四叉树, 八叉树, BSP 树 与 KD 树

-- 空间数据的划分与查找

内容

- 四叉树（Quadtree）
- 八叉树（Octree）

- 二叉空间划分树（BSP tree）
- KD 树（K Dimensional tree）

问题：1维空间数据查找

- 1维数组 int a[] = {35, 17, 39, 9, 28, 65, 56, 87};
- 实现 bool find(int a[], int val);

- 顺序查找，时间复杂度 O(N)
- 二分查找，时间复杂度 O(logN)；排序复杂度 O(N * logN)

- 如果要查询 k 次？
- 如果 a[] 中的数据允许插入、删除？

顺序查找 & 二分查找

```
bool find(int a[], int val)
{
    int n = length(a);

    for (int i = 0; i < n; ++i)
        if (a[i] == val)
            return true;

    return false;
}
```

```
int binary_search(int a[],
      int left, int right, int val)
{
    if (left > right)  return -1;
    int mid = left + (right - left) / 2;
    if (val < a[mid])
        return binary_search(a, left, mid - 1, val);
    else if (val > a[mid])
        return binary_search(a, mid + 1, right, val);
    else
        return mid;
}
```

二叉查找树（Binary Search Tree）

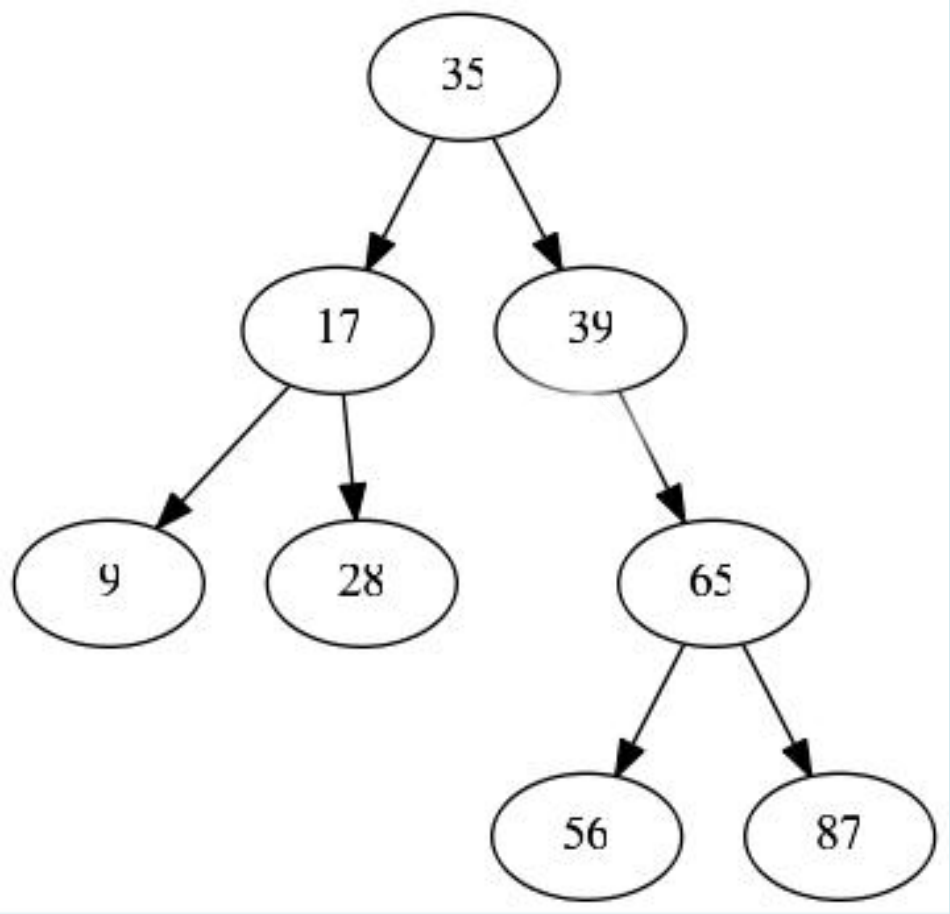
- 对原始数据按照一定规则进行组织、划分：
  - 左子树上所有节点的值小于根节点的值
  - 右子树上所有节点的值大于根节点的值
  - 左、右子树都各是一棵 BST

- 时间复杂度：
  - 建树 O(N * logN)，插入 O(logN)，删除 O(logN)，查找 O(logN)

- 空间复杂度：O(N)

二叉查找树例子演示

http://zh.visualgo.net/bst.html



问题：2维空间数据查找

- 2维平面上有 1000 个点，vector<Point2> points = {...};
- 实现 bool find(vector<Point2> &points, Point2 p);

- 暴力查找，时间复杂度 O(N)
- 空间换时间，假设所有点都在 1000 * 1000 的 2D 网格上，用一个二维数组记录每个网格点是否有点，时间复杂度 O(1)，空间复杂度 O(1000 * 1000)
- 四叉树（Point Quadtree），即二维数据情况下的 BST

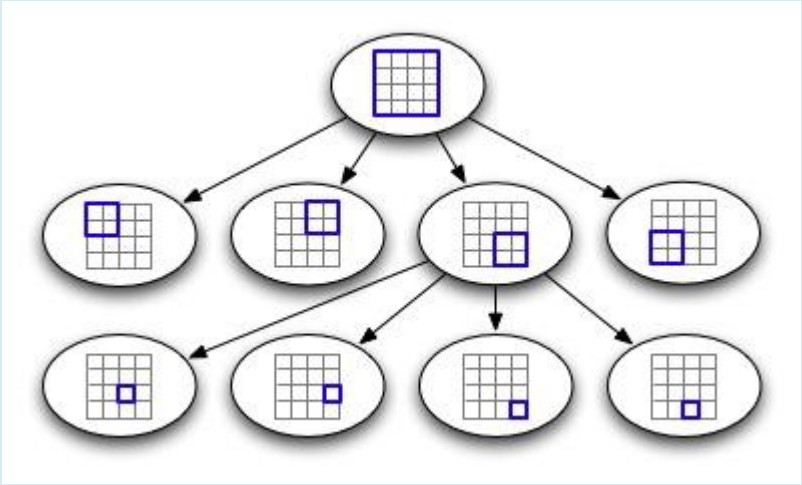
内容

- **四叉树（Quadtree）**
- 八叉树（Octree）

- 二叉空间划分树（BSP tree）
- KD 树（K Dimensional tree）

四叉树

- 四叉树是一种树形数据结构，其每个节点至多有四个子节点，表示将当前空间划分为四个子空间，如此递归下去，直到达到一定深度或者满足某种要求后停止划分。



四叉树定义

- 每个节点对应着一个正方形区域
- 如果一个节点有子节点，该节点对应的正方形区域被划分为四个小正方形区域，每个子节点对应一个小正方形区域
- 递归划分每个子节点
- 划分终止条件：层数（深度） 或 正方形区域面积 或 正方形区域内点（物体）数

