消隐算法一景物空间消隐

冯结青

浙江大学 CAD&CG国家重点实验室

主要内容

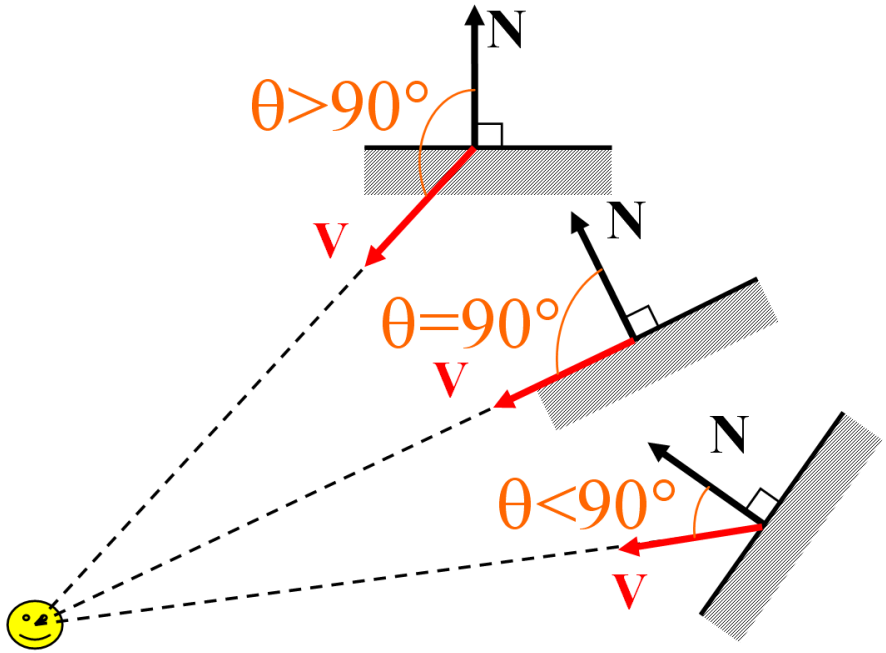
- 背面剔除算法
- 表优先级算法
- 几种典型消隐算法的比较数据
- 关于秋学期的上机作业

主要内容

- 背面剔除算法
- 表优先级算法
- 几种典型消隐算法的比较数据
- 关于秋学期的上机作业

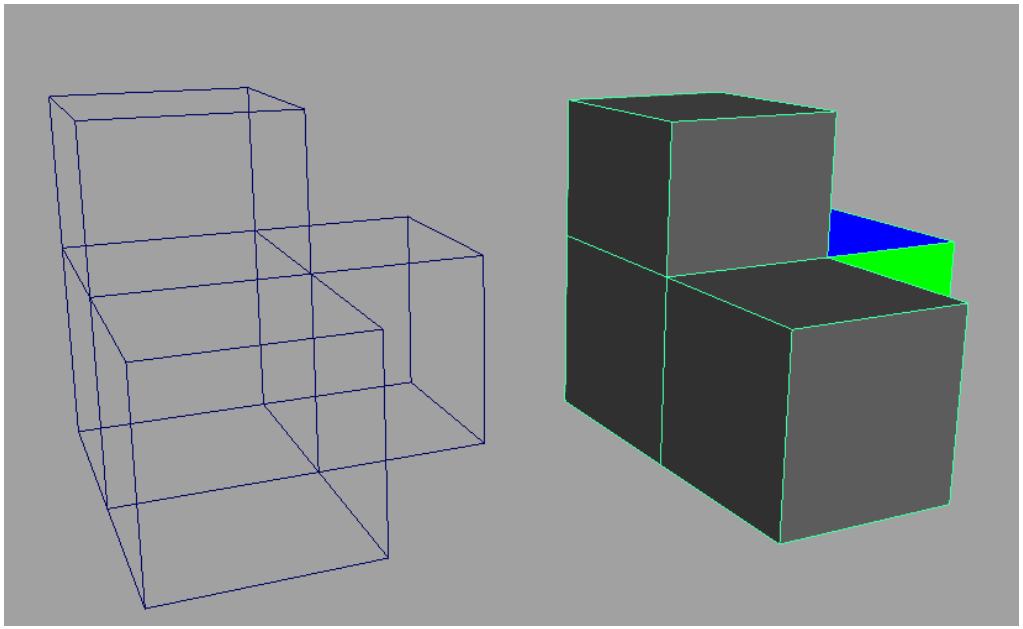
背面剔除算法

- 利用视线方向**V**和物体表面法向**N**之间的关系
  - **N·V**<0：不可见
  - **N·V**≥0：可见



背面剔除算法

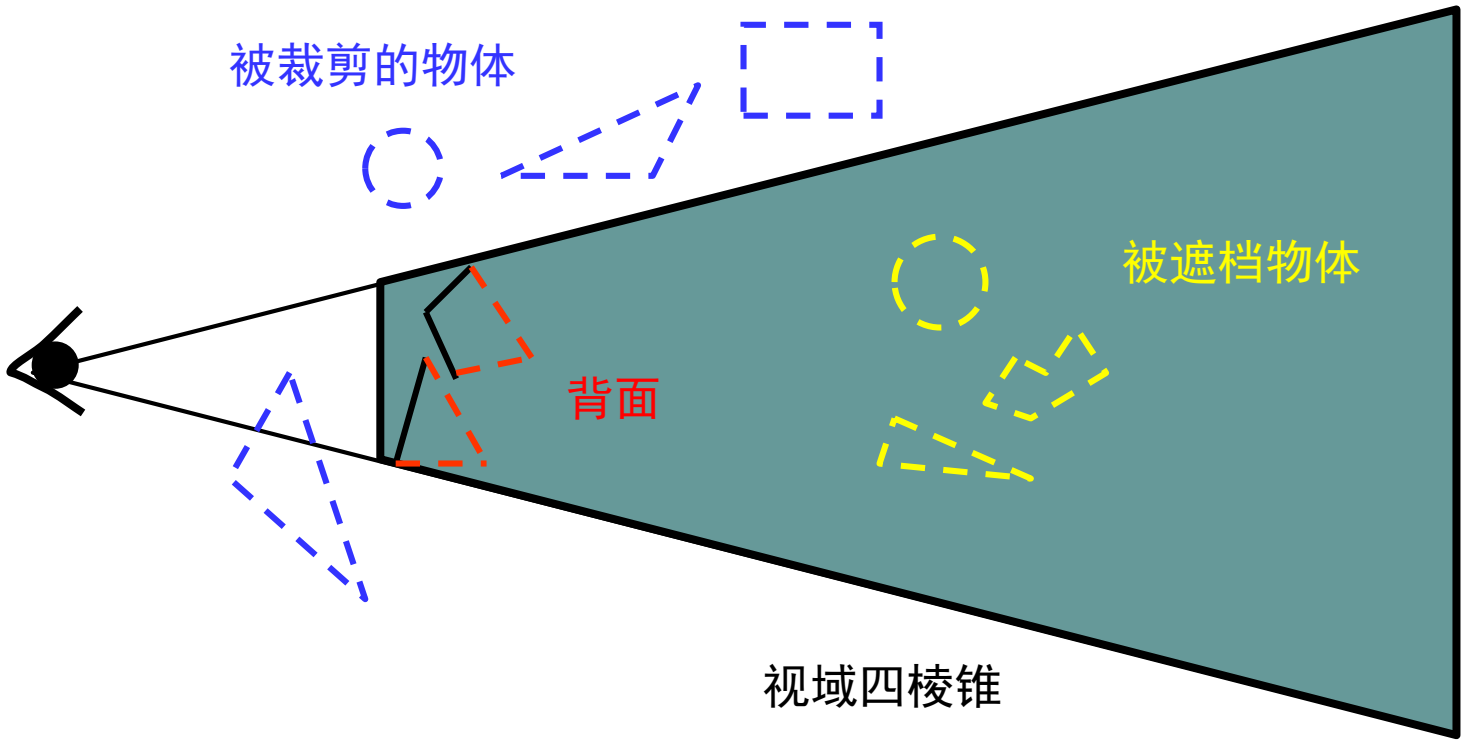
- 作为消隐算法，背面剔除适用于凸多面体，不适用于非凸多面体或其它复杂物体



对于蓝色与绿色的面，简单的背面剔除不能不能实现完全消隐

背面剔除算法

- 适用于场景消隐的预处理：消除一些显然不可见表面，从而提高其它消隐算法的效率

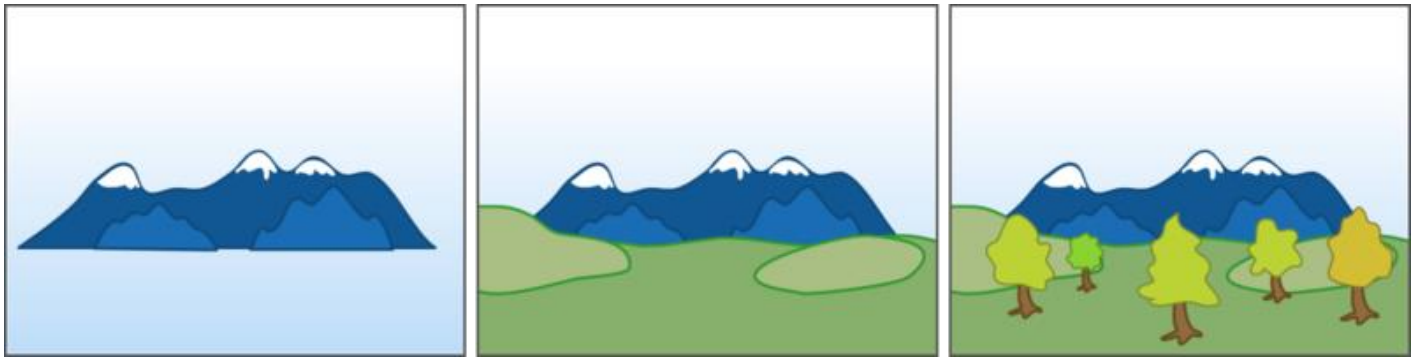


主要内容

- 背面剔除算法
- 表优先级算法
  - 三维物体的深度排序算法
  - 二叉空间剖分树算法
- 几种典型消隐算法的比较数据
- 关于秋学期的上机作业

景物空间中的表优先级算法

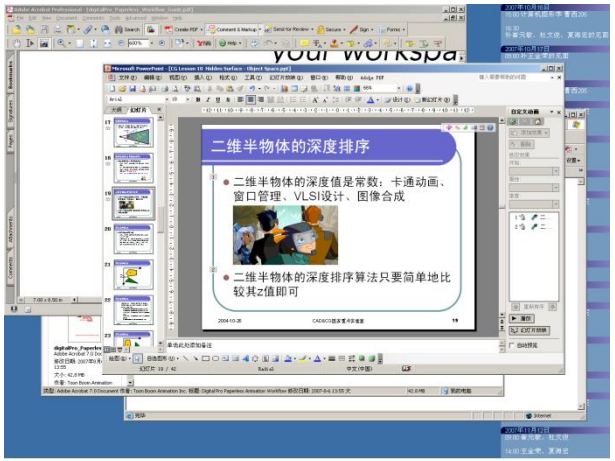
- 原理：离视点近的物体可能遮挡离视点远的物体
- 在景物空间确定物体之间的可见性顺序(物体离视点远近)，由远及近地绘制出正确的图像结果——油画家算法



- 条件：场景中物体在z方向上没有相互重叠

二维半物体的深度排序

- 二维半物体的深度值是常数：卡通动画、窗口管理、VLSI设计、图像合成



- 二维半物体的深度排序算法只要简单地比较其z值即可

主要内容

- 背面剔除算法
- 表优先级算法
  - 三维物体的深度排序算法(视点坐标系)
  - 二叉空间剖分树算法
- 几种典型消隐算法的比较数据
- 关于秋学期的上机作业

