Ray tracing acceleration 加速度 //怎样让ray tracing 更快

更快的 intersection 交叉--bounding volume 给object设定一个范围

Fewer Ray-Object Intersection 更少的交叉

Hierarchical bounding volumes 分层的bounding volume

Spatial data structures

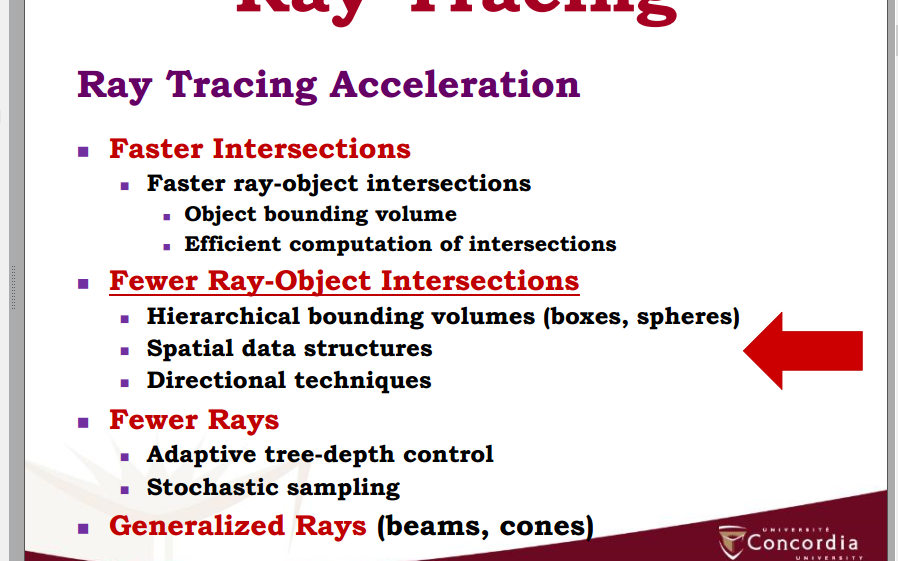
Directional techniques

Fewer Rays

Adaptive tree-depth control

Stochastic sampling

Generalized Rays (beams,cones)



