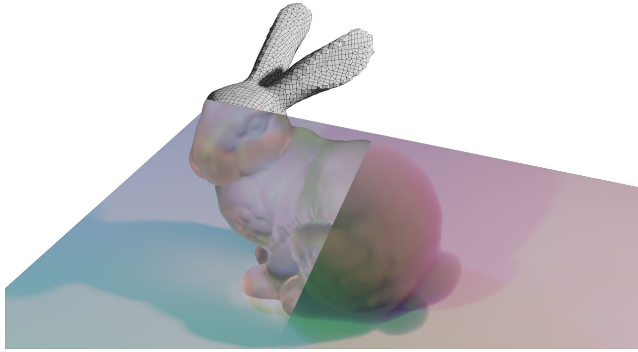
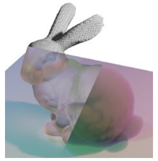


3. Path Tracing

Physically Based Rendering

Shinyoung Yi (이신영)

Whiteboard to review

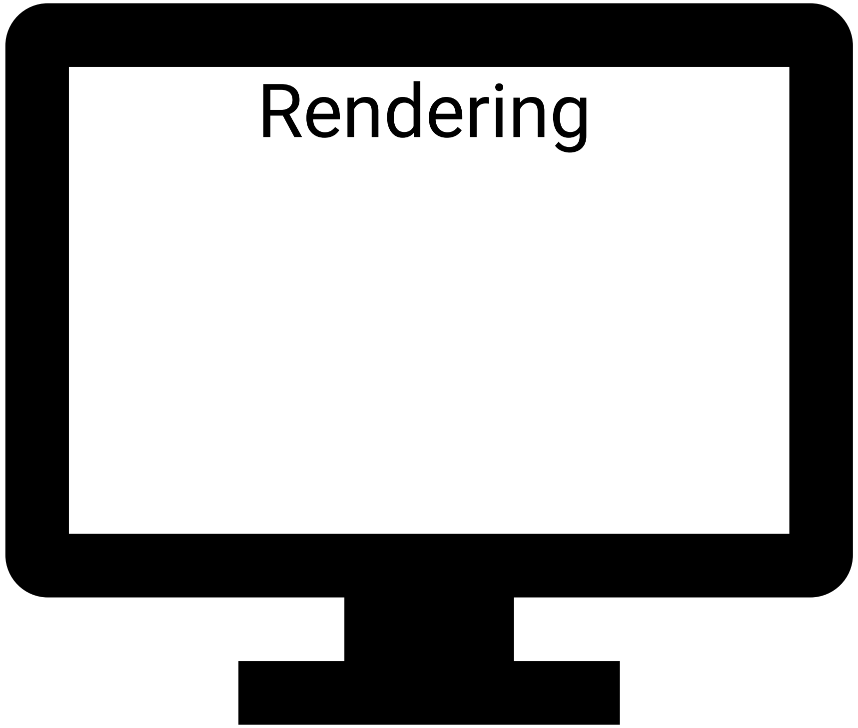


Natural phenomena (physical law)

Radiometry:

Rendering equation

$$L(p, \widehat{\omega}) = L_e(p, \widehat{\omega}) + \int_{\mathbb{S}^2} L(\mathrm{r}(p, \widehat{\omega}'), -\widehat{\omega}')\rho(p, \widehat{\omega}', \widehat{\omega})\mathrm{d}\widehat{\omega}$$

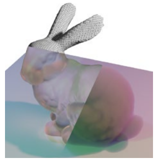


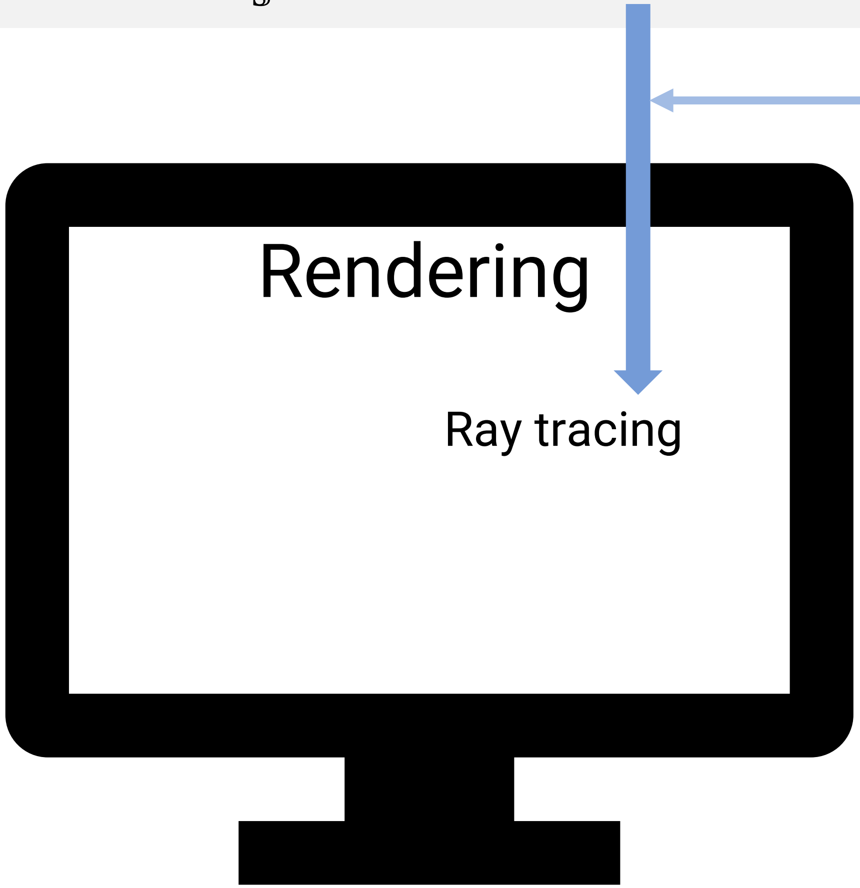
Natural phenomena
(physical law)

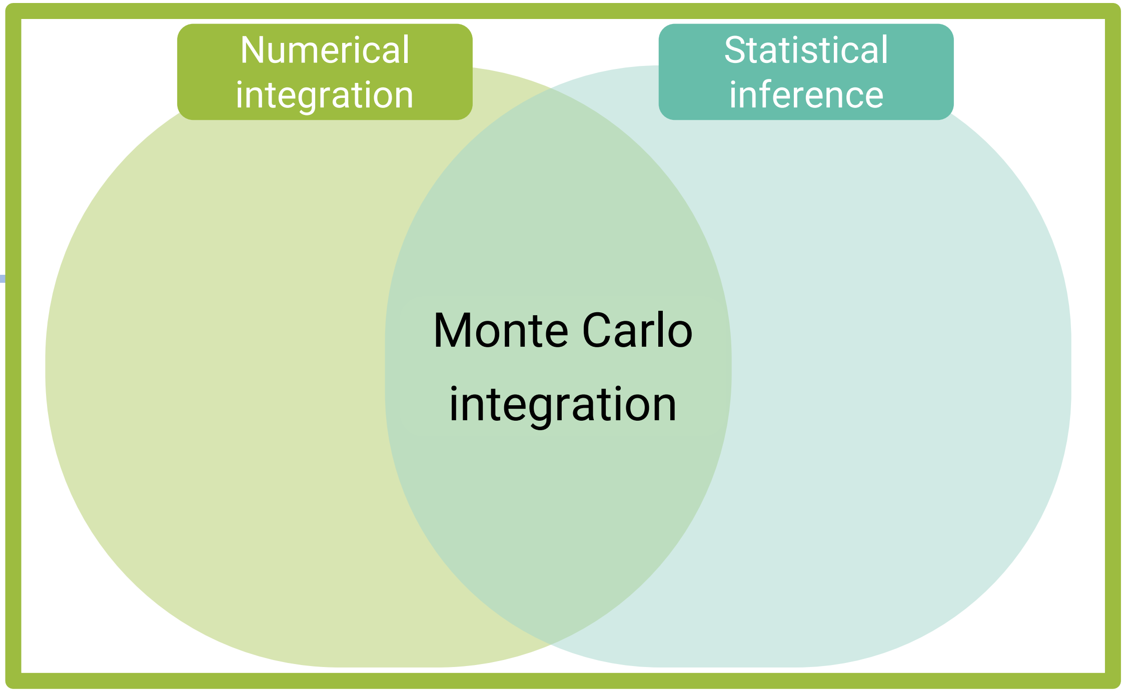
Radiometry:

Rendering equation

$$L(p, \hat{\omega}) = L_e(p, \hat{\omega}) + \int_{\mathbb{S}^2} L(\mathrm{r}(p, \hat{\omega}'), -\hat{\omega}') \rho(p, \hat{\omega}', \hat{\omega}) \mathrm{d}\hat{\omega}$$





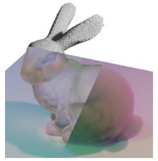


Radiometry

Monte Carlo

Implementation

Ver 0.1: Normal integrator — Radiometry — Impl.



Ver 0.2: + inheriting `mi.Integrator` — Impl.

Ver 0.3: Direct illumination
using BSDF sampling — Radiometry — Monte Carlo

Ver 0.4: Direct illumination
using Emitter sampling — Monte Carlo

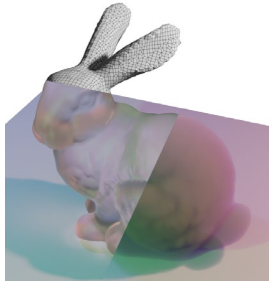
Ver 0.5: Multiple importance sampling — Monte Carlo

Ver 0.6: + Dirac delta — Monte Carlo

Ver 0.7: Path tracing with fixed depth — Radiometry

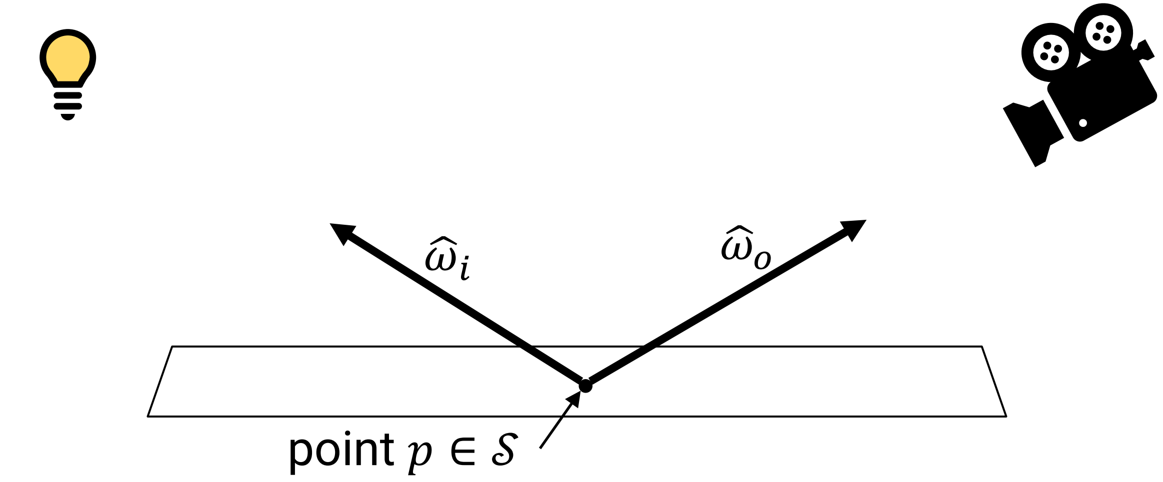
Ver 0.8: + Dr.Jit megakernel:
`active` masks and `mi.Loop` — Impl.

Ver 1.0: + Russian roulette — Radiometry — Monte Carlo



Review: rendering equation

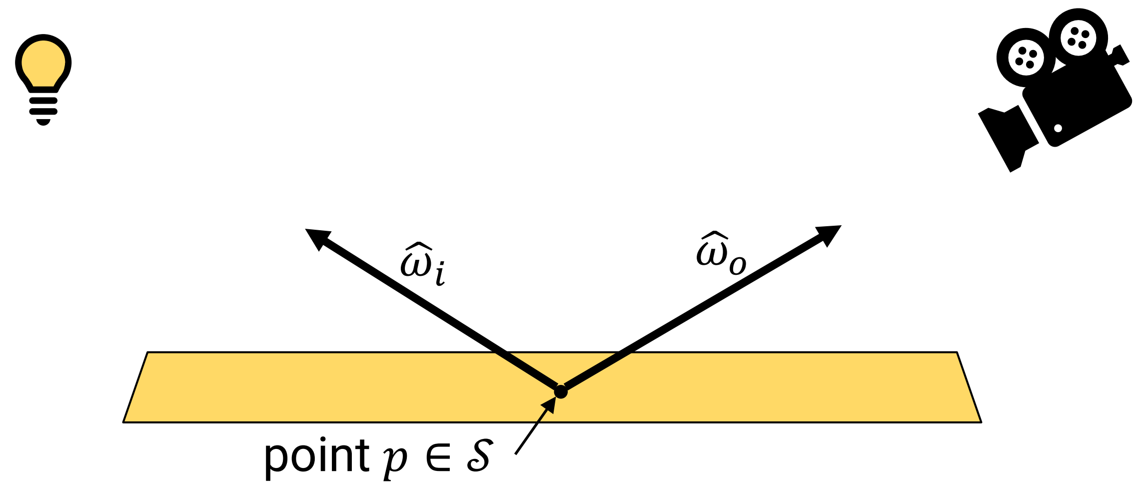
Rendering equation



From definition of BRDFs…

$$L^{(\text{out})}(p, \widehat{\omega}_o) = \int_{\mathbb{S}^2} L^{(\text{in})}(p, \widehat{\omega}_i) f_s(p, \widehat{\omega}_i, \widehat{\omega}_o) |\hat{n} \cdot \widehat{\omega}_i| \mathrm{d}\widehat{\omega}_i$$

Rendering equation



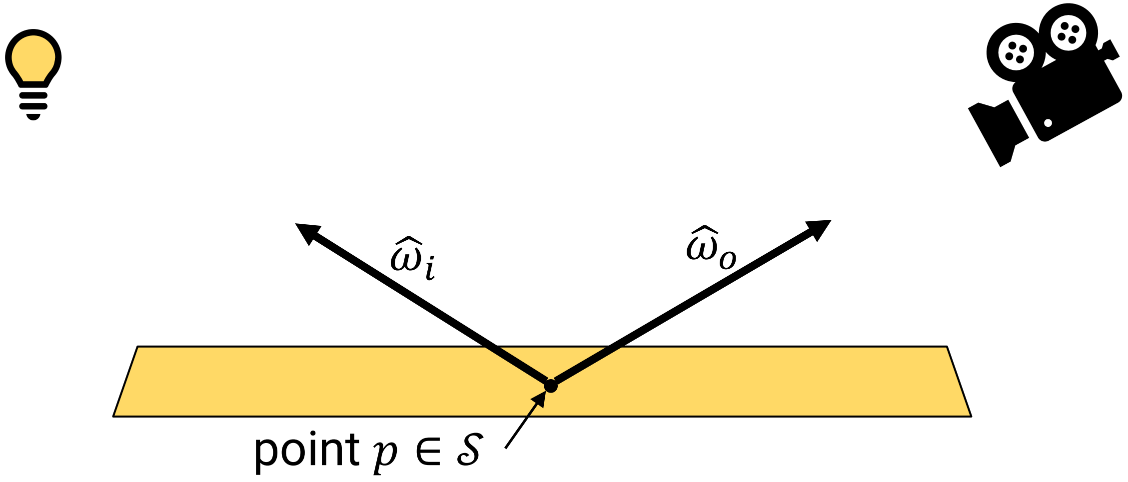
From definition of BRDFs…

+ Emission

Rendering equation
(light transport equation)

$$L^{(\mathrm{out})}(p, \widehat{\omega}_o) = L_e(p, \omega_{\hat{o}}) + \int_{\mathbb{S}^2} L^{(\mathrm{in})}(p, \widehat{\omega}_i) f_s(p, \widehat{\omega}_i, \widehat{\omega}_o) |\hat{n} \cdot \widehat{\omega}_i| \mathrm{d}\widehat{\omega}_i$$

Rendering equation

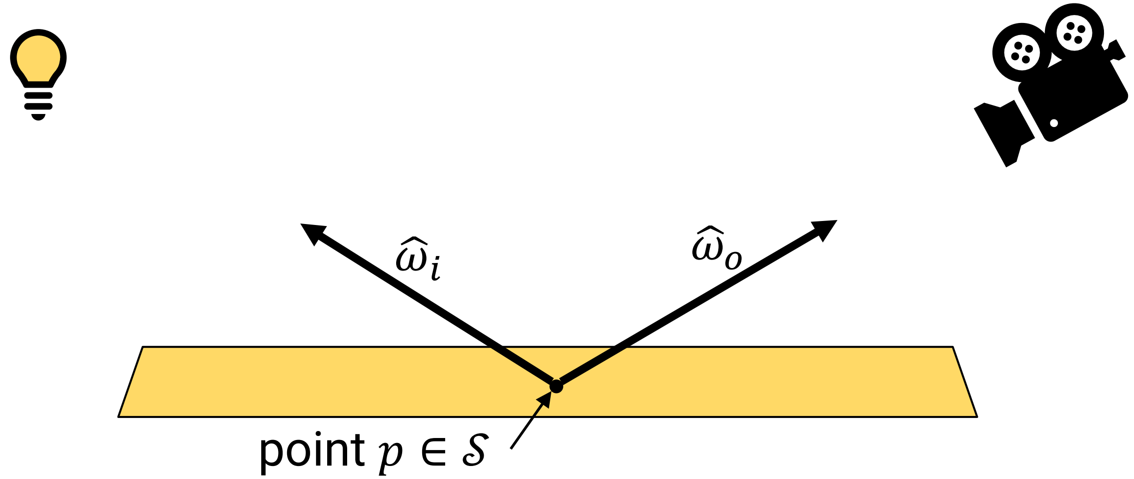


Q. What are knowns and unknowns?

Rendering equation
(light transport equation)

$$L^{(\mathrm{out})}(p, \widehat{\omega}_o) = L_e(p, \omega_{\hat{o}}) + \int_{\mathbb{S}^2} L^{(\mathrm{in})}(p, \widehat{\omega}_i) f_s(p, \widehat{\omega}_i, \widehat{\omega}_o) |\hat{n} \cdot \widehat{\omega}_i| \mathrm{d}\widehat{\omega}_i$$

Rendering equation



Q. What are knowns and unknowns?

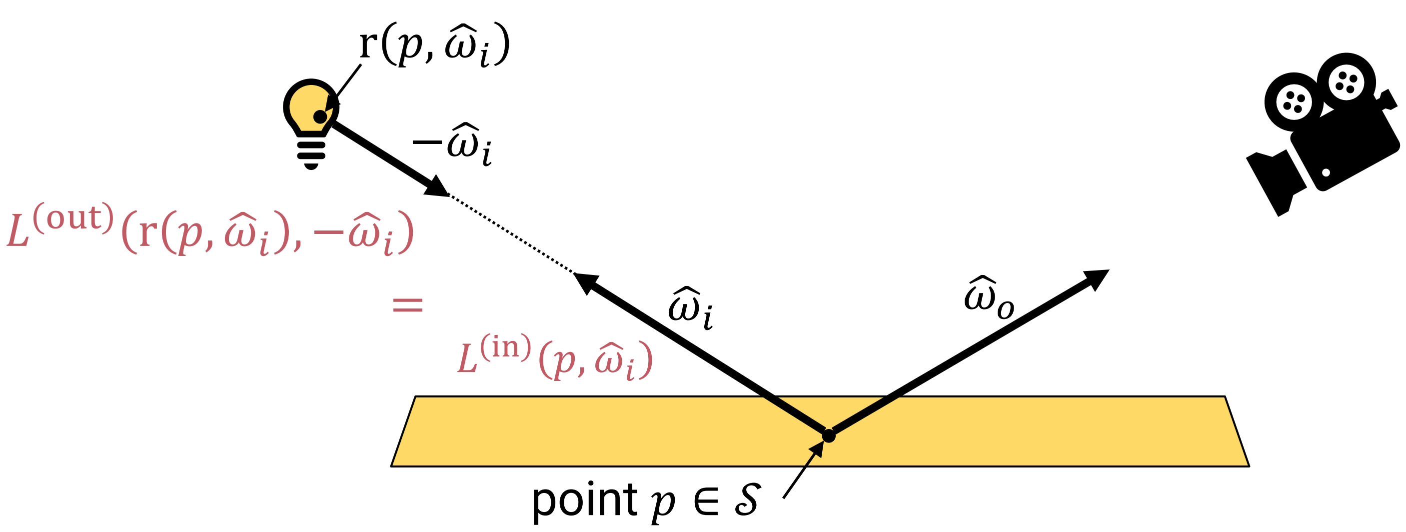
Rendering equation
(light transport equation)

known

$$L^{(\text{out})}(p, \widehat{\omega}_o) = L_e(p, \omega_{\hat{o}}) + \int_{\mathbb{S}^2} L^{(\text{in})}(p, \widehat{\omega}_i) f_s(p, \widehat{\omega}_i, \widehat{\omega}_o) |\hat{n} \cdot \widehat{\omega}_i| \mathrm{d}\widehat{\omega}_i$$

unknown

Rendering equation

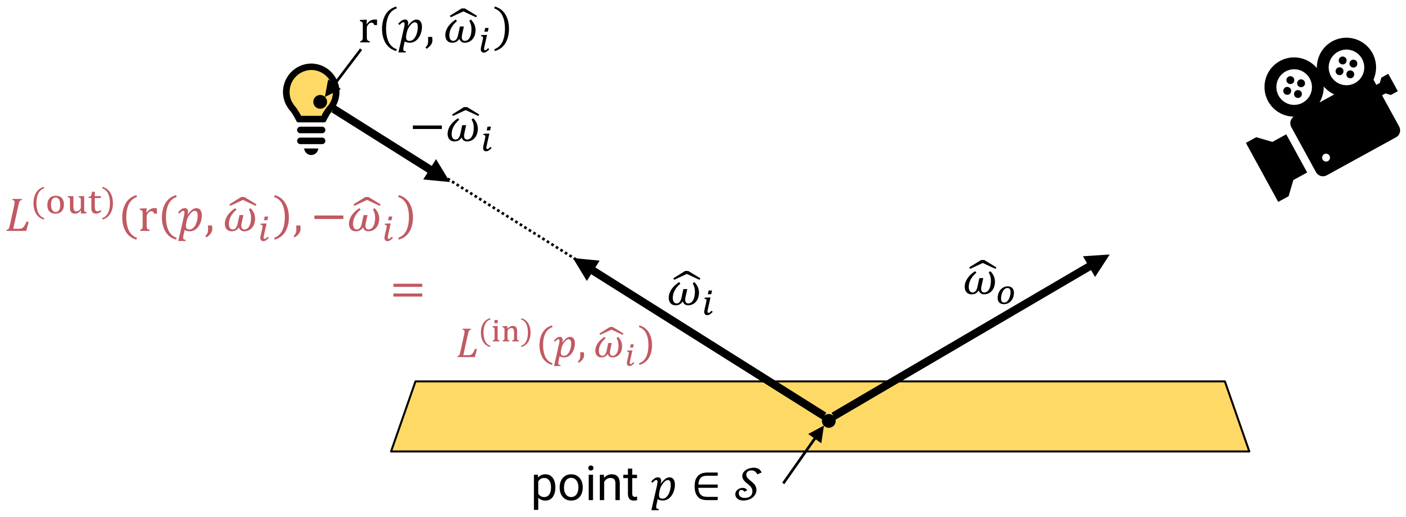


Q. Any relationship between $L^{(\mathrm{out})}$ and $L^{(\mathrm{in})}$?

Rendering equation
(light transport equation)

$$L^{(\mathrm{out})}(p,\widehat{\omega}_o) = L_e(p,\omega_{\hat{o}}) + \int_{\mathbb{S}^2} L^{(\mathrm{in})}(p,\widehat{\omega}_i) f_s(p,\widehat{\omega}_i,\widehat{\omega}_o)|\hat{n}\cdot\widehat{\omega}_i|\mathrm{d}\widehat{\omega}_i$$

