

计算机图形学

变换 (Transformation)

二维变换

线性变换

定义

常见变换矩阵

缩放矩阵 (Scale Matrix)

对称矩阵 (Refletion Matrix)

错切矩阵 (Shear Matrix)

旋转矩阵 (Rotation Matrix)

仿射变换 (Affine Map)

齐次坐标

复合变换 (Composite Transforms)

三维变换

线性变换

缩放矩阵

平移矩阵

旋转矩阵

仿射变换

观测变换 (Viewing Transformation)

视图变换 (View/Camera Transformation)

投影变换 (Projection Transformation)

正交投影 (Orthographic Projection)

透视投影 (Perspective Projection)

视口变换 (Viewport Transformation)

坐标变换

切线空间=>世界空间

光栅化 (Rasterize)

反走样 (Antialiasing)

模糊处理

滤波

其它的抗锯齿方案:

超分辨率

可见性与遮挡 (Visibility/Occlusion)

深度缓存 (Z-Buffer)

着色 (Shading)

一种简单的着色模型: 冯氏反射 (Blinn-Phong Reflectance Model)

着色频率 (Shading Frequency)

插值 (Interpolation)

重心坐标 (Barycentric Coordinates)

插值的应用

双线性插值 (Bilinear Interpolation)

纹理 (Texture)

Mipmap

各向异性过滤 (Anisotropic Filtering)

Ripmap

EWA filtering

纹理的其它应用

存储环境光

伪造物体表明凹凸情况

补充

几何 (Geometry)

表达几何的方式

隐式几何 (Implicit Geometry)

Constructive Solid Geometry
Signed Distance Functions (距离场)
显式几何 (Explicit Geometry)
Point Cloud (点云)
Polygon Mesh
The Wavefront Object file (.obj) Format
曲线 (Curve)
 贝塞尔曲线 (Bezier Curve)
曲面 (Surface)
 贝塞尔曲面
 曲面处理
 Mesh Subdivision
 Mesh Simplification

阴影
 点光源的阴影 (Shadow Mapping)

光线追踪 (Ray Tracing)
 Pinhole camera Model
 Recursive (Whitted-Style) Ray Tracing
 光线与物体表面的交点
 光线与隐式表面的交点
 光线与显式表面的交点
 Triangle Mesh
 Uniform Spatial Partitions (Grids)
 Spatial Partitions
 Oct-Tree
 BSP-Tree
 KD-Tree
 Object Partitions & Bounding Volume Hierarchy (BVH)

辐射度量学 (Basic radiometry)
 双向反射分布函数 Bidirectional Reflectance Distribution Function (BRDF)
 蒙特卡洛积分 (Monte Carlo Integration)
 路径追踪 (Path Tracing)
 一种简单的处理策略
 解决光线数量爆炸增长的策略
 限制光线弹射
 提高效率，对较小光源精确采样
 更多

其他的现代光线追踪技术

材质 (Materials)
 反射与折射
 反射
 折射
 菲涅尔项 (Fresnel Term)

微表面材质 (Microfacet Material)
 Microfacet BRDF
 各向同性 (isotropic) 与各向异性 (anisotropic)
 BRDF的性质
 测量BRDF

Advanced Technology
 Advanced Light Transport
 双向路径追踪 (Bidirectional Path Tracing, BDPT)
 Metropolis Light Transport (MLT)
 光子映射 (Photon Mapping)
 Vertex Connection and Merging
 实时辐射度 (Instant Radiosity, IR)

Advanced Appearance Modeling
 非表面模型 (Non-surface models)
 Participating media 散射 (参与) 介质

Hair/fur/fiber
 Hair
 Fur
 颗粒材质 (Granular Material)
 Translucent Material
 表面模型 (Surface models)
 程序化生成 (Procedural appearance)

相机
 FOV
 曝光 (Exposure)
 光圈 (Aperture)
 快门 (Shutter)
 感光度 (ISO gain)
 薄透镜近似 (Thin Lens Approximation)
 Circle of Confusion (CoC)
 Ray Tracing for Defocus Blur (Thin Lens)
 景深 (depth of field)
 光场 (Light Field / Lumigraph)
 光场的引入
 光场的应用
 色彩 (Color)
 颜色空间 (Color Space)
 (s)RGB Color Space
 XYZ Color Space
 HSV Color Space (Hue-Saturation-Value/Brightness/Lightness)
 LAB Color Space
 减色系统
 色域 (Gamut)

变换 (Transformation)

MVP

- Model transformation (placing objects)
- View transformation (placing camera)
- Projection transformation

二维变换

线性变换

定义

对于变换后的 x' 、 y' ，如果满足以下关系

$$\begin{aligned}x' &= ax + bx \\y' &= cx + dy\end{aligned}$$

则称这种变换为线性变换。线性变换可以用矩阵方式表示为

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = M \times \begin{bmatrix} x \\ y \end{bmatrix}$$

其中M称为变换矩阵，有

$$M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

常见变换矩阵

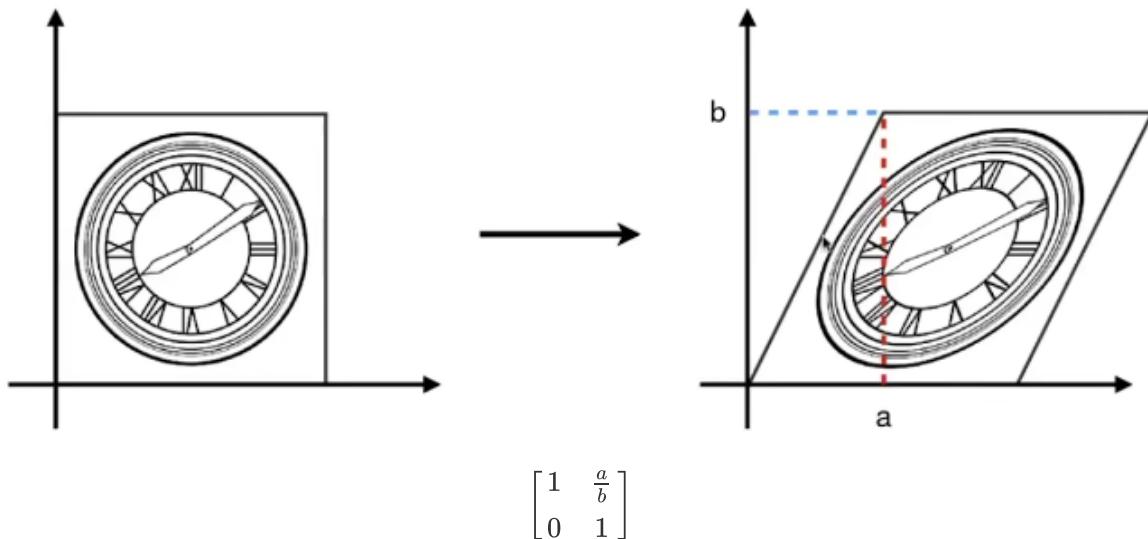
缩放矩阵 (Scale Matrix)

$$S(s_x, s_y) = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

对称矩阵 (Refletion Matrix)

关于 y 轴对称 $\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$ 关于 x 轴对称 $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

错切矩阵 (Shear Matrix)



$$\begin{bmatrix} 1 & \frac{a}{b} \\ 0 & 1 \end{bmatrix}$$

旋转矩阵 (Rotation Matrix)

$$R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

R_θ 是正交矩阵。

- 由三角函数性质可知 $R_{-\theta} = R_\theta^T$
- 由定义知 $R_{-\theta} = R_\theta^{-1}$

仿射变换 (Affine Map)

当引入平移(Translation)时，图形的变换不再是简单的线性变换。对于以下关系

$$\begin{aligned} x' &= ax + by + t_x \\ y' &= cx + dy + t_y \end{aligned}$$

不能简单地用矩阵乘法表示，而需要引入加法。

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

称以上变换为仿射变换。

齐次坐标

为消除平移的特例，引入齐次坐标的概念。

令 $2D\ point = (x, y, 1)^T$, $2D\ vector = (x, y, 0)$

此时，矩阵的平移可以表示为

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix}$$

对于 $point + point$ 会出现的第三位非0、1的数，我们做以下规定

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \text{ is the } 2D\ point \begin{bmatrix} x/w \\ y/w \\ 1 \end{bmatrix}, w \neq 0$$

由此，可以发现两个点的和其实是它们的中点。

可以将仿射变换用齐次坐标写作下面的形式：

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

复合变换 (Composite Transforms)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = A_n A_{n-1} A_{n-2} \dots A_1 \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

观察仿射变换公式可以发现，进行仿射变换时先进行线性变换后进行平移，所以

$$T_{(t_x, t_y)} \times \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

三维变换

线性变换

缩放矩阵

$$S(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

平移矩阵

$$T(t_x, t_y, t_z) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

旋转矩阵

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_y(\alpha) = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_z(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

绕一通过原点的轴 \vec{n} 转过，有如下矩阵

$$R(n, \alpha) = \cos(\alpha) + (1 - \cos(\alpha))\vec{n}\vec{n}^T + \sin(\alpha) \begin{bmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{bmatrix}$$

其中， $\mathbf{n} = [n_x, n_y, n_z]^T$

仿射变换

对二维变换直接增加一个维度，基本性质保持不变。

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

观测变换 (Viewing Transformation)

视图变换 (View/Camera Transformation)

假设有这样一个相机

- 位置Posion \vec{e}
- Look-at / gaze direction \hat{g}
- Up direction \hat{t}

现将相机移动到原点，使其面向-Z方向，向上为Y方向（约定俗成的相机）。

$$M_{view} = R_{view} T_{view}$$

$$T_{view} = \begin{bmatrix} 1 & 0 & 0 & -x_e \\ 0 & 1 & 0 & -y_e \\ 0 & 0 & 1 & -z_e \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

直接考虑旋转 \hat{g} 到-Z和 \hat{t} 到Y并不方便，由于旋转矩阵是正交矩阵，不妨考虑它的逆。

由

$$\hat{t} = R_{view}^{-1} \times \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, -\hat{g} = R_{view}^{-1} \times \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

可以得到

$$R_{view}^{-1} = \begin{bmatrix} x_{\hat{g} \times \hat{t}} & x_{\hat{t}} & x_{-\hat{g}} & 0 \\ y_{\hat{g} \times \hat{t}} & y_{\hat{t}} & y_{-\hat{g}} & 0 \\ z_{\hat{g} \times \hat{t}} & z_{\hat{t}} & z_{-\hat{g}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

由此

$$R_{view} = \begin{bmatrix} x_{\hat{g} \times \hat{t}} & y_{\hat{g} \times \hat{t}} & z_{\hat{g} \times \hat{t}} & 0 \\ x_{\hat{t}} & y_{\hat{t}} & z_{\hat{t}} & 0 \\ x_{-\hat{g}} & y_{-\hat{g}} & z_{-\hat{g}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

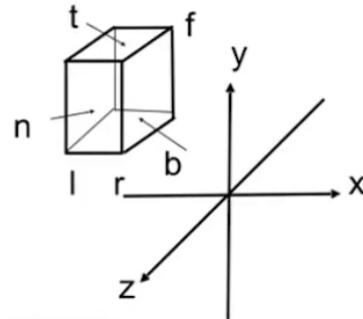
至此，就得到了视图转换的矩阵。再将对模型进行同样的转换，即完成了视图转换。由于视图转换也涉及到模型的转换，因此也被称为模型视图转换（ModelView Transformation）。

投影变换 (Projection Transformation)

正交投影 (Orthographic Projection)

我们将一个中心在原点的边长为2的且各边分别与x、y、z、轴平行的正方体成为标准立方体 (canonical cube)。

进行正交投影时，定义一个长方体 $[l, r] \times [b, t] \times [f, n]$ ：（注意由于摄像机朝着-Z，远处的值更小）

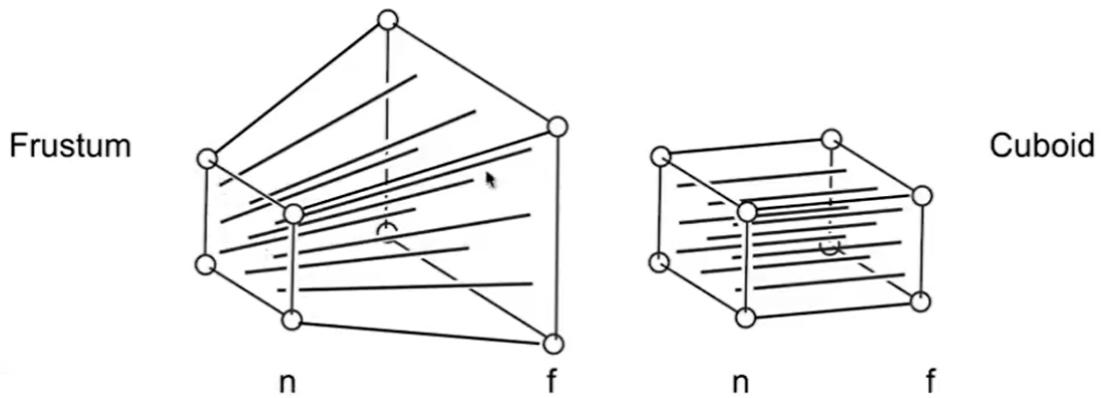


对它进行变换使之成为标准立方体，变换的矩阵为：

$$M_{ortho} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & \frac{2}{n-f} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -\frac{r+l}{2} \\ 0 & 1 & 0 & -\frac{t+b}{2} \\ 0 & 0 & 1 & -\frac{n+f}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

OpenGL采用左手系是为了让远处的值更大

透视投影 (Perspective Projection)



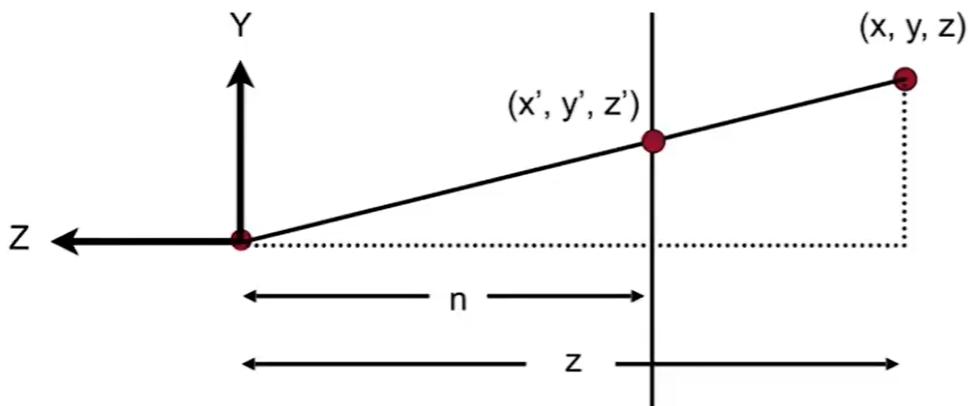
先对透视投影进行处理使其转变为正交投影，再进行正交投影。

如图所示，为将透视投影转变为正交投影，将透视的光线形成的立体图进行挤压，变成一个长方体。记变换前的点坐标为 $[x, y, z]^T$ ，变换后的点坐标为 $[x', y', z']^T$ 。

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = M_{persp \rightarrow ortho} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- 推导过程

