# 加入阴影会怎样？

华中科技大学软件学院　万琳

# 提纲

1. 阴影的概念
2. 阴影计算算法

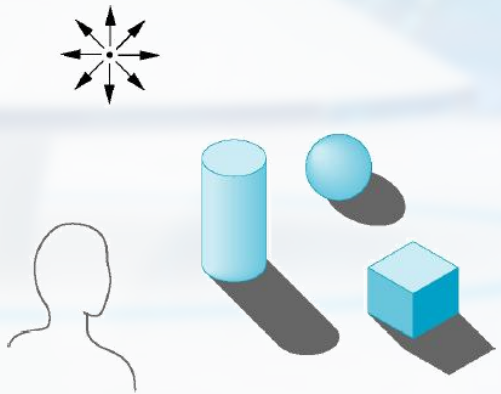
# 1 阴影的概念

生活中的常识：有光就有影

# 1 阴影的概念

阴影是由于物体截断了光线而产生的，所以如果光源位于物体一侧的话，阴影总是位于物体的另一侧，也就是与光源相反的一侧。

从理论上来说，从视点以及从光源看过去都是可见的面不会落在阴影中，只有那些从视点看过去是可见的，而从光源看过去是不可见的面，肯定落在阴影之内。



## 2 阴影的计算算法

◆基本思想：将视点移到光源位置

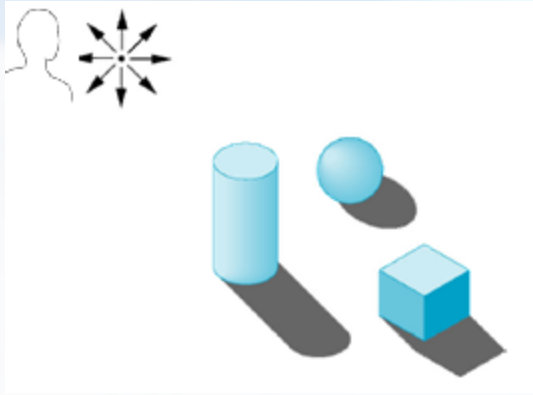
用多边形区域排序消隐算法将多边形分成两大类：

向光多边形和背光多边形。

➢向光多边形是指那些从光源看过去是可见的多边形；

➢背光多边形是指那些从光源看过去是可见的多边形，包括被其它面遮挡了的多边形和反向面多边形。

向光多边形不在阴影区内，背光多边形在阴影区内。



## 2 阴影的计算算法

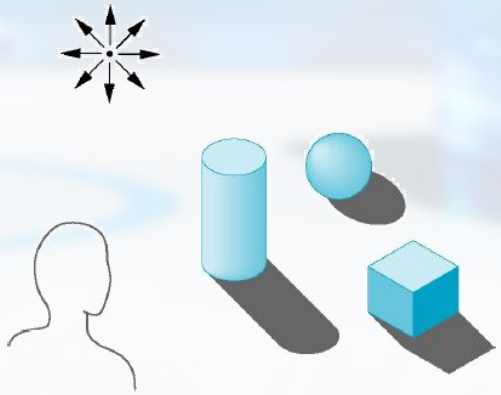
◆基本思想：将视点移到原来的观察位置

