**04 Ray Tracing**

**Rendering**

**What:**

渲染是一个将一组对象作为输入并生成一个像素数组作为输出的过程。

**Why:**

**How:**

**渲染分类**

Object-order Rendering：依次考虑每个对象，为每个对象找到并更新其影响的所有像素。

Image-order Rendering：依次考虑每个像素，对每个像素找到影响它的所有对象并计算像素值。

**Ray tracing**

**What:**

光线跟踪是一种用于绘制三维场景的Image-order Rendering算法。

**Why:**

**How:**

光线追踪原理

A ray tracer works by computing one pixel at a time, and for each pixel the basic

task is to find the object that is seen at that pixel’s position in the image. Each

pixel “looks” in a different direction, and any object that is seen by a pixel must

intersect the viewing ray, a line that emanates from the viewpoint in the direction

that pixel is looking. The particular object we want is the one that intersects

the viewing ray nearest the camera, since it blocks the view of any other objects

behind it. Once that object is found, a shading computation uses the intersection

point, surface normal, and other information (depending on the desired type of

rendering) to determine the color of the pixel.

**简单的光线追踪器的三个部分**

1. Ray generation, which computes the origin and direction of each pixel’s

viewing ray based on the camera geometry;

2. Ray intersection, which finds the closest object intersecting the viewing ray;

3. Shading, which computes the pixel color based on the results of ray intersection.

Ray-Triangle Intersection

使用三角形重心坐标方程和三角形的重心坐标方程联立。

xe + txd = f(u,v)

ye + tyd = g(u,v)

ze + tzd = h(u,v)

t,u,v为未知数；

通过重心坐标方程结果α，β，γ可以判断交点是否在三角形内。

Ray-Polygon Intersection

联立方程

(p – p1) · n = 0

p = e + td

求出p，然后判断p是否在多边形内。

把点p和多边形投射到xy平面，然后从p发出射线，根据交点个数奇偶判断。

另一个方法是将多边形划分为多个三角形。

Shading

Lambertian Shading

L = kd I max(0, n·l)

Blinn-Phong Shading

H = (v + l) / |v + l|

L = kd I max(0, n·l) + ks I max(0,n·h)p

Ambient Shading

L = kaIa + kd I max(0, n·l) + ks I max(0,n·h)p