从侧面观察可以得到如下关系。



由相似关系易得， $x' = \frac{n}{z}x$, $y' = \frac{n}{z}y$ 。

易知，对于近平面n上的任一点，变换后的点位置不变。对于远平面上的任一点，变换后的 $z'=z$ 。

需要注意的是，对于夹在两平面之间的点，它的 $z' \neq z$ 。

$$\begin{aligned} \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} &\Leftrightarrow \begin{bmatrix} \frac{n}{z}x \\ \frac{n}{z}y \\ \text{unknown} \\ 1 \end{bmatrix} \Leftrightarrow \begin{bmatrix} nx \\ ny \\ \text{unknown} \\ z \end{bmatrix} \\ \begin{bmatrix} nx \\ ny \\ \text{unknown} \\ z \end{bmatrix} &= M_{persp \rightarrow ortho} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \end{aligned}$$

由此可得

$$M_{persp \rightarrow ortho} = \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ ? & ? & ? & ? \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

而由近平面上点位置变化关系，可得

$$M_{persp \rightarrow ortho} \begin{bmatrix} x \\ y \\ n \\ 1 \end{bmatrix} = \begin{bmatrix} nx \\ ny \\ n^2 \\ n \end{bmatrix}$$

所以 $M_{persp \rightarrow ortho}$ 的第三行为 $(0 \ 0 \ A \ B)$ ，且有

$$An + B = n^2 \quad (1)$$

由远平面上中心点的位置变化关系可得

$$M_{persp \rightarrow ortho} \begin{bmatrix} 0 \\ 0 \\ f \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ f^2 \\ f \end{bmatrix}$$

于是有

$$Af + B = f^2 \quad (2)$$

联立(1)式、(2)式可得

$$\begin{aligned} A &= n + f \\ B &= -nf \end{aligned}$$

所以

- 结论

$$M_{persp \rightarrow ortho} = \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -nf \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

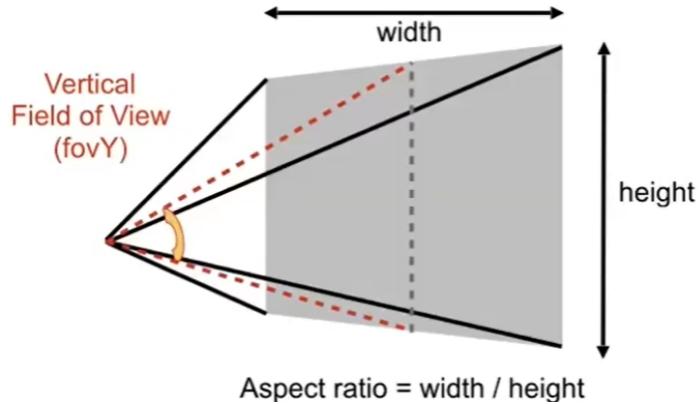
$$M_{persp} = M_{ortho} M_{persp \rightarrow ortho}$$

带入两平面中点的坐标 $[0, 0, \frac{n+f}{2}, 1]$ 可以发现经变换后该点的 z' 更接近远平面。

视口变换 (Viewport Transformation)

对于压缩成标准立方体之前的投影面，定义如下量

- Aspect ratio = width / height
- vertical field-of-view(fovY)



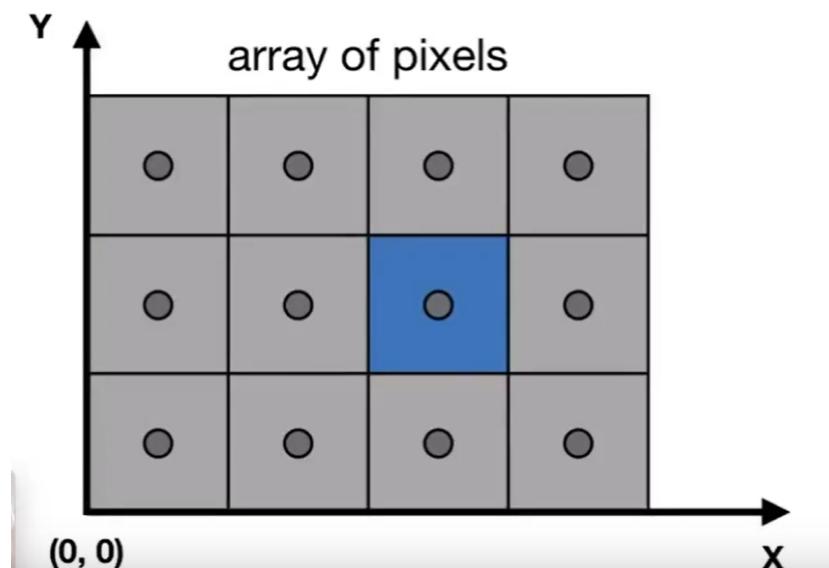
有

$$\tan \frac{fovY}{2} = \frac{t}{|n|}$$

$$aspect = \frac{r}{t}$$

通过视口变换将标准立方体变换到屏幕上。

定义屏幕如下



- 屏幕坐标从(0, 0)到(width - 1, height - 1)
- 像素点的中心为(x + 0.5, y + 0.5)
- 屏幕覆盖范围为(0, 0)到(width, height)

则视口变换矩阵为

$$M_{viewport} = \begin{bmatrix} \frac{width}{2} & 0 & 0 & \frac{width}{2} \\ 0 & \frac{height}{2} & 0 & \frac{height}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

坐标变换

切线空间=>世界空间

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}_{3 \times 1} = \begin{bmatrix} T_x & B_x & N_x \\ T_y & B_y & N_y \\ T_z & B_z & N_z \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{3 \times 1}$$

- 其中 $M_{TBN} = \begin{bmatrix} T_x & B_x & N_x \\ T_y & B_y & N_y \\ T_z & B_z & N_z \end{bmatrix}$ 被称为TBN矩阵，N为法向量，

$$T = \begin{bmatrix} N_x * N_y / \sqrt{(N_x^2 + N_z^2)} \\ \sqrt{N_x^2 + N_z^2} \\ N_z * N_y / \sqrt{N_x^2 + N_z^2} \end{bmatrix}, \quad B = N \times T$$

光栅化 (Rasterize)

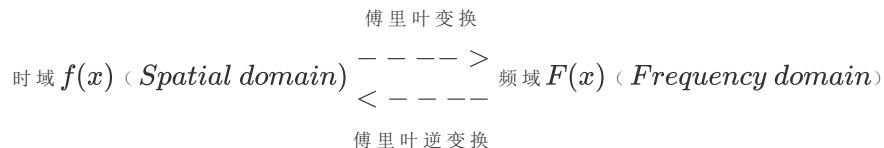
光栅化：将图形绘制在屏幕上。

- 对图形进行采样 (Sampling)，像素中心点在图形内部的绘上颜色。
 - 对所有像素进行遍历
 - 生成Bounding Box，对Box内的像素点进行遍历。
 - 找到最底下的顶点后一层一层沿着边扫描上去。
- 基于三角形的良好性质
- 采样频率过低会导致走样 (Aliasing)，产生锯齿 (Jaggies)

反走样 (Antialiasing)

对图形先进行模糊处理，再采样。

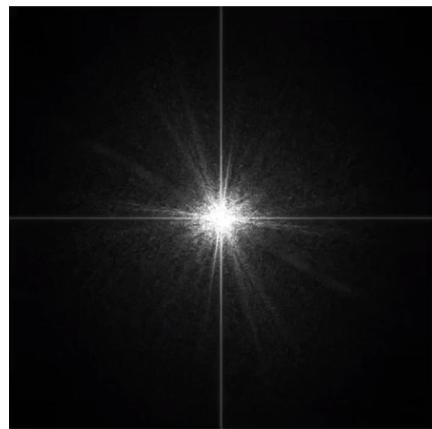
模糊处理



时域



转换为频域

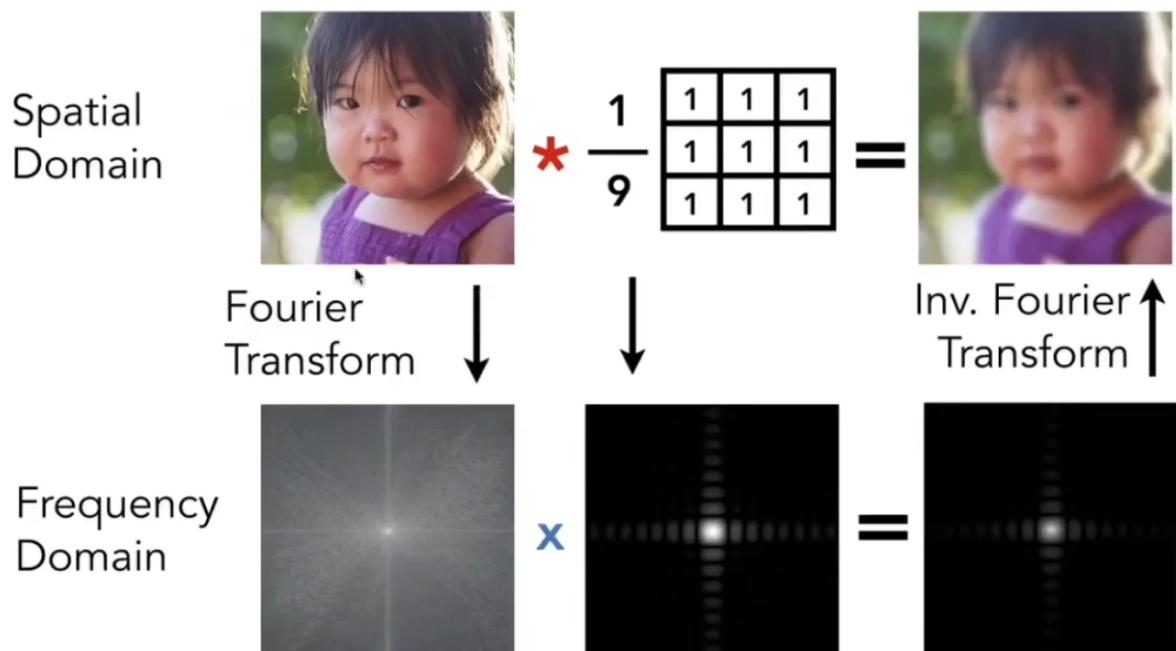


频域图中心表示低频，周围表示高频，亮度表示信息的量

十字线是由于做变换时图形四方延拓导致的

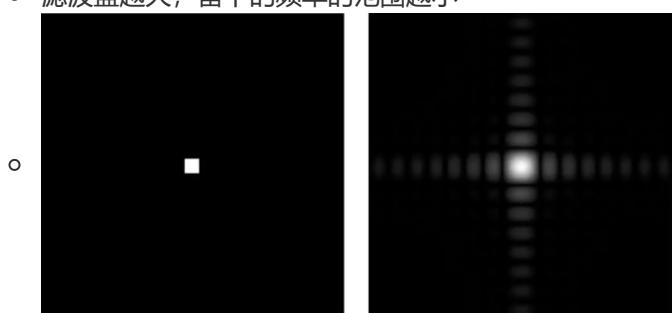
滤波

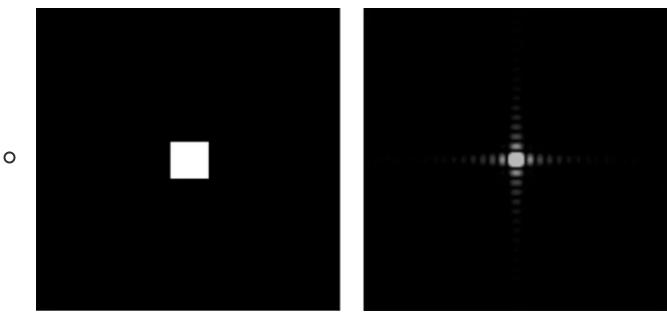
- 过滤掉特定的频率信息。
- 高通滤波 (High-pass filter)：只留下边界信息。
- 低通滤波 (Low-pass filter)：抹去边界信息，起到模糊的效果。
- 卷积 (Convolution)：用一个滤波盒，滤波盒与图像对应部分进行点乘，计算结果写回该部分中心，即取了一个区域的加权平均，能够起到模糊的作用。
- 定理：时域的卷积等于频域的乘积，时域上的乘积意味着频域上的卷积。



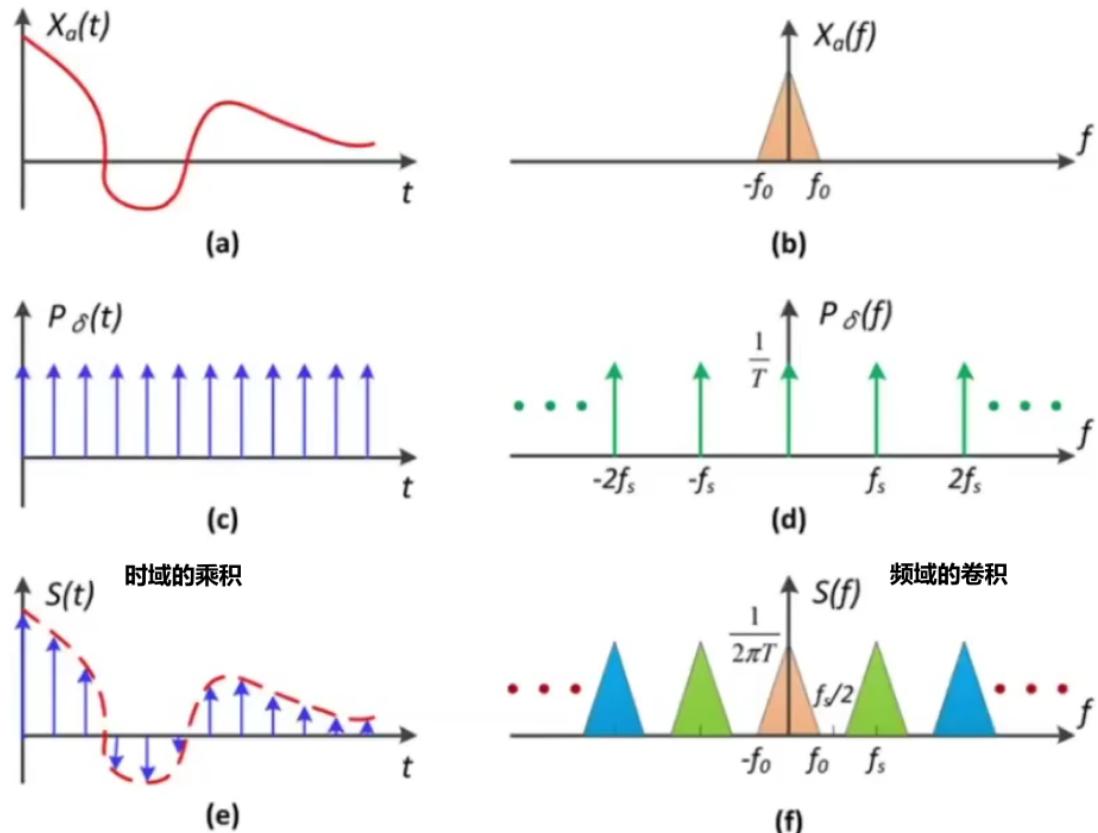
- 滤波盒时域与频域的对应关系：

- 滤波盒越大，留下的频率的范围越小



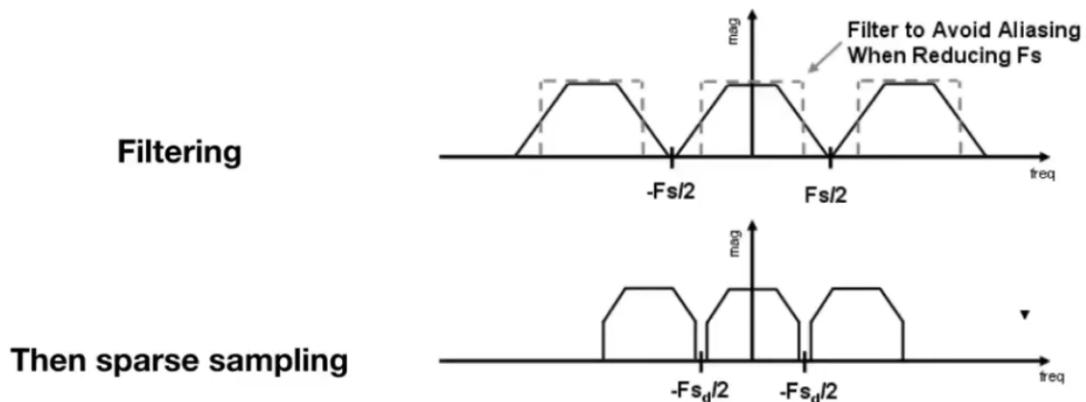


走样在频域下的解释：



采样间隔太长会导致 f_s 太小，频域的波形图发生混叠。

由此，当我们将高频信号去掉时，就不会发生混叠，那么就能够达到反走样的目的。这也就是为什么先模糊后采样可以达到反走样的效果。



实际的操作中，用一个像素大小作为滤波盒，作卷积操作。

对于一个像素，三角形可能覆盖部分，根据覆盖的面积进行求平均。

一种近似方法：MSAA (Multiple Sample Antialiasing)。

- 将一个像素划分成多个小的像素，判断小像素是否在三角形内，取平均值

其它的抗锯齿方案：

- FXAA (Fast Approximate AA)
 - 先得到一个有锯齿的图，再通过图像匹配找到边界并替换成没有锯齿的边界。
- TAA (Temporal AA)
 - 根据上一帧的信息。

