aoi 比较

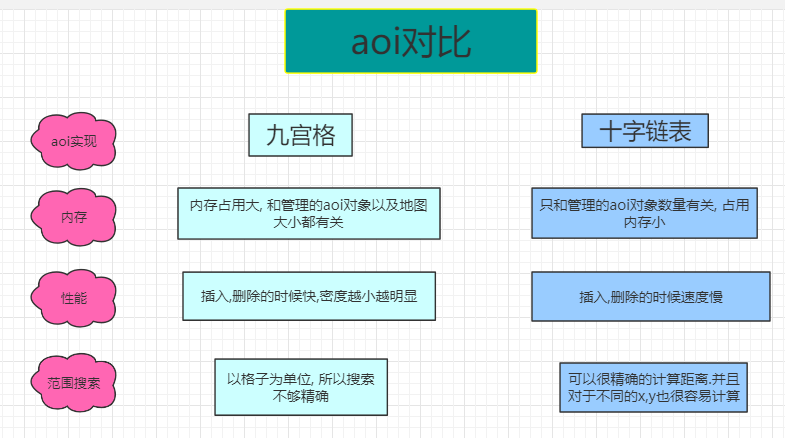
##### aoi 比较

所谓 AOI ( Area Of Interest ) ，就是只需要关心自己感兴趣范围内的变化(xx加入,xx消失,xx 移动)

因为一张地图非常大, 会有很多自己不需要关心的对象. 如果不用区域划分, 管理这些对象的成本会非常高

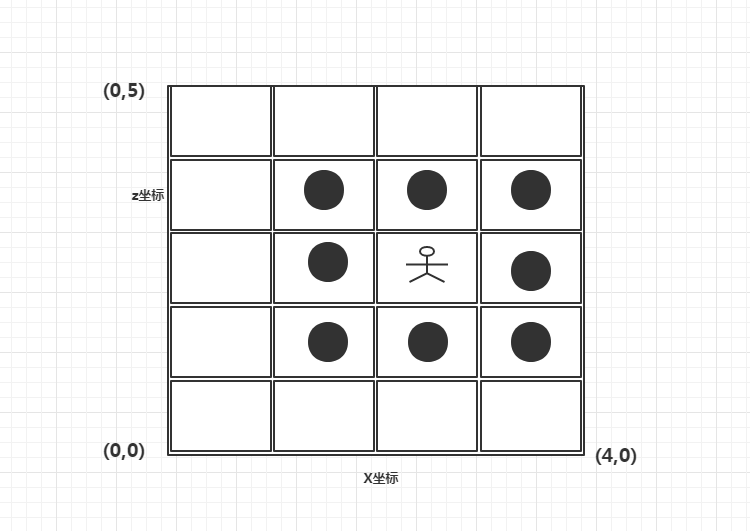
目前有两种划分方法. **网格和十字链表**

十字链表适合不同的xy. 并且距离判定比较精准. 但是插入,删除效率不如九宫格.(见最后面的测试数据)

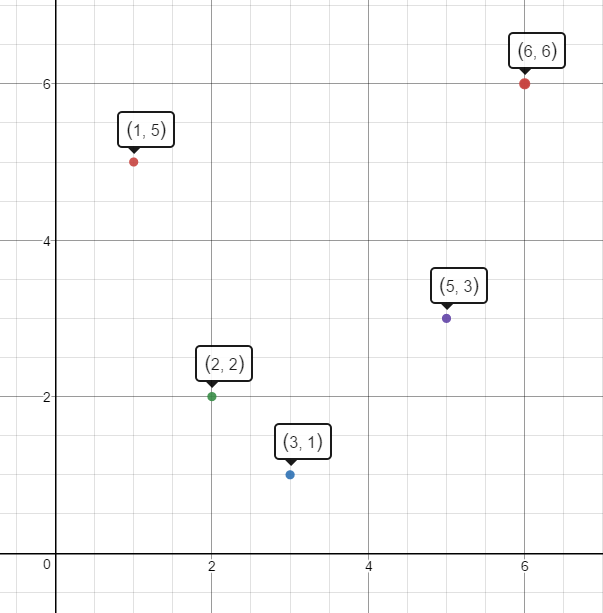


九宫格占用内存大，因为每个格子都有一个兴趣列表，大地图格子多，人很少的情况下就会造成大量的浪费。 十字链表占用内存少，有多少人，链表就多长。

网格(灯塔): 将地图划分成多个等分的, 某种形状的格子(目前一般是矩形 , 也可以是多边形). 节点会落在其中的某个格子上. 每个格子称为一个灯塔.上面会有集合保存格子上面的对象标识(label)



