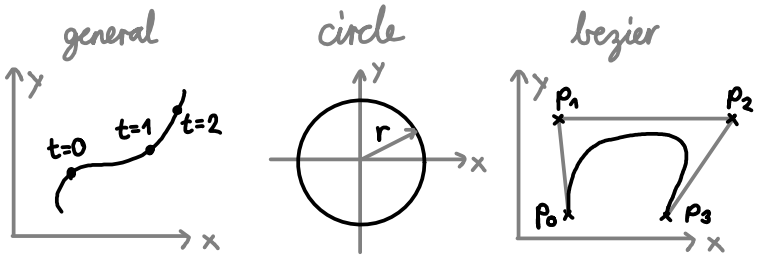
Computer Graphics

Parametric geometry representation



- planar curve $(\mathbb{R}^1 \to \mathbb{R}^2)$: $\underline{s}(t) = (x(t), y(t))$

  $\underline{s}(t) = r \cdot (\cos t, \sin t)$ (circle) $\quad t \in [0, 2\pi[$

  $\underline{s}(t) = \sum_{i=0}^{n} \underline{p}_i \cdot B_i^n(t)$ (bezier) $\quad \begin{cases} B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i} \\ t \in [0,1] \end{cases}$

- space curve $(\mathbb{R}^1 \to \mathbb{R}^3)$: $\underline{s}(t) = (x(t), y(t), z(t))$
- surface in 3D $(\mathbb{R}^2 \to \mathbb{R}^3)$: $\underline{s}(u,v) = (x(u,v), y(u,v), z(u,v))$

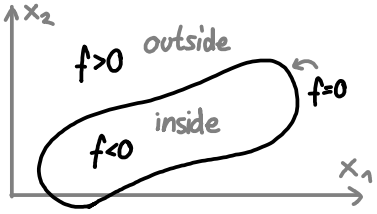
  sphere: $\underline{s}(u,v) = r\,(\cos(u)\cos(v), \sin(u)\cos(v), \sin(v)) \quad (u,v) \in [0, 2\pi[\times [-\frac{\pi}{2}, \frac{\pi}{2}]$

  bezier surface: $\underline{s}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} p_{i,j}\, B_i^m(u)\, B_j^n(v) \quad (u,v) \in [0,1] \times [0,1]$

- normal vector: $\underline{n} = \dfrac{\underline{s}_u \times \underline{s}_v}{\|\underline{s}_u \times \underline{s}_v\|} \quad \left(\underline{s}_u = \dfrac{\partial s(u,v)}{\partial s_u} \quad \underline{s}_v = \dfrac{\partial s(u,v)}{\partial s_v} \right)$

- Advantages
  - easy to generate points on surface, line
  - analytic formula for derivatives
- Disadvantages
  - hard to determine inside/outside
  - hard to determine if point on line/plane

Implicit geometry representation



- curve in 2D: $S = \{ \underline{x} \in \mathbb{R}^2,\ f(\underline{x}) = 0 \}$

  circle: $f(x,y) = x^2 + y^2 - r^2$
- surface in 3D: $S = \{ \underline{x} \in \mathbb{R}^3,\ f(\underline{x}) = 0 \}$

  sphere: $f(x,y,z) = x^2 + y^2 + z^2 - r^2$
- normal vector: $\underline{n} = \nabla f(x,y,z) = \left(\dfrac{\partial f}{\partial x}, \dfrac{\partial f}{\partial y}, \dfrac{\partial f}{\partial z} \right)^T$
- set operations:

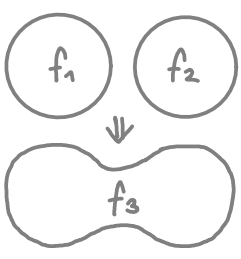
  unite f_1 and f_2: $f_3 = \min(f_1, f_2)$ smooth: $f_3 = \frac{1}{1+a}\left(f_1 + f_2 - \sqrt{f_1^2 + f_2^2 - 2a f_1 f_2}\right)$

  intersect f_1 and f_2: $f_3 = \max(f_1, f_2)$ smooth: $f_3 = \frac{1}{1+a}\left(f_1 + f_2 + \sqrt{f_1^2 + f_2^2 - 2a f_1 f_2}\right)$

  subtract f_2 from f_1: $f_3 = \max(f_1, -f_2)$
- blobs: $S: \{ f(\underline{x}) = \text{Const.} \}$ with $f(\underline{x})$ having smooth fall-off to 0.

  sphere blob: $f_i(\underline{x}) = b_i e^{-a_i \|\underline{x} - \underline{x}_i\|^2}$ with surface at $f(\underline{x}) = e^{-1}$

  ↳ unite blobs: $f_3 = f_1 + f_2$

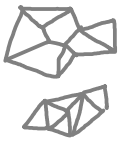


- Advantages
  - easy to determine inside/outside
  - easy to determine if point is on curve/surface
- Disadvantages
  - hard to generate points on curve/surface
  - no good for real-time rendering

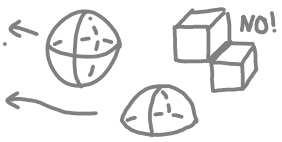
Polygonal Meshes

- **polygon**: vertices $v_0,\dots,v_{n-1}$ and edges $\{(v_0,v_1),\dots,(v_{n-2},v_{n-1})\}$ form polygon.
  - closed: $v_0=v_{n-1}$ - planar: all v_i on plane - simple: not self-intersecting

- **polygonal mesh**: set M of closed, simple polygons Q_i only intersecting with edges/vertices
  - vertex degree: # of incident edges - boundary: set of edges belonging to just 1 polygon



- **triangle mesh**: polygon mesh with all polygon n=3 (triangle)
- **manifold**: - 2-manifold: region around any point on mesh is homeomorphic to an open disk.
  - manifold with boundary: same as above, but can be half-disc



- **genus**: # of "handles" of mesh 0:  1:  2 ...
- **Euler-Poincaré formula**: $\chi(M)=v-e+f$ = #vertices - #edges + #faces = const. for a shape, no matter the mesh.

Texture Mapping

Each vertex (x,y,z) has a mapping (u,v) in the texture image containing RGB color. The color of a pixel inside a triangle is located at the (u,v) coord. calculated by weighting the 3 vertice (u,v) with the barrycentric coord.
Good mapping $(x,y,z)\to(u,v)$ tries to preserve: 2D angles, 2D distances, 2D areas → minimal distortion

light sources

- **incandescence**: produced from heat = black body radiation (see thermo III part I notes)
- **luminescence**: all that is not incandescence. (everything that follows)
- **atomic emission**: emitted when electrons in an atom go to a lower energy orbital. (sodium-vapour-lights)
- **molecular emission**: same as atomic, but with molecular orbitals (northern lights)
- **stimulated emission**: photons trigger atomic/molecular emission without being absorbed (laser)
- **fluorescence**: photon is absorbed and then emitted at a lower wavelenght (neon-lamp)
- **phosphorescence**: same as fluorescence, but with longer delay between absorption and emission (dark-glow-paint)
- **chemiluminescence**: emission of light due to a chemical reaction (glowstick)
- **bioluminescence**: same as chemiluminescence, when inside an organism. (firefly)

Radiometry

$(d\tilde{\omega}=\sin\theta\cdot d\theta d\phi \;/\; dA_p=\cos\theta\, dA)$

- **photon energy**: $E=hc/\lambda$ [J]

$h=6.63\cdot10^{-34}\ m^2\cdot kg/s$ (Planck const.)
$c=299'792'458\ m/s$ (light-speed)
$k_B=1.381\cdot10^{-23}\ J/K$ (Boltzmann c.)
$\lambda=$ variable m (wavelenght)

- **flux**: $\Phi(A)$ [W]
- **irradiance**: $E=d\Phi/dA$ $(\swarrow_A)$
- **radiosity**: $B=d\Phi/dA$ $(\nearrow_A)$
- **intensity**: $I(A,\theta)=d\Phi/d\tilde{\omega}$
- **radiance**: $L=d^2\Phi/dA_p\,d\tilde{\omega}$

$\int_A(B\text{ or }E)\,dA=\Phi$
$\int_H I\,d\omega=\Phi$
$\int_H L\cdot\cos\theta\,d\omega=(B\text{ or }E)$
$\int_A L\cdot\cos\theta\,dA=I$
$\int_H\int_A L\cdot\cos\theta\,dA\,d\omega=\Phi$

↳ black-body: $L_b(T)=\int_0^\infty 2hc^2/(\lambda^5\cdot(e^{(hc/\lambda k_B T)}-1))\cdot d\lambda$

! L is const. along ray !

