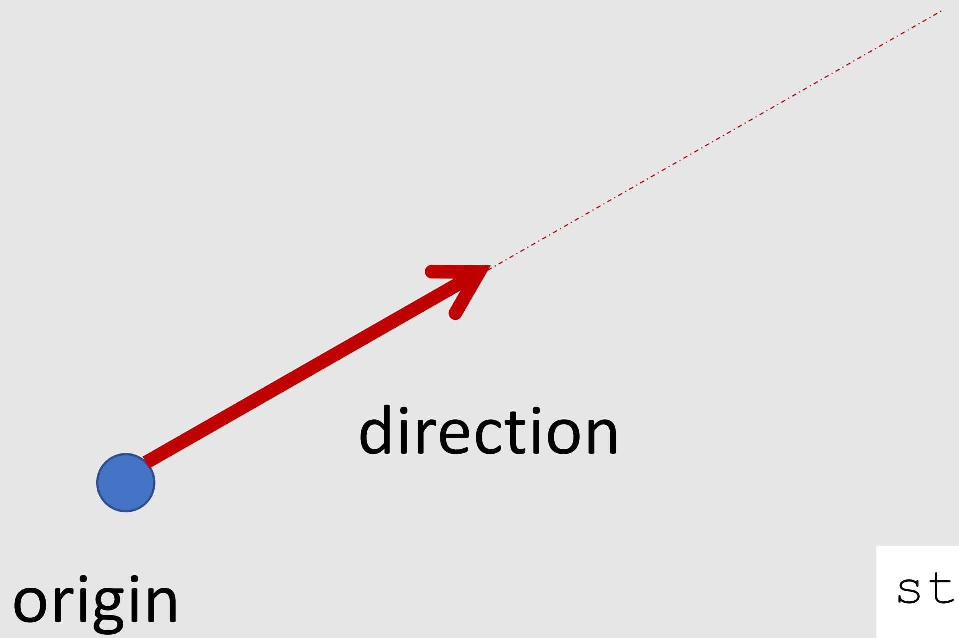
Ray Casting

What is a Ray (half line, 半直線)?

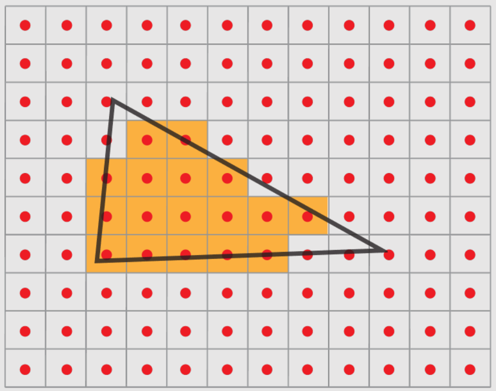



```
struct{
  Vector3d origin;
  Vector3d direction;
};
```

Rasterization vs Ray Casting

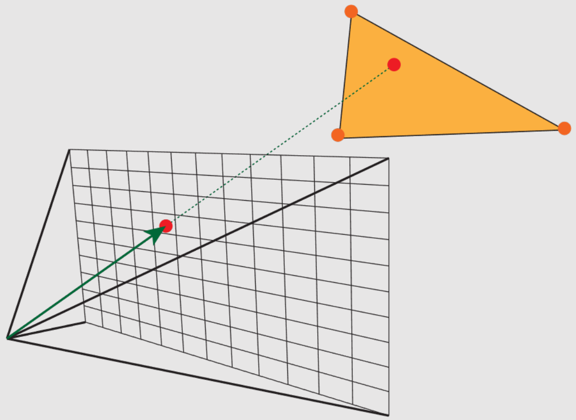
Rasterization

```
for each triangle
  for each pixel (x,y)
    if (x,y) is inside triangle
      framebuffer[x,y]=shade()
```

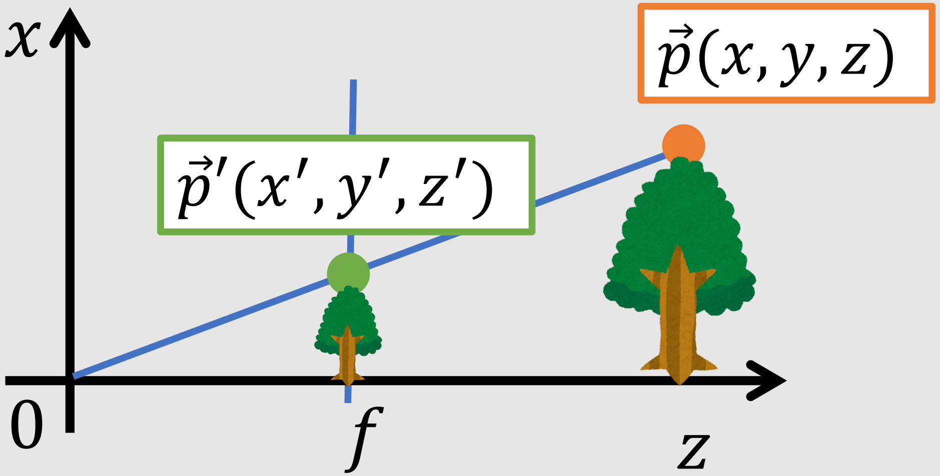


Ray Casting

```
for each pixel (x,y)
  for each triangle
    if ray hits triangle
      framebuffer[x,y]=shade()
```

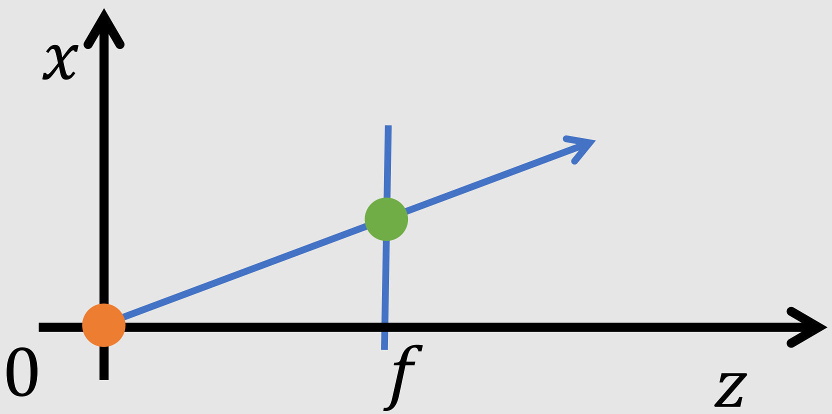


Ray at Pixel: 3x3 Homography Matrix



$$z' = f, \qquad \frac{x'}{x} = \frac{y'}{y} = \frac{z'}{z}$$

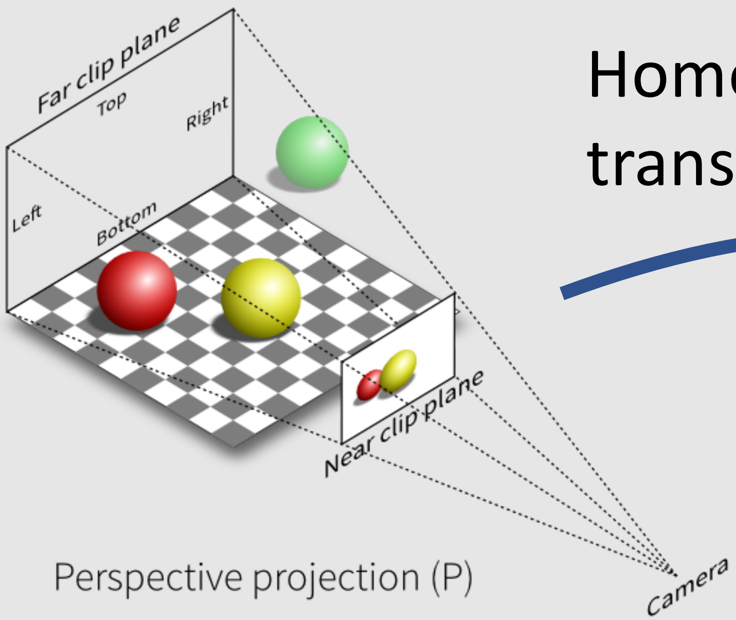
$$\begin{Bmatrix} x' \\ y' \\ 1 \end{Bmatrix} \propto \begin{Bmatrix} x'' \\ y'' \\ w \end{Bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}$$



