

average: $q \cdot \dfrac{X}{q} + (1-q) \cdot 0 = X$

Stopping, way two: Russian roulette



each path stops at a random hit

- At each hit, go on with chance q.
- If it stops, return black.
- If it goes on, divide by q.

average: $q \cdot \dfrac{X}{q} + (1-q) \cdot 0 = X$

- Right on average, with no fixed limit.

X: what going on would bring back

Worked: the roulette keeps the average

path	roulette	brings back
1	on	
2	stop	
3	on	
4	stop	
average		

- Going on would bring back $X = 1.0$. The chance to go on is $q = 0.5$.
- Try first: each path's value, and the average.

Worked: the roulette keeps the average

path	roulette	brings back
1	on	$1.0/0.5 = 2$
2	stop	0
3	on	2
4	stop	0
average		

- Going on would bring back $X = 1.0$. The chance to go on is $q = 0.5$.
- Try first: each path's value, and the average.

Worked: the roulette keeps the average

path	roulette	brings back
1	on	$1.0/0.5 = 2$
2	stop	0
3	on	2
4	stop	0
average		$(2 + 0 + 2 + 0)/4 = 1.0$

- Going on would bring back $X = 1.0$. The chance to go on is $q = 0.5$.
- Try first: each path's value, and the average.
- The average is X: nothing is lost.

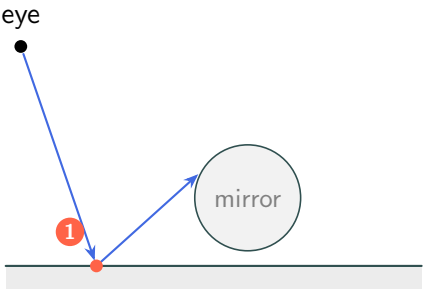
Worked: the roulette keeps the average

path	roulette	brings back
1	on	$1.0/0.5 = 2$
2	stop	0
3	on	2
4	stop	0
average		$(2+0+2+0)/4 = 1.0$

- Going on would bring back $X = 1.0$. The chance to go on is $q = 0.5$.
- Try first: each path's value, and the average.
- The average is X: nothing is lost.
- Each path is now 0 or 2, not 1: more noise.

`trace`: chalk now bounces; mirrors and glass stay

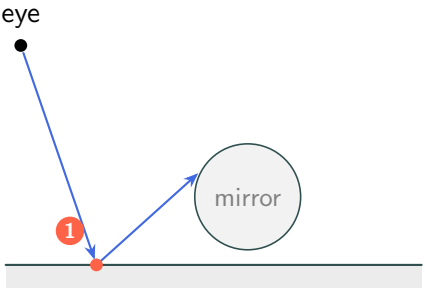
```
def trace(ray, depth):
    rec = scene.hit(ray)
    if rec.t == math.inf:
        return the sky's colour
    material = rec.surface.material
    if isinstance(material, Lambertian):
        return one random bounce, times R      1

    elif isinstance(material, Reflective):
        return the mirror's colour             3
    elif isinstance(material, Glass):
        return shade_glass(ray, rec, depth)    4
```



Grey: lecture 18's `trace`, unchanged. `BlinnPhong` keeps lecture 17's lamp sum (not shown).

trace: chalk now bounces; mirrors and glass stay

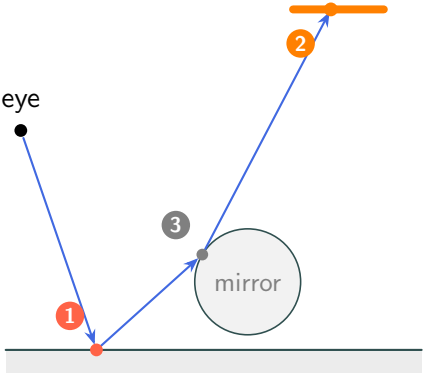
```
def trace(ray, depth):
    rec = scene.hit(ray)
    if rec.t == math.inf:
        return the sky's colour
    material = rec.surface.material
    if isinstance(material, Lambertian):
        return one random bounce, times R      1
    elif isinstance(material, DiffuseLight):
        return its glow, L_e                   2
    elif isinstance(material, Reflective):
        return the mirror's colour             3
    elif isinstance(material, Glass):
        return shade_glass(ray, rec, depth)    4
```



Grey: lecture 18's `trace`, unchanged. `BlinnPhong` keeps lecture 17's lamp sum (not shown).