四叉树定义

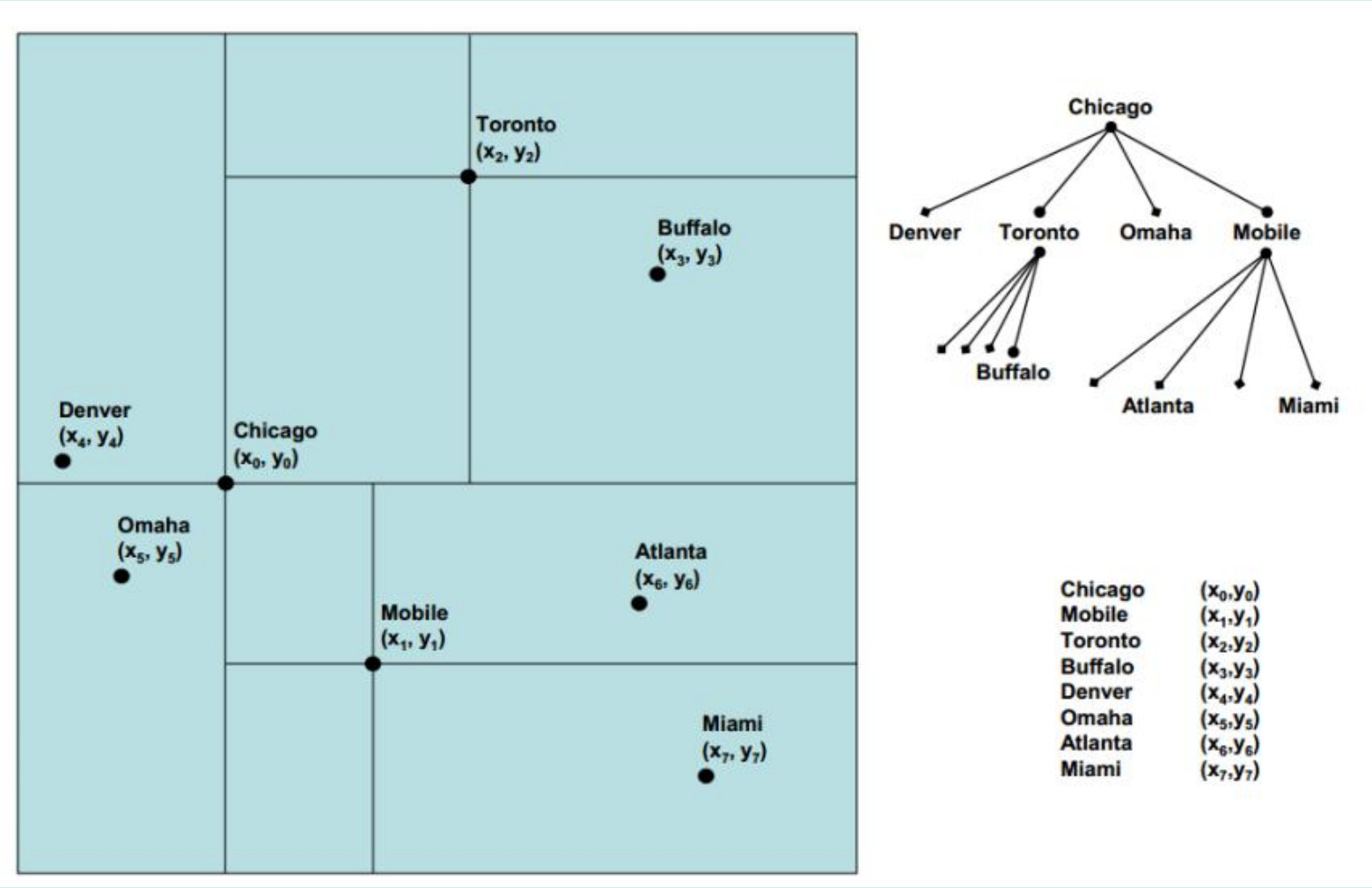
```
class Quadtree
{
public:
    Quadtree(float x, float y, float w, float h,
        int level, int max_levels);
    ~Quadtree();
    void add_object(Object *object);
    vector<Object*> get_object_at(float x, float y);

private:
    float x, y;
    float width, height;
    int level;
    int max_levels;

    vector<Object*> objects;
    Quadtree *parent;
    Quadtree *NW, *NE, *SW, *SE;
};
```

