7 消隐

- 基本概念
- 消隐算法

7.1 基本概念

- 物体的消隐或隐藏线面的消除：在给定视点和视线方向后，决定场景中哪些物体的表面是可见的，哪些是被遮挡不可见的。

基本概念

- 消隐算法按实现方式分类
  - 图像空间消隐算法：以屏幕像素为采样单位，确定投影于每一像素的可见景物表面区域，并将其颜色作为该像素的显示颜色。
    - 深度缓冲器算法、A缓冲器算法、区间扫描线算法等。

基本概念

- 景物空间消隐算法：直接在景物空间（观察坐标系）中确定视点不可见的表面区域，并将它们表达成同原表面一致的数据结构。如**BSP**算法、多边形区域排序算法等。
- 介于二者之间的算法：如深度排序算法、区域细分算法、光线投射算法等。

基本概念

- 基本的原则
  - 排序：各景物表面按照距离视点远近排序的结果，用于确定消隐对象之间的遮挡关系。
  - 连贯性：连贯性是指所考察的物体或视区内的图像局部保持不变的一种性质，用于提高排序效率。

7.2 消隐算法

- 深度缓存器算法
- 区间扫描线算法
- 深度排序算法
- 区域细分算法
- 光线投射算法
- **BSP**树算法
- 多边形区域排序算法

7.2.1 深度缓存器算法（Z-buffer）

- 基本原理
  - 帧缓存：保存各点的颜色。
  - Z缓存：保存屏幕坐标系上各象素点所对应的深度值。

深度缓存器算法（Z-buffer）

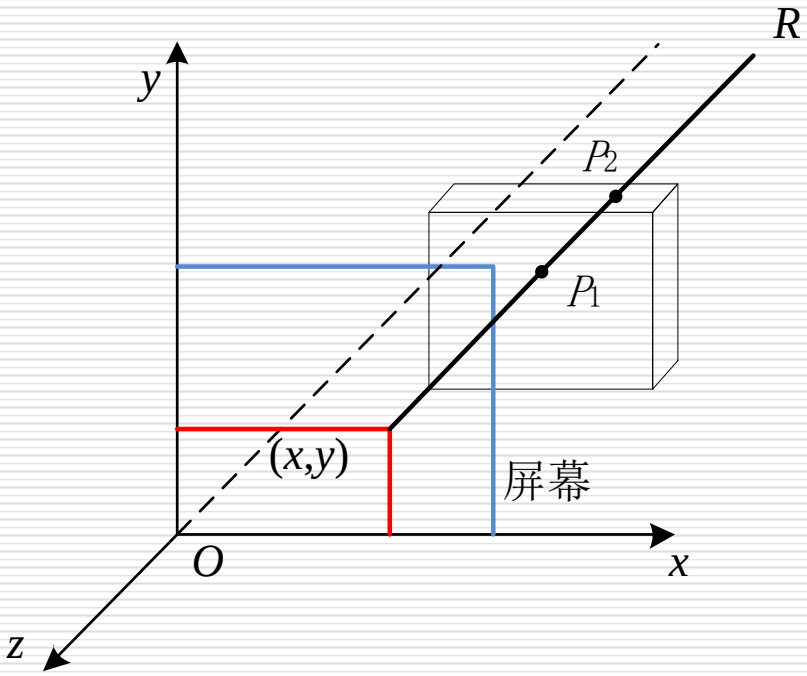


图7.1 深度缓存器算法的原理

深度缓存器算法（Z-buffer）

- 算法步骤
  - 初始化：把Z缓存中各(x,y)单元置为z的最小值，而帧缓存各(x,y)单元置为背景色。
  - 在把物体表面相应的多边形扫描转换成帧缓存中的信息时，对于多边形内的每一采样点(x,y)进行处理：

深度缓存器算法（Z-buffer）

- 计算采样点(x,y)的深度z(x,y)；
- 如z(x,y)大于Z缓存中在(x,y)处的值，则把z(x,y)存入Z缓存中的(x,y)处，再把多边形在z(x,y)处的颜色值存入帧缓存的(x,y)地址中。