三维物体的深度排序算法

1. 将场景中的多边形序列按其z坐标的最小值z_{min} (物体上离视点最远的点)进行排序

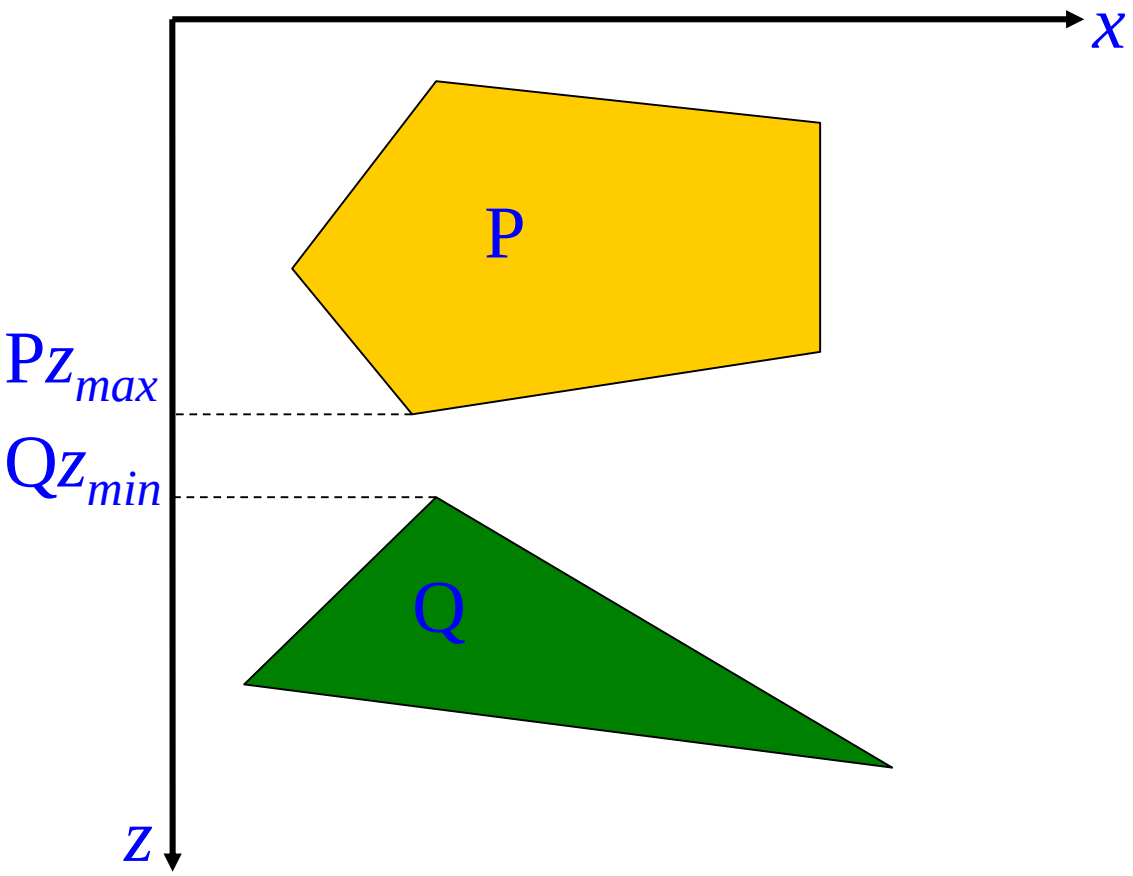
2. 当物体间的z值范围不重叠时

   如果多边形P的z值范围与Q的z值范围不重叠；假设多边形P的z_{min}在上述排序中最小，多边形P的z_{min}在上述排序中最小，即

$$Pz_{min} < Qz_{min}$$

   则多边形P的优先级最低 (见下图)

三维物体的深度排序算法

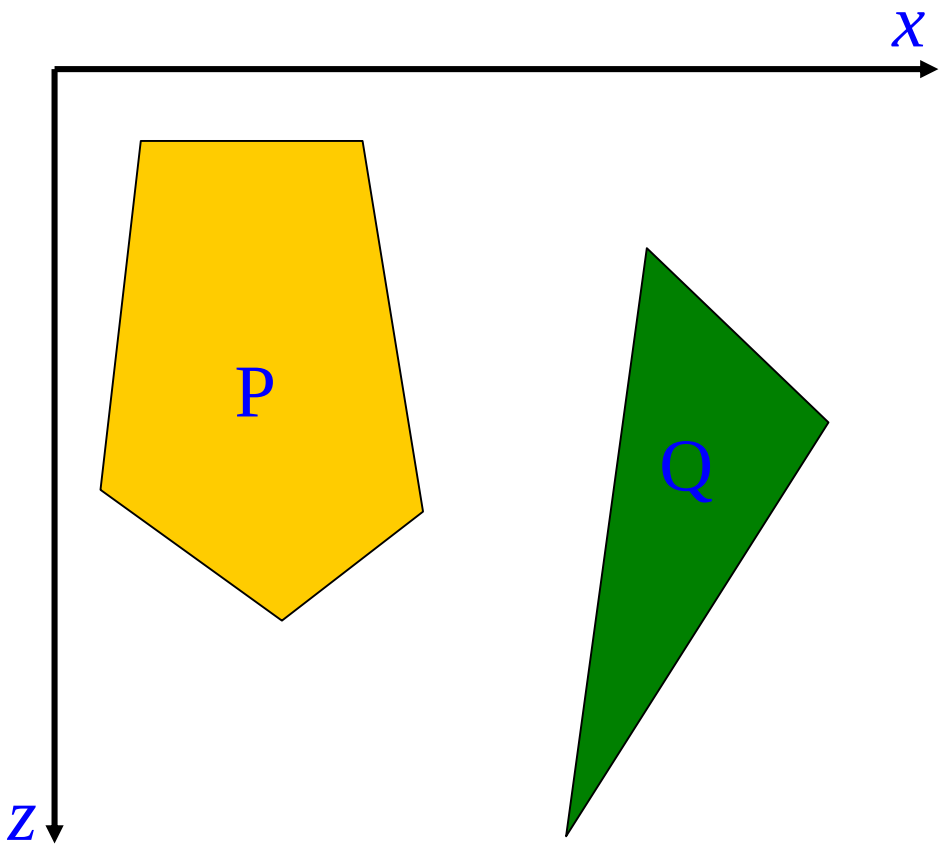


物体间的z值范围不重叠

三维物体的深度排序算法

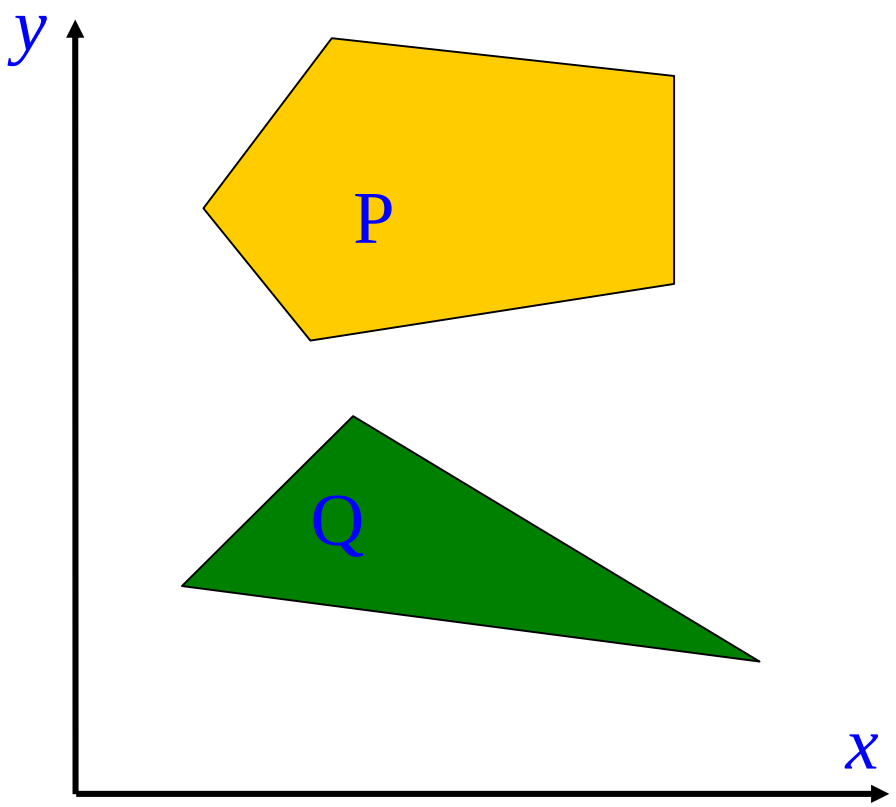
3. 当物体间的z值范围重叠时：判断多边形P是否遮挡场景中多边形Q，需作如下5个判别步骤
    1. 多边形P和Q的x坐标范围是否不重叠
    2. 多边形P和Q的y坐标范围是否不重叠
    3. 从视点看去，多边形P是否完全位于Q的背面
    4. 从视点看去，多边形Q是否完全位于P的同一侧
    5. 多边形P和Q在xy平面上的投影是否不重叠