trace: chalk now bounces; mirrors and glass stay

```
def trace(ray, depth):
    rec = scene.hit(ray)
    if rec.t == math.inf:
        return the sky's colour
    material = rec.surface.material
    if isinstance(material, Lambertian):
        return one random bounce, times R      (1)
    elif isinstance(material, DiffuseLight):
        return its glow, L_e                   (2)
    elif isinstance(material, Reflective):
        return the mirror's colour             (3)
    elif isinstance(material, Glass):
        return shade_glass(ray, rec, depth)    (4)
```

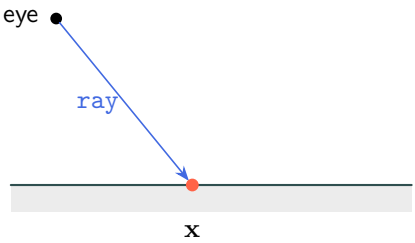


- The mirror and glass branches do not change. Their rays carry the path on.

Grey: lecture 18's `trace`, unchanged. `BlinnPhong` keeps lecture 17's lamp sum (not shown).

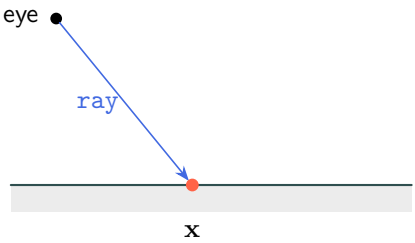
The two new branches, in Python

```
class DiffuseLight:
    def __init__(self, emission):
        self.emission = emission

    if isinstance(material, Lambertian):
```



The two new branches, in Python

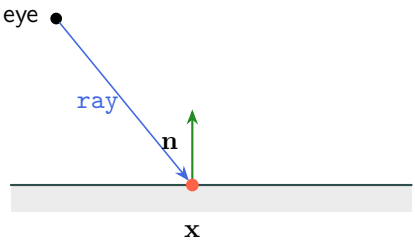
```
class DiffuseLight:
    def __init__(self, emission):
        self.emission = emission

    if isinstance(material, Lambertian):
        if depth == 0:
            return Vec3(0,0,0)
```



- No bounces left: black.

The two new branches, in Python

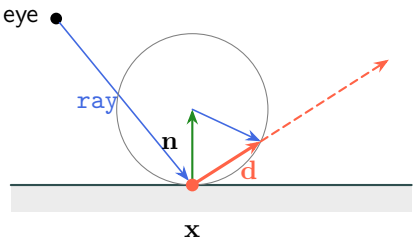
```
class DiffuseLight:
    def __init__(self, emission):
        self.emission = emission

    if isinstance(material, Lambertian):
        if depth == 0:
            return Vec3(0,0,0)
        x = ray.evaluate(rec.t)
```



- No bounces left: black.

The two new branches, in Python

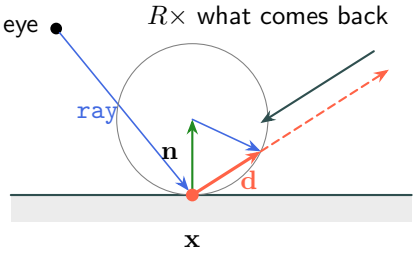
```
class DiffuseLight:
    def __init__(self, emission):
        self.emission = emission

    if isinstance(material, Lambertian):
        if depth == 0:
            return Vec3(0,0,0)
        x = ray.evaluate(rec.t)
        d = rec.normal + random_unit_vector()
```



- No bounces left: black.

The two new branches, in Python

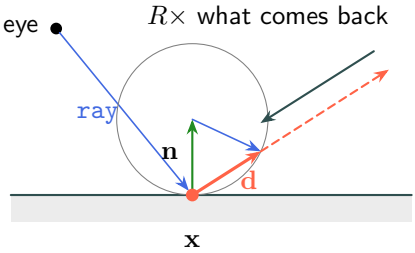
```
class DiffuseLight:
    def __init__(self, emission):
        self.emission = emission

    if isinstance(material, Lambertian):
        if depth == 0:
            return Vec3(0,0,0)
        x = ray.evaluate(rec.t)
        d = rec.normal + random_unit_vector()
        return material.albedo * trace(Ray(x, d), depth - 1)
```



- No bounces left: black.
- One recursive call: the path goes on.