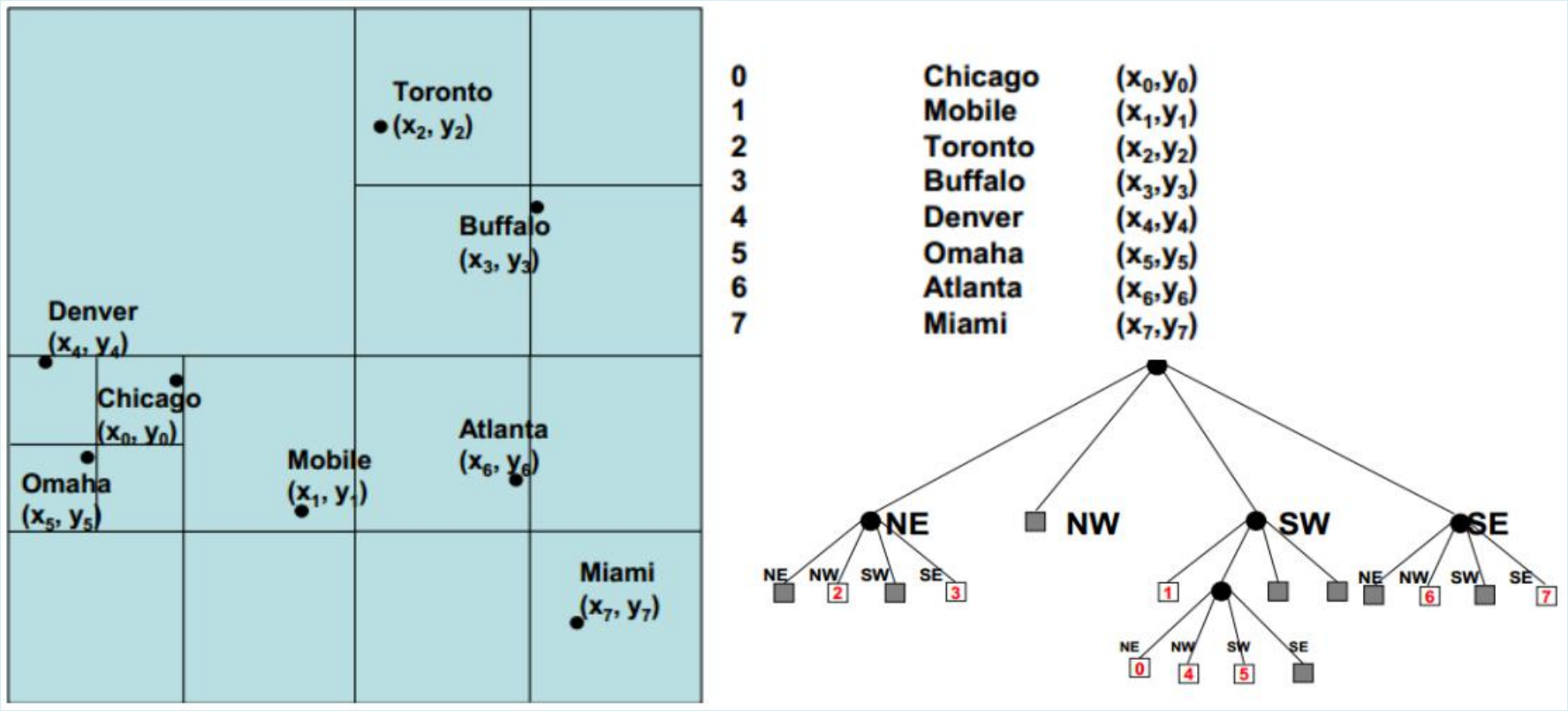
点-四叉树（Point Quadtree）

- 树节点为数据点，每个节点有四个子节点
- 每个子节点进行递归划分



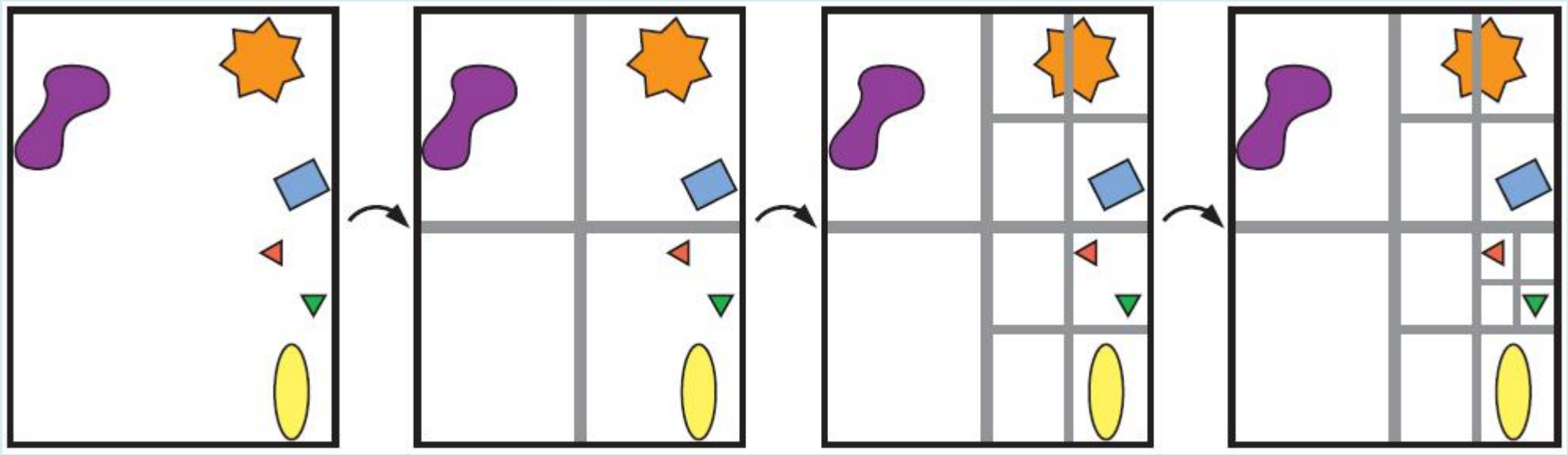
区域-四叉树（Region Quadtree）

- 树节点表示空间区域分区，数据点保存在非空叶子节点中。



四叉树的构造

- 选取初始区域作为根节点，根据定义进行递归构造



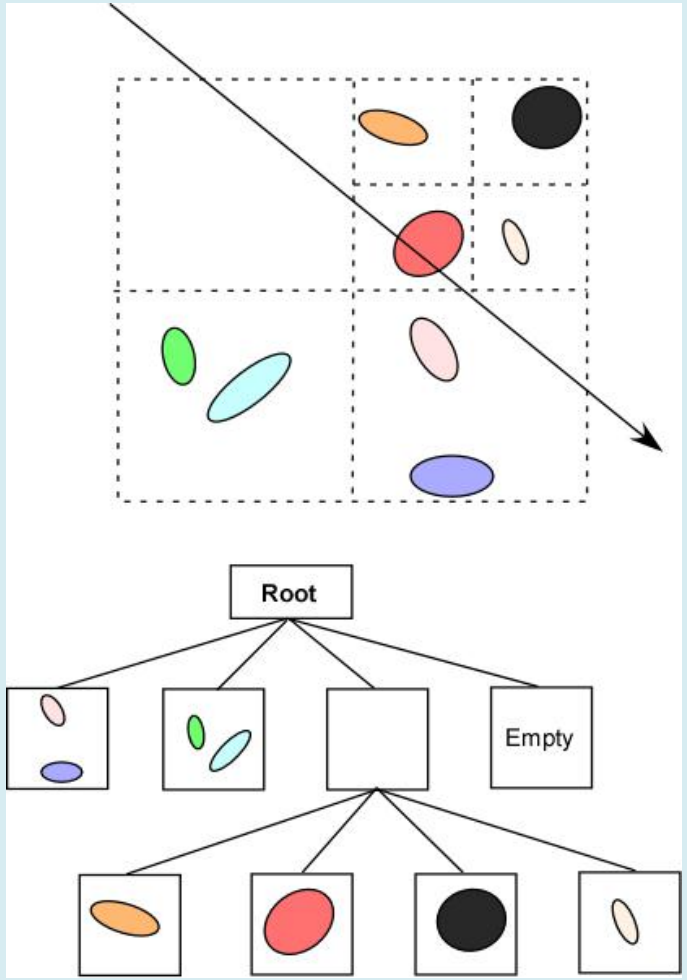
四叉树的遍历

- 从根节点开始，如果当前查询位置在当前节点区域中，继续遍历其子节点区域，否则进行剪枝；最终在叶子节点中完成查找。

- 例子：光线求交，地形渲染

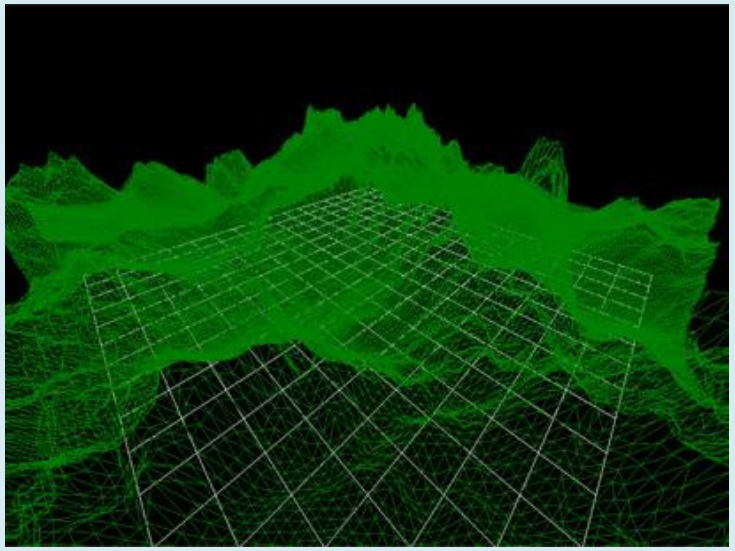
例子：使用四叉树加速光线求交

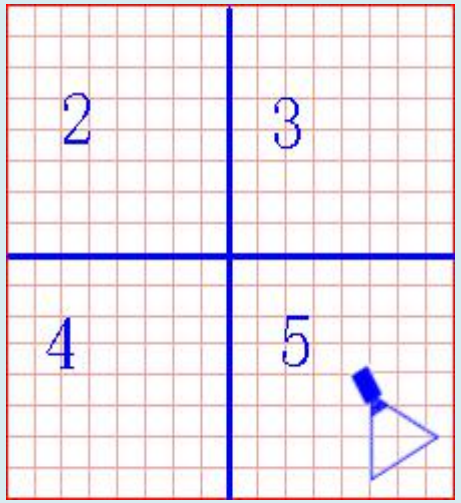
- 找出场景中与给定光线相交的所有物体
  - 遍历所有物体进行求交测试，效率低
- 使用四叉树事先对场景进行组织
- 遍历四叉树，如果当前区域中没有物体或不与光线相交，剪枝