三维物体的深度排序算法



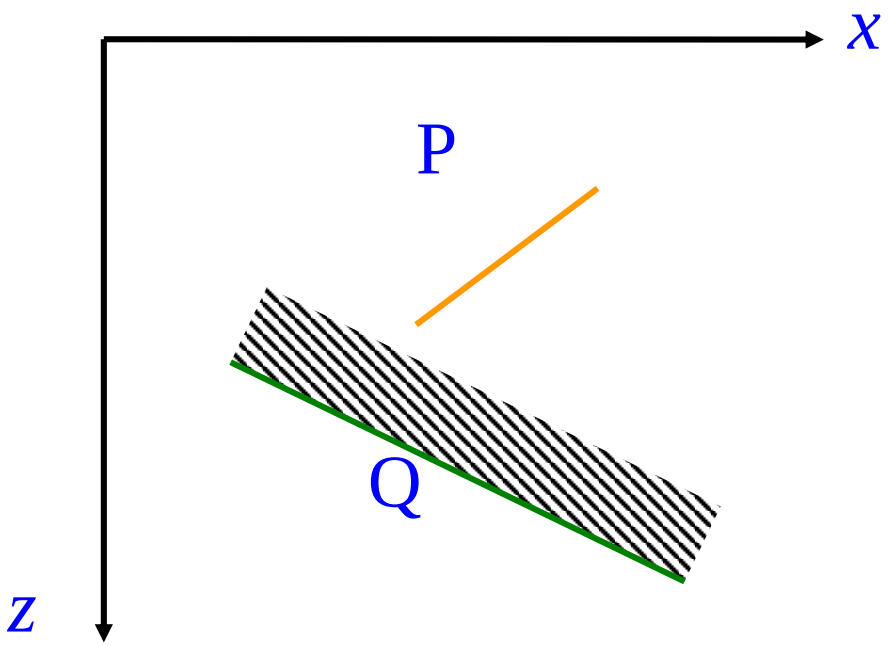
1. 多边形P和多边形Q的x坐标范围不重叠

三维物体的深度排序算法



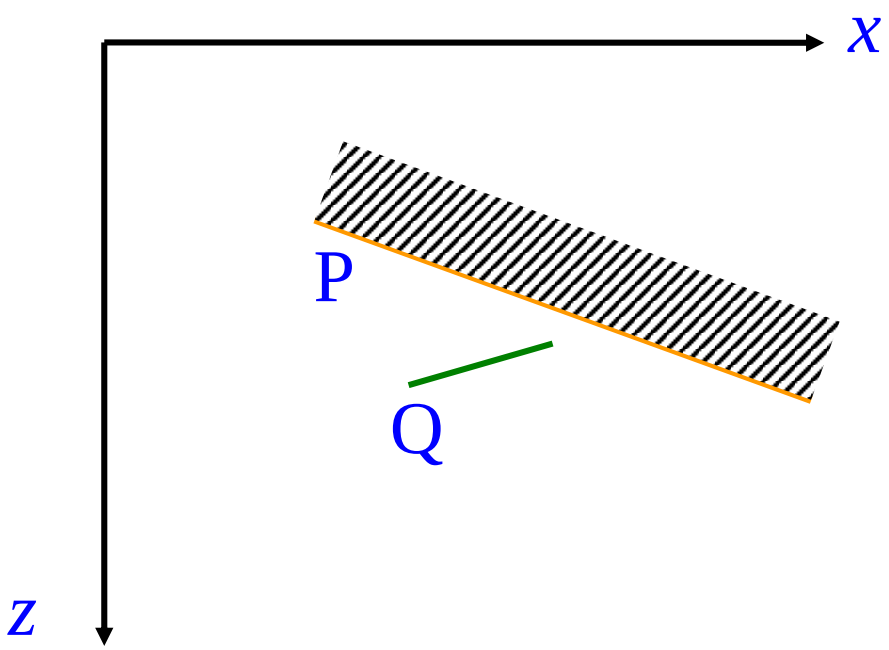
2. 多边形P和多边形Q的y坐标范围不重叠

三维物体的深度排序算法



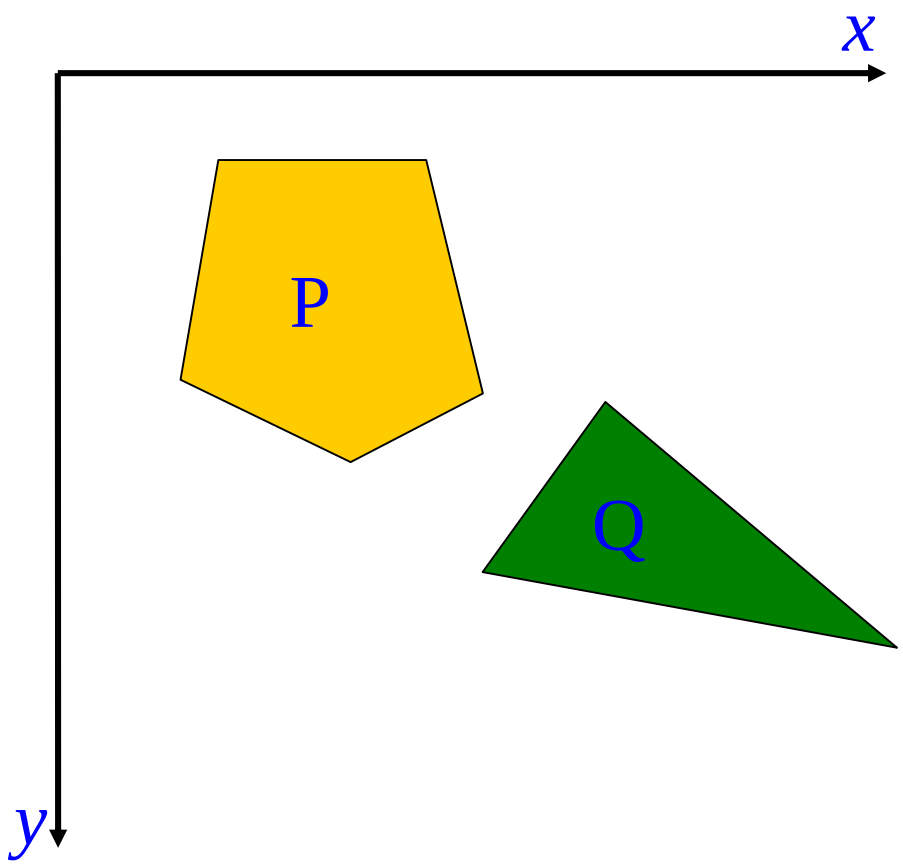
3.从视点看去，多边形P完全位于Q的背面

三维物体的深度排序算法



4. 从视点看去，多边形Q完全位于P的同一侧

三维物体的深度排序算法



5. 多边形P和Q在xy平面上的投影不重叠

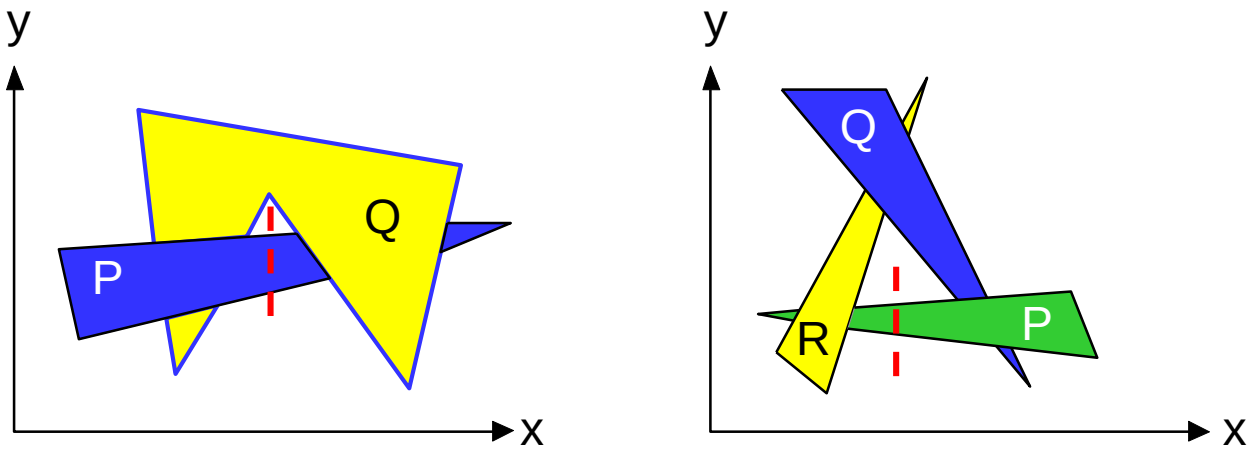
三维物体的深度排序算法

- 如果上述五种情况中只要有一种成立，则多边形P和Q是互不遮挡的，即多边形P的绘制优先级低于Q
- 如果上述判断都不成立，说明多边形P有可能遮挡Q，此时把多边形P和Q进行互换重新进行判断，而重新判断只要对上述条件(3)和(4)进行即可

三维物体的深度排序算法

4. P和Q交换顺序后，仍不能判断其优先级顺序，可以按如下方法处理：将其中一个多边形沿另一个物体剖分
   - 避免循环判断：P做标记
   - 多边形剖分：将P沿Q剖分

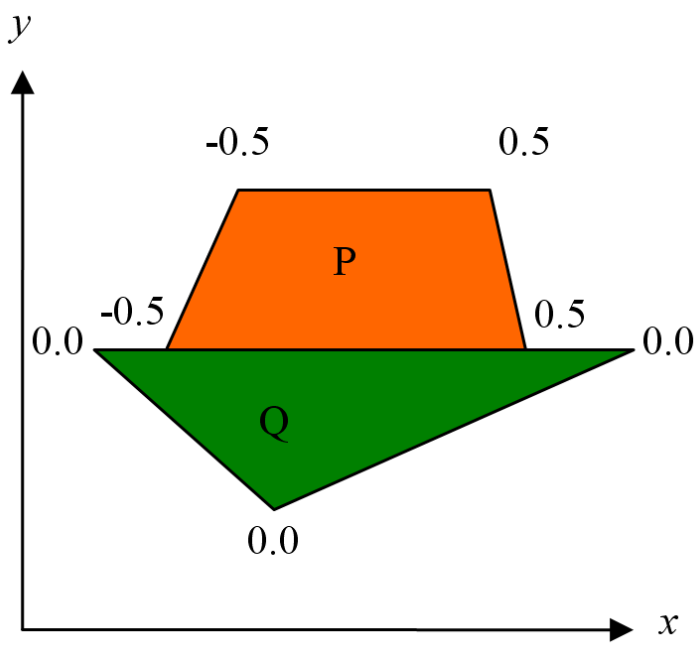
三维物体的深度排序算法

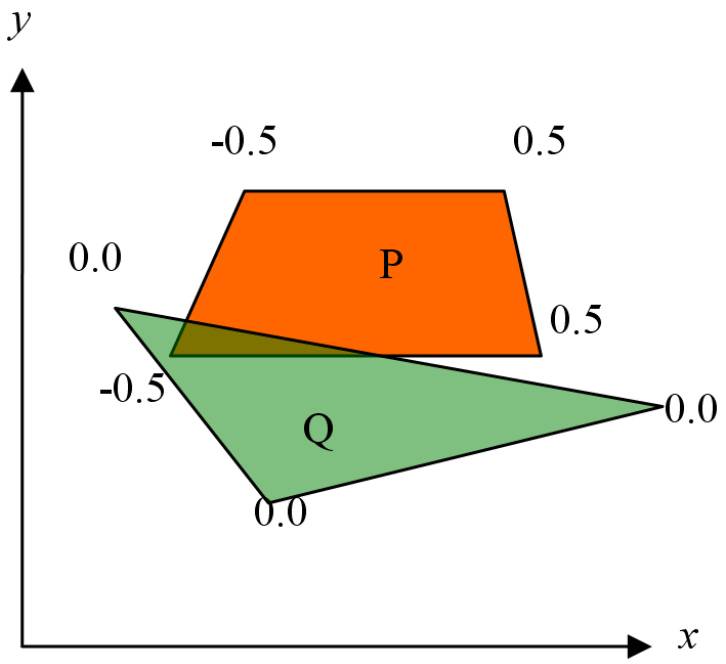


相互遮挡时，将其中一个多边形沿另一个多边形进行剖分

三维物体的深度排序算法

- 深度排序算法有时可能将具有正确深度顺序的多边形进行剖分

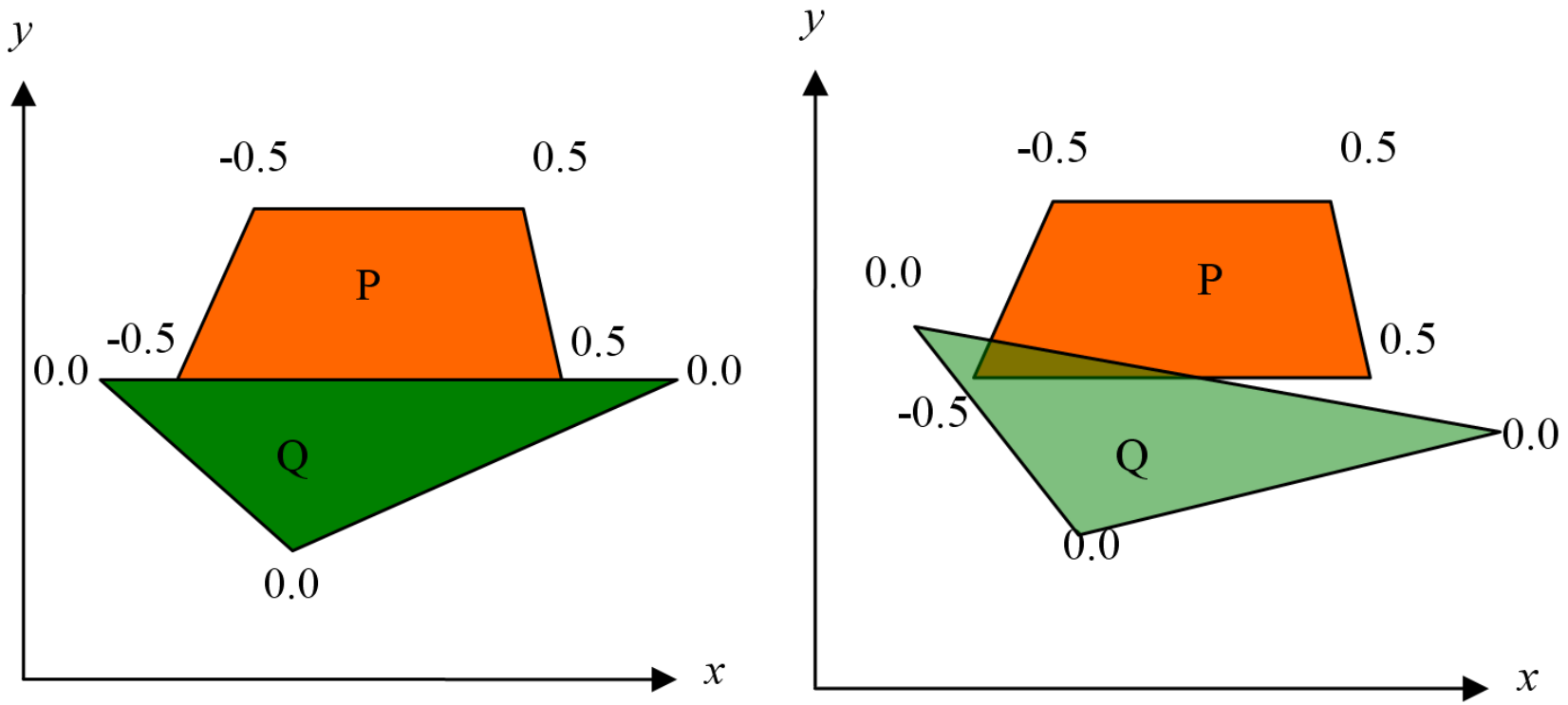




三维物体的深度排序算法

- 深度排序算法有时可能将具有正确深度顺序的多边形进行剖分
  - 根据上述规则，P的优先级低于Q
  - 把多边形Q沿z轴算转如图的角度后，P的优先级仍然低于Q
  - 当用上述五个判别条件对它们进行判断时都不成立，P需要相对于Q进行剖分
  - 多余的剖分会增加计算量！！！