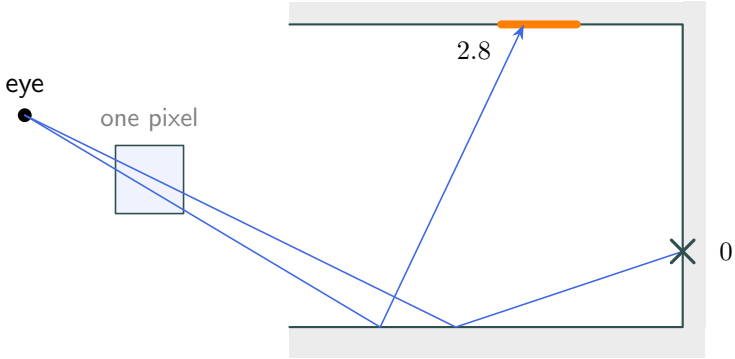
The two new branches, in Python

```
class DiffuseLight:
    def __init__(self, emission):
        self.emission = emission

    if isinstance(material, Lambertian):
        if depth == 0:
            return Vec3(0,0,0)
        x = ray.evaluate(rec.t)
        d = rec.normal + random_unit_vector()
        return material.albedo * trace(Ray(x, d), depth - 1)
    elif isinstance(material, DiffuseLight):
        return material.emission
```



- No bounces left: black.
- One recursive call: the path goes on.
- A lamp returns its glow and makes no new ray.

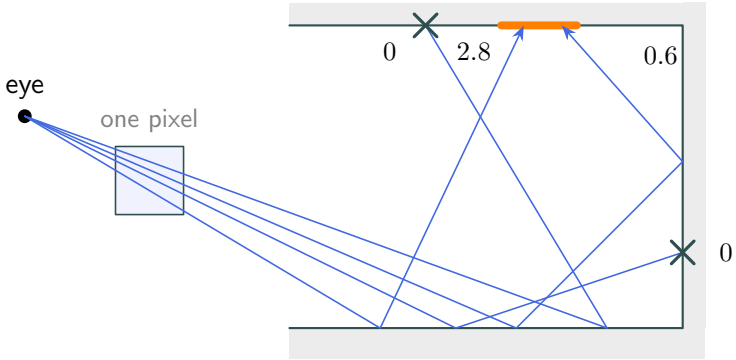
Many paths per pixel, averaged



- Each path gives a different answer.

```
for s in range(SAMPLES_PER_PIXEL):
    a ray through a random spot of the pixel
    color = color + trace(Ray(eye, ray_dir), depth)
img.putpixel((x, y), (color / SAMPLES_PER_PIXEL).to_color())
```

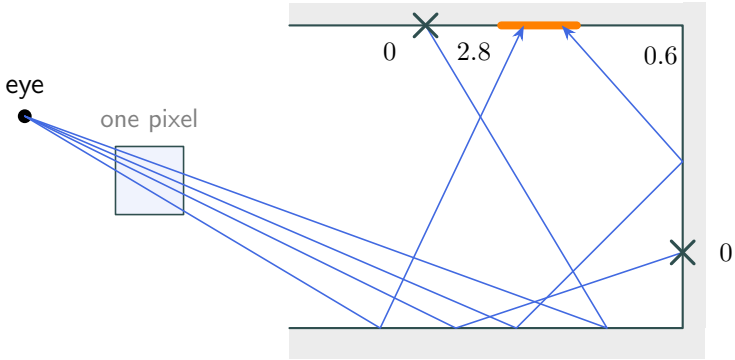
Many paths per pixel, averaged



- Each path gives a different answer.

```
for s in range(SAMPLES_PER_PIXEL):
    a ray through a random spot of the pixel
    color = color + trace(Ray(eye, ray_dir), depth)
img.putpixel((x, y), (color / SAMPLES_PER_PIXEL).to_color())
```

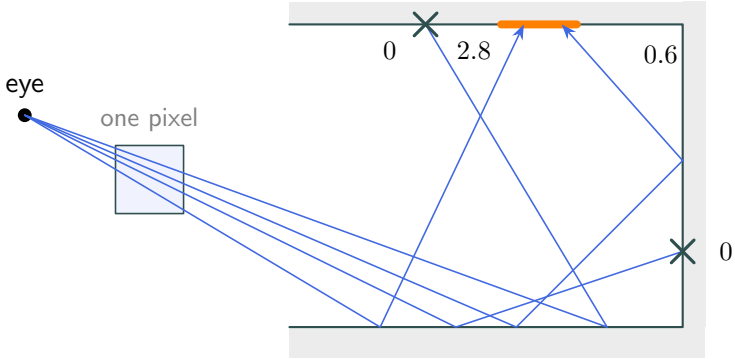
Many paths per pixel, averaged



- Each path gives a different answer.
- The pixel shows their average.

```
for s in range(SAMPLES_PER_PIXEL):
    a ray through a random spot of the pixel
    color = color + trace(Ray(eye, ray_dir), depth)
img.putpixel((x, y), (color / SAMPLES_PER_PIXEL).to_color())
```