超分辨率

DLSS (Deep Learning Super Sampling)

可见性与遮挡 (Visibility/Occlusion)

(油) 画家算法 (Painter's Algorithm)

先画远的事物，再画近的事物。

但这种算法无法解决循环遮挡等问题。

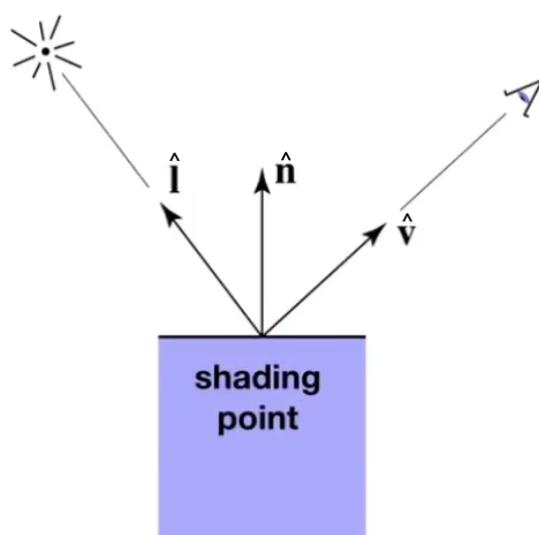
深度缓存 (Z-Buffer)

- 为了简单起见，在这里假定z都为正数，则z越小说明越近，越大说明与越远。
- 维护一个光栅化图形信息和一个深度缓存信息。
- 对于每一个要绘制的三角形，比较三角形上的每一个像素的深度与当前已绘制的图上该像素点的深度，如果较小，则覆盖绘制并更新深度记录。

着色 (Shading)

一种简单的着色模型：冯氏反射 (Blinn-Phong Reflectance Model)

- Viewer direction \hat{v}
- Surface normal, \hat{n}
- Light direction, \hat{l}
- Surface parameters (color, shininess (非亮度))
-



- 漫反射 (Diffuse Reflection)
 - 光线被均匀地散射到各个方向。
 - 散射光的强度 $L_d = k_d(I/r^2) \max(0, \hat{n} \cdot \hat{l})$, k_d 为散射系数, 用 vector3 时可以表示颜色
- 高光 (Specular Highlights)
 - 当观察方向 \hat{v} 与镜面反射光线方向 \hat{R} 比较接近时会出现高光, 此时 \hat{v} 和 \hat{l} 的半程向量 $\hat{h} = \text{besector}(\hat{v}, \hat{l}) = \frac{\hat{v} + \hat{l}}{\|\hat{v} + \hat{l}\|}$ 与法线 \hat{n} 非常接近。
 - $L_s = k_s(I/r^2) \max(0, \cos\alpha)^p$
 $= k_s(I/r^2) \max(0, \hat{n} \cdot \hat{h})^p$
 - 此处忽略了 \hat{l} 和 \hat{n} 的夹角的影响, 指数 p 是为了使角度分布更小更集中。
- 环境光照 (Ambient Lighting)
 - 大胆假设物体各个部分收到的环境光照相同。
 - $L_a = k_a I_a$
 - 给物体增加一个常数光照, 使它不会完全处于黑色。
- $L = L_d + L_s + L_a$

着色频率 (Shading Frequency)

- 平面着色 (Flat shading)
- 顶点着色 (Gouraud shading)
 - 顶点的法向量: 对与顶点相关的面的法向量求平均 (或按面积加权平均)
 - 用插值法构建顶点之间平滑过渡的法线
- 像素着色 (Phong shading)

插值 (Interpolation)

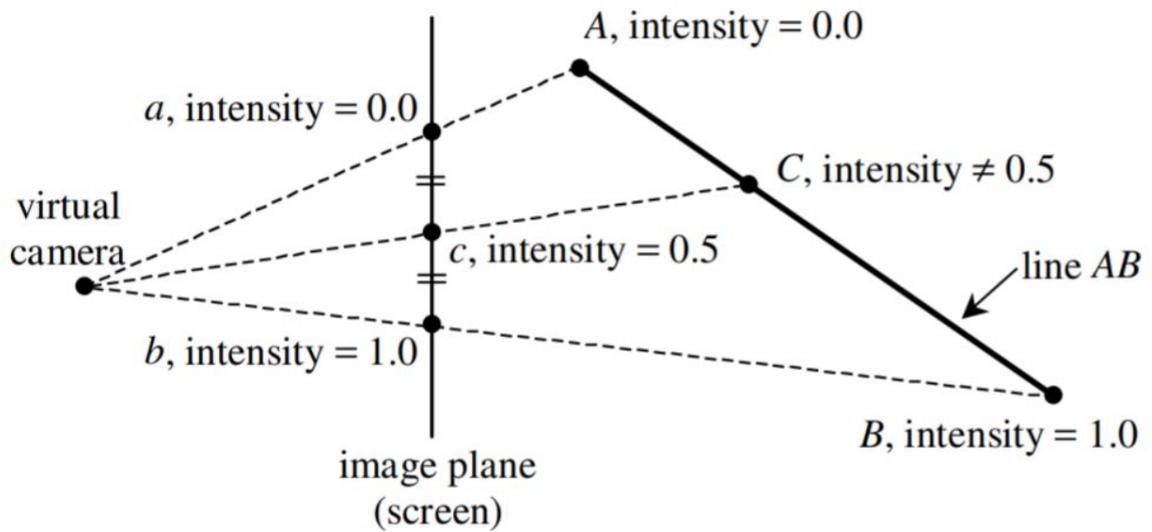
重心坐标 (Barycentric Coordinates)

- 对于与 A、B、C 共面的任一点 (x, y) , 若满足 $(x, y) = \alpha A + \beta B + \gamma C, (\alpha + \beta + \gamma = 1)$, 则称 (α, β, γ) 为点 (x, y) 的重心坐标。其中 $\alpha + \beta + \gamma = 1$ 是为了确保在同一平面。
- α, β, γ 满足 $\alpha = \frac{S_A}{S_A+S_B+S_C}, \beta = \frac{S_B}{S_A+S_B+S_C}, \gamma = \frac{S_C}{S_A+S_B+S_C}$
- α, β, γ 可以通过下式算得

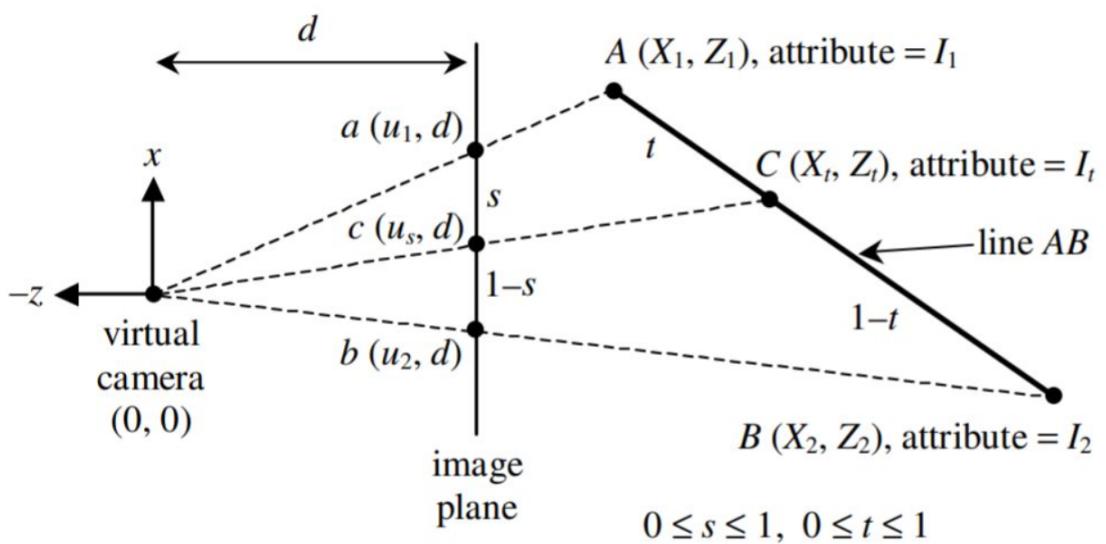
$$\begin{aligned}\alpha &= \frac{-(x - x_B)(y_C - y_B) + (y - y_B)(x_C - x_B)}{-(x_A - x_B)(y_C - y_B) + (y_A - y_B)(x_C - x_B)} \\ \beta &= \frac{-(x - x_C)(y_A - y_C) + (y - y_C)(x_A - x_C)}{-(x_B - x_C)(y_A - y_C) + (y_B - y_C)(x_A - x_C)} \\ \gamma &= 1 - \alpha - \beta\end{aligned}$$

插值的应用

- 与 A、B、C 共面的任一点的属性 $V = \alpha V_A + \beta V_B + \gamma V_C$
- 对于三维坐标下的物体, 应先进行插值再进行投影变换, 因为插值在投影变换下不具有不变性。

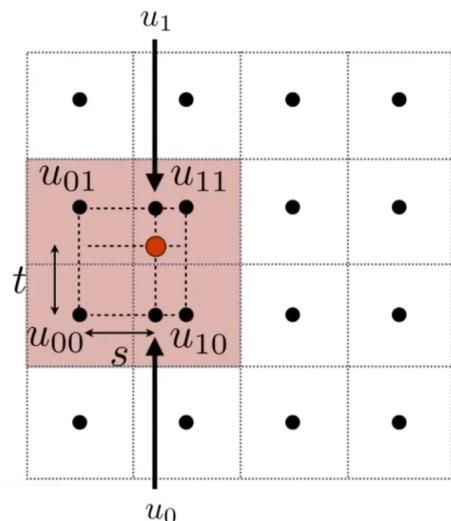


- 透视矫正插值



- $Z_t = \frac{1}{\frac{s}{Z_2} + \frac{1-s}{Z_1}}$, 对于重心坐标有 $Z_t = \frac{1}{\frac{\alpha}{Z_A} + \frac{\beta}{Z_B} + \frac{\gamma}{Z_C}}$
- 对于任意属性 $I_t = (\frac{I_1}{Z_1} + s(\frac{I_2}{Z_2} - \frac{I_1}{Z_1})) / \frac{1}{Z_t}$, 对于重心坐标有 $I_t = (\alpha \frac{I_A}{Z_A} + \beta \frac{I_B}{Z_B} + \gamma \frac{I_C}{Z_C}) / \frac{1}{Z_t}$
- 详细推导过程见[透視矫正插值](#)

双线性插值 (Bilinear Interpolation)



- Linear interpolation (1D) : $lerp(x, v_0, v_1) = v_0 + x(v_1 - v_0)$
- 实际运用时 $lerp(x, v_0, v_1) = (1 - x)v_0 + xv_1$ 效果更好

$$u_0 = lerp(s, u_{00}, u_{10})$$

$$u_1 = lerp(s, u_{01}, u_{11})$$

$$f(x, y) = lerp(t, u_0, u_1)$$

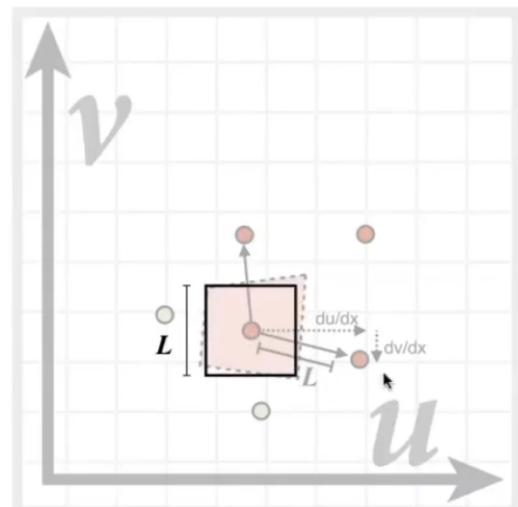
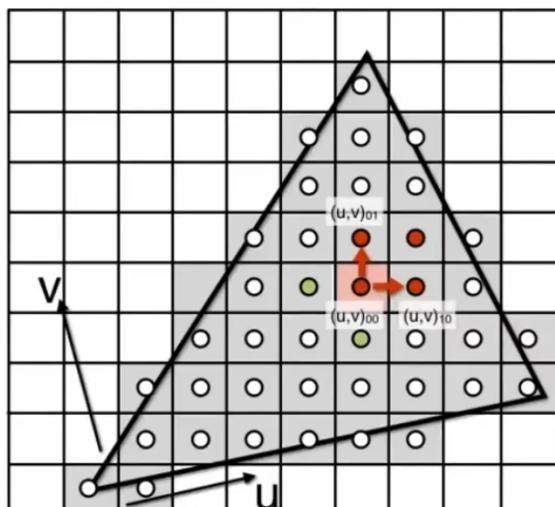
纹理 (Texture)

- 纹理元素 (纹素, texel) : A pixel on texture
- 找到物体上一像素点(x, y, z)与纹素(u, v)的映射关系
 - 当纹理过小时
 - 为了应用到整个物体，纹理会被拉大，出现锯齿。可以用双线性插值去锯齿，双三次 (Bicubic) 线性插值的视觉效果更好，但开销更大，具体操作为取像素点附近16进行插值
 - 当纹理过大时
 - 因物体的采样频率过低，会出现走样的情况。屏幕上越小的物体的一个像素点对应的纹理的范围越大。因此需要一种能够快速得到范围内的纹理的平均的方法 (Range Query) —— Mipmap。

Mipmap

- 快速、近似、正方形的范围查询。
- 由一张图生成一系列图。
- 原始图为Level0，分辨率每缩小一倍就是下一个level，直到图片变为1*1，存储开销是原本的 $\frac{4}{3}$
- 在计算机视觉中Mipmap被称为image pyramid

为计算应采用的level D，需要得到屏幕上一像素点(x, y)对应的纹理上的范围。范围可以近似为以相邻像素点对应的纹素的距离的较长边作为边长形成的正方形。



$$\delta u = \frac{du}{dx} \delta x, \delta x = 1 \therefore \delta u = \frac{du}{dx}, \text{同理 } \delta v = \frac{dv}{dy}$$