三维物体的深度排序算法



具有正确深度顺序而被剖分的多边形：多边形顶点只标记了其深度值。左图可以直接采用油画家算法进行绘制；右图虽然具有正确的深度顺序，但是它们在进行深度比较的时，五个条件均不满足。根据上述排序算法，需要进行对P多边形剖分

深度排序算法

- 三维物体的深度排序算法适合于固定视点的消隐
  - 通过多边形的剖分，总是可以实现多边形物体在三维空间中的深度排序
  - 深度排序算法可以有效地实现透明效果
- 在视点变化的场合中(如飞行模拟)，深度排序算法难以满足实时性的要求
  - 算法复杂度O(*n*log*n*)

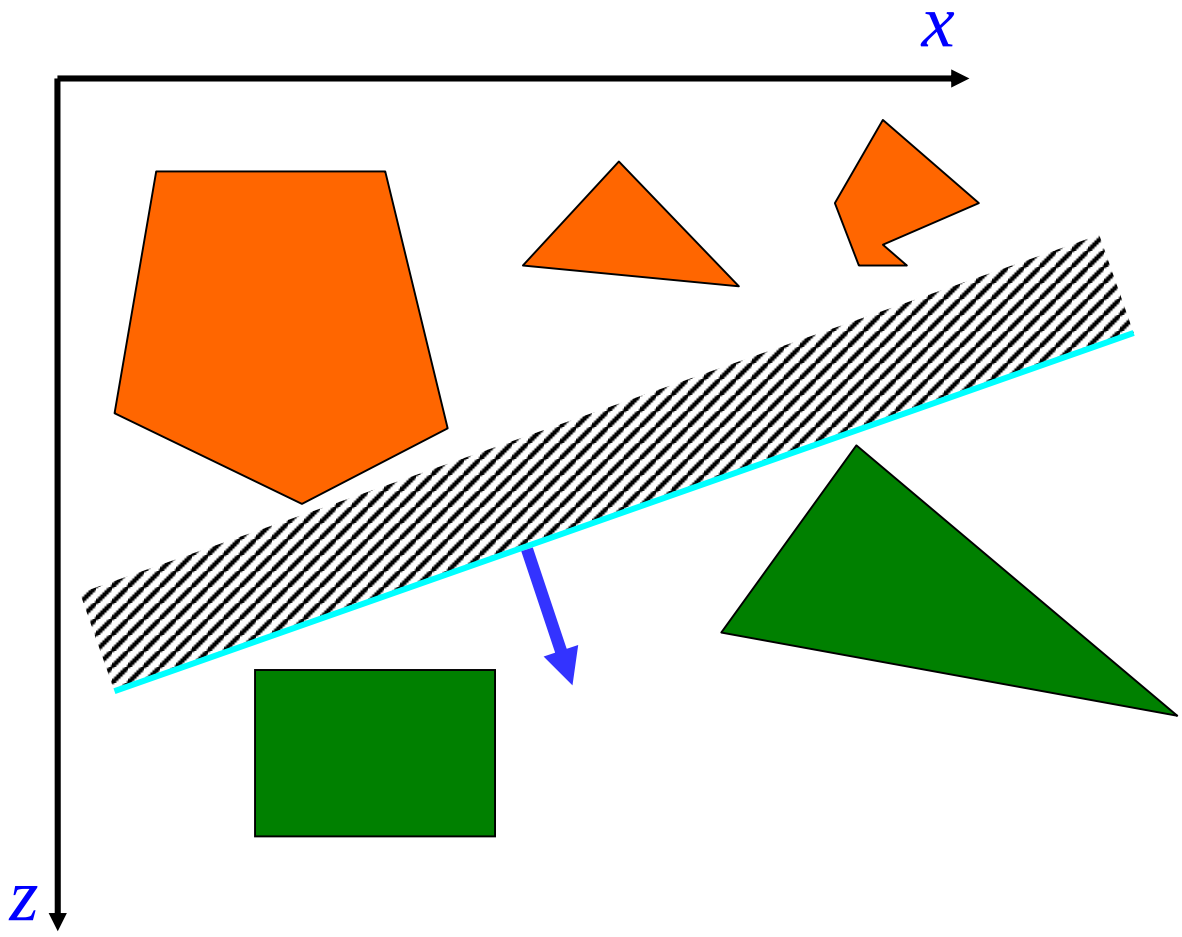
主要内容

- 背面剔除算法
- 表优先级算法
  - 三维物体的深度排序算法
  - 二叉空间剖分树算法
- 几种典型消隐算法的比较数据
- 关于秋学期的上机作业

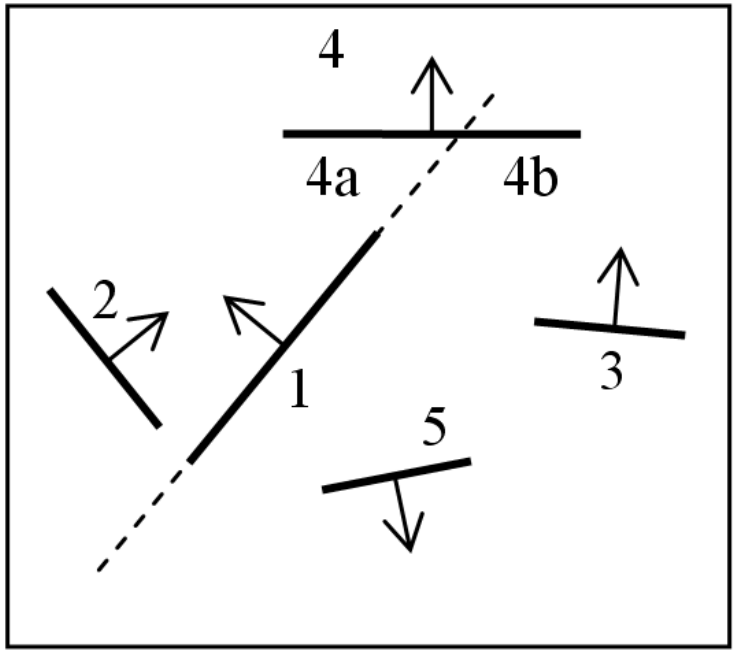
二叉空间剖分树

- 二叉空间剖分树 (BSP树-Binary Space Partitioning)的基本原理：
  - 如果场景中多边形被一个平面分割成两部分，那么当视点位于分割平面的正侧时，位于分割平面正侧的多边形会遮挡位于分割平面另一侧的多边形
  - 对位于分割平面两侧的多边形继续进行递归分割，直至每一个分割平面两侧或一侧只有一个多边形
  - 分割过程可以用一个二叉树的数据结构来表示
  - 在BSP树算法中，分割平面取作场景中多边形

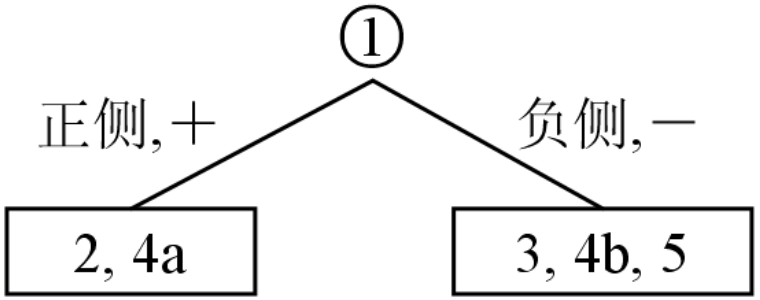
二叉空间剖分树



二叉空间剖分树实例

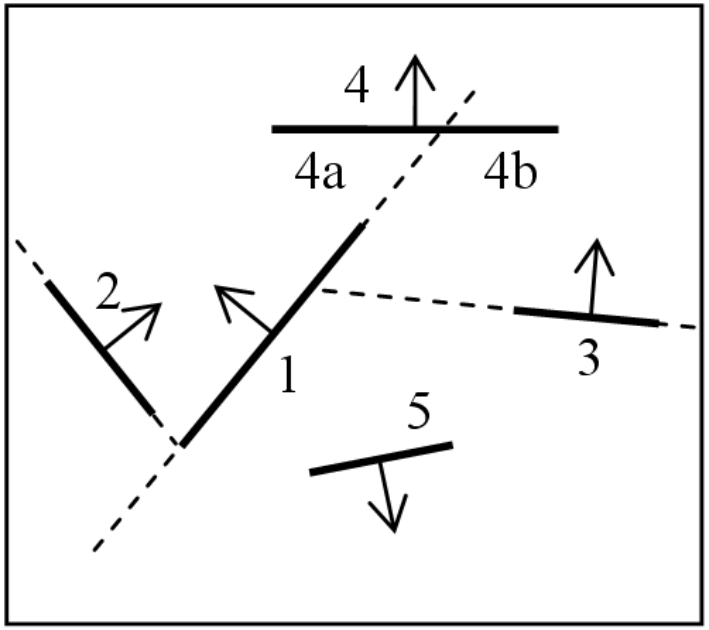


(a)

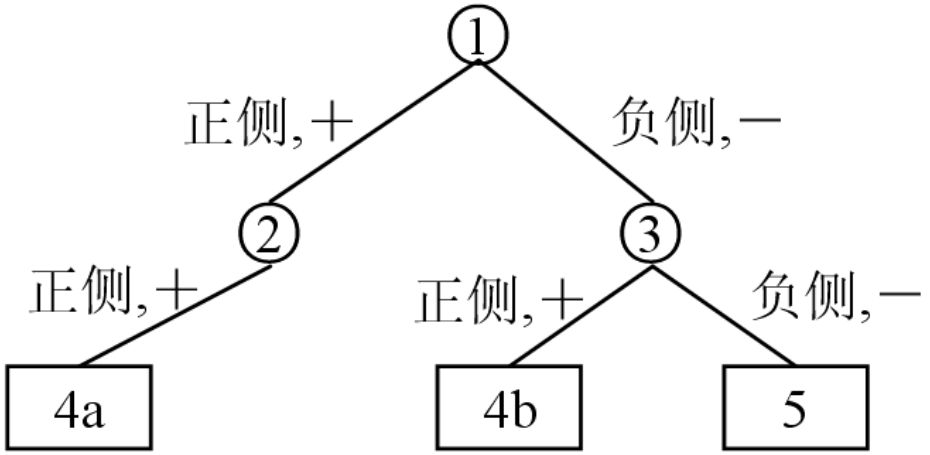


箭头表示多边形的正侧。在图 (a)中，首先选取1作为分割平面，2位于1的正侧，3和5位于1的负侧。4被1分割为4a和4b。其中，4a位于1的正侧，4b位于1的负侧。

二叉空间剖分树实例



(b)

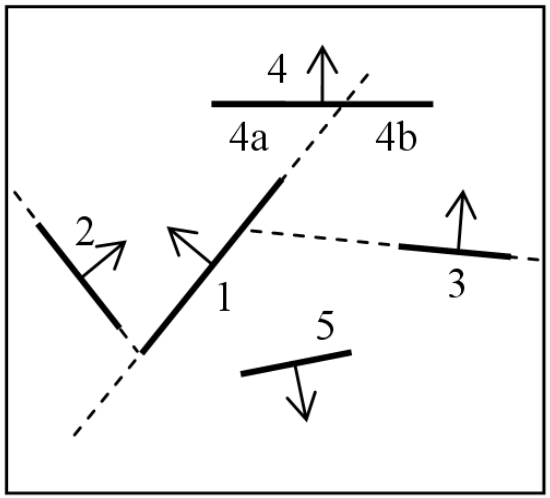


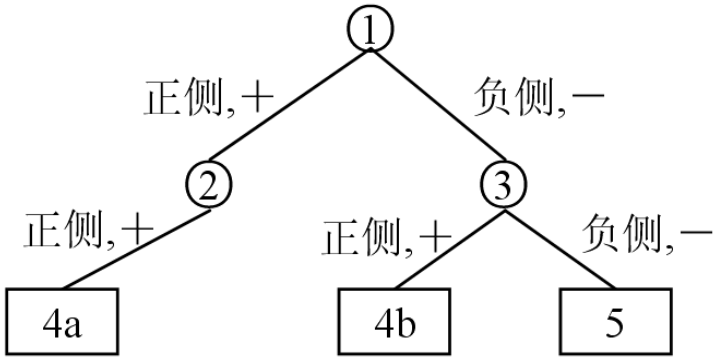
对上述二叉树的左右两个分支进行进一步的分割。对于左侧分支，再取2所在平面为分割平面，4a位于2的正侧；对于右侧分支，取3所在的平面为分割平面，4b位于3的正侧，5位于3的负侧。至此建立了对所给场景的BSP树，树的每个叶节点是一个多边形