Spatial data structure的好处 空间数据结构

数据结构储存的是几何信息

加速了大量的应用例如：

碰撞检测 collision detection

位置查询Location queries

模拟 simulation

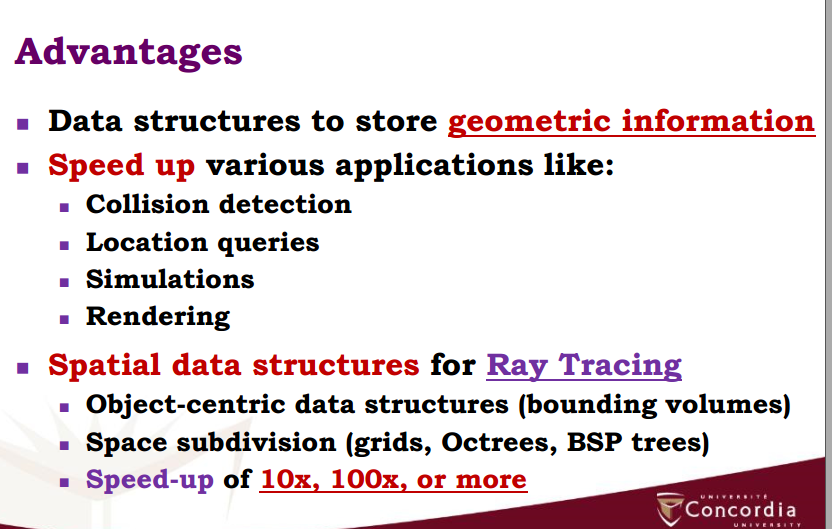
Rendering

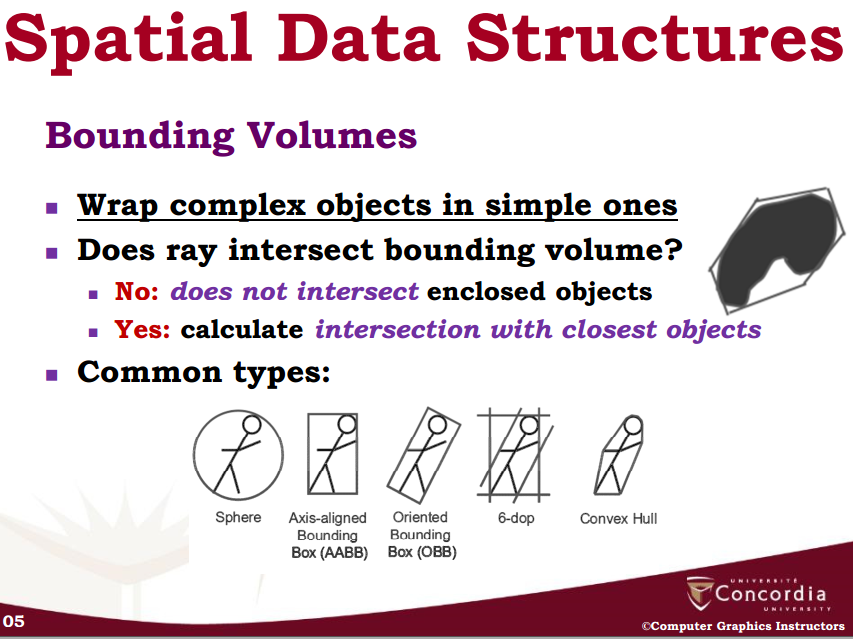
Spatial data structures for Ray Tracing

是以object为核心的数据结构

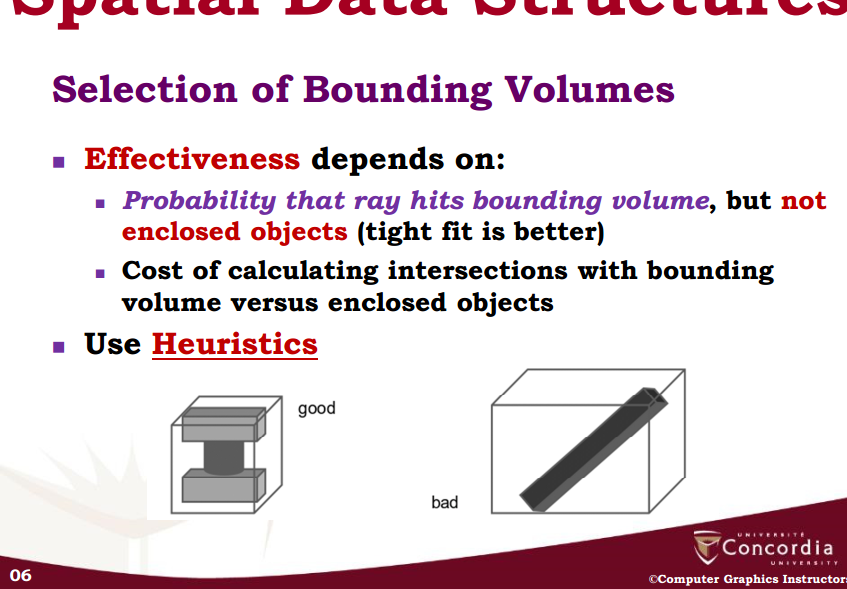
切割了空间

加速了10x 100x甚至更多





把复杂object bound到一个简单图形里



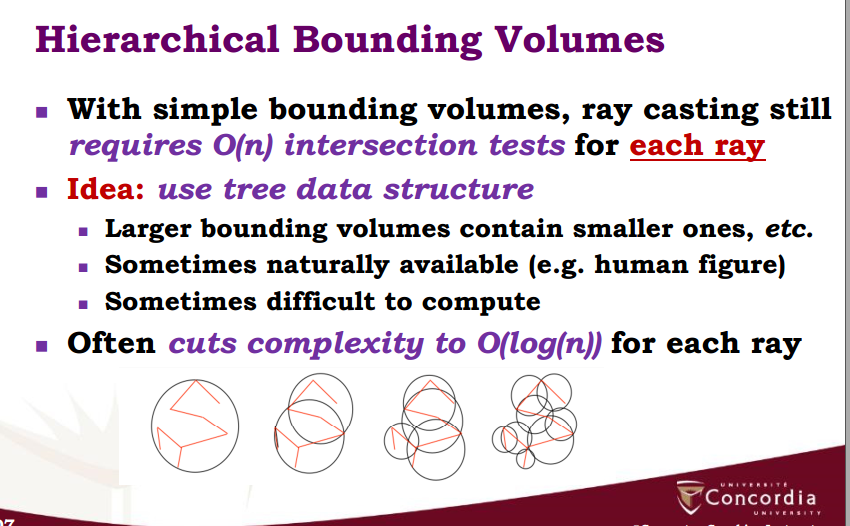
选取合适的bounding volume