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

要得到 $L \times L$ 范围纹理的平均，level 0 每个纹素都是它自身的平均，level 1 每个纹素都是原图 2×2 纹素的平均，因此 $L \times L$ 范围的平均应在level $D = \log_2 L$ 中寻找。

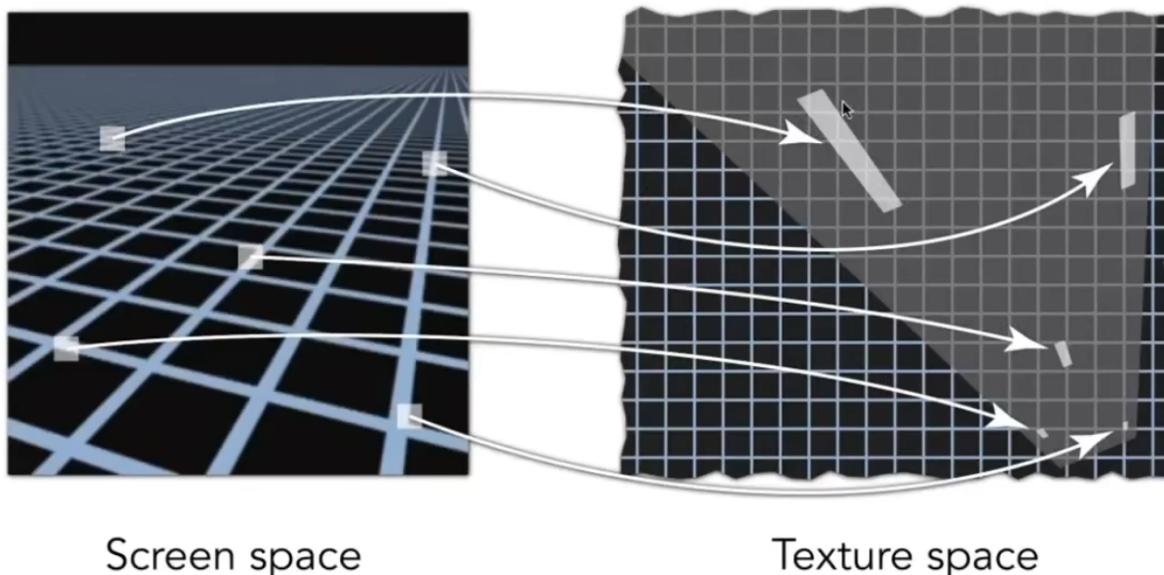
但这种方法得到的纹理存在着不连续的问题，为此，我们引入Trilinear interpolation。

对于 (x, y) , 在Mipmap Level D 中进行插值, 在Mipmap Level D+1中进行插值, 这两个插值的结果再次进行插值, 就可以得到非整数的连续的level。

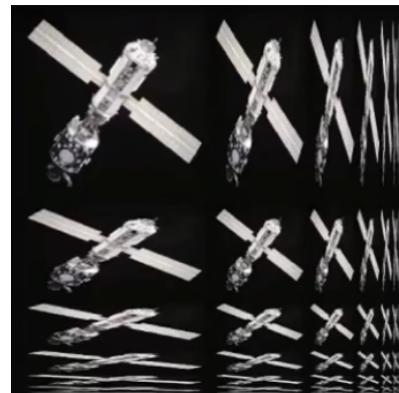
各向异性过滤 (Anisotropic Filtering)

Ripmap

Mipmap由于采用正方形各向同性的特点, 难以较好地处理对应纹理上细长 (axis-aligned rectangular) 或倾斜 (diagonal footprint) 的情况, 可能会出现Overblur的现象。



引入矩形可以较好地解决对应的细长纹理, 但不能解决倾斜的纹理。该方法称为Ripmap。存储开销为原本的四倍。



EWA filtering

用椭圆加权平均 (Elliptically Weighted Average) 进行过滤, 能够较好地解决对应纹理呈对角线形的情况, 但开销较大。

纹理的其它应用

存储环境光

- Spherical Environment Map
 - 可以将环境光信息储存在球体上, 通过球体展开还原环境光, 但是球的两极展开会存在较大失真, 显得偏大
- Cube Map
 - 将环境光信息储存在立方体上, 失真较小, 但需要知道方向。

伪造物体表明凹凸情况

- 凹凸贴图 (Bump Mapping)
 - 通过纹理定义高度差，实现扰动(perturb)表面法线的效果，从而产生凹凸感。
 - 普通平面的法线在切线空间中为 $(0, 0, 1)$ ，而经过扰动的纹理的法线在切线空间中为 $(-\frac{dp}{du}, -\frac{dp}{dv}, 1)$. *normalized()*
 - $\frac{dp}{du} = c_1 [height(u+1) - height(u)], \frac{dp}{dv} = c_2 [height(v+1) - height(v)]$
 - 切线空间中扰动后的法线左乘TBN矩阵转化为世界空间中的法线
 - 通过该方法实现的凹凸在图形边缘以及凹凸形成的阴影有瑕疵
- 法线贴图 (Normal Mapping) :与凹凸贴图比较相近
- 位移贴图 (Displacement Mapping)
 - 同样通过纹理定义高度差，但移动了顶点。

补充

- 纹理也可以是三维的，通过噪声函数的一系列处理计算出纹理。

几何 (Geometry)

表达几何的方式

隐式几何 (Implicit Geometry)

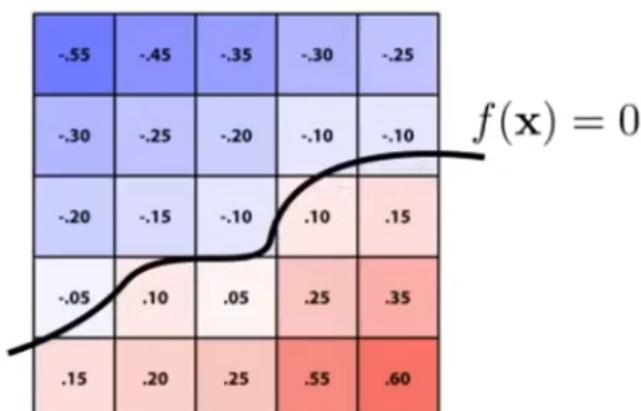
- 给定点需要满足的关系的几何体，如 $x^2 + y^2 + z^2 = 1$ 。
- 标准形式为 $f(x,y,z) = 0$
- 优点：易于判断点是否在几何体内部，若 $f(x_1, y_1, z_1) < 0$ ，则 (x_1, y_1, z_1) 在几何体内部，若等于0则在几何体表面，若大于0则在几何体外部。
- 缺点：不够直观

Constructive Solid Geometry

通过对简单集合体的布尔运算（交并差）形成复杂集合体。

Signed Distance Functions (距离场)

- 距离函数：描述任何一个点到这个几何体表面的最小距离（外正内负）。
- 可以通过找函数值为0的点由函数还原几何体。
- 已知两个几何体的距离函数，对它们进行blend操作，可以得到融合的中间过程的几何图形对应的距离函数，从而还原融合过程的几何图形。
- Level Set Methods (水平集) 表述的距离函数
 - 通过双线性插值还原 $f(x) = 0$ 的函数，从而得到几何体的表面



显式几何 (Explicit Geometry)

- 所有点都直接给出或通过映射关系给出。
- $f : R^2 \rightarrow R^3; (u, v) \rightarrow (x, y, z)$
- 优点：直观
- 缺点：不宜判断点和几何体的位置关系

Point Cloud (点云)

- 直接给出一系列点

Polygon Mesh

- often triangles or quads

The Wavefront Object file (.obj) Format

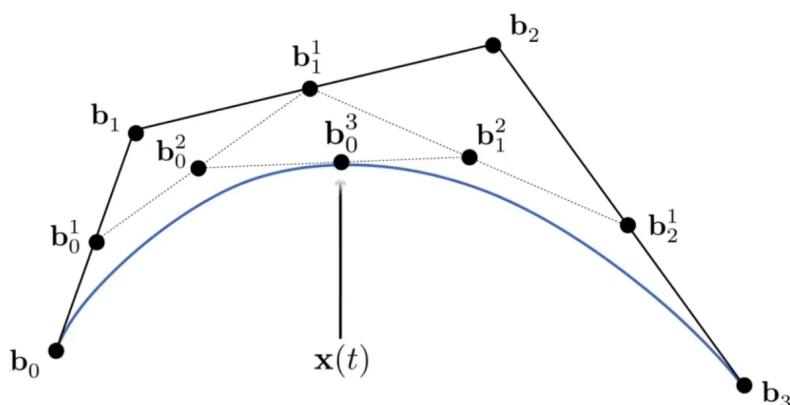
```
1 # This is a comment
2
3 v 1.000000 -1.000000 -1.000000
4 v 1.000000 -1.000000 1.000000
5 v -1.000000 -1.000000 1.000000
6 v -1.000000 -1.000000 -1.000000
7 v 1.000000 1.000000 -1.000000
8 v 0.999999 1.000000 1.000001
9 v -1.000000 1.000000 1.000000
10 v -1.000000 1.000000 -1.000000
11
12 vt 0.748573 0.750412
13 vt 0.749279 0.501284
14 vt 0.999110 0.501077
15 vt 0.999455 0.750380
16 vt 0.250471 0.500702
17 vt 0.249682 0.749677
18 vt 0.001085 0.750380
19 vt 0.001517 0.499994
20 vt 0.499422 0.500239
21 vt 0.500149 0.750166
22 vt 0.748355 0.998230
23 vt 0.500193 0.998728
24 vt 0.498993 0.250415
25 vt 0.748953 0.250920
26
27 vn 0.000000 0.000000 -1.000000
28 vn -1.000000 -0.000000 -0.000000
29 vn -0.000000 -0.000000 1.000000
30 vn -0.000001 0.000000 1.000000
31 vn 1.000000 -0.000000 0.000000
32 vn 1.000000 0.000000 0.000001
33 vn 0.000000 1.000000 -0.000000
34 vn -0.000000 -1.000000 0.000000
35
36 f 5/1/1 1/2/1 4/3/1
37 f 5/1/1 4/3/1 8/4/1
38 f 3/5/2 7/6/2 8/7/2
39 f 3/5/2 8/7/2 4/8/2
40 f 2/9/3 6/10/3 3/5/3
41 f 6/10/4 7/6/4 3/5/4
42 f 1/2/5 5/1/5 2/9/5
43 f 5/1/6 6/10/6 2/9/6
44 f 5/1/7 8/11/7 6/10/7
45 f 8/11/7 7/12/7 6/10/7
46 f 1/2/8 2/9/8 3/13/8
47 f 1/2/8 3/13/8 4/14/8
```

按照顶点(v)、纹理坐标(vt)、法线(vn)、平面(f)的顺序进行描述，会存在冗余。其中f的格式为第几个v/第几个纹理坐标(vt)/第几个法线(vn)。

曲线 (Curve)

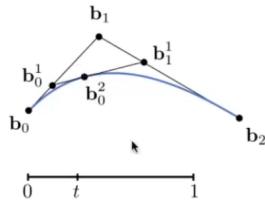
贝塞尔曲线 (Bezier Curve)

- 几何定义：给定 $0 \sim n$ 个点，对相邻两个点取插值 $t(0 \leq t \leq 1)$ ，连接形成的相邻的点，对新生成的点重复操作，直到只生成一个点，这个点即为贝塞尔曲线上的一点。遍历 t 形成的曲线即贝塞尔曲线。



- 代数定义:

- 对于二次贝塞尔曲线 (两轮插值) :



$$\mathbf{b}_0^1(t) = (1-t)\mathbf{b}_0 + t\mathbf{b}_1$$

$$\mathbf{b}_1^1(t) = (1-t)\mathbf{b}_1 + t\mathbf{b}_2$$

$$\mathbf{b}_0^2(t) = (1-t)\mathbf{b}_0^1 + t\mathbf{b}_1^1$$

$$\mathbf{b}_0^2(t) = (1-t)^2\mathbf{b}_0 + 2t(1-t)\mathbf{b}_1 + t^2\mathbf{b}_2$$

- 扩展到n轮插值 (Bezier curve order n) :

$$b(t) = b^n(t) = b_0^n(t) = \sum_{j=0}^n b_j B_j^n(t), \text{ 其中 } B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}, \text{ 为伯恩施坦多项式}$$

- 贝塞尔曲线性质

- 对于三次贝塞尔曲线 (Cubic Bezier)

- $b(0) = b_0; b(1) = b_3$
- $b'(0) = 3(b_1 - b_0); b'(1) = 3(b_3 - b_2)$

- 仿射变换不变性: 仿射变换前后曲线和控制点 (derivative) 都对应, 可以通过变换控制点得到变换后的曲线。

- 投影变换下没有不变性

- 曲线在控制点形成的凸包内 (Convex hull, 控制点形成的最大的凸多边形)

- 逐段的贝塞尔曲线 (Piecewise Bezier Curve)

- 当n较大时曲线较难控制, 且动一个点会影响整条曲线。

- 将曲线分段, 通常每段曲线都由四个控制点控制。

- 曲线连续

- C^0 连续 (continuity) 即两端曲线相接, C^1 连续在相接处的一阶导数相等
- C^1 连续条件: a段曲线与b段曲线满足: $a_n = b_0 = \frac{1}{2}(a_{n-1} + b_1)$

- 样条 (spline)

- 由一系列控制点控制的连续曲线

- 贝塞尔曲线属于spline中的一种曲线

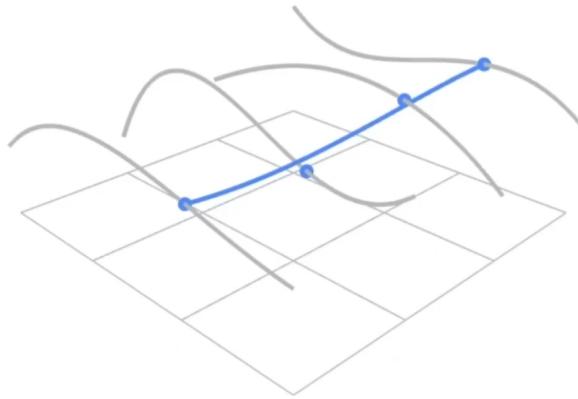
- B-spline (basis spline)

- 基函数由伯恩施坦多项式扩展为其他多项式
- 相比贝塞尔曲线需要更多的信息
- 满足贝塞尔曲线的所有重要性质
- 更多详见<https://www.bilibili.com/video/av66548502?from=search&seid=65256805876131485>

曲面 (Surface)

贝塞尔曲面

- 取多条贝塞尔曲线上的点u作为控制点构建贝塞尔曲线, 贝塞尔曲线上取点u, 遍历(u,v)即可得到曲面, $u, v \in [0, 1]$ 。



曲面处理

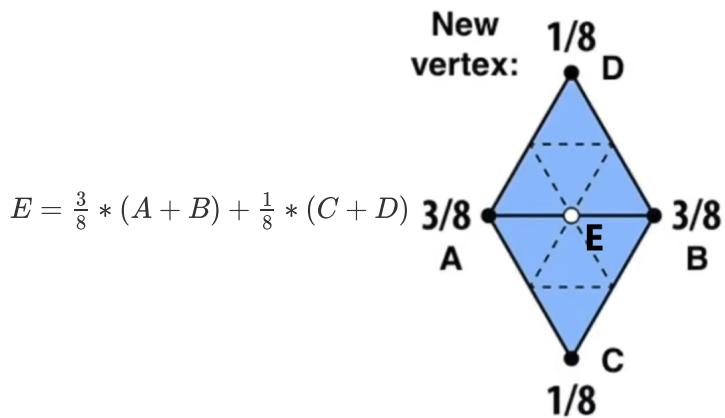
- Mesh subdivision、Mesh simplification、Mesh regularization