Rendering equation

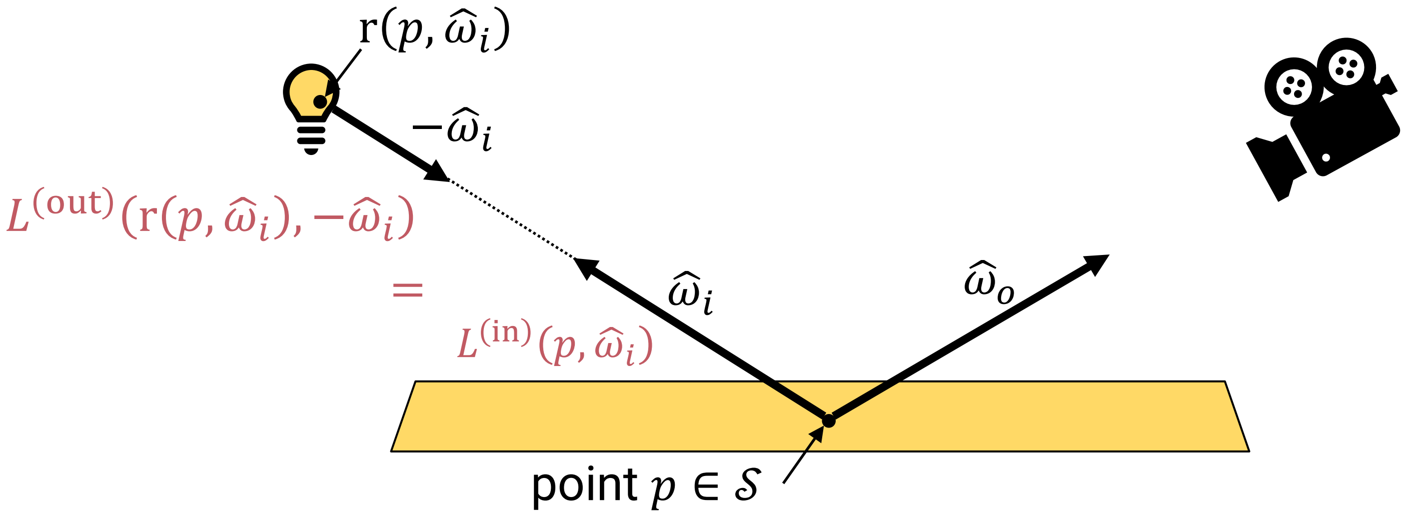


Q. Any relationship between $L^{(\mathrm{out})}$ and $L^{(\mathrm{in})}$?

Rendering equation
(light transport equation)

$$L^{(\mathrm{out})}(p, \widehat{\omega}_o) = L_e(p, \omega_{\hat{o}}) + \int_{\mathbb{S}^2} L^{(\mathrm{out})}(\mathrm{r}(p, \widehat{\omega}_i), -\widehat{\omega}_i) f_s(p, \widehat{\omega}_i, \widehat{\omega}_o) |\hat{n} \cdot \widehat{\omega}_i| \mathrm{d}\widehat{\omega}_i$$

Rendering equation

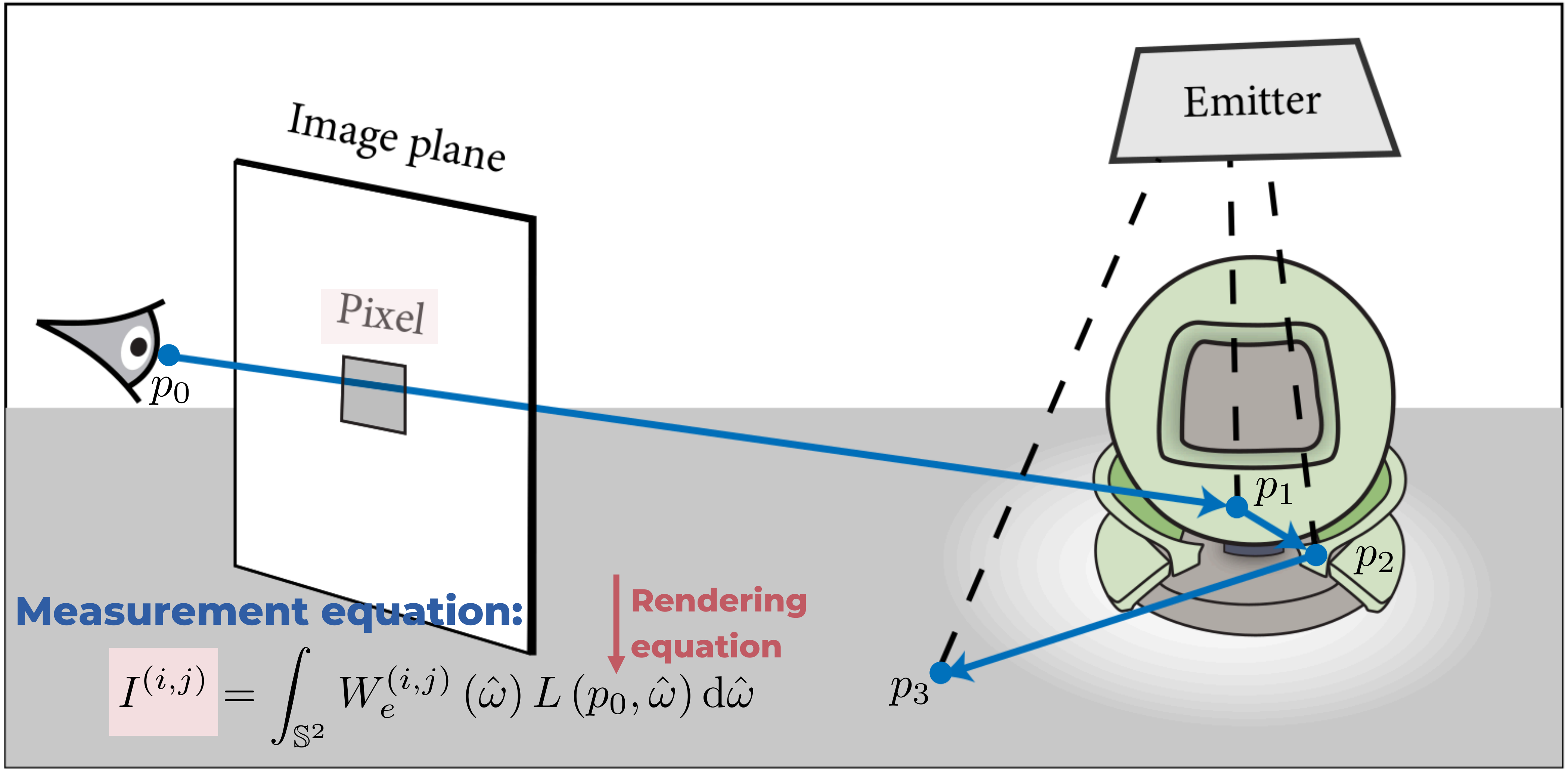


Q. Any relationship between $L^{(\mathrm{out})}$ and $L^{(\mathrm{in})}$?

Rendering equation
(light transport equation)

$$L^{(\mathrm{out})}(p, \widehat{\omega}_o) = L_e(p, \omega_{\hat{o}}) + \int_{\mathbb{S}^2} L^{(\mathrm{out})}(\mathrm{r}(p, \widehat{\omega}_i), -\widehat{\omega}_i) f_s(p, \widehat{\omega}_i, \widehat{\omega}_o) |\hat{n} \cdot \widehat{\omega}_i| \mathrm{d}\widehat{\omega}_i$$

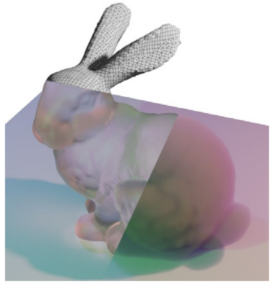
Rendering equation



Measurement equation:

$$I^{(i,j)} = \int_{\mathbb{S}^2} W_e^{(i,j)}(\hat{\omega})\, L(p_0, \hat{\omega})\, \mathrm{d}\hat{\omega}$$

[https://mitsuba.readthedocs.io/en/latest/_images/integrator_path_figure.png]



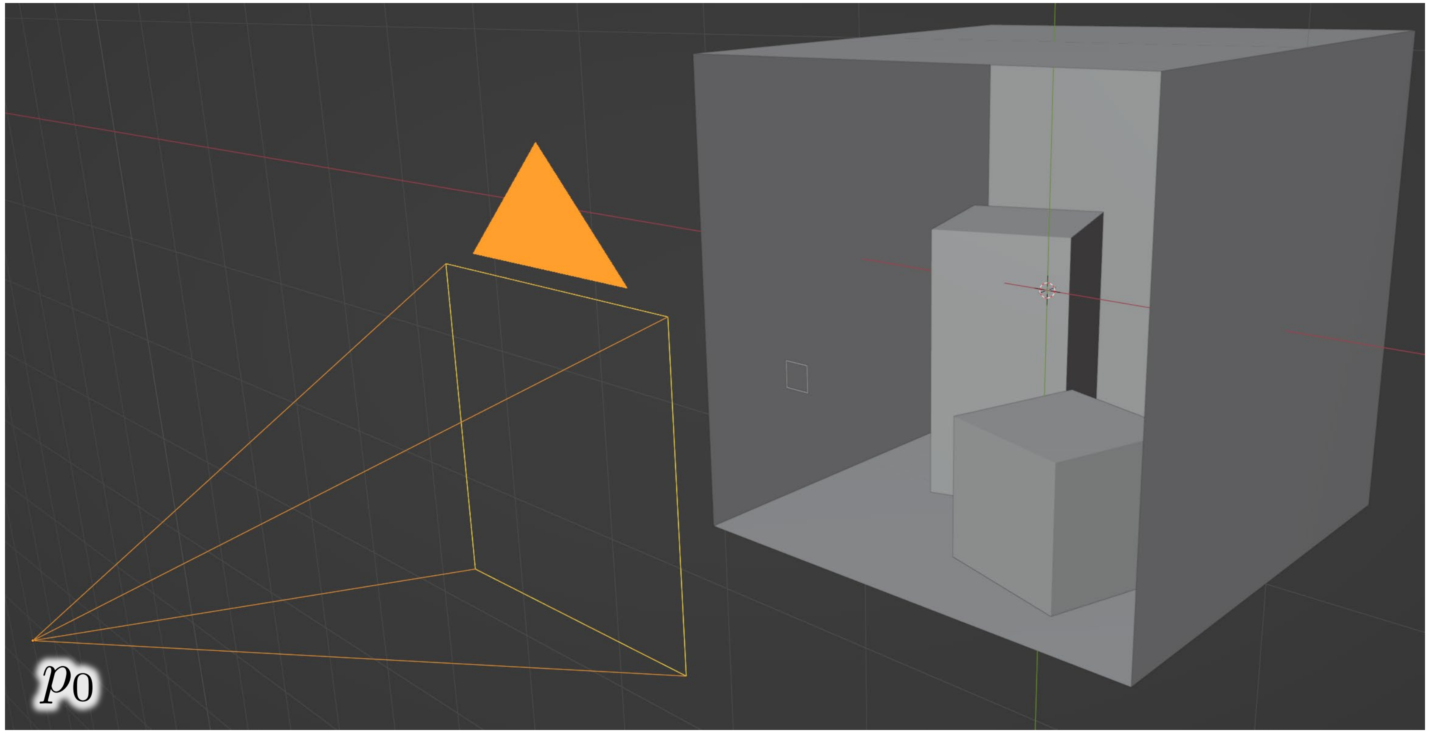
Path Tracing

HW Ver. 0.1

Radiometry

Impl.

`scene: mi.Scene`



`sensor: mi.Sensor`

HW Ver. 0.1

Radiometry

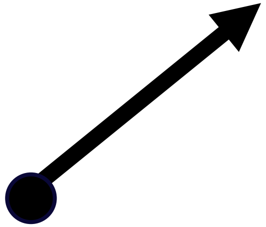
Impl.

1. ray = sensor.sample_ray()

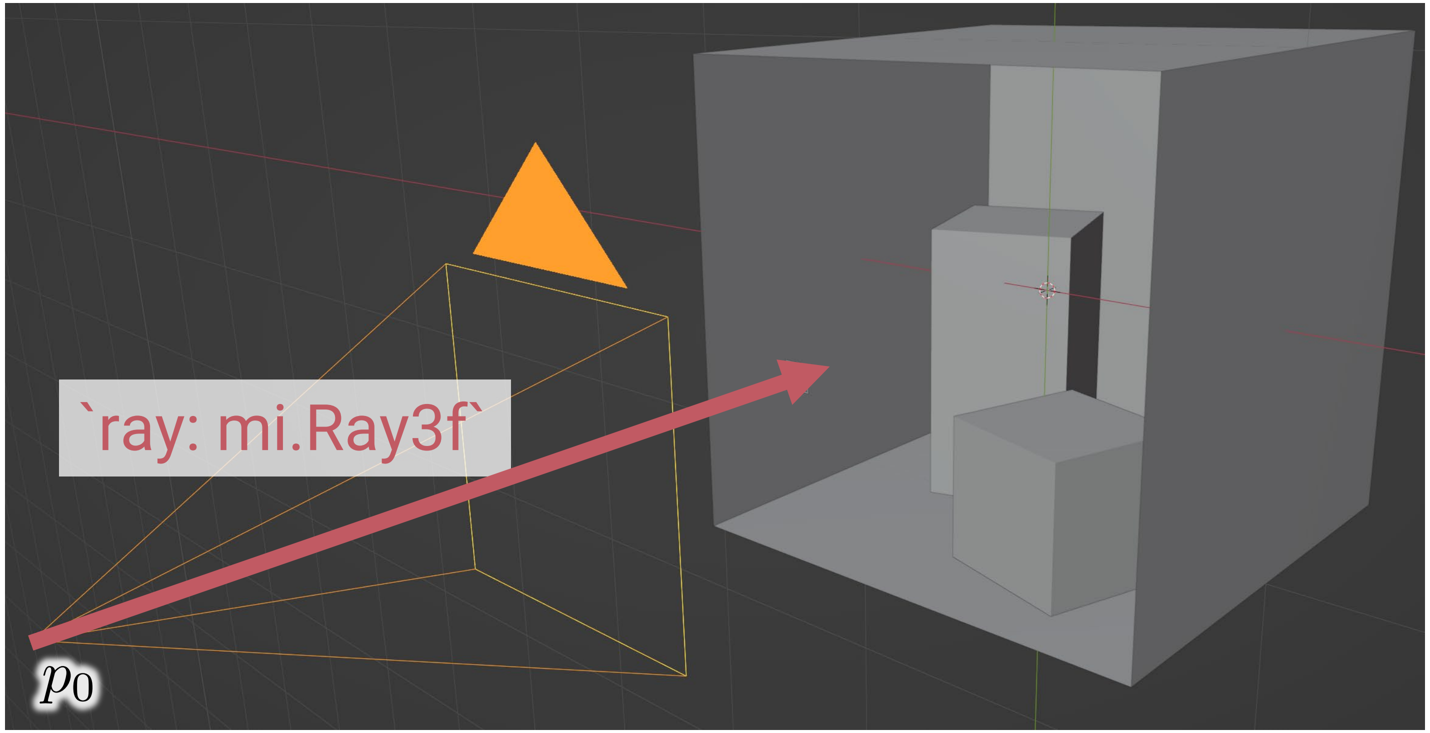
class: mi.Ray3f

ray.o: Point3f

ray.d: Vector3f



`scene: mi.Scene`



`sensor: mi.Sensor`

HW Ver. 0.1

Radiometry

Impl.

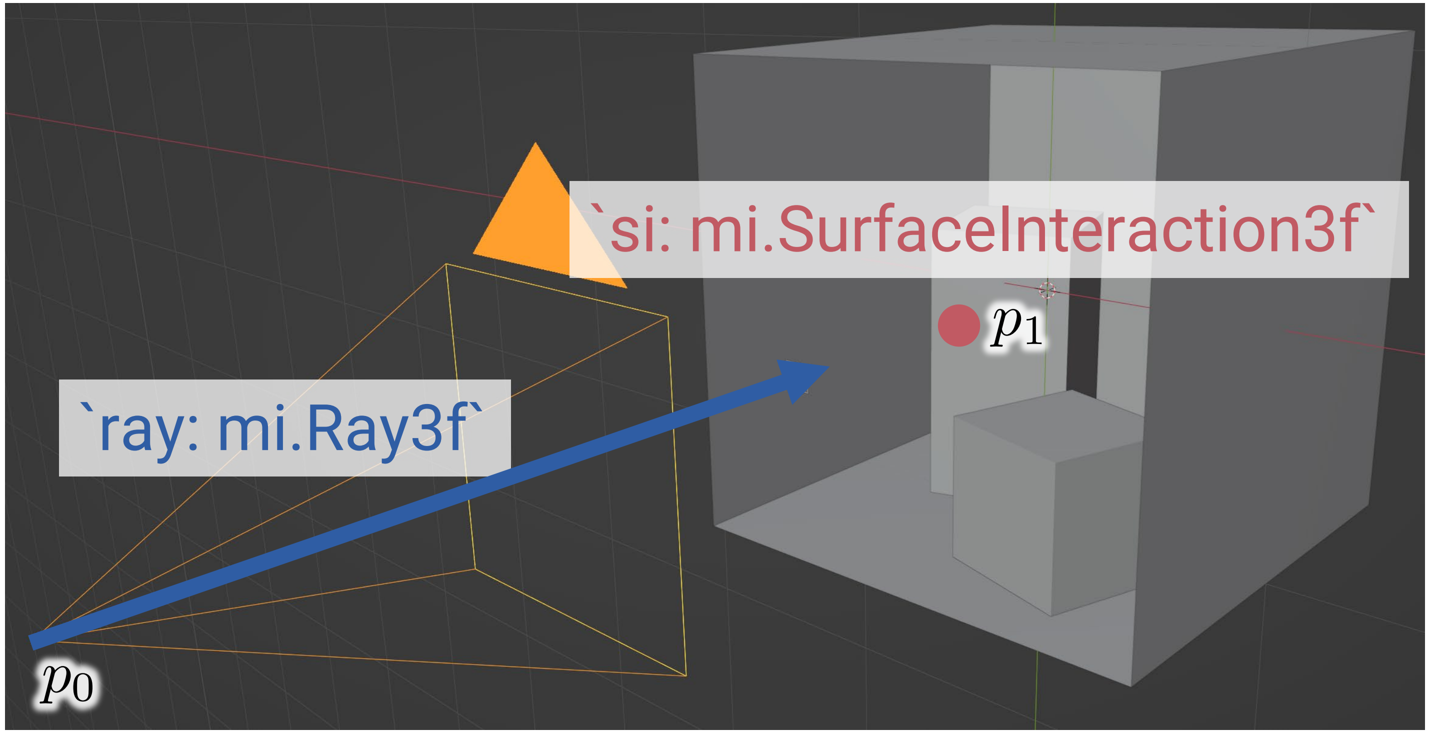
1. ray = sensor.sample_ray()
2. si = scene.ray_intersect(ray)

class: mi.SurfaceInteraction3f

si.p: Point3f

si.wi: -ray.d in local cooridnates

`scene: mi.Scene`

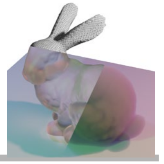


`sensor: mi.Sensor`

HW Ver. 0.1

Radiometry

Impl.