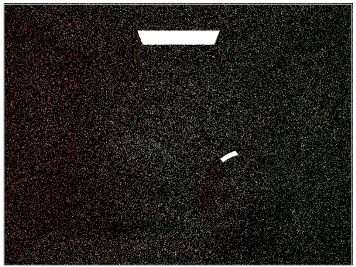
Many paths per pixel, averaged



- Each path gives a different answer.
- The pixel shows their average.
- Lecture 15's loop does it, unchanged.

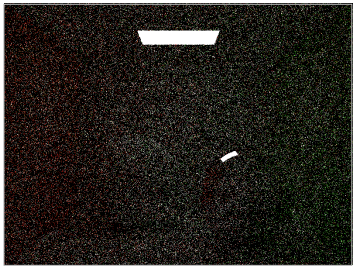
```
for s in range(SAMPLES_PER_PIXEL):
    a ray through a random spot of the pixel
    color = color + trace(Ray(eye, ray_dir), depth)
img.putpixel((x, y), (color / SAMPLES_PER_PIXEL).to_color())
```

Rendered: one room at 1, 16 and 256 paths per pixel

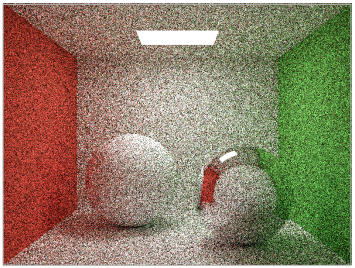


1 path per pixel

Rendered: one room at 1, 16 and 256 paths per pixel

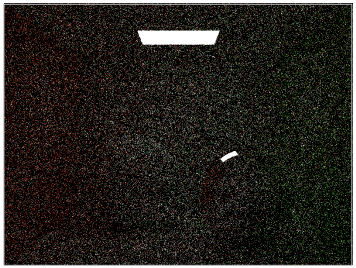


1 path per pixel

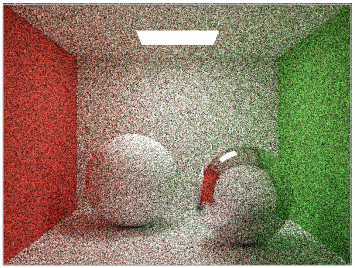


16 paths

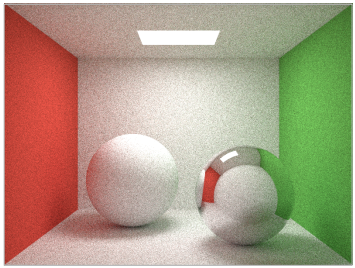
Rendered: one room at 1, 16 and 256 paths per pixel



1 path per pixel

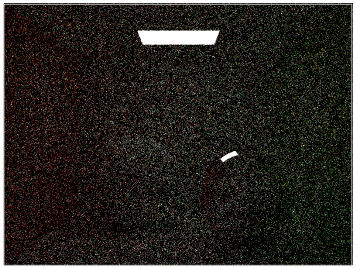


16 paths

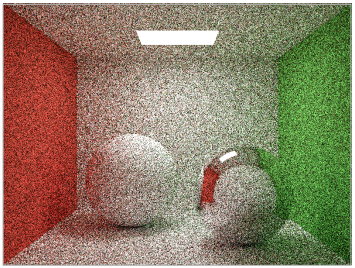


256 paths

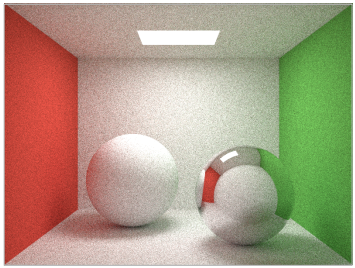
Rendered: one room at 1, 16 and 256 paths per pixel



1 path per pixel



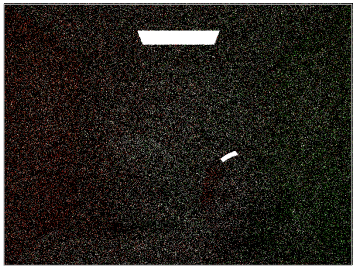
16 paths



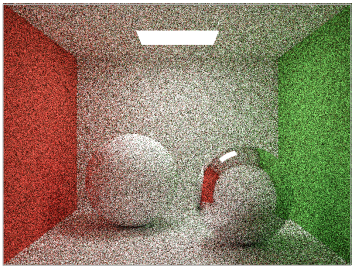
256 paths

- Noise is the price of random paths.