对向光多边形和背光多边形进行消隐，选用一种光照模型计算多边形的亮度，就可得到有阴影效果的图形。

若选用之前的Phong模型：

➢对于背光多边形，由于不能得到光源的直接照射，只有环境光对其光强有贡献，因此关闭漫反射和镜面反射；

➢对于向光多边形，正常进行光照计算。

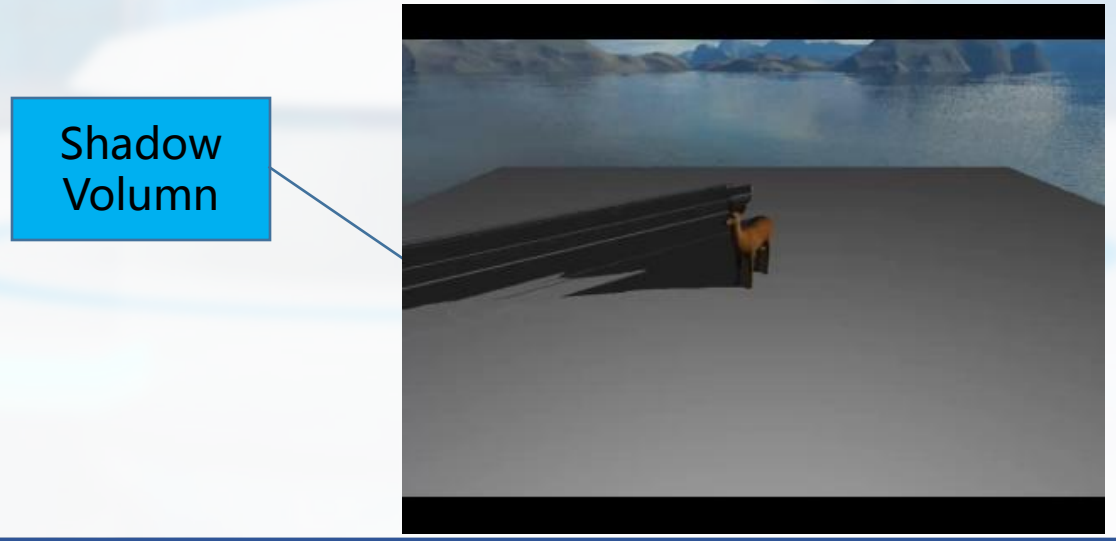


## 2 阴影的计算算法

◆主流算法

➢**Shadow Mapping**：一个物体之所以会处在阴影当中，是由于在它和光源之间存在着遮蔽物，或者说遮蔽物离光源的距离比物体要近。

➢**Shadow Volumn**：根据光源和遮蔽物的位置关系计算出场景中会产生阴影的区域，然后对所有物体进行检测，以确定其会不会受阴影的影响。



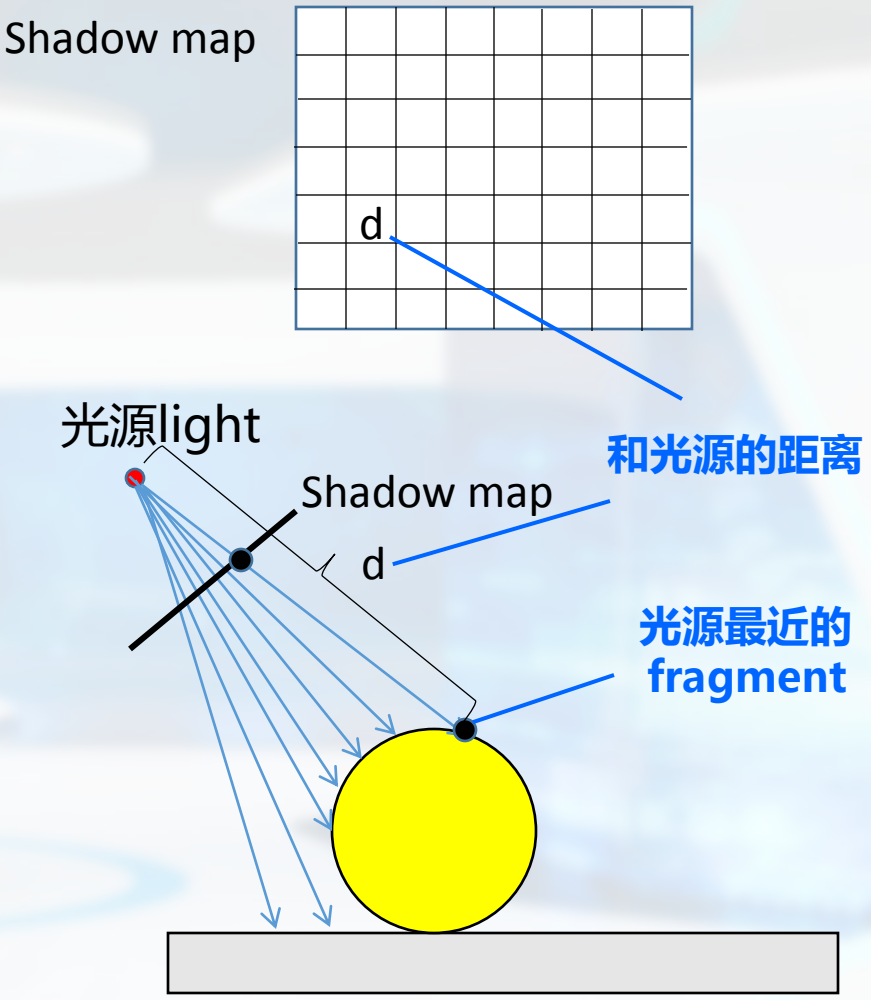
# 2 阴影的计算算法

## ◆主流算法

### ➢Shadow Mapping：

Step 1: 以光源为视点，或者说在光源坐标系下面对整个场景进行渲染，目的是要得到一副所有物体相对于光源的 depth map（也就是我们所说的shadow map），也就是这副图像中每个象素的值代表着场景里面离光源最近的 fragment 的深度值。