深度缓存器算法（Z-buffer）

- 如何计算采样点(x,y)的深度z(x,y)。
  - 假定多边形的平面方程为：

    Ax+By+Cz+D=0

$$z(x,y)=\frac{-Ax-By-D}{C}$$

深度缓存器算法（Z-buffer）

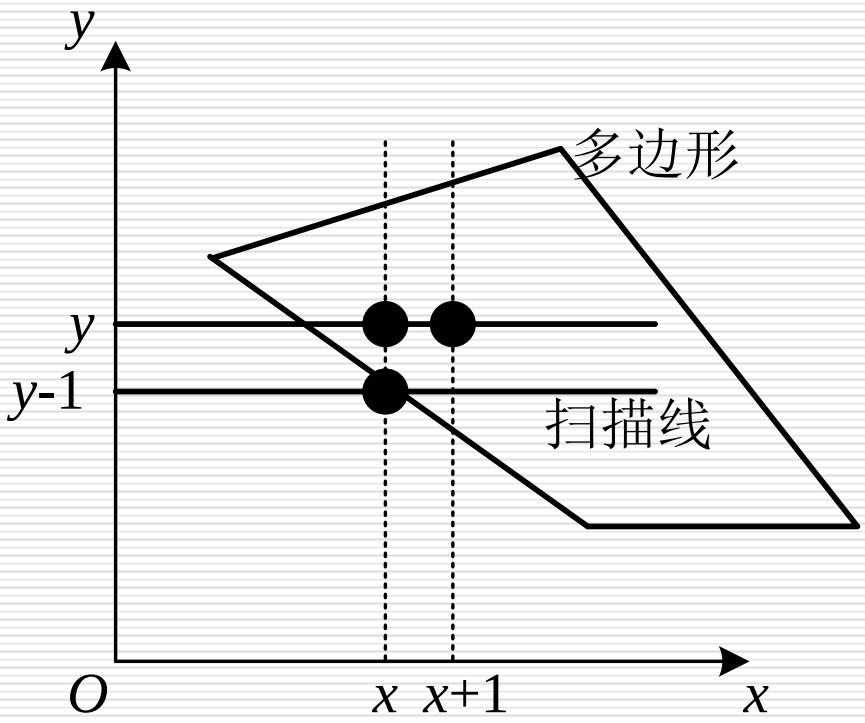


图7.2 利用扫描线的连贯性加速深度的计算

深度缓存器算法（Z-buffer）

- 扫描线上所有后继点的深度值：

$$z(x+1,y)=\frac{-A(x+1)-By-D}{C}=z(x,y)-\frac{A}{C}$$

- 当处理下一条扫描线y=y-1时，该扫描线上与多边形相交的最左边（x最小）交点的x值可以利用上一条扫描线上的最左边的x值计算：$x\big|_{y-1,\min}=x\big|_{y,\min}-\frac{1}{k}$