效率取决于

Ray击中bounding volume，但是没有击中被包含的物体的可能性（紧的更好）

计算bounding volume与 包含物体 打的intersection所需需要的损耗

使用Heuristics 启发式教学法

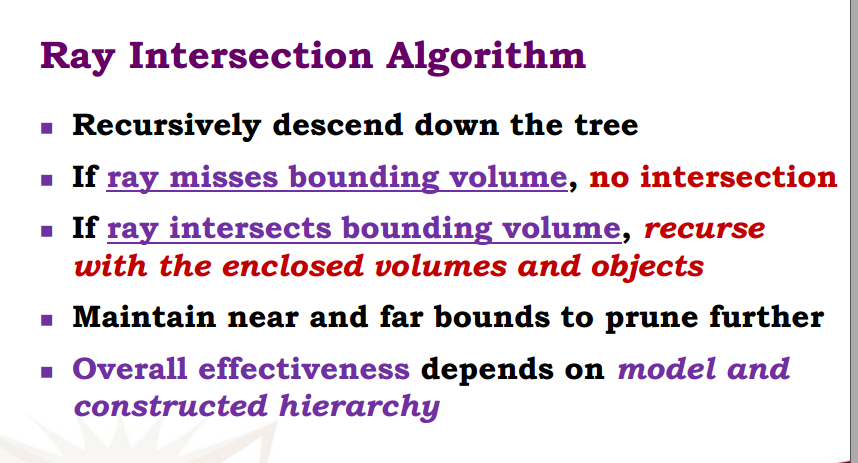


通过简单的bounding volume， ray casting仍然需要 O（n）复杂度的inter section tests来test 每一条ray

所以我们用树状数据结构

更大的bounding volume 包括了更小的部分，

复杂度是O（logn）



Ray intersection algorithm

不停的重复往下画这棵树

如果ray 没有射到这个bounding volume里，就没有intersection

如果ray射到了bounding volume里 ，就在enclosed volume 与objects之间循环