***Task:** Check we obtained correct `ray` and `si` by visualizing normal vectors at `si`*

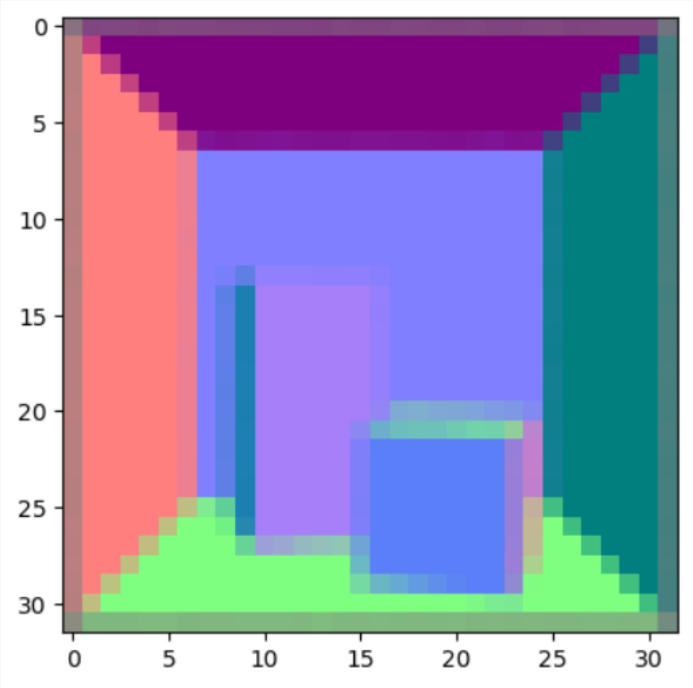
1. ray = sensor.sample_ray()
2. si = scene.ray_intersect(ray)

```
image = render_normal(cbox1_scene, spp=1024)
plt.imshow(image*0.5+0.5)
```

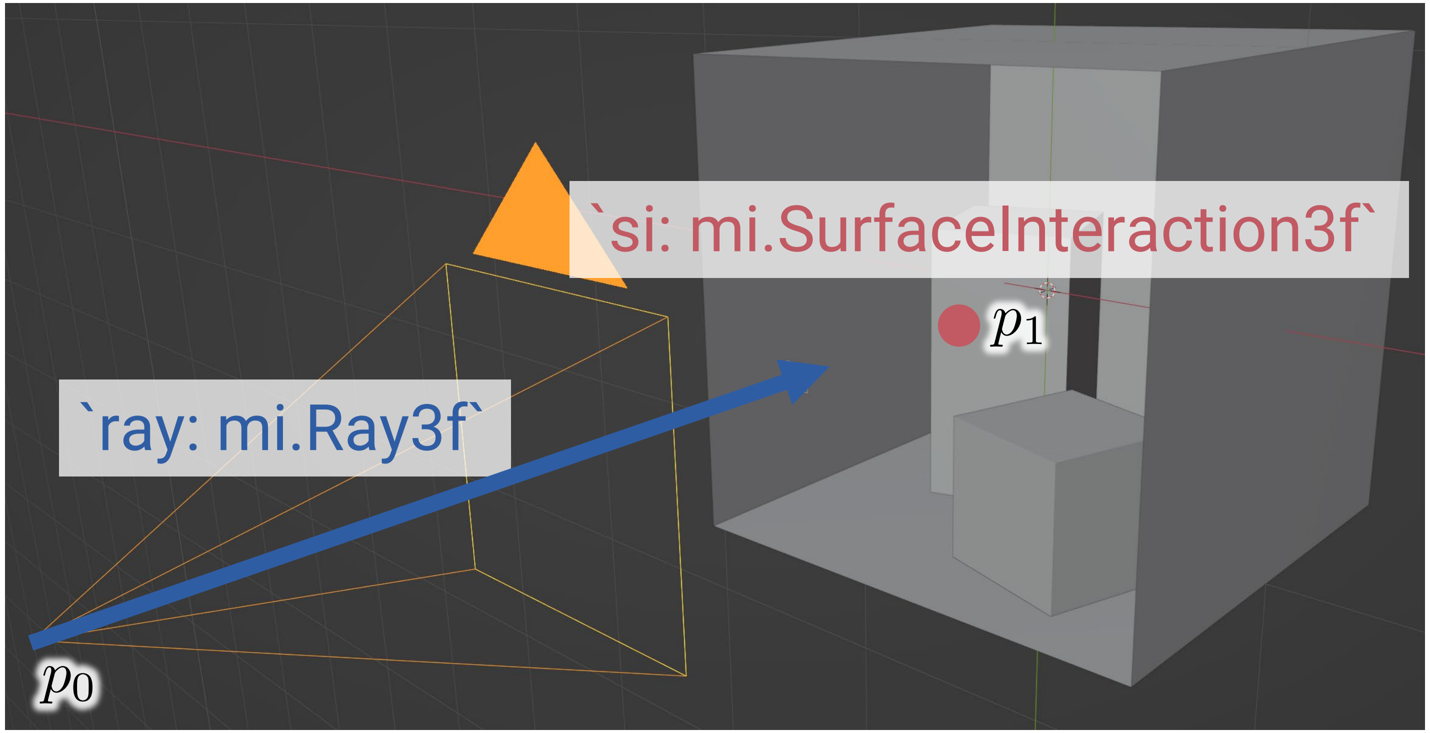
[11] Python

... <matplotlib.image.AxesImage at 0x268e12eeb50>

...

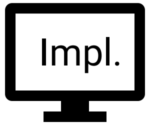


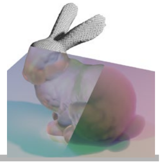
`scene: mi.Scene`



`sensor: mi.Sensor`

HW Ver. 0.2





Task: *Rewrite your code as a subclass of* `mi.Integrator`

class: mi.Integrator

Just use the existing method

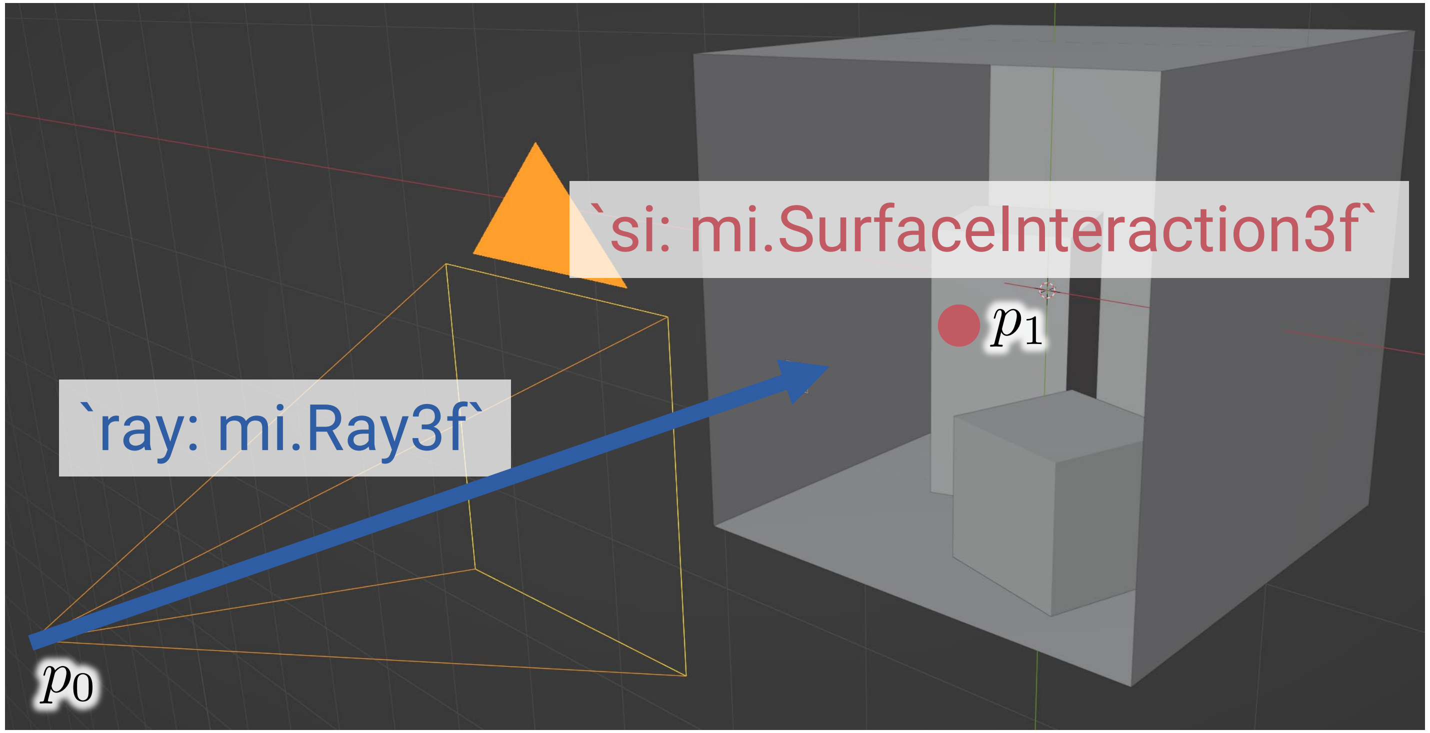
```
def render(scene):
    generate `ray`
    L̂ ←self.sample(scene, ray)
    Estimate measurement equation
```

$$\hat{I} \leftarrow \frac{W_e(W)}{p_{\mathrm{Sensor}}(W)} \hat{L}$$

Override to estimate $L(p_1, \hat{\omega})$ from Ver. 0.3

```
def sample(scene, ray):
    given `ray`
    find L̂(si.p, −ray.d)
```

`scene: mi.Scene`

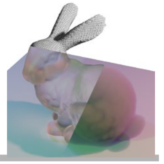


`sensor: mi.Sensor`

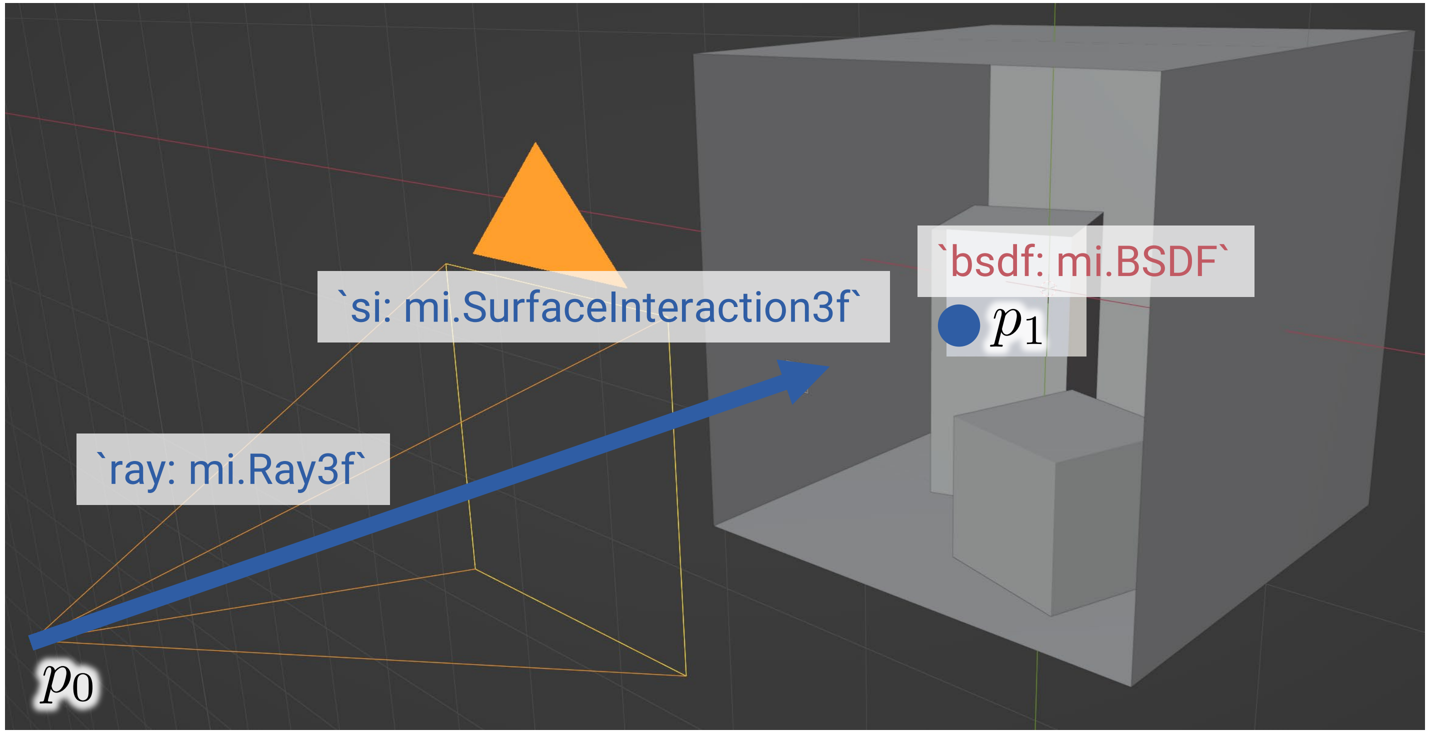
HW Ver. 0.3: BSDF sampling

Radiometry

Monte Carlo



1. ray = sensor.sample_ray()
2. si = scene.ray_intersect(ray)
3. bsdf = si.bsdf()



class: mi.BSDF

    # Methods

    def eval(si, w):

        $\mathrm{f}_s^{\perp} = f_s(\mathrm{w}, \mathrm{si.wi}) \cos\theta_i$

    def sample(..., si):

        generate a random direction w,

        with a PDF

        hopefully $\propto f_s(w, \mathrm{si.wi}) \cos\theta$

    def pdf():

        get PDF for the above method

HW Ver. 0.3: BSDF sampling

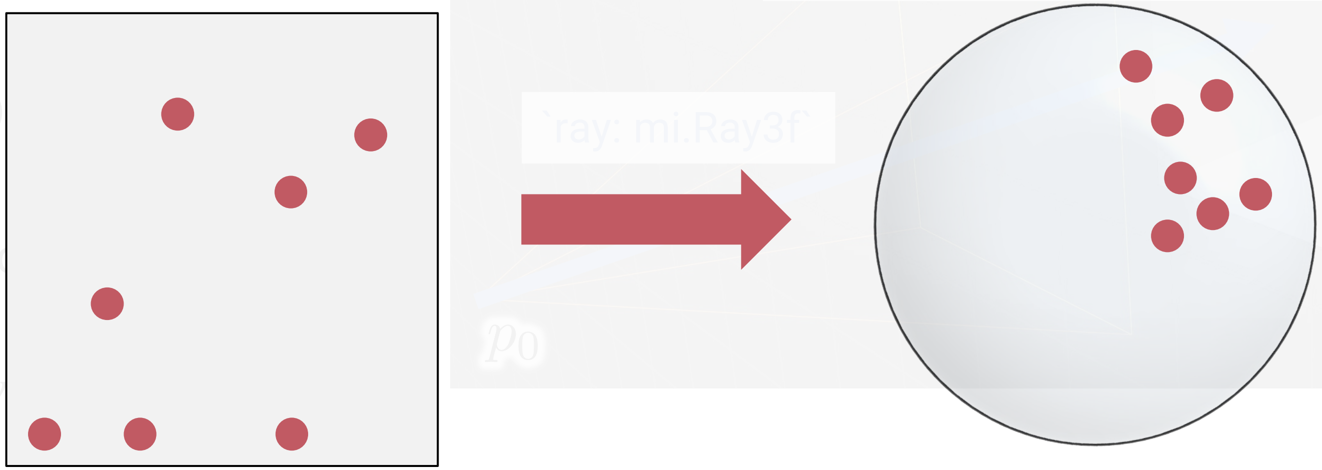
Radiometry

Monte Carlo

Mitsuba 3 implementation convention

Impl.

```
ctx = mi.BSDFContext()   # Configuration of finite options. Just use default one now
u_1d = sampler.next_1d() # np.random.rand(1), not used now
u_2d = sampler.next_2d() # np.random.rand(2)
bsdf_sample, bsdf_weight = bsdf.sample(ctx, si, u_1d, u_2d)
```

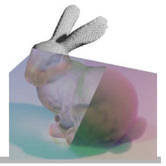


`u_2d = sampler.next_2d()`   `bsdf.sample(ctx, si, u_1d, u_2d)`

HW Ver. 0.3: BSDF sampling

Radiometry

Monte Carlo



1. ray = sensor.sample_ray()
2. si = scene.ray_intersect(ray)
3. bsdf = si.bsdf()

class: mi.BSDF

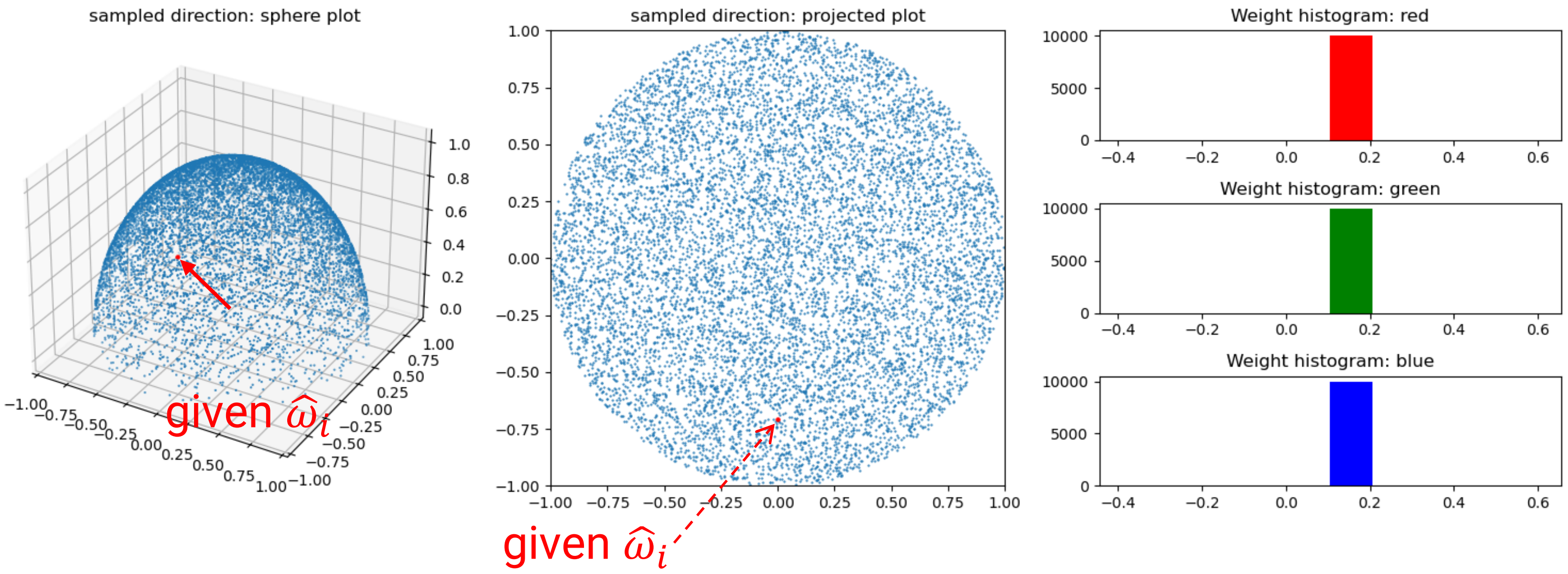
  # Methods

  def eval():

  def sample():

  def pdf():

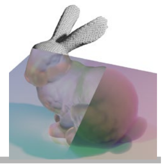
Diffuse BSDF



HW Ver. 0.3: BSDF sampling

Radiometry

Monte Carlo



1. ray = sensor.sample_ray()
2. si = scene.ray_intersect(ray)
3. bsdf = si.bsdf()

class: mi.BSDF

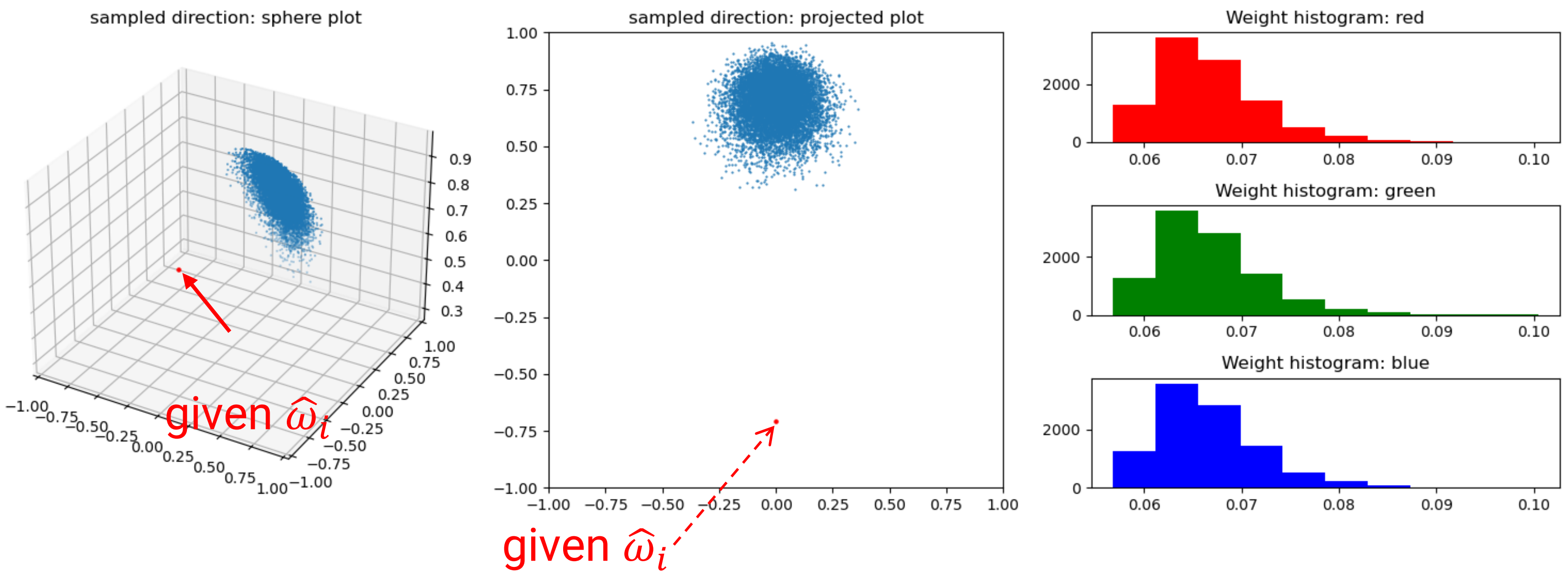
  # Methods

  def eval():

  def sample():

  def pdf():

Rough plastic BSDF

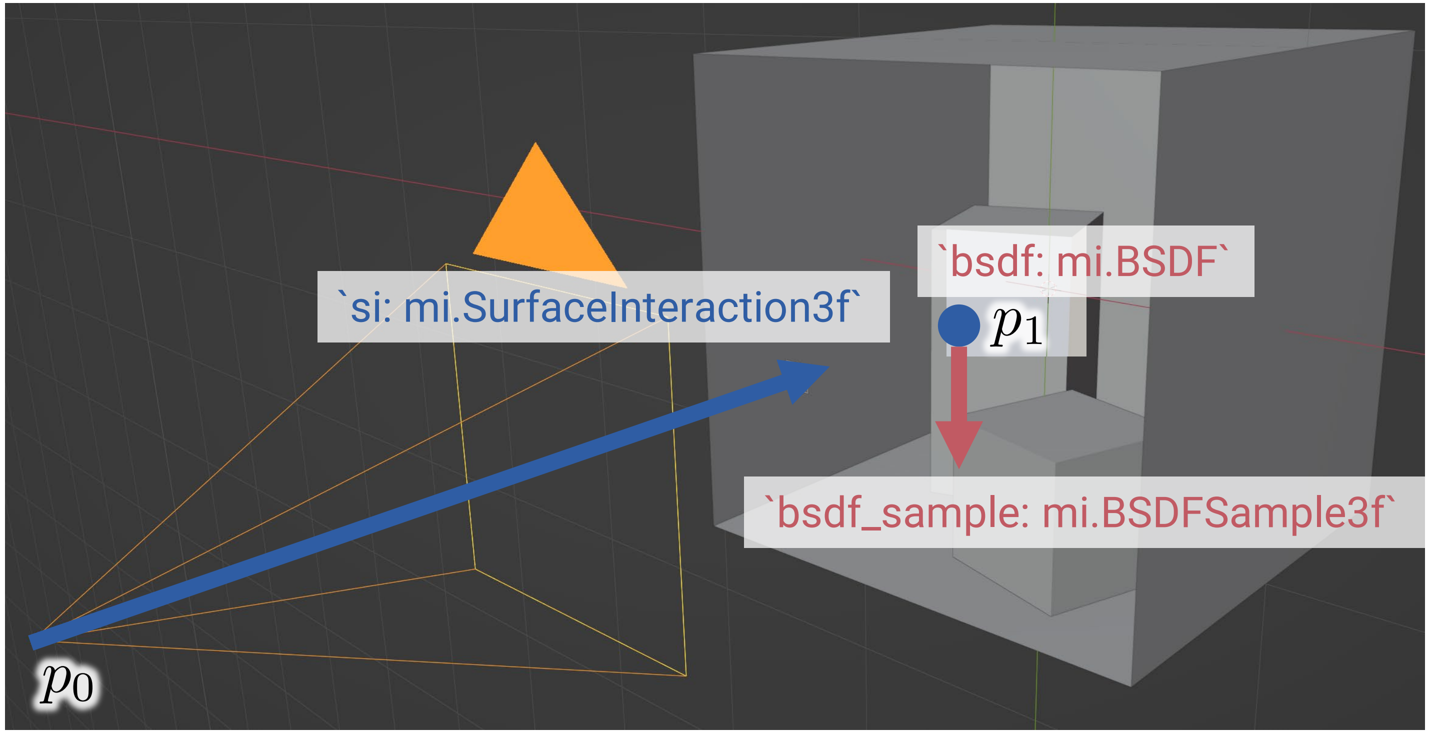


HW Ver. 0.3: BSDF sampling

Radiometry

Monte Carlo

1. ray = sensor.sample_ray()
2. si = scene.ray_intersect(ray)
3. bsdf = si.bsdf()
4. bsdf_sample, bsdf_weight
   = bsdf.sample(..., si, ...)