深度缓存器算法（Z-buffer）

$$z(x|_{y-1,\min}, y-1) = \frac{-Ax|_{y-1,\min} - B(y-1) - D}{C}$$

$$= \frac{-A(x|_{y,\min} - \frac{1}{k}) - B(y-1) - D}{C}$$

$$= z(x|_{y,\min}, y) + \frac{\frac{A}{k} + B}{C}$$

- 扫描线深度缓存器算法

深度缓存器算法（Z-buffer）

- 优点
  - 简单
  - 便于硬件实现
- 缺点
  - 占用太多的存储单元
  - 在实现反走样、透明和半透明等效果方面有困难

7.2.2 区间扫描线算法

- 避免对被遮挡区域的采样是进一步提高扫描线算法计算效率的关键。

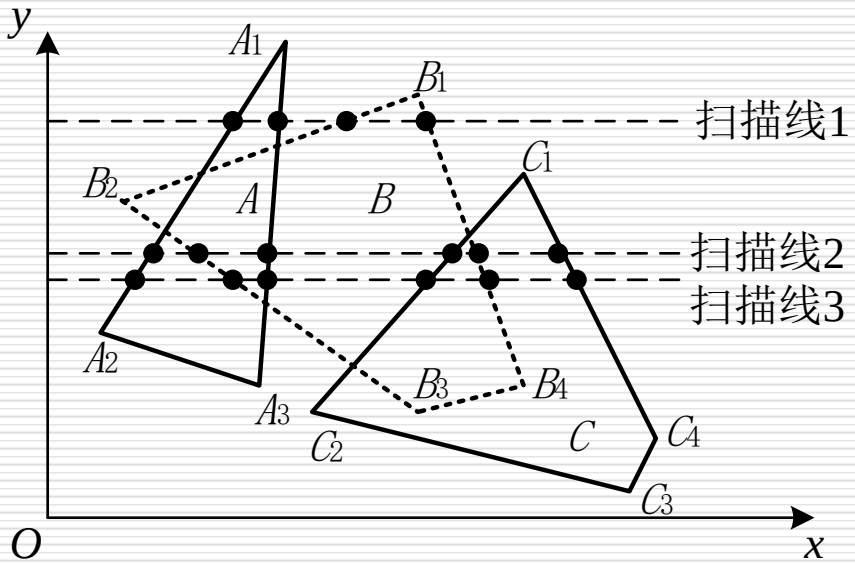


图7.3 区间扫描线算法原理

区间扫描线算法

- 算法
  - 边表

    场景中所有线段的端点坐标、斜率、指向多边形表的指针。
  - 多边形表

    多边形的平面方程系统、颜色值、指向边表的指针。

区间扫描线算法

- 有效边表

与当前扫描线相交的边的信息。

- 分割子区间，确定子区间上的唯一可见面。

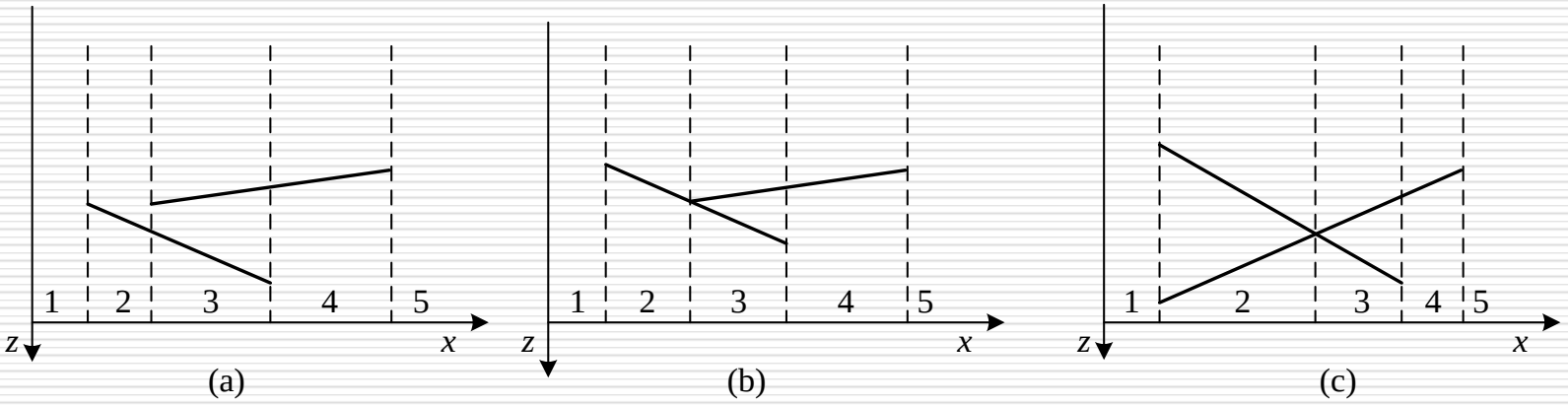
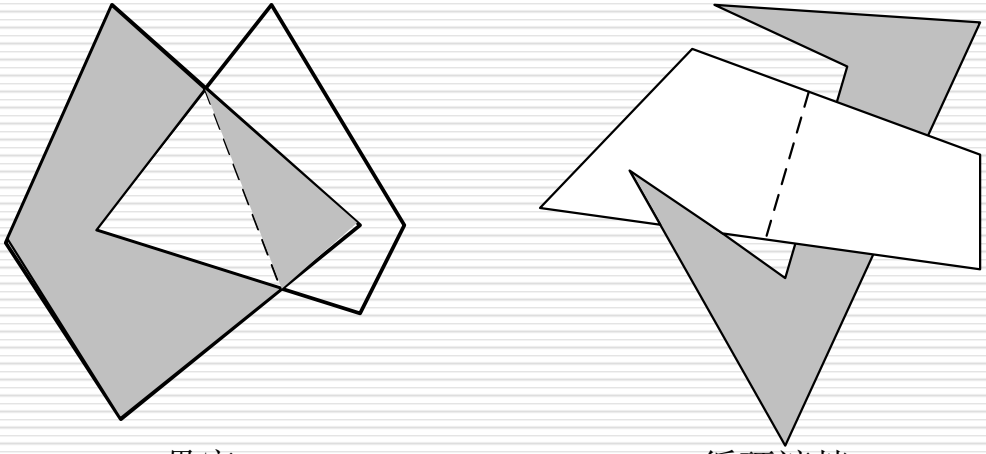


图7.4 扫描线子区间

区间扫描线算法

- 特殊情形



(a)贯穿　　(b)循环遮挡

图7.5 多边形贯穿和循环遮挡的情形

区间扫描线算法

- 贯穿情形：为了使算法能处理互相贯穿的多边形，扫描线上的分割点不仅应包含各多边形的边与扫描线的交点，而且应包含这些贯穿边界与扫描线的交点。
- 循环遮挡：将多边形进行划分以消除循环遮挡。