保持near与far bound来进一步修建

总共的来说效率取决于model 与层次化的构建结构

这里可以搜索 hierarchy bounding volume来看例子

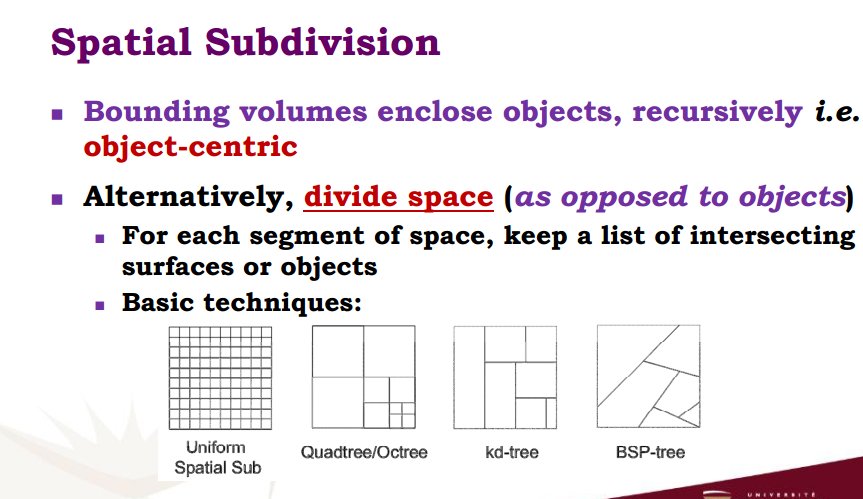
Spatial Subdivision 空间切割

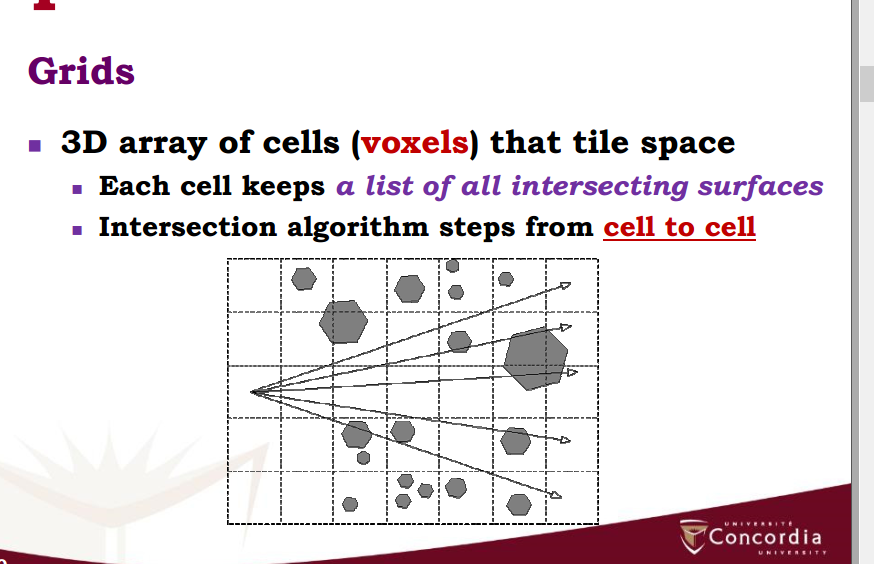
Bounding volume会重复的封装object ：它是以object为中心的

我们也可以选择divide space ，与 以object无关

对于每一部分space，记录交叉的面或object，

基本的技巧有方格，octree, kd tree bsp tree





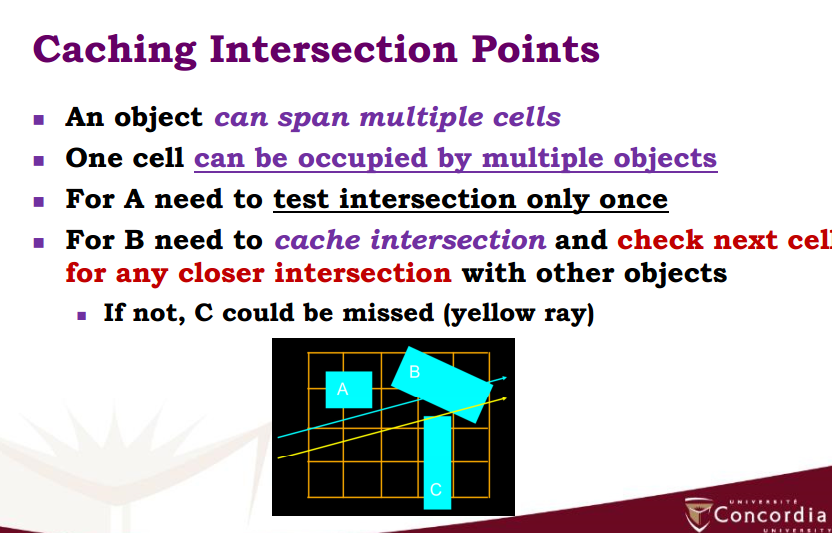
方格，

tile(以瓷砖的方式铺满)

