Sampling

Antialiasing: Removing frequencies above the Nyquist frequency (2 times the highest frequency) before sampling. Done by 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.
Fourier transform: $F(\omega) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i \omega x}\, dx$
Inverse transform: $f(x) = \int_{-\infty}^{\infty} F(\omega) e^{2\pi i \omega x}\, d\omega$
Euler's formula: $e^{ix} = \cos x + i \sin x$

Transforms

Homogeneous coordinates: Points are defined as $(x, y, 1)$ and vectors by $(x, y, 0)$. Points and vectors don't reside on the same xy plane.
Affine transformations: Consist of linear map and a translation.
Camera transformations: Given eye point e, up vector u, view direction v, where u and v are orthonormal, find right vector $r = v \times u$. Transform camera (e, r, u, v) to standard camera located at the origin, looking down the negative z-axis, where the up vector is the y-axis.

Transform 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

Basic Transformations

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

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

Coordinate System Transform

Frame-to-world transformation converts frame axes to world axes centered at origin with x mapped to u and y mapped to v.

$$F = \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{o} \end{bmatrix} = \begin{bmatrix} u_x & v_x & o_x \\ u_y & v_y & o_y \\ 0 & 0 & 1 \end{bmatrix}$$

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.

$$\begin{bmatrix} \frac{near}{right} & 0 & 0 & 0 \\ 0 & \frac{near}{top} & 0 & 0 \\ 0 & 0 & -\frac{far+near}{far-near} & \frac{-2(far \cdot near)}{far-near} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

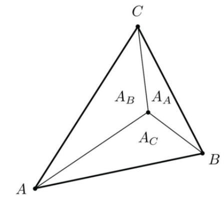
Texture Mapping

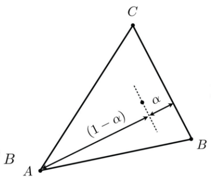
Barycentric Coordinates

Linearly interpolate values at vertices for a point in the triangle

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

$$\alpha + \beta + \gamma = 1$$





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

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

$$\gamma = 1 - \alpha - \beta$$

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

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.

$$D = \log_2 L$$

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

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.

Graphics Pipeline

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.
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. $L_a = k_a I_a$

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

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

Splines, Curves, and Surfaces

Cubic Polynomial Interpolation

$$\begin{aligned} P(t) &= at^3 + bt^2 + ct + d \\ P'(t) &= 3at^2 + 2bt + c \end{aligned}$$

Hermite matrix:

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

$$\begin{aligned} H_0(t) &= 2t^3 - 3t^2 + 1 \\ H_1(t) &= -2t^3 + 3t^2 \\ H_2(t) &= t^3 - 2t^2 + t \\ H_3(t) &= t^3 - t^2 \end{aligned}$$

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

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

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

Geometry Processing

Catmull-Clark Vertex Update

$$f = \frac{v_1 + v_2 + v_3 + v_4}{4} \qquad e = \frac{v_1 + v_2 + f_1 + f_2}{4}$$

Accelerating Ray Tracing

Ray: $r(t) = o + td, 0 \le t < \infty$
General implicit surface: $p : f(p) = 0$
Substitute ray equation: $f(o + td) = 0$
Sphere: $(p - c)^2 - R^2 = 0$

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

Radiometry and Photometry

Radiant Flux
Radiant Intensity: Power per unit solid angle emitted from a point light source.
Irradiance: Power per unit area (W/m^2)., follows Lambert's cosine law, irradiance falloff follows inverse square law
Radiance: Power per unit area per unit solid angle ($W/sr/m^2$).

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:

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

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

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

Expectation and Variance

$$\begin{aligned} \mathbf{E}[X + c] &= \mathbf{E}[X] + c \\ \mathbf{E}[X + Y] &= \mathbf{E}[X] + \mathbf{E}[Y] \\ \mathbf{E}[aX] &= a\mathbf{E}[X] \end{aligned}$$

$$\begin{aligned} \mathrm{Var(X + a)} &= \mathrm{Var(X)} \\ \mathrm{Var(aX)} &= a^2 \mathrm{Var(X)} \\ \mathrm{Var(aX + bY)} &= a^2 \mathrm{Var(X)} + b^2 \mathrm{Var(Y)} + 2ab\mathrm{Cov(X, Y)} \end{aligned}$$

Reflection and Materials

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.