aoi 部分代码.md

#### aoi 部分代码

九宫格 添加 删除:

void onAddNode(IAoi aoi, CellNode node)
  
{
  
 CellAoi cellAoi = CellAoi.class.cast(aoi);
  
   
 int xCell = node.x / cellAoi.cellSize;
  
 int yCell = node.y / cellAoi.cellSize;
  
 ...
  
   
 for (int i = left; i <= right; i++)
  
 {
  
 for (int j = upper; j <= down; j++)
  
 {
  
 Tower tower = cellAoi.towers[i][j];
  
 if (tower != null)
  
 {
  
 Set<Long> set = tower.set;
  
 if (set.size() > 0)
  
 {
  
 for (Long aid : set)
  
 {
  
 if (aid != node.label)
  
 {
  
 //互相广播 进入视野, 并建立联系
  
 node.addRelation(cellAoi.getCellNode(aid));
  
 }
  
 }
  
 }
  
 }
  
 }
  
 }
  
}

移动:

void onMoveTo(IAoi aoi, int x, int y, int newX, int newY)
  
{
  
 CellAoi cellAoi = CellAoi.class.cast(aoi);
  
 final int xCell = newX / cellAoi.cellSize;
  
 final int yCell = newY / cellAoi.cellSize;
  
   
 final int \_xCell = towerX;
  
 final int \_yCell = towerY;
  
   
 for (int i = left; i <= right; i++)
  
 {
  
 for (int j = upper; j <= down; j++)
  
 {
  
 cellAoi.marks[i][j] = true;
  
 }
  
 }
  
   
 for (int i = \_left; i <= \_right; i++)
  
 {
  
 for (int j = \_upper; j <= \_down; j++)
  
 {
  
 if (cellAoi.marks[i][j])
  
 {
  
 cellAoi.marks[i][j] = false;
  
 // 广播移动
  
 Tower tower = cellAoi.towers[i][j];
  
 if (tower != null)
  
 {
  
 Set<Long> set = tower.set;
  
 if (!set.isEmpty())
  
 {
  
 for (Long aid : set)
  
 {
  
 if (aid != label)
  
 {
  
 this.onMoveBroad(cellAoi.getCellNode(aid));
  
 }
  
 }
  
 }
  
 }
  
 }
  
 else
  
 {
  
 // 广播移除
  
 Tower tower = cellAoi.towers[i][j];
  
 if (tower != null)
  
 {
  
 Set<Long> set = tower.set;
  
 if (!set.isEmpty())
  
 {
  
 for (Long aid : set)
  
 {
  
 if (aid != label)
  
 {
  
 // 告诉对方移除自己, 并删除双方联系
  
 CellNode cellNode = cellAoi.getCellNode(aid);
  
 if (cellNode != null)
  
 {
  
 cellNode.removeRelation(this);
  
 }
  
 }
  
   
 }
  
 }
  
 }
  
 }
  
 }
  
 }
  
   
 for (int i = left; i <= right; i++)
  
 {
  
 for (int j = upper; j <= down; j++)
  
 {
  
 if (cellAoi.marks[i][j])
  
 {
  
 cellAoi.marks[i][j] = false;
  
 // 广播添加
  
 Tower tower = cellAoi.towers[i][j];
  
 if (tower != null)
  
 {
  
 Set<Long> set = tower.set;
  
 if (set.size() > 0)
  
 {
  
 for (Long aid : set)
  
 {
  
 if (aid != label)
  
 {
  
 //互相广播 进入视野, 并建立联系
  
 this.addRelation(cellAoi.getCellNode(aid));
  
 }
  
 }
  
 }
  
 }
  
 }
  
 }
  
 }
  
}

十字链表

class CrossLinkNode extends BaseNode
  
{
  
 // x 轴
  
 public CrossLinkNode xPrev;
  
 public CrossLinkNode xNext;
  
   
 // y 轴
  
 public CrossLinkNode yPrev;
  
 public CrossLinkNode yNext;
  
}

/\*\*
  
 \* 插入node
  
 \*
  
 \* @param prev
  
 \* @param insert
  
 \* @param isX
  
 \*/
  
private void insertDoubleLink(CrossLinkNode prev, CrossLinkNode insert, boolean isX)
  
{
  
 if (isX)
  
 {
  
 CrossLinkNode xNext = prev.xNext;
  
   
 insert.xNext = xNext;
  
 insert.xPrev = prev;
  
   
 prev.xNext = insert;
  
 xNext.xPrev = insert;
  
 }
  
 else
  
 {
  
 CrossLinkNode yNext = prev.yNext;
  
   
 insert.yNext = yNext;
  
 insert.yPrev = prev;
  
   
 prev.yNext = insert;
  
 yNext.yPrev = insert;
  
 }
  
}

// 查找广播范围
  
void chooseBroadSet(CrossLinkNode node, Set<CrossLinkNode> set)
  
{
  
 //x 轴 往前
  
 for (CrossLinkNode cur = node.xPrev; cur != CrossAoi.xHead; cur = cur.xPrev)
  
 {
  
 if (Math.abs(node.x - cur.x) <= searchXRange)
  
 {
  
 if (Math.abs(node.y - cur.y) <= searchYRange)
  
 {
  
 set.add(cur);
  
 }
  
 }
  
 else
  
 {
  
 break;
  
 }
  
 }
  
   
 // x 轴 往后
  
 for (CrossLinkNode cur = node.xNext; cur != CrossAoi.xTail; cur = cur.xNext)
  