由于这个部分我们感兴趣的只是象素的深度值，所以可以把所有的光照计算关掉。

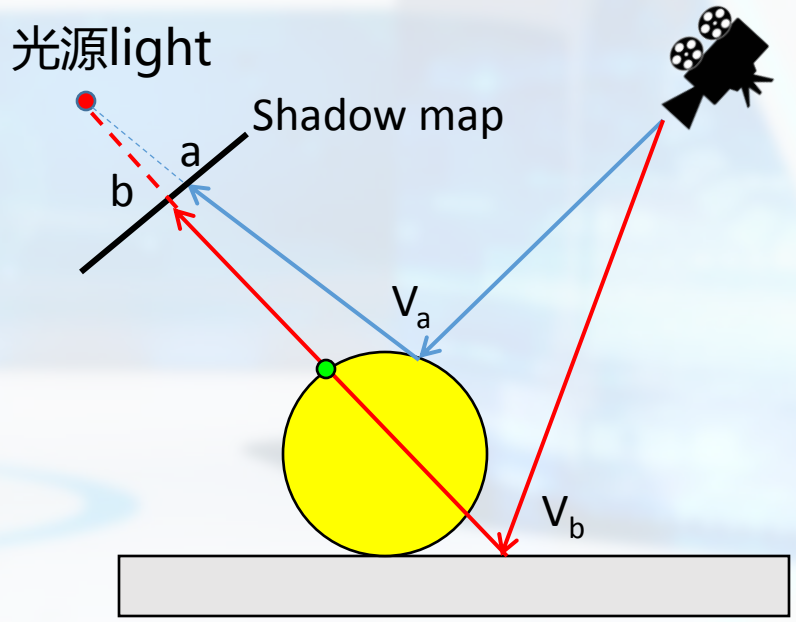


# 2 阴影的计算算法

## ◆主流算法

### ➢Shadow Mapping：

Step 2：将视点恢复到原来的正常位置，渲染整个场景，对每个象素计算它和光源的距离，然后将这个值和 depth map（shadow map）中相应的值比较，以确定这个象素点是否处在阴影当中。然后根据比较的结果，对 shadowed fragment（有阴影的片元）和 lighted fragment（有光照的片元）分别进行不同的光照计算，这样就可以得到阴影的效果了。