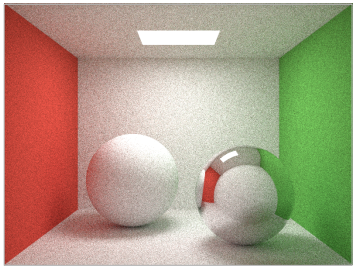
Rendered: one room at 1, 16 and 256 paths per pixel



1 path per pixel



16 paths



256 paths

- Noise is the price of random paths.
- Sixteen times the paths give a quarter of the noise.

Quick check 2: one path, and the roulette

A path: eye, floor $R = 0.8$, green wall $R = (0.1, 0.6, 0.1)$, lamp $L_e = 10$.

1. What colour does the path bring to the eye?
2. The same path with `trace(ray, 1)` from the eye?
3. With roulette at $q = 0.8$, a path that goes on is multiplied by?
4. 4 paths per pixel give noise 0.2. How many give 0.05?

- A path's value is L_e times the albedos on the way.

Quick check 2: one path, and the roulette

A path: eye, floor $R = 0.8$, green wall $R = (0.1, 0.6, 0.1)$, lamp $L_e = 10$.

1. What colour does the path bring to the eye? $(0.8, 4.8, 0.8)$
2. The same path with `trace(ray, 1)` from the eye? black
3. With roulette at $q = 0.8$, a path that goes on is multiplied by? $1/0.8 = 1.25$
4. 4 paths per pixel give noise 0.2. How many give 0.05? 64

- A path's value is L_e times the albedos on the way.

Recap: the path tracer

chalk: `albedo * trace(Ray(x, n + u), depth - 1)`

a lamp ends the path; depth or roulette stops it

many paths per pixel; noise $\propto 1/\sqrt{N}$

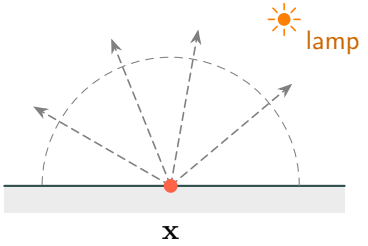
mirrors and glass: unchanged

Aim at the lamp

Next-event estimation, and where we are

Next-event estimation: aim a shadow ray at the lamp

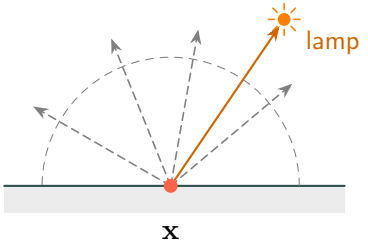
```
if isinstance(material, Lambertian):
    if depth == 0:
        return Vec3(0,0,0)
    x = ray.evaluate(rec.t)
    d = rec.normal + random_unit_vector()
    return material.albedo * trace(Ray(x, d), depth - 1)
```



- A random ray rarely finds a small lamp. It never finds a point lamp.

Next-event estimation: aim a shadow ray at the lamp

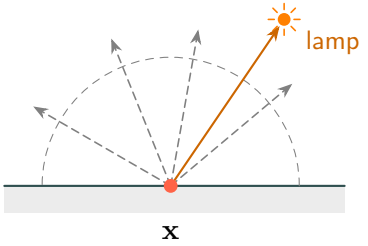
```
if isinstance(material, Lambertian):
    c = Vec3(0,0,0)
    for light in lights:
        c = c + light.illuminate(ray, rec)
    if depth == 0:
        return Vec3(0,0,0)
    x = ray.evaluate(rec.t)
    d = rec.normal + random_unit_vector()
    return material.albedo * trace(Ray(x, d), depth - 1)
```



- A random ray rarely finds a small lamp. It never finds a point lamp.
- So at every chalk hit, also aim lecture 17's shadow ray at each lamp.