Raytracing

- transformations $\underline{\underline{M}}$:

$$\begin{pmatrix} x'/w' \\ y'/w' \\ z'/w' \end{pmatrix} \hat{=} \begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix} = \underline{\underline{M}} \cdot \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}$$ { point → w=1, vector → w=0

- ray: $\underline{r}(t) = \underline{o} + t \cdot \underline{d}$ (ray origin, direction)

$\underline{\underline{M}}$ translate:
$$\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$\underline{\underline{M}}$ scale:
$$\begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

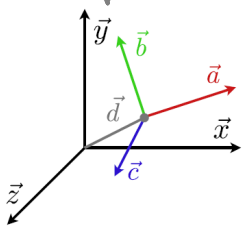
$\underline{\underline{M}}$ rotate:
$$\begin{pmatrix} \top & \top & \top & 0 \\ {}_A e_x^B & {}_A e_y^B & {}_A e_z^B & 0 \\ \bot & \bot & \bot & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$\underline{\underline{M}}$ general:
$$\begin{pmatrix} \top & \top & \top & \top \\ \underline{a} & \underline{b} & \underline{c} & \underline{d} \\ \bot & \bot & \bot & \bot \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

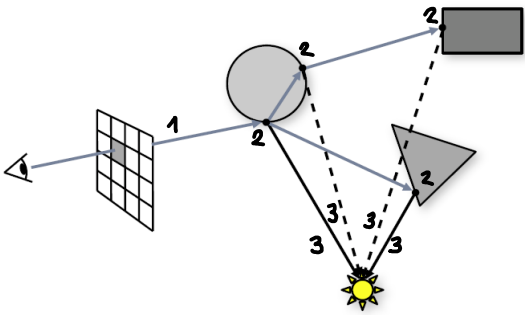
$\underline{\underline{M}}$ rotation from tangent $\vec{n}$:
$$\begin{pmatrix} \top & \top & \top & 0 \\ t & b & n & 0 \\ \bot & \bot & \bot & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$\vec{t}$: zero one component of $\vec{n}$ → swap other 2 → negate one → normalize

$\vec{b}$: $\vec{b} = \vec{n} \times \vec{t}$

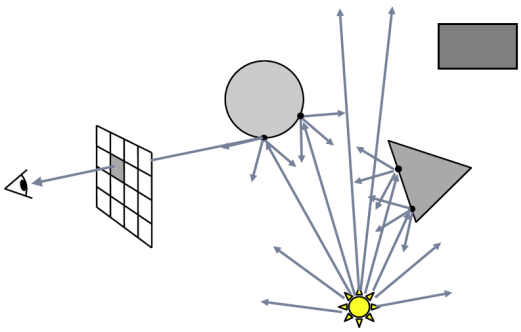


- backward raytracing:



1: ray generation
↓
2: intersections
↓
3: shading

- forward raytracing:

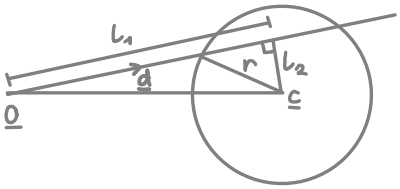


- ray-sphere intersection

$l_1 = (\underline{c} - \underline{o}) \cdot \underline{d}$ (if $l_1 < 0$: break)

$l_2^2 = \|\underline{c} - \underline{o}\|^2 - l_1^2$ (if $l_2^2 > r^2$: break)

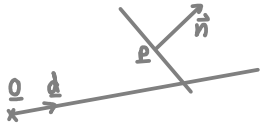
$t = l_1 \pm \sqrt{r^2 - l_2^2}$



- ray-plane intersection

$(\underline{o} + t\underline{d} - \underline{p}) \cdot \underline{n} = 0$

↳ $t = (\underline{o} - \underline{p})\underline{n} \,/\, \underline{d} \cdot \underline{n}$

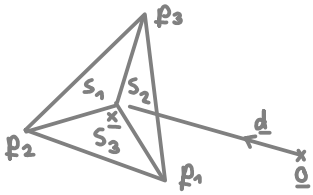


- ray-triangle intersection

$\underline{o} + t\underline{d} = s_1\underline{p}_1 + s_2\underline{p}_2 + s_3\underline{p}_3$

solve for t, s_1, s_2 ($s_1 + s_2 + s_3 = 1$)

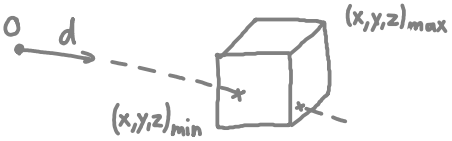
check if $0 \le s_i \le 1$ for $i = 1,2,3$



Acceleration techniques

- axis-aligned bounding boxes AABB:

1) compute bounding box of scene
2) recursively split box until max depth or object/cell limit
3) take ray and walk down the box tree checking intersections
4) intersect ray with objects in last box



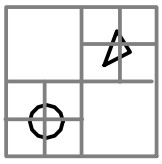
$t_{x1} = \frac{x_{min} - o_x}{d_x}$, $t_{x2} = \frac{x_{min} - o_x}{d_x}$, $t_{y1}, t_{y2}, t_{z1}, t_{z2}$

if $t_{x1} > t_{x2}$ → swap (t_{x1}, t_{x2}), same for y,z

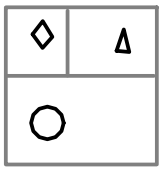
$t_{min} = \max(t_{x1}, t_{y1}, t_{z1})$, $t_{max} = \min(t_{x2}, t_{y2}, t_{z2})$

if $t_{min} < t_{max}$ → hit

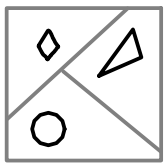
- octree (split in 8 equal cubes)



- kd-tree (split along x, y or z plane)

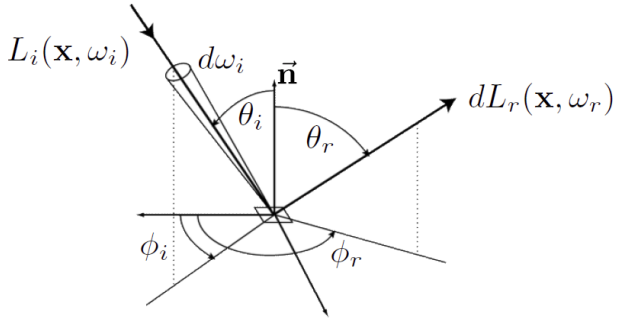


- BSP-tree (split along arbitrary plane)



- alternatives: regular box grid instead of tree / one bounding box per object / bounding sphere

Bidirectional Reflectance Distribution Function BRDF



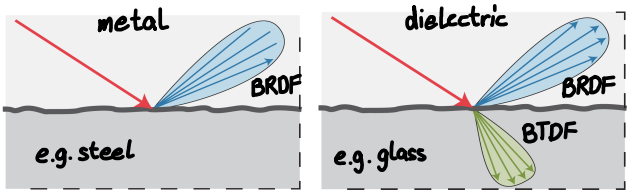
$dL_r(x,\omega_r) = f_r(x,\omega_i,\omega_r) \cdot dE_i(x,\omega_i)$ (f_R: BRDF)

$L_r(x,\omega_r) = \int_{H^2} f_r(x,\omega_i,\omega_r) \cdot L_i(x,\omega_i) \cdot \cos\theta_i \, d\omega_i$ (H^2: hemisphere)

Helmholtz reciprocity: $f_r(x,\omega_i,\omega_r) = f_r(x,\omega_r,\omega_i)$

Energy conservation: $\int_{H^2} f_r(x,\omega_i,\omega_r) \cdot \cos\theta_i \, d\omega_i \leq 1 \quad \forall \omega_r, x$

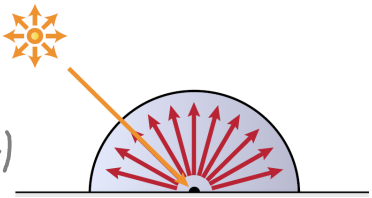
isotropic BRDF: $f_r(x,\omega_i,\omega_r) \rightarrow f_r(\theta_i,\theta_r,\Delta\phi)$



• diffuse BRDF:

$f_r = \varrho/\pi$

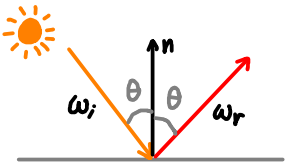
($\varrho \in [0,1]$: albedo)



• ideal specular BRDF:

if $\omega_r = 2(n\cdot\omega_i)\cdot n - \omega_i$ then:

$L_r(\omega_r) = L_i(\omega_i)$ else $L_r(\omega_r) = 0$



($\phi_r = \phi_i + \pi$, $\theta_r = \theta_i$)

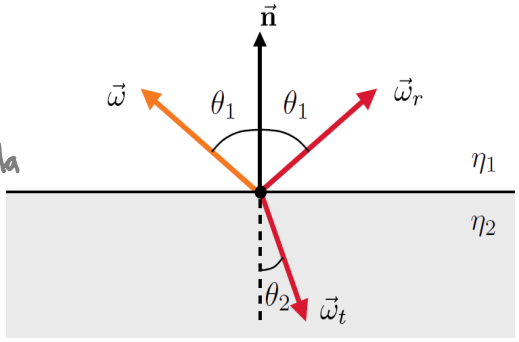
• Fresnel reflection/refraction BRDF/BTDF: (for dielectric)

$\frac{\eta_1}{\eta_2} = \frac{\sin\theta_2}{\sin\theta_1}$

$\varrho_\parallel = \frac{\eta_2\cos\theta_1 - \eta_1\cos\theta_2}{\eta_2\cos\theta_1 + \eta_1\cos\theta_2}$ $\varrho_\perp = \frac{\eta_1\cos\theta_1 - \eta_2\cos\theta_2}{\eta_1\cos\theta_1 + \eta_2\cos\theta_2}$

$F_r = \frac{1}{2}(\varrho_\parallel^2 + \varrho_\perp^2)$ ≙ prob of refl.