7.2.3 深度排序算法（画家算法）

- 介于图像空间消隐算法和景物空间消隐算法之间的一种算法。
- 算法原理：按照距视点的远近，由远及近生成图像。
- 类似画家创作油画的过程，故称画家算法。

深度排序算法（画家算法）

- 算法步骤
  - 将多边形按照深度进行排序，距视点近的优先级高，距视点远的优先级低。
  - 由优先级低的多边形开始，逐个对多边形进行扫描转换。

深度排序算法（画家算法）

- 算法的关键是多边形排序
  - **Zmin(B)>Zmax(A)**

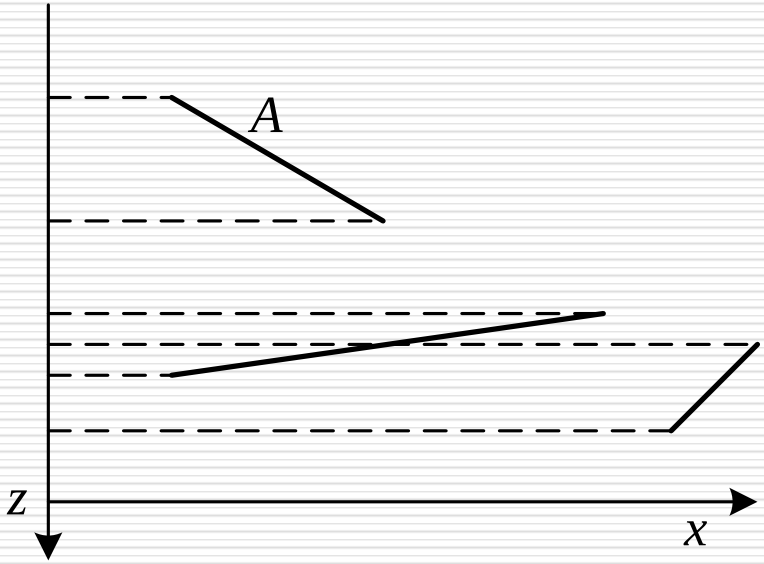


图7.6 A与其它多边形均无深度重叠

深度排序算法（画家算法）

- 包围盒判定A和B是否存在遮挡关系

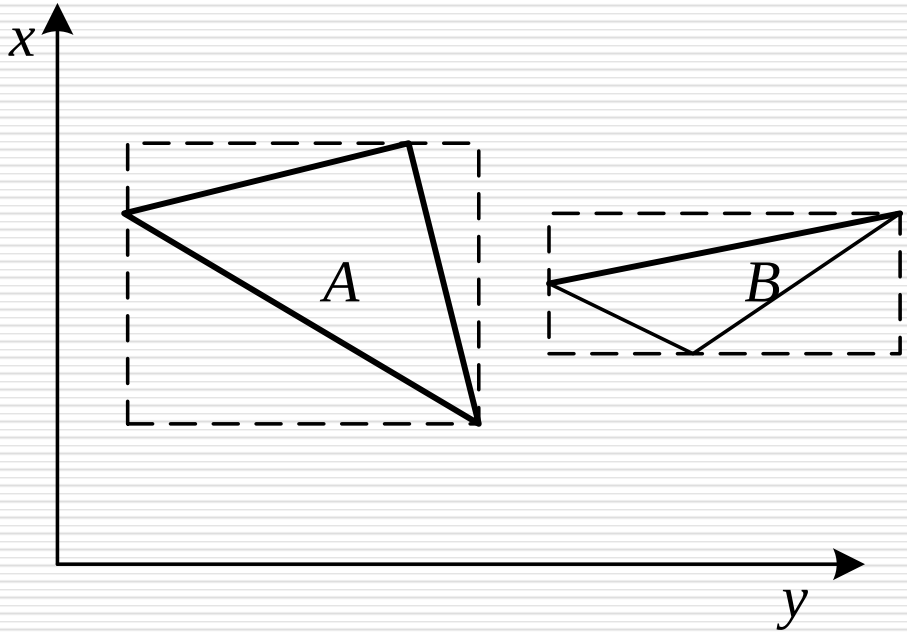


图7.7 A和B在xoy平面上投影的包围盒无重叠

深度排序算法（画家算法）

- 分别判定A和B上AB重叠区域的关系

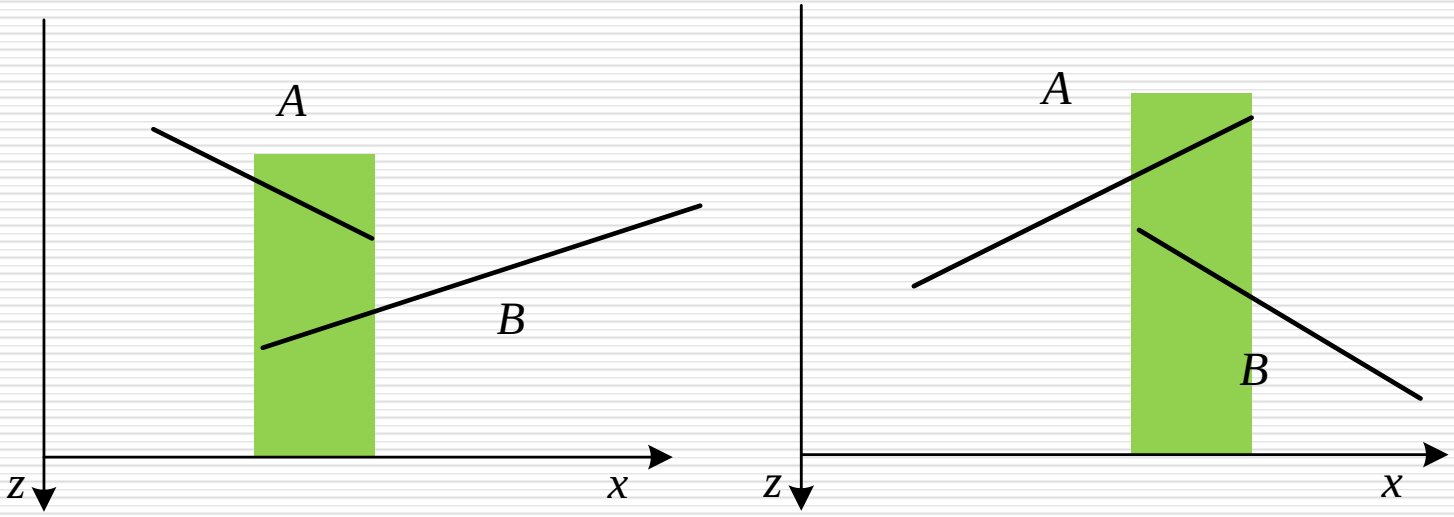


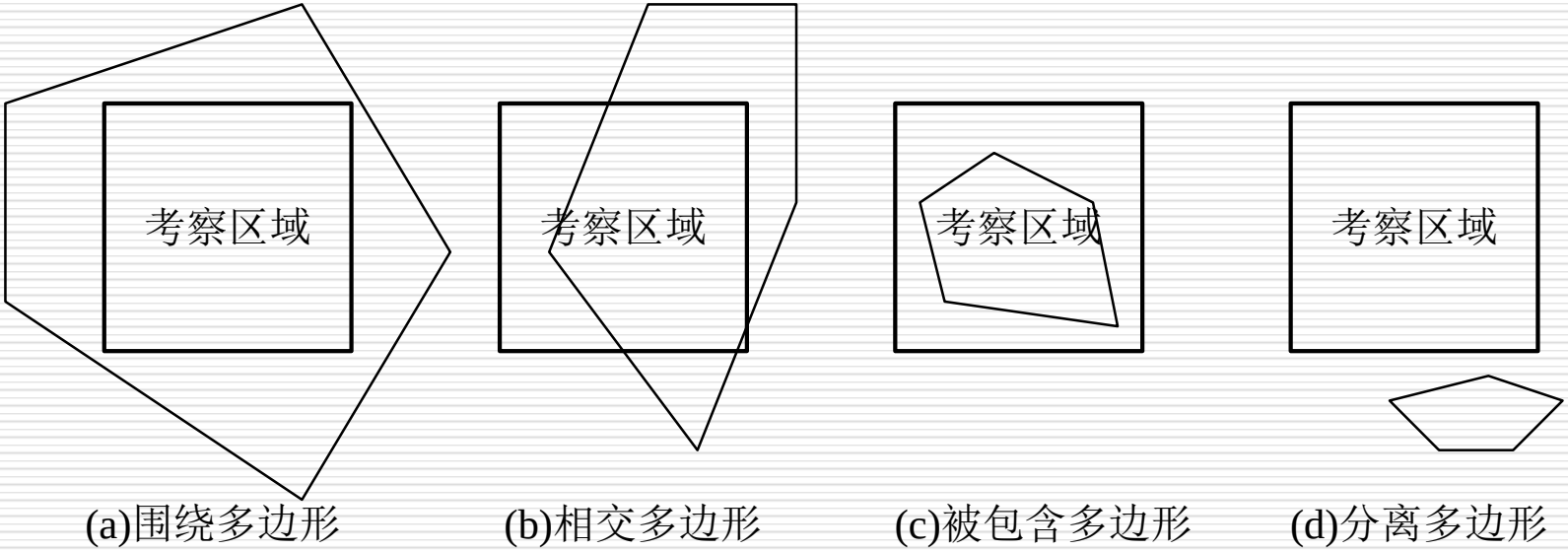
图7.8 A和B上AB重叠区域的关系

7.2.4 区域细分算法

- 图像空间消隐算法
- 投影平面上一块足够小的区域可以至多被一个多边形所覆盖
- 算法原理：考察投影平面上的一块区域，判断覆盖该区域中的可见多边形，否则就将这块区域细分为若干较小的区域。

区域细分算法

- 多边形的分类



(a)围绕多边形　(b)相交多边形　(c)被包含多边形　(d)分离多边形

图7.9 多边形的投影与考察区域之间的关系

区域细分算法

- 可见性测试
  - 所有多边形均是该区域的分离多边形。直接将该区域中的所有像素点置为背景颜色。
  - 仅存在一个相交多边形，或仅存在一个被包含多边形，或仅存在一个围绕多边形。先将所有像素置为背景颜色，再将相应多边形的颜色值填入对应像素点的帧缓存中。

区域细分算法

- 有多于一个的相交多边形、被包含多边形或围绕多边形。计算所有围绕的、相交的、以及被包含的多边形在该区域4个顶点处的z坐标，如果存在一个围绕多边性，它的4个z坐标比其它任何多边性的z坐标都大（最靠近视点），那么，可将该区域中的所有像素点置为该多边形的颜色值。