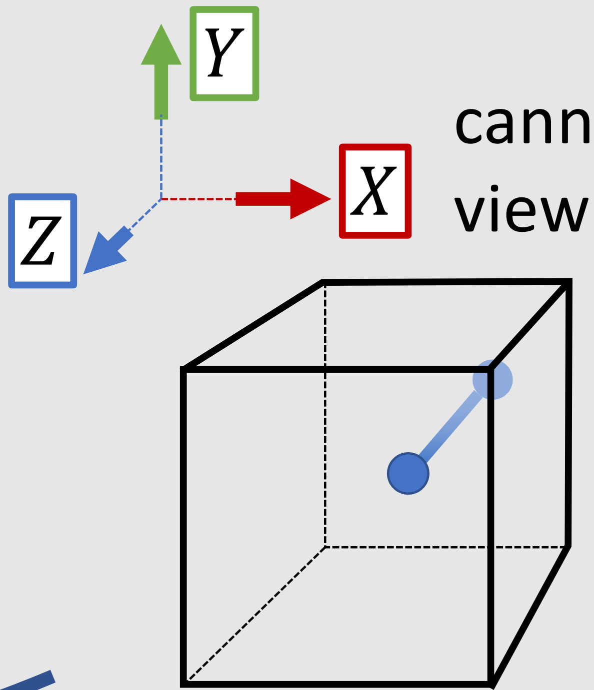
ray from camera

$$\vec{s} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix}, \vec{d} = \begin{Bmatrix} x \\ y \\ f \end{Bmatrix}$$

Ray at Pixel: 4x4 Homography Matrix



Homography transformation H



cannonical view volume

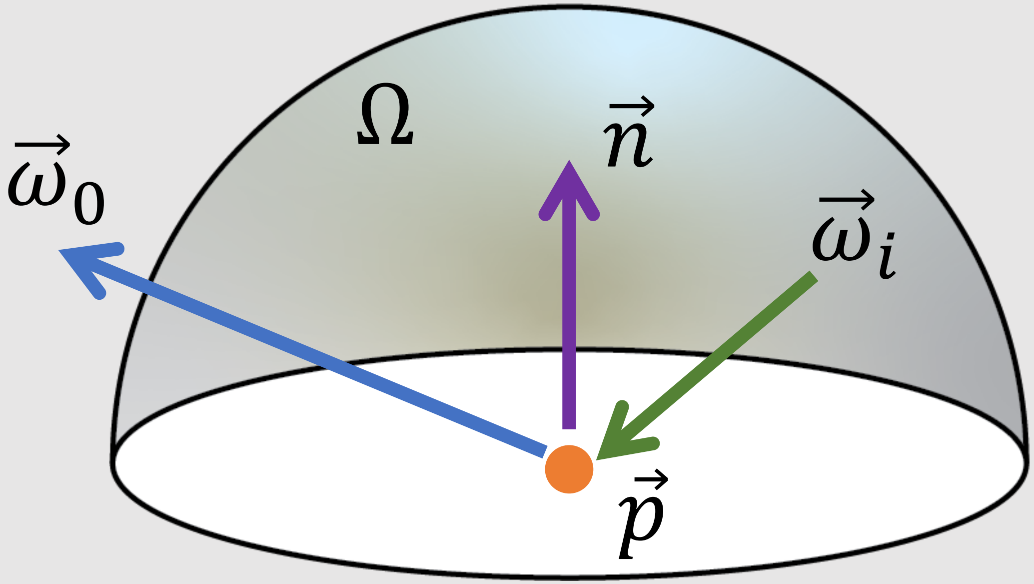
$$\vec{s} = H^{-1}\vec{s}'$$
$$\vec{t} = H^{-1}(\vec{t}' - \vec{s}')$$

Homography transformation H^{-1}

$$\vec{s}' = \begin{Bmatrix} x \\ y \\ 1 \end{Bmatrix}, \vec{t}' = \begin{Bmatrix} x \\ y \\ -1 \end{Bmatrix}$$

Rendering Equation [Kajiya 1986]

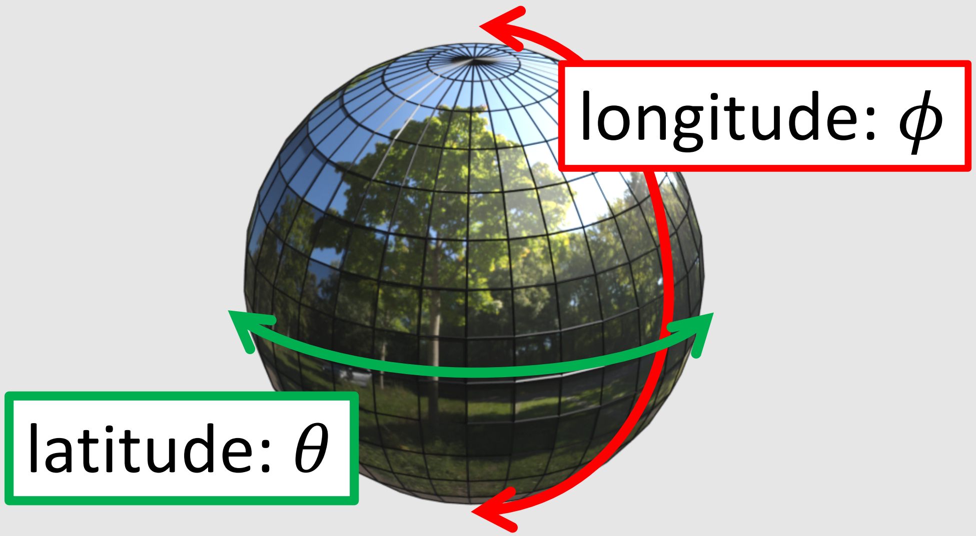
$$L_o(\vec{p}, \vec{\omega}_0) = \int_\Omega f(\vec{\omega}_0, \vec{\omega}_i) L_i(\vec{p}, \vec{\omega}_i)(\vec{\omega}_i \cdot \vec{n})\, d\vec{\omega}_i$$



James T. Kajiya. 1986. The rendering equation. SIGGRAPH Comput. Graph. 20, 4

Environment Map

Far light approximation: $L_i(\vec{p}, \vec{\omega}_i) \simeq L_i(\vec{\omega}_i)$



High Dynamic Range (HDR) Image

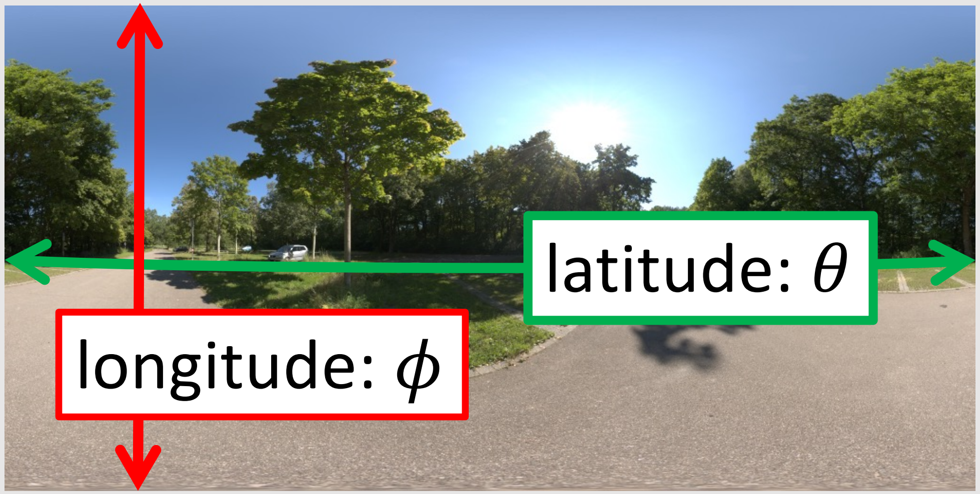


Image from PolyHaven: https://polyhaven.com/a/rooitou_park

Ambient Light: Uniform Light

- Omni-directional, uniform color, uniform intensity

- (ambient light) + (no occusion) + (Lambertian reflection)
  = constant reflection