Mesh Subdivision

- Loop Subdivision (只试用Triangle Mesh)

1. 细分规则：将一个三角形各边中点相连拆分成四个，这些中点作为新的顶点。
2. 新顶点更新规则：对新的顶点E，E受原本的顶点的影响，应是它们的加权平均。



3. 原顶点更新规则：对于原有的顶点，它的位置与该顶点本身以及顶点周围的原来的顶点有关，所以新的顶点 $E = (1 - n * u) * original_position + u * neighbor_position_sum$ ，其中n是该顶点的度， $u = (n == 3 ? \frac{3}{16} : \frac{3}{8n})$

- Catmull-Clark Subdivision (General Mesh)

- 奇异点 (Extraordinary vertex) : degree != 4

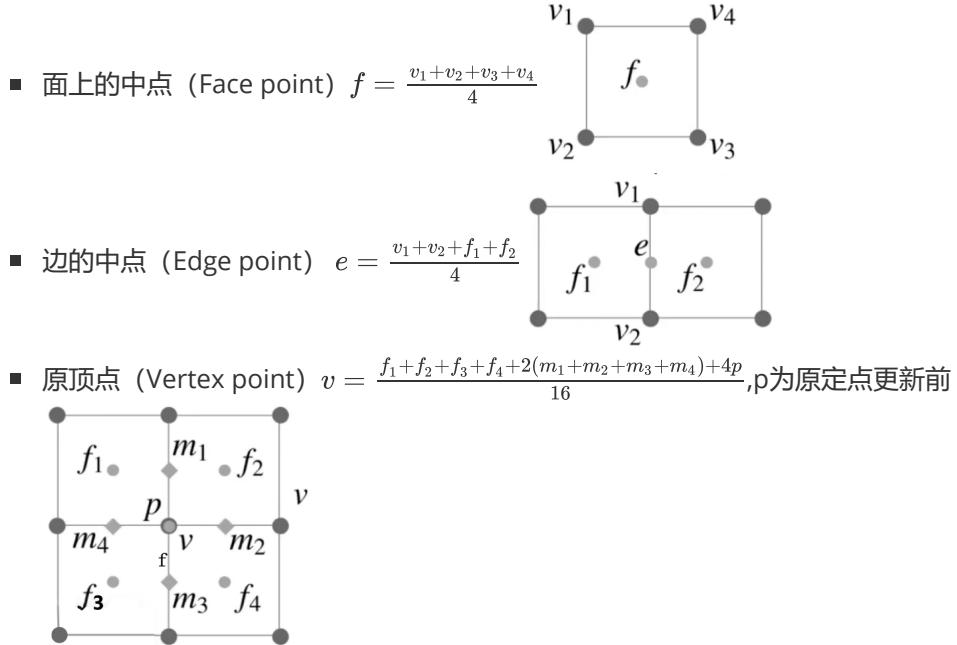
- 细分的步骤

1. 给每个面添加顶点
2. 给每条边添加中点
3. 连接面的新增的顶点与边的中点

- 第一次细分的结果：非四边形面消失，奇异点增加，非四边形面等量转化为奇异点个数

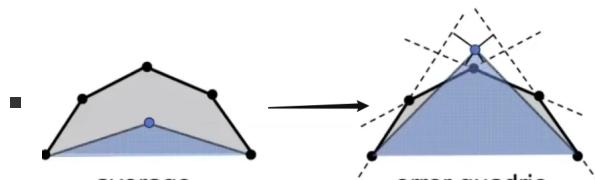
- 之后的细分不会再增加奇异点个数

- 顶点更新规则Catmull-Clark Vertex Update Rules(Quad Mesh)



Mesh Simplification

- 边坍缩 (Edge collapsing)
 - 二次误差度量 (Quadratic Error Metrics)
 - 找一个新的顶点，使其到原本各个面距离的平方和最小



- 将坍缩各边算出的二次度量误差进行排序，坍缩最小的并更新受影响而变化的二次度量误差值（用优先队列实现）

阴影

点光源的阴影 (Shadow Mapping)

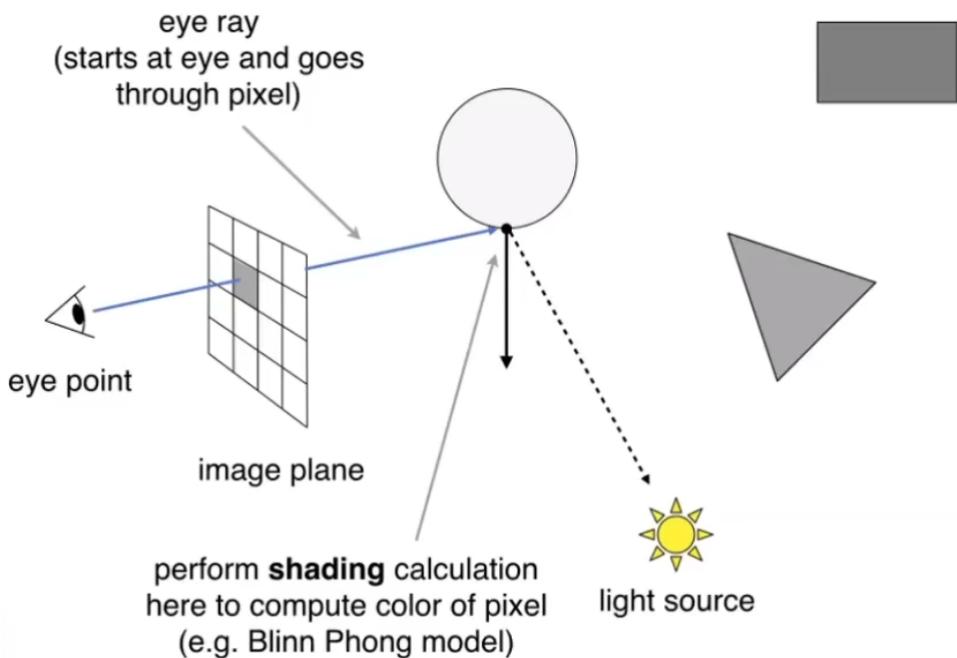
1. 将虚拟相机放在点光源处，记录看到的各个像素点的深度 z ，这一系列深度构成shadow map。
 2. 从真实的相机位置出发，观察各个物体，检测观测点到光源的深度 z' ，与观测点投影变换到光源视角处的像素点的深度 z ，若 $z' > z$ ，则光线无法照射到，应形成阴影
- 这种方法形成的阴影为硬阴影，分界线十分鲜明。
 - 点光源形成的为硬阴影，有体积的光源才能形成软阴影（如太阳光的本影、半影）
 - 造成shadow mapping效果较脏的原因有：浮点数比较相等不精确，shadow map的分辨率太低。

光线追踪 (Ray Tracing)

- 使用光线追踪的原因：光栅化难以很好地处理全局效果（软阴影、光线多次反弹）
- 光线沿直线传播、光线间不发生碰撞、光路具有可逆性 (path reversal-reciprocity)

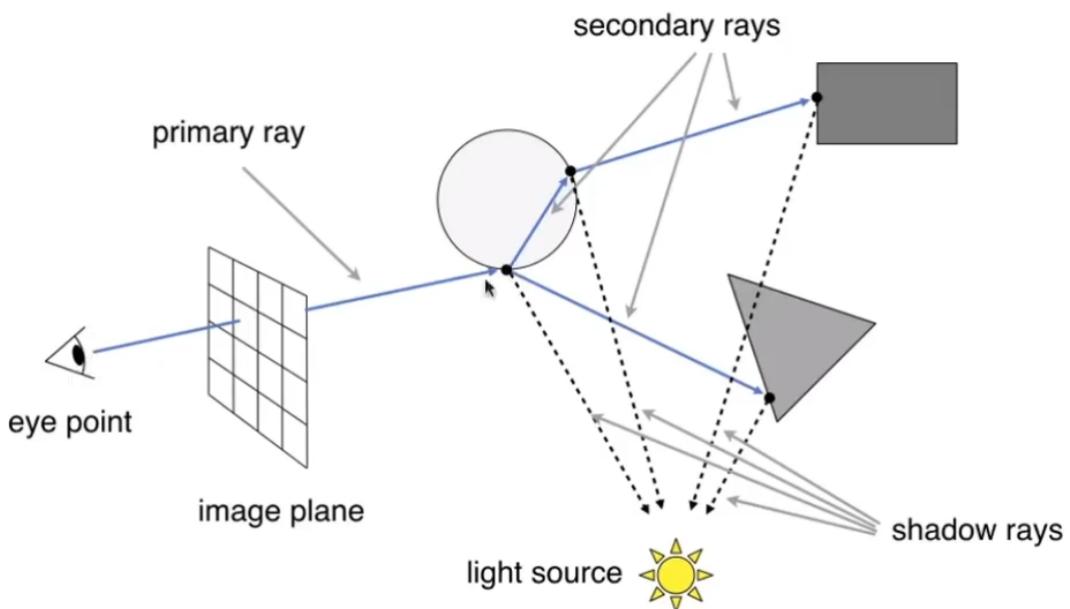
Pinhole camera Model

1. 从相机出发向投影面的每个像素投射一条eye ray。
2. 每条eye ray碰到的最近的点与光源进行连线，如果之间没有阻挡，则可以计算出光源在此点反射回相机的光线强度。



Recursive (Whitted-Style) Ray Tracing

- 对eye ray经过反射、折射后的光线进行同样操作，像素点最终的着色结果为各光线之和，反射、折射的光线有能量衰减。



光线与物体表面的交点

- 光线传播方向上的点 $\vec{r}(t) = \vec{o} + t\vec{d}$, \vec{o} 为光线起点 (origin), \vec{d} 为光线传播方向 (normalized), $0 \leq t < d$

光线与隐式表面的交点

- 对于隐式曲面上的点 \vec{p} , 有 $f(\vec{p}) = 0$, 所以有 $f(\vec{o} + t\vec{d}) = 0$

光线与显式表面的交点

Triangle Mesh

一种较为简单但是计算量很大的思路是光线与每一个三角形求交点。

- 经过 p' 的平面方程为 $(\vec{p} - \vec{p}') \cdot \vec{N} = 0$, 先计算与平面的交点再判断是否在三角形内。
- 直接利用重心坐标。 $\vec{o} + t\vec{d} = (1 - b_1 - b_2)\vec{P}_0 + b_1\vec{P}_1 + b_2\vec{P}_2$, ($b_1, b_2, b_1 + b_2 \in (0, 1)$ 且 $t \geq 0$) 三维的点刚好可以解出三个未知量。
(Moller Trumbore Algorithm)

Where:

$$\begin{bmatrix} t \\ b_1 \\ b_2 \end{bmatrix} = \frac{1}{\vec{S}_1 \cdot \vec{E}_1} \begin{bmatrix} \vec{S}_2 \cdot \vec{E}_2 \\ \vec{S}_1 \cdot \vec{S} \\ \vec{S}_2 \cdot \vec{D} \end{bmatrix}$$

Cost = (1 div, 27 mul, 17 add)

$$\begin{aligned} \vec{E}_1 &= \vec{P}_1 - \vec{P}_0 \\ \vec{E}_2 &= \vec{P}_2 - \vec{P}_0 \\ \vec{S} &= \vec{O} - \vec{P}_0 \\ \vec{S}_1 &= \vec{D} \times \vec{E}_2 \\ \vec{S}_2 &= \vec{S} \times \vec{E}_1 \end{aligned}$$

为加速计算，引入包围盒 (Bounding Volume)

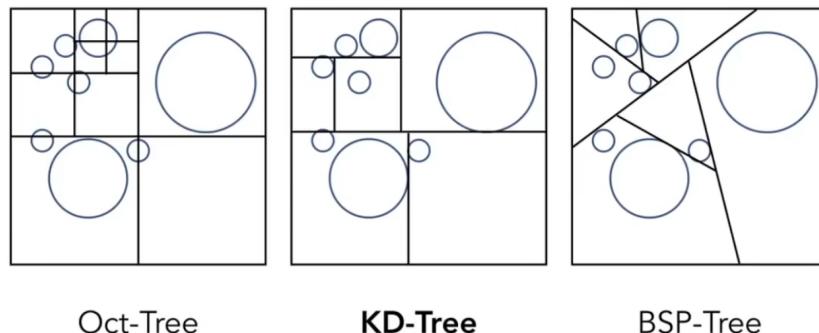
- 当光线与包围盒没有交点时，不可能与三角形有交点。
- 包围盒通常是轴对齐的 (Axis-Aligned Bounding Box, AABB)，轴对齐的包围盒可以有效降低解光线与包围盒的交点时的维度。
- 任意一条光线所在直线与包围盒的两个对面都有交点 t_{min}, t_{max} 。
- 光线进入包围盒的时刻应是光线进入所有对平面之间，光线离开包围盒应是光线离开任一对平面之间。
- $t_{enter} = \max\{t_{min}\}, t_{exit} = \min\{t_{max}\}$
- 当且仅当 $t_{enter} < t_{exit} \& t_{exit} \geq 0$ 时光线与AABB盒有交点。

Uniform Spatial Partitions (Grids)

1. 找到包围盒，将其细分为小格子 (grid)，在每一个与物体相交的格子中存储物体的信息。
 2. 发射光线，对于与光线相交的格子，若各自内有物体，就对光线和该物体检测相交。
- 为保证效率，拆成的格子个数=27*objs

当场景内的物体分布不均匀时，该方法的效率并不高。

Spatial Partitions



Oct-Tree

- 对于每一块区域沿着各个轴方向各切一刀，在三维情况下就是八等分，分割出的区域递归分割，当区域变小或含的物体较少则停止递归。
- 缺点：当维度变高时，分割出的区域数增大很快

BSP-Tree

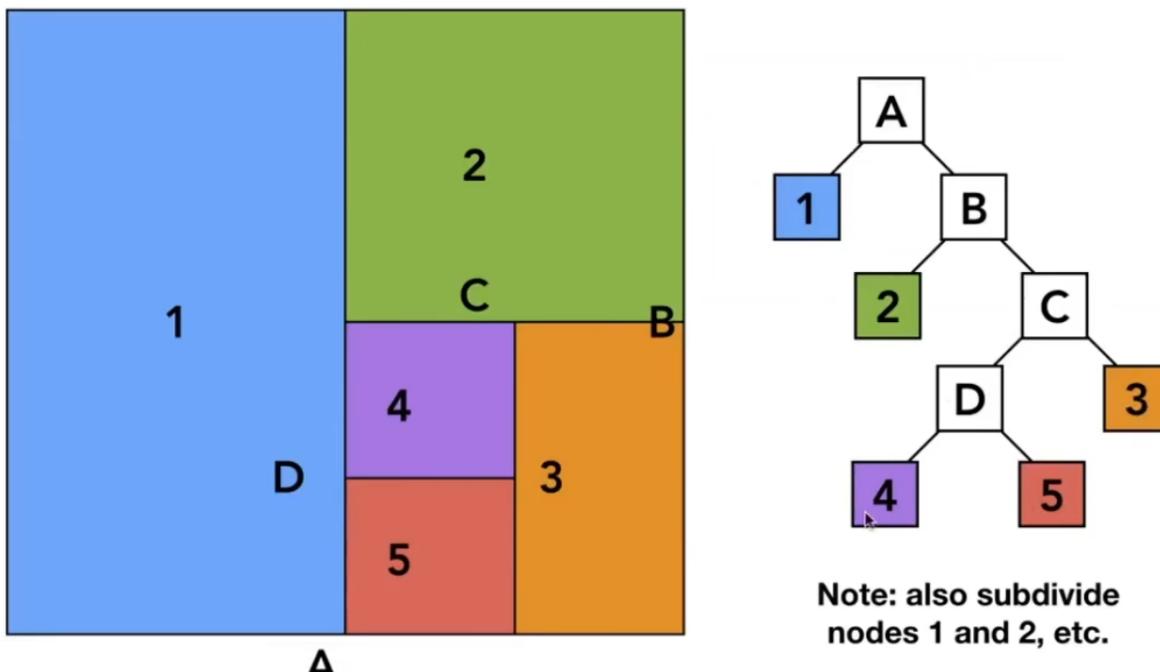
- 对每一块区域一次切一刀，尽可能使一个物体不出现在多个区域内
- 优点：分割出的区域数与维度无关
- 缺点：计算复杂