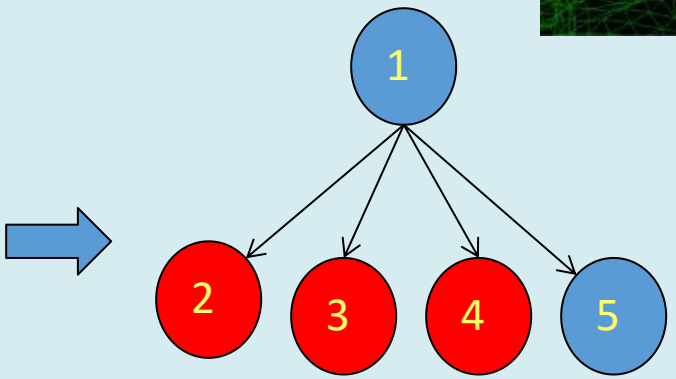


例子：使用四叉树进行地形渲染

- terrain地形，二维平面上的高度数据，一个二维数组
- 只渲染当前可见区域的地形
  - 遍历二维数组
  - 预先构建四叉树，遍历四叉树







四叉树的应用

- 2维数据组织与查找
  - http://www.mikechambers.com/files/html5/javascript/QuadTree/examples/retrieve.html
- 碰撞检测
  - http://www.mikechambers.com/files/html5/javascript/QuadTree/examples/collision.html
- 图像表示/压缩
  - http://cn.mathworks.com/help/images/ref/qtdecomp.html