区域细分算法

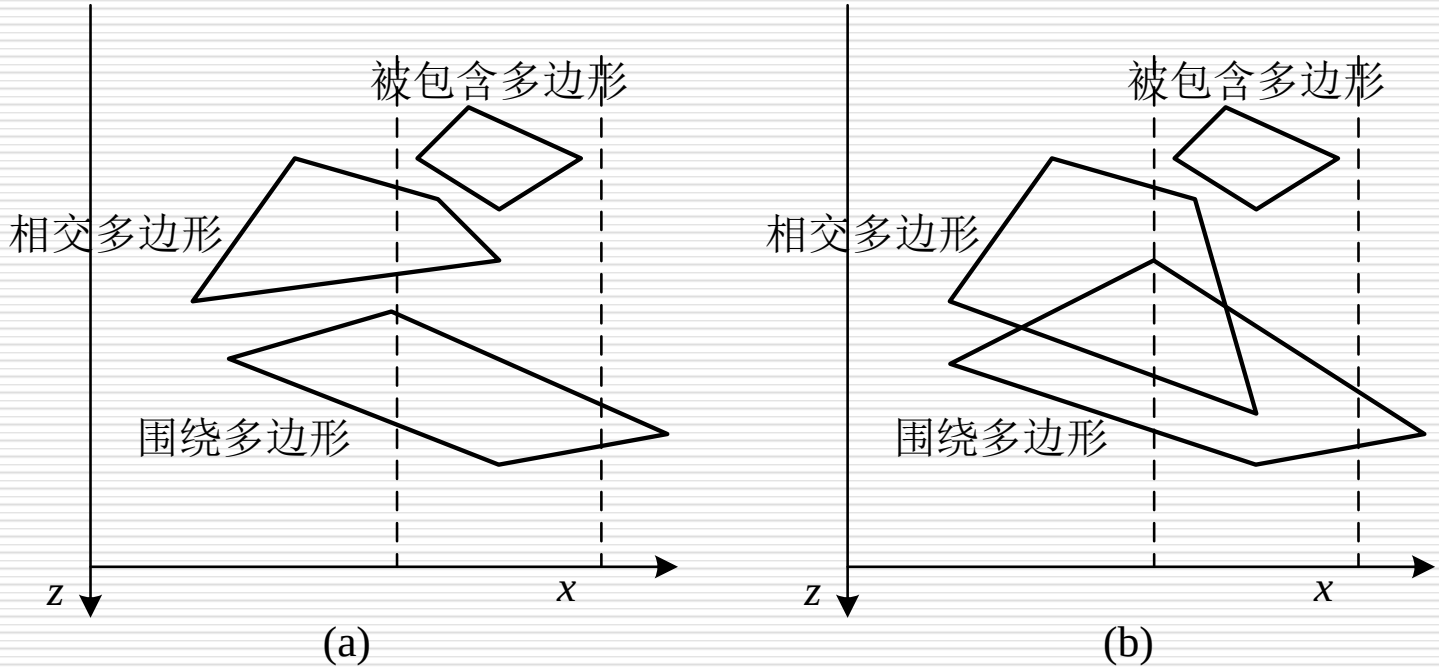


图7.10 两个例子

区域细分算法

- 区域细分方式
  - 将区域简单地分割为四块大小相等的矩形。
  - 自适应细分，即沿多边形的边界对区域进行细分。

7.2.5 光线投射算法

- 算法原理

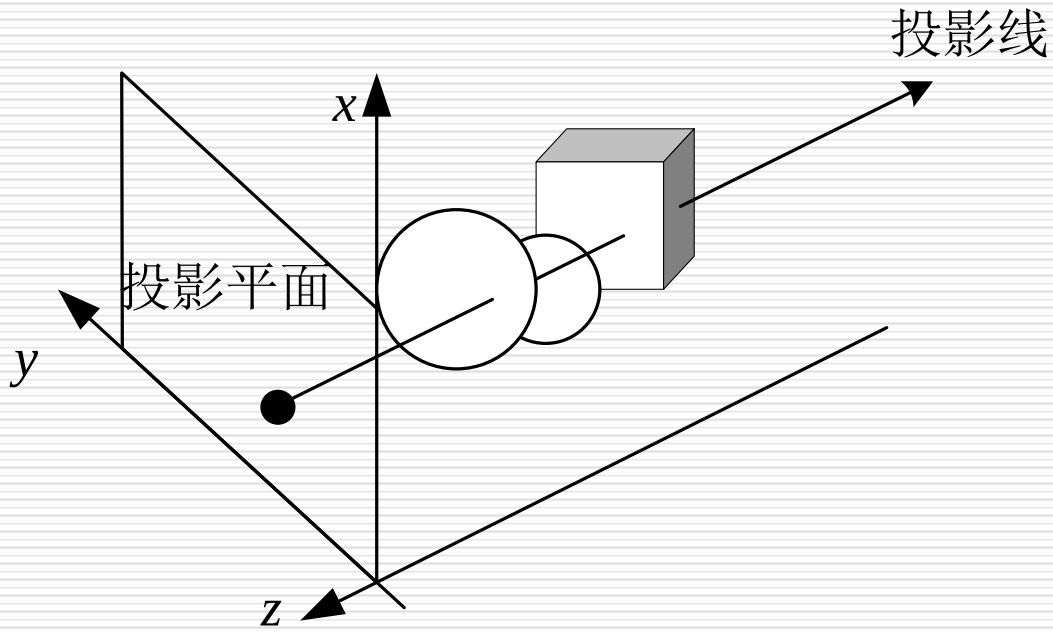


图7.11 光线投射算法

光线投射算法

- 算法步骤
  - 通过视点和投影平面（显示屏幕）上的所有像素点作一入射线，形成投影线。
  - 将任一投影线与场景中的所有多边形求交。
  - 若有交点，则将所有交点按z值的大小进行排序，取出最近交点所属多边形的颜色；若没有交点，则取出背景的颜色。
  - 将该射线穿过的像素点置为取出的颜色。

光线投射算法

- 特点
  - 不需要帧缓存；
  - 需要进行求交运算；
  - 利用连贯性，外接矩形和空间分割可以进行交点计算的加速计算；
  - 对于包含曲面，尤其是球面的场景有很高的计算效率。

7.2.6 BSP树算法

- 适用于视点变换频繁的静态场景绘制。
- 二叉空间剖分（**Binary space partitioning**）
- 原理：组成场景的多边形可以被一个多边形分割成两个部分，视点位于分割平面正侧，则正侧多边形会遮挡另一侧多边形。

BSP树算法

- 分割过程
  - 用场景中的一个多边形作为分割多边形将场景中所有多边形进行分割，分别放入两个子空间中；
  - 与分割多边形相交的多边形被分成两个多边形；
  - 不断细分，直到所有的子空间中只有一个多边形为止。