bsdf_weight: mi.Spectrum

$$= \frac{f_s(\text{bsdf\_sample}, \text{si}) \cos\theta_i}{p(\text{bsdf\_sample})}$$

class: mi.BSDFSample3f

wo: sampled direction, in local coordinates

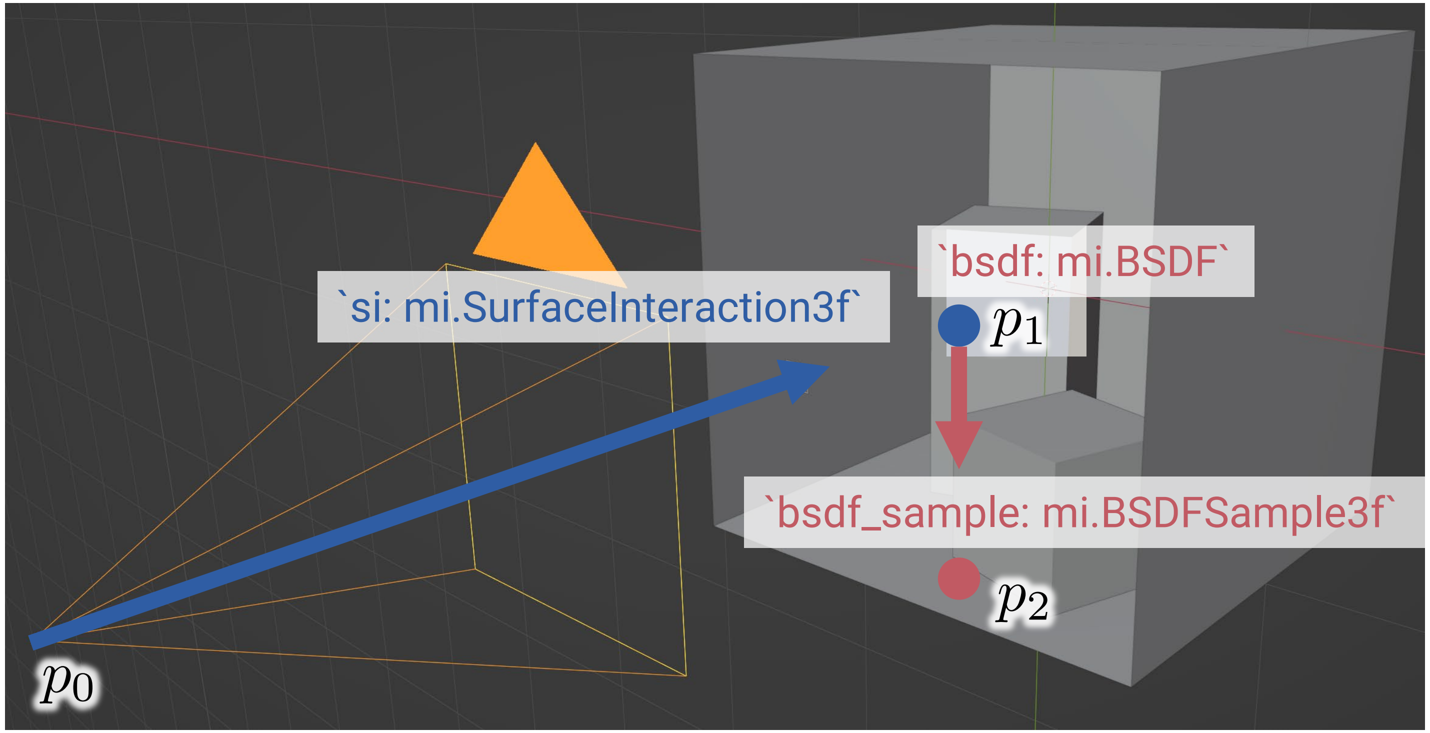
pdf: PDF value at `wo`

HW Ver. 0.3: BSDF sampling

Radiometry

Monte Carlo

1. ray = sensor.sample_ray()
2. si = scene.ray_intersect(ray)
3. bsdf = si.bsdf()
4. bsdf_sample, bsdf_weight
   = bsdf.sample(..., si, ...)
5. find p_2
   - See `spawn_ray` in `tutorial3_BSDF.ipynb`
6. Evaluate $\hat{L}$
   - $\hat{L} = L_{e,10} + \hat{L}_{BSDF,210}$
   - $\hat{L}_{BSDF,210} = \underbrace{\frac{f_{s,210}^{\perp}}{p_{BSDF}(\widehat{\omega}_{12})}}_{\text{bsdf\_weight}} L_{e,21}$

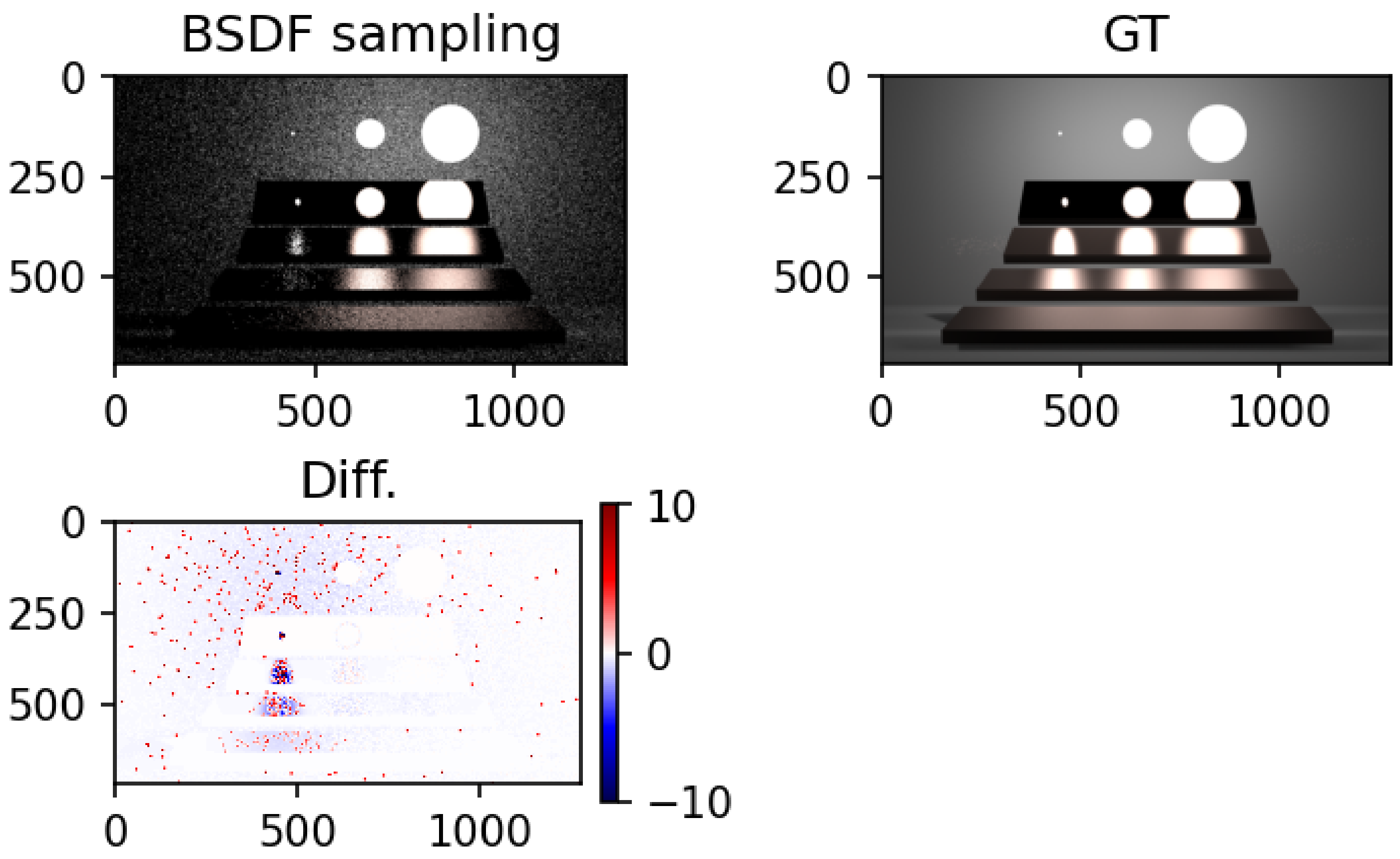


HW Ver. 0.3: BSDF sampling

Radiometry

Monte Carlo

1. ray = sensor.sample_ray()
2. si = scene.ray_intersect(ray)
3. bsdf = si.bsdf()
4. bsdf_sample, bsdf_weight
   = bsdf.sample(…, si, …)
5. find p_2
   - See `spawn_ray` in `tutorial3_BSDF.ipynb`
6. Evaluate $\hat{L}$
   - $\hat{L} = L_{e,10} + \hat{L}_{BSDF,210}$
   - $\hat{L}_{BSDF,210} = \frac{f_{s,210}^{\perp} L_{e,21}}{p_{BSDF}(\widehat{\omega}_{12})}$

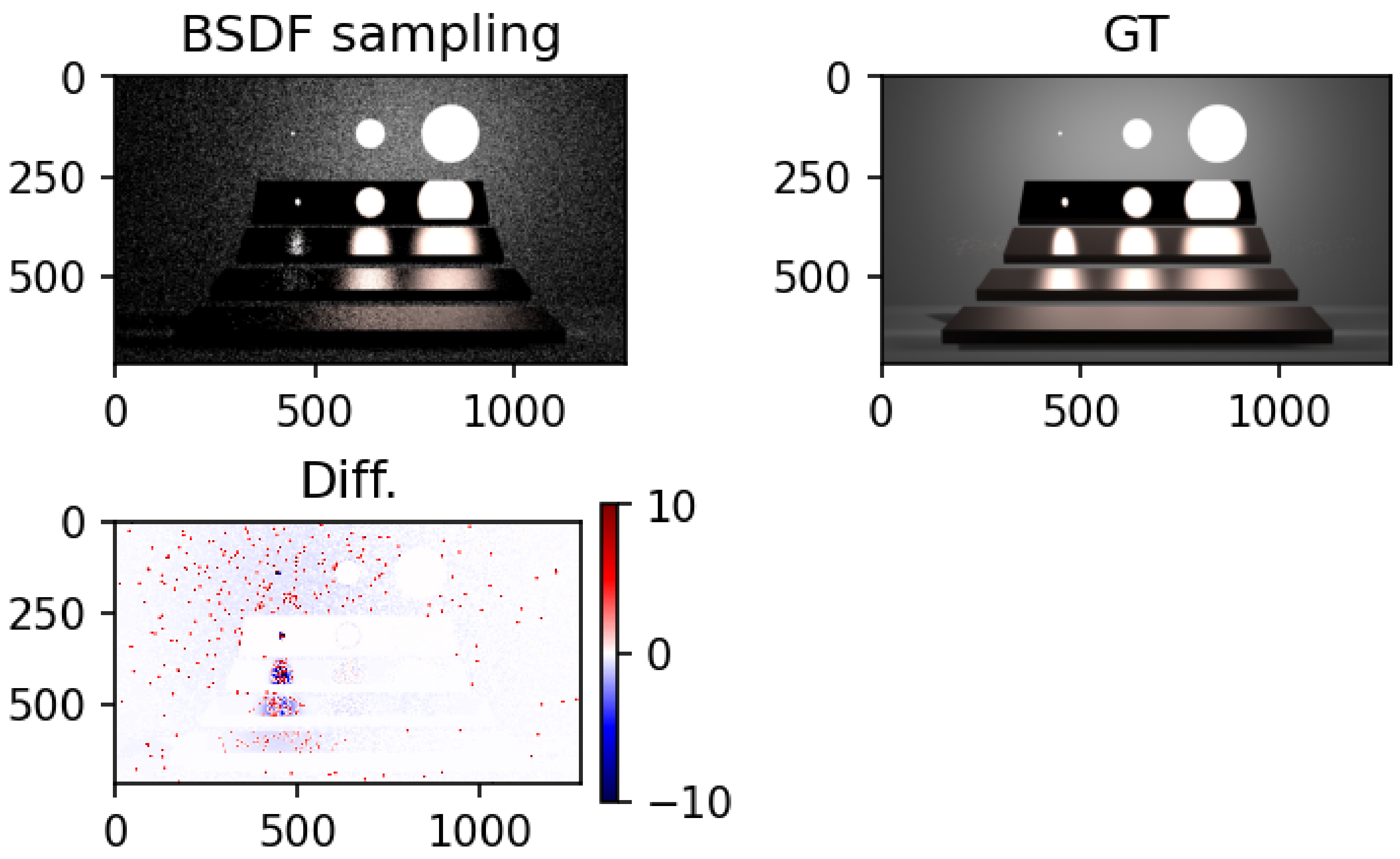


HW Ver. 0.3: BSDF sampling

Radiometry

Monte Carlo

Where does noise come from?



HW Ver. 0.3: BSDF sampling

Radiometry

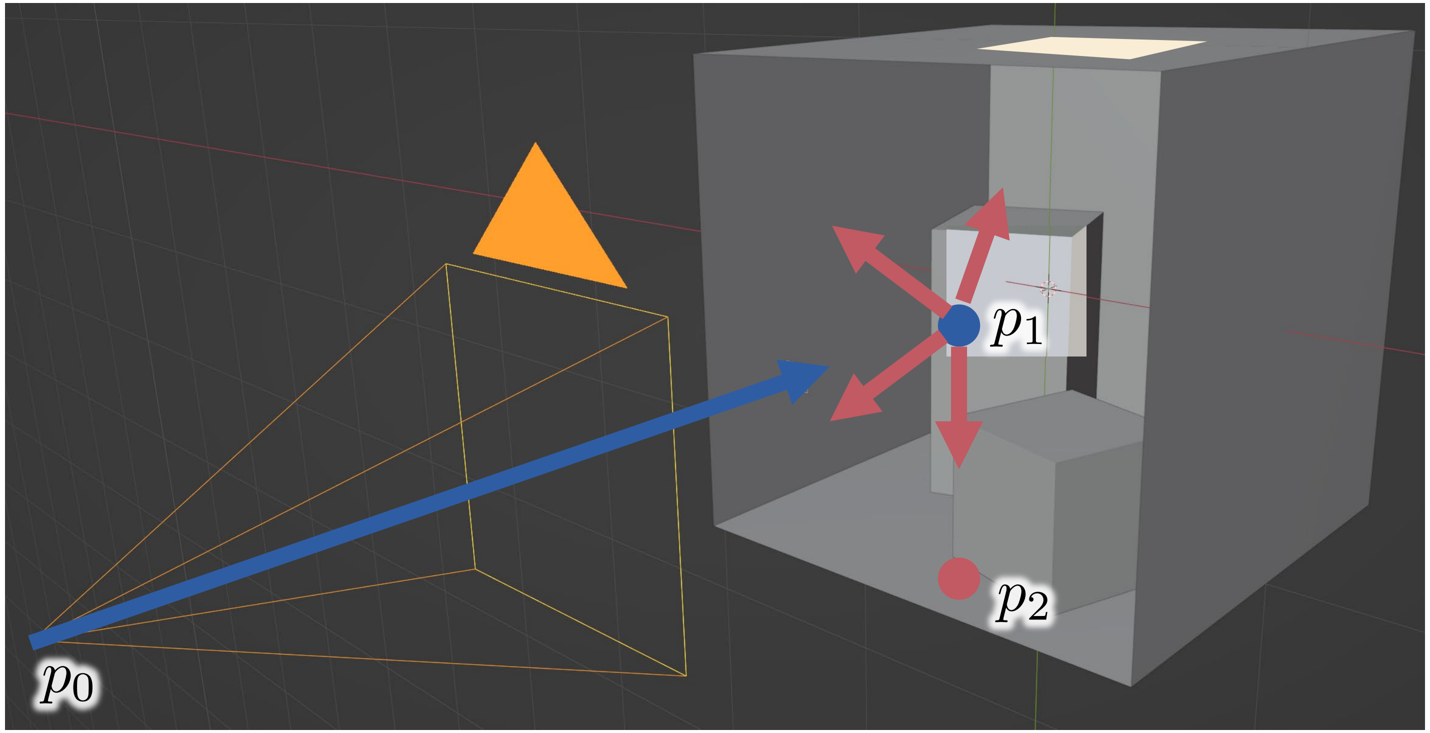
Monte Carlo

Where does noise come from?

Resulting radiance $\hat{L}$ has high variance

← $L_e(\widehat{\omega}_{21})$ has high variance

← $\widehat{\omega}_{21}$ = bsdf.sample() rarely hits an emitter



How will amount of noise change if emitters become smaller?

Variance (noise) ↑

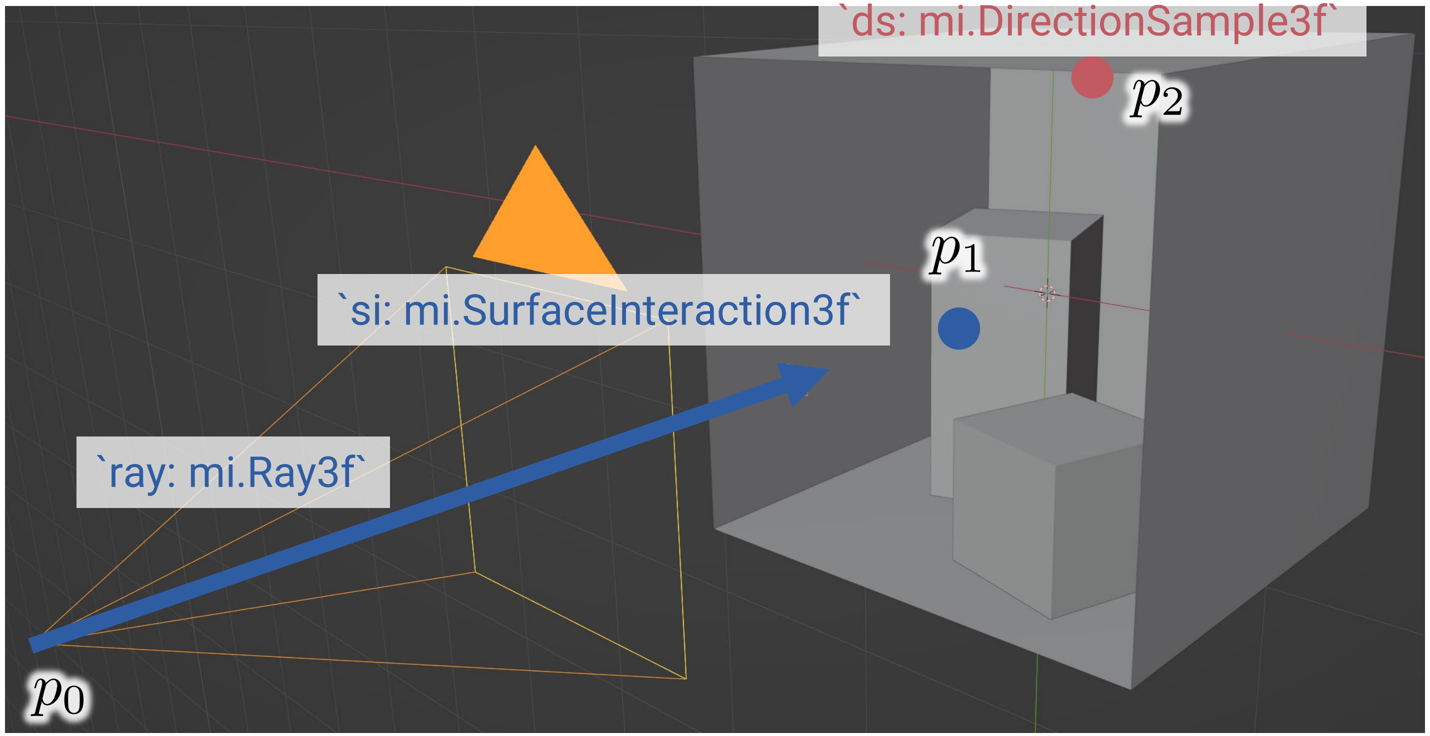
How zero-area emitter?