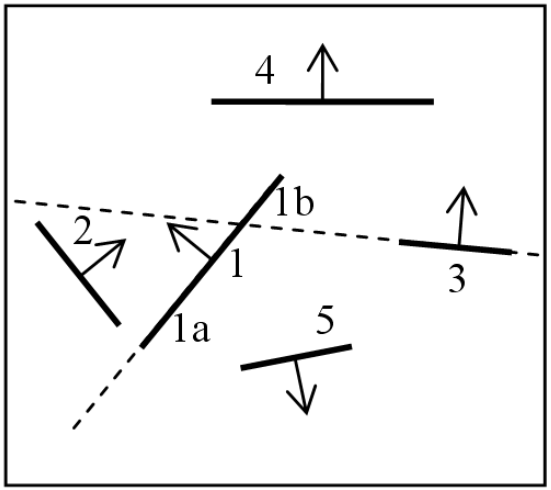
二叉空间剖分树

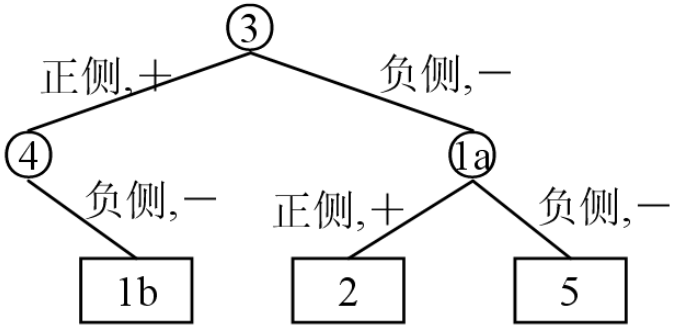
- 给定场景的BSP树不是唯一的
- “最佳”的BSP树的两个标准
  - 使BSP树尽可能平衡
  - 尽可能减少多边形的剖分

二叉空间剖分树另一种实现









二叉空间剖分树描述性算法

- 从列表中选择一个多边形 P
- 在 BSP 树中创建一个节点 N，并将 P 添加到该节点的多边形列表中
- 对于列表中的每个其它多边形:
  - 如果该多边形完全位于包含 P 的平面的前面，则将该多边形添加到 P 前面的节点列表中
  - 如果该多边形完全位于包含 P 的平面的后面，则将该多边形添加到 P 后面的节点列表中
  - 如果该多边形与包含 P 的平面相交，则将其剖分为两个多边形并将它们添加到 P 后面和前面的相应多边形列表中
  - 如果该多边形位于包含 P 的平面中，则将其添加到节点 N 处的多边形列表中
- 将此算法应用于 P 前面的多边形列表
- 将此算法应用于 P 后面的多边形列表

二叉空间剖分树的遍历

- BSP树遍历：递归建立多边形的优先级
  - 如果视点位于分割平面的正侧，那么该BSP树的遍历过程应当是：

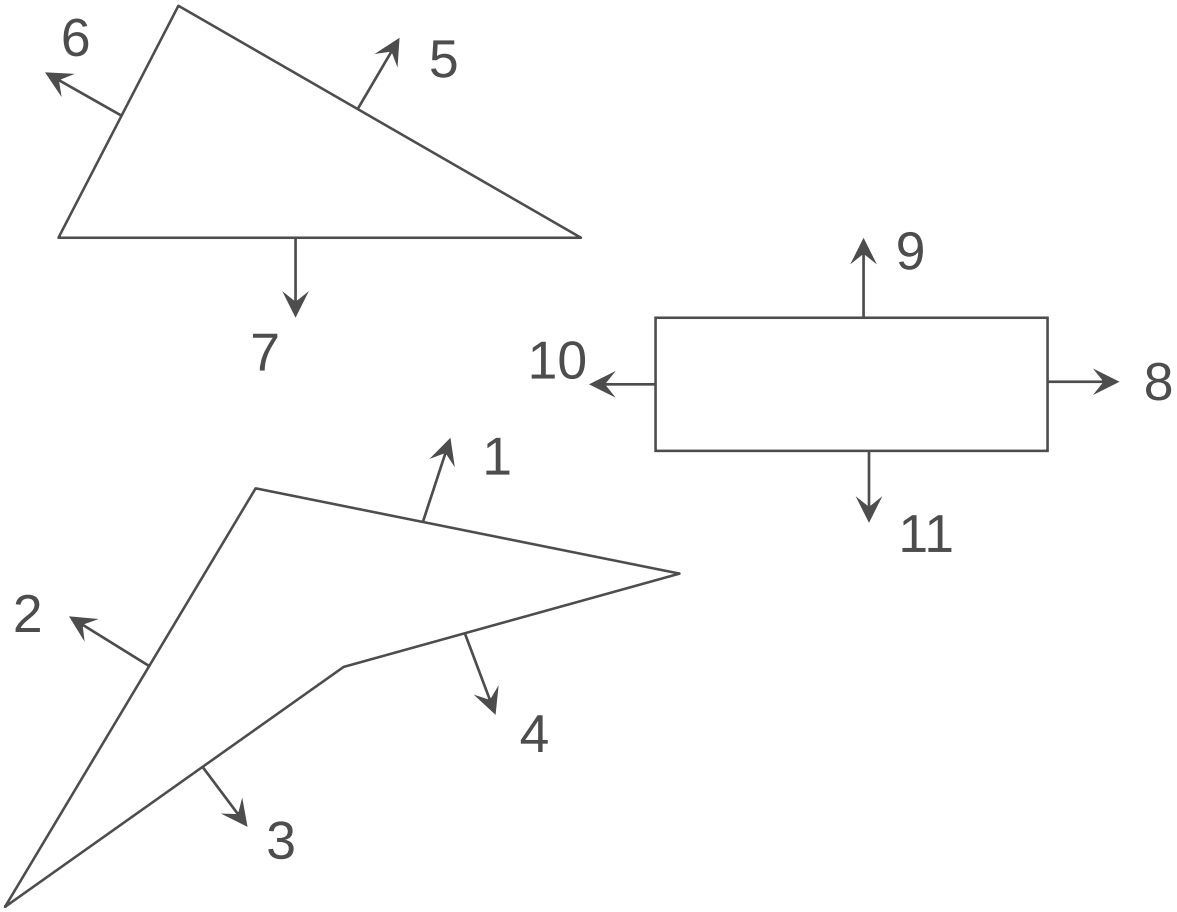
    负侧分支→根结点多边形→正侧分支
  - 如果视点位于分割平面的负侧，那么该BSP树的遍历过程应当是：

    正侧分支→根结点多边形→负侧分支
  - 这个判定标准递归地应用于每个子分支

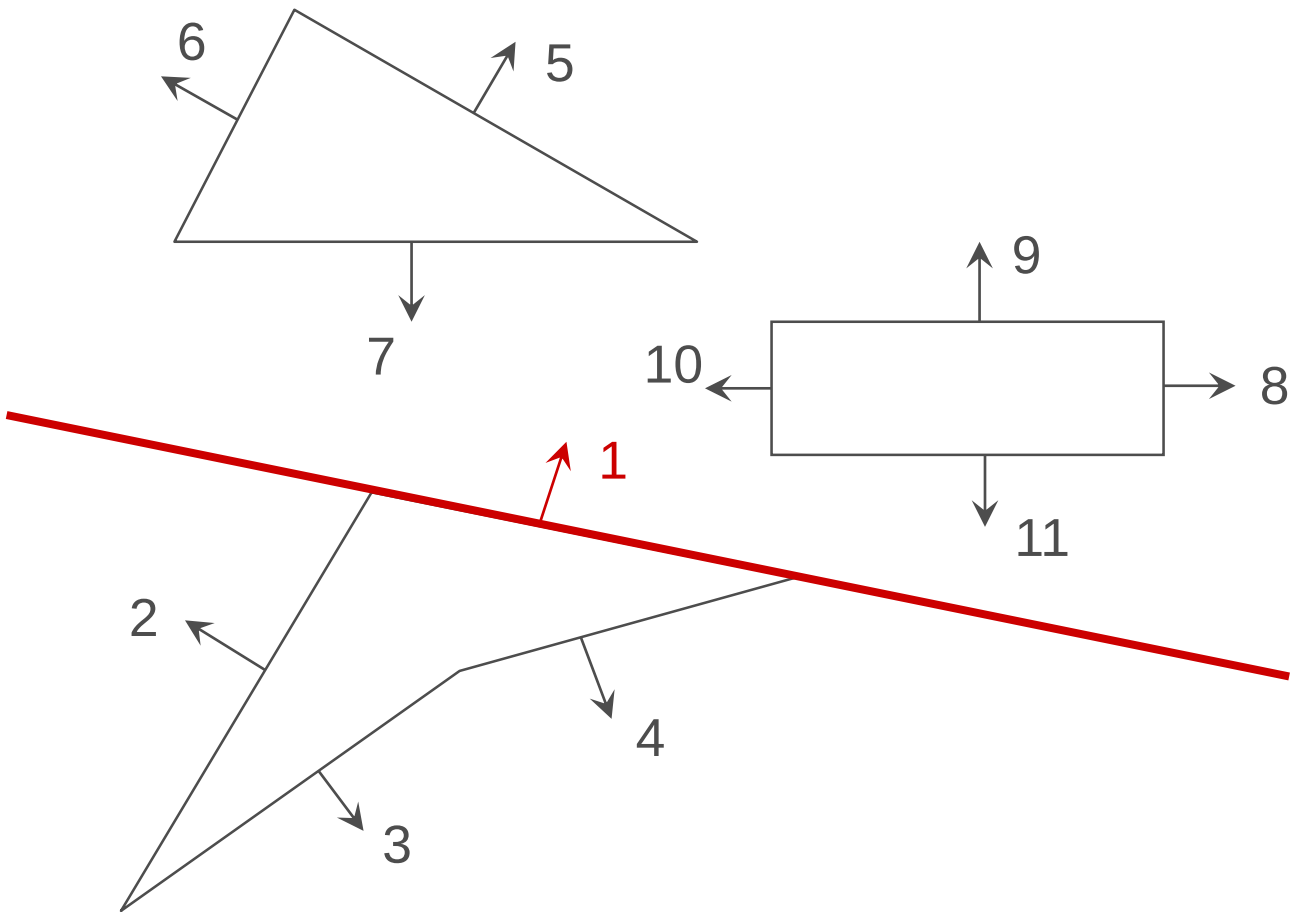
二叉空间剖分树的遍历算法

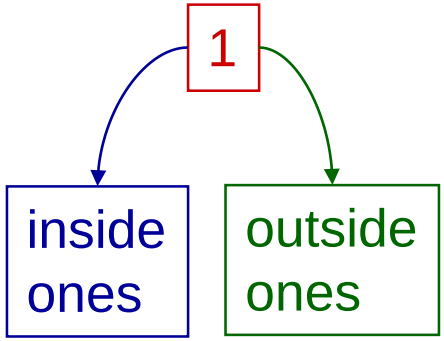
```
void showBSP(v: Viewer, T: BSPtree) {
    if (T is empty) return;
    P = root of T;
    if (viewer is in front of P) {
        showBSP(v: Viewer, back subtree of T);
        draw P;
        showBSP(v: Viewer, front subtree of T);
    } else {
        showBSP(v: Viewer, front subtree of T);
        draw P;
        showBSP(v: Viewer, back subtree of T);
    }
}
```