$F_t = 1 - F_r$



$\omega_r^* = 2(n\cdot\omega_i)n - \omega_i$ → $f_r(\omega_i,\omega_r) = F_r \cdot \frac{\delta(\omega_r - \omega_r^*)}{\cos(\theta_i)}$ (dirac delta)

$\omega_t^* = -\frac{\eta_1}{\eta_2}(\omega_i - (\omega_i\cdot n)n) - n\sqrt{1-\left(\frac{\eta_1}{\eta_2}\right)^2(1-(\omega_i\cdot n)^2)}$ → $f_t(\omega_i,\omega_t) = F_t \cdot \frac{\delta(\omega_t - \omega_t^*)}{\cos(\theta_i)}$

• Fresnel reflection BRDF: (for metals)

same as above except for: $\varrho_\parallel = \sqrt{\frac{(\eta^2+k^2)\cos^2\theta_i - 2\eta\cos\theta_i + 1}{(\eta^2+k^2)\cos^2\theta_i + 2\eta\cos\theta_i + 1}}$ $\varrho_\perp = \sqrt{\frac{(\eta^2+k^2) - 2\eta\cos\theta_i + \cos^2\theta_i}{(\eta^2+k^2) + 2\eta\cos\theta_i + \cos^2\theta_i}}$

(η: refraction index, k: absorption coeff.)

• normalized Phong BRDF:

$f_r(\omega_i,\omega_r) = \frac{\alpha+2}{\pi}(\omega_r^*\cdot\omega_r)^\alpha$ ($\omega_r^* = 2n(n\cdot\omega_i) - \omega_i$ ideal reflection; α: blur)

• Blinn-Phong BRDF:

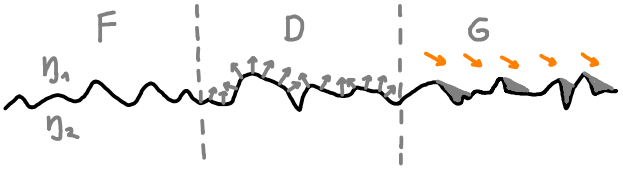
$f_r(\omega_i,\omega_r) = \frac{\alpha+2}{2\pi}(\omega_h\cdot n)^\alpha$ ($\omega_h = (\omega_i+\omega_r)/\|\omega_i+\omega_r\|$ mid-vector; α: blur)

• Microfacet BRDF:

$$f(\omega_i,\omega_r) = \frac{F(\omega_h,\omega_r)\cdot D(\omega_h)\cdot G(\omega_i,\omega_r)}{4\cdot|(\omega_i\cdot n)(\omega_r\cdot n)|}$$

($\omega_h = \frac{\omega_i+\omega_r}{\|\omega_i+\omega_r\|}$)

- Fresnel coeff. $F(\omega_h,\omega_r)$: F_r of Fresnell refl.
- self-shadowing/masking coeff $G(\omega_i,\omega_r)$: ?
- Microfacet distribution $D(\omega_h)$: (normals distrib.)
  - ↳ Beckmann: $D(\omega_h) = 1/\pi\alpha^2\cos^4\theta_h \cdot e^{-\tan^2\theta_h/\alpha^2}$
  - ↳ Blinn: $D(\omega_h) = \frac{e+2}{\pi}(\omega_h\cdot n)^e$
  - ↳ GXX: prettier, but complicated...



• Oren-Nayar BRDF:

$f_r(\omega_i,\omega_r) = \frac{\varrho}{\pi}(A + B\cdot\max(0,\cos(\theta_i-\theta_r))\cdot\sin(\max(\theta_i,\theta_r))\cdot\tan(\min(\theta_i,\theta_r))$

($A = 1 - \frac{\sigma^2}{2\sigma+0.66}$, $B = \frac{0.45\sigma^2}{\sigma^2+0.09}$)

Importance Sampling

X : "random" variable with below characteristics

$p(x)$: probability density function (PDF) $\quad p(x) = dP(x)/dx \;;\; p(x) \geq 0 \;\forall x \;;\; \int_{-\infty}^{\infty} p(x)dx = 1$

$P(x)$: cumulative distribution function (CDF) $\quad P(x) = \text{prob}(X \leq x) \;;\; P(b) - P(a) = \int_a^b p(x)dx = \text{prob}(a \leq X \leq b)$

$Y = f(X)$ → expected value: $E\{Y\} = \int_{-\infty}^{\infty} f(x)\,p(x)dx \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i)$ (with x_i distributed according to $p(x_i)$)

→ variance: $\text{var}\{Y\} = E\{(Y - E\{Y\})^2\}$

- **montecarlo integration:** $\int_a^b f(x)dx = \int_{-\infty}^{\infty}\left[\frac{f(x)}{p(x)}\right]\cdot p(x)dx \approx \frac{1}{N}\cdot\sum_{i=1}^{N} f(x_i)/p(x_i)$ with $p(x) = \begin{cases} a \leq x \leq b : 1/b-a \\ \text{else} : 0 \end{cases}$
- **sampling arbitrary $p(x)$:** 1) get CDF: $P(x) = \int_0^x p(x')dx'$ 2) invert $*=P(x)$: $x = P^{-1}(*)$ 3) obtain rand.nr. ξ: $p_\xi(x) = \begin{cases} x \in [0,1] : 1 \\ \text{else} : 0 \end{cases}$ 4) compute $X_i = P^{-1}(\xi)$
- **sampling arb. 2D $p(x,y)$:** 1) get marginal: $p(x) = \int_{-\infty}^{\infty} p(x,y)dy$ 2) get conditional: $p(y|x) = p(x,y)/p(x)$ 3) sample $p(x)$ like above → X_i 4) sample $p(y|X_i)$ like above → Y_i
- $p_Y(y)$ from $p_X(x)$, $y = T(x)$ → $p_Y(y) = p_Y(T(x)) = p_x(x)/|J_T(x)|$; $|J_T(x)| = \text{abs}(\det(\begin{bmatrix} \partial T_1/\partial x_1 & \cdots & \partial T_1/\partial x_n \\ \vdots & & \vdots \\ \partial T_m/\partial x_n & \cdots & \partial T_m/\partial x_n \end{bmatrix}))$
- **area preserving sampling:** 1) define prob. density in cartesian coord. 2) pick convenient coord. for sampling 3) use above instructions to relate PDF's 4) use 2D or 1D sampling from above

$L_r(x,\omega_r) = \int_{H^2} f_r(x,\omega_r,\omega_i)\cdot L_i(x,\omega_i)\cos\theta_i d\omega_i \approx \frac{1}{N}\sum_{n=1}^{N} f_r(x,\omega_r,\omega_{in})\cdot L_i(x,\omega_{in})\cos\theta_{in} \;/\; p(\omega_{in})$

↑ choose $p(\omega_{in})$? (pdf for picking ω_{in}) → best if p high where $f_r \cdot L_i \cdot \cos\theta_i$ high.