Biased (inaccurate)

HW Ver. 0.4: Emitter sampling (light sampling)

1. ray = sensor.sample_ray()
2. si = scene.ray_intersect(ray)
3. ds, em_weight = scene.sample_emitter_direction(si)

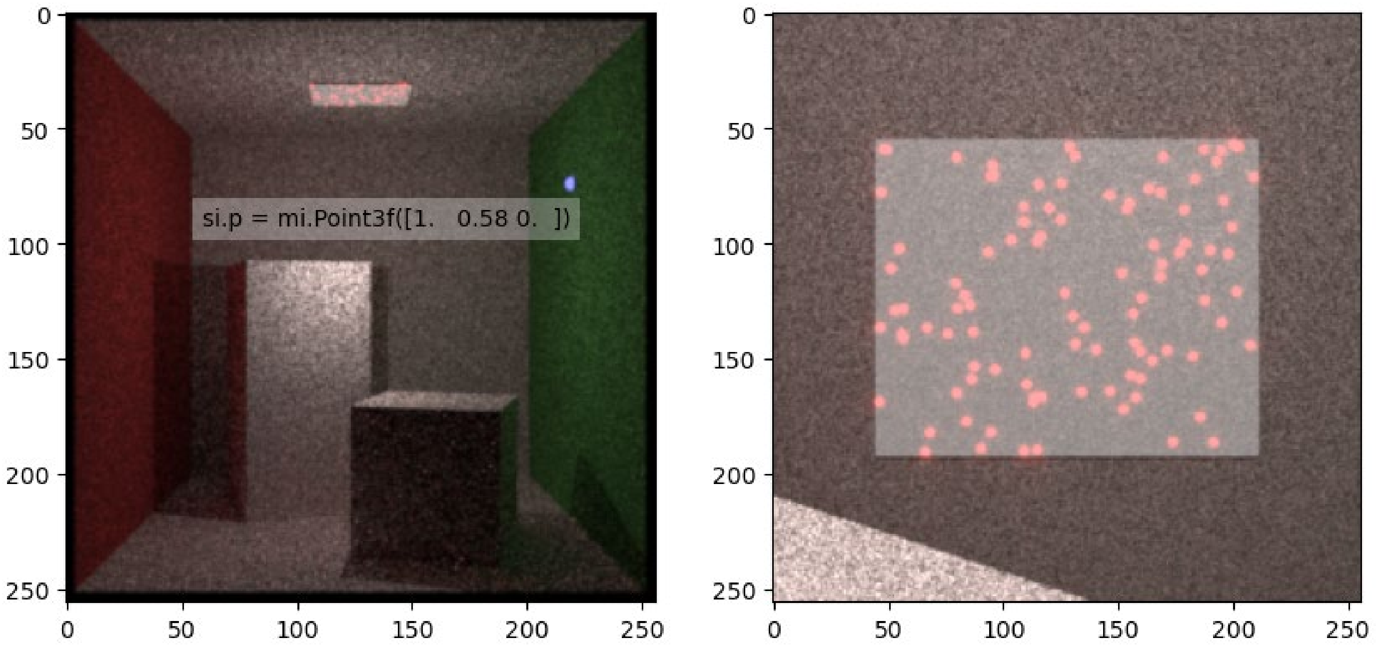
```
class: mi.scene
    # Methods
    def sample_emitter_direction():
    def eval_emitter_direction():
    def pdf_emitter_direction():
```



HW Ver. 0.4: Emitter sapmling

1. ray = sensor.sample_ray()
2. si = scene.ray_intersect(ray)
3. ds, em_weight = scene.sample_emitter_direction(si)

```
for i in range(N):
    scene.sample_emitter_direction(si, sampler.next_2d())
```



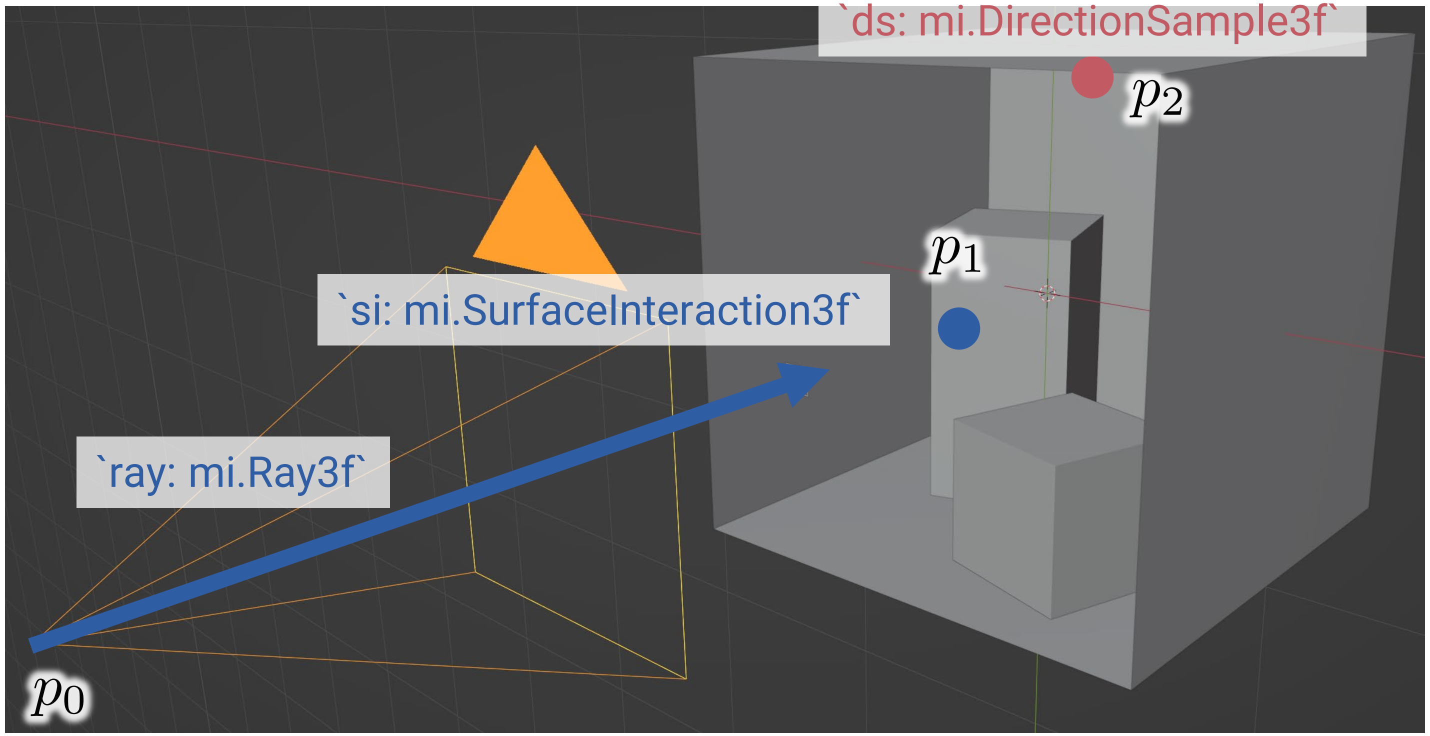
class: mi.scene

    # Methods
    def sample_emitter_direction():
        (occlusion test option argument)
        (see tutorial4)
    def eval_emitter_direction():
    def pdf_emitter_direction():

HW Ver. 0.4: Emitter sapmling (light sampling)

1. ray = sensor.sample_ray()
2. si = scene.ray_intersect(ray)
3. ds, em_weight = scene.sample_emitter_direction(si)
4. Evaluate $\hat{L}$
   - $\hat{L} = L_{e,10} + \hat{L}_{Emitter,210}$
   - $\hat{L}_{Emitter,210} = \frac{f^{\perp}_{s,210} L_{e,21}}{p_{Emitter}(\widehat{\omega}_{12})}$

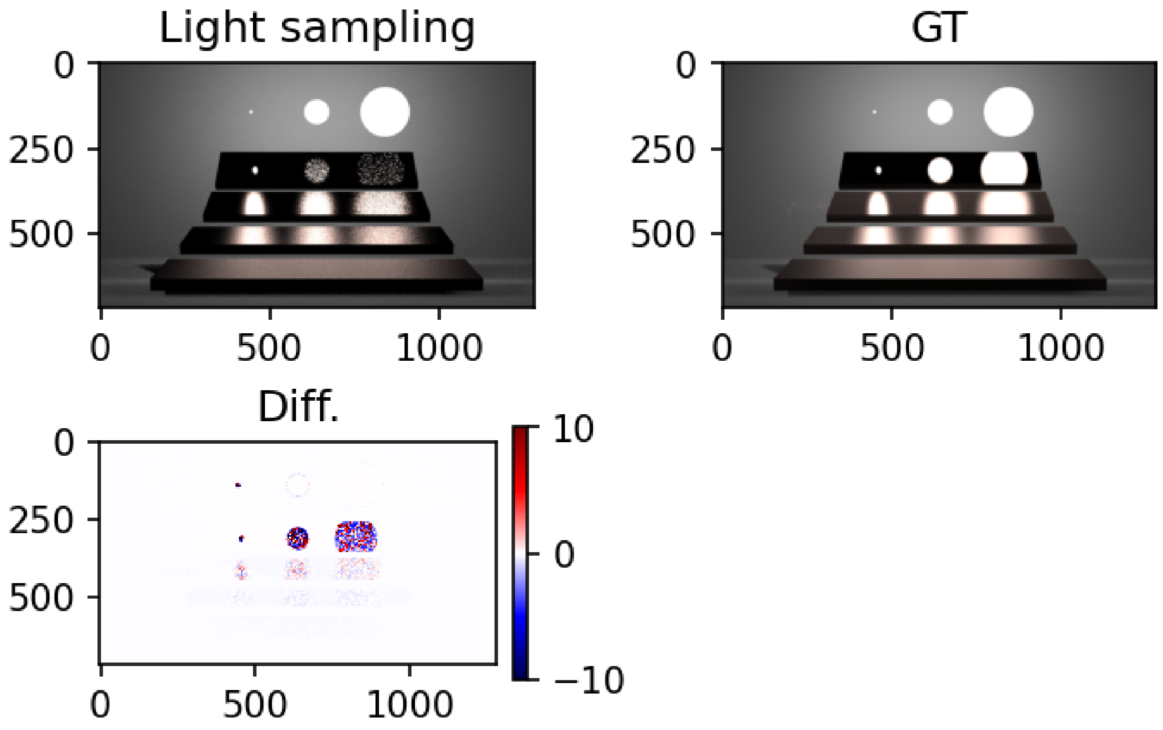
em_weight



HW Ver. 0.4: Emitter sapmling (light sampling)

1. ray = sensor.sample_ray()
2. si = scene.ray_intersect(ray)
3. ds, em_weight = scene.sample_emitter_direction(si)
4. Evaluate $\hat{L}$
   - $\hat{L} = L_{e,10} + \hat{L}_{Emitter,210}$
   - $\hat{L}_{Emitter,210} = \frac{f^{\perp}_{s,210} L_{e,21}}{p_{Emitter}(\hat{\omega}_{12})}$

     em_weight



HW Ver. 0.4: Emitter sapmling (light sampling)

Where does noise come from?

Resulting radiance $\hat{L}$ has high variance

← $f_s(p_1, \widehat{\omega}_{21}, \widehat{\omega}_{20})$ has high variance

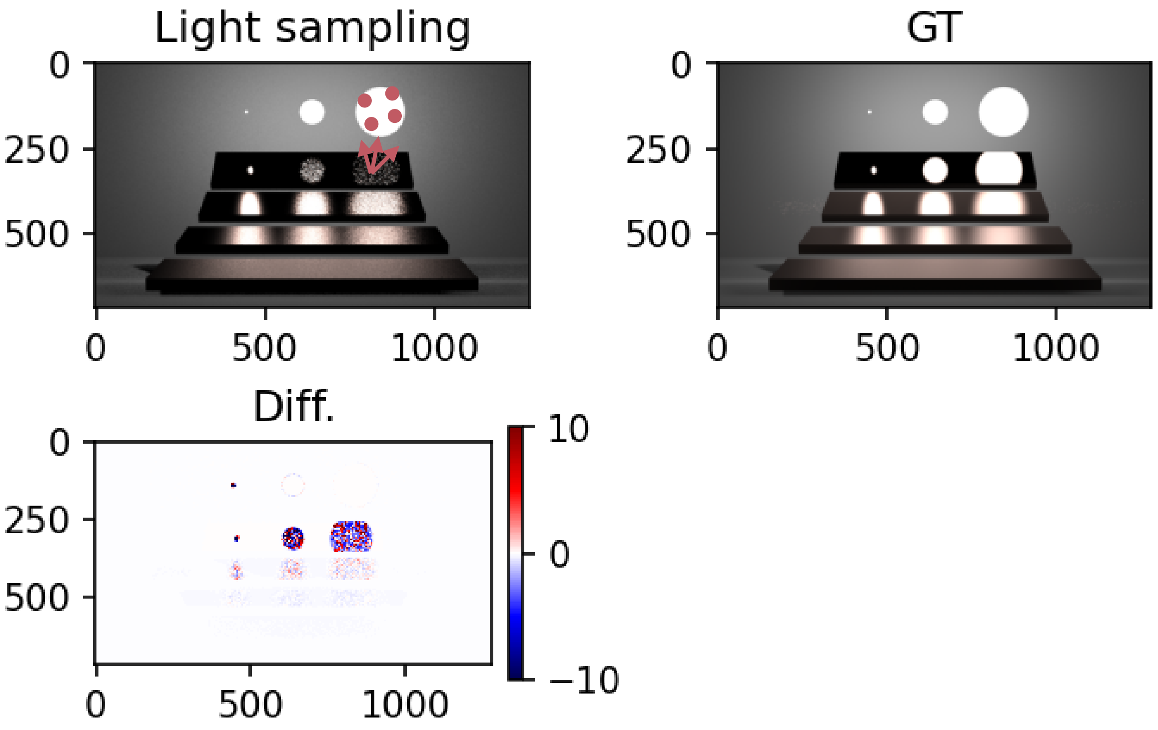
← p_2 = scene.sample_emitter_direction() rarely hits a (major part of) BSDF lobe

How will amount of noise change if BSDFs become more specular?

Variance (noise) ↑

How about ideal specular BSDF?

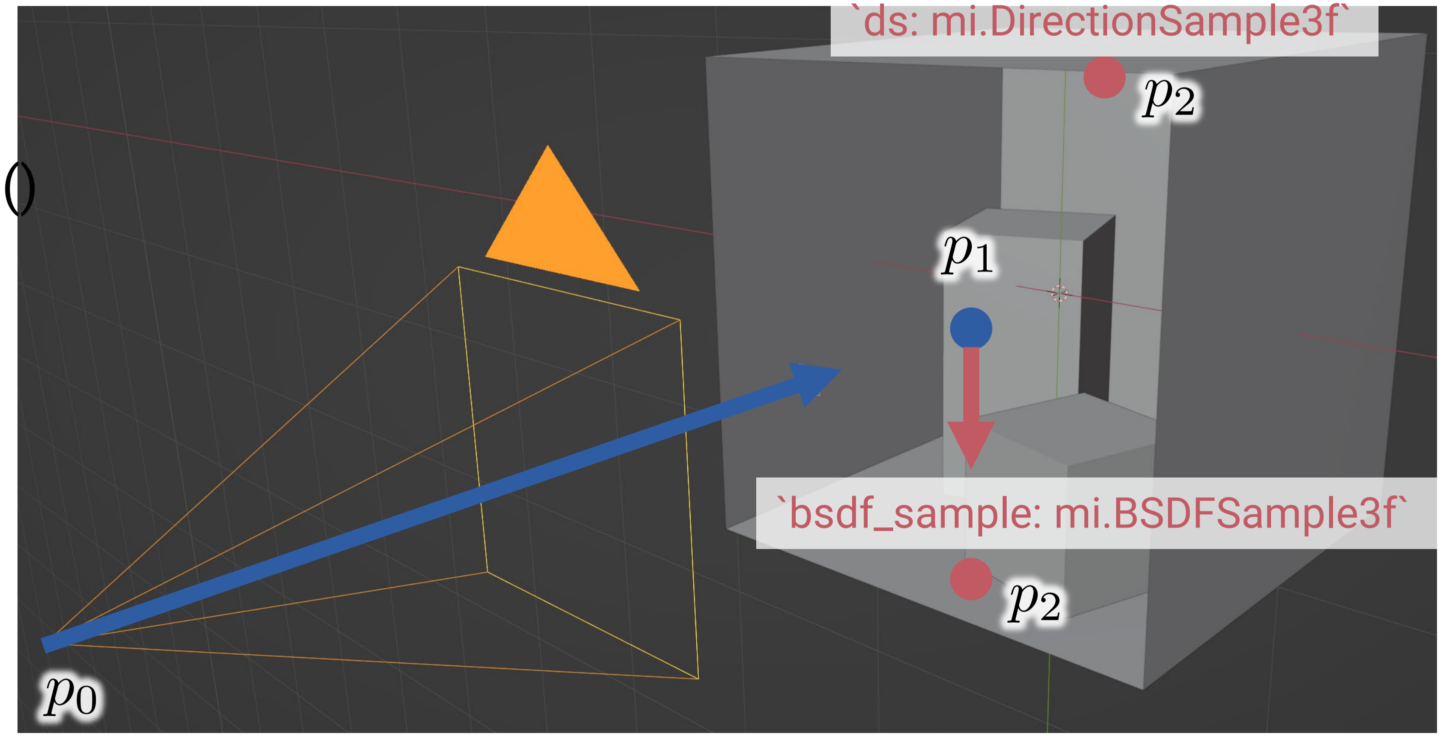
Biased (inaccurate)

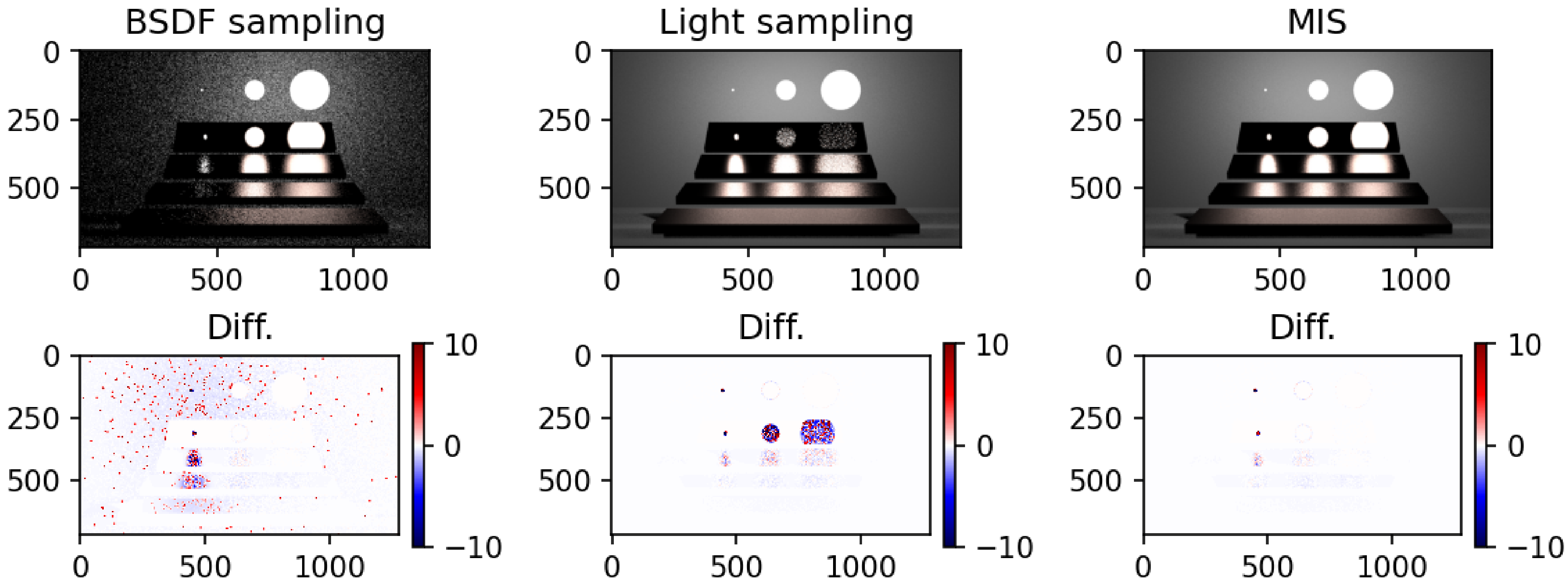


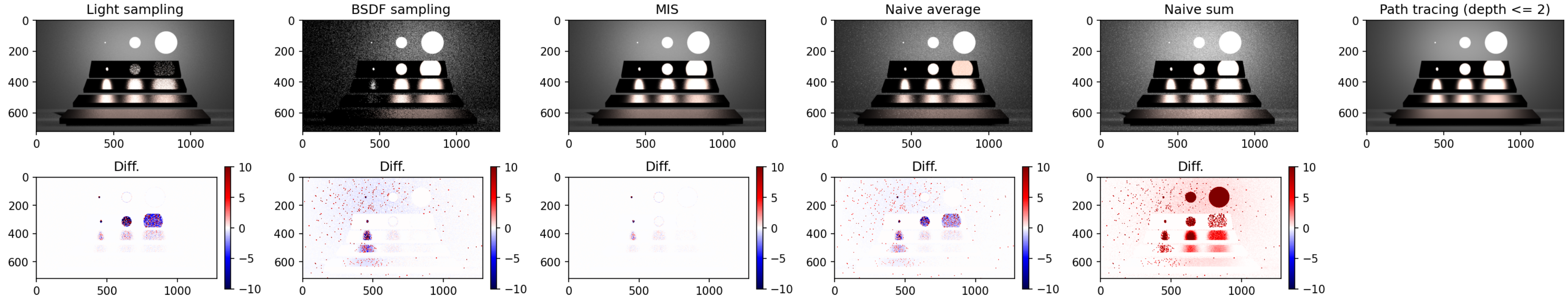
HW Ver. 0.5: Multiple importance sampling

1. ray = sensor.sample_ray()
2. si = scene.ray_intersect(ray)
3. bsdf = si.bsdf()
4. bsdf_sample, bsdf_weight
   = bsdf.sample()
5. ds, em_weight
   = scene.sample_emitter_direction()
6. Evaluate $\hat{L}$
   - $\hat{L}_{BSDF,210} = \dfrac{f_{s,210}^{\perp} L_{e,21}}{p_{BSDF}(\hat{\omega}_{12})}$
   - $\hat{L}_{Emitter,210} = \dfrac{f_{s,210}^{\perp} L_{e,21}}{p_{Emitter}(\hat{\omega}_{12})}$
   - $\hat{L} = L_{e,10} + w_B \hat{L}_{BSDF,210} + w_E \hat{L}_{Emitter,210}$

MIS weight



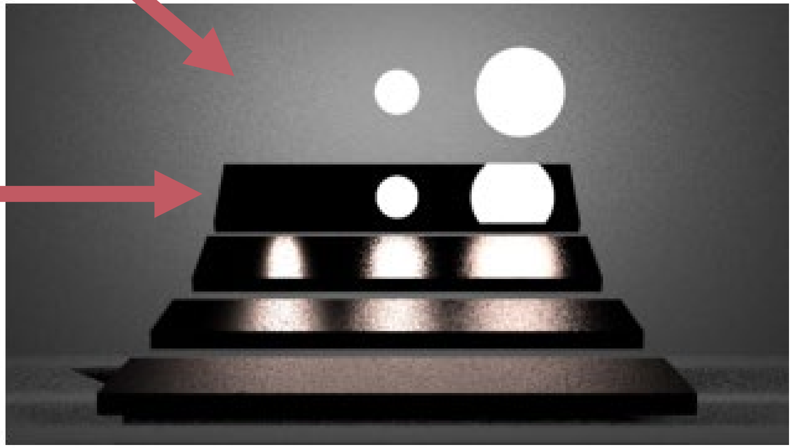

BSDF sampling
Light sampling
MIS
GT
Diff.
Diff.
Diff.
0
250
500
0
500
1000
10
0
−10


Light sampling
BSDF sampling
MIS
Naive average
Naive sum
Path tracing (depth <= 2)
Diff.
0
200
400
600
500
1000
10
5
−5
−10