KD-Tree

- 对每一块区域沿轴方向切一刀，方向在轴方向之间轮换，能够较好地保证空间切割得较均匀
- 分割方向与轴对齐计算量小
- 缺点：一个物体可能存在于多个包围盒内，且判断包围盒与物体相交是一件相当困难的事情。

KD-Tree的建立

- 将区域作为一个节点，判断是否要再分割，分割成的区域作为它的孩子节点



- 对于每个内部节点，记录分割的轴方向、分割的点以及它的孩子节点
- 对于叶子节点，记录区域内的所有的物体

Traversing a KD-Tree

- 检测光线与节点的空间是否有交点（包围盒交点检测的充要条件），如果有，递归检测与两个孩子是否有交点，直到节点为叶子节点。检测光线与叶子节点内所有物体的交点。

Object Partitions & Bounding Volume Hierarchy (BVH)

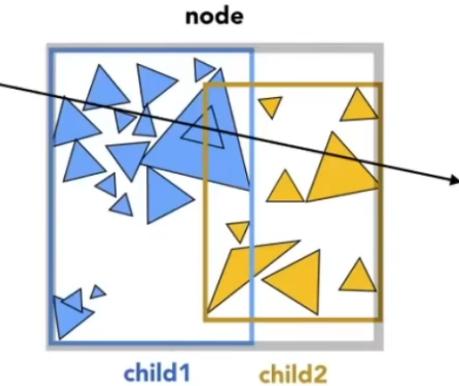
- 解决了KD-Tree存在的显著问题，得到了广泛应用
- 不再对空间进行划分，而是将物体进行划分，对划分完成的物体求包围盒，这样子物体只会出现在一个包围盒内，虽然包围盒之间会存在部分相交
- 划分物体时：
 - 总是选择对最长的轴进行划分
 - 划分时以中间的物体（中位数）为分界线划分开，使空间尽可能均匀（运用快速选择算法，时间复杂度为O(n)）

- BVH Traversal

```
Intersect(Ray ray, BVH node) {
    if (ray misses node.bbox) return;
    if (node is a leaf node)
        test intersection with all objs;
        return closest intersection;

    hit1 = Intersect(ray, node.child1);
    hit2 = Intersect(ray, node.child2);

    return the closer of hit1, hit2;
}
```



辐射度量学 (Basic radiometry)

- Radiant energy $Q[J = Joule]$
- Radiant flux $\phi = \frac{dQ}{dt}[W = Watt][lm = lumen]$ 单位时间辐射的能量 (Power)
- Radiant Intensity: $I(\omega) = \frac{d\phi}{d\omega}[\frac{W}{sr}][\frac{lm}{sr}] = cd = candela]$ 光源单位时间向单位立体角 (solid angle) 辐射的能量
- Irradiance: $E(x) = \frac{d\phi(x)}{dA}[\frac{W}{m^2}][\frac{lm}{m^2}] = lux$ 物体表面单位面积在单位时间内接收到的能量, ϕ 应与面法向量进行点乘
- Radiance: $L(p, \omega) = \frac{d^2\phi(p, \omega)}{d\omega dA \cos\theta}[\frac{W}{sr \cdot m^2}][\frac{cd}{m^2}] = \frac{lm}{sr \cdot m^2} = nit$ 物体表面单位投影面积单位时间内发射、或接收的单位立体角的光的能量

立体角: 角度在三维空间中的表示。立体角的大小可用球体进行衡量。 $\Omega = \frac{A}{r^2}$ sr[steradians], 球有 4π sr

孤立点光源 (Isotropic Point Source) 的 $I = \frac{\phi}{4\pi}$

- Incident Radiance $L(p, \omega) = \frac{dE(p)}{d\omega \cos\theta}$ 到达物体表面的单位立体角的Irradiance
- Exiting Radiance $L(p, \omega) = \frac{dI(p, \omega)}{dA \cos\theta}$ 物体表面发射单位投影面积发射出的Intensity
- Irradiance && Radiance

$$\circ E(p) = \int_{H^2} L_i(p, \omega) \cos\theta d\omega, H^2 \text{ 代表半球(Hemisphere)}$$

双向反射分布函数 Bidirectional Reflectance Distribution Function (BRDF)

光线在一个点上的反射 (散射.....) 过程就是Radiance转化为点的Irradiance, Irradiance再将能量发射出去作为Radiance的过程。

为描述光线由Irradiance转化为Radiance的过程, 定义BRDF

$f_r(\omega_i \rightarrow \omega_r) = \frac{dL_r(\omega_r)}{dE_i(\omega_i)} = \frac{dL_r(\omega_r)}{L_i(\omega_i) \cos\theta_i d\omega_i} [\frac{1}{sr}]$ 用于表征朝各立体角发出的光的能量的比例, 则有

$$L_r(p, \omega_r) = \int_{H^2} f_r(p, \omega_i \rightarrow \omega_r) L_i(p, \omega_i) \cos\theta_i d\omega_i \quad (\text{The Reflection Equation})$$

推广上式到会发光物体同样适用, 有

$$L_r(p, \omega_r) = L_e(p, \omega_o) + \int_{H^2} f_r(p, \omega_i \rightarrow \omega_r) L_i(p, \omega_i) (n \cdot \omega_i) d\omega_i \quad (\text{The Rendering Equation})$$

上式中的 ω_i 方向都由点指向平面外。

经数学方法可化作

$$L = E + KL$$

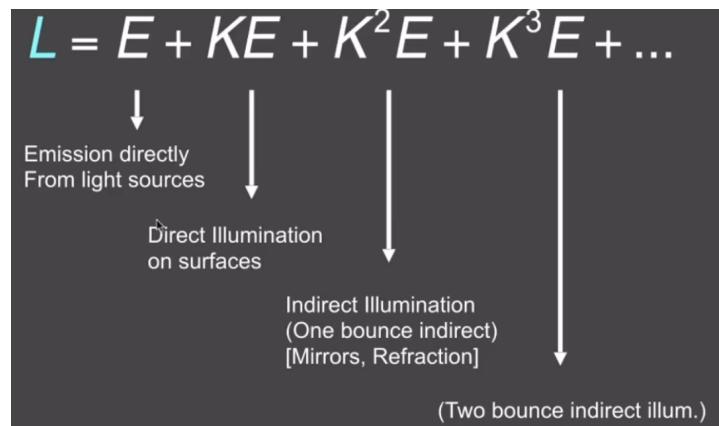
即

$$L = (I - k)^{-1} E$$

展开可得

$$L = (I + K + K^2 + K^3 + \dots)E$$

$$L = E + KE + K^2 E + K^3 E + \dots \quad (\text{全局光照})$$



蒙特卡洛积分 (Monte Carlo Integration)

$$F_N = \frac{1}{N} \sum_{i=1}^N \frac{f(X_i)}{p(X_i)}, \text{ 推导过程如下:}$$

$$\begin{aligned} \text{由 } E[f(x)] &= \int f(x)p(x)dx, \\ \text{得 } E\left[\frac{1}{N} \sum_{i=1}^N \frac{f(X_i)}{p(X_i)}\right] &= \frac{1}{N} E\left[\sum_{i=1}^N \frac{f(X_i)}{p(X_i)}\right] = \frac{1}{N} N \int \frac{f(x)}{p(x)} p(x)dx = \int f(x)dx = F_N \end{aligned}$$

路径追踪 (Path Tracing)

路径追踪的过程，其实就是解渲染方程的过程。

一种简单的处理策略

取 $p(\omega_i) = \frac{1}{2\pi}$ ，对半球内的各个方向均匀采样。

```

shade(p, wo)
    Randomly choose N directions  $w_i \sim \text{pdf}$ 
     $Lo = Le$ 
    For each  $w_i$ 
        Trace a ray  $r(p, w_i)$ 
        If ray r hit the light
             $Lo += (1 / N) * L_i * f_r * \cosine / \text{pdf}(w_i)$ 
        Else If ray r hit an object at q
             $Lo += (1 / N) * \text{shade}(q, -w_i) * f_r * \cosine / \text{pdf}(w_i)$ 
    Return Lo

```

但是由于各个点都要考虑来自各个方向的光线，多次弹射的光线会直接爆炸增长，不具有实用意义。

解决光线数量爆炸增长的策略

由于 1 的指数次依然是 1 ，每个点需要考虑的光线只随机取一个方向。

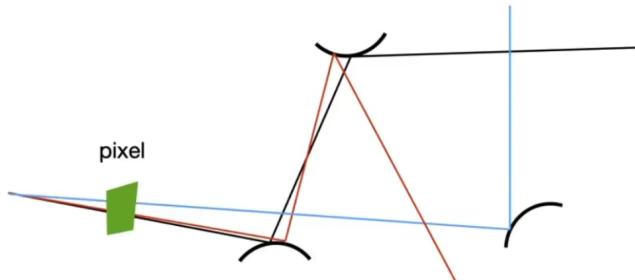
```

shade(p, wo)
    Randomly choose ONE direction  $w_i \sim \text{pdf}(w)$ 
    Trace a ray  $r(p, w_i)$ 
    If ray r hit the light
        Return  $L_i * f_r * \cosine / \text{pdf}(w_i)$ 
    Else If ray r hit an object at q
        Return  $\text{shade}(q, -w_i) * f_r * \cosine / \text{pdf}(w_i)$ 

```

但这样采样存在非常大的噪声，为减小噪声，必须多次采样。

对通过一个像素范围内的多条光线均作路径追踪然后取平均值。



```

ray_generation(camPos, pixel)
    Uniformly choose N sample positions within the pixel
    pixel_radiance = 0.0
    For each sample in the pixel
        Shoot a ray r(camPos, cam_to_sample)
        If ray r hit the scene at p
            pixel_radiance += 1 / N * shade(p, sample_to_cam)
    Return pixel_radiance

```

但shade函数仍有可能无限递归，需要对光线弹射进行一定限制。

限制光线弹射

俄罗斯轮盘赌 (Russian Roulette, RR)：左轮手枪的轮盘内装有部分子弹，转动左轮手枪的轮盘并开枪，若没子弹则没事。

利用俄罗斯轮盘赌的思想，让光线有一定概率不再反弹。

着色结果为 L_o ，指定概率P，让函数以概率P返回值 L_o/p ，以概率(1-P)直接返回0。可以求得返回值的期望 $E = P * (L_o/P) + (1 - P) * 0 = L_o$

shade(p, wo)

```

Manually specify a probability P_RR
Randomly select ksi in a uniform dist. in [0, 1]
If (ksi > P_RR) return 0.0;

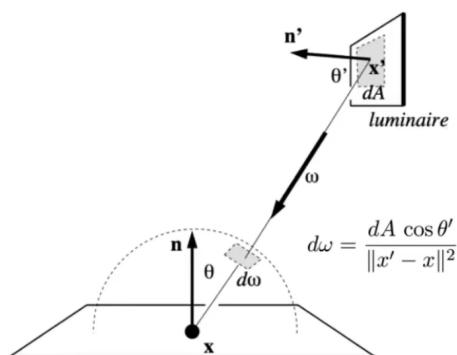
Randomly choose ONE direction wi~pdf(w)
Trace a ray r(p, wi)
If ray r hit the light
    Return L_i * f_r * cosine / pdf(wi) / P_RR
Else If ray r hit an object at q
    Return shade(q, -wi) * f_r * cosine / pdf(wi) / P_RR

```

提高效率，对较小光源精确采样

当光源非常小时，对于半球积分实际上是非常低效的，需要进行非常多次的采样才能取得好的效果。

可以将积分区域由半球转为光源所在面。



$$\begin{aligned}
L_o(x, \omega_o) &= \int_{\Omega_+} L_i(x, \omega_i) f_r(x, \omega_i, \omega_o) \cos\theta d\omega_i \\
&= \int_A L_i(x, \omega_i) f_r(x, \omega_i, \omega_o) \frac{\cos\theta \cos\theta'}{\|x' - x\|^2} dA
\end{aligned}$$

用蒙特卡洛积分法求解时， $p(x) = \frac{1}{A}$ 。

除了对光源的采样外，还需要对其它环境光进行采样，由于已经对最明显的光源进行精确采样，对其它环境光的采样不用再像采取这种方法之前一样高频，从而提升了效率。

```

shade(p, wo)

# Contribution from the light source.

Uniformly sample the light at x' (pdf_light = 1 / A)
L_dir = L_i * f_r * cos θ * cos θ' / |x' - p|^2 / pdf_light
    需判断光源与点之间是否有遮挡

# Contribution from other reflectors.

L_indir = 0.0

Test Russian Roulette with probability P_RR

Uniformly sample the hemisphere toward wi (pdf_hemi = 1 / 2pi)
Trace a ray r(p, wi)

If ray r hit a non-emitting object at q
    L_indir = shade(q, -wi) * f_r * cos θ / pdf_hemi / P_RR

Return L_dir + L_indir

```

更多

- 蒙特卡洛积分的重点采样 (importance sampling)
- 结合半球和光线采样 (multiple imp. sampling, MIS)
- pixel reconstruction filter (相较于像素内各点直接取平均)
- radiance到color的转化 (gamma correction, curves, color space)

其它的现代光线追踪技术

- Photon mapping
- Metropolis light transport
- VCM/UPBP

材质 (Materials)

物体的不同材质表现为它与光的不同作用，实际上就是渲染方程中的BRDF。

- 漫反射 (diffuse)

漫反射各方向反射的radiance都相等，所以有

$$\begin{aligned}
L_o(\omega_o) &= f_r L_i \int_{H^2} \cos\theta_i d\omega_i = \pi f_r L_i \\
\therefore f_r &= \frac{\rho}{\pi}
\end{aligned}$$

ρ 为反照率 (albedo, 可多维、可含颜色信息)，表征光线能量损失。

- Glossy material (金属类)
- Ideal reflective / refractive material

此处的BRDF更准确地应描述为BSDF (散射) (BRDF+BTDF(折射)=BSDF) , 但通常不要紧。

反射与折射

反射

给定入射光的入射方向角 ω_i 和法线 \vec{n} , 可以解得出射光出射方向角 $\omega_o = -\omega_i + 2(\omega_i \cdot \vec{n})\vec{n}$ 。

给定入射光的方位角 ϕ_i , 可得出射光方位角 $\phi_o = (\phi_i + \pi) \bmod 2\pi$ 。

折射

- Snell's Law: $n_i \sin\theta_i = n_t \sin\theta_t$

菲涅尔项 (Fresnel Term)

当视角与物体表面法线呈不同角度时, 反射和折射的光线的比例不同。

对于绝缘体而言, 当光线与物体表面接近垂直时反射的光线很少, 当光线与物体表面接近平行时反射的光线很多。

对导体而言, 反射的光线占大部分且随光线与物体表面角度变化不明显。

菲涅尔项的表达式非常复杂, 通常可以用下式近似。

$$R(\theta) = R_0 + (1 - R_0)(1 - \cos\theta)^5,$$

其中 $R_0 = \left(\frac{n_1 - n_2}{n_1 + n_2}\right)^2$

微表面材质 (Microfacet Material)