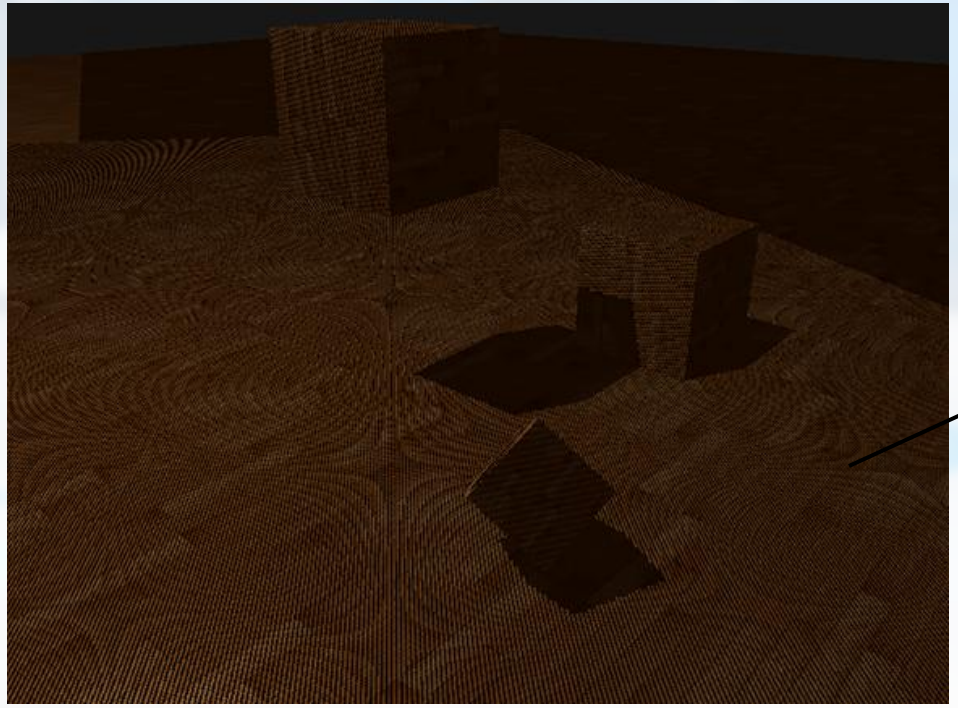


# 2 阴影的计算算法

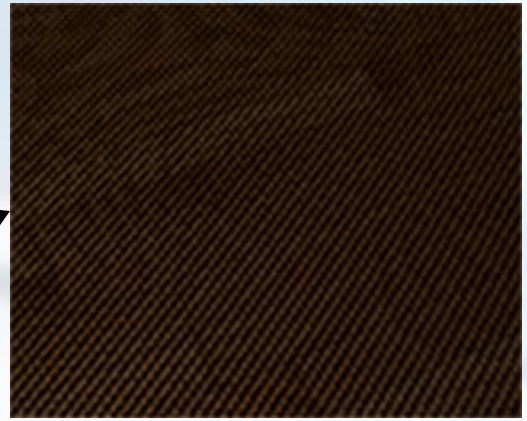
## ◆主流算法

### ➢Shadow Mapping：

存在的问题：阴影失真



放大

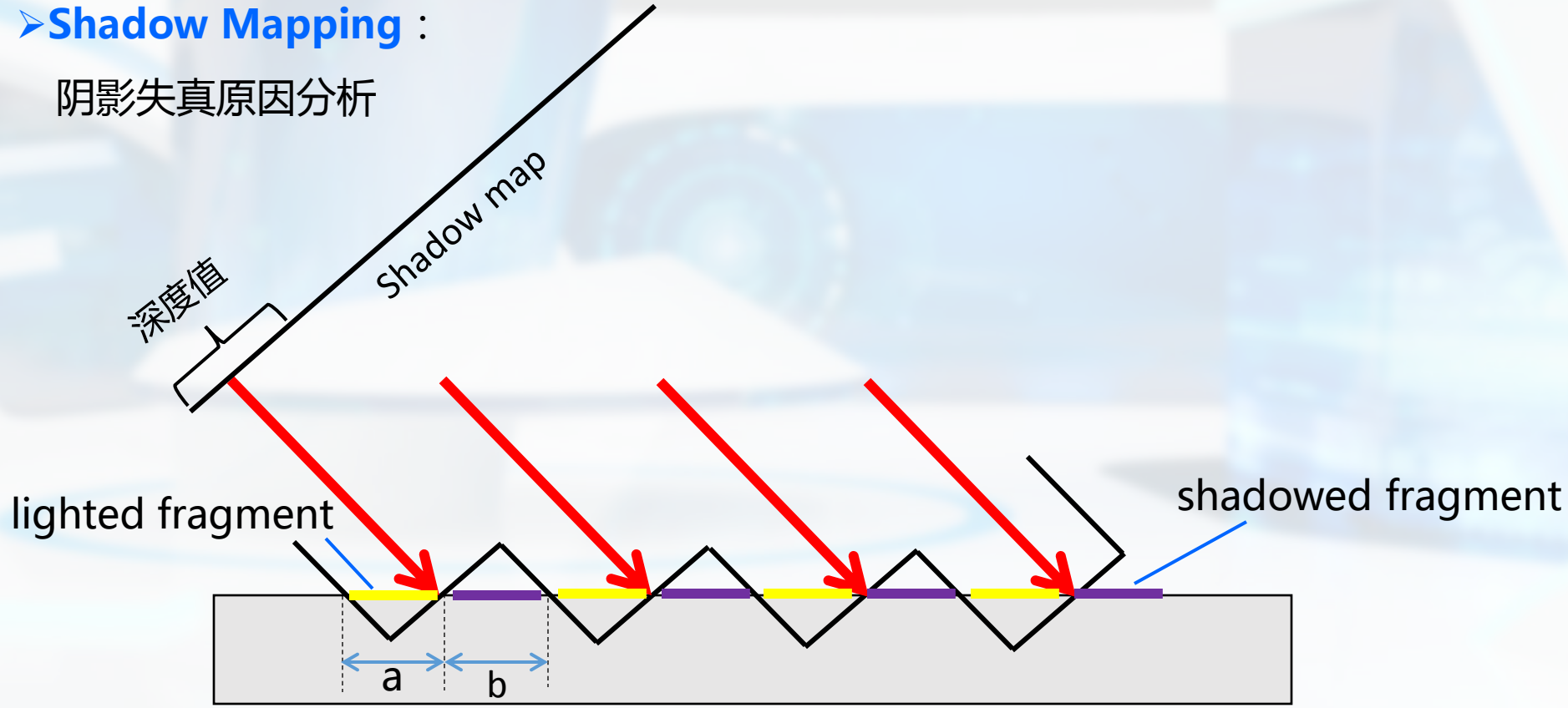


# 2 阴影的计算算法

## ◆主流算法

### ➢Shadow Mapping：

阴影失真原因分析



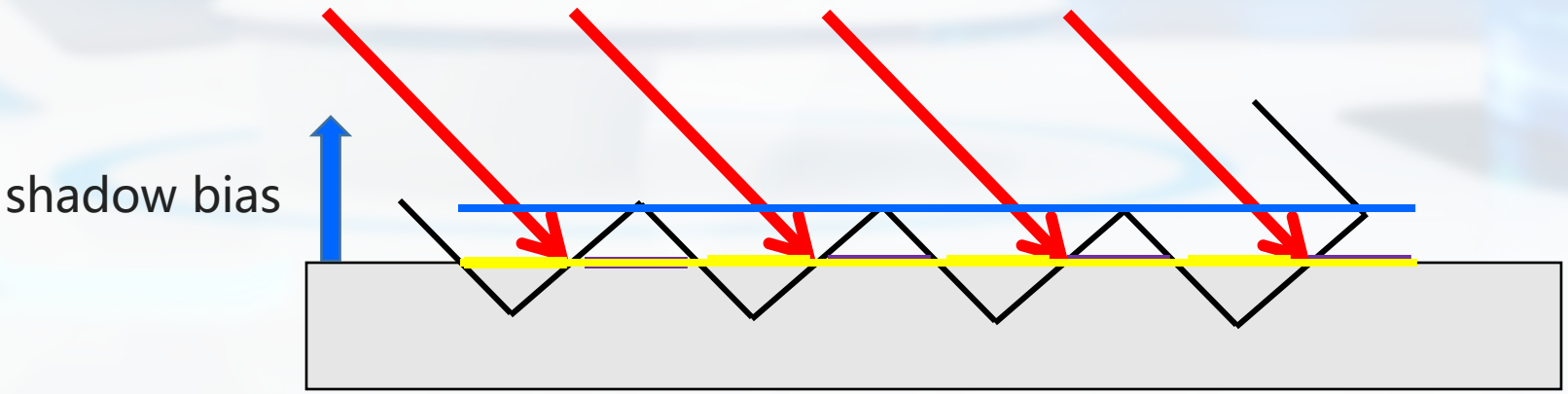