BSP树算法

- BSP树的构造

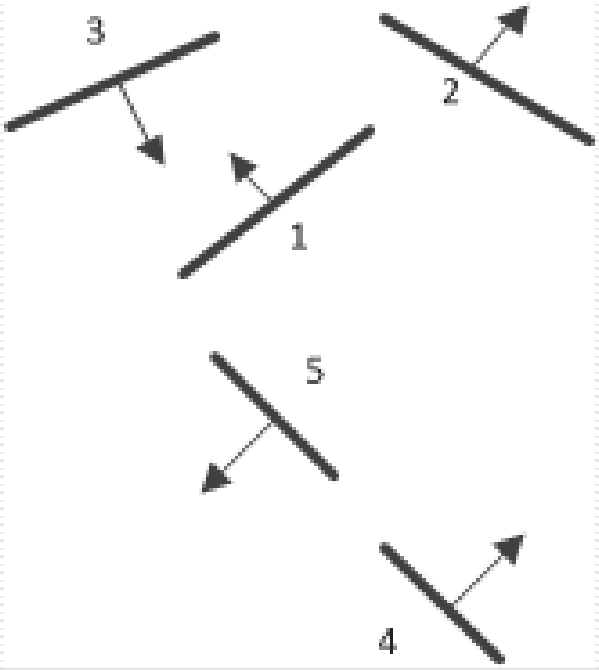


图7.12 BSP树构造

BSP树算法

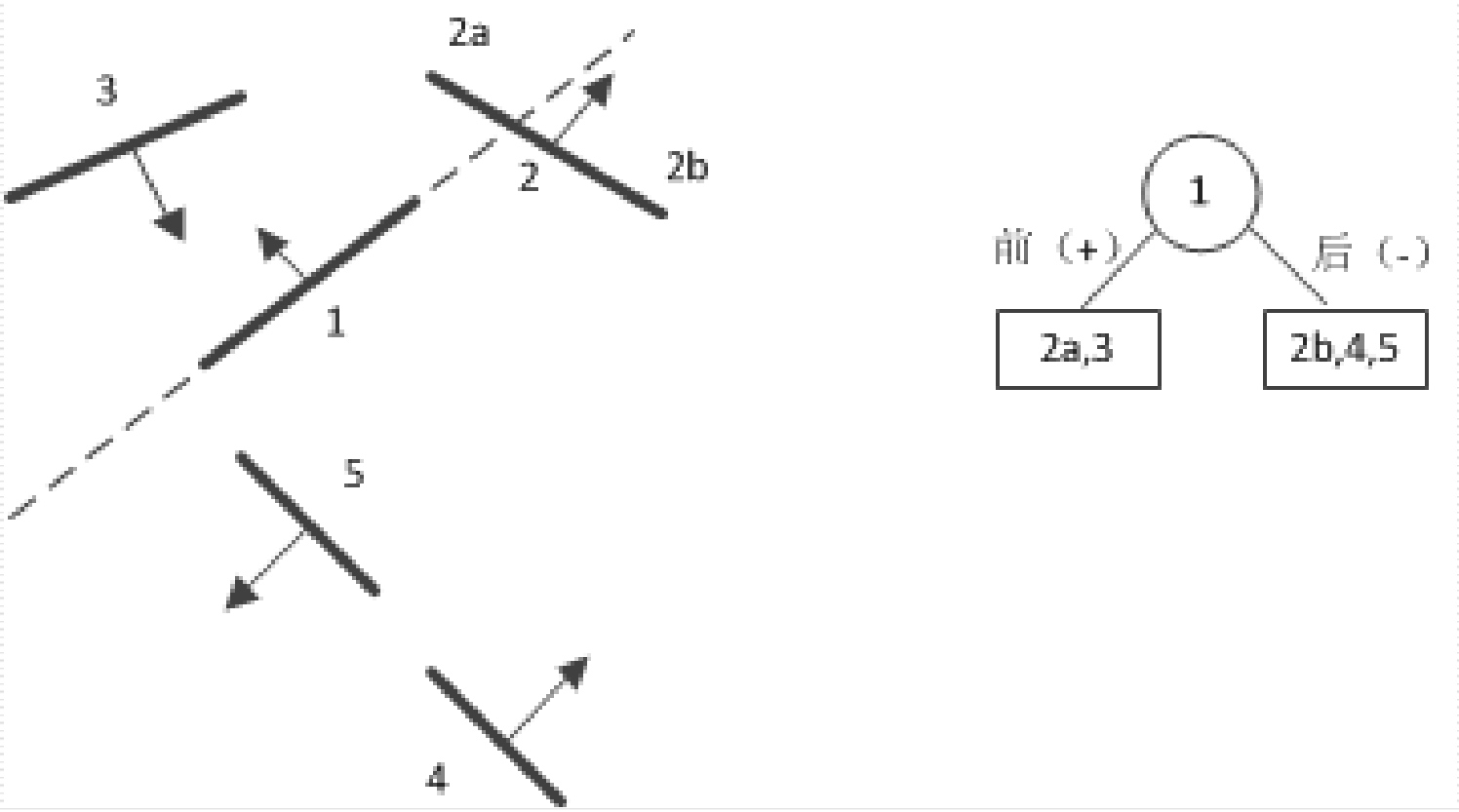


图7.12 BSP树构造

BSP树算法

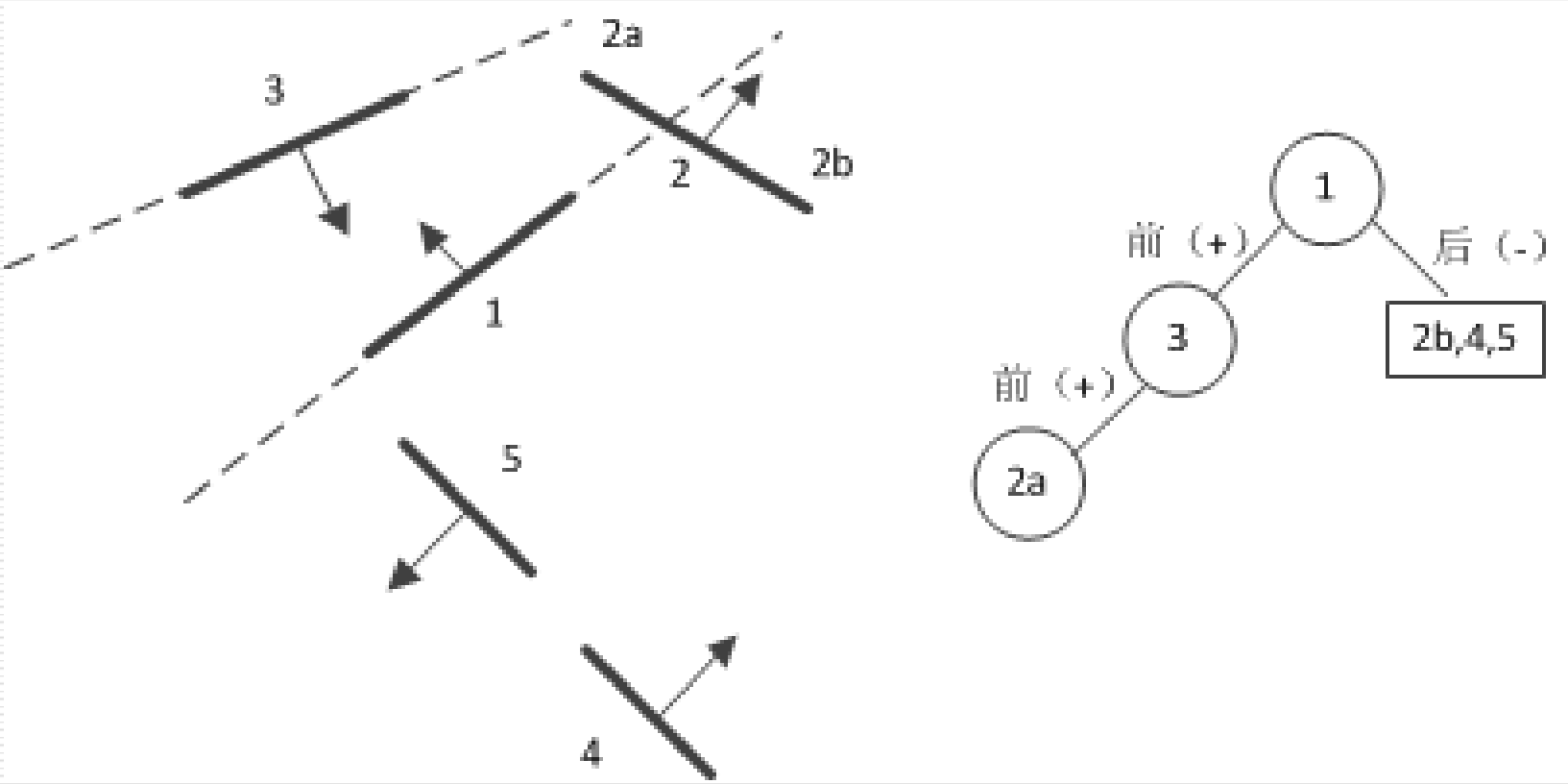


图7.12 BSP树构造

BSP树算法

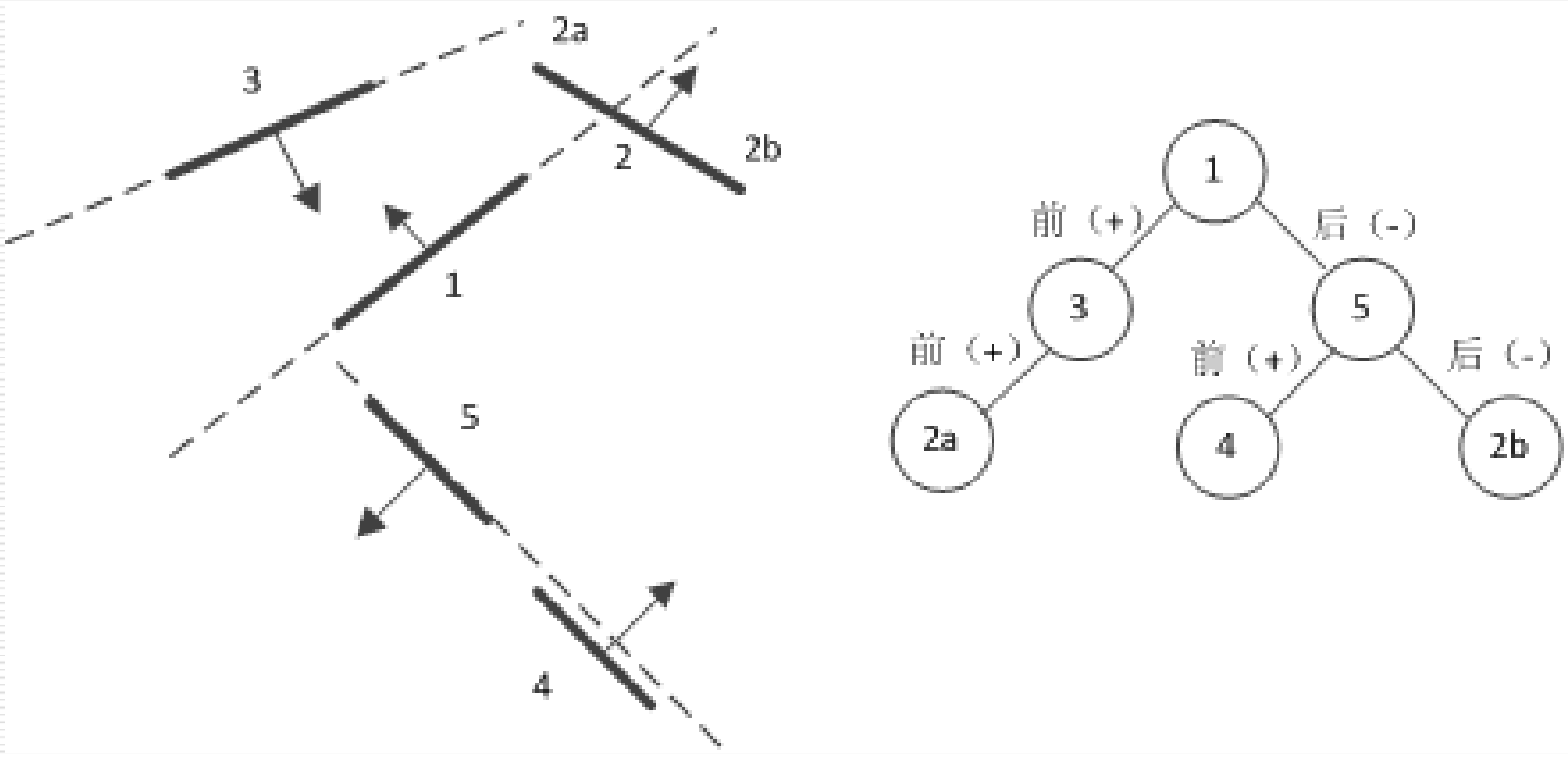


图7.12 BSP树构造

BSP树算法

- BSP树的遍历
  - BSP树与视点无关
  - 根据视点与分割多边形的相对位置进行遍历
  - 视点在根节点多边形的前面

    前侧分支—根节点—后侧分支
  - 视点在根节点多边形的前面

    后侧分支—根节点—前侧分支

7.2.7 多边形区域排序算法

- 算法原理

将多边形按深度值由小到大排序，用前面的可见多边形去切割位于其后的多边形，使得最终每一个多边形要么是完全可见的，要么是完全不可见的。

多边形区域排序算法

- 算法过程
  - 多边形按照深度进行预排序；
  - 用距离视点最近的多边形对所有的多边形进行裁剪和区域分类；
  - 删去除距视点最近多边形之后的其它多边形；
  - 递归处理，直到消除所有的不确定性。

多边形区域排序算法

- 特点
  - 适用于任意多边形构成场景的消隐处理。
  - 可以正确的处理循环遮挡的情况