对于微观上凹凸不平的材质, 从远处宏观上看看到的是粗糙的平面。

远处看到的是材质, 近处看到的是几何。

Microfacet BRDF

微表面法线完全集中于同一个方向时, 表现为mirror。

微表面的法线较集中于同一个方向时, 表现为glossy。

微表面的法线发散到各个方向时, 表现为diffuse。

- BRDF在微表面下的表达式

$$f(i, o) = \frac{F(i, h)G(i, o, h)D(h)}{4(n_i)(n_o)}, \text{ 其中 } F \text{ 为菲涅尔项, } G \text{ 为shadowing-masking term, } D \text{ 为distribution of normals}$$

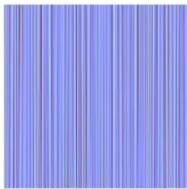
微表面的自遮挡、自投影 (在入射光线的入射角非常大时较多, grazing angle)

D=使入射光线沿出射方向的half vector \vec{h} 的占比, 即微表面上半程向量与法向量相同的比例

各向同性 (isotropic) 与各向异性 (anisotropic)

当 $f_r(\theta_i, \phi_i; \theta_r, \phi_r) \neq f_r(\theta_i, \theta_r, \phi_r - \phi_i)$ 时, 称该种材质为各项异性。

一种各向异性的材质——



各向异性的材质会呈现一些奇怪的现象，如单向打磨的电梯门光线扭曲、圆周擦洗的锅底呈辐射状。

BRDF的性质

- 非负性
- 线性可加性
- 可逆性 $f_r(\omega_r \rightarrow \omega_i) = f_r(\omega_i \rightarrow \omega_r)$
- 能量守恒 $\int_{H^2} f_r(\omega_i \rightarrow \omega_r) \cos\theta_i d\omega_i \leq 1$

测量BRDF

- 物理上推导得到的菲涅尔项与实际仍存在较大出入，因而近似得到的方程也不可靠，因此直接测量BRDF并记录可以得到更真实的效果。
- 对于各向异性的材质，BRDF有四个参数（维度），对于各向同性的材质，BRDF有三个参数并且由可逆性可以减少一半数据量。
- MERL BRDF Database存储了大量材质的测量数据。
- 存储BRDF数据也是一个研究重点。

Advanced Technology

Advanced Light Transport

双向路径追踪 (Bidirectional Path Tracing, BDPT)

思想：从光源和摄像机各生成一系列半路径，将半路径合成为一条路径。

无偏估计

适用：从光源出发的光线传播路径较为复杂时。

缺点：难实现、渲染慢。

Metropolis Light Transport (MLT)

思想：若给定一个路径，可以通过马尔可夫链 (Markov Chain Monte Carlo, MCMC) 在周围产生一系列相似的路径。蒙特卡洛积分的p的函数与待积分的函数越接近，采样效率越高，而通过马尔科夫链可以生成与待积分函数非常一致的函数。

无偏估计

适用：光路复杂、焦散 (caustics) (先经过specular、再经过diffuse, 再经过specular, SDS)

缺点：

- 无法估计渲染的速度
- 是一种局部的方法，得出的结果可能整体上很脏，前后帧图像收敛可能不一致导致动画抖动厉害

光子映射 (Photon Mapping)

有偏估计但一致

适用：特别适合用于解决Specular-Diffuse-Specular (SDS) 路径和产生caustics

有许多种实现方法，下面是一种。

1. photon tracing：从光源出来的光线不断做反射、折射直到遇到diffuse表面。
2. photon collection：从摄像机发射光线不断做反射、折射直到遇到diffuse表面
3. calculation local density estimation：对任一个着色点，找到最近的N个的光子，计算占的面积，得到光子的密度

N取越大图的效果越好。但因为 $\frac{\Delta N}{\Delta A} \neq \frac{dN}{dA}$ ，所以光子映射是有偏估计，最后得出的结果是糊的。 ΔA 越小，这两者就会越接近，当 ΔA 无穷接近0时，结果会是无偏的。只需要增大发射出的光子数，就可以使 ΔA 变小，当 ΔA 足够小时，图片就不会糊。这也是为什么是找最近的N个光子而不是找固定面积内的光子的数量，因为固定面积内光子的数量是有偏的且 $\frac{\Delta N}{\Delta A}$ 不会收敛于 $\frac{dN}{dA}$ 。

Vertex Connection and Merging

思想：结合双向路径追踪和光子映射。对于BDPT形成的不能相连但能够合并（处于非常接近的一个面内）的子路径端点，用光子映射合并这些端点。

实时辐射度 (Instant Radiosity, IR)

思想：从光源形成一系列路径，将路径的端点作为虚拟点光源 (Virtual Point Light, VPL)，再用虚拟点光源渲染场景。（many-light rendering）

优点：快速并且在漫反射场景中有较好效果

缺点：无法处理glossy材质，且当虚拟点光源间太过靠近（如在角落）会出现一些异常的亮点。

Advanced Appearance Modeling

非表面模型 (Non-surface models)

Participating media 散射 (参与) 介质

- 云、雾
- Phase Function描述了光线如何散射
- 从相机触发，随机弹向一个方向，沿直线前进随机一段距离形成一个着色点，着色点与光源连接进行着色。

Hair/fur/fiber

Hair

- Marschner Model

将头发丝视作圆柱 (cuticle和cortex组成)，光线与头发丝有三种互动——在表面直接反射 (R)，从头发一侧进入另一侧穿出 (TT)，从头发一侧进入在内部经反射再穿出 (TRT)，其中进入头发的光线会被cortex部分吸收。

Fur

头发模型并不能直接用于毛发模型。

毛发由cuticle、cortex和medulla (髓质) 组成 (头发其实也由他们组成，但在头发中髓质非常小，可以忽略)。

由此有一种双层圆柱模型 (double cylinder model) , cortex在外, medulla在内, cortex吸收部分光线, medulla负责散射。光线与头发的交互有五种——TT、R、TRT、TTS、TRTS (S为散射) 。

颗粒材质 (Granular Material)

Translucent Material

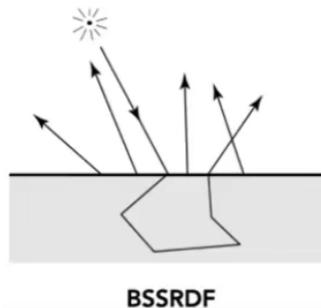
Translucent: 形容光线从一个点进入物体, 在内部经过一系列反射、折射从其他地方出去而形成的半透明的感觉, 实际上并不是半透明。

玉石 (Jade) 、人的皮肤等。

这种光线传播被称为Subsurface Scattering。

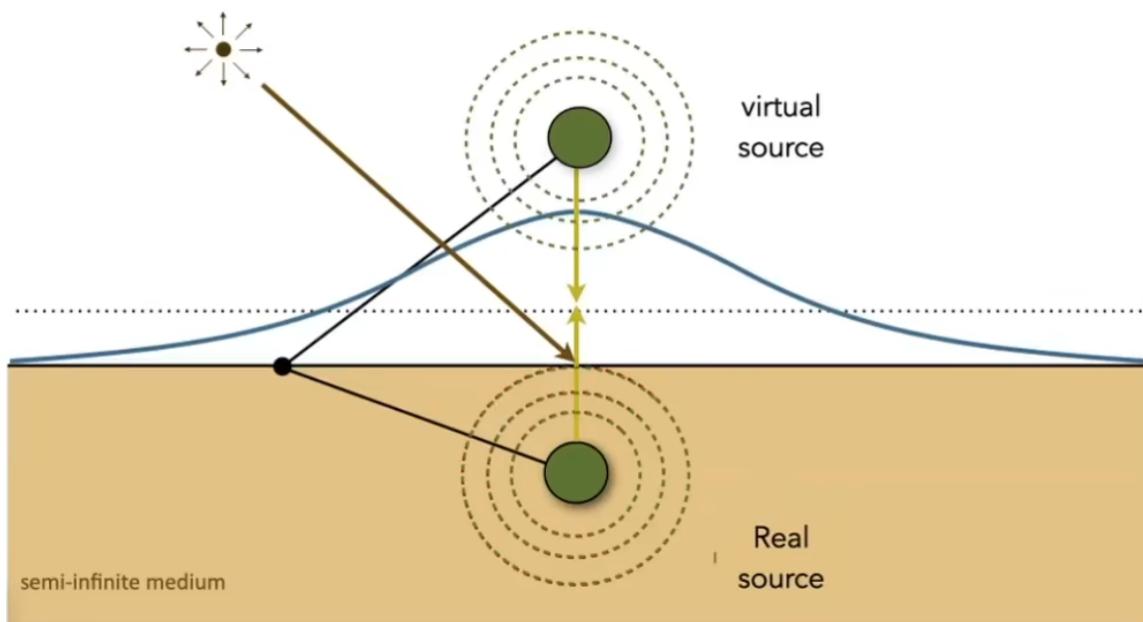
对于这种传播, 需将BRDF推广到BSSRDF。

$$L(x_o, \omega_o) = \int_A \int_{H^2} S(x_i, \omega_i, x_o, \omega_o) L_i(x_i, \omega_i) \cos\theta_i d\omega_i dA$$



BSSRDF

为模拟这种光线传播, 人们提出来Dipole Approximation, 即在物体表面虚拟一个光源, 与原本的真实光源共同作用。

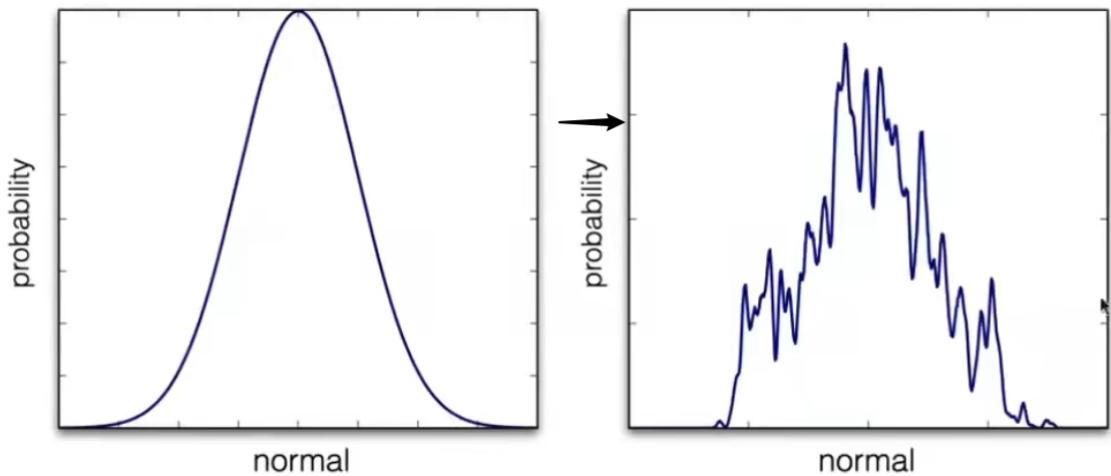


这种模型渲染的大理石、人脸效果相比BRDF很好。

布料的渲染: BRDF、participating media

表面模型 (Surface models)

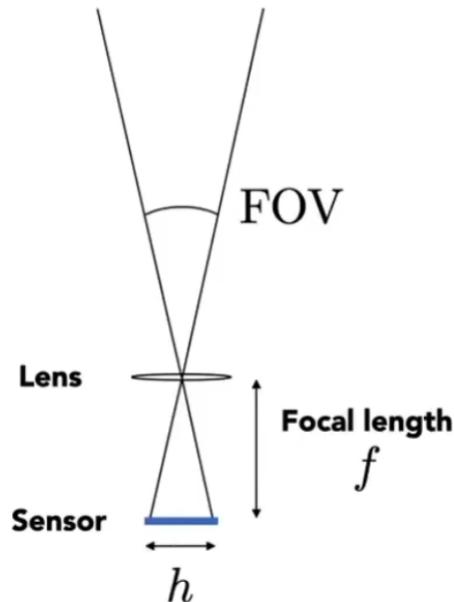
对Distribution of normals进行处理可以得到更真实的结果。



程序化生成 (Procedural appearance)

相机

FOV



$$FOV = 2 \arctan\left(\frac{h}{2f}\right)$$

人们描述FOV时往往假定胶片为35mm格式 (36*24mm)，给出焦距f。事实上胶片未必为35mm格式，所以实际焦距应等比例缩放。

曝光 (Exposure)

Exposure $H = T * E$, T 为曝光时间, E 为irradiance。

曝光时间由快门 (shutter) 决定, irradiance由光圈 (aperture) 和焦距 (focal length) 决定。

光圈 (Aperture)

f数 (f-stop) : 表征光圈的大小, 定义为 $\frac{f}{d}$, d 为光圈直径, f 为焦距, f数越小表面光圈越大。通常写作FN或F/N, N为f-number。

快门 (Shutter)

快门速度会影响曝光的时间，往往以曝光时间表征，单位通常为ms。

机械快门打开有一个过程，这会导致不同点采样实际上不同时，对于高速运动的物体会出现扭曲。

长时间开启会有运动模糊 (motion blur)，运动模糊可以体现快，也可以抗锯齿。

- f数和快门时间在摄影中往往相互制约，f数变为原来两倍时，光圈大小变为原来二分之一，曝光面积变为原来的四分之一，为保证曝光的量不变，曝光时间应变为原来的四倍。

感光度 (ISO gain)

硬件或数学方法直接放大传感器得到的值。

这是一个线性的值，实际上就是直接对传感器得到结果做一个乘法，当它过大时会出现明显的噪点。

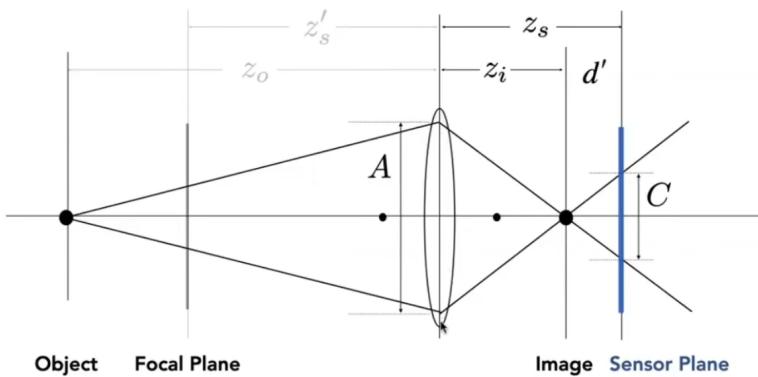
薄透镜近似 (Thin Lens Approximation)

- 薄凸透镜的性质
 - 所有平行于光轴入射透镜的光线折射后必经过焦点
 - 所有经过焦点入射的光线经薄凸透镜折射后必平行光轴出射
 - 经过凸透镜中心的光线折射前后方向不变

由上述性质可以在几何上推出

$$\frac{1}{f} = \frac{1}{z_i} + \frac{1}{z_o} \quad (\text{The Thin Lens Equation})$$

Circle of Confusion (CoC)