- **none:** (uniformly sample hemisphere)

$p(\omega_i) = 1/2\pi \rightarrow \omega_i = \begin{pmatrix} \xi_1 \\ \sqrt{1-\xi_1^2}\cdot\cos(2\pi\xi_2) \\ \sqrt{1-\xi_1^2}\cdot\sin(2\pi\xi_2) \end{pmatrix}$

↳ $L_r(x,\omega_r) \approx \frac{2\pi}{N}\sum_{n=1}^{N} f_r(x,\omega_r,\omega_{in})\cdot L_i(x,\omega_{in})\cos\theta_{in}$

- **$\cos\theta_i$:** (cos-weighted hemisphere)

$p(\omega_i) = \cos\theta_i/\pi \rightarrow \omega_i = \begin{pmatrix} \sqrt{1-\xi_1} \\ \sqrt{\xi_1}\cdot\cos(2\pi\xi_2) \\ \sqrt{\xi_1}\cdot\sin(2\pi\xi_2) \end{pmatrix}$

↳ $L_r(x,\omega_r) \approx \frac{\pi}{N}\sum_{n=1}^{N} f_r(x,\omega_r,\omega_{in})\cdot L_i(x,\omega_{in})$

- **normalized Phong BRDF:**

$p(\Delta\theta,\Delta\phi) = \frac{\alpha+2}{2\pi}\cos(\Delta\theta)^\alpha \rightarrow \begin{matrix} \Delta\theta = \cos^{-1}((1-\xi_1)^{1/\alpha+2}) \\ \Delta\phi = 2\pi\xi_2 \end{matrix}$

→ ω_i = rotate ω_r by $\Delta\theta$ in dir. $\Delta\phi$ → mirror on n

↳ $L_r(x,\omega_r) \approx \frac{1}{N}\sum_{n=1}^{N} L_i(x,\omega_{in})\cos\theta_{in}$

- **Beckmann BRDF + $\cos\theta_i$:**

$p(\omega_N) = e^{-\tan^2\theta_i/\alpha^2}\cdot\frac{1}{\pi\alpha^2\cos^3\theta_i}\cdot\cos\theta_i$

→ $\theta_N = \tan^{-1}(\sqrt{-\alpha^2\cdot\ln(1-\xi_1)})$, $\phi_N = 2\pi\xi_2$ → $\omega_i = (\theta_N,\phi_N)\cdot\omega_o - \omega_o$

↳ $L_r(x,\omega_r) \approx \frac{1}{N}\sum_{n=1}^{N} L_i(x,\omega_{in})$

- **sample environment light:** (sky-sphere)

$p(\omega_i) = \underbrace{L_{env}(\theta_i,\phi_i)\sin\theta_i}_{\text{mercator map of sky}} \;/\; \underbrace{\sum^{\theta\phi} L_{env}(\theta,\phi)\sin\theta}_{\text{normalization}}$

$p(\theta_i) = \sum_{\phi_n=0}^{\phi_{max}} L_{env}(\theta_i,\phi_n)\sin\theta_i$, $p(\phi|\theta) = p(\theta,\phi)/p(\theta)$

$P(\theta_i) = \sum_{\theta_n=0}^{\theta_i} p(\theta_n)$, $P(\phi_i) = \sum_{\phi_n=0}^{\phi_i} p(\phi_i|\theta_i)$

$\theta_i = P^{-1}(\pi\xi_1 - \pi/2)$, $\phi_i = P^{-1}(2\pi\xi_2)$

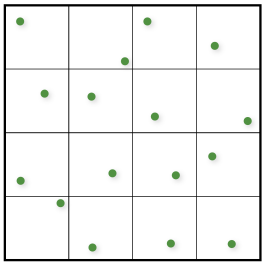
- **combine 2 samplers:** $(p_1(\omega_i), p_2(\omega_i), \omega_{i1}(\xi), \omega_{i2}(\xi))$

↳ $L_r(x,\omega_r) \approx \frac{1}{N}\sum_{n=1}^{N} \frac{f_r(x,\omega_r,\omega_{in})\cdot L_i(x,\omega_{in})\cos\theta_{in}}{1/2\cdot p_1(\omega_{in}) + 1/2\cdot p_2(\omega_{in})}$

(use ω_{i1} 50% of the time and ω_{i2} 50% for ω_{in})

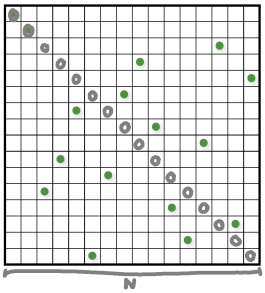
low discrepancy sampling (generate random number, avoiding clusters)

stratified (= jittered)



subdivide sampling space in as many subspaces (line, square, cube, ..) as needed samples. sample inside each subspace.

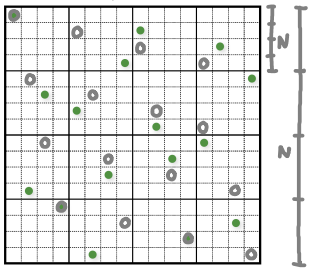
N-rooks (= latin hypercube)



place N samples randomly in cells along the diagonal (0: init. pos.)
shuffle each dimension:

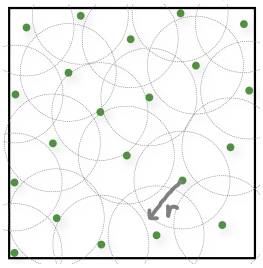
```
for i=(1:dim) ; for j=(1:count)
    swap sample(j,i)
    with sample(rand(j:count), i)
```

multi-jittered (stratified + N-rooks)



initialize like N-rooks but on cells shown left. shuffle dimensions like N-rooks, but shuffle x-coord only within each column (N·N×N) and y-coord within each row (N×N·N)

Poisson-disc (enforce min. dist. between samples)



throw random darts at sampling space and only keep it if no previous darts landed closer than r. continue until required # of samples "stick".

Van der Corput sequence (binary):

for every sample and every dimension:
1) assign ascending integer numbers to each sample coord.
2) express that nr. in binary
3) mirror digits arround decimal point.

e.g. $5 \to 5_{10} = 101_b \to 0.101_b = 0.625$
$6 \to 6_{10} = 110_b \to 0.011_b = 0.375$

Halton sequence:

similar to Van der Corput, but have ascending base (only prime!) and fixed number i (= 15 below)

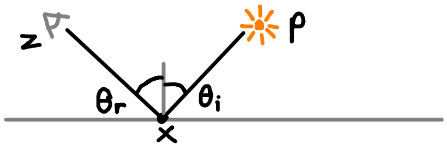
e.g: $2 \to 15_{10} = 1111_2 \to 0.1111_2 = 0.9375$
$3 \to 15_{10} = 120_3 \to 0.021_3 = 0.2592$
$5 \to 15_{10} = 30_5 \to 0.03_5 = 0.12$
$7 \to 15_{10} = 21_7 \to 0.12_7 = 0.1837$

Hammersley sequence: same as Halton, but first number is $1/N$ (N: # of samples. then continue like Halton.

direct illumination

- **point light**: visibility, I

$$L_r(x,z) = f_r(x,p,z)\,V(x,p)\cdot\frac{\Phi}{4\pi}\cdot\frac{|\cos\theta_i|}{\|x-p\|^2}$$



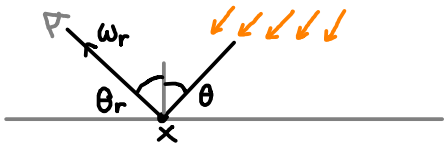
- **spot light**:

like point light but $I \neq \Phi/4\pi\ V(x,p)$

$$I(x,p) = (n_p\cdot(x-p)/\|x-p\| < \cos(\alpha))\cdot I$$

- **directional light**:

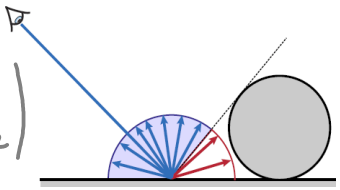
$$L_r(x,\omega_r) = f_r(x,\omega,\omega_r)V(x,\omega)\cdot L_d\cdot\cos\theta$$



- **ambient white sky** (for diffuse objects)

$$L_r(x,\omega_r) = \frac{\rho}{\pi}\int_{H^2} V(x,\omega_i)\cos\theta_i\,d\omega_i \approx \frac{\rho}{N}\sum_{k=1}^{N} V(x,\omega_{i,k})$$

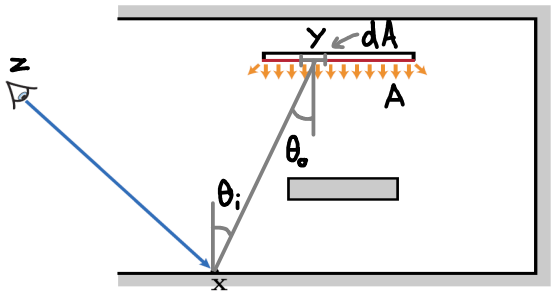
(PDF: $p(\omega_{i,k}) = \cos\theta_{i,k}/\pi$; $V(x,\omega_i)$: 1 if sky visible, 0 else)