Next-event estimation: aim a shadow ray at the lamp

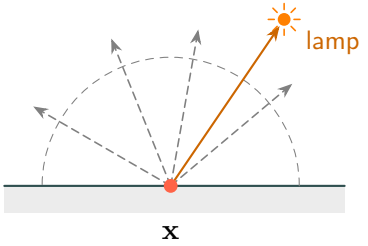
```
if isinstance(material, Lambertian):
    c = Vec3(0,0,0)
    for light in lights:
        c = c + light.illuminate(ray, rec)
    if depth == 0:
        return c
    x = ray.evaluate(rec.t)
    d = rec.normal + random_unit_vector()
    return c + material.albedo * trace(Ray(x, d), depth - 1)
```



- A random ray rarely finds a small lamp. It never finds a point lamp.
- So at every chalk hit, also aim lecture 17's shadow ray at each lamp.
- Whitted's lamp sum is back, as a shortcut.

Next-event estimation: aim a shadow ray at the lamp

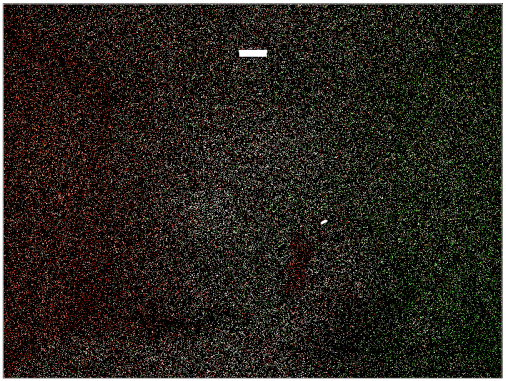
```
if isinstance(material, Lambertian):
    c = Vec3(0,0,0)
    for light in lights:
        c = c + light.illuminate(ray, rec)
    if depth == 0:
        return c
    x = ray.evaluate(rec.t)
    d = rec.normal + random_unit_vector()
    return c + material.albedo * trace(Ray(x, d), depth - 1)
```



- A random ray rarely finds a small lamp. It never finds a point lamp.
- So at every chalk hit, also aim lecture 17's shadow ray at each lamp.
- Whitted's lamp sum is back, as a shortcut.

  With depth 0, only the lamp sum is left: lecture 19's bounce 0, our old tracer.

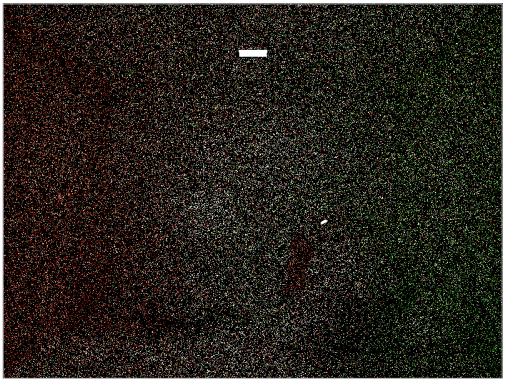
Rendered: a small lamp, without and with the shadow ray



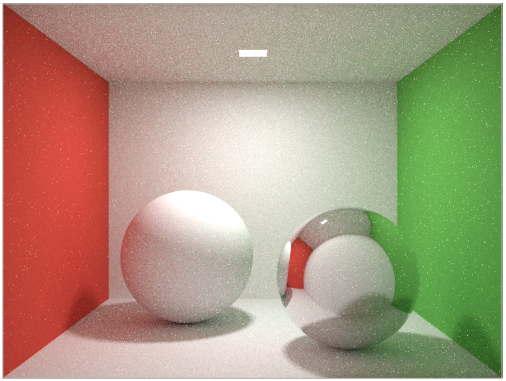
random rays only, 16 paths

- Same room, same lamp, same work.

Rendered: a small lamp, without and with the shadow ray



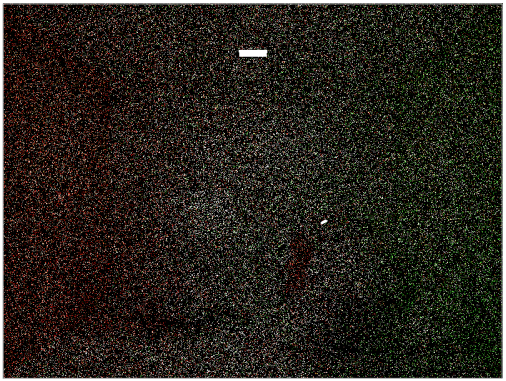
random rays only, 16 paths



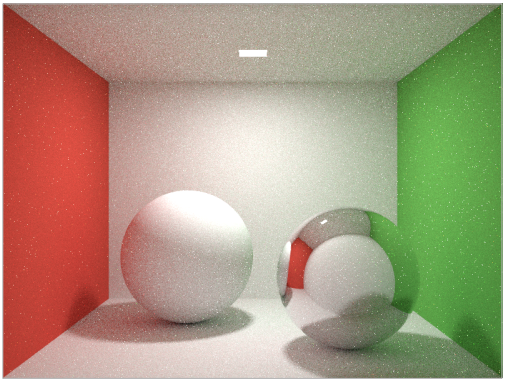
with the shadow ray, 16 paths

- Same room, same lamp, same work.