voxel 3D单元格

你可以看见有些cell不止一个交点，因此每个cell存储的是交点的list

Intersection算法从一个cell单元格到下一个单元格



caching cache的动词， 缓存

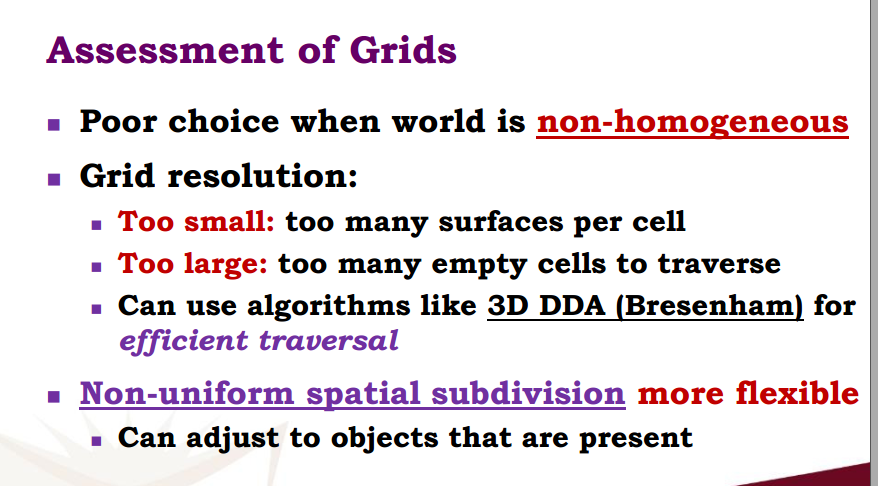
一个object能够占据多个cell

一个cell也有可能被多个object占据

对于A来说只需要被test intersection一次 //因为没有intersection

对B来说就需要cache intersection缓存intersection，然后检查下一个单元格是否有与其他object更近的Intersection

如果不是，C可能Miss



assessment 评估

non-homogeneous非齐次的

resolution分辨率

traverse穿过

non-uniform 不统一的

spatial 空间

subdivision 细分

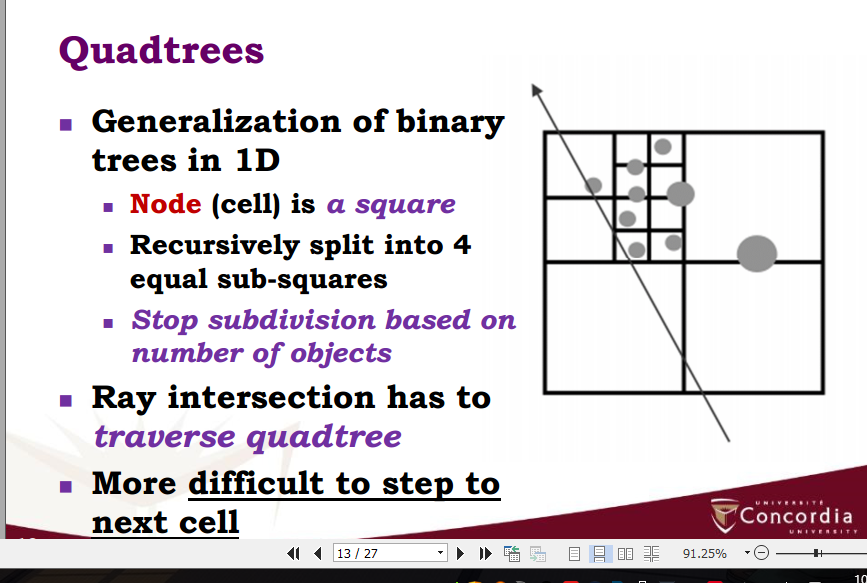
如果世界是非齐次的，那么使用grid非常差