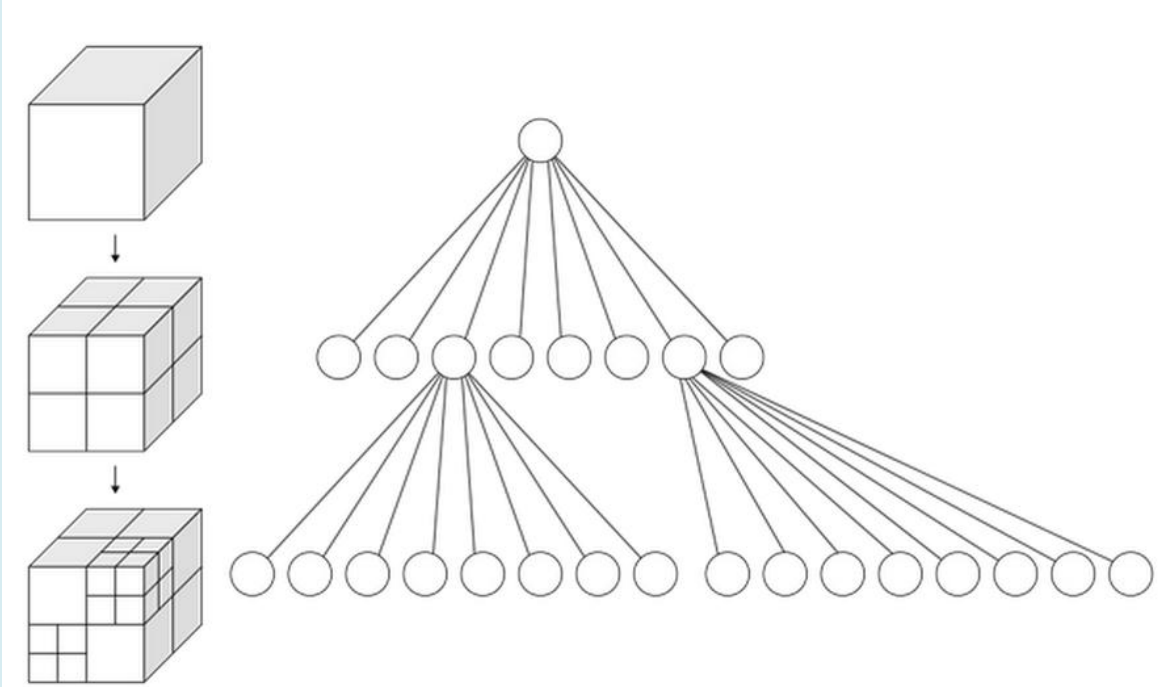
内容

- 四叉树（Quadtree）
- **八叉树（Octree）**

- 二叉空间划分树（BSP tree）
- KD 树（K Dimensional tree）

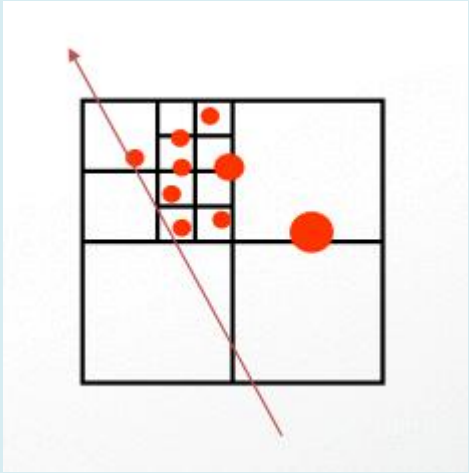
八叉树定义

- 四叉树的三维空间推广
- 原理与四叉树相同
- 把一个立方体空间分割为八个小立方体，然后递归分割小立方体



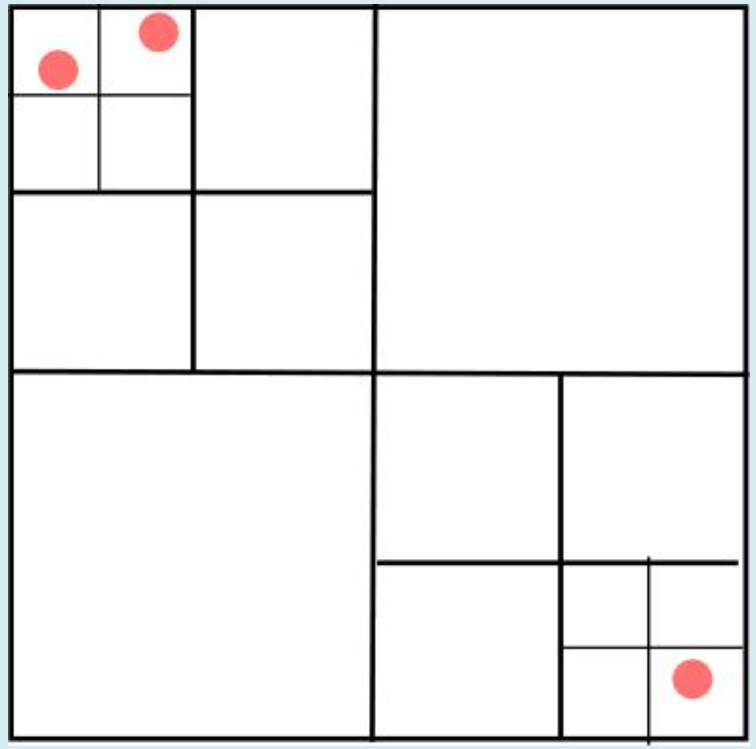
八叉树应用

- 三维数据组织与查找
- 碰撞检测
- 光线跟踪
- 颜色量化



四叉树、八叉树的局限性

- 当数据分布不均匀时，生成的树会十分不平衡（unbalanced）
  - 大部分节点中没有数据



内容

- 四叉树（Quadtree）
- 八叉树（Octree）

- **二叉空间划分树（BSP tree）**
- KD 树（K Dimensional tree）

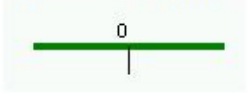
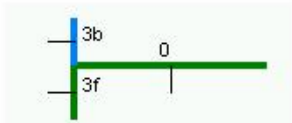
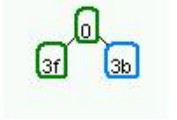
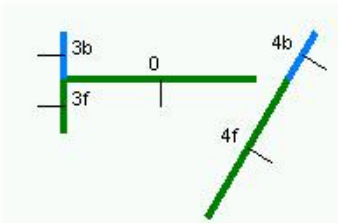
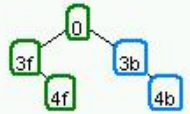
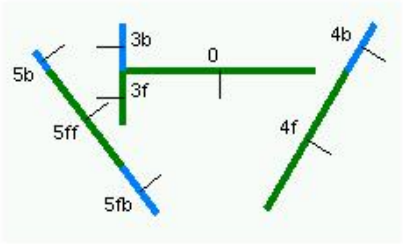
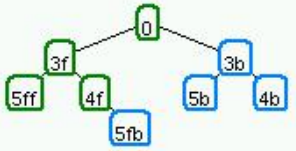
二叉空间划分树（BSP tree）

- Binary Space Partition Tree
- BSP tree 是一棵二叉树，每个节点表示一个有向超平面，其将当前空间划分为前向（front）和背向（back）两个子空间，分别对应当前节点的左子树和右子树
- 对子空间进行递归划分

BSP 树的构造

- 从空树开始，每次选择一个面片作为节点插入树中
- 每次插入一个新节点，从树的根节点开始遍历
  - 如果新节点面片与当前结点片面相交，将新面片分割成两个面片
  - 新节点在当前节点前向空间，插入左子树
  - 新节点在当前节点背向空间，插入右子树
  - 当前节点为空，直接插入新节点
- 直到所有面片都被插入树中

BSP 树的构造

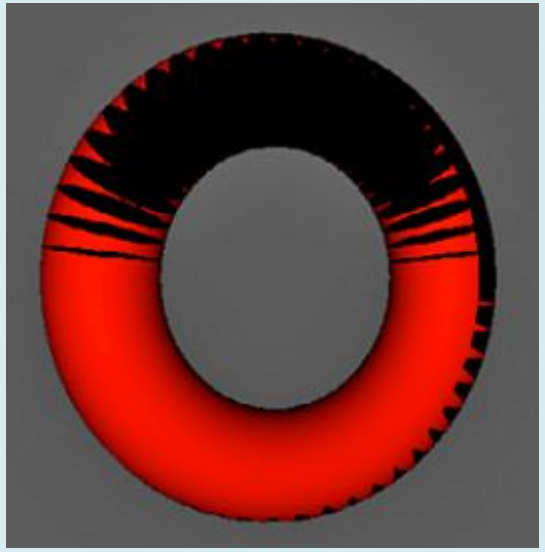