二叉空间剖分树举例

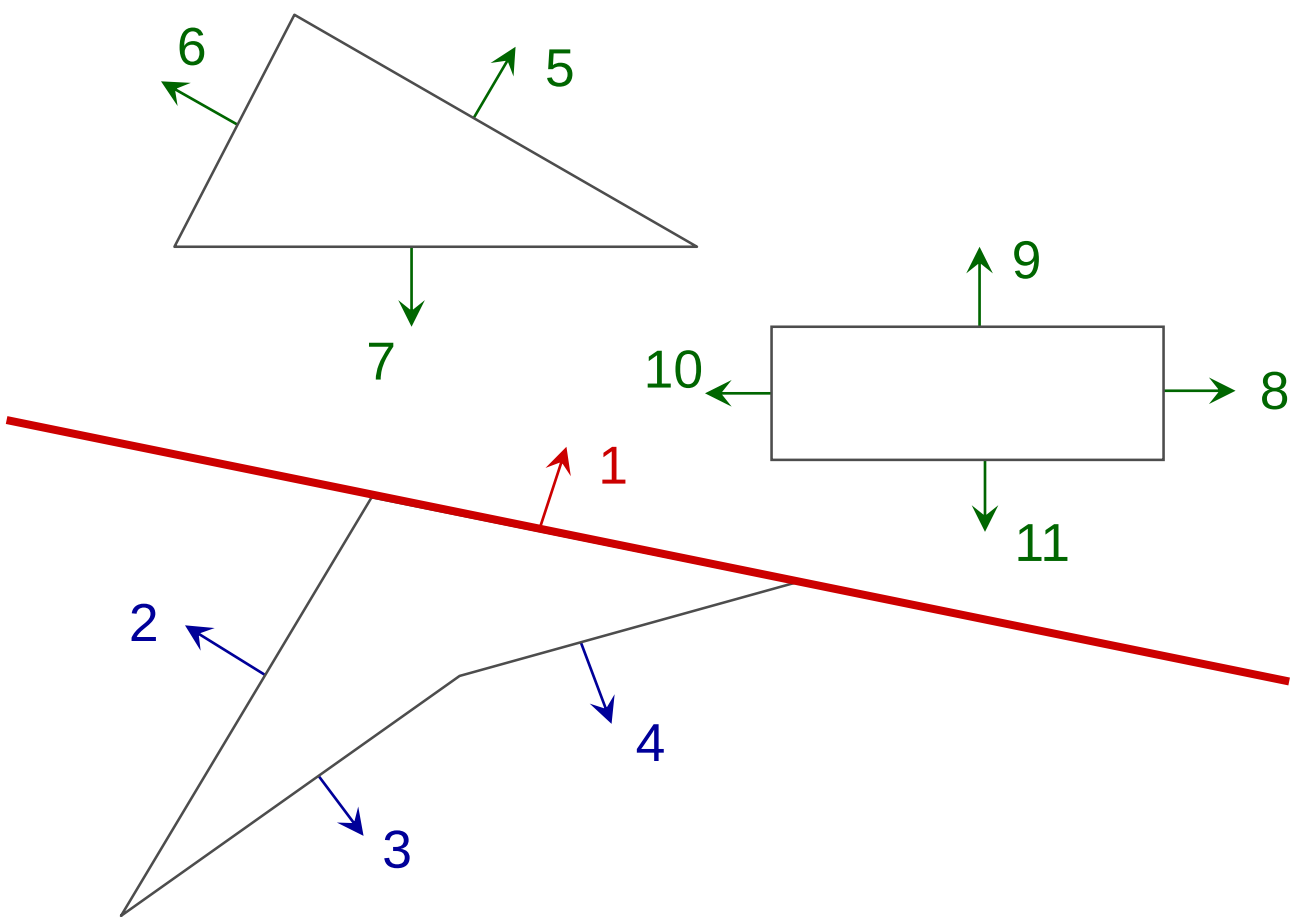


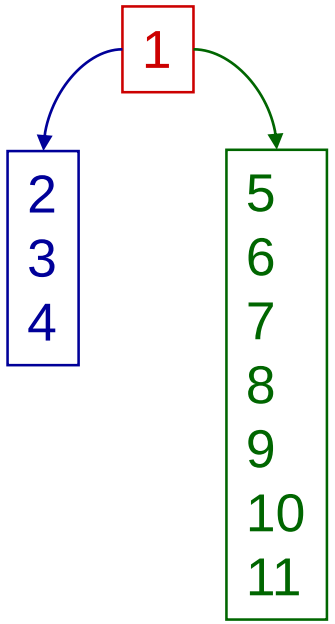
二叉空间剖分树举例



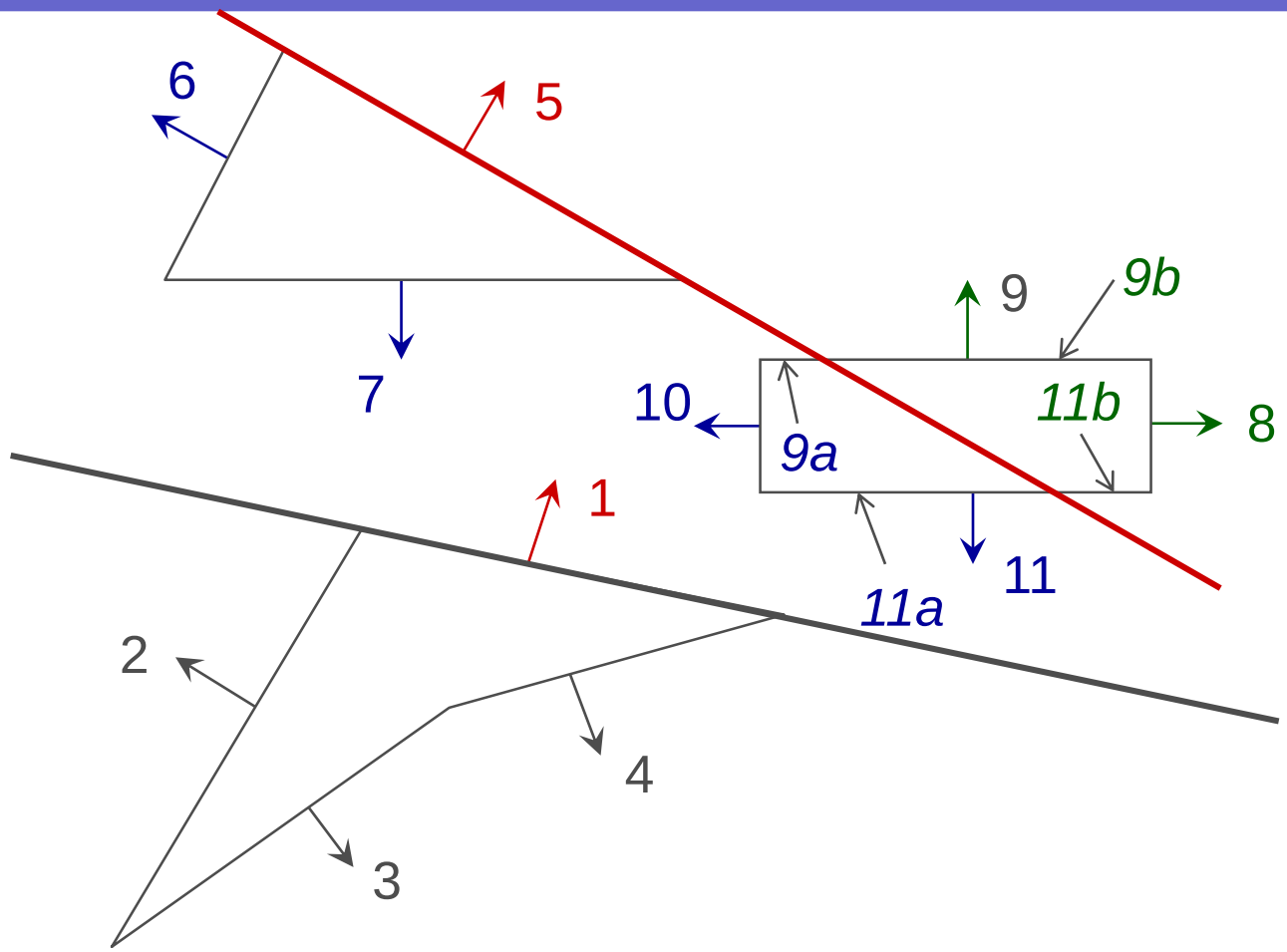


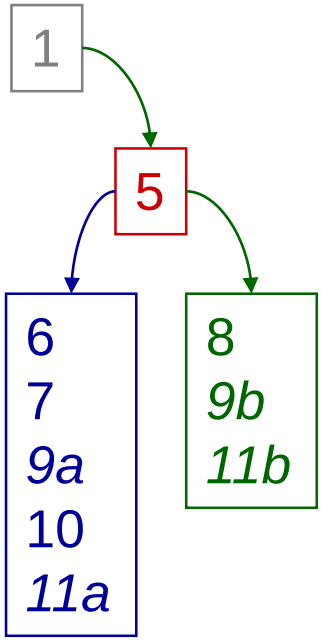
二叉空间剖分树举例



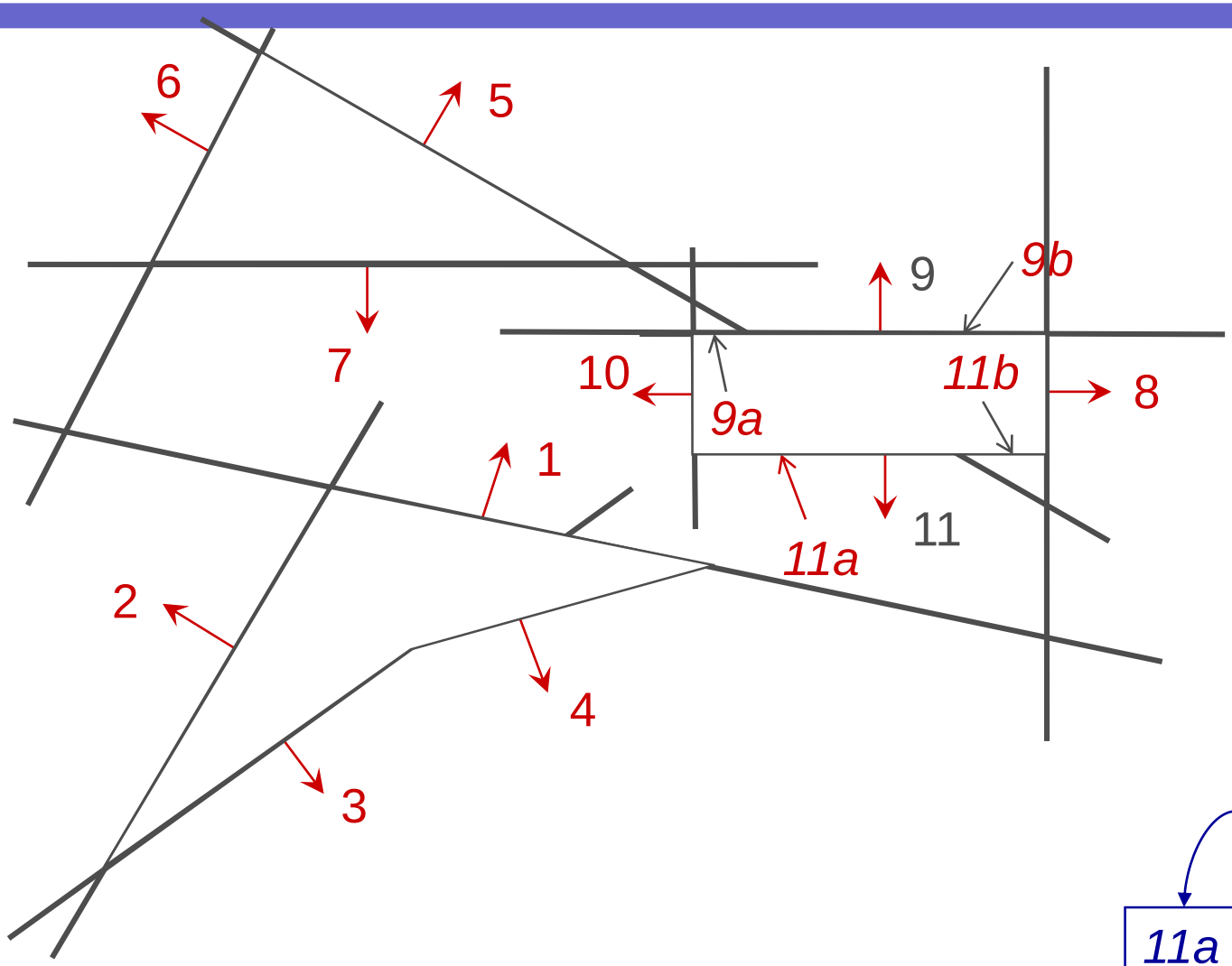


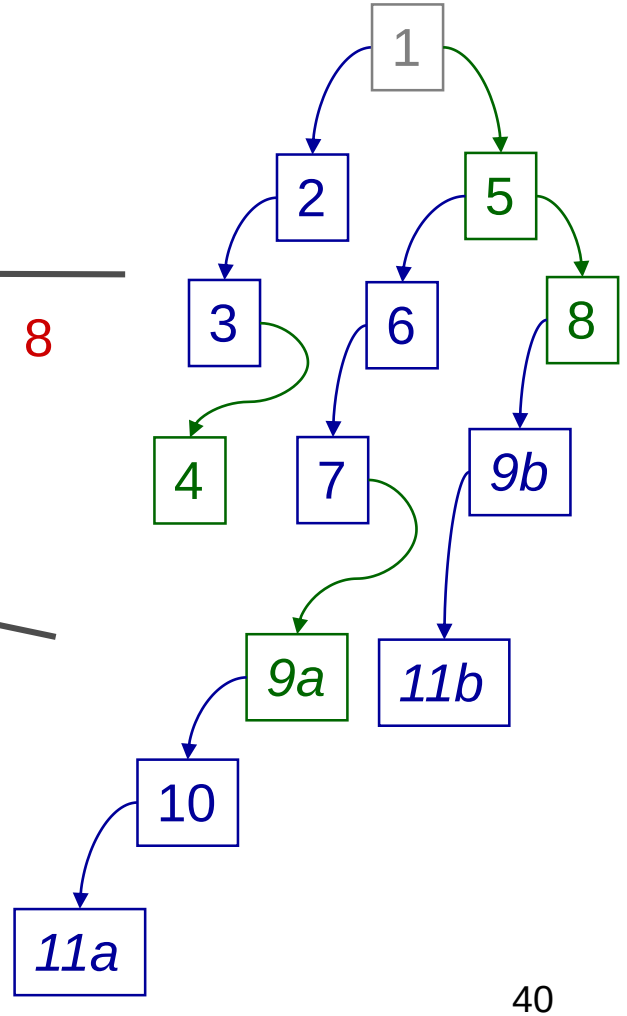
二叉空间剖分树举例



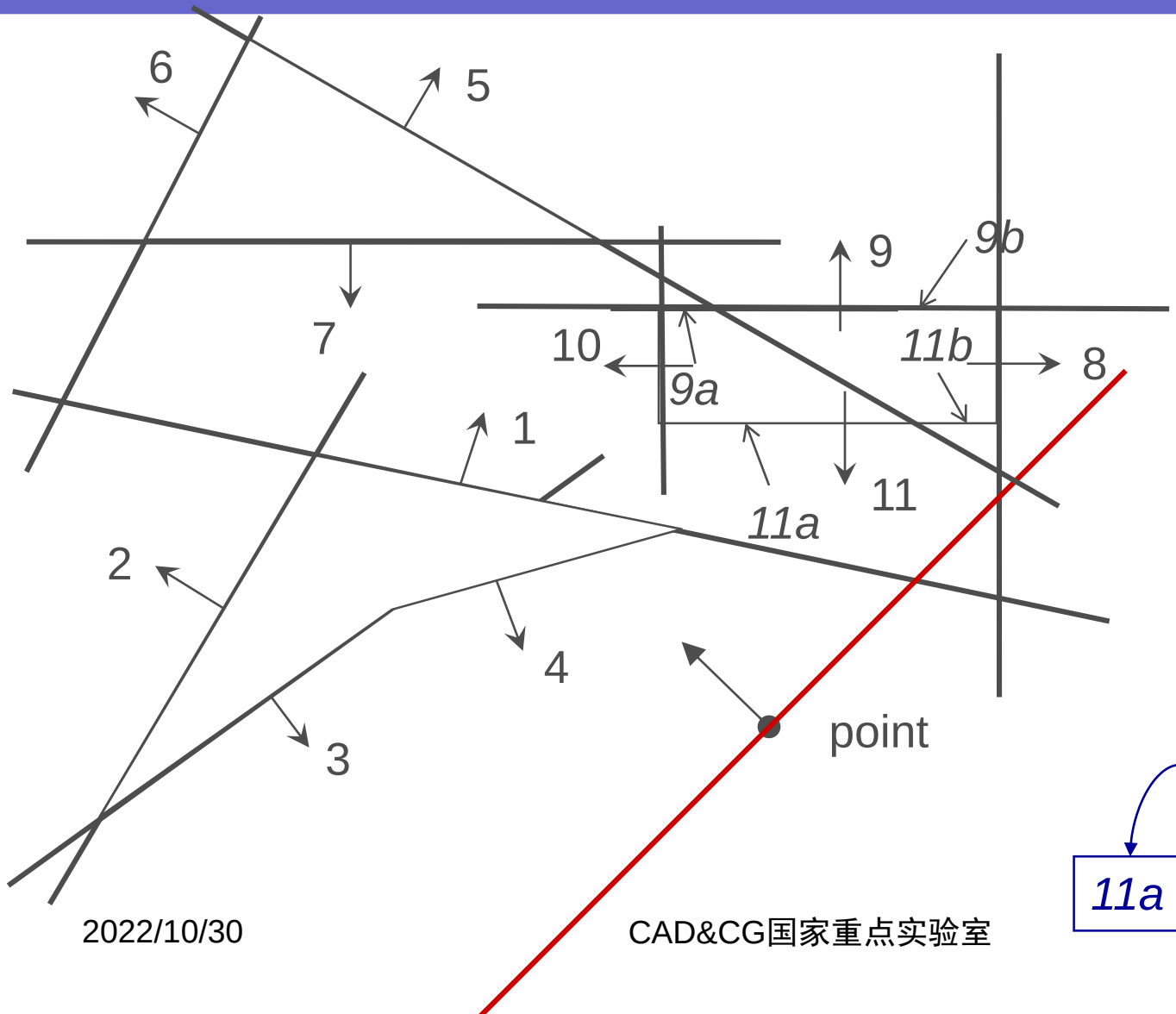


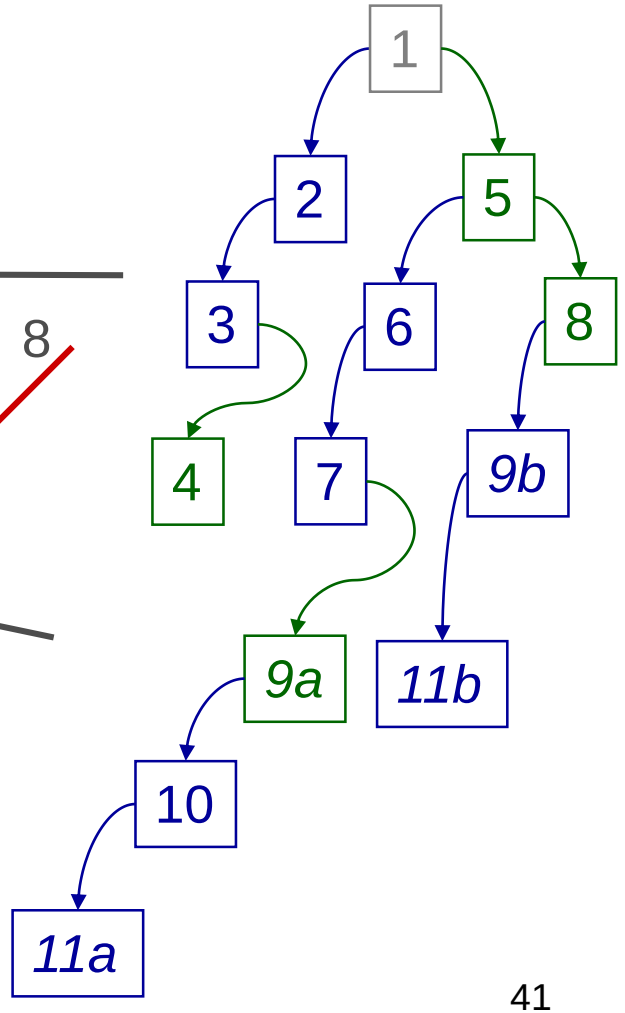
二叉空间剖分树举例



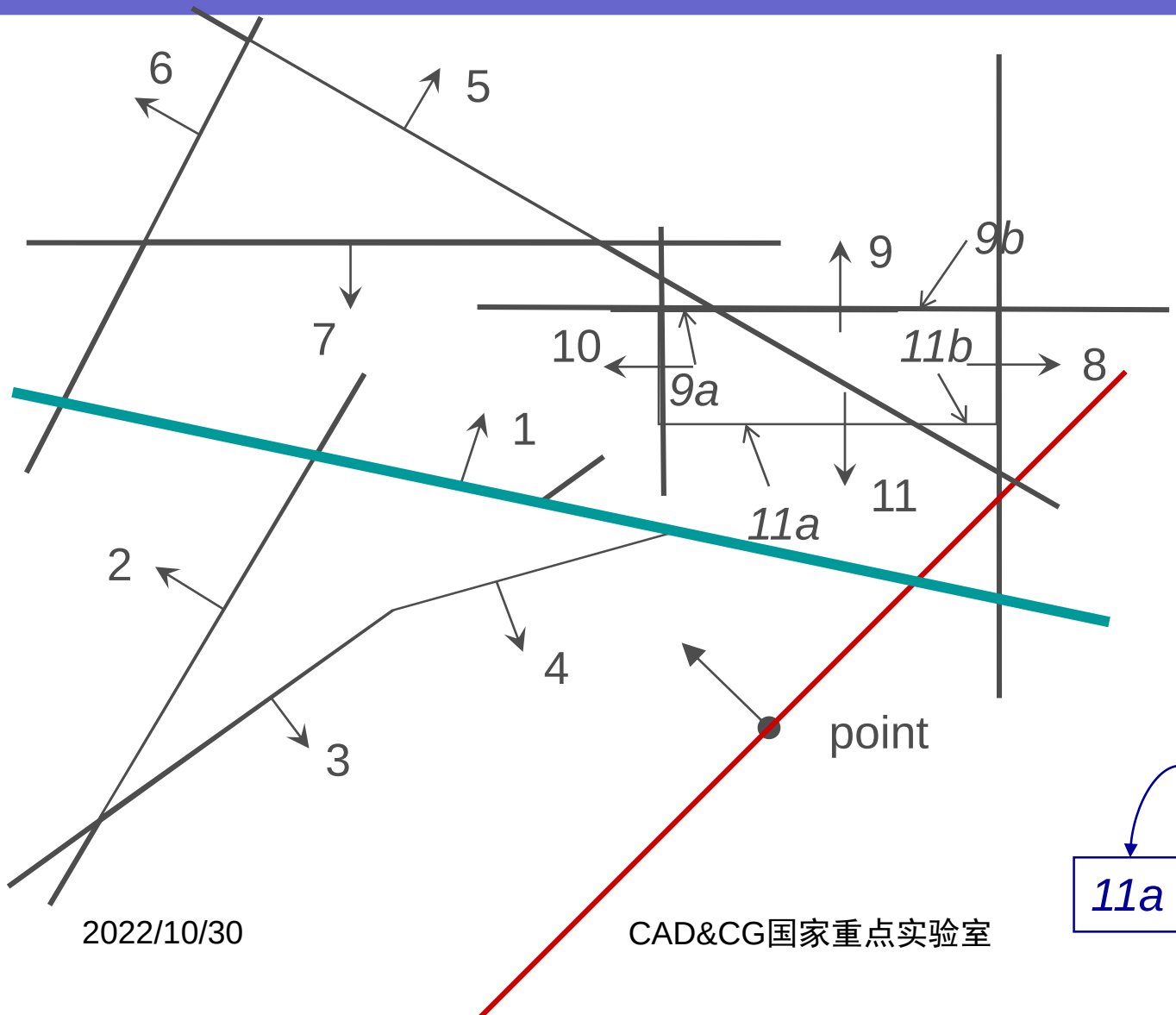


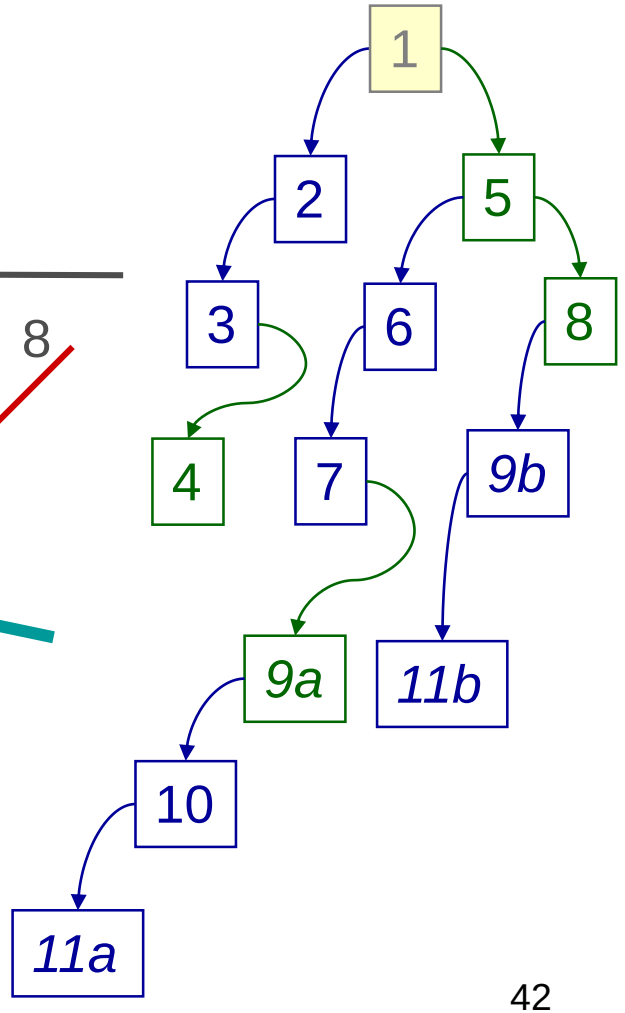
二叉空间剖分树举例



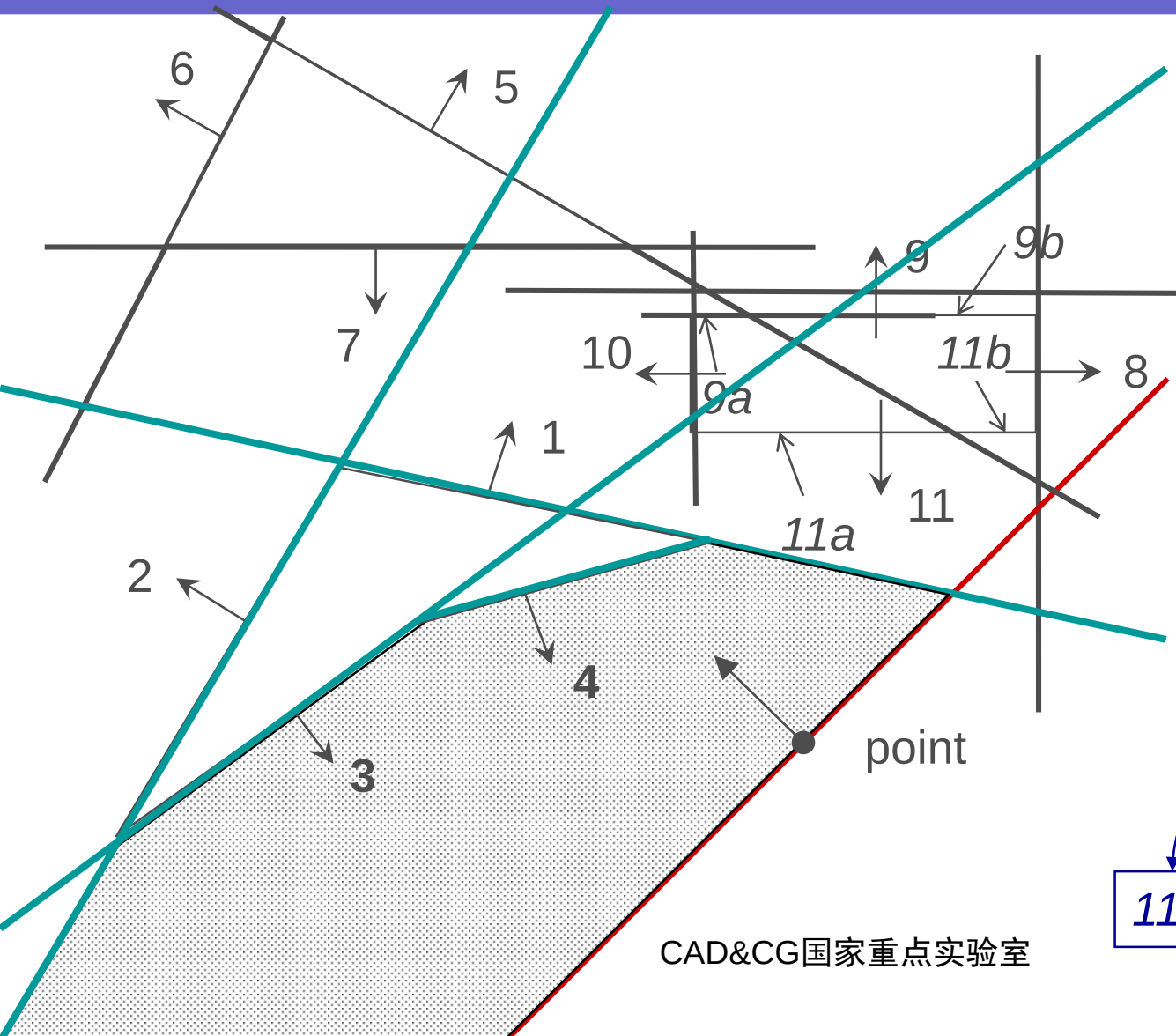


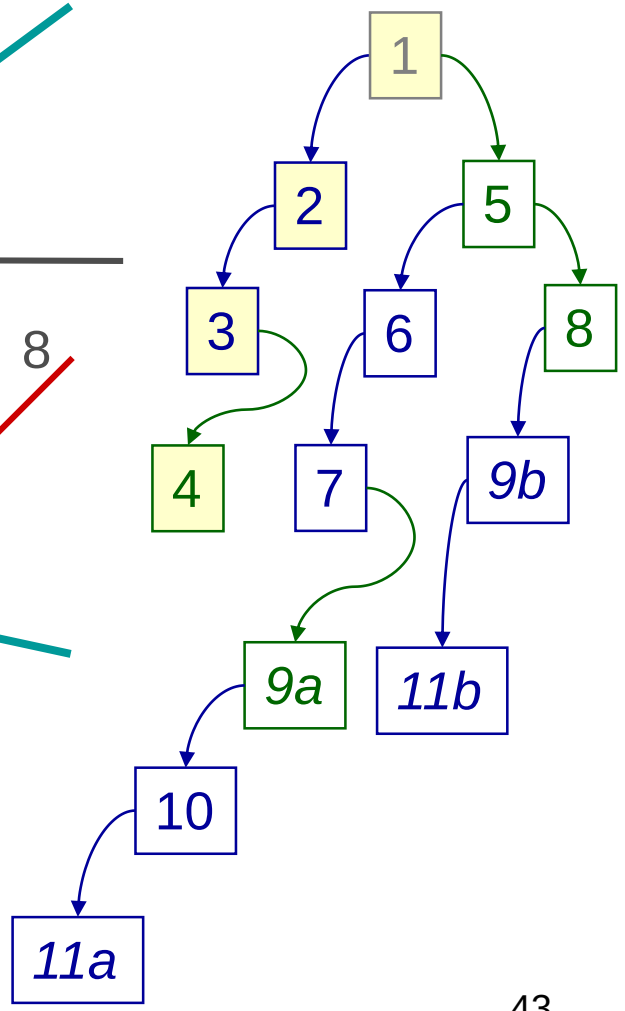
二叉空间剖分树举例





二叉空间剖分树举例





课后练习

- 手工遍历上述BSP树，给出正确的绘制顺序
- 如何建立一个平衡的BSP树？

主要内容

- 消隐的基本概念
- 区域细分算法(Warnock)
- 背面剔除算法
- 表优先级算法
- 几种典型消隐算法的比较数据
- 关于秋学期的上机作业

几种典型的消隐算法比较

算法	多边形数目		
	100	2,500	60,000
深度排序算法	1	10	507
z-缓冲器算法	54	54	54