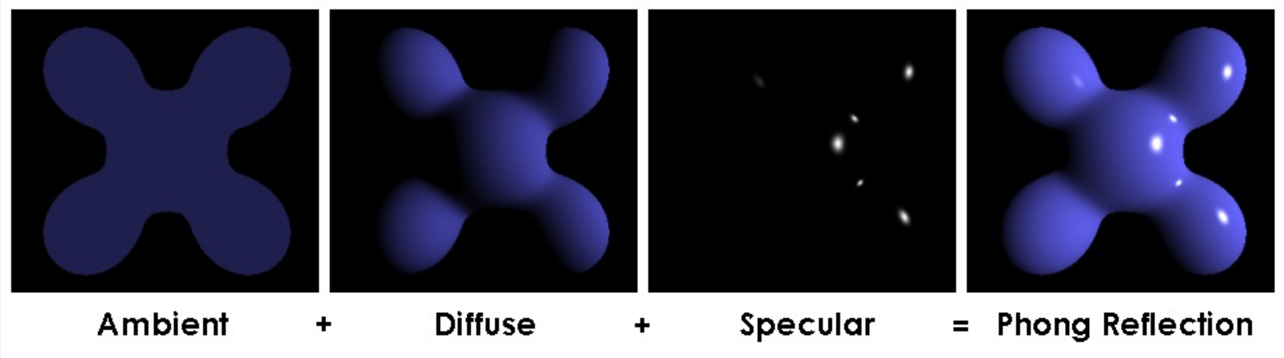


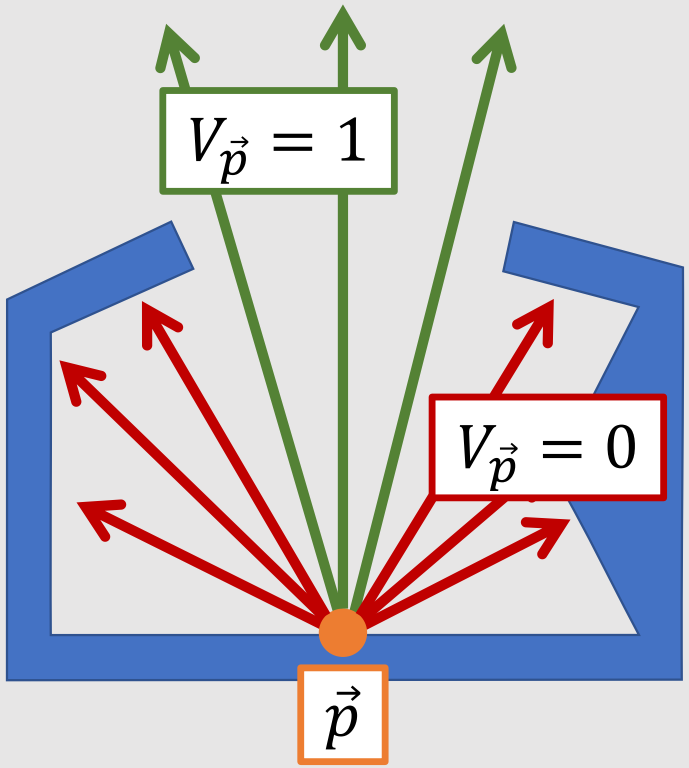
Image Credit: Brad Smith @ Wikipedia

Ambient Occlusion: Occlusion Ratio For Ambient Light

$$A_{\vec{p}} = \frac{1}{\pi} \int_{\Omega} V_{\vec{p}}(\vec{\omega})(\vec{n} \cdot \vec{\omega})\, d\vec{\omega}$$