First, create a root node and partition plane. 	Obviously the root does not have any children. 
	
	
	

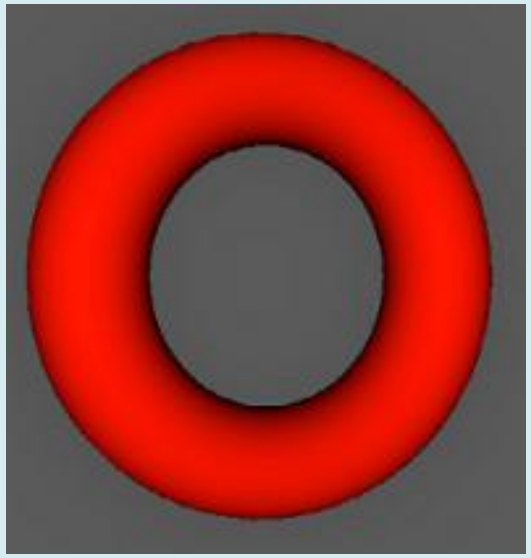
BSP 树的遍历

- 从根节点开始，判断输入位置与当前分割平面的“前”、“后”关系，“前”则遍历左子树，“后”则遍历右子树，递归到叶子节点终止。
- 如何判断前后
  - 平面方程 $Ax + By + Cz + D = 0$，可用 $D(x_0, y_0, z_0) = Ax_0 + By_0 + Cz_0 + D$ 判别
    - $D > 0$：在平面前面
    - $D = 0$：在平面上
    - $D < 0$：在平面后面
- 应用：遮挡判定，可见物筛选

例子：画家算法（Painter's Algorithm）

- 按照离开视点从远到近的顺序绘制多边形
- 首先绘制最远的，依次靠近，避免出现不可见面遮挡问题





BSP树实现画家算法

- 事先构建场景面片的 BSP 树，进行从后向前遍历（Back to Front Traversal）

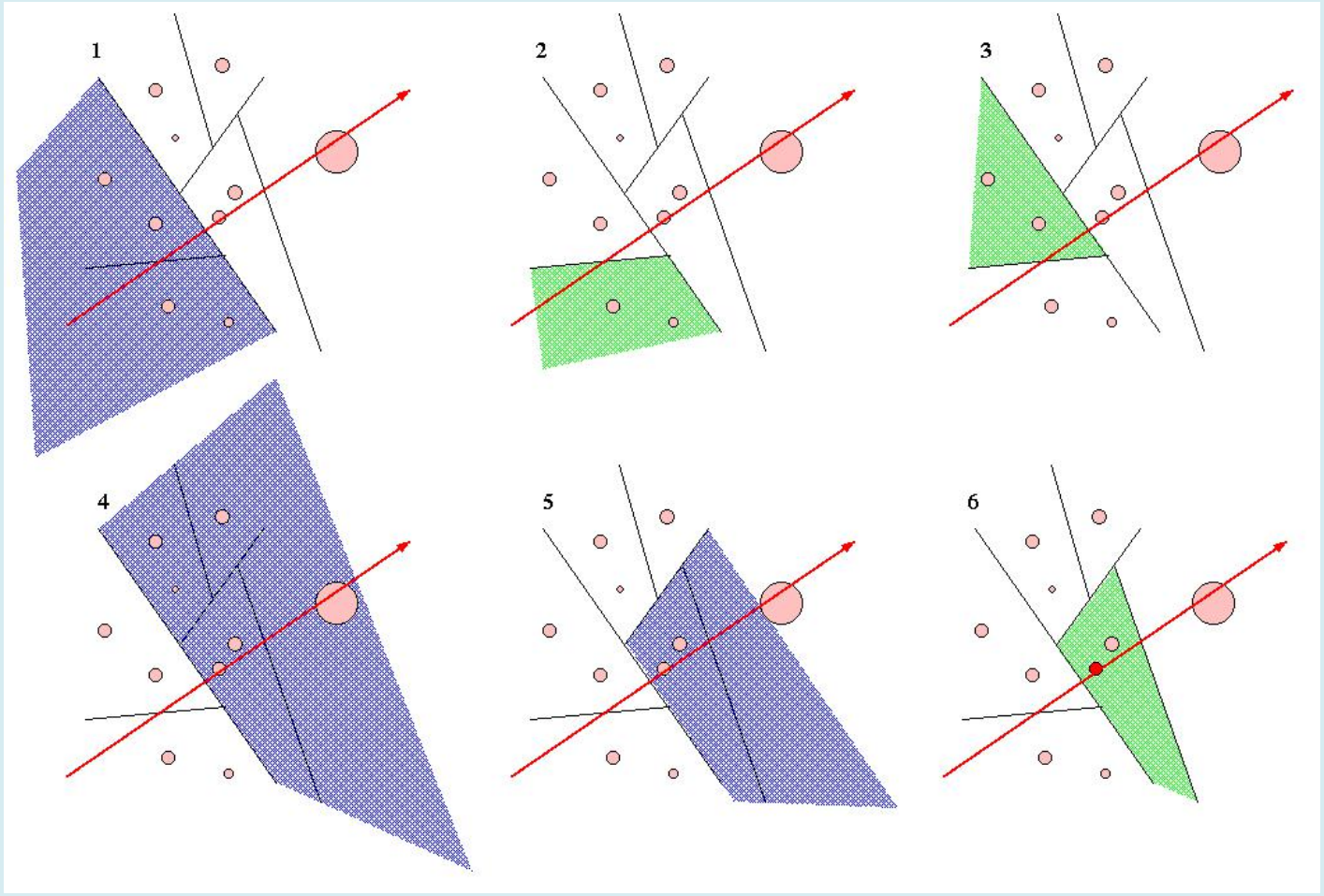
```
void traverse_tree(bsp_tree* tree, point eye)
{
    location = tree->find_location(eye);

    if (tree->empty())
        return;

    if (location > 0)  // if eye in front of location
    {
        traverse_tree(tree->back, eye);
        display(tree->polygon_list);
        traverse_tree(tree->front, eye);
    }
    else if (location < 0) // eye behind location
    {
        traverse_tree(tree->front, eye);
        display(tree->polygon_list);
        traverse_tree(tree->back, eye);
    }
    else  // eye coincidental with partition hyperplane
    {
        traverse_tree(tree->front, eye);
        traverse_tree(tree->back, eye);
    }
}
```

