Equations

Fourier transform: $F(\omega) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i \omega x} \, \mathrm{d}x$
Inverse transform: $f(x) = \int_{-\infty}^{\infty} F(\omega) e^{2\pi i \omega x} \, \mathrm{d}\omega$
Euler's formula: $e^{ix} = \cos x + i \sin x$
Poisson distribution: $p(k; \lambda) = \frac{\lambda^k e^{-\lambda}}{k!}, \mu = \lambda, \sigma = \sqrt{\lambda}$
Convolution: $(f * g)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j) I(x - i, y - j)$
lerp: $lerp(x, v_0, v_1) = v_0 + t(v_1 - v_0)$
$\mathbf{E}[X + c] = \mathbf{E}[X] + c$
$\mathbf{E}[X + Y] = \mathbf{E}[X] + \mathbf{E}[Y]$
$\mathbf{E}[aX] = a\mathbf{E}[X]$
$\mathrm{Var(X)} = \mathbf{E}[X - \mathbf{E}[X]^2] = \mathbf{E}[x^2] - (\mathbf{E}[x])^2$
$\mathrm{Var(X + a)} = \mathrm{Var(X)}$
$\mathrm{Var(aX)} = a^2\mathrm{Var(X)}$
$\mathrm{Var(aX + bY)} = a^2\mathrm{Var(X)} + b^2\mathrm{Var(Y)} + 2ab\mathrm{Cov(X, Y)}$

Rasterization

Antialiasing: Removing frequencies above the Nyquist frequency (2 times the highest frequency) before sampling (Filtering before sampling)
Line equation: $L(x, y) = Ax + By + C$. On line: $L(x, y) = 0$, on right: $L(x, y) > 0$, on left: $L(x, y) < 0$. A point is inside a convex polygon if line tests for all lines (going clockwise) is greater than 0.

Transforms

Transformation Order:
- Object coordinates: Apply modeling transforms
- World (scene) coordinates: Apply viewing transform
- Camera (eye) coordinates: Apply perspective transform and homogeneous division
- Normalized device coordinates: Apply 2D screen transform
- Screen coordinates: Rasterization

Homogeneous coordinates: Points are defined as $(x, y, z, 1)$ and vectors by $(x, y, z, 0)$.
Changing reference frame: New coordinate frame defined by origin and coordinate axes:

Scaling

$$\begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Translation

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Shear

$$\begin{bmatrix} 1 & \lambda & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Standard camera: Located at origin, pointing down negative z-axis, up vector is y axis. $r = v \times u$.
To standard camera (inverse)

$$\begin{bmatrix} r_x & r_y & r_z & 0 \\ u_x & u_y & u_z & 0 \\ -v_x & -v_y & -v_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -e_x \\ 0 & 1 & 0 & -e_y \\ 0 & 0 & 1 & -e_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation by angle α around axis n:

$$\cos(\alpha)\mathbf{I} + (1 - \cos(\alpha))nn^T + \sin(\alpha) \begin{bmatrix} 0 & n_z & -n_y \\ -n_z & 0 & n_x \\ n_y & -n_x & 0 \end{bmatrix}$$

Hierarchical Transform:
Projective transform: Project some point in the scene $(x, y, z)^T$ to a point on the image plane. If the center of projection is at $(0, 0, 0)^T$ and the image plane is at $z = d$, then $(x, y, z)^T \to (xd/z, yd/z, d)^T$.

Requires division by z, so use homogeneous coordinates:

$$q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} xd/z \\ yd/z \\ d \\ 1 \end{bmatrix}$$

Perspective projection: parameterized by *fovy*: vertical angular field of view, *aspect*: width/height of field of view, *near*: depth of near clipping plane, *far*: depth of far clipping plane

Derived quantities: $top = near \cdot \tan(fovy)$, $bottom = -top$, $right = top \cdot aspect$, $left = -right$

Convert from camera coordinates to normalized device coordinates (NDC) by mapping the view volume frustum into a cube, where $(left, bottom, -near) \to (-1, -1, -1)$ and $(right, top, -far) \to (1, 1, 1)$. Linear transformation in homogeneous coordinates.