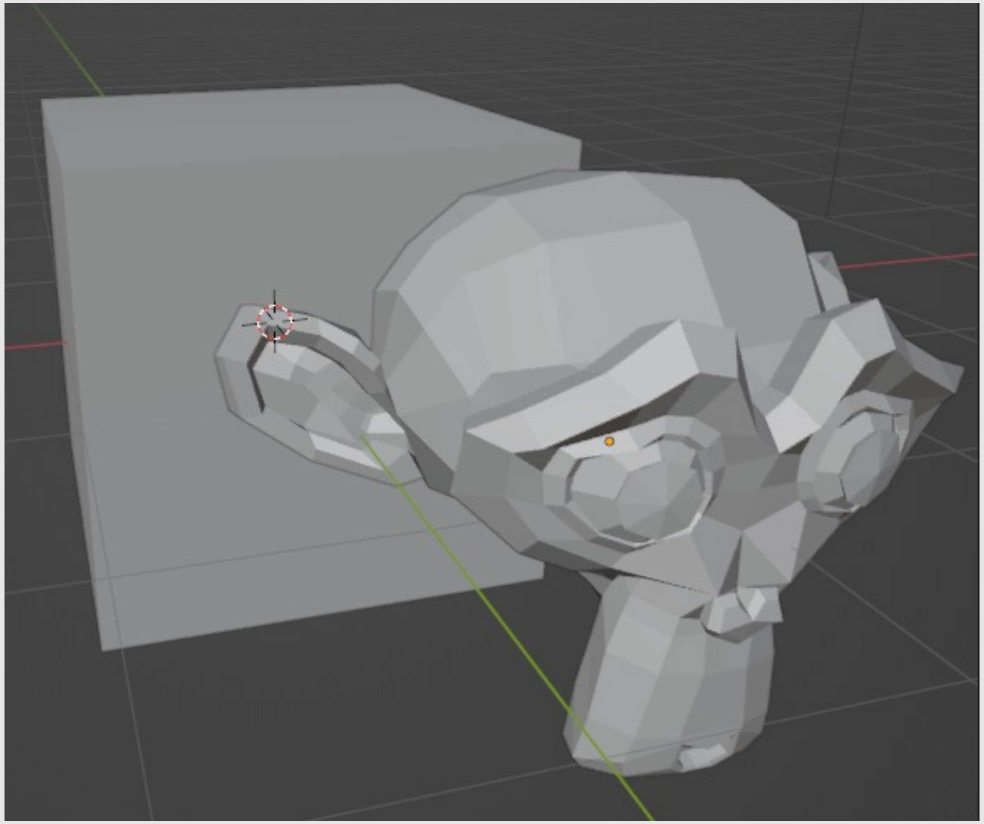
normalizing constant

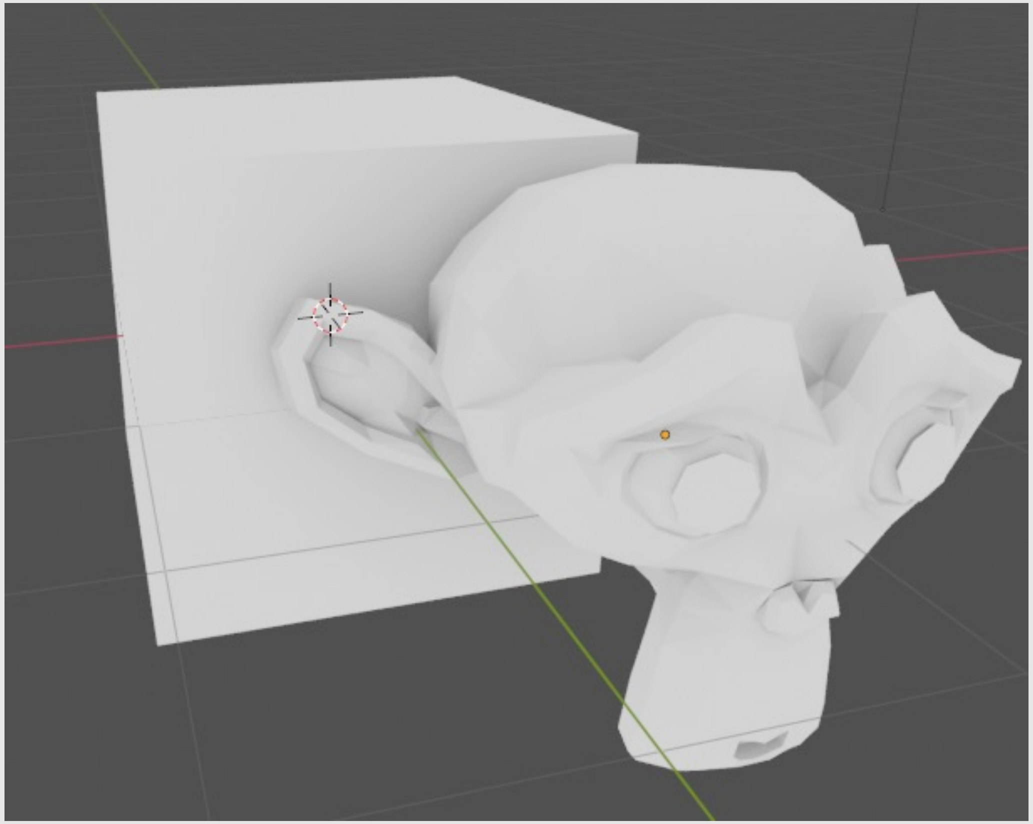
$A_{\vec{p}} = 1$: no occlusion



Example of Ambient Occlusion

- Ambient occlusion is fully depends on geometry





Monte Carlo Integration

- Integration of a “difficult” function (i.e., we can only evaluate at discrete sample locations)

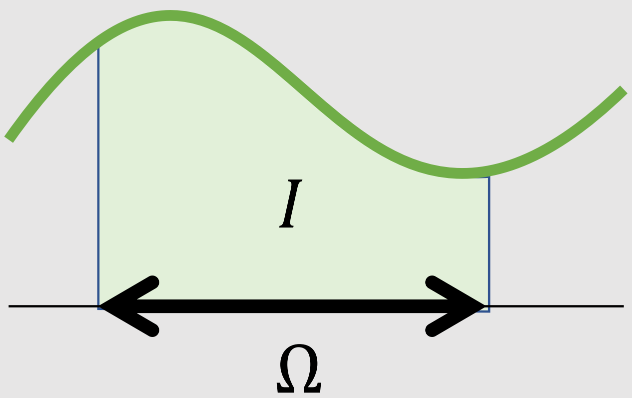
$$I = \int_\Omega f(x)dx$$

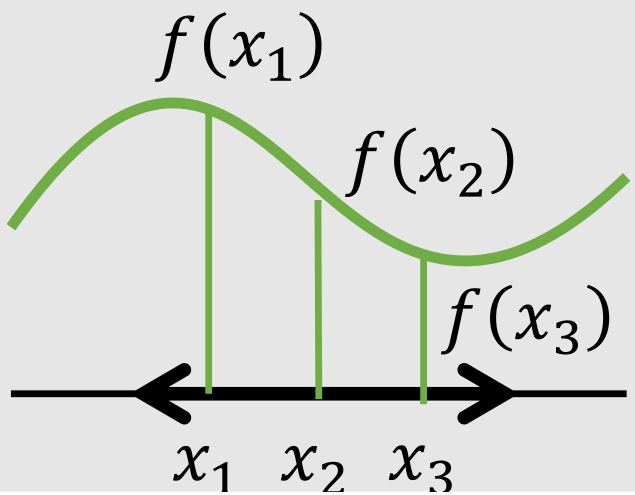
approximation

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

$$V = \int_\Omega dx$$

$$x_1, \ldots, x_N \in \Omega$$



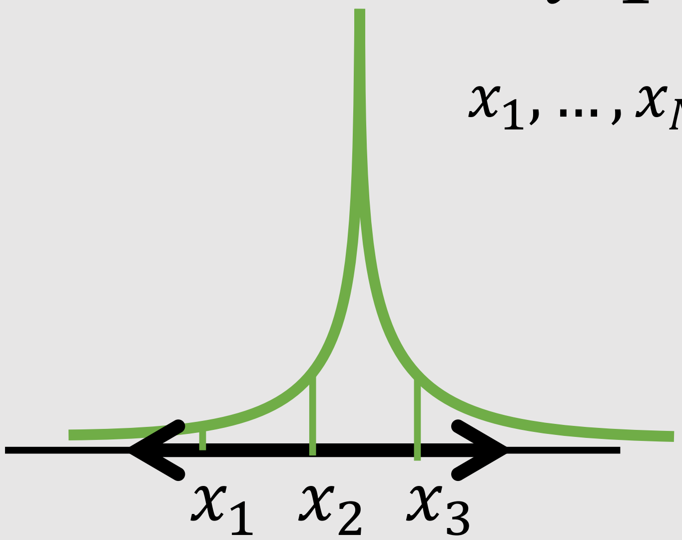


Acceleration: Importance Sampling

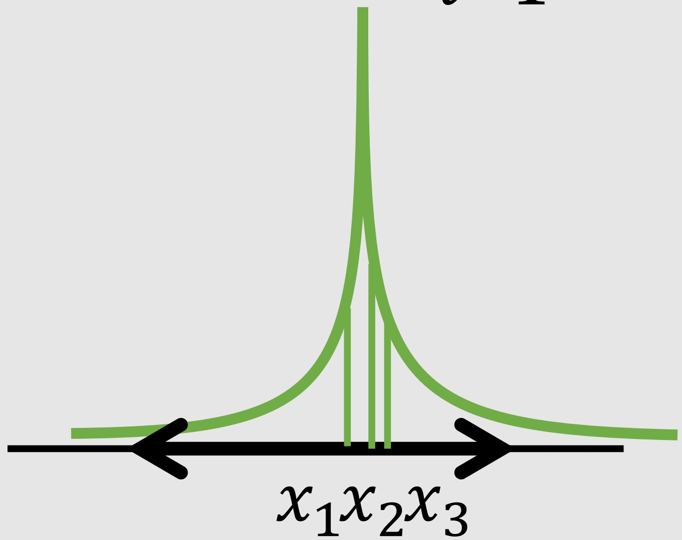
- Sample densely where the integrand is large

$$E\big(f(X)\big) \simeq \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

$$x_1, \ldots, x_N \in \Omega$$



$$E\big(f(X)\big) \simeq \frac{1}{N}\sum_{i=1}^{N} \frac{f(x_i)}{\mathrm{pdf}(x_i)}$$



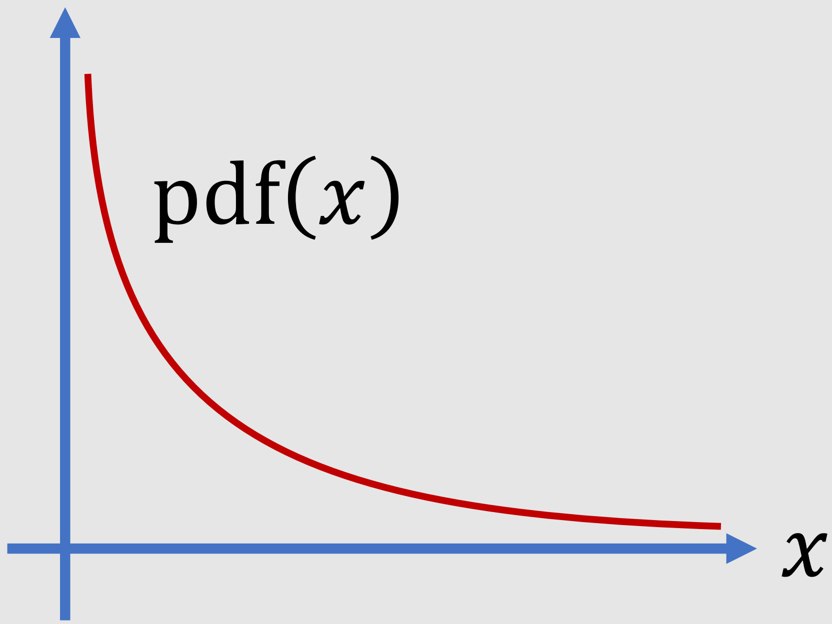
Probablity Density Function (PDF,確率密度関数)