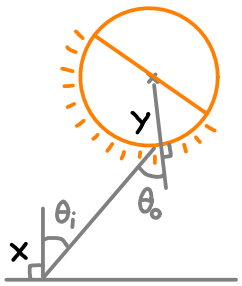


- **surface area light** (emitter area sampling)

$$L_r(x,z) = \int_A f_r(x,y,z)\cdot L_e(x,y)\cdot G(x,y)\,dA(y)$$

$$G(x,y) = \underbrace{V(x,y)}_{\text{visibility}}\cdot|\cos\theta_i|\cdot\underbrace{|\cos\theta_o|/\|x-y\|^2}_{d\omega/dA}$$





- quad light: $L_r(x,z) = \frac{\Phi}{AN}\sum^{N} f_r(x,y_i,z)\cdot G(x,y_i)$ (PDF: sample y_i uniformly on A)

- sphere light: $L_r(x,z) = \frac{\Phi}{4\pi r^2 N}\cdot\sum^{N} f_r(x,y_i,z)\cdot G(x,y_i)$ (PDF: sample y_i cos-weighted on visible hemisphere)

↳ ! drop $\cos\theta_o$!

global illumination

Heckberts classification: L = light, E = eye, D = diffuse refl., S = specular refl.
X+: one or more event, X*: zero or more event, (X|Y) one of 2 events

↳ direct illumination: L(D|S)E / classical Whitted: LDS*E / global illumination: L(D|S)*E

$$L(x,\omega) = L_e(x,\omega) + \int_{H^2} f_r(x,\omega',\omega)\,L(r(x,\omega'),-\omega')\cos\theta'\,d\omega'$$

$$\hat{=}$$

$$L(x,z) = L_e(x,y) + \int_A f_r(x,y,z)L(y,x)\,G(x,y)\,dA(y)$$

$L(x,\omega)$: outgoing radiance at x in dir ω
$r(x,\omega')$: next intersection point from x in dir. ω'
$L(r(x,\omega'),-\omega')$: incoming radiance to x from dir ω'

$\|z-x\| \hat{=} \omega$, $\|y-x\| \hat{=} \omega'$, $G(x,y) = V(x,y)\cdot\frac{\cos\theta_o\cos\theta_i}{\|y-x\|^2}$ ← (see above "surface area light"!)

- **partitioned integral form**:

$$L(x,\omega) = \underbrace{\text{direct illumination } L_d(x,\omega)}_{\text{sample only } L_e \text{ from emitters}} + \underbrace{\text{indirect illumination } L_i(x,\omega)}_{\text{sample } L \text{ without } L_e \text{ on hemisph.}}$$

(ignore emitter in L_i, except if last bounce was purely specular!)

$$L(x,\omega) = \int_A f_r\cdot L_e\cdot G\,dA + \int_{H^2} f_r\cdot L\cdot\cos\theta\,d\omega$$

(recursive / multiple bounces!)

- **path integral form**:

$$I_j = \int W_e(x_0,x_1)L_e(x_k,x_{k-1})\cdot T(\bar{x})\,d\bar{x} \approx \frac{1}{N}\sum^{vp} W_e(x_0,x_1)L_e(x_k,x_{k-1})T(\bar{x})/p(\bar{x})$$

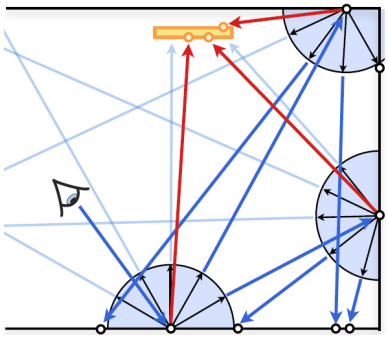
$$T(\bar{x}) = G(x_0,x_1)\prod_{j=1}^{k-1} f(x_j,x_{j+1},x_{j-1})\,G(x_j,x_{j+1}) \;;\; p(\bar{x}) = p(x_0)\times p(x_1|x_0)\times p(x_2|x_0x_1)\times\dots$$

I_j: radiance of pixel j
$W_e(x_0,x_1)$: susceptibility of pixel at x_0
$L_e(x_k,x_{k-1})$: emission of last point in path
$T(\bar{x})$: throughput of path $\bar{x} = x_0\to x_1\to\dots\to x_k$
$p(\bar{x})$: joint PDF of all path choices

monte-carlo ray tracing:

1) generate rays from eye to each pixel
2) intersect ray with scene
3) sample intersection BRDF multiple times
4) sample emitter at intersection and add radiance to output

! do not add radiance if intersection is on an emitter (avoid double counting) except if last intersection was specular. !



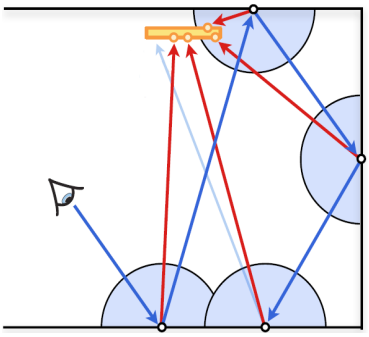
path tracing (+russian roulette):

like monte carlo, but sample BRDF at intersection only once.

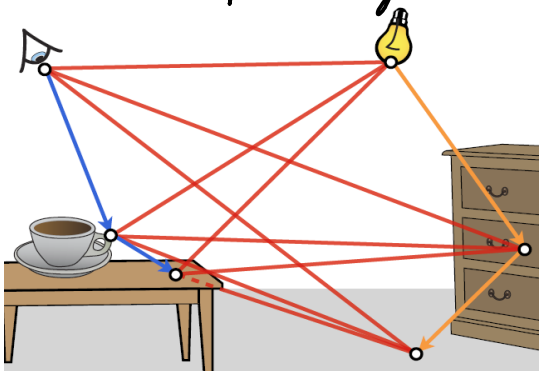
russian roulette: stop ray with prob. equal to curr. throughput (and compensate for it!)

↳ $T(\bar{x}) = T(\bar{x}) * \text{stop\_prob}$ ↵

← can do multiple importance sampling to avoid this.

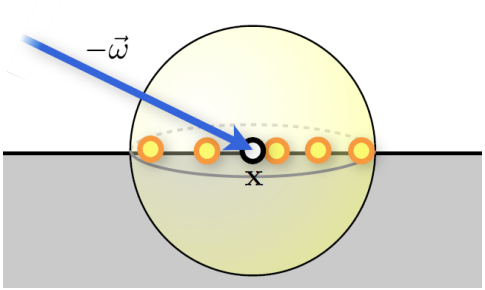


bidirectional path tracing:



1) generate path of len.t from eye
2) generate path of len.s from light
3) connect all t·s possible paths
4) use path integral form eq.

photon mapping:

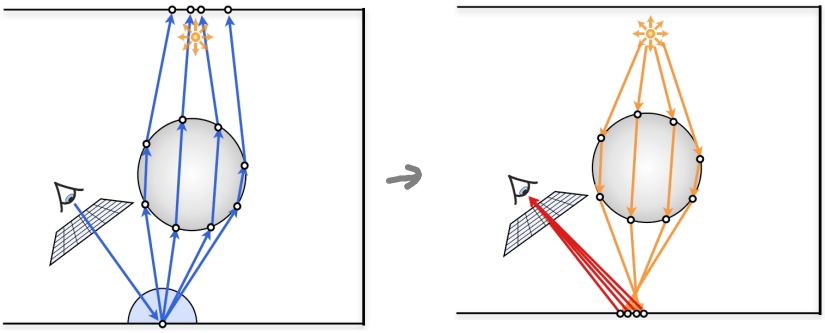


1) emitt photons from emitters (photon = energy packet)
2) scatter photons trough scene (like pat tracing)
3) save loc+dir. where photon hits diffuse surface in "photon-map"
4) run path tracing alg., but calc. indirect illum with avg. energy of photons per area near intersection point.

light tracing:

start rays at the emitters instead of the eye and replace emitter sampling with "eye sampling"

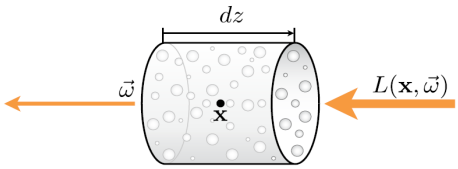
↳ better LSDE, worse LDSE



Volume rendering (rendering of dense participating volumes)

! radiance L of ray is no longer const. along ray when traversing dense volume !