Geometry

Implicit geometry: Surface defined where $f(x, y, z) = 0$. Can test if point is inside or outside easily, but hard to sample (what points lie on the surface?) Description can be compact, good for ray-to-surface intersections. No sampling error, easy to handle changes in topology. Hard to model complex shapes.
Explicit geometry: All points given directly, $f : \mathbb{R}^2 \to \mathbb{R}^3; (u, v) \to (x, y, z)$. Sampling easy, but hard to tell if point is inside/outside surface.
Topological validity: A 2D manifold is a surface that when cut with a sphere always yields a disk. Mesh manifolds always have the properties: edge connects two faces and two vertices, face consists of ring of edges and vertices, vertex consists of ring of edges and faces. $F - E + V = 2$ holds for a surface topologically equivalent to a sphere.

```
struct Halfedge {
    Halfedge *twin;
    Halfedge *next;
    Vertex *vertex;
    Edge *edge;
    Face *face; }
```

```
struct Vertex {
    Halfedge *halfedge; }
struct Edge {
    Halfedge *halfedge; }
struct Face {
    Halfedge *halfedge; }
```

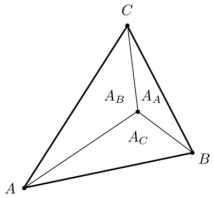
Loop subdivision: Split edges of original mesh in any order, then flip new edges that touch a new and old vertex.

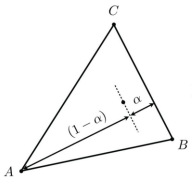
Catmull-Clark vertex update: Converts an existing mesh to quads. Extraordinary points can increase during first iteration, but stays constant afterwards. Add a face point to the centroid of each face, add edge point (average of neighboring vertices and face points), connect face point to each edge point, update original points such that (n is valence of P, F is average of face points, R is average of edge points):

$$P = \frac{F + 2R + (n - 3)P}{n}$$

Barycentric Coordinates: linearly interpolate values at vertices for a point in the triangle

$$(x, y) = \alpha A + \beta B + \gamma C$$
$$\alpha + \beta + \gamma = 1$$





$$\alpha = \frac{-(x - x_B)(y_C - y_B) + (y - y_B)(x_C - x_B)}{-(x_A - x_B)(y_C - y_B) + (y_A - y_B)(x_C - x_B)}$$
$$\beta = \frac{-(x - x_C)(y_A - y_C) + (y - y_C)(x_A - x_C)}{-(x_B - x_C)(y_A - y_C) + (y_B - y - C)(x_A - x_C)}$$
$$\gamma = 1 - \alpha - \beta$$

$\alpha = \frac{A_A}{A_A + A_B + A_C}$ $\beta = \frac{A_B}{A_A + A_B + A_C}$ $\gamma = \frac{A_C}{A_A + A_B + A_C}$

Texture sampling: Affine screen-space interpolation doesn't work, because linear interpolation in world coordinates yields nonlinear interpolation in screen coordinates. *Magnification*: Each pixel is a small part of the texel. *Minification*: each pixel includes many texels, leading to aliasing.
Mipmaps: Filter before sampling; use resolution that matches screen sampling rate. Estimate texture footprint using texture coordinates of neighboring screen samples, then use that to figure out which level of mipmap to use.

$$D = \log_2 max\left(\sqrt{\left(\frac{du}{dx}\right)^2 + \left(\frac{dv}{dx}\right)^2}, \sqrt{\left(\frac{du}{dy}\right)^2 + \left(\frac{dv}{dy}\right)^2}\right)$$