- PDF is a density, not prbobablity itself so tometime exceed 1

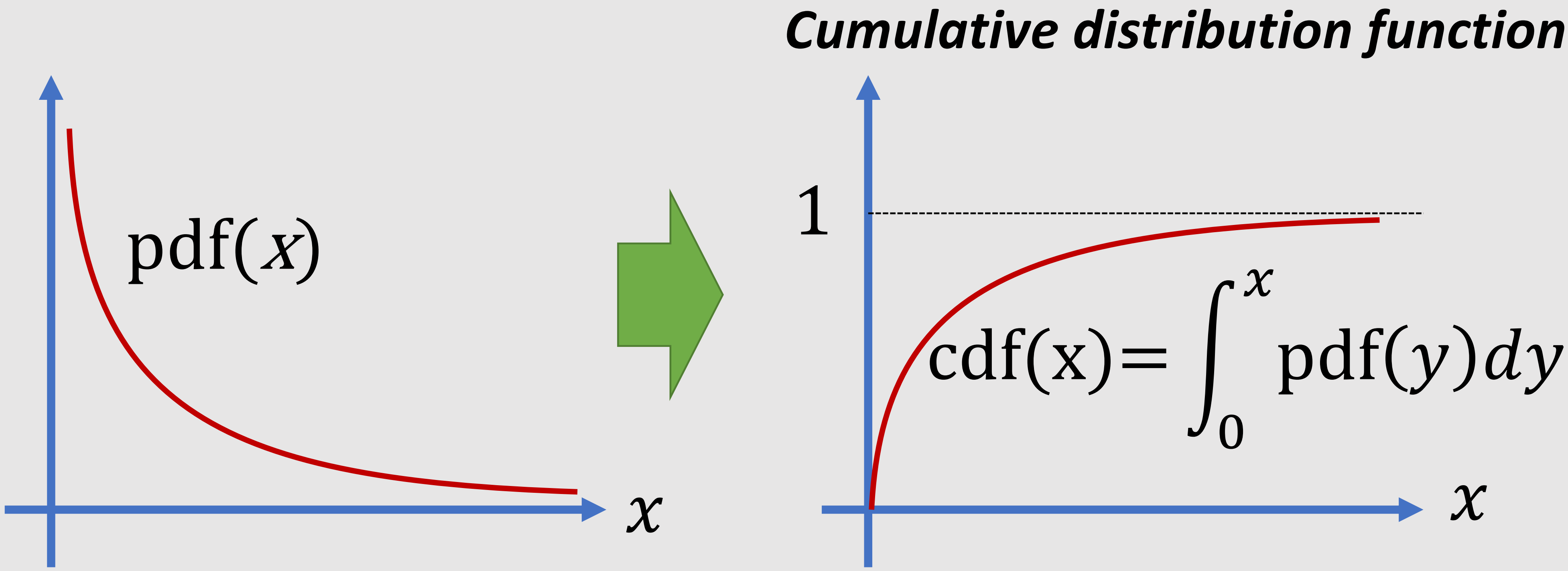
$$\mathrm{pdf}(x) > 0 \; for \; all \; x \in \Omega$$

$$\int_{\Omega} \mathrm{pdf}(x) dx = 1$$

$$P(a \le X \le b) = \int_a^b \mathrm{pdf}(x)\, dx$$



Inverse Transform Method



$$u_i \sim U(0,1)$$

$$x_i = \mathrm{cdf}^{-1}(u_i) \sim \mathrm{pdf}(X)$$

Cosine Importance Sampling

Uniform sampling over hemisphere: $\mathrm{pdf}(\omega_i) = \frac{1}{2\pi}$

$$A_{\vec{p}} \simeq \frac{1}{\pi}\frac{1}{N}\sum_{i=1}^{N}\frac{V_{\vec{p}}(\vec{\omega})(\vec{n}\cdot\vec{\omega})}{\mathrm{pdf}(\omega_i)} = \frac{2}{N}\sum_{i=1}^{N}V_{\vec{p}}(\vec{\omega})(\vec{n}\cdot\vec{\omega})$$

Cosine importance sampling: $\mathrm{pdf}(\omega_i) = \frac{\vec{n}\cdot\vec{\omega}_i}{\pi}$

$$A_{\vec{p}} \simeq \frac{1}{\pi}\frac{1}{N}\sum_{i=1}^{N}\frac{V_{\vec{p}}(\vec{\omega})(\vec{n}\cdot\vec{\omega})}{\mathrm{pdf}(\omega_i)} = \frac{1}{N}\sum_{i=1}^{N}V_{\vec{p}}(\vec{\omega})$$

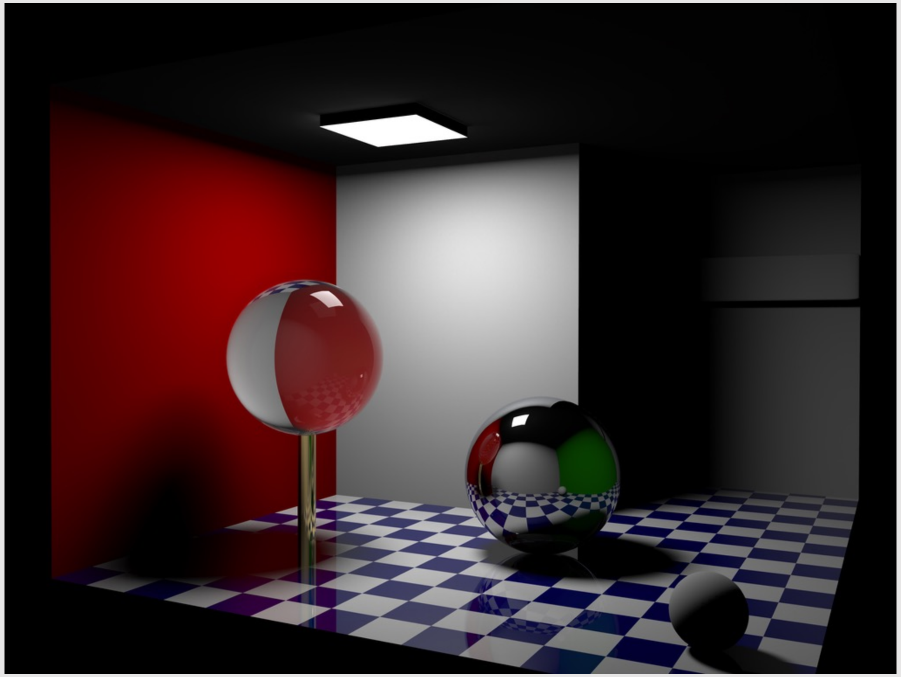
Strategy for Cosine Importance Sampling

- Polar coodinate θ, ϕ
- the Jacobian for polar coordinate: $\sin\theta$
- $\mathrm{pdf}(\theta) = \sin\theta\cos\theta$
- Cumulative distribution: $\mathrm{cdf}(\theta) = \int \mathrm{pdf}(\theta) d\theta = \cos^2\theta$
- Inverse cumulative distribution: $\mathrm{cdf}^{-1}(x) = \cos^{-1}(\sqrt{x})$