$dL = -\sigma_a L\,dz - \sigma_s L\,dz$ } gains
$+\sigma_a L_e\,dz + \sigma_s L_s\,dz$ } losses



L: incomming radiance
$L+dL$: outgoing radiance
L_e: emitted radiance
L_s: in-scattered radiance
σ_a: absorption coeff.
σ_s: scattering coeff.
$\sigma_t = \sigma_a + \sigma_s$: extinction coeff.

mean free distance: $1/\sigma_t$

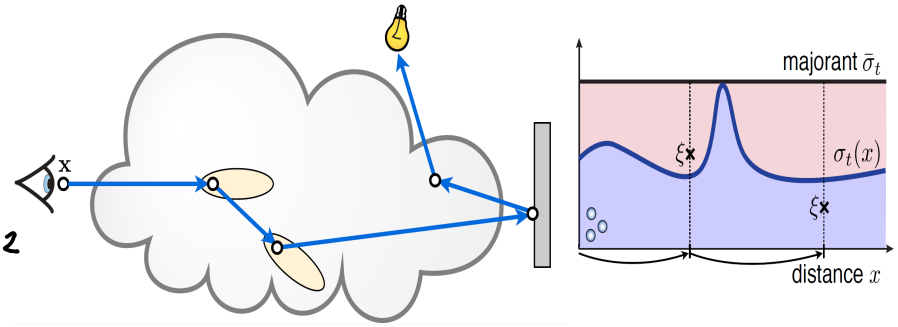
Transmittance: $T_r = \frac{L_{out}}{L_{in}} = e^{-\int_0^z \sigma_t(t)dt}$

in-scattering radiance:
$L_s(x,\omega) = \int_{S^2} f_p(x,\omega,\omega') L_i(x,\omega')\,d\omega'$
↳ phase function (analog BRDF)

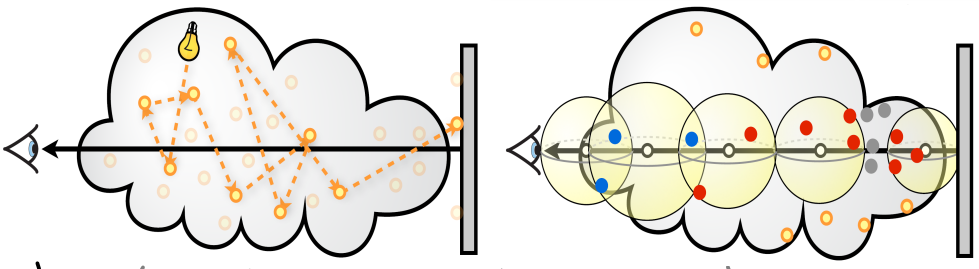
volume rendering equation:
$$L(x,\omega) = \underbrace{T_r(x,x_z)L(x_z,\omega)}_{\text{background radiance}} + \underbrace{\int_0^z T_r(x,x_t)\sigma_a(x_t)L_e(x_t,\omega)dt}_{\text{emitted radiance}} + \underbrace{\int_0^z T_r(x,x_t)\sigma_s(x_t)L_s(x_t,\omega)dt}_{\text{in-scattered radiance}}$$

- uniform scattering: $f_p(\omega',\omega) = 1/4\pi$
- Henyey+Gr. scattering: $f_p(\theta) = 1/4\pi \cdot (1-g^2)/(1+g^2-2g\cdot\cos\theta)^{3/2}$ { $g<0$: backward scattering (smoke); $g>0$: forward scattering (steam)
- Schlick scattering: $f_p(\theta) = 1/4\pi \cdot (1-k^2)/(1-k\cos\theta)^2$; $k = 1.55g - 0.55g^3$ (cheap approx. of above)
- Rayleigh scattering: $f_p(\theta,\lambda,\eta,..) = ...$ (approx. of Maxwell eq. for tiny scatterers)
- Lorenz-Mie scattering: $f_p(\theta,\lambda,\eta,..) = ...$ (approx. of Maxwell eq. for large scatterers)

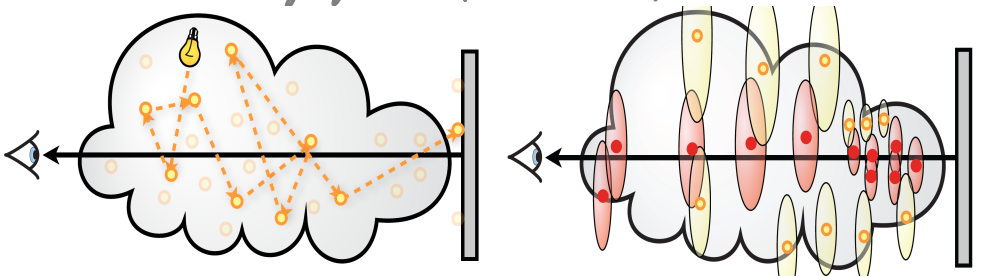
- delta-tracking:
  1) assume a homogeneous volume $\bar{\sigma}_t = \max(\sigma_t)$
  2) sample mean free dist. $t = -\log(1-\xi)/\bar{\sigma}_t$
  3) step distance t along ray.
  4) the collision is real if $\sigma_t(x)/\bar{\sigma}_t > \xi$, else goto 2
  5) perform scattering with f_p, then goto 2.



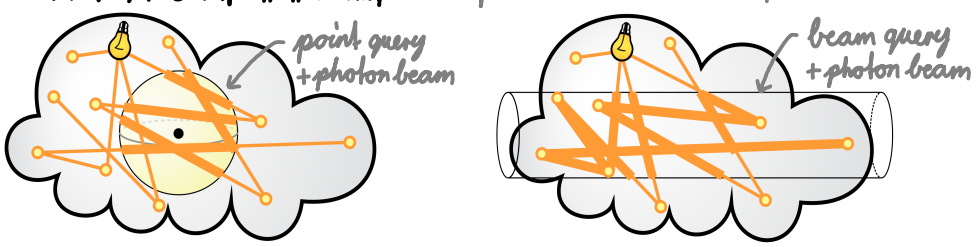
- Volumetric photon mapping:
  1) use light-tracing + delta-tracking to deposit photons in volume.
  2) estimate in-scattering along ray with
  $\sum_{i=1}^{N} T_r(x,x_i)\sigma_s(x_i)\cdot\left(\sum_{j=1} f_p(x_i,\omega_j,\omega)\,\Phi_j/\frac{4}{3}\pi r^3\right)\cdot\Delta t$ (i: steps along ray of Δt, j: photons inside sphere r)



- beam radiance estimate:
  1) use light-tracing + delta-tracking to deposit photons in volume.
  2) assign a sphere to each photon based on photon density in region
  3) estimate in-scattering along ray with $\sum_{i=1}^{M} T_r(x,x_i)\,\sigma_s(x_i)\,f_p(x_i,\omega_i,\omega)\,\Phi_i/\pi r_i^2$ (i: photons) (can also use fixed radius!)



- photon beams, beam query:
  instead of depositing photon points, deposit photon beams!

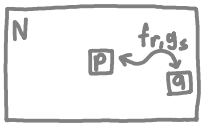


denoising (lower noise in render without using more rays)

- **bilateral filter**: replace pixel intensity by function of nearby pixels distance and/or intensity

$$I_{filt}(p) = \sum_{q \in N} I(q) \cdot f_r(I(p), I(q)) \cdot g_s(p,q) \;/\; \sum_{q \in N} f_r(I(p), I(q)) \cdot g_s(p,q)$$

(p,q: pixel coord $I(\cdot)$: intensity of pixel N: filter window
f_r: range kernel g_s: spatial kernel (both e.g. gaussian func.))



assuming normal distrib.: $I(x) \sim \mathcal{N}(I_{real}(x), varI(x))$

$$f_r(I(p), I(q)) = e^{-\frac{(I(p)-I(q))^2 - (varI(p) + varI(q))}{k^2 \cdot (varI(p) + varI(q))}}$$

$(E(I(p))-E(I(q)))^2 = E((I(p)-I(q))^2 - varI(p) - varI(q))$

k: scaling variable. pick k e.g. 0.45

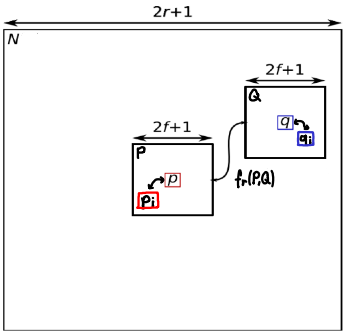
$$g_s(p,q) = 1$$

whole window equaly important

- white noise (e.g random sampling) : calculate $varI(x)$ from I of all rays starting at x
- correlated noise (e.g. low discrepancy sampling) :
  - render 2 images with different seed → $varI(x) = (I_1(x) - I_2(x))^2/4$
  - blur $varI(x)$ with e.g. box blur and take max to original to preserve high varI outliers

- **non-local means filter**: same as bilateral, but consider patches arround pixels

```
INPUT: I, r, f, k, sigma_I
flt = wgtsum = 0                  // 1. Initialize output
for each pixel p:                 // 2. Loop over pixels
   for q in N(p):                 // 3. Loop over neighbors
      wgt = 0                     // 4. Loop over pairs
      for p_i, q_i in P,Q:        //    (pi, qi) to get
         wgt += f_r(I(p_i), I(q_i))//    mean over patch
      wgt /= area(P)              //    Weight of Q
      for p_i, q_i in P,Q:        // 5. Store weighted
         wgtsum(p_i) += wgt       //    contribution of Q
         flt(p_i) += wgt*I(q_i)   //
flt /= wgtsum                     // 6. Normalize output
```