Anisotropic filtering: Mipmaps change depending on viewing angle. One way is to generate more mipmaps with variations on both x and y, instead of just taking the max of both.
Accomodation: Changes optical power to focus different distances.
Vergence: rotation of eye in socket to ensure projection of object centered on retina
Hermite matrix:

$$\begin{aligned} P(t) &= at^3 + bt^2 + ct + d \\ &= \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \end{bmatrix} \\ &= H_0(t)h_0 + H_1(t)h_1 + H_2(t)h_2 + H_3(t)h_3 \end{aligned}$$

Bezier Curves

$$B_i^n(t) = \binom{n}{i} t^i (1 - t)^{n - i}$$

$$\begin{aligned} P(t) &= \sum_{i=0}^{3} P_i B_i(t) \\ &= (1 - t)^3 P_0 + 3t(1 - t)^2 P_1 + 3t^2(1 - t)P_2 + t^3 P_3 \\ &= \begin{bmatrix} (1 - t)^3 & 3t(1 - t) & 3t^2(1 - t) & t^3 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix} \\ &= \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix} \end{aligned}$$

de Casteljau Algorithm:

$$\begin{aligned}B(t) &= (1-t)((1-t)P_0 + tP_1) + t((1-t)P_1 + tP_2)\\ &= (1-t)^2P_0 + 2t(1-t)P_1 + t^2P_3\\ &= \begin{bmatrix}(1-t)^2 & 2t(1-t) & t^2\end{bmatrix}\begin{bmatrix}P_0\\P_1\\P_2\end{bmatrix}\\ &= \begin{bmatrix}1 & t & t^2\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\-2 & 2 & 0\\1 & -2 & 1\end{bmatrix}\begin{bmatrix}P_0\\P_1\\P_2\end{bmatrix}\end{aligned}$$

$$b_{i,j}^{r,r} = \begin{bmatrix}1-u & u\end{bmatrix}\begin{bmatrix}b_{i,j}^{r-1,r-1} & b_{i,j+1}^{r-1,r-1}\\ b_{i+1,j}^{r-1,r-1} & b_{i+1,j+1}^{r-1,r-1}\end{bmatrix}\begin{bmatrix}1-v\\v\end{bmatrix}$$

Color/Radiometry

Solid angle: $\Omega = \frac{A}{r^2}$, sphere has 4π steradians.
$dA = (rd\theta)(r\sin\theta d\phi)$, $d\omega = \sin\theta d\theta d\phi$,
$\Omega = \int_{S^2} d\omega = \int_0^\phi \int_0^\theta \sin\theta d\theta d\phi$.
Radiant flux: $\Phi \equiv \frac{dQ}{dt}$ [W] [lm], energy emitted over time (power)
Radiant intensity: $I(\omega) \equiv \frac{d\Phi}{d\omega}$ $[W/sr]$ $[lm/sr = cd]$, light emitted from a source, flux per unit solid angle
Irradiance: $E(x) \equiv \frac{d\Phi(x)}{dA}$ $[W/m^2]$ $[lm/m^2 = lux]$, light falling on a surface, flux per unit area. Project area of object on unit sphere, project onto plane.
Radiance: $L(p,\omega) \equiv \frac{d^2\Phi(p,\omega)}{d\omega dA\cos\theta}$ $[W/(sr\cdot m^2)]$
$[cd/m^2 = lm/(sr\cdot m^2) = nit]$, light travelling along a ray, flux per unit area per unit solid angle. Incident and exiting surface radiance are different (depends on material of surface)
Lambert's cosine law: Light per unit area is proportional to $\cos(\theta) = I \cdot n$, where θ is the angle between the normal and the light source and I is a vector from hit point to light source.