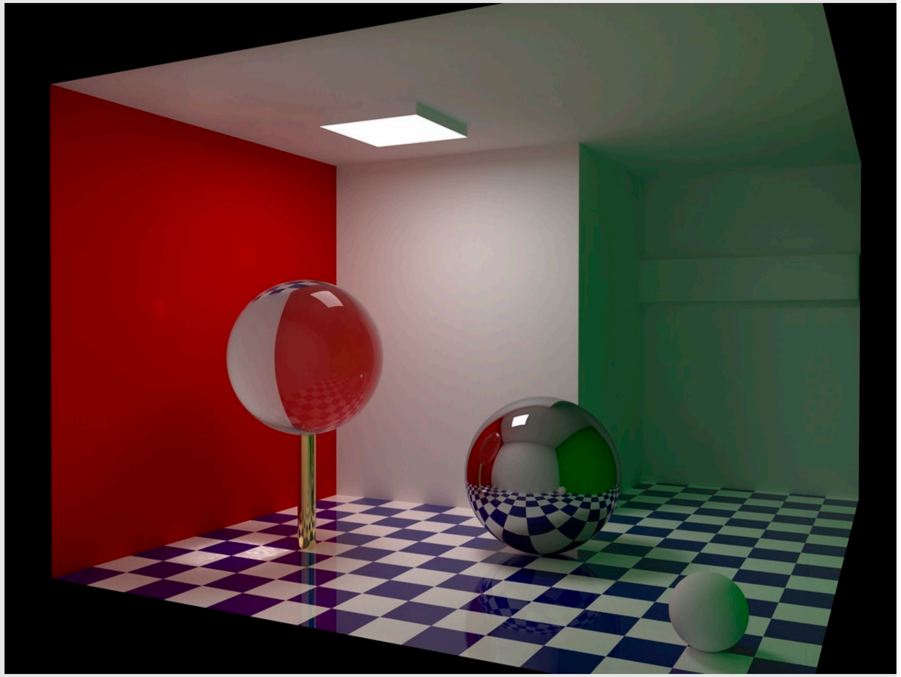
...

Local Illumination vs Global Illumination

Light come only from lighting



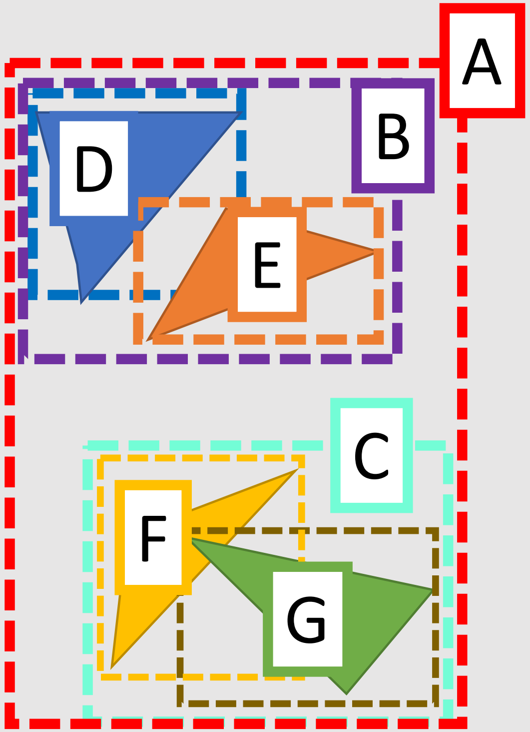
Every surface is lightsource

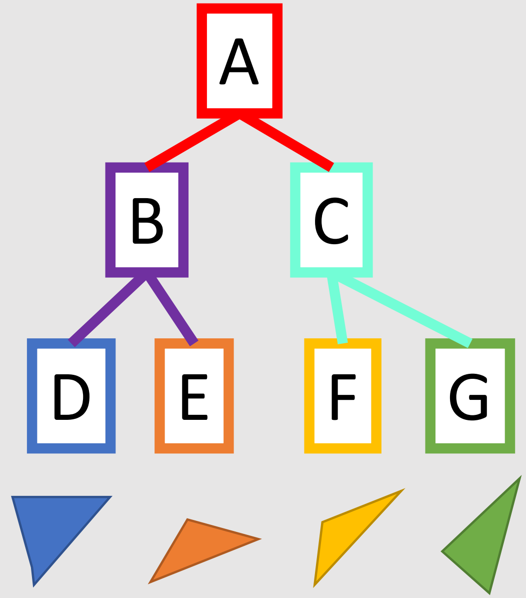


Ray Triangle Collision

Bounding Volume Hierarchy (BVH)

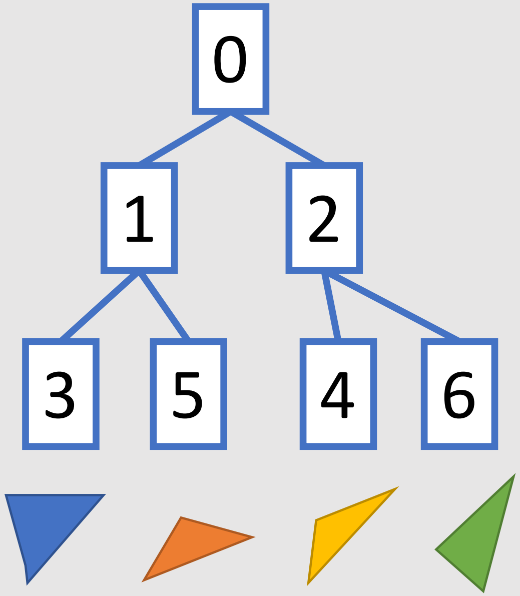
- Near triangles are in the same branch
- Each node has a BV that includes two child BVs





Example of BVH Data Structure in C++

index	0	1	2	3	4	5	6
left-child index	1	3	4	tri index	tri index	tri index	tri index
Right-child index	2	5	6	-1	-1	-1	-1
BV data	…	…	…	…	…	…	…

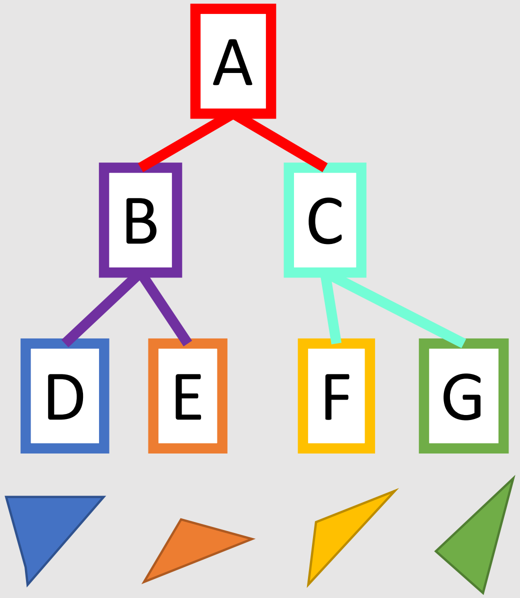



```
template <class T>
class CNodeBVH {
       unsigned int ichild_left;
       unsigned int ichild_right;
       T BV;
};

std::vector<CNodeBVH<CAABB>> aNodeBVH;
```