扫描线算法	5	21	100
区域细分算法	11	64	307

Ivan E. Sutherland，Robert F. Sproull and Robert A. Schumacker，**A Characterization of Ten Hidden-Surface Algorithms**，*ACM Computing Survey*, 6(1):1-55, 1974

主要内容

- 背面剔除算法
- 表优先级算法
- 几种典型消隐算法的比较数据
- 关于秋学期的上机作业

上机(四选一) + 读书报告(二选一)

上机1：实现**扫描线z-buffer算法** (50分)

- 场景实现：可以导入场景，如OBJ、3DS等；不同模型规模(1K/10K/100K面片)。也可自己编程实现简单的建模功能(可以得高分)
- 评价指标：面片数不少于1000个；算法效率；是否有加速等，综合考虑。

上机2：实现**Warnock算法和Weiler-Atherton算法**(60分)

- 在同一个界面中选择上述两个消隐算法；
- 给出不同模型规模下(1K/10K/100K面片) 两个算法的效率对比

上机3：实现**层次z-buffer算法**(70分)

- 完整模式(层次z-buffer+场景八叉树)
- 分别对比简单模式和完整模式与扫描线z-buffer算法的加速比

作业：上机(四选一) + 读书报告

上机4：实现**曲面物体的线消隐算法**(70分)

- 绘制Utah Teapot模型：见课件**CG13.zip**
- 每个Bézier曲面的等参数线数目可以由用户输入；
- 算法不限，自由选择：基于z-buffer消隐的线消隐算法(**50分！**)

读书报告：30分

读书报告1：计算机图形学的历史、现状和展望

读书报告2：图形处理器(GPU)的历史、现状和展望