P: patch arround center pixel p
Q: patch arround pixel q
p_i, q_i: pixels inside of P, Q

(algorithm can be simplified to one for loop + convolution)

- **joint non-local means filter**: (leverage scene information)

include more kernels in formula for $I_{filt}(p)$: depth, normals, albedo, direct visib., ...
- alternative: only use largest kernel instead of multiplying them

(pixel based! not patch based like f_r above!)

$$h_{\square}(p,q) = (feature(p) - feature(q))^2 \;/\; k_{\square}^2 \cdot |grad(feature(p))|^2$$

(eventually blur these if noisy)

- **separate direct/indirect illumination**: split render in direct/indirect illumination → apply above denoising methods to both separately → recombine them

- **adaptive rendering**
  - a posteriori: advanced denoisers like above
    - weighted local regression: use features like joint non-local means filter, but ..... smarter..
    - machine learning: let CNN learn to denoise with low/high sample/pixel references
  - in between: shoot rays uniformly → get pixel variances → shoot more rays on high var. pixels
  - a priori: use knowledge of light transport to determine noise prone pixels before rendering → sample them more
    - gradient analysis: somehow get grad(I) and sample more where it is high
    - frequency analysis: somehow get fft(I) use nyquist theorem → sample at 2x freq.
    - lightfield analysis: zzzzzzzzzzzzzzzzzzzzzz

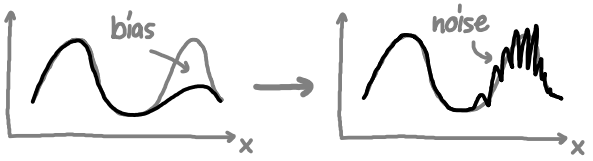
Machine learning for light transport

- predict global illumination:
  inputs: 1st intersection position, normal, albedo, dir. to eye, dir. to light
  output: indirect illumination radiance

  (works well for unchanging scene geometry and changing light, camera pos, material colors.)

- predict volumetric rendering:
  input: 1st bounce position from free dist. sampling, grid of densities in neighbourhood (different scales), dir. to light
  output: radiance from 2+ bounces in cloud

- improvement: predict distribution
  predicting a value → bias
  predict a pdf → less bias, more noise
  how to store, sample, evaluate pdf?



usually better to have noise than bias!

- path guiding (done while rendering) (helps with LSDSE rays)
  improve hemisphere prob. distribution at all intersections that likely hit light from prev. result
  input: incoming radiance of a intersection + direction
  output: improved sampling method for that region ← (ML used to learn parameters of functions describing that) (could also use mixt. of gaussians, histograms,.. instead of ML!)

- ML-renderer: have CNN that generates paths and trains itself to generate better paths (more troughput) as the scene is rendered. (work in progres)

Material acquisition

goal: find material normals + BRDF from photograph
prequisites: camera position/calibration fixe + lighting with controlled dir.+radiance | no indirect illum.

$$L_{eye}(x,\omega_o) = \int_{H^2} f_r(x,\omega_o,\omega_i)\cdot L_i(x,\omega_i)\cdot(\omega_i\cdot n)\,d\omega_i$$

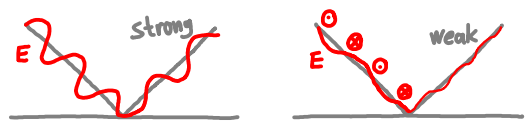
(■ known ■ unknown)

$$L_{eye}(x,\omega_o) = f_{r,diff}\cdot L_i\cdot(\omega_i\cdot n)$$

(assumptions: directional light + diffuse BRDF)

→ 3 unknowns $f_r, n(\theta,\phi)$ → solve with 3 different ω_i and const. L_i !

- avoid diffuse BRDF constraint (by filtering out specular reflections with polarization)
  use linear polarization filter on light and on camera at ⊥
  (diffuse refl. randomizes polarization, specular refl. keeps it.)
  can approx. specular BRDF with: $I_{diff} = I_\perp \rightarrow f_{r,diff}$, $I_{spec} = I_{||} - I_{diff} \rightarrow f_{r,spec}$



depends on surface normal trough fresnell!
↳ could reconstruct n from rotating filter + 2 view

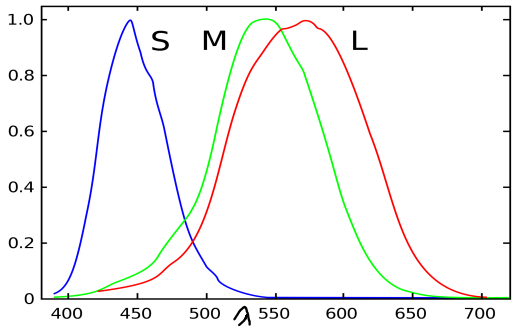
(different normals can be computed for each color channel and each of diffuse/specular to fake sub-surface scattering)

- estimate roughness (by comparing rendering with photo)

- camera problems:
  - vignetting
  - chrom. aberr.
  - bad colorspace
  - over/under exposure

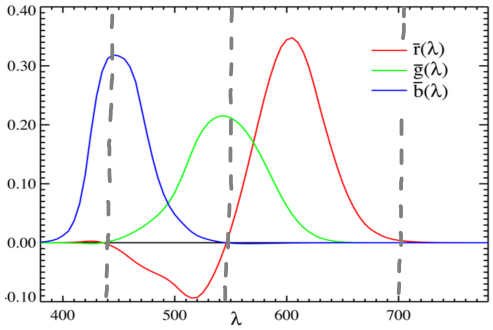
Color representation (ways to describe radiance L color and BRDF f_r color)

tristimulus values



normalized wavelength sensitivity of the 3 human cone cells.

CIE RGB

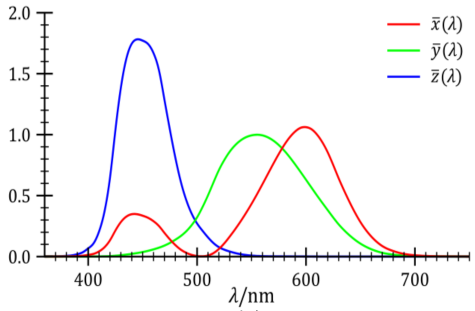


$$R = \int_\lambda L(\lambda)\cdot \bar{r}(\lambda)\, d\lambda$$
$$G = \int_\lambda L(\lambda)\cdot \bar{g}(\lambda)\, d\lambda$$
$$B = \int_\lambda L(\lambda)\cdot \bar{b}(\lambda)\, d\lambda$$

(allow R negative when no combination exists)

describe a wavelength by mixing 3 primary wavelengths

CIE XYZ (standard observer)

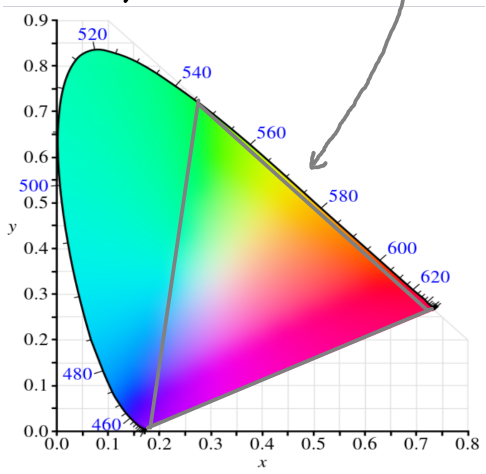


XYZ calculated like RGB

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \frac{1}{0.176\,97} \begin{bmatrix} 0.490\,00 & 0.310\,00 & 0.200\,00 \\ 0.176\,97 & 0.812\,40 & 0.010\,63 \\ 0.000\,00 & 0.010\,00 & 0.990\,00 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

(XYZ such that Y = perceived brightness)

CIE xyY



like XYZ, but replace XZ chromaticity with xy and keep Y brightness

$$x = X/(X+Y+Z)$$
$$y = Y/(X+Y+Z)$$

(x,y contain all chromacities in range [0,1] unlike RGB, XZ)

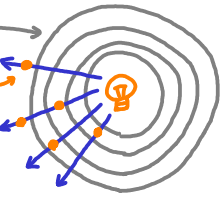
- sRGB: triangle in XYZ (like CIE RGB but smaller) and "gamma" function applied to each channel
- adobe RGB: like sRGB, but covering more chromaticies
- HDR (high dynamic range): encode brightness (Y) with higher resolution

typical image signal processing (ISP) of a camera:

1) sensor reads total radiance filtered trough 3 spectral fillers (effectively like CIE RGB)
2) ISO gain amplifies the signal (for dark environments) and temp.+position dependence is compensated
3) demosaicing gets RGB values for pixels from Bayer mosaic sensor values
4) optional denoising is performed
5) convert sensor RGB to CIE XYZ (with color calibration)
6) optional white balancing is done (shift colors such that they appear as if under white light)
7) optional other fillers (like tone mapping to map dynamic range)
8) convert CIE XYZ to sRGB
9) save image to file (optionally compress with e.g. JPEG)

lens camera optics

light as wave
light as particle
light as ray

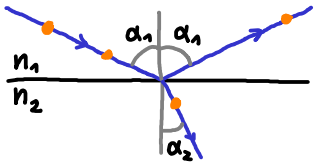


$f = c/\lambda$ freq. = speed/wavelen.