十字链表: 在x轴y轴上, 使用单链表或者双向链表来将各个节点有序的串联起来



上面这张图会被表示成下面的链表

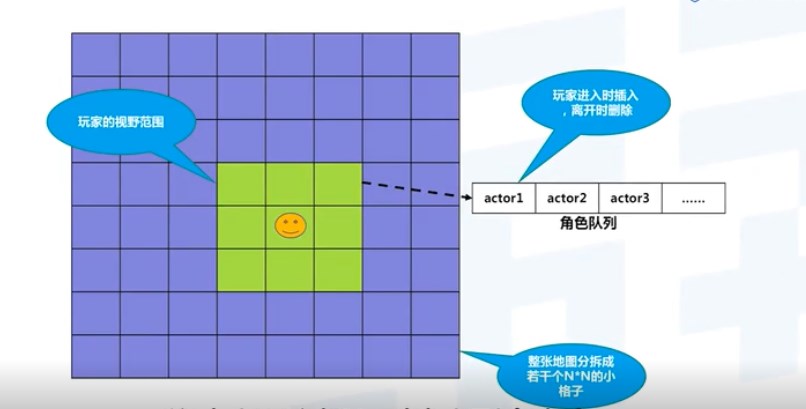
x轴 xHead->1->2->3->5->6->xTail(反向也有个链接, 这里没画)

y轴(一般在左上角) yHead->6->5->3->2->1-> yTail (反向也有个链接, 这里没画)

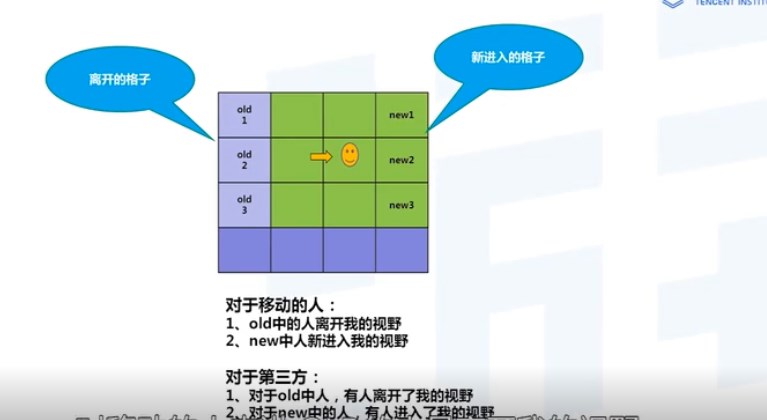
实现:

九宫格 :

1. 添加, 删除

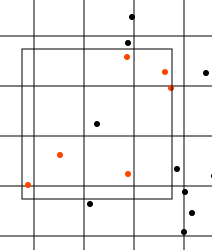
* 
* 某个对象A被添加进aoi的某个格子(T) 的时候(可能是直接落在, 也可能是从别的格子移动到某个 首先会得到格子T(以下称为Tower). 然后获取从X(Tx -1, Tx+1), Y(Ty-1, Ty+1)的最多九个格子中的所有对象集合Ps. 遍历Ps, 将A和Ps集合中的每个对象P, 都建立起某种关系(AddRelation)
* 删除是同理的. 某个对象A被从Aoi移除的时候(从场景移除, 或者离开某个T) . 遍历Ps, 将A和Ps集合中的每个对象P, 都移除某种关系(RemoveRelation)

1. 移动更新

* 
* 见上图. (左上角为原点) 当对象A从(2, 2) 移动到(3,2)的时候. 对于old格子(Told)里面的所有对象Ps来说, 双方会进行移除关系操作(RemoveRelation), 对于new格子(Tnew)里面所有的对象Ps来说. 双方会进行添加关系操作(AddRelation). 同时, 对于 old和new中间的绿色格子所有对象来说, 他们会看到A从(2,2)移动到(3,2)

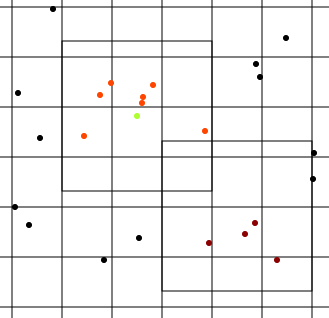
十字链表: (这里只做了基本的实现, 没有优化. )

通用方法, 记为 F 从某个对象A的位置 Ax, AY起, 向x轴的前后, y轴的上下遍历每个节点P, 将满足 abs(Px, Ax) <= xRange && abs(Py, Ay) <= yRange的点P, 加入到集合Set中.

1. 添加, 删除
2. 

* 某个对象A被添加进aoi的时候. 分别从xHead和YHead开始, 依次比较x,y的大小, 将A对象插入合适的链表位置. 然后对A进行F方法的调用. 得到集合Set. 对set中的每个对象P, 和A建立关系(AddRelation),或者删除关系(RemoveRelation)

1. 移动更新

* 
* 绿色的是移动的点. 褐色的是移动之前的广播集合setOld, 橙色是移动之后的广播集合SetNew
* 当某个对象A从old移动到new的时候.(比如上面的3,3 移动到4,4) . 使用F函数对 old, new分别计算得到 SetOld, SetNew.
* 将SetNew-SetOld得到移动通知集合SetDiff. 将SetNew-SetDiff得到新增集合SetAdd, 将SetOld-SetDiff得到移除集合SetRemove
* 分别对三个集合中的对象, 和A进行相应关系的处理

测试数据得到的结论.

添加删除: **密度(格子数/节点数)越大, 双方越接近. 密度越小, 九宫格算法越有优势**

移动: 效率差不多

部分代码 <aoi 部分代码.html>

测试数据 <aoi测试数据.xlsx>

部分类图