 {
  
 if (Math.abs(node.x - cur.x) <= searchXRange)
  
 {
  
 if (Math.abs(node.y - cur.y) <= searchYRange)
  
 {
  
 set.add(cur);
  
 }
  
 }
  
 else
  
 {
  
 break;
  
 }
  
 }
  
}

添加删除

void addNode(CrossLinkNode node)
  
{
  
 // 插入x
  
 int x = node.x;
  
 int y = node.y;
  
   
 for (CrossLinkNode cur = xHead; cur != xTail; cur = cur.xNext)
  
 {
  
 CrossLinkNode xNext = cur.xNext;
  
 if (x > cur.x && x <= xNext.x)
  
 {
  
 insertDoubleLink(cur, node, true);
  
 break;
  
 }
  
 }
  
   
 //插入y
  
 for (CrossLinkNode cur = yHead; cur != yTail; cur = cur.yNext)
  
 {
  
 CrossLinkNode yNext = cur.yNext;
  
 if (y > cur.y && y <= yNext.y)
  
 {
  
 insertDoubleLink(cur, node, false);
  
 break;
  
 }
  
 }
  
}

void onAddNode(IAoi aoi, CrossLinkNode node)
  
{
  
 chooseBroadSet(node, set);
  
   
 if (!set.isEmpty())
  
 {
  
 for (Iterator<CrossLinkNode> iterator = set.iterator(); iterator.hasNext(); )
  
 {
  
 CrossLinkNode baseNode = iterator.next();
  
 node.addRelation(baseNode);
  
 }
  
 }
  
}

移动:

void moveNode(CellNode node, int x, int y)
  
{
  
 final int \_x = node.x;
  
 final int \_y = node.y;
  
   
 onTriggerBeforeMoveToListener(this, node, x, y);
  
   
 node.moveTo(this, x, y);
  
   
 onTriggerAfterMoveToListener(this, node, \_x, \_y);
  
}

/\*\*
  
 \* 新增节点(根据相对距离查找) 适合移动的节点
  
 \*
  
 \* @param left 向左的差值
  
 \* @param upper 向上的差值
  
 \* @param node
  
 \*/
  
public void addNode(CrossLinkNode node, int left, int upper)
  
{
  
 // 插入x
  
 int x = node.x;
  
 int y = node.y;
  
   
 if (left != 0)
  
 {
  
 if (left < 0)
  
 {
  
 for (CrossLinkNode cur = node.xNext; cur != xHead; cur = cur.xPrev)
  
 {
  
 CrossLinkNode xPrev = cur.xPrev;
  
 if (x < cur.x && x >= xPrev.x)
  
 {
  
 insertDoubleLink(cur, node, true);
  
 break;
  
 }
  
 }
  
 }
  
 else
  
 {
  
 for (CrossLinkNode cur = node.xPrev; cur != xTail; cur = cur.xNext)
  
 {
  
 CrossLinkNode xNext = cur.xNext;
  
 if (x > cur.x && x <= xNext.x)
  
 {
  
 insertDoubleLink(cur, node, true);
  
 break;
  
 }
  
 }
  
 }
  
 }
  
   
 if (upper != 0)
  
 {
  
 if (upper < 0)
  
 {
  
 for (CrossLinkNode cur = node.yNext; cur != yHead; cur = cur.yPrev)
  
 {
  
 CrossLinkNode yPrev = cur.yPrev;
  
 if (y < cur.y && y >= yPrev.y)
  
 {
  
 insertDoubleLink(cur, node, false);
  
 break;
  
 }
  
 }
  
 }
  
 else
  
 {
  
 //插入y
  
 for (CrossLinkNode cur = node.yPrev; cur != yTail; cur = cur.yNext)
  
 {
  
 CrossLinkNode yNext = cur.yNext;
  
 if (y > cur.y && y <= yNext.y)
  
 {
  
 insertDoubleLink(cur, node, false);
  
 break;
  
 }
  
 }
  
 }
  
 }
  
   
 nodes.put(node.label, node);
  
}

void onMoveTo(IAoi aoi, int x, int y, int newX, int newY)
  
{
  
 // 移除
  
 CrossAoi crossAoi = CrossAoi.class.cast(aoi);
  
 crossAoi.makeDoubleLink(xPrev, xNext, true);
  
 crossAoi.makeDoubleLink(yPrev, yNext, false);
  
   
 // 添加
  
 crossAoi.addNode(this, newX - x, newY - y);
  
}

void beforeMoveTo(IAoi aoi, CrossLinkNode node, int x, int y)
  
{
  
 chooseBroadSet(node, set);
  
}

public void afterMoveTo(IAoi aoi, CrossLinkNode node, int fromX, int fromY)
  
{
  
 chooseBroadSet(node, setBack);
  
   
 // move = (setBack && set) = move broadcast
  
 Sets.SetView<CrossLinkNode> move = Sets.intersection(setBack, set);
  
 for (CrossLinkNode baseNode : move)
  
 {
  
 node.onMoveBroad(baseNode);
  
 }
  
   
 // setBack - move = add
  
 Sets.SetView<CrossLinkNode> add = Sets.difference(setBack, move);
  
 for (CrossLinkNode baseNode : add)
  
 {
  
 node.addRelation(baseNode);
  
 }
  
   
 // set - move = remove
  
 Sets.SetView<CrossLinkNode> remove = Sets.difference(set, move);
  
 for (CrossLinkNode baseNode : remove)
  
 {
  
 baseNode.removeRelation(node);
  
 }
  
   
 set.clear();
  
 setBack.clear();
  
}