grid分辨率太小的话：一个网格面积太大，所以一个网格可能包含多个表面

太大：面积太小，有很多空的网格要穿过

不统一的空间切割 更加flexible （就是接下来的例子）

可以适应多种情况



quadTree

生成一维的binary tree

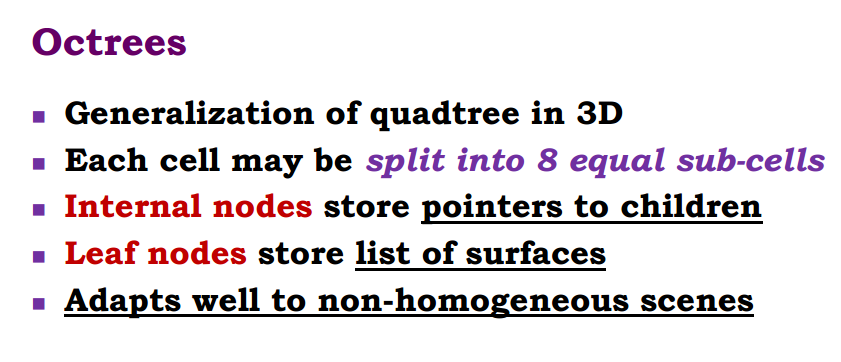
每一个cell都是一个正方形

不停地把这个正方形四等切

何时停止切割：取决于object的数量

ray必须穿过这个quad tree

更加难以进入下一个cell



octree：

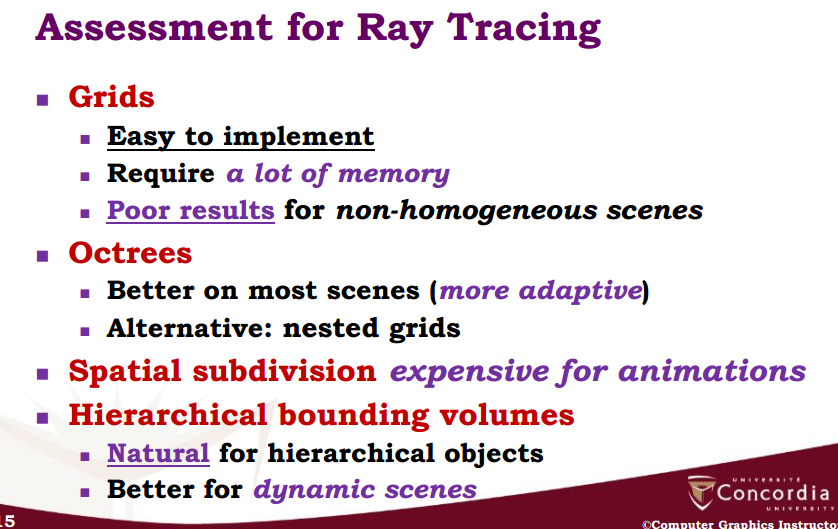
生成一个3D的quadtree

每一个cell被切割成八份

内部的node存储着对他的八个children的pointer

leaf node存储着list of surface

适应非齐次场景



grid

容易安装

需要大量内存

无法应对非齐次场景

octree

更能应对大多场景

替代选项：nested grids

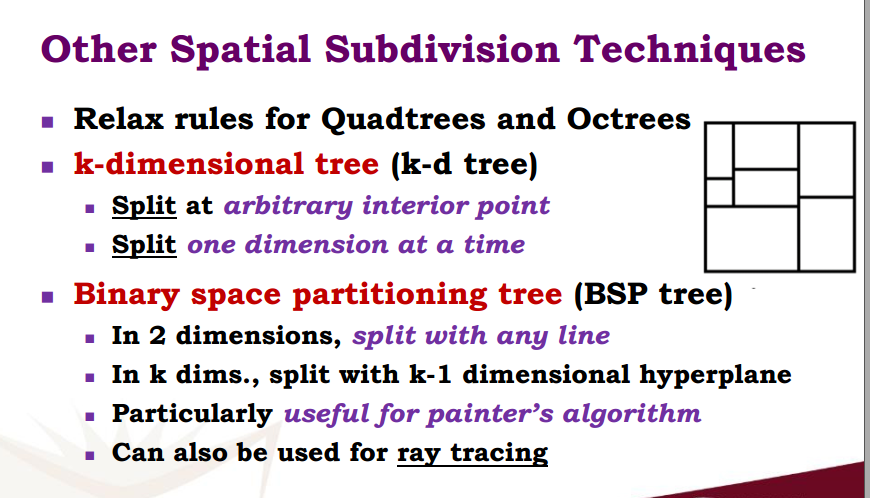
spatial subdivision 空间切割