例子：使用 BSP 树进行光线求交

- 使用 BSP 树事先对场景进行组织
- 遍历 BSP 树，如果当前区域中没有物体或不与光线相交，剪枝



BSP 树的应用

- 三维室内场景组织
  - DOOM，QUAKE

- 构造实体几何（Constructive Solid Geometry）
  - http://evanw.github.io/csg.js/more.html

内容

- 四叉树（Quadtree）
- 八叉树（Octree）

- 二叉空间划分树（BSP tree）
- **KD 树（K Dimensional tree）**

KD 树

- K-Dimensional Tree
- 用于 K 维空间中数据点的组织

- 一种特殊的 BSP 树
- 其每个节点是 K 维空间中的一个点，每个非叶节点隐式定义了一个分割超平面，左右子树对应不同的半空间
- KD 树每个节点处的分割超平面垂直与 K 维空间的一条坐标轴

KD 树的构造

- 超平面的选择：每个节点对应 K 维中的一维（即一个坐标轴），该节点处的超平面与该坐标轴方向正交；

- 例如，K=2，某个节点对应了 x 轴，则所有比当前节点的 x 值小的点被划分到左子树，比 x 值大的划分到右子树；假设当前节点的 x 值为 x_0，则对应的超平面为 $x = x_0$ ；

- 如何为每个节点选择维度？方法不唯一。

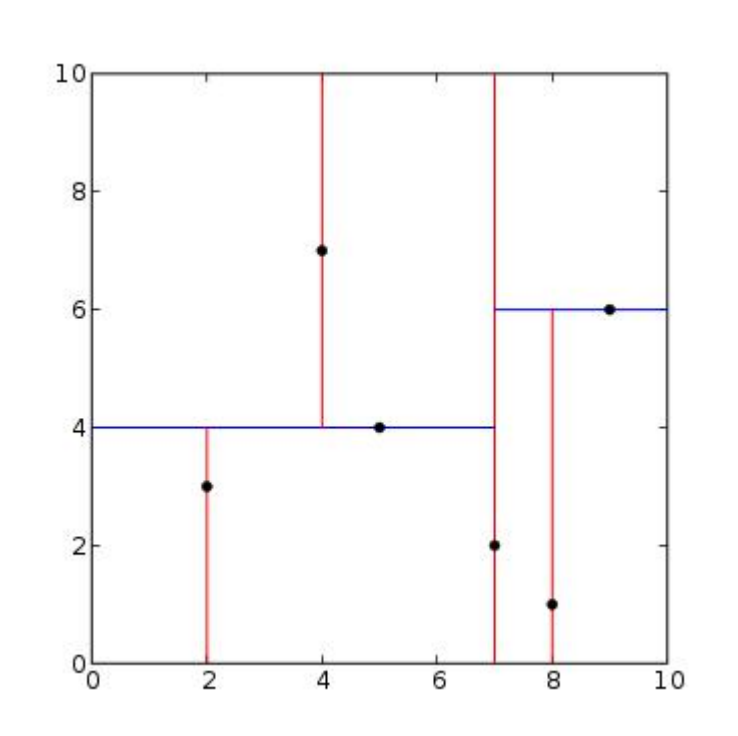
- 一种标准的分割方法：将 K 维空间维度编号为 {0, 1, ..., K-1}，按树的每一层循环选取，即如当前层节点对应 i 维，则下一层节点对应 (i+1) % K 维

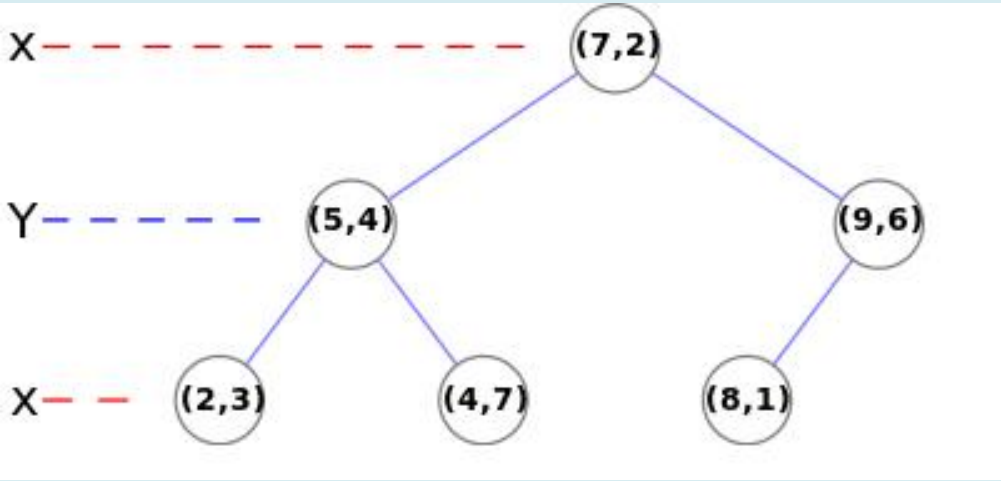
KD 树的构造--例子

- 二维数据点：[(2,3), (5,4), (9,6), (4,7), (8,1), (7,2)]
- KD 树构造除了依赖于维度选取，还依赖于数据点选取
- 本例中维度使用循环选取方式，每次选取当前维度下的中值（median）数据点（也可以使用平均值）

- 第一次选 x 轴，所有数据点的 x 坐标为 [2, 4, 5, **7**, 8, 9]，选 (7, 2)
- 第二次选 y 轴，(7, 2) 左子空间中数据点的 y 坐标为 [3, **4**, 7]，选(5, 4)
- ……