Rendered: a small lamp, without and with the shadow ray



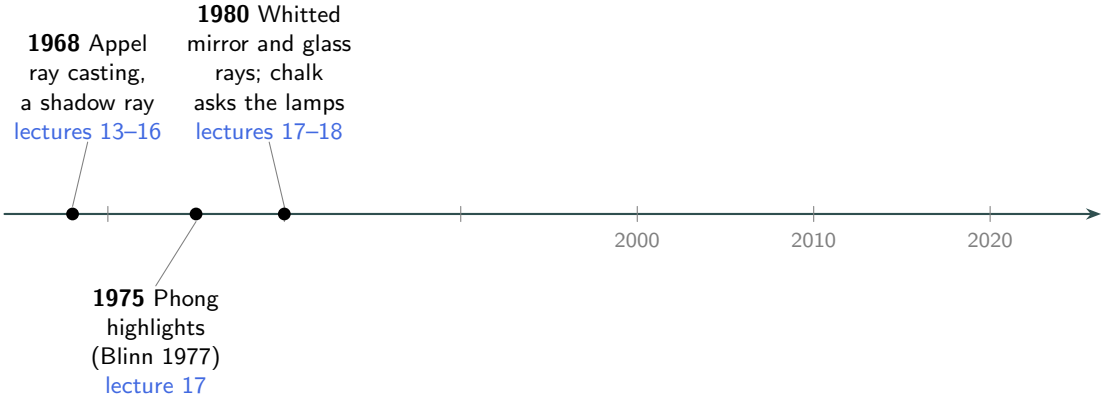
random rays only, 16 paths



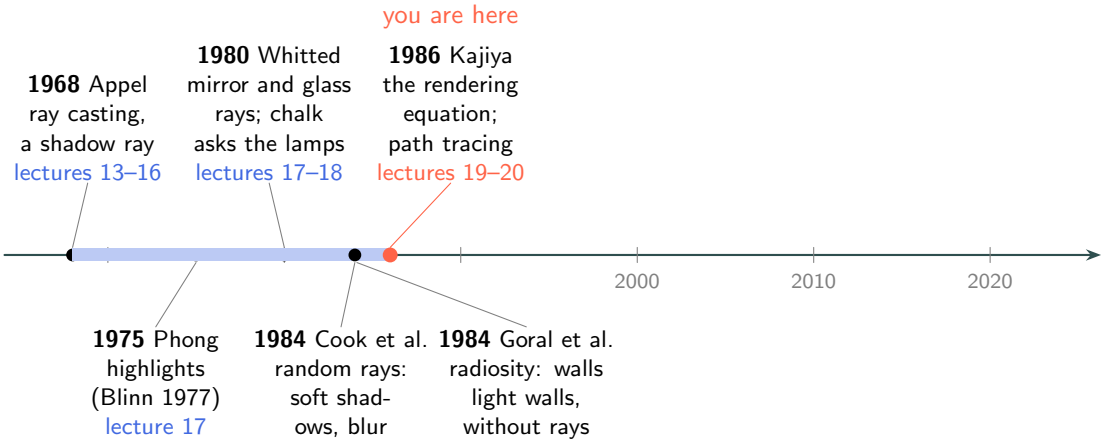
with the shadow ray, 16 paths

- Same room, same lamp, same work.
- The shadow ray removes most of the noise.

