以下是generically rigidity 相关属性的判断功能.

Rigidity\_Tools

这个功能,只有用户使用PebbleGamePlayStage内的初始化方法以后,才会被初始化.以减少对内存的占用.

**属性:**

Nodes; %从用户输入的图直接导出的节点表对象

XData;

YData; %绘图数据，这个数据只能在初始化的时候写入，一直不变；

Edges;%从用户输入的图直接导出的边对象

**构成函数**

function obj = Rigidity\_Tools(nodes,xdata,ydata,edges)

nodes:用户输入图的节点列表

xdata,ydata：当时在用户输入图上绘制时，各节点的位置。

edges：用户输入图的边列表。

构成函数比较简单，初始化这些属性。

**方法：**

function [Redundantly,NotRedundantlyEdgeTables] = IsRedundantlyRigidGraph(obj,Detail)

Detail：是否详细打印每一个图的检测过程。

1. 首先判断原图是不是Generically Rigid Graph。如果原图都不是刚性的，何谈冗余刚性。
2. 利用RedundantlyEdges生成一个待检测边列表。这个列表是将原来的边列表每去掉一行以后，得到的一个元素
3. 穷举法，每一次从待检测边列表中取出一组，然后用IsGenericallyRigidGraph检测一下。

function [Rigid] = IsGenericallyRigidGraph(EdgeTable,NodeTable,Detail, varargin)

静态方法，可以独立调用。

EdgeTable ：是一个标准图的边描述表.

NodeTable ：是一个标准图的节点描述表.

Detail: 如果Detail是’detail’那么,程序才会输出动态图形.如果不是,那么仅仅输出是不是刚性图.这个功能会成为设计的焦点的.

Varargin{1:1} 如果是’ position’,那么将Varargin{1:2}和Varargin{1:3}做为XData, YData。用来确定绘图数据。

Rigid：返回这个图是Generically Rigid Graph 还是不是

算法：

算法的主要过程是模拟一个PebbleGamePlayStage。

1. 做一个pebble覆盖。
2. 识别Rigid Cluster。过程中如果不需要显示细节的话，只要发现多余一个Rigid Cluster则立即跳出。如果要显示细节，则识别出每一个Rigid Cluster.

[EdgeTableList] = RedundantlyEdges(EdgeTable)

静态方法。

利用EdgeTable来生成一个列表，这个列表里面每一个元素均是EdgeTable抽调一个元素生成的。

[kConnected,FailedGraph] = kConnectedCheck(EdgeTable,NodeTable,k,XDATA,YDATA,Detail)

EdgeTable：图的边列表。

NodeTable：图的节点列表

k:需要检查几链接。

XDATA,YDATA：绘图数据。

Detail：detail，仅在某个子图连接性检查失败的时候，打印信息。

VeryDetailed，每生成一个子图，均打印一次图形信息。

静态方法。

K-连接检查函数。

1. K=1。时，特殊处理，利用连接件个数检查连通性。
2. 利用排列组合函数，从NodeTable里面移除1到k-1个节点。每一次均生成一个新的子图，这个子图对应的绘图信息。
3. 每个子图判断连接件个数，以确定连通性。