Rendering

Lambertian (Diffuse) shading: Shading independent of view direction. $L_d = k_d(I/r^2)max(0, n\cdot l)$
Specular shading: Close to mirror at the half-vector near normal. $h = \text{bisector}(v,l) = \frac{v+l}{\|v+l\|}$
$L_s = k_s(I/r^2)\max(0, n\cdot h)^p$
Ambient shading: doesn't depend on anything, adds constant color to account for disregarded illumination and fill in black shadows. Not physically accurate, since it's not based on any incoming radiance. $L_a = k_a I_a$
Ray: $r(t) = o + td, 0 \le t < \infty$
General implicit surface: $p : f(p) = 0$
Substitute ray equation: $f(o + td) = 0$
Sphere: $(p-c)^2 - R^2 = 0$

$$t = \frac{(p-o)\cdot N}{d\cdot N} \to \frac{p_x - o_x}{d_x}$$

Monte Carlo Integration:

$$F_N = \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}$$

Inversion method: Given $P(x) = Pr(X < x)$, solve for $x = P^{-1}(\xi)$. Need to know the integral of the PDF $p(x)$ for this to work. Example:

$$p_\lambda(x) = \lambda e^{-\lambda x}$$

$$\begin{aligned}F(t) = P_\lambda(X \le t) &= \int_0^t \lambda e^{-\lambda x}dx\\ &= 1 - e^{-\lambda t}\end{aligned}$$

$$F^{-1}(\xi) = \frac{-\log(1-\xi)}{\lambda}$$

Bidirectional Reflectance Distribution (BRDF):
Non-negative (always returns a positive value for any input ray), respects linearity, reciprocity principle:
$f_r(w_i \to w_r) = f_r(w_r \to w_i)$, conserves energy.
Reflection equation:

$$L_r(p,\omega_r) = \int_{H^2} f_r(p,\omega_i \to \omega_r)L_i(p,\omega_i)\cos\theta_i d\omega_i$$

Transport function: $tr(p,\omega)$ returns first surface intersection point in the scene along ray (p,ω)
Radiance invariance: $L_o(tr(p,\omega_i), -\omega_i) = L_i(p,\omega_i)$
Rendering equation:

$$L_o(p,\omega_o) = L_e(p,\omega_o) + \int_{H^2} f_r(p,\omega_i \to \omega_o)L_o(tr(p,\omega_i),-\omega_i)\cos\theta_i d\omega_i$$

Animation

Inverse kinematics: Solve for joint angles satisfying position of end-effector

$$\theta_2 = \arccos\left(\frac{p_z^2 + p_x^2 - l_1^2 - l_2^2}{2l_1l_2}\right)$$

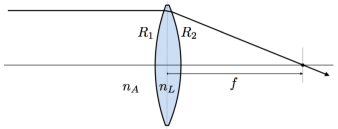
$$\theta_1 = \frac{-p_z l_2\sin(\theta_2) + p_x(l_1 + l_2\cos(\theta_2))}{p_x l_2\sin(\theta_2) + p_z(l_1 + l_2\cos(\theta_2))}$$

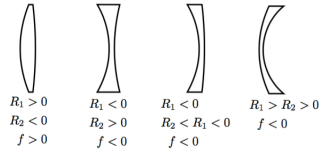
Forward Euler: $x(t+\Delta t) = x(t) + \Delta(t)\cdot v(x(t),t)$,
$v(t+\Delta t) = v(t) + \Delta t\cdot f(x,v,t)/m$
Midpoint method: $x_{mid} = x(t) + \Delta t/2\cdot v(x(t),t)$,
$x(t+\Delta t) = x(t) + \Delta t\cdot v(x_{mid},t)$
Implicit Euler: $(t+\Delta t) = x(t) + \Delta(t)\cdot v(x(t+\Delta t),t)$
Adaptive step size: Compute x_T, Euler step of size T. Computer $x_{T/2}$, two Euler steps of size T/2. Compute error, check if greater than threshold, reduce step size if necessary.