Fifty years of rays: where this course stands

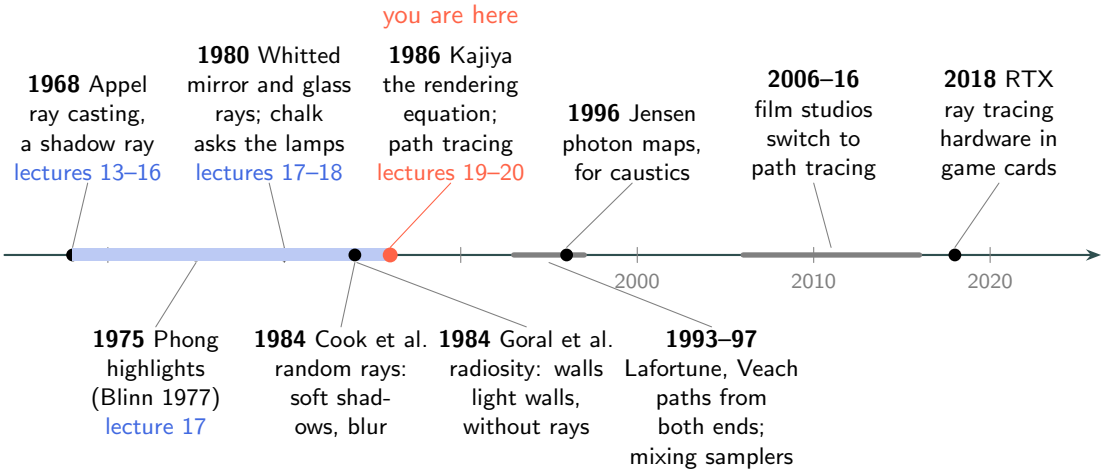


Fifty years of rays: where this course stands



- The blue band is this course so far: 1968 to 1986.

Fifty years of rays: where this course stands



- The blue band is this course so far: 1968 to 1986.

Summary

1 Monte Carlo

$$\int f \approx \frac{1}{N} \sum \frac{f(x_k)}{p(x_k)}$$

error $\propto 1/\sqrt{N}$

2 On the dome

cosine picks: $\frac{\rho\, L_i \cos\theta}{\cos\theta/\pi} = R\, L_i$

$\mathbf{d} = \mathbf{n} + \mathbf{u}$ (RTIOW)

3 The path tracer

```
albedo * trace(Ray(x, d), depth - 1)
```

a lamp ends the path

depth or roulette stops it

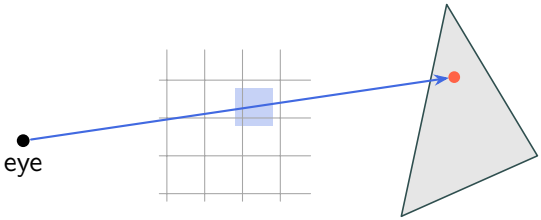
4 Aim at the lamp

plus each lamp's share, by a shadow ray

depth 0: Whitted's tracer

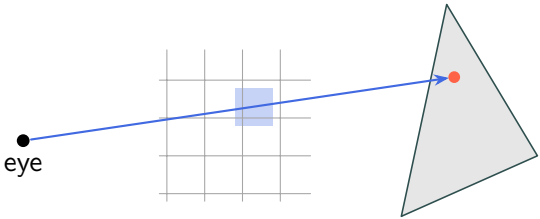
noise is the price; more paths per pixel

Next: draw triangles instead of tracing rays

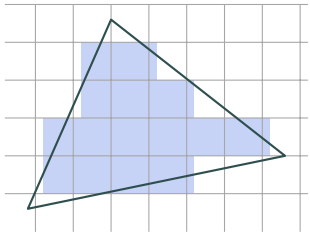


tracing: each pixel
looks for its surface

Next: draw triangles instead of tracing rays

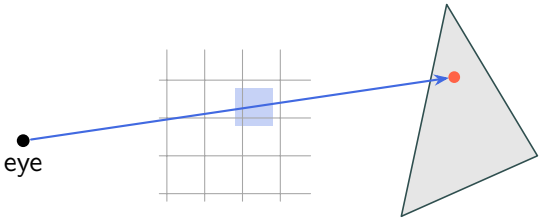


tracing: each pixel
looks for its surface

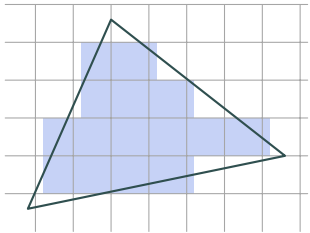


drawing: each triangle
fills its pixels

Next: draw triangles instead of tracing rays



tracing: each pixel
looks for its surface



drawing: each triangle
fills its pixels

- Everything after the midterm draws triangles instead of tracing them.

Exit Ticket

Today's question *(2 minutes)*

A chalk floor point ($R = 0.6$) lies under an even sky of radiance $L = 2$. One random direction is picked: straight up, $\theta = 0$.

(a) With even picks, what does this one sample estimate L_o to be?

(b) With cosine picks?

CS 453 — Exit Ticket #20

Name: _____

Roll #: _____ Date: _____

Answer:

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