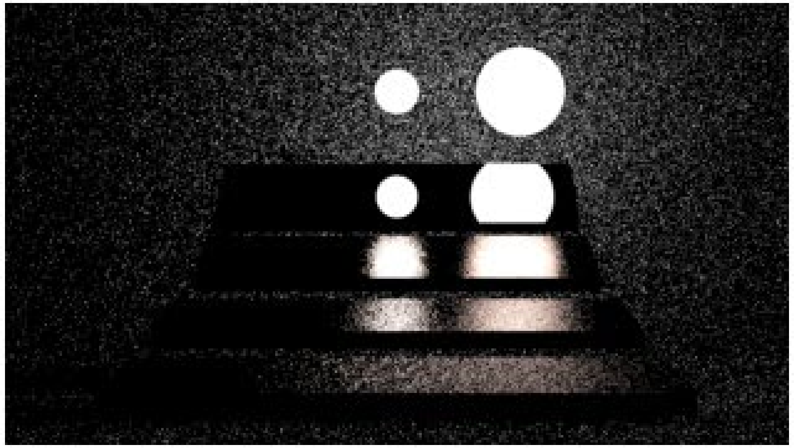
HW Ver. 0.6: + Dirac delta

Dirac delta source
(point light)

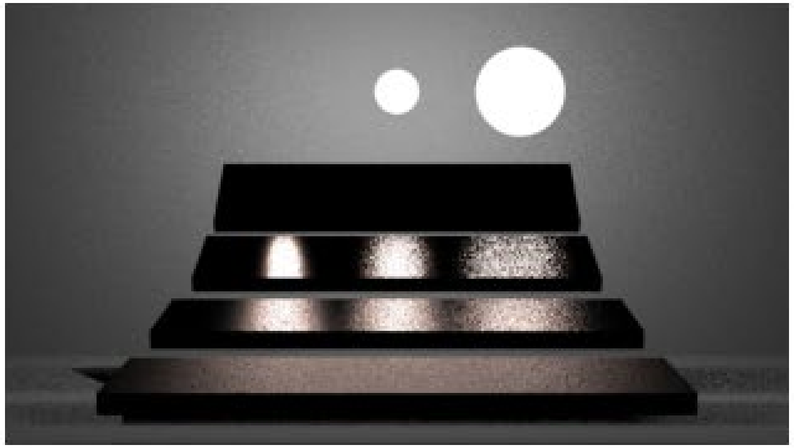
Dirac delta BSDF
(ideal specular)



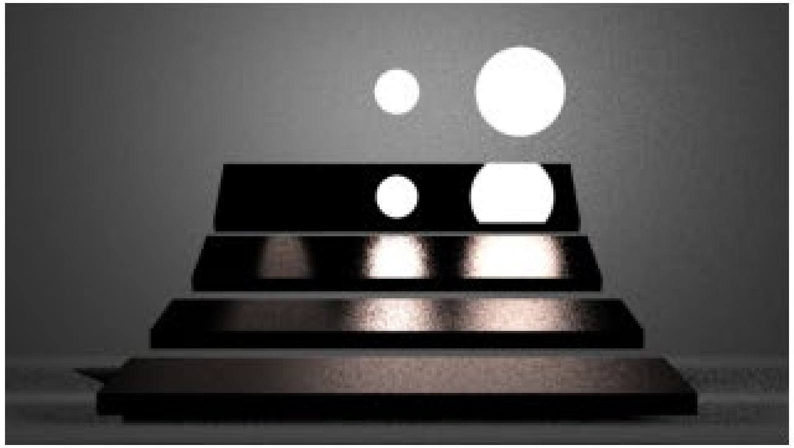
BSDF sampling



Emitter sampling



MIS (Ver. 0.5)



Monte Carlo Integration: revisit

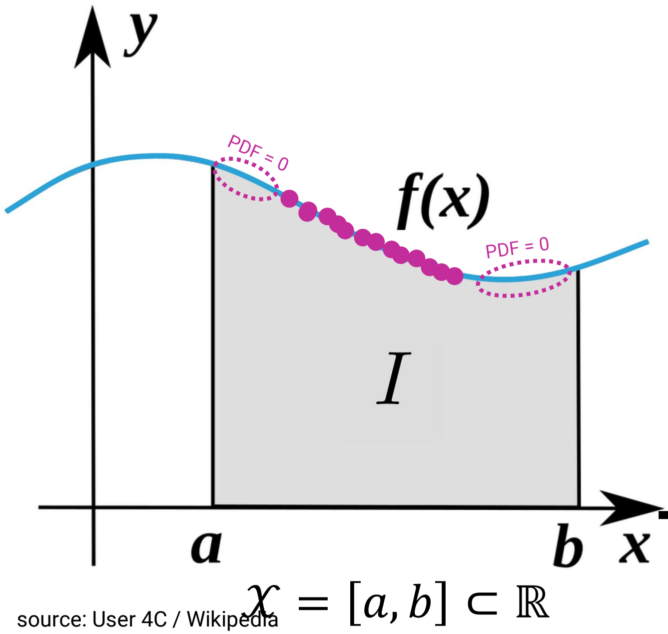
When is the MC integration correct (unbiased)?

$$\int_{\mathcal{X}} f(x)\mathrm{d}x = I = \mathbb{E}\left[\hat{I} = \frac{1}{N}\sum_{n=1}^{N}\frac{f(X_n)}{p_X(X_n)}\right]$$

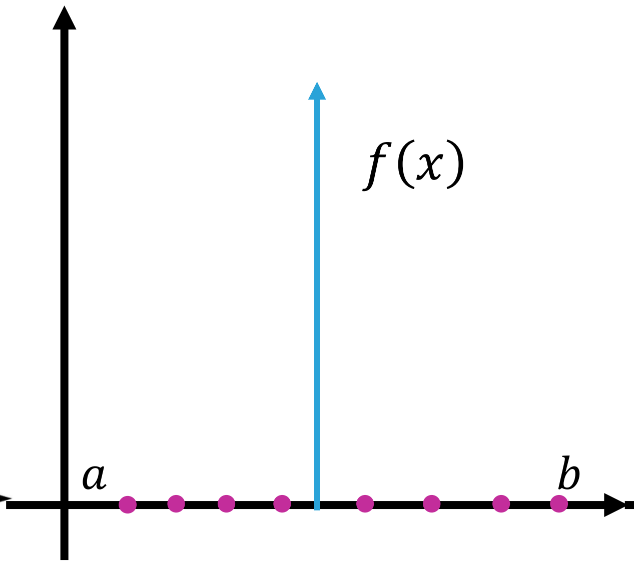
For any $x \in \mathcal{X}$, if $f(x) \neq 0$ then $p_X(x) > 0$

, and if $f(x) \propto \delta(x)$ then $p_X(x) \propto \delta(x)$

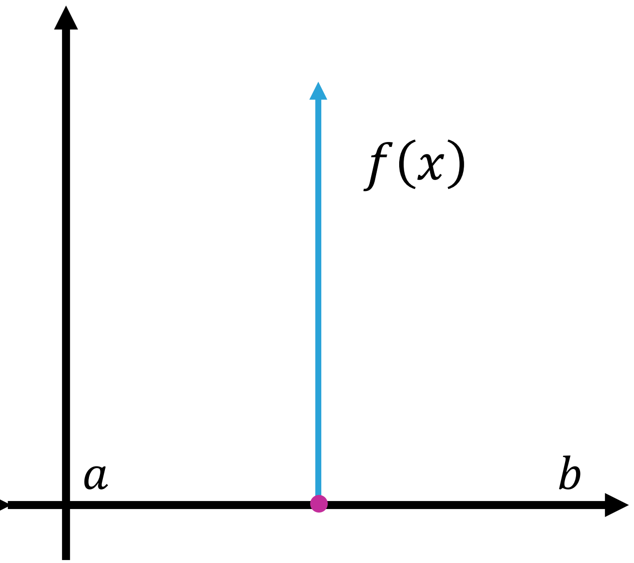
Failure case



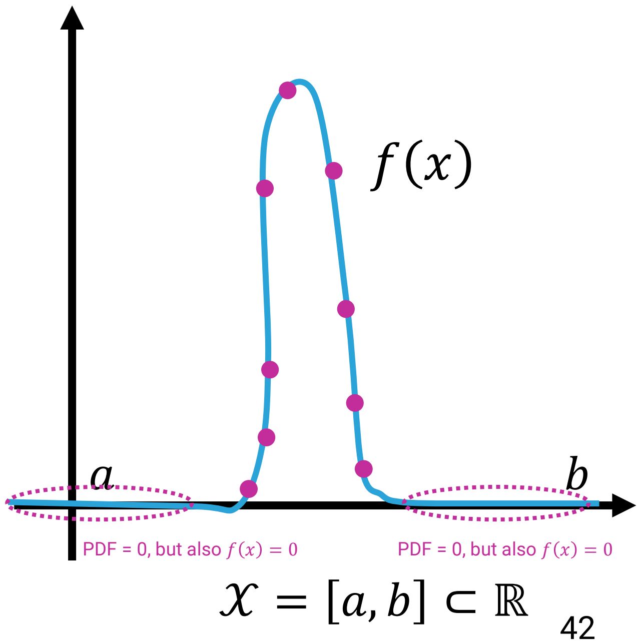
Failure case



Okay



Okay



[Adam Celarek]

HW Ver. 0.6: + Dirac delta

Rendering equation

$$L^{(\text{out})}(p, \widehat{\omega}_o) = L_e(p, \omega_{\hat{o}}) + \int_{\mathbb{S}^2} L^{(\text{out})}(\mathrm{r}(p, \widehat{\omega}_i), -\widehat{\omega}_i) f_s(p, \widehat{\omega}_i, \widehat{\omega}_o) |\hat{n} \cdot \widehat{\omega}_i| \mathrm{d}\widehat{\omega}_i$$

Recursive!

$$L(p, \widehat{\omega}_o) = L_e + \int_{\mathbb{S}^2} L f_s |\hat{n} \cdot \widehat{\omega}_i| \mathrm{d}\widehat{\omega}_i$$

Rendering equation

$$L^{(\mathrm{out})}(p, \widehat{\omega}_o) = L_e(p, \omega_{\hat{o}}) + \int_{\mathbb{S}^2} L^{(\mathrm{out})}(\mathrm{r}(p, \widehat{\omega}_i), -\widehat{\omega}_i) f_s(p, \widehat{\omega}_i, \widehat{\omega}_o) |\hat{n} \cdot \widehat{\omega}_i| \mathrm{d}\widehat{\omega}_i$$

Recursive!

$$L(p, \widehat{\omega}_o) = L_e + \int_{\mathbb{S}^2} \left[L_e' + \int_{\mathbb{S}^2} L f_s' |\hat{n} \cdot \widehat{\omega}_i'| \mathrm{d}\widehat{\omega}_i' \right] f_s |\hat{n} \cdot \widehat{\omega}_i| \mathrm{d}\widehat{\omega}_i$$

Rendering equation

$$L^{(\mathrm{out})}(p, \widehat{\omega}_o) = L_e(p, \omega_{\hat{o}}) + \int_{\mathbb{S}^2} L^{(\mathrm{out})}(\mathrm{r}(p, \widehat{\omega}_i), -\widehat{\omega}_i) f_s(p, \widehat{\omega}_i, \widehat{\omega}_o) |\hat{n} \cdot \widehat{\omega}_i| \mathrm{d}\widehat{\omega}_i$$

Recursive!

$$L(p, \widehat{\omega}_o) = L_e + \int_{\mathbb{S}^2} \left[L_e' + \int_{\mathbb{S}^2} \left[L_e'' + \int_{\mathbb{S}^2} L f_s'' |\hat{n} \cdot \widehat{\omega}_i''| \mathrm{d}\widehat{\omega}_i'' \right] f_s' |\hat{n} \cdot \widehat{\omega}_i'| \mathrm{d}\widehat{\omega}_i' \right] f_s |\hat{n} \cdot \widehat{\omega}_i| \mathrm{d}\widehat{\omega}_i$$

$$\cdots$$

$$\begin{aligned} L(p, \widehat{\omega}_o) = {} & L_e \\ & + \int_{\mathbb{S}^2} L_e' f_s |\hat{n} \cdot \widehat{\omega}_i| \mathrm{d}\widehat{\omega}_i \\ & + \int_{\mathbb{S}^2} \left[\int_{\mathbb{S}^2} L_e'' f_s' |\hat{n} \cdot \widehat{\omega}_i'| \mathrm{d}\widehat{\omega}_i' \right] f_s |\hat{n} \cdot \widehat{\omega}_i| \mathrm{d}\widehat{\omega}_i \\ & + \int_{\mathbb{S}^2} \left[\int_{\mathbb{S}^2} \left[\int_{\mathbb{S}^2} L_e''' f_s'' |\hat{n} \cdot \widehat{\omega}_i''| \mathrm{d}\widehat{\omega}_i'' \right] f_s' |\hat{n} \cdot \widehat{\omega}_i'| \mathrm{d}\widehat{\omega}_i' \right] f_s |\hat{n} \cdot \widehat{\omega}_i| \mathrm{d}\widehat{\omega}_i \end{aligned}$$

Rendering equation

Doesn't this series diverge?

Simple analogy, rather than rigorous functional analysis

For equation & infinite series for a simple real number

$$L = L_e + f_s L$$

$$L = \frac{L_e}{1 - f_s} = L_e \sum_{n=0}^{\infty} f_s^n \text{, if } |f_s| < 1$$

energy conservation property makes it holds

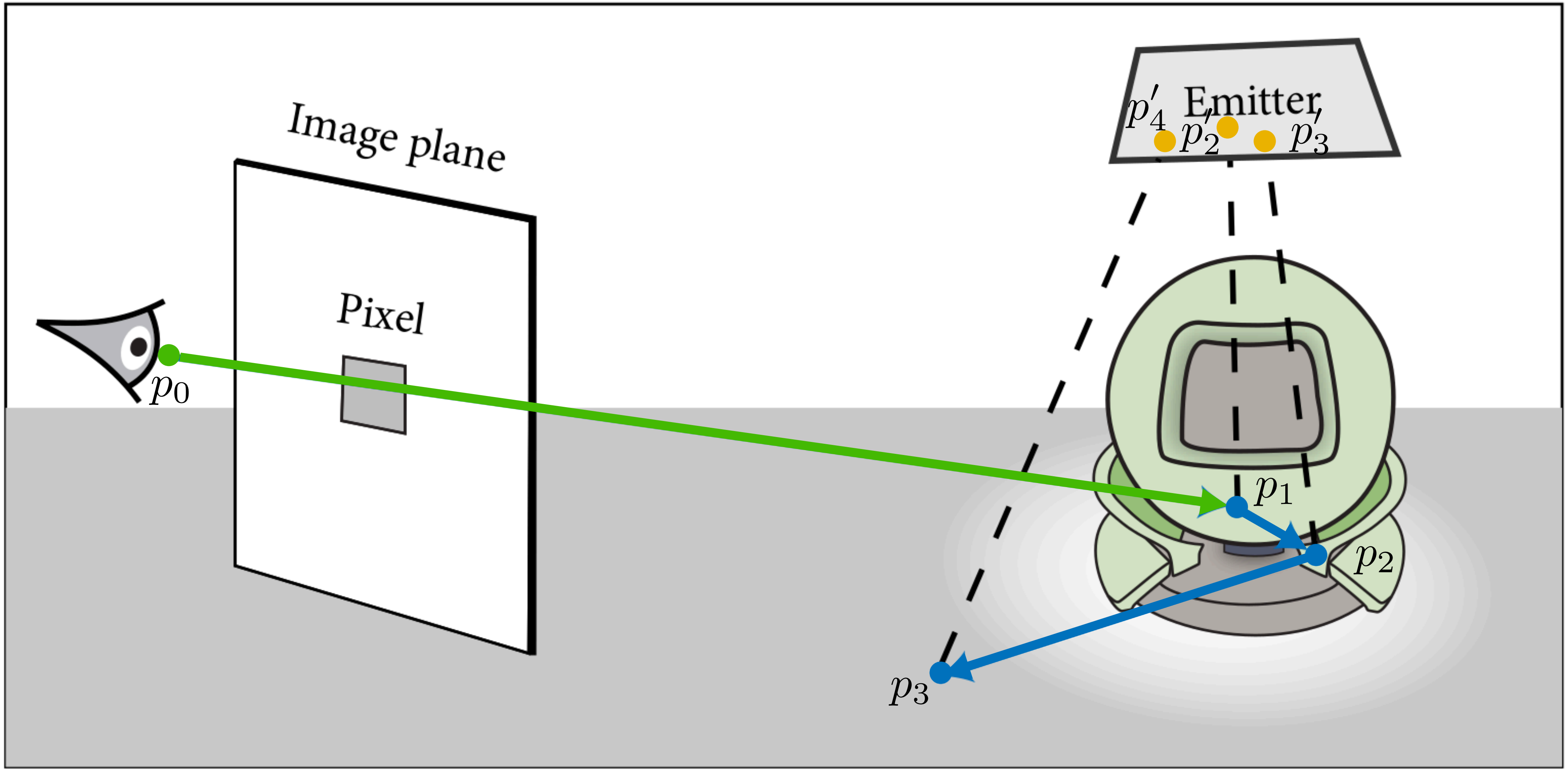
$$\begin{aligned} L(p, \hat{\omega}_o) &= L_e \\ &+ \int_{\mathbb{S}^2} L'_e f_s |\hat{n} \cdot \hat{\omega}_i| \mathrm{d}\hat{\omega}_i \\ &+ \int_{\mathbb{S}^2} \left[\int_{\mathbb{S}^2} L''_e f'_s |\hat{n} \cdot \hat{\omega}'_i| \mathrm{d}\hat{\omega}'_i \right] f_s |\hat{n} \cdot \hat{\omega}_i| \mathrm{d}\hat{\omega}_i \\ &+ \int_{\mathbb{S}^2} \left[\int_{\mathbb{S}^2} \left[\int_{\mathbb{S}^2} L'''_e f''_s |\hat{n} \cdot \hat{\omega}''_i| \mathrm{d}\hat{\omega}''_i \right] f'_s |\hat{n} \cdot \hat{\omega}'_i| \mathrm{d}\hat{\omega}'_i \right] f_s |\hat{n} \cdot \hat{\omega}_i| \mathrm{d}\hat{\omega}_i \end{aligned}$$

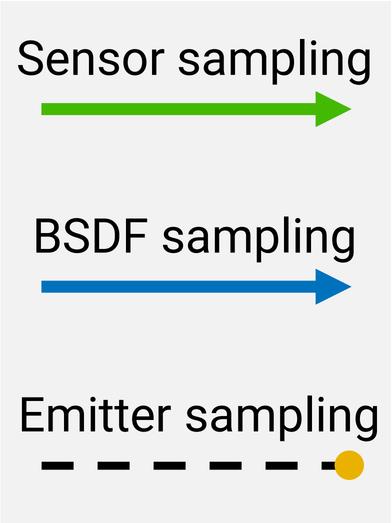
What image each line produces?

Rendering equation

$$
\begin{aligned}
L(p, \hat{\omega}_o) &= L_e \\
&+ \int_{\mathbb{S}^2} L'_e f_s |\hat{n} \cdot \hat{\omega}_i| \mathrm{d}\hat{\omega}_i \\
&+ \int_{\mathbb{S}^2} \left[\int_{\mathbb{S}^2} L''_e f'_s |\hat{n} \cdot \hat{\omega}'_i| \mathrm{d}\hat{\omega}'_i \right] f_s |\hat{n} \cdot \hat{\omega}_i| \mathrm{d}\hat{\omega}_i \\
&+ \int_{\mathbb{S}^2} \left[\int_{\mathbb{S}^2} \left[\int_{\mathbb{S}^2} L'''_e f''_s |\hat{n} \cdot \hat{\omega}''_i| \mathrm{d}\hat{\omega}''_i \right] f'_s |\hat{n} \cdot \hat{\omega}'_i| \mathrm{d}\hat{\omega}'_i \right] f_s |\hat{n} \cdot \hat{\omega}_i| \mathrm{d}\hat{\omega}_i
\end{aligned}
$$

Rendering equation for direct illumination





[https://mitsuba.readthedocs.io/en/latest/_images/integrator_path_figure.png]

Rendering equation

$$
\begin{aligned}
L(p_1, \widehat{\omega}_{10}) = {} & L_{e,10} \\
& + \int_{\mathbb{S}^2} L_{e,21} f_{s,210}^{\perp} \mathrm{d}\widehat{\omega}_{12} \\
& + \int_{\mathbb{S}^2} \left[\int_{\mathbb{S}^2} L_{e,32} f_{s,321}^{\perp} \mathrm{d}\widehat{\omega}_i' \right] f_{s,210}^{\perp} \mathrm{d}\widehat{\omega}_{23} \\
& + \int_{\mathbb{S}^2} \left[\int_{\mathbb{S}^2} \left[\int_{\mathbb{S}^2} L_{e,43} f_{s,432}^{\perp} \mathrm{d}\widehat{\omega}_i'' \right] f_{s,321}^{\perp} \mathrm{d}\widehat{\omega}_i' \right] f_{s,210}^{\perp} \mathrm{d}\widehat{\omega}_{34}
\end{aligned}
$$

Rendering equation

$$
\begin{aligned}
L(p_1, \widehat{\omega}_{10}) = L_{e,10} \\
&+ \int_{\mathbb{S}^2} f_{s,210}^{\perp} L_{e,21} \, \mathrm{d}\widehat{\omega}_{12} \\
&+ \int_{\mathbb{S}^2} f_{s,210}^{\perp} \left[\int_{\mathbb{S}^2} f_{s,321}^{\perp} L_{e,32} \, \mathrm{d}\widehat{\omega}_{23} \right] \mathrm{d}\widehat{\omega}_{12} \\
&+ \int_{\mathbb{S}^2} f_{s,210}^{\perp} \left[\int_{\mathbb{S}^2} f_{s,321}^{\perp} \left[\int_{\mathbb{S}^2} f_{s,432}^{\perp} L_{e,43} \, \mathrm{d}\widehat{\omega}_{34} \right] \mathrm{d}\widehat{\omega}_{23} \right] \mathrm{d}\widehat{\omega}_{12}
\end{aligned}
$$