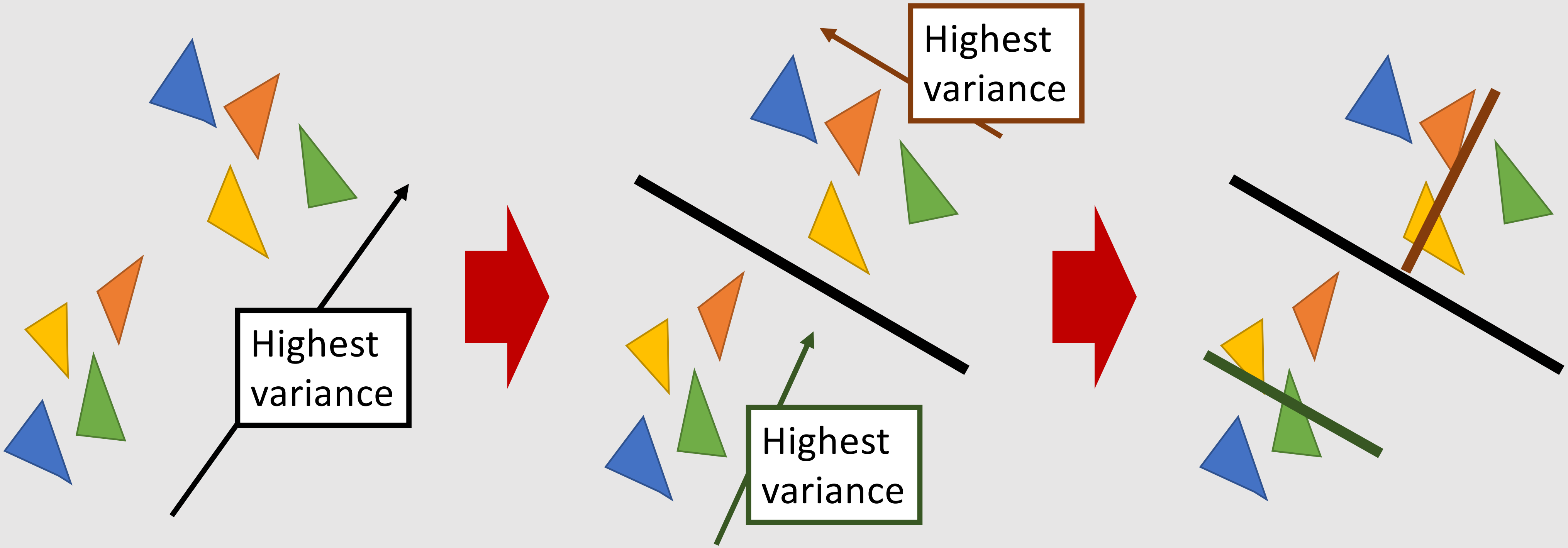
Evaluation of BVH using Recursion

- Ask question to the root node -> if true the node asks the same question to two child nodes and so on



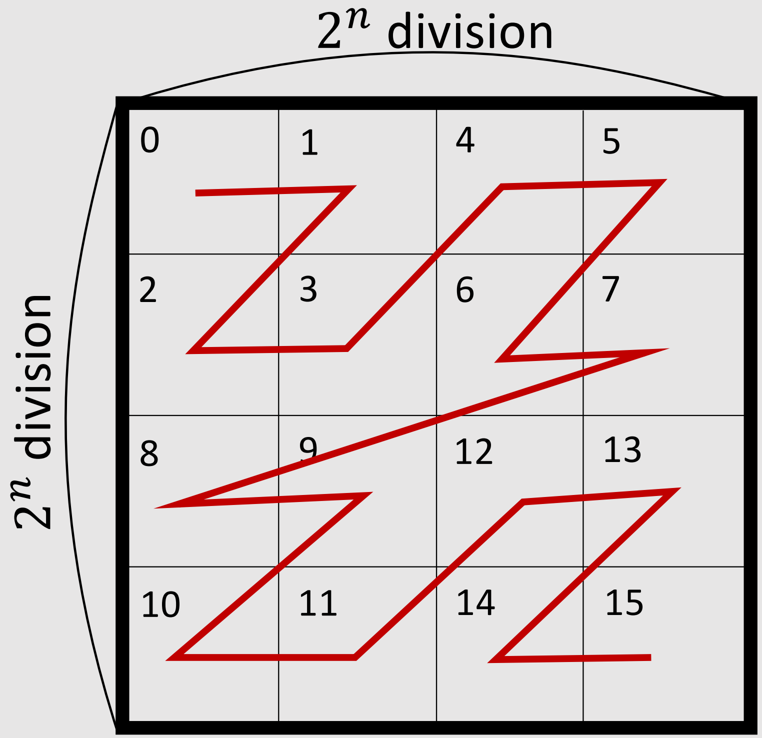
Top-down Approach to Build BVH

- Use PCA for separating triangles into two groups

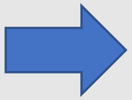


Linear BVH: Fully Parallel Construction

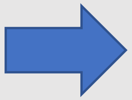
- Construct BVH based on Morton code (i.e., Z-order curve)
- Two cells with close Morton codes tends to be near



2D square domain with 2^n edge division



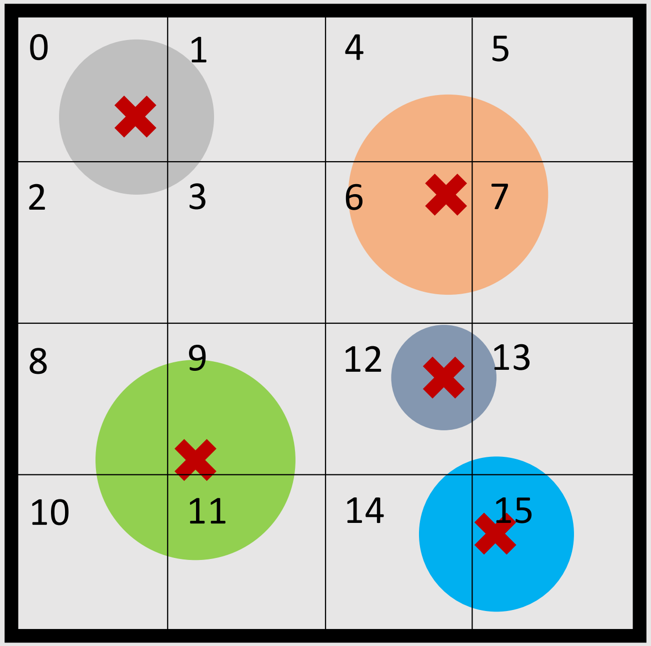
2^{2n} number of cells

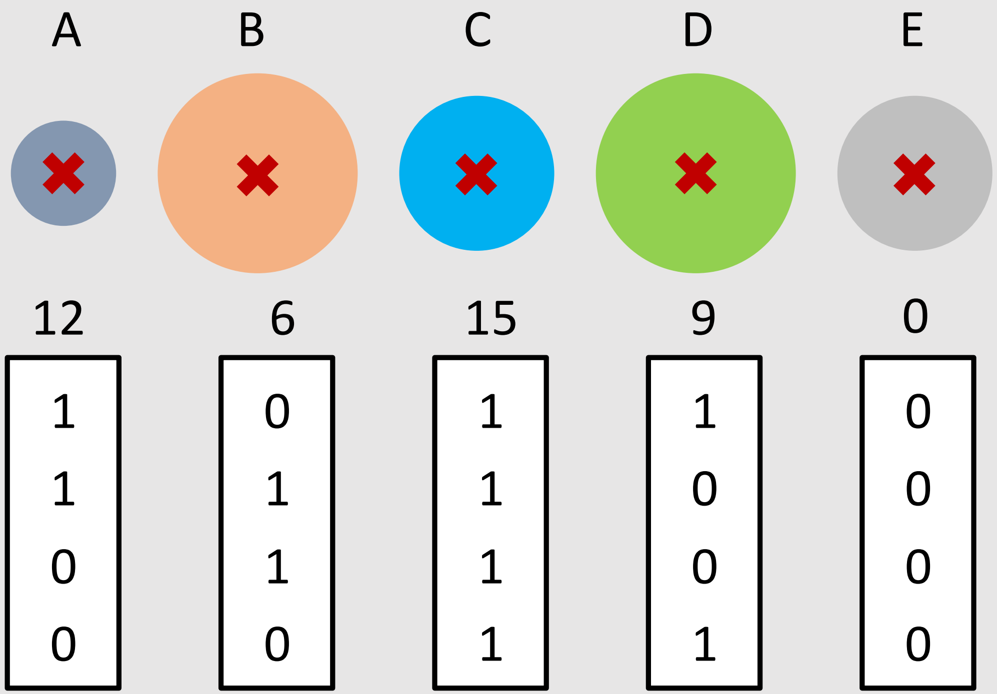


Cell index is size of $2n$ in binary

Linear BVH: Fully Parallel Construction

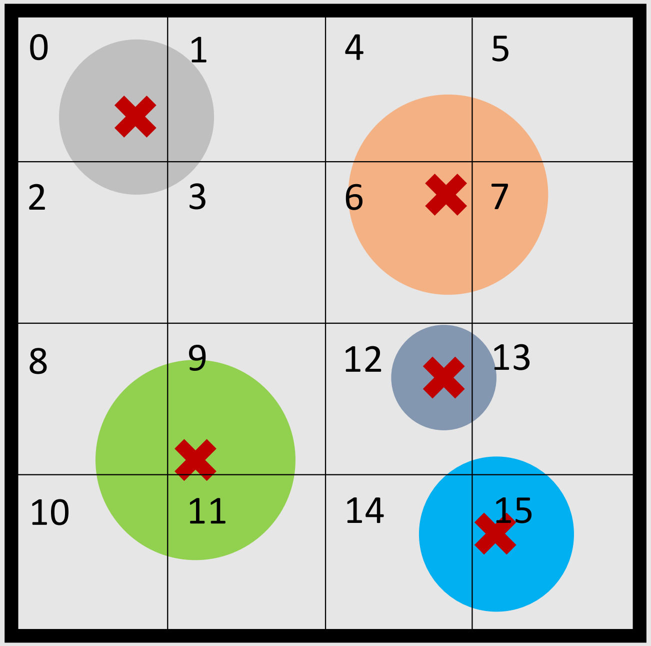
- Convert XYZ coordinate into 1D (linear) integer coordinate