对动画来说很贵

hierarchical bounding volumes 分层bounding volume

对层级object很自然

更能应对动态场景

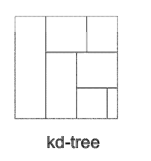


arbitrary 任意

hyperplane:超平面

k-d tree

k维空间tree



在任意内部点切割

一次切割一维

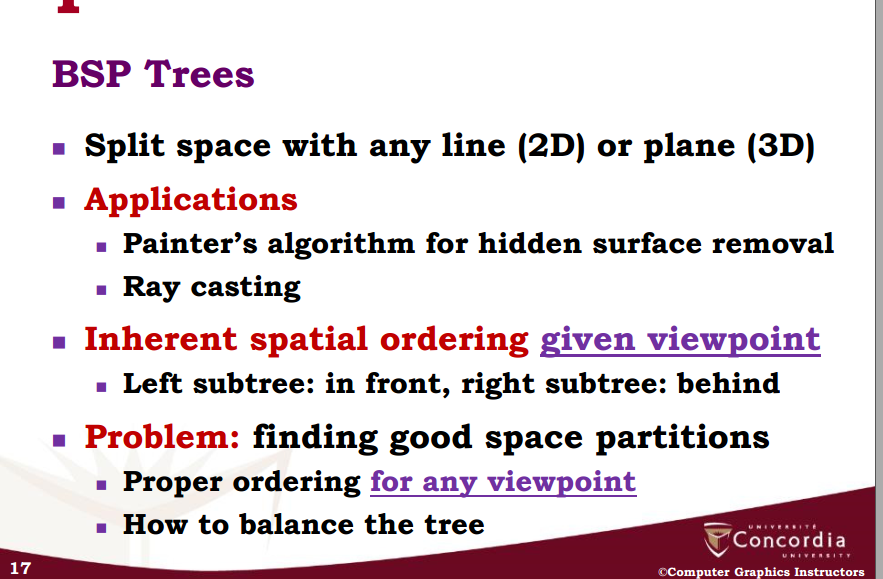
bsp tree



在二维空间，用任意线切割

在K 维空间，用任意k-1维超平面切割

BSP tree对painter算法很有效

.

inherent spatial ordering 固有的空间顺序

BSP tree

使用线（2D的时候）或者面（3D）来切割空间

应用

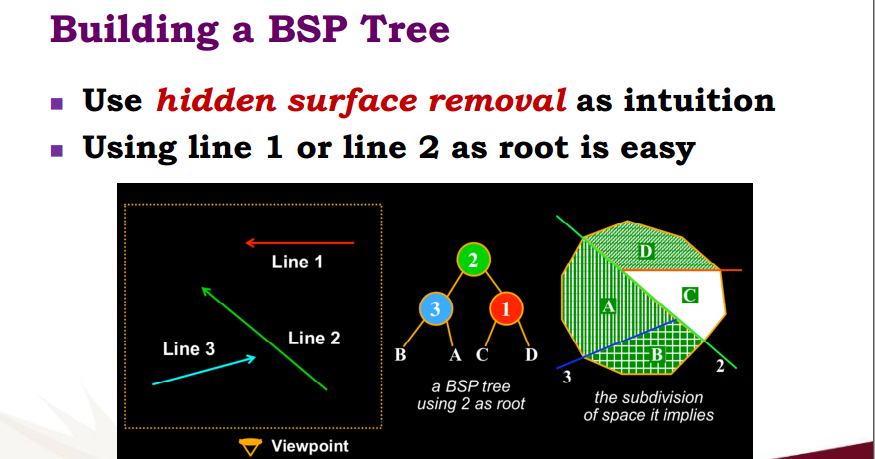
painter算法

ray casting

固有的空间顺序给定的视角

左subtree:front, 右subtree:behind

问题：找到良好的空间切割



看最新三个油管收藏