Integration domain: all paths (tuples of vertices)

integrand: path contribution $f\left(p_0 \ldots . p_j\right)$

Rendering equation

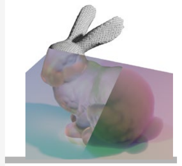
$$L(p_1, \widehat{\omega}_{10}) \approx L_{e,10} + \frac{f^{\perp}_{s,210} L_{e,21}}{p(W_{12})} + \frac{f^{\perp}_{s,210} f^{\perp}_{s,321} L_{e,32}}{p(W_{12})p(W_{23})} + \frac{f^{\perp}_{s,210} f^{\perp}_{s,321} f^{\perp}_{s,432} L_{e,43}}{p(W_{12})p(W_{23})p(W_{34})}$$

si=p_{i+1}

ds=p'_{i+2}

bsdf_sample.wo=$\widehat{\omega}_{i+1,i+2}$

```python
def sample(self,
           mode: dr.ADMode,
           scene: mi.Scene,
           sampler: mi.Sampler,
           ray: mi.Ray3f,
           **kwargs # Absorbs unused arguments
) -> Tuple[mi.Spectrum,
           mi.Bool, mi.Spectrum]:
    # Standard BSDF evaluation context for path tracing
    bsdf_ctx = mi.BSDFContext()

    # Copy input arguments to avoid mutating the caller's state
    depth = mi.UInt32(0)    # Depth of current vertex
    L = mi.Spectrum(0)      # Radiance accumulator
    β = mi.Spectrum(1)      # Path throughput weight

    # Variables caching information from the previous bounce
    prev_si         = dr.zeros(mi.SurfaceInteraction3f)
    prev_bsdf_pdf   = mi.Float(1.0)
    prev_bsdf_delta = mi.Bool(True)

    for i in range(self.max_depth):
        si = scene.ray_intersect(ray)

        # Get the BSDF, potentially computes texture-space differentials
        bsdf = si.bsdf(ray)

        # --------------------- Direct emission ---------------------

        # Compute MIS weight for emitter sample from previous bounce
        ds = mi.DirectionSample3f(scene, si=si, ref=prev_si)

        mis = mis_weight(
            prev_bsdf_pdf,
            scene.pdf_emitter_direction(prev_si, ds, ~prev_bsdf_delta)
        )

        Le = β * mis * ds.emitter.eval(si)

        # --------------------- Emitter sampling ---------------------
        ds, em_weight = scene.sample_emitter_direction(
            si, sampler.next_2d(), True)

        # Evaluate BSDF * cos(theta) differentiably
        wo = si.to_local(ds.d)
        bsdf_value_em, bsdf_pdf_em = bsdf.eval_pdf(bsdf_ctx, si, wo)
        mis_em = dr.select(ds.delta, 1, mis_weight(ds.pdf, bsdf_pdf_em))
        Lr_dir = β * mis_em * bsdf_value_em * em_weight

        # ------------------ Detached BSDF sampling ------------------
        bsdf_sample, bsdf_weight = bsdf.sample(bsdf_ctx, si,
                                               sampler.next_1d(),
                                               sampler.next_2d())
        L = L + Le + Lr_dir
        ray = si.spawn_ray(si.to_world(bsdf_sample.wo))
        β *= bsdf_weight

        prev_si = si
        prev_bsdf_pdf = bsdf_sample.pdf
        prev_bsdf_delta = mi.has_flag(bsdf_sample.sampled_type, mi.BSDFFlags.Delta)

    return (
        L,                   # Radiance/differential radiance
        dr.neq(depth, 0),    # Ray validity flag for alpha blending
        L                    # State for the differential phase
    )
```



Rendering equation

$$
\begin{aligned}
L(p_1, \widehat{\omega}_{10}) \approx L_{e,10} & \\
& + \frac{f^{\perp}_{s,210} L_{e,21}}{p(W_{12})} \\
& + \frac{f^{\perp}_{s,210} f^{\perp}_{s,321} L_{e,32}}{p(W_{12}) p(W_{23})} \\
& + \frac{f^{\perp}_{s,210} f^{\perp}_{s,321} f^{\perp}_{s,432} L_{e,43}}{p(W_{12}) p(W_{23}) p(W_{34})}
\end{aligned}
$$

Rendering equation

$$
\begin{aligned}
L(p_1, \widehat{\omega}_{10}) \approx L_{e,10} \\
&+ w_B(W_{12,B}) \frac{f^{\perp}_{s,210} L_{e,21}}{p_B(W_{12,B})} + w_E(W_{12,E}) \frac{f^{\perp}_{s,210} L_{e,21}}{p_E(W_{12,E})} \\
&+ w_B(W_{23,B}) \frac{f^{\perp}_{s,210} f^{\perp}_{s,321} L_{e,32}}{p_B(W_{12,B}) p_B(W_{23,B})} + w_E(W_{23,E}) \frac{f^{\perp}_{s,210} f^{\perp}_{s,321} L_{e,32}}{p_B(W_{12,B}) p_E(W_{23,E})} \\
&+ w_B(W_{34,B}) \frac{f^{\perp}_{s,210} f^{\perp}_{s,321} f^{\perp}_{s,432} L_{e,43}}{p_B(W_{12,B}) p_B(W_{23,B}) p_B(W_{34,B})} + w_E(W_{34,E}) \frac{f^{\perp}_{s,210} f^{\perp}_{s,321} f^{\perp}_{s,432} L_{e,43}}{p_B(W_{12,B}) p_B(W_{23,B}) p_E(W_{34,E})}
\end{aligned}
$$

BSDF sampling

Emitter sampling

Rendering equation

$$L(p_1, \widehat{\omega}_{10}) \approx L_{e,10}$$

$$+ w_B(W_{12,B}) \frac{f^{\perp}_{s,210} L_{e,21}}{p_B(W_{12,B})} + w_E(W_{12,E}) \frac{f^{\perp}_{s,210} L_{e,21}}{p_E(W_{12,E})}$$

$$+ \frac{f^{\perp}_{s,210}}{p_B(W_{12,B})} \left[w_B(W_{23,B}) \frac{f^{\perp}_{s,321} L_{e,32}}{p_B(W_{23,B})} + w_E(W_{23,E}) \frac{f^{\perp}_{s,321} L_{e,32}}{p_E(W_{23,E})} \right]$$

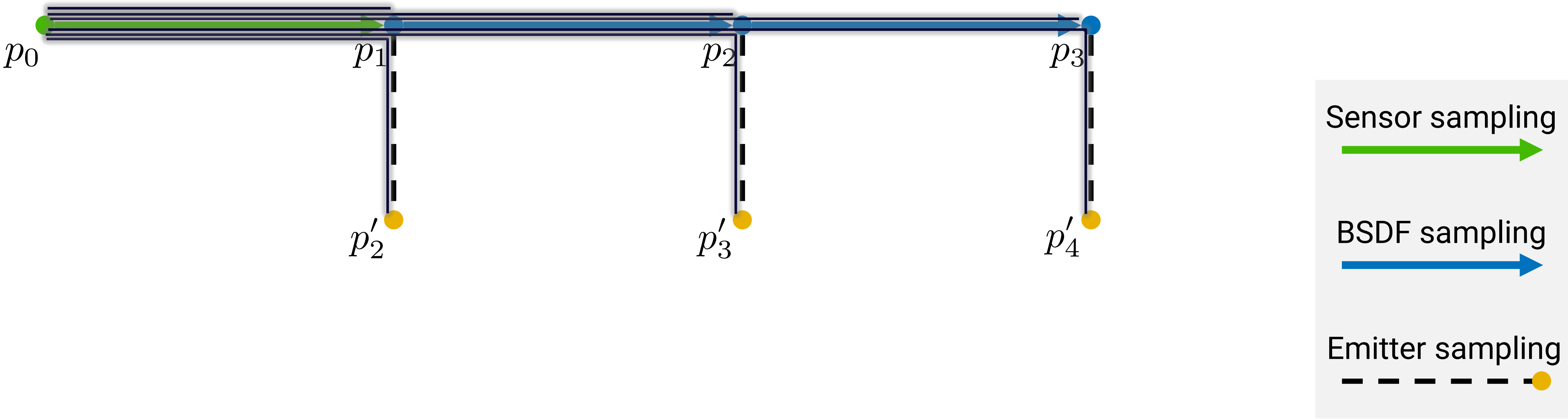
$$+ \frac{f^{\perp}_{s,210} f^{\perp}_{s,321}}{p_B(W_{12,B}) p_B(W_{23,B})} \left[w_B(W_{34,B}) \frac{f^{\perp}_{s,432} L_{e,43}}{p_B(W_{34,B})} + w_E(W_{34,E}) \frac{f^{\perp}_{s,432} L_{e,43}}{p_E(W_{34,E})} \right]$$

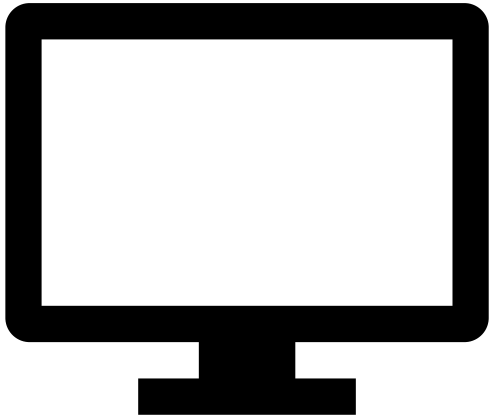
throughput variable β

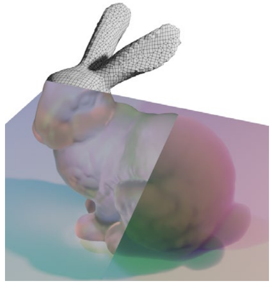
BSDF sampling

Emitter sampling

Rendering equation

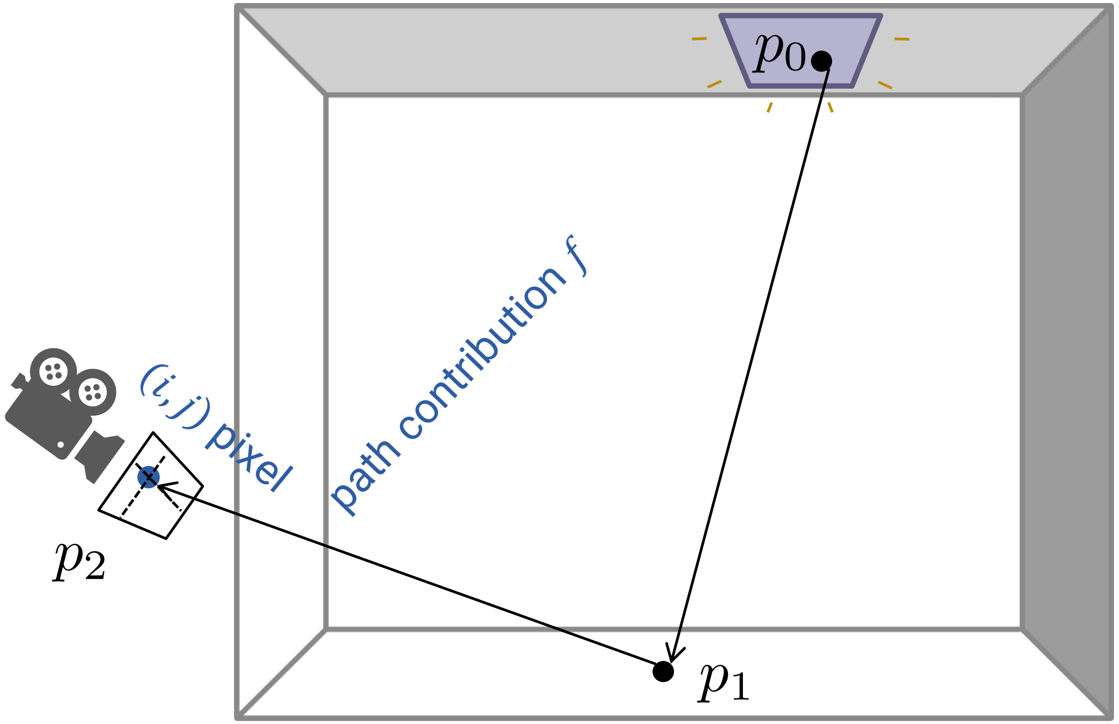






Preview: a simple path tracing

Rendering equation for direct illumination



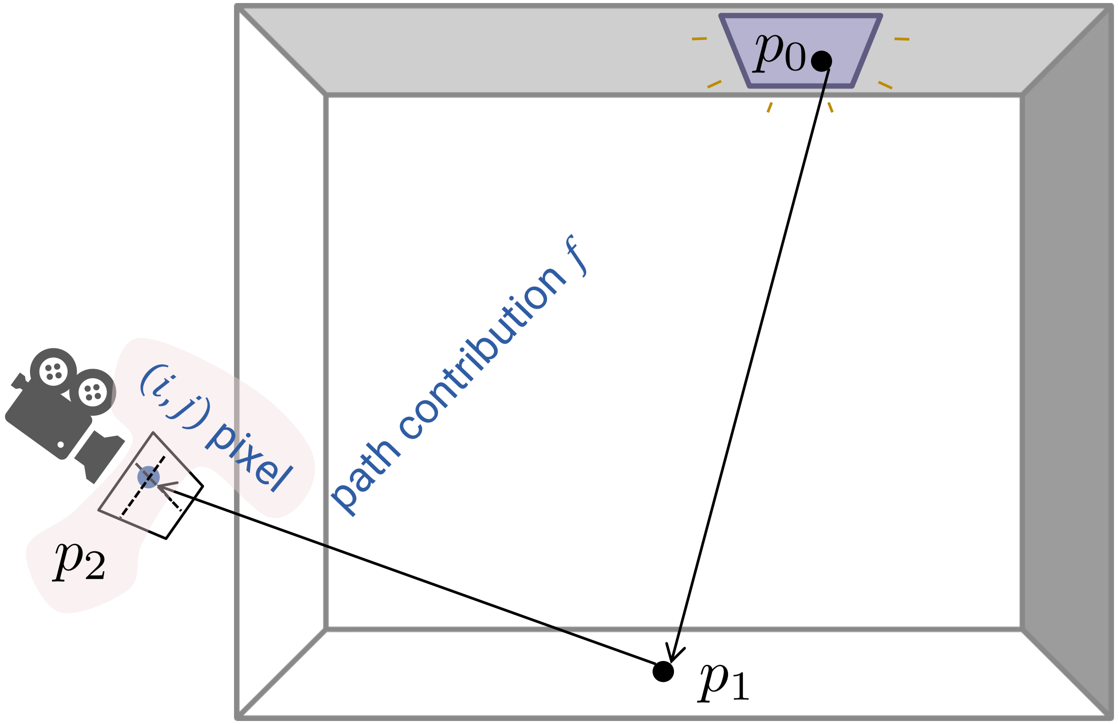
Terminology and convention:

- Notation / actual light: $p_0 \rightarrow p_1 \rightarrow p_2$
- Computation: $p_2 \rightarrow p_1 \rightarrow p_0$

- # of bounces < depth < # of vertices

1 **2** **3**

Rendering equation for direct illumination



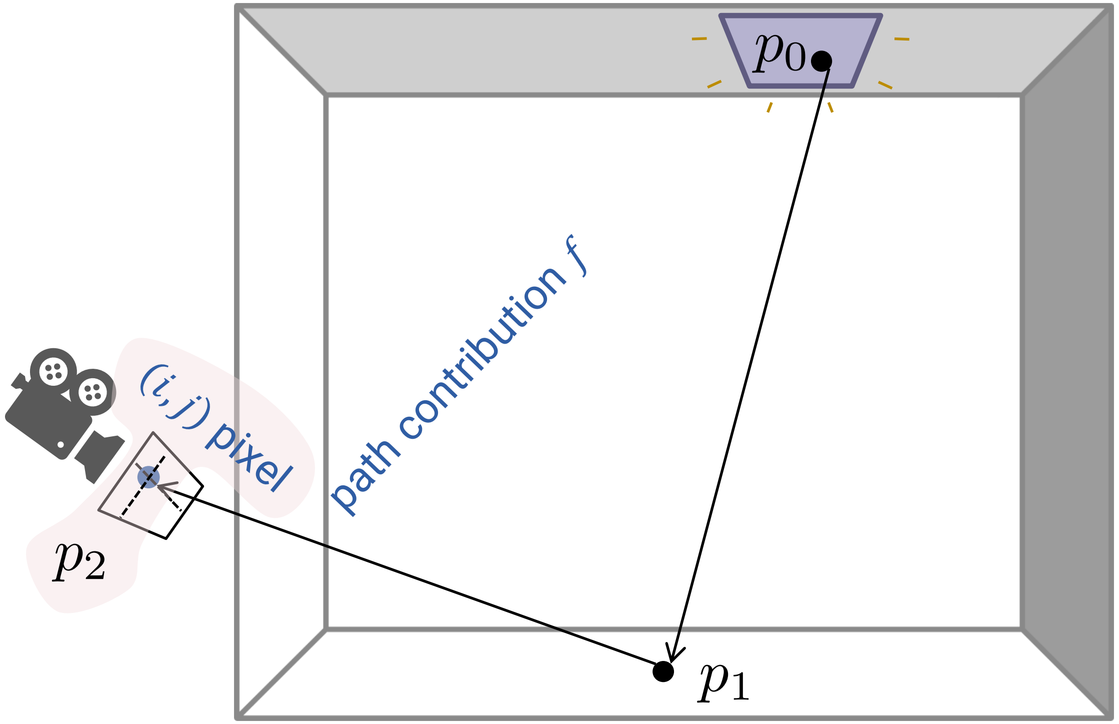
Notation:

- $\widehat{\omega}_{p_0 p_1} := \frac{p_1 - p_0}{\|p_1 - p_0\|}$

Final measurement:

$$I^{(i,j)} = \int_{\mathbb{S}} W_e^{(i,j)}(\hat{\omega})\, L(p_2, \hat{\omega})\, \mathrm{d}\hat{\omega}$$

Rendering equation for direct illumination



Notation:

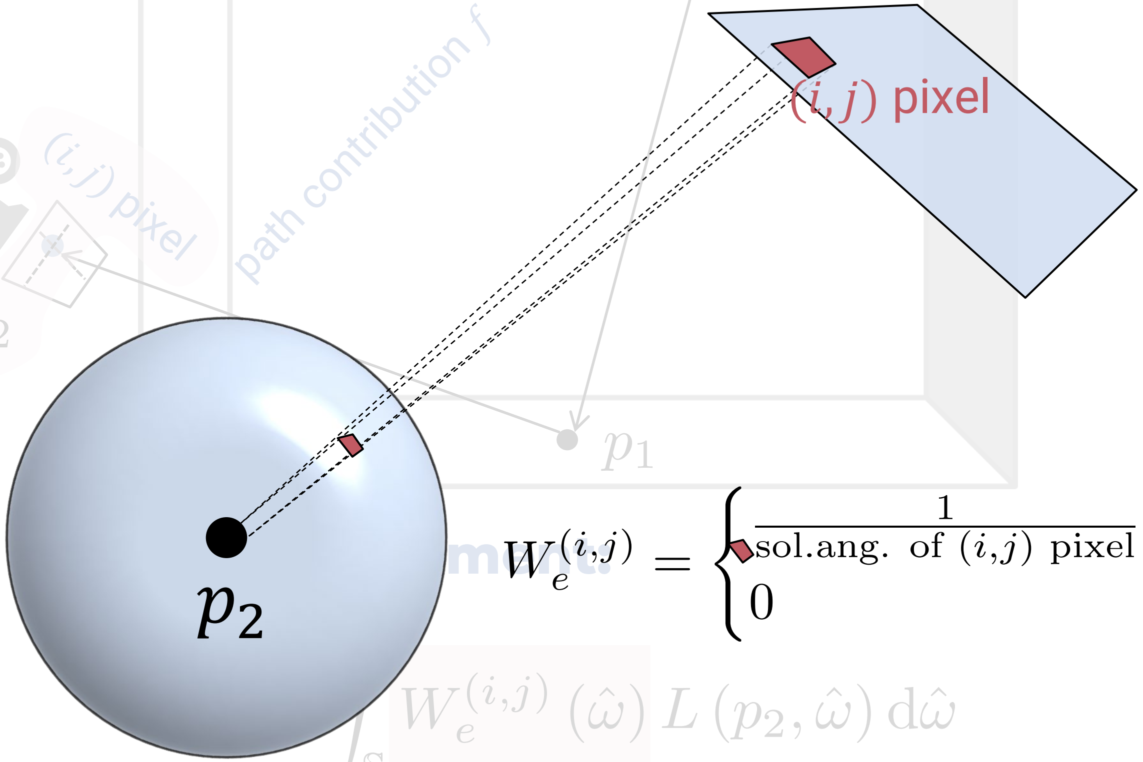
- $\widehat{\omega}_{p_0 p_1} := \frac{p_1 - p_0}{\|p_1 - p_0\|}$

Final measurement:

$$I^{(i,j)} = \int_{\mathbb{S}} W_e^{(i,j)}(\hat{\omega}) \, L(p_2, \hat{\omega}) \, \mathrm{d}\hat{\omega}$$

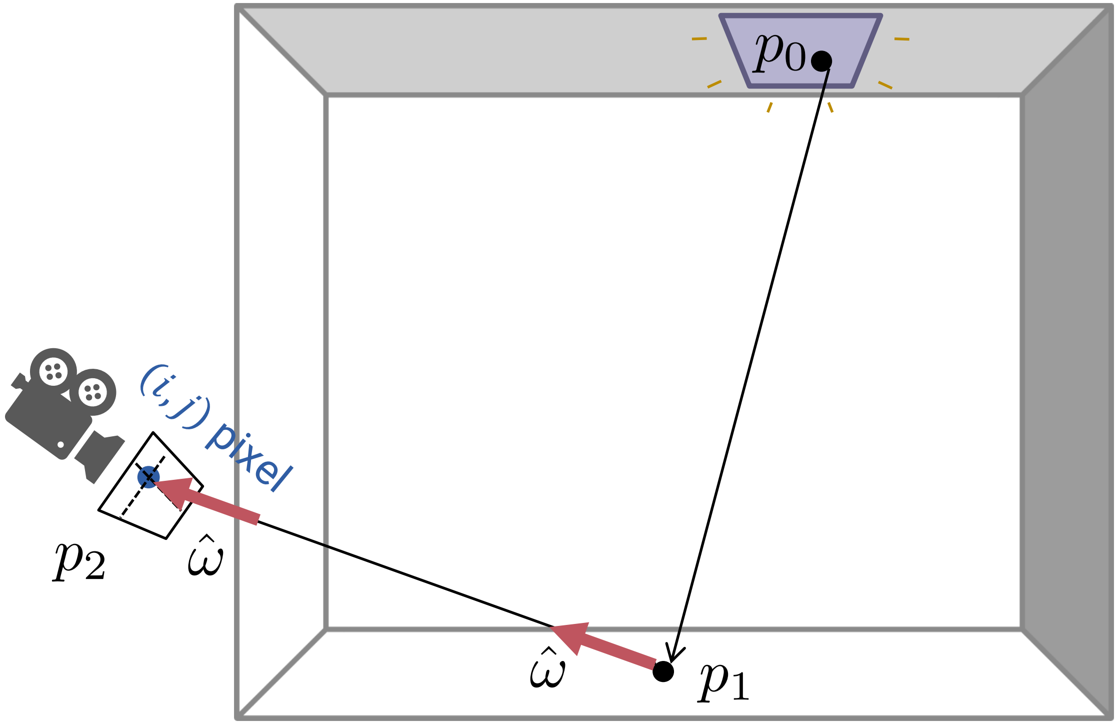
Rendering equation for direct illumination

Importance (pixel reconstruction filter) $W_e(p, \hat{\omega})$



$$W_e^{(i,j)} = \begin{cases} \dfrac{1}{\text{sol.ang. of } (i,j) \text{ pixel}} & \hat{\omega} \text{ in } (i,j) \text{ pixel} \\ 0 & \text{otherwise} \end{cases}$$

Rendering equation for direct illumination



Notation:

- $\widehat{\omega}_{p_0 p_1} \coloneqq \frac{p_1 - p_0}{\|p_1 - p_0\|}$
- r: $\mathbb{R}^3 \times \mathbb{S}^2 \rightarrow \mathbb{R}^3$
  - ray intersection

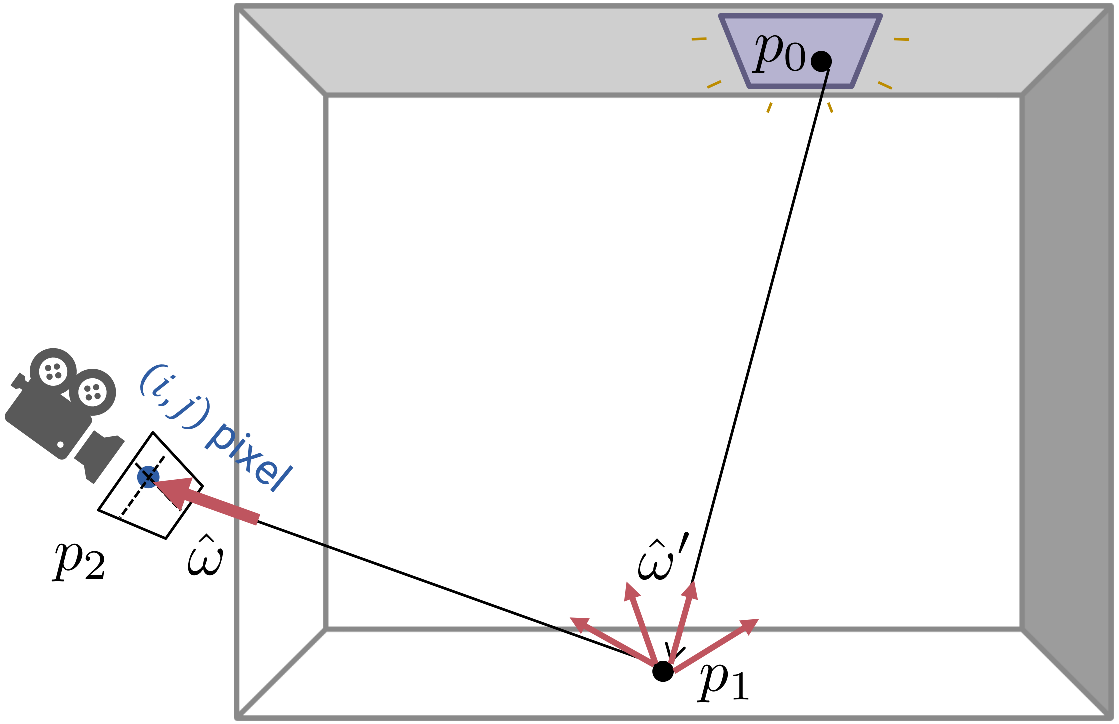
Final measurement:

$$I^{(i,j)} = \int_{\mathbb{S}} W_e^{(i,j)}(\hat{\omega})\, L(p_2, \hat{\omega})\, \mathrm{d}\hat{\omega}$$

$$L(p_2, \hat{\omega}) = L(p_1, \hat{\omega})$$

$p_1 = \mathrm{r}(p_2, -\hat{\omega})$

Rendering equation for direct illumination



Notation:

- $\widehat{\omega}_{p_0 p_1} := \frac{p_1 - p_0}{\|p_1 - p_0\|}$
- r: $\mathbb{R}^3 \times \mathbb{S}^2 \to \mathbb{R}^3$
  - ray intersection

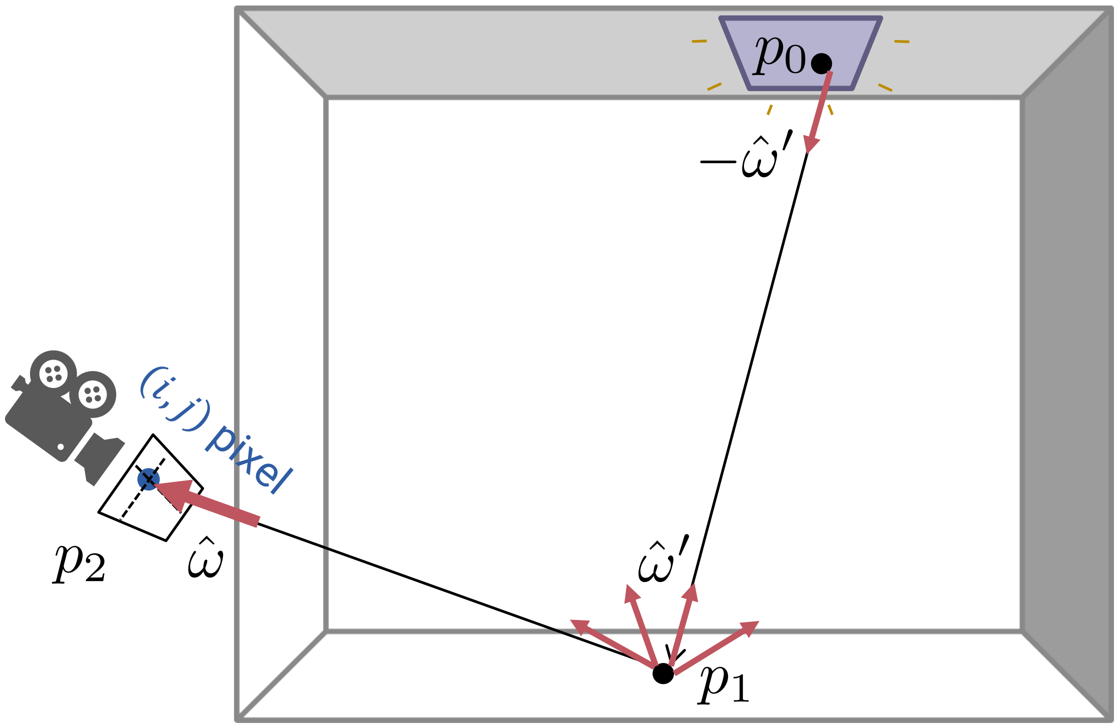
Final measurement:

$$I^{(i,j)} = \int_{\mathbb{S}} W_e^{(i,j)}(\hat{\omega})\, L(p_1, \hat{\omega})\, \mathrm{d}\hat{\omega}$$

$$L(p_1, \hat{\omega}) = L_e(p_1, \hat{\omega}) + \int_{\mathbb{S}} L^{(\mathrm{in})}(p_1, \hat{\omega}')\, \rho(p_1, \hat{\omega}', \hat{\omega})\, |\hat{n} \cdot \hat{\omega}'|\, \mathrm{d}\hat{\omega}'$$

$p_1 = \mathrm{r}(p_2, -\hat{\omega})$

Rendering equation for direct illumination



Notation:

- $\widehat{\omega}_{p_0 p_1} := \frac{p_1 - p_0}{\|p_1 - p_0\|}$
- r: $\mathbb{R}^3 \times \mathbb{S}^2 \rightarrow \mathbb{R}^3$
    - ray intersection

Final measurement:

$$I^{(i,j)} = \int_{\mathbb{S}} W_e^{(i,j)}(\hat{\omega})\, L(p_1, \hat{\omega})\, \mathrm{d}\hat{\omega}$$

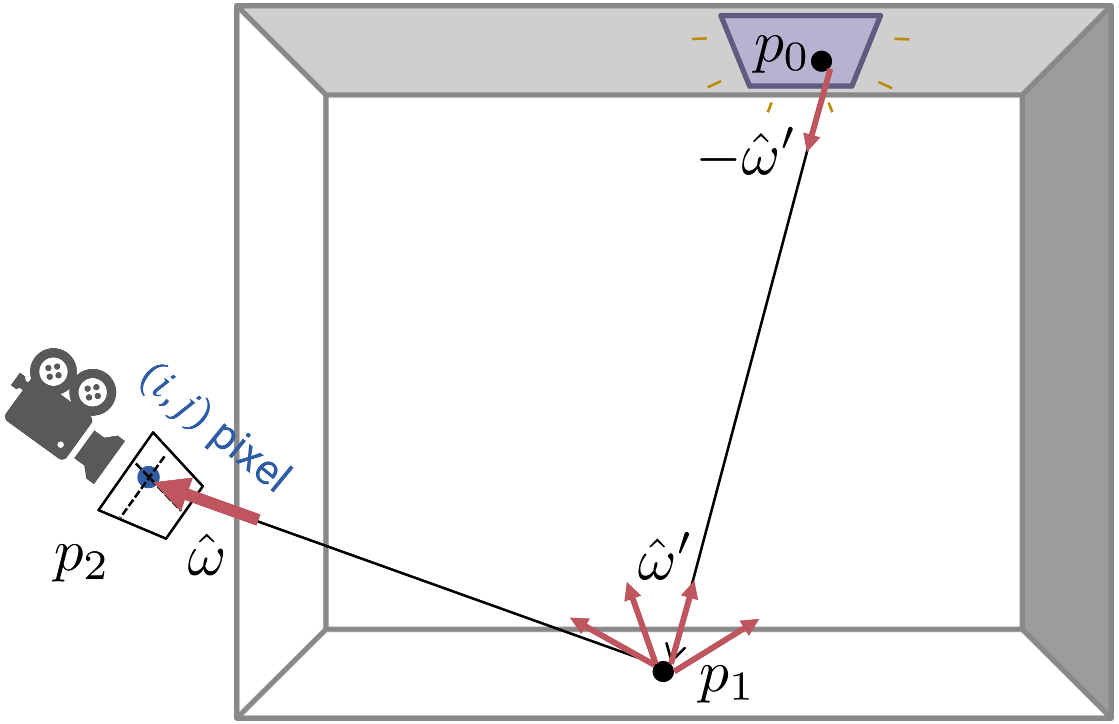
$p_1 = \mathrm{r}(p_2, -\hat{\omega})$

$$L(p_1, \hat{\omega}) = L_e(p_1, \hat{\omega}) + \int_{\mathbb{S}} L^{(\mathrm{in})}(p_1, \hat{\omega}')\, \rho(p_1, \hat{\omega}', \hat{\omega})\, |\hat{n} \cdot \hat{\omega}'|\, \mathrm{d}\hat{\omega}'$$

$p_0 = \mathrm{r}(p_1, \hat{\omega}')$

$$L^{(\mathrm{in})}(p_1, \hat{\omega}') = L(p_0 \ -\hat{\omega}')$$

Rendering equation for direct illumination



Notation:

- $\widehat{\omega}_{p_0 p_1} := \frac{p_1 - p_0}{\|p_1 - p_0\|}$
- r: $\mathbb{R}^3 \times \mathbb{S}^2 \to \mathbb{R}^3$
  - ray intersection

Final measurement:

$$I^{(i,j)} = \int_{\mathbb{S}} W_e^{(i,j)}(\hat{\omega})\, L(p_1, \hat{\omega})\, \mathrm{d}\hat{\omega}$$

$p_1 = \mathrm{r}(p_2, -\hat{\omega})$

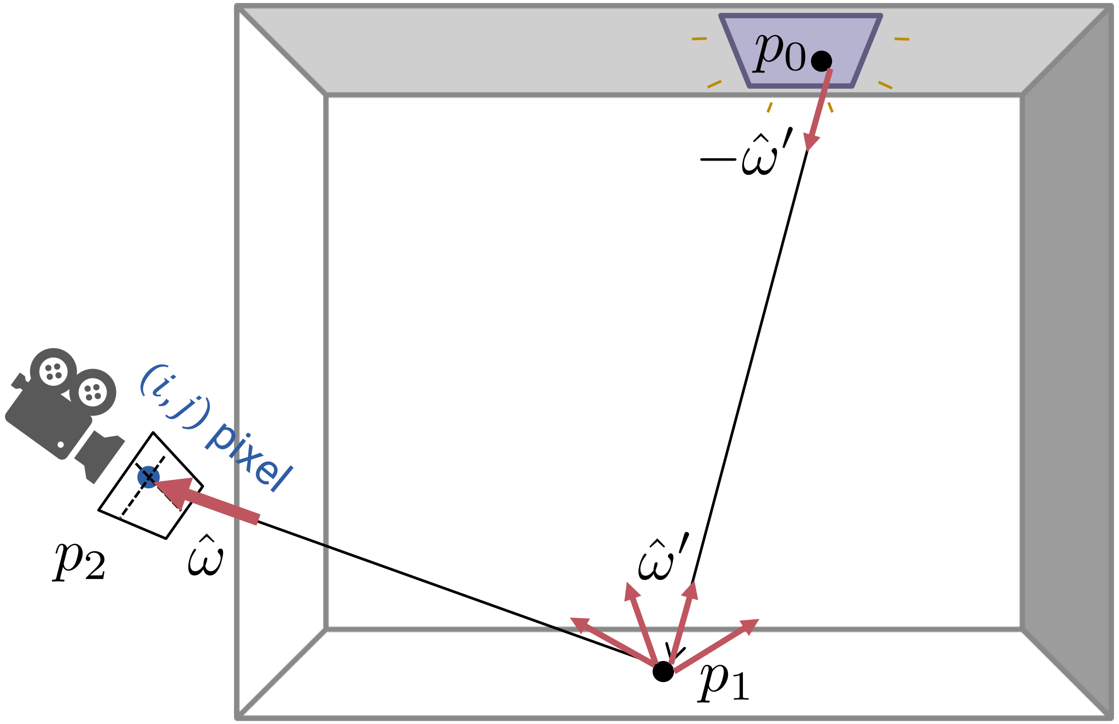
$$L(p_1, \hat{\omega}) = L_e(p_1, \hat{\omega}) + \int_{\mathbb{S}} L^{(\mathrm{in})}(p_1, \hat{\omega}')\, \rho(p_1, \hat{\omega}', \hat{\omega})\, |\hat{n} \cdot \hat{\omega}'|\, \mathrm{d}\hat{\omega}'$$

$p_0 = \mathrm{r}(p_1, \hat{\omega}')$

Direct illumination only

$$L(p_0, -\hat{\omega}') = L_e(p_0, -\hat{\omega}') + \cancel{\int_{\mathbb{S}} L^{(\mathrm{in})}(p_0, \hat{\omega}'')\, \rho(p_0, \hat{\omega}'', -\hat{\omega}')\, |\hat{n} \cdot \hat{\omega}''|\, \mathrm{d}\hat{\omega}''}$$

Rendering equation for direct illumination



Notation:

- $\widehat{\omega}_{p_0 p_1} := \frac{p_1 - p_0}{\|p_1 - p_0\|}$
- r: $\mathbb{R}^3 \times \mathbb{S}^2 \to \mathbb{R}^3$
  - ray intersection

Final measurement:

$$I^{(i,j)} = \int_{\mathbb{S}} W_e^{(i,j)}(\hat{\omega})\, L(p_1, \hat{\omega})\, \mathrm{d}\hat{\omega}$$

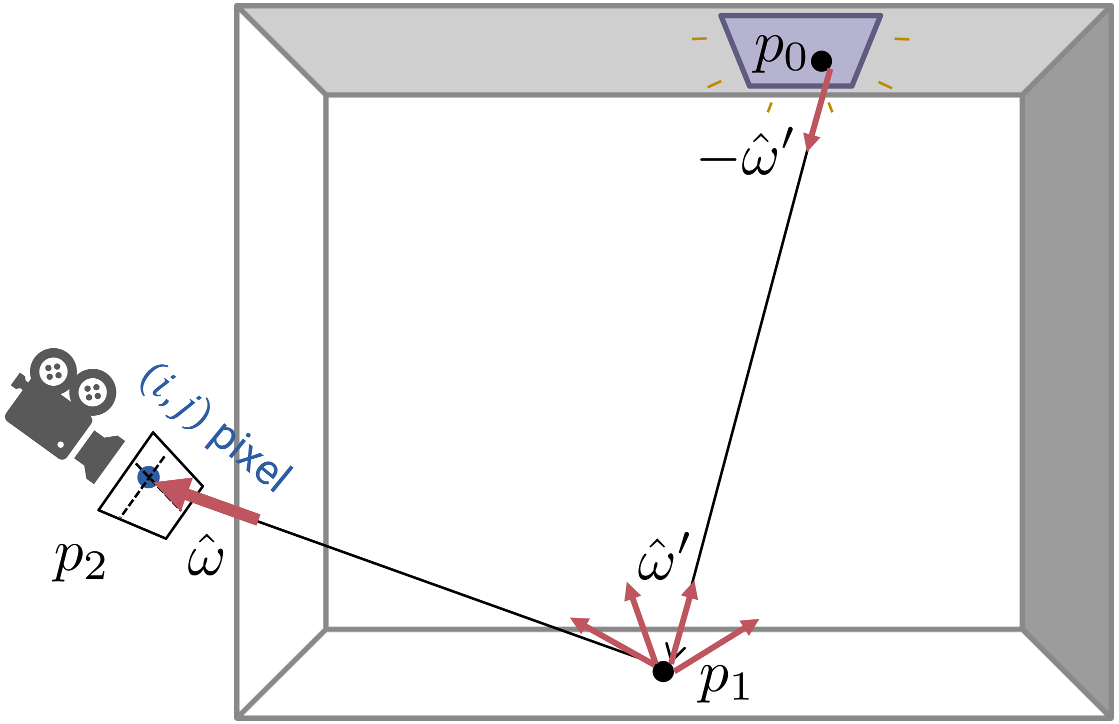
$p_1 = \mathrm{r}(p_2, -\hat{\omega})$

$$L(p_1, \hat{\omega}) = L_e(p_1, \hat{\omega}) + \int_{\mathbb{S}} L^{(\mathrm{in})}(p_1, \hat{\omega}')\, \rho(p_1, \hat{\omega}', \hat{\omega})\, |\hat{n} \cdot \hat{\omega}'|\, \mathrm{d}\hat{\omega}'$$

$p_0 = \mathrm{r}(p_1, \hat{\omega}')$

$$L(p_0, -\hat{\omega}') = L_e(p_0, -\hat{\omega}')$$

Rendering equation for direct illumination



Notation:

- $\widehat{\omega}_{p_0 p_1} \coloneqq \frac{p_1 - p_0}{\|p_1 - p_0\|}$
- r: $\mathbb{R}^3 \times \mathbb{S}^2 \rightarrow \mathbb{R}^3$
  - ray intersection

Final measurement:

$$I^{(i,j)} = \int_{\mathbb{S}} W_e^{(i,j)}(\hat{\omega})\, L(p_1, \hat{\omega})\, \mathrm{d}\hat{\omega}$$

$$L(p_1, \hat{\omega}) = L_e(p_1, \hat{\omega}) + \int_{\mathbb{S}} L_e(p_0, -\hat{\omega}')\, \rho(p_1, \hat{\omega}', \hat{\omega})\, |\hat{n} \cdot \hat{\omega}'|\, \mathrm{d}\hat{\omega}'$$

$p_1 = \mathrm{r}(p_2, -\hat{\omega})$
$p_0 = \mathrm{r}(p_1, \hat{\omega}')$

Ver 0.1: normal integrator

Why deterministic pseudo-random

1. Debugging
2. Several technique: [Path replay backprogation]

Terminology

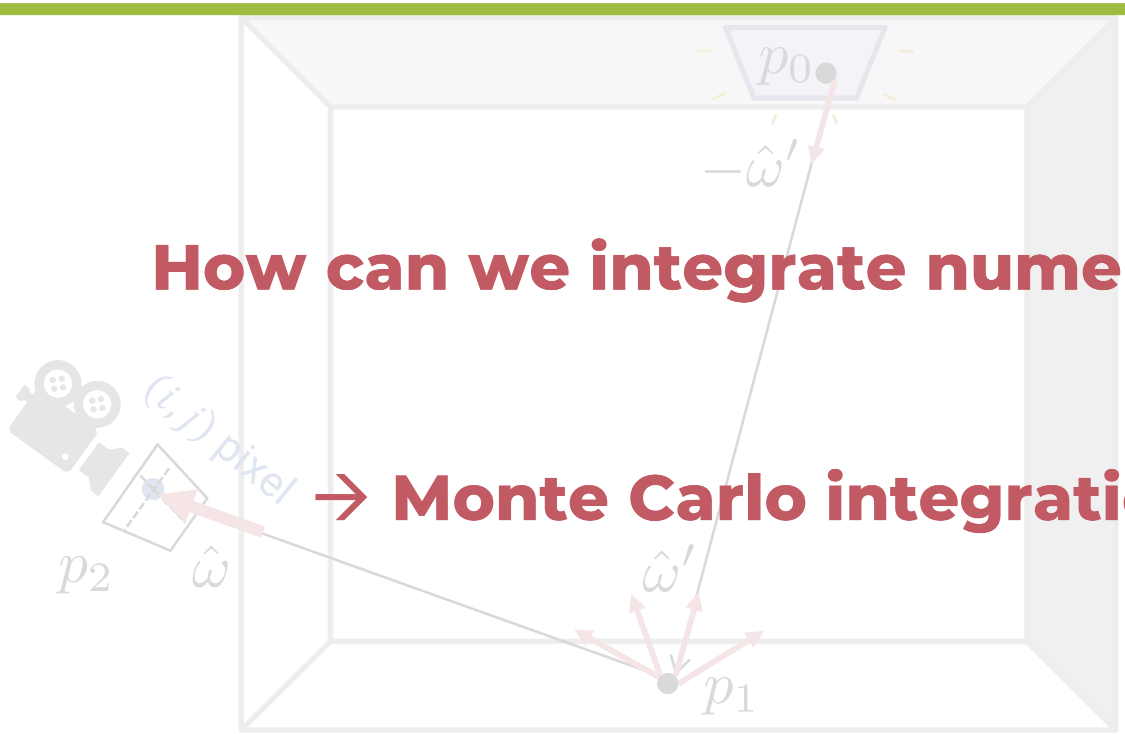
Vector

1. Math: addition, scalar multiplcation
2. Bitmap vs. vector
3. (e.g. data structure, programing language)
   - Array, list, …
   - ~ SIMD

Renderer implementation convention

- Random process method
  - Argument uniform dist. Generated

Rendering equation for direct illumination



Notation:

- $\hat{\omega}_{p_0 p_1} := \frac{p_1 - p_0}{\|p_1 - p_0\|}$
- r: $\mathbb{R}^3 \times \mathbb{S}^2 \to \mathbb{R}^3$
  - ray intersection

How can we integrate numerically?

→ Monte Carlo integration

Final measurement:

$$I^{(i,j)} = \int_{\mathbb{S}} W_e^{(i,j)}(\hat{\omega})\, L(p_1, \hat{\omega})\, \mathrm{d}\hat{\omega}$$

$$L(p_1, \hat{\omega}) = L_e(p_1, \hat{\omega}) + \int_{\mathbb{S}} L_e(p_0, -\hat{\omega}')\, \rho(p_1, \hat{\omega}', \hat{\omega})\, |\hat{n} \cdot \hat{\omega}'|\, \mathrm{d}\hat{\omega}'$$

$p_1 = \mathrm{r}(p_2, -\hat{\omega})$
$p_0 = \mathrm{r}(p_1, \hat{\omega}')$