# 2 阴影的计算算法

## ◆主流算法

### ➢Shadow Mapping：

阴影失真的修正：阴影偏移

我们可以用一个叫做**阴影偏移**（shadow bias）的技巧来解决这个问题，我们简单的对表面的应用一个偏移量bias ，这样片元就不会被错误地认为在表面之下了。



# 2 阴影的计算算法

## ◆主流算法

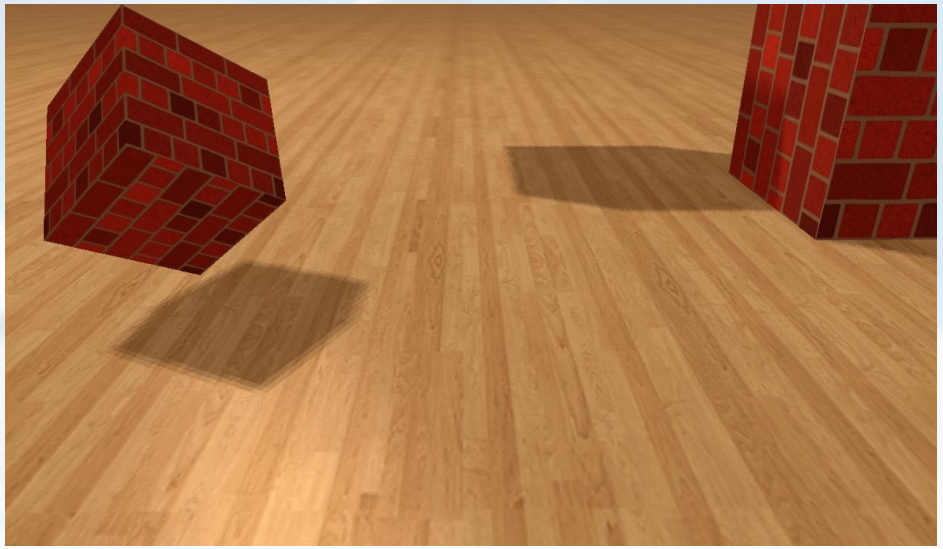
### ➢Shadow Mapping：

修正后的结果

# 2 阴影的计算算法

◆实验

要求：实现基于Shadow Mapping一个实时动态的阴影效果




谢谢
软件学院　万琳