- 中文撰写，结构清晰，图文并茂，5000字以上。禁止整段复制(连续三个句号之间的语句复制)
- 10篇以上参考文献，英文文献不低于5篇
- 展望部分：独立思考(10分！)

秋学期的上机作业要求

- 除题目(4)之外，不能调用z-buffer算法API函数！面片着色函数自选
- 编程语言及环境：C/C++，常见的开发环境，如MS Visual Studio；
- 场景实现：可以导入场景，如OBJ或3DS等等。也可自己编程实现简单的建模功能(可以得高分)
- 评价指标：面片数目、模型复杂度、算法效率、是否有加速、程序注释和文档等，综合考虑。

关于秋学期作业的提交

- 上机作业：说明文档(编程环境、用户界面使用说明、数据结构说明、加速与否等)、源代码和工程文件、实验结果报告。
- 读书报告：docx 或者 pdf文件
- 作业提交： **2023年1月11日之前**将作业发至我的邮箱

  jqfeng@cad.zju.edu.cn

  非常重要：邮件主题为“**CG_姓名_学号**”。将上机作业和读书报告打包压缩，作为邮件附件**链接(用浙大邮件系统的云盘链接,不接收校外云盘：网易、QQ等)**，附件命名为“**CG_姓名_学号.zip/rar**”。不要直接作为附件！
- 若没有收到**邮件回复**，说明作业没有收到，请重新提交

关于秋学期作业附注说明

- 按时提交：**迟一天，“-1”分**
  - 秋学期作业
    - 消隐算法作业“四选一”+ 读书报告“二选一”
    - 提交：冯结青老师 **2023年1月11日之前**
  - 冬学期作业：王锐教授布置和要求
    - 提交：参照王锐老师要求，**不要交给冯结青**老师
- 缺少任何一部分作业，算作缺考，无成绩
- 抄袭：**0分**
  - 如参考相关代码，需要给出出处，并指出不同之处
  - 如没有说明，且查到相同或类似代码，按抄袭处理