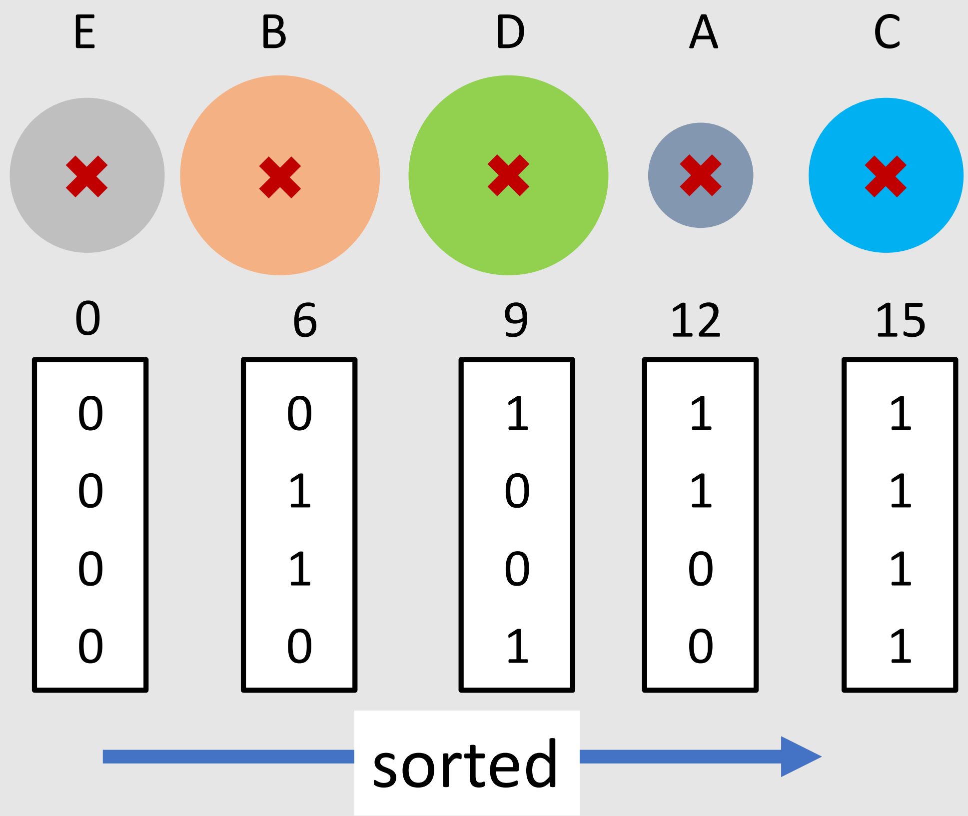




Linear BVH: Fully Parallel Construction

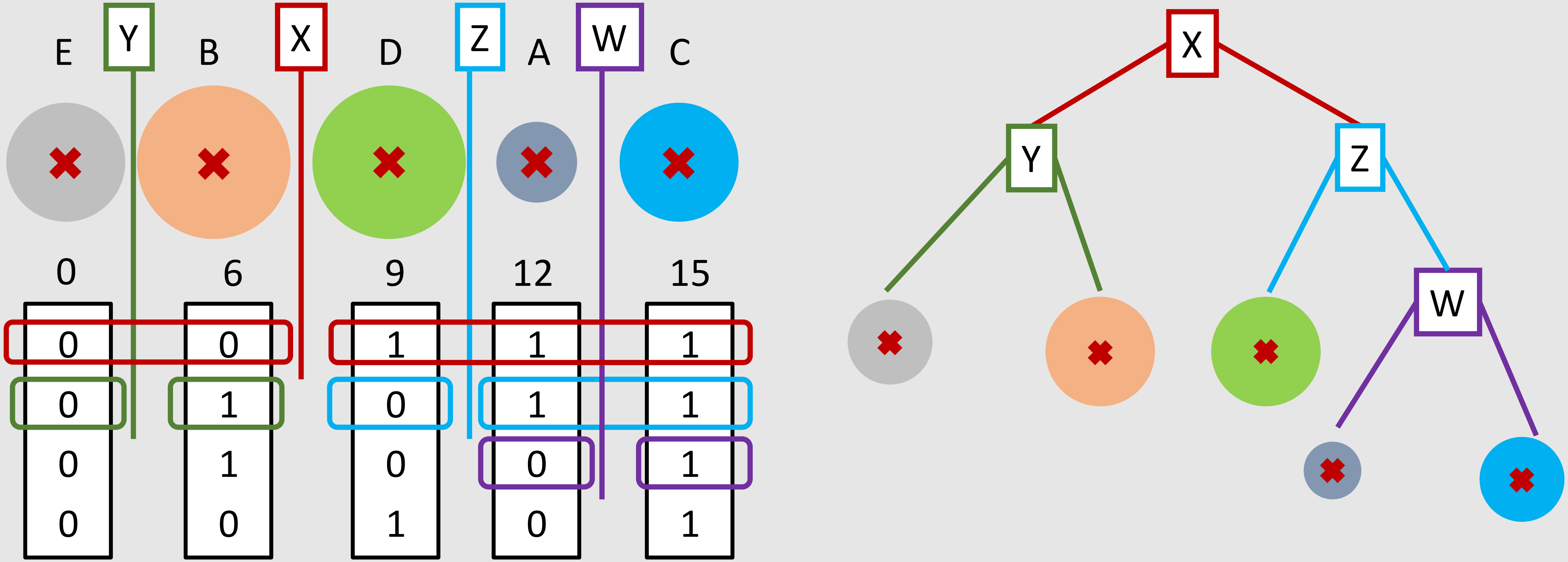
- Sort objects by their Morton codes





From Morton Code to BVH Tree

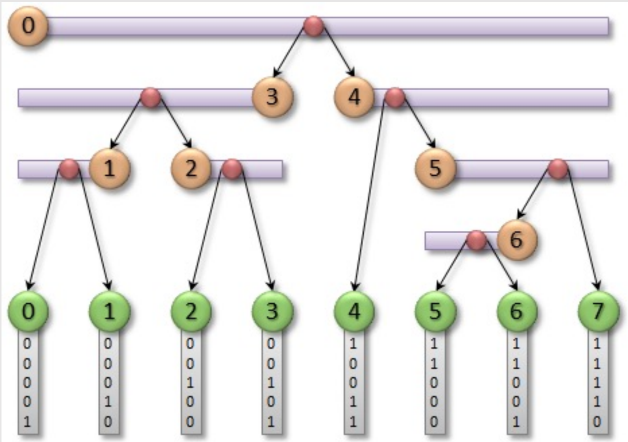
- Divide tree when digits of sorted Morton codes are different



Reference on Linear-BVH

- Thinking Parallel, Part III: Tree Construction on the GPU

by Tero Karras



https://developer.nvidia.com/blog/thinking-parallel-part-iii-tree-construction-gpu/