Cameras

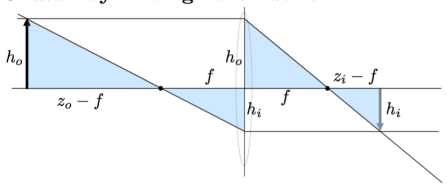
Exposure: $H = T \times E$, exposure [controlled by shutter] = (time)(irradiance) [controlled by lens aperture and focal length]
Aperture number (f-stop): $N = \frac{f}{D}$, focal length f, aperture diameter D. Irradiance proportional to square of D, inverse square f. Multiplying f-stop by $\sqrt{2}$ doubles exposure. Increasing 2 f-stops doubles depth of field.
ISO (gain): multiply signal before analog/digital conversion. Linear effect: ISO 200 needs half as much light as ISO 100.
Field of view: $2\arctan(\frac{h}{2f})$, dependent on sensor size, focal length
Depth of field: aperture, focal length, object distance
Exposure: aperture, shutter, ISO
Motion blur: shutter
Grain/noise: ISO
Phase-detection autofocus: ray bundles from target object converge to points at different depths in camera depending on lens focus. Distance between phase-detect images correlates to distance in focus to target object.
Lens maker's equation: f, R1, R2 are signed, positive pointing to the right, radii point from surface to center

$$\frac{1}{f} = \left(\frac{n_L}{n_A} - 1\right)\left(\frac{1}{R_1} - \frac{1}{R_2}\right)$$



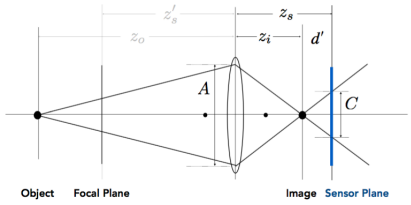


Gauss' ray-tracing construction:



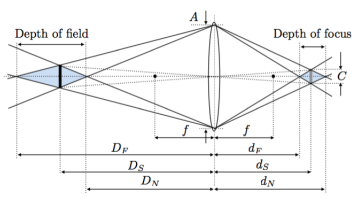
$\frac{h_o}{z_o - f} = \frac{h_i}{f}$ $\frac{h_o}{f} = \frac{h_i}{z_i - f}$ $\frac{1}{f} = \frac{1}{z_i} + \frac{1}{z_o}$ $m = \frac{h_i}{h_o} = \frac{z_i}{z_o}$

Circle of confusion



$$C = A\left(\frac{d'}{z_i}\right) = A\frac{|z_s - z_i|}{z_i} = \frac{f}{N}\frac{|z_s - z_i|}{z_i}$$

Hyperfocal distance: Maximizes depth of field (such that infinity is at limit of acceptable sharpness)



$\frac{d_N - d_S}{d_N} = \frac{C}{A}$
$\frac{d_S - d_F}{d_F} = \frac{C}{A}$
$N = \frac{f}{D}$
$\frac{1}{D_F} + \frac{1}{d_F} = \frac{1}{f}$
$\frac{1}{D_S} + \frac{1}{d_S} = \frac{1}{f}$
$\frac{1}{D_N} + \frac{1}{d_N} = \frac{1}{f}$

$$\text{DOF} = D_F - D_N$$

$$D_F = \frac{D_S f^2}{f^2 - NC(D_S - f)} \qquad D_N = \frac{D_S f^2}{f^2 + NC(D_S - f)}$$

FSI vs BSI: Back-side illuminated puts metal below diode below lens, whereas front-side illuminated puts metal in-between lens and diode. BSI makes more light hit the diode (doesn't have to make it through a tunnel of metal).
Signal-to-noise (SNR): $\frac{\mu}{\sigma} = \frac{\lambda}{\sqrt{\lambda}} = \sqrt{\lambda}$

Light-field camera: Uses radiance flowing along every ray (4D) rather than irradiance at every pixel on plane (2D). Microlenses cover multiple pixels. Light-field data preserves directional intensity information, provides flexibility over focus/aberrations
VR displays: Add stereo (different left/right eye views), parallax (different views as user moves)