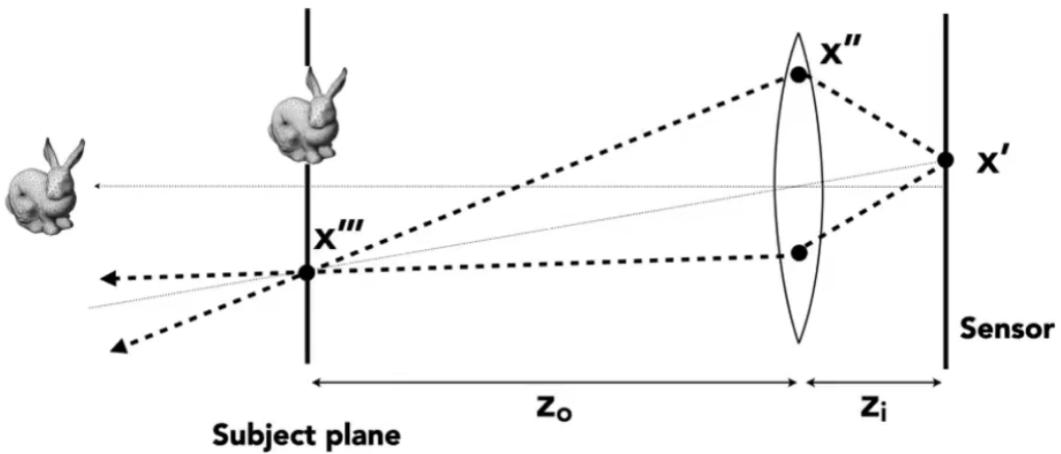
对于非焦平面上的点，它的光线会在传感器的平面上投出一个弥散圆，这就是图片变糊的原因，也就是景深出现的原因。

由几何关系可得 $\frac{C}{A} = \frac{d'}{z_i} = \frac{|z_s - z_i|}{z_i}$ 。

由此可以得到光圈 (A) 越大，弥散圆越大，图片上不在焦平面上的图像越糊。

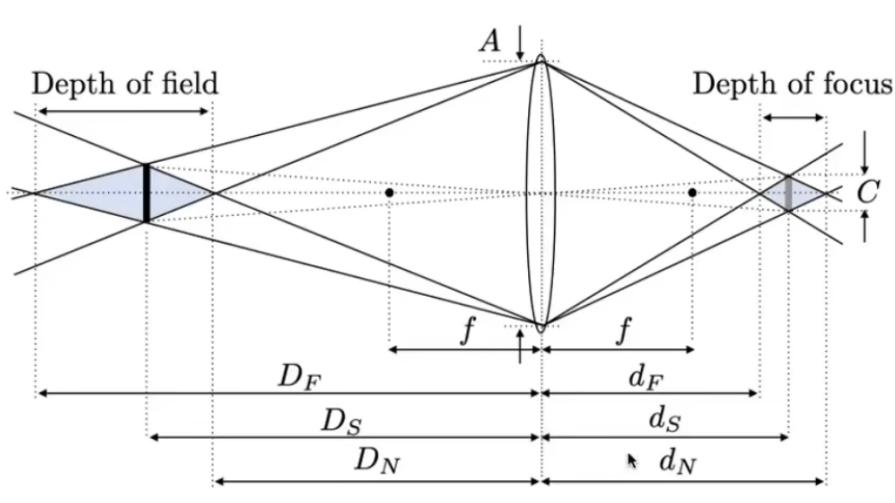
Ray Tracing for Defocus Blur (Thin Lens)

设定好透镜的焦距、像距，从传感器出发向透镜发射一系列光线，折射的光线用透镜公式得出。



景深 (depth of field)

场景中使对应CoC很小（小于或与像素差不多大小）的深度范围。



$$\frac{d_N - d_S}{d_N} = \frac{C}{A}$$

$$\frac{d_S - d_F}{d_F} = \frac{C}{A}$$

$$N = \frac{f}{A}$$

$$\frac{1}{D_F} + \frac{1}{d_F} = \frac{1}{f}$$

$$\frac{1}{D_S} + \frac{1}{d_S} = \frac{1}{f}$$

$$\frac{1}{D_N} + \frac{1}{d_N} = \frac{1}{f}$$

$$DOF = D_F - D_N$$

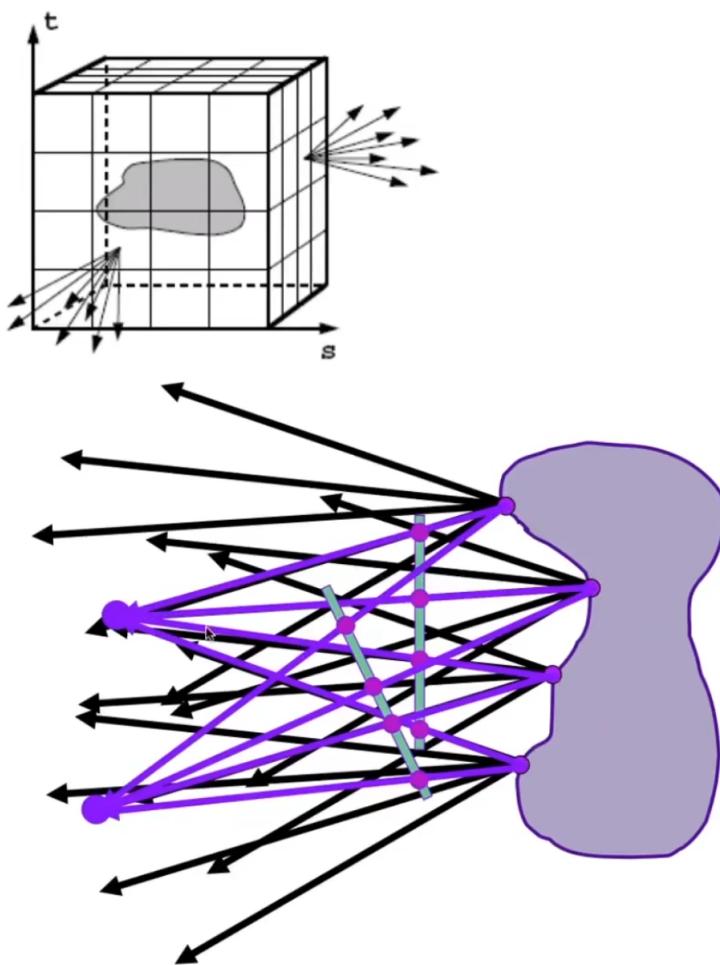
$$D_F = \frac{D_S f^2}{f^2 - NC(D_S - f)} \quad D_N = \frac{D_S f^2}{f^2 + NC(D_S - f)}$$

光场 (Light Field / Lumigraph)

光场的引入

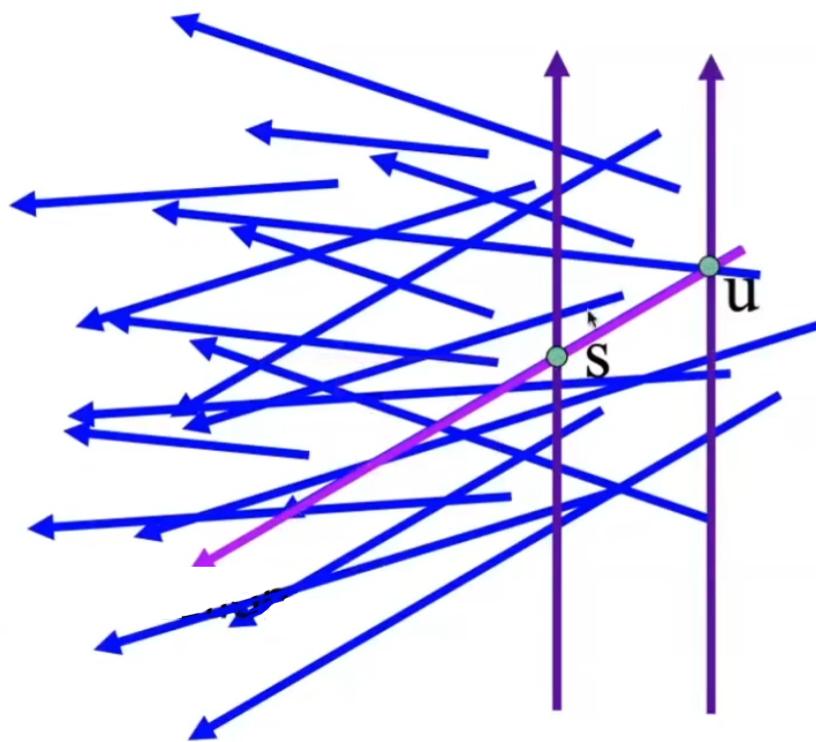
- 全光函数 (The Plenoptic Function) $P(\theta, \phi, \lambda, t, V_x, V_y, V_z)$ 可以用于表述在任何时间任何时刻任何角度看到的任意波长的光强度。

为了描述一个物体，只需要从它外部任何一个角度观察它，记录观察到的每一个点。如果有一个平面能提供相同的信息，则能让我们视觉上得到相同的观察结果。由此，可以引入一个包围盒，包围盒上记录了各个观察方向观察到的各点的样子 (plenoptic surface) 。



光场记录了任何一个位置往任何一个方向去的光的强度。

光的描述可以采用点的位置加上方向，同样可以用两个点的位置进行。为了更好地描述光场，我们在我们关心的物体两侧引入两个平面。



两个平面上的点s,u就可以表述光线。若点s的坐标为(s,t)，点u的坐标为(u,v)，光就可以用s,t,u,v进行描述。

光场的应用

- 光照照相机

运用光场的原理，可以做到先拍照，后进行动态变焦的操作。

色彩 (Color)

- 谱功率密度 (Spectral Power Distribution, SPD)
 - 描述一束光中各波长光线的相对强度。
 - 具有线性可加性。
- 颜色的生物学原理

人眼内有视锥细胞，它们负责识别光的颜色。视锥细胞有三种S、M、L，不同人眼中的比例不一样。

三种不同的细胞对不同波长的光的响应不同，有不同的响应曲线，响应函数分别记作 $r_S(\lambda), r_M(\lambda), r_L(\lambda)$ 。

人们看到的颜色实际上是它们的响应与光的SPD积分的结果，即

$$S = \int r_S(\lambda)s(\lambda)d\lambda$$
$$M = \int r_M(\lambda)s(\lambda)d\lambda$$
$$L = \int r_L(\lambda)s(\lambda)d\lambda$$

人在接收到(S, M, L)后会产生颜色的感觉。

- 同色异谱现象 (Metamerism)

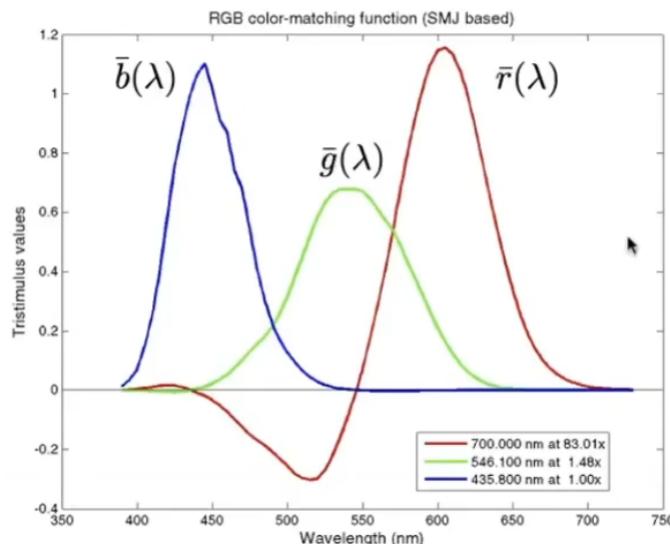
不同的SPD可能对应同一种颜色。因此匹配一种颜色时只需要任意调节SPD，直到想要的颜色出现。

颜色空间 (Color Space)

(s)RGB Color Space

用R、G、B三种颜色去调节得到各种波长对应的颜色。

人们在实验中发现有些颜色并不能通过直接混合得到，而可以通过目标颜色上加颜色 (R) 得到我们能直接配出的颜色。这种在目标颜色上加颜色的比例记作负数。实验结果如下：



同样地可以计算得到RGB分量

$$R = \int_{\lambda} s(\lambda) \bar{r}(\lambda) d\lambda$$

$$G = \int_{\lambda} s(\lambda) \bar{g}(\lambda) d\lambda$$

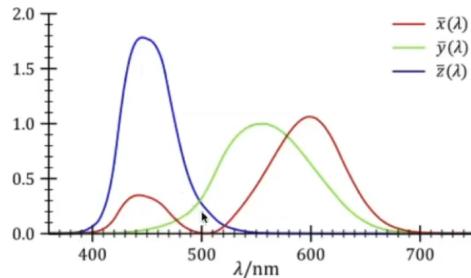
$$B = \int_{\lambda} s(\lambda) \bar{b}(\lambda) d\lambda$$

- RGB系统并不能定义所有颜色。

XYZ Color Space

直接定义各个颜色匹配函数如下图：

CIE XYZ color matching functions



其中Y分量代表颜色的亮度 (luminance) 。

颜色匹配函数满足以下性质：

- 匹配函数都严格非负。
- 覆盖了所有可以观察到的颜色。

为了将该色域可视化，对各分量作归一化处理，定义归一化后的变量为色度 (Chromaticity) 。

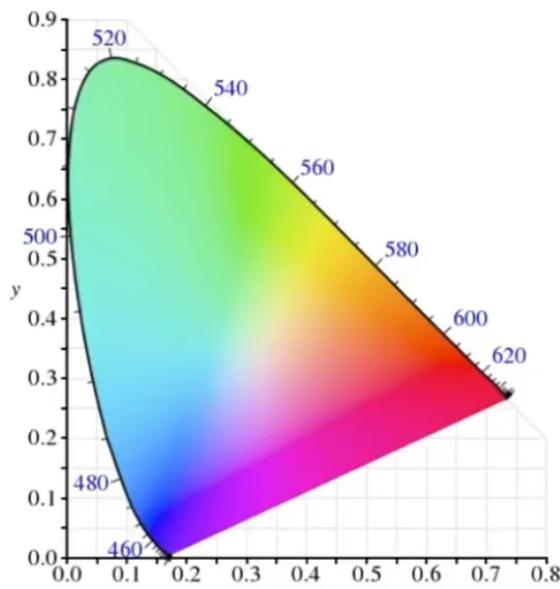
$$x = \frac{X}{X + Y + Z}$$

$$y = \frac{Y}{X + Y + Z}$$

$$z = \frac{Z}{X + Y + Z}$$

因为 $x+y+z=1$ ，所以只需要记录两个参数（通常是 x, y ）。

实际上，该图仍然受三个参数 X, Y, Z 控制，为了改变 x, y ，人们往往固定 Y ，改变 X 和 Z ，因为 Y 控制的只是亮度。所以最终可视化的色域图实际上是一固定亮度下的图。



色域图边缘的颜色最纯净，最中心的白色最不纯净。

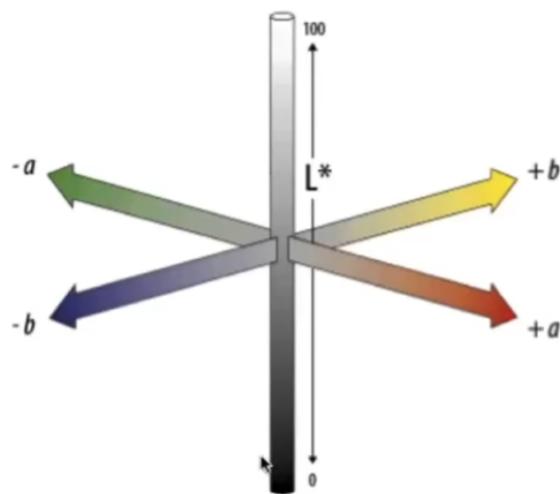
HSV Color Space (Hue-Saturation-Value/Brightness/Lightness)

色调-饱和度-亮度

大量运用于取色器。

LAB Color Space

L是亮度，a、b是两对互补色，其中a为红绿互补色对，b为黄蓝互补色对。



减色系统

CMYK (Cyan、Magenta、Yellow and black)

色域 (Gamut)

一个颜色空间所有可能显示的颜色。