KD 树的构造--例子





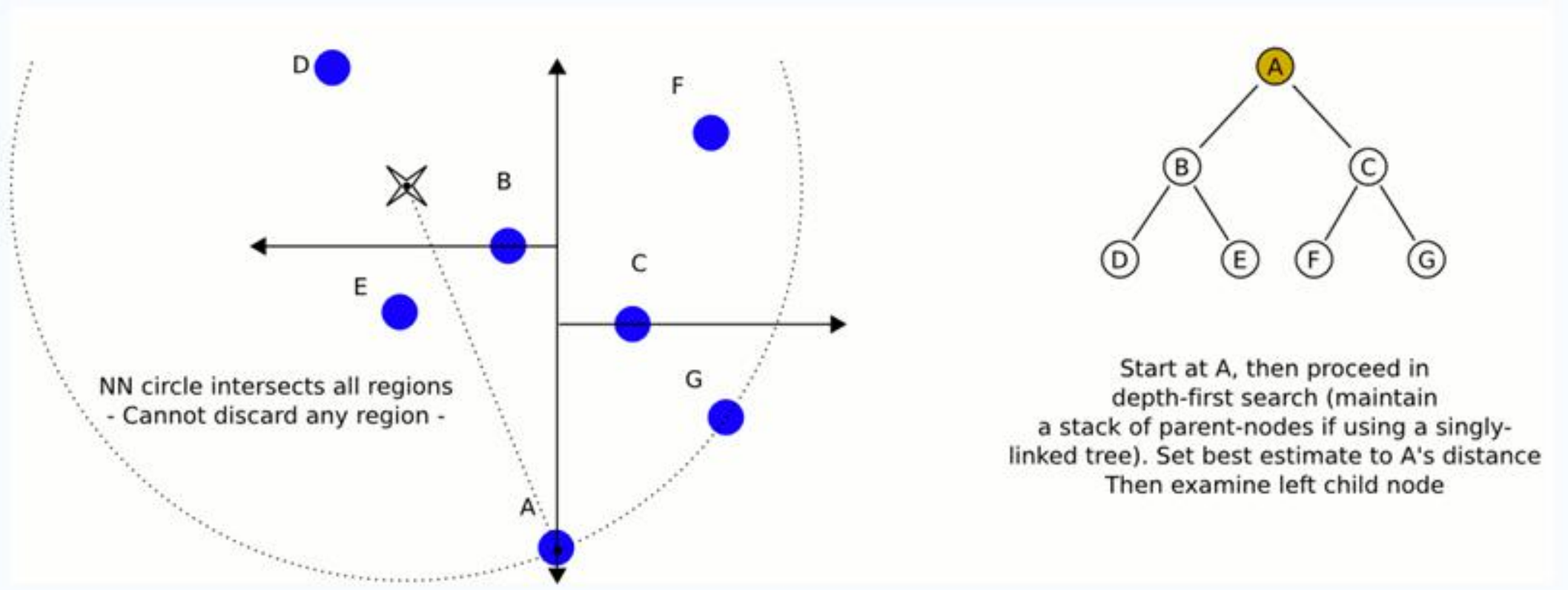
KD 树的遍历

- 同 BSP 树的遍历，判断输入位置与当前分割平面的关系向相应的子树中进行遍历

- 例子，最近邻查找

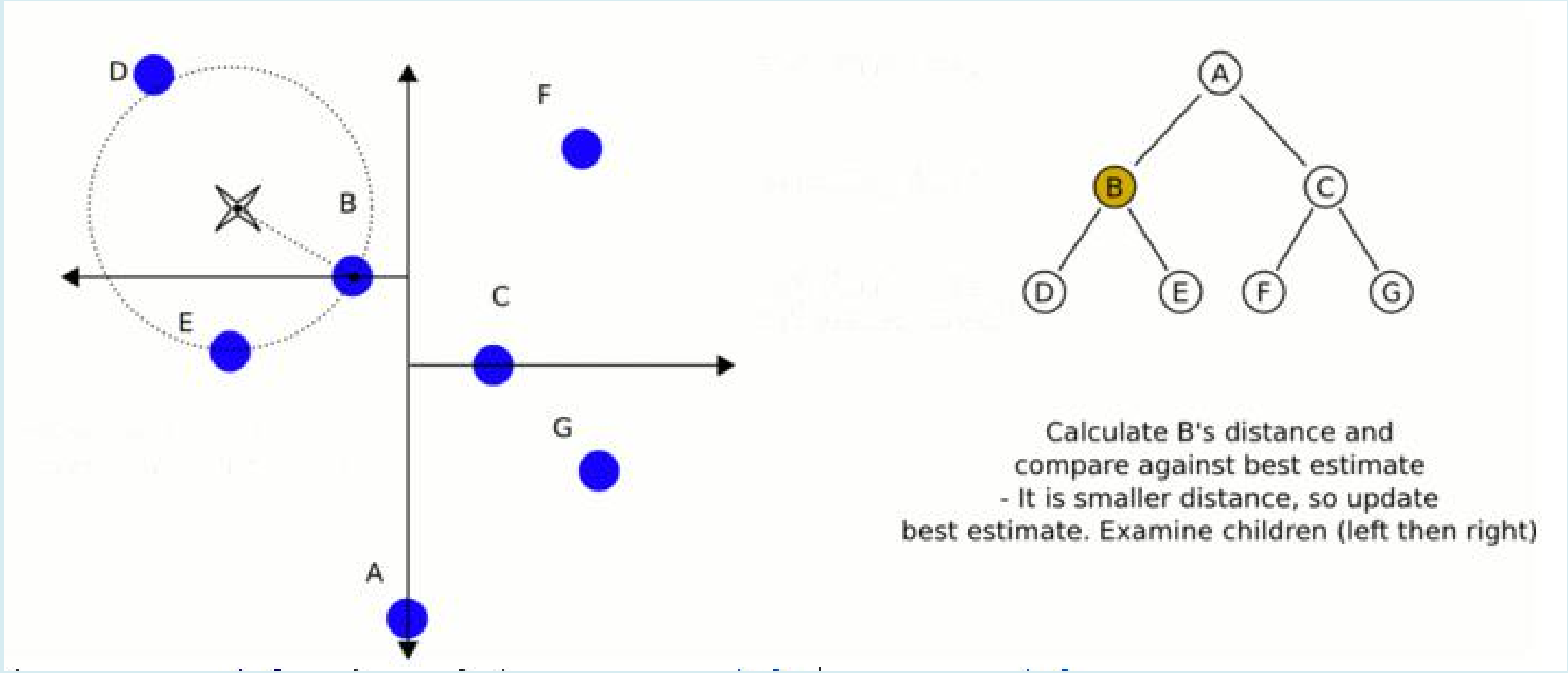
例子：使用 KD 树实现最近邻查找

- 从一个点集中找出距离输入位置最近的点
- 从根节点开始进行深度优先遍历

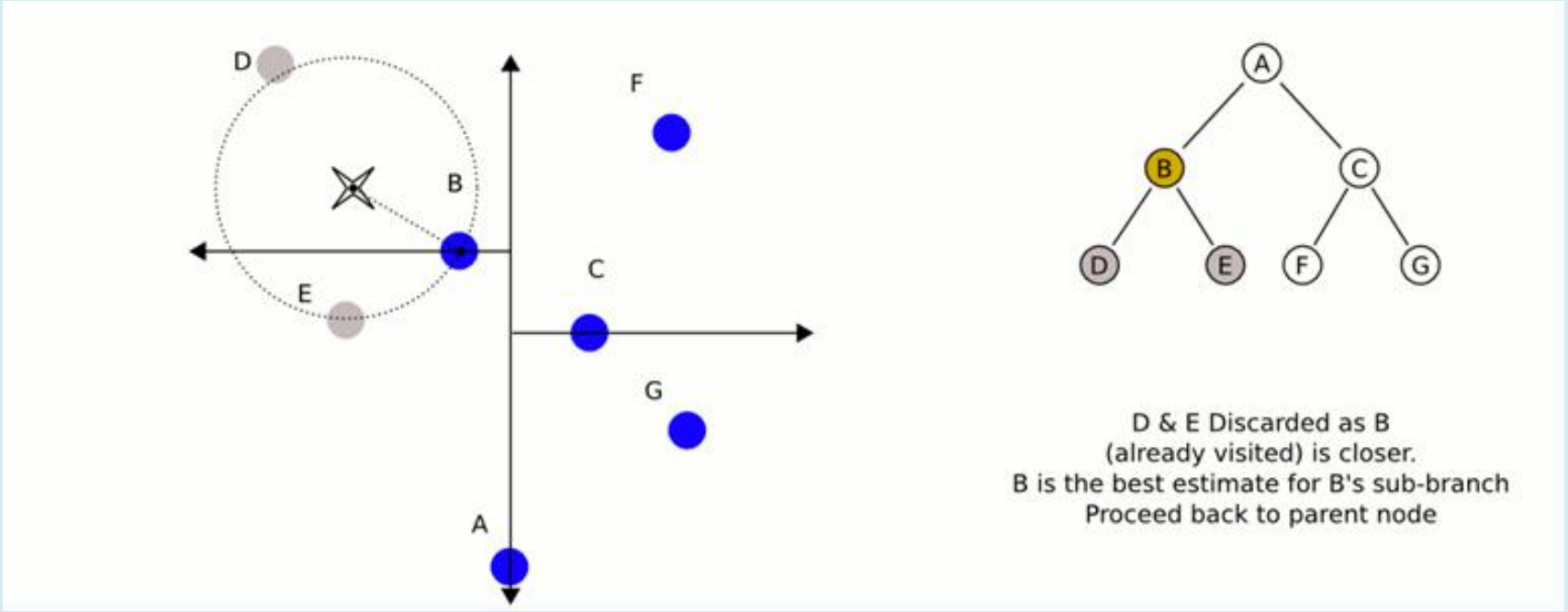


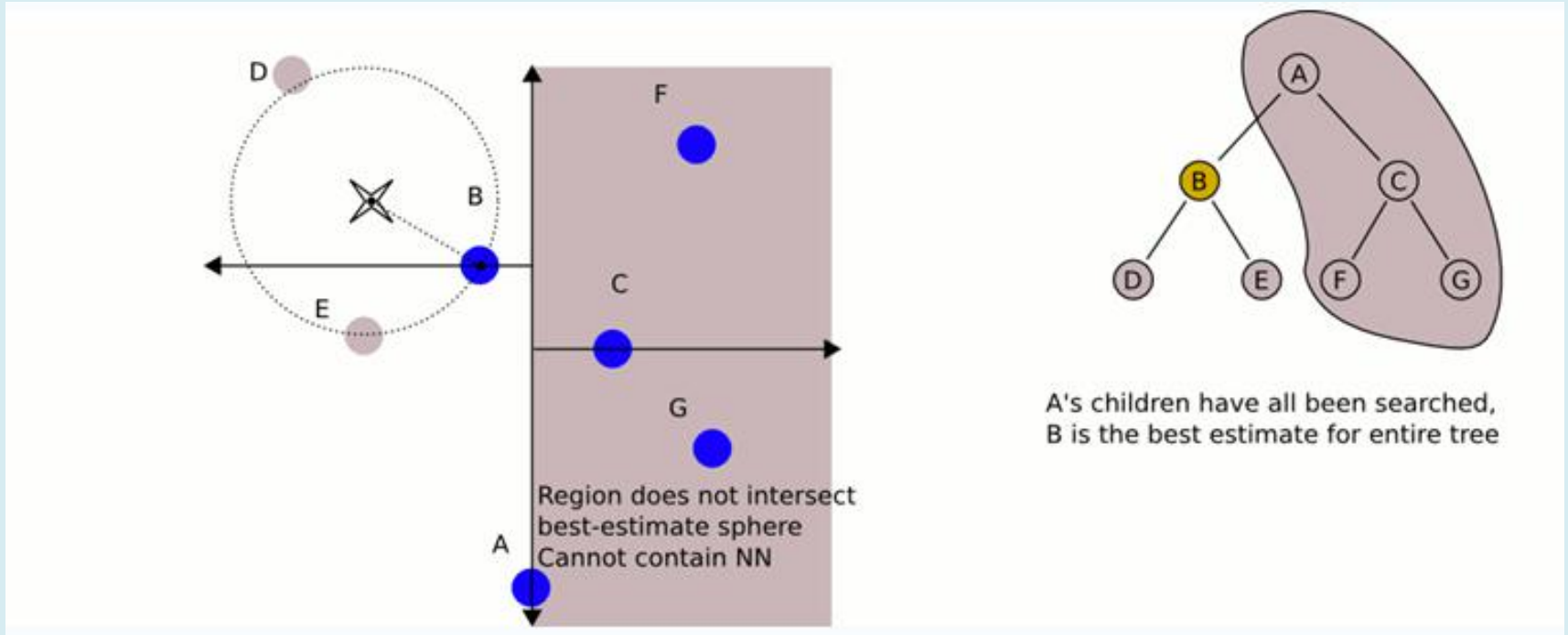
例子：使用 KD 树实现最近邻查找

- 对每个节点，计算输入位置与当前节点的距离，更新最短距离



- 如果当前节点所在空间与当前最近点所在空间不相交，剪枝





KD 树的应用

- 最近邻查找，e.g. kNN(k Nearest Neighbors)
  - 图像特征匹配，三维点云配准

- 三维游戏场景管理理
  - KD 树基于数据点建树，当数据量大时开销过大，一般不单独用于场景管理
  - 与八叉树结合，使用八叉树进行大粒度的划分和查找，使用 KD 树进行邻域划分和查找

KD 树的应用演示

- https://github.com/ubilabs/kd-tree-javascript

总结--几种空间划分树结构的比较

技术名称	适用场景	应用
四叉树	二维空间，或基于高度场的地形	二维场景管理，地形绘制、地图导航
八叉树	大规模三维场景	三维场景管理与渲染
BSP 树	二维、三维室内场景	碰撞检测、光线跟踪（由于现代显卡均支持硬件 Z buffer，基本不用 BSP 树进行场景管理）
KD 树	高维空间数据查询，邻域查询	辅助其他场景管理方法进行邻域查询

参考资料

- http://www.hao-li.com/cs420-fs2015/slides/Lecture10.1.pdf
- https://www.cs.ucf.edu/~dcm/Teaching/COT4810-Spring2011/Presentations/JonLeonardSpacePartitioningDataStructures.pdf
- https://www.cs.utexas.edu/~ckm/teaching/cs354_f11/lectures/Lecture21.pdf
- http://web.eecs.utk.edu/~cphillip/cs594_spring2014/quadtree-Allan.pdf
- http://ccftp.scu.edu.cn:8090/Download/5eb3de6d-2514-44ac-b1c1-eba08bb864a4.pptx

谢谢!

Email: yilin.gui@gmail.com