(more formulas in physics I recap.)

assumptions:
- geometric optics (no quantum effects like diffraction)
- thin lenses approx.
- paraxial approx.
  - ↳ Gaussian lens formula

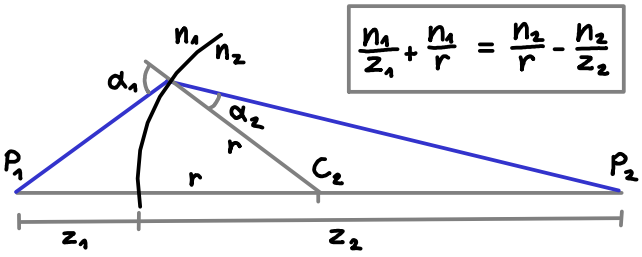
• refraction



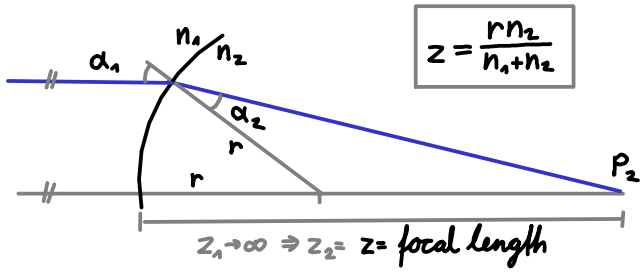
$c_{1,2} = c_0(\text{vacuum})/n_{1,2}$

$\alpha_2 = \sin^{-1}\left(\frac{n_1}{n_2}\cdot\sin(\alpha_1)\right)$

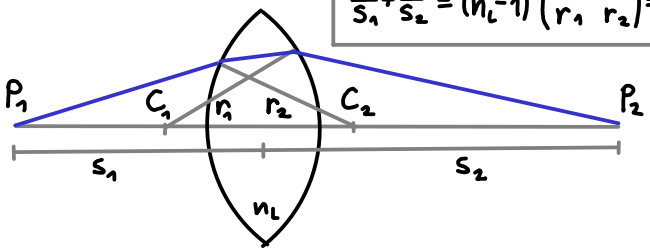
(see fresnell for par. proportions)



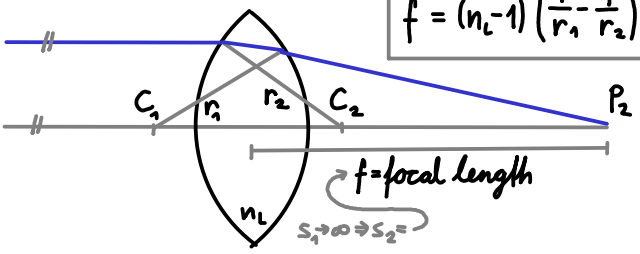
$$\frac{n_1}{z_1} + \frac{n_1}{r} = \frac{n_2}{r} - \frac{n_2}{z_2}$$



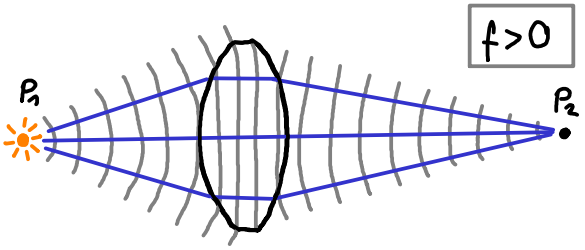
$$z = \frac{r n_2}{n_1 + n_2}$$



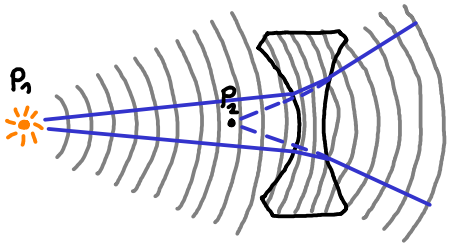
$$\frac{1}{s_1} + \frac{1}{s_2} = (n_L - 1)\left(\frac{1}{r_1} - \frac{1}{r_2}\right) = f$$



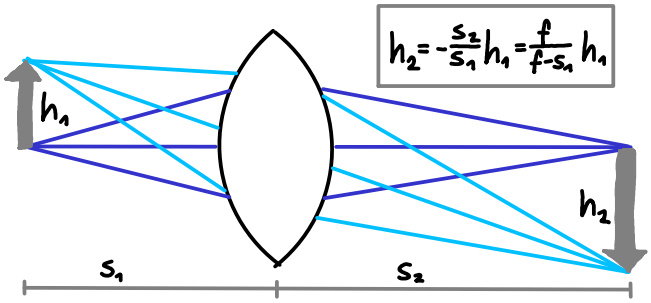
$$f = (n_L - 1)\left(\frac{1}{r_1} - \frac{1}{r_2}\right)$$



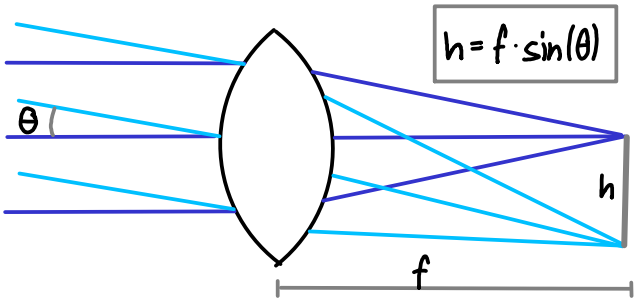
$f > 0$



$f < 0$



$$h_2 = -\frac{s_2}{s_1} h_1 = \frac{f}{f - s_1} h_1$$



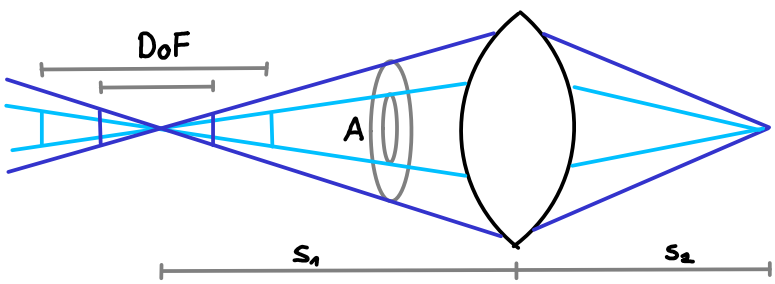
$$h = f \cdot \sin(\theta)$$

combining lenses: $\frac{1}{f_{tot}} = \frac{1}{f_1} + \frac{1}{f_2}$; $P_{tot} = P_1 + P_2$ (P lens power: $P = 1/f$)

aperture f-number: $N = f/A$
(size of hole in front of lens)

depth of field: $\text{DoF} \approx 2 s_1^2 N c / f^2$ (pixel size)
(change in s_2 without blurring pixels)

$A \downarrow \Rightarrow \text{DoF} \uparrow$ but less light

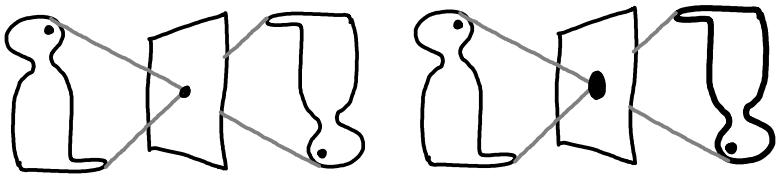


defects of lens cameras

- **spherical aberration**: spherical lenses only appoximately focus. parabolic lens fixes this.
- **chromatic aberration**: different wavelengths refract differently and have different focal lengths. fix for 2 wavelengts with achromatic doublet.
- **vignetting**: less light at periphery of image.
- **field curvature**: focus is on a sphere (radius s_i) not a plane → problem for flat sensor.
- **radial distortion**: magnification changes with image position
- **veiling glare**: glare from unintended reflections between lenses

pinhole camera optics

- infinite depth of field (every depth is in focus)
- pinhole small → image sharp + dimm
  pinhole big → image bright + blury



light fields