第一周大作业技术文档

自25 张洵恺 2012011486

功能简介：

在0.2版solveRST中，我实现了大作业全部基础要求和扩展要求，并添加了随机生成数据、保存数据、切换背景色、保存图像等小功能。

模块划分：

为了更好地实现功能，提高开发效率，具体到每一个源文件（类），我使用了如下的模块划分方式。

|  |  |
| --- | --- |
| MainWindow | 负责绘制主界面，有相应的ui文件。连接菜单、工具栏中的选项到具体的槽，并在槽中转发给RST、RSTViewer等模块。 |
| RST | 负责处理直角斯坦纳树，重要的成员有读取数据loadPoints()；修改数据addPoint(), deletePoint(), deleteAndAdd()；调用策略计算结果solve()等。也负责标记数据是否是新的isNew()、结果是否已经被计算isSolved()等。 |
| RSTStrategy | 为RST所组合，是几个具体Strategy的抽象基类。在此处连接到老师所给的算法源码（在算法源码中集成了各自的Adapter） |
| RSTViewer | 负责管理大地图、小地图两个View，同时从友元RST中获得必要的数据。两个View发送的信号也会先发到此处，由RSTViewer处理后再转发给两个View，使流程更加条理。重要的成员包括绘制paint()、修改视口setRealPort()等。 |
| MainView | 继承自QGraphicsView，负责主视图（大地图）的管理。重载了mousePressEvent(), mouseReleaseEvent(), mouseMoveEvent()和wheelEvent()，以实现用鼠标拖动、放大、缩小、增点、删点、移动点等功能。视口变化后会发送信号至RSTViewer其方便统一管理大小地图的视口。 |
| MiniView | 继承自QGraphicsView，负责迷你视图（小地图）的管理。对场景进行调整以全部显示在视图中，并绘制视口矩形供用户拖动。类似地重载了mousePressEvent(), mouseReleaseEvent(), mouseMoveEvent()，同样会在视口变化后发送信号给RSTViewer。 |
| GenDialog | 继承自QDialog，有相应的ui文件。提供给用户调整生成参数的接口，集成了简单的随机数据生成算法。 |
| APDialog | 继承自QDialog，有相应的ui文件，供用户通过对话框输入数据来增加数据点。 |

技术细节：

本版程序致力于提供更细致、更稳定的用户体验。主要体现在以下几点：

1. 实时计算开关。实时计算开关打开时，在每次数据变更、算法切换同时都会立即调用RSTStrategy计算结果；然而在数据量较大时，开关打开可能会造成鼠标移动数据点时的卡顿。在数据量更大时，使用ZMST算法可能导致崩溃，此时会弹出警示框提示用户关闭即时计算。
2. 当前数据保存。藉由使用File-Save Current Data选项，可以保存当前随机生成或经过更改的数据，方便情景复现。
3. 精细的小地图导航。用户可以拖动小地图上标示视口的点阵矩形来调整视口中心位置。通过一个EventFilter截获widget的大小变更，点阵矩形的标示可以随着主视图组件的尺寸变化而变化。
4. 用户自选的配色方案。用户可以自由选定背景（电路板）的颜色，程序会自动计算出边框颜色。另外，当用户所选背景色较深时，小地图的视口矩形也会由深色点阵自动调节为浅色点阵。用户甚至可以将当前的简单、美观的电路板直接通过File-Save an Image保存为本地jpg或png文件。
5. 灵活的鼠标操作。在小地图上，支持用户用鼠标左键按下拖动视口框；在大地图上，支持滚轮调节缩放倍数、左键拖动画布实现视口位移、右键单击已存在的点进行删除、右键单击空白（背景或线）区域增加新点、右键拖动当前已存在的点而更改数据（配合实时计算效果更佳）。代码中使用canMove\_、isLeft\_、currentClickPos\_等多个标记来保证程序的鲁棒性。

实现难点：

1. QGraphicsView和QgraphicsScene之间的坐标变换，以及MainView和MiniView之间的坐标变换。最终代码采用了集中管理的方式，将视口统一保存在RSTViewer中，MainView和MiniView设定视口时均读取RSTViewer中所设置的信息。
2. 鼠标增删点。由于Qt并未提供QGraphicsRectItem\*向QgraphicsItem\*的强制转换办法，也没有提供判断Item类型的方法，且其他方法也不是很好用，因此鼠标增删点遇到了一定的困难，最终采用了较繁琐的指针转换，结合sceneBoundingRect方法，通过长宽比来猜测Item类型。

改进空间：

1. 当增加、删除、拖动点时，更改的信息其实很少，不需要每次进行推倒重来重做一个Scene。若可通过一个std::map管理当前元素及其指针，应该可以大大提升在增加、删除、拖动点时的程序运算速度。
2. 键盘的快捷键并未配置。
3. 并未提供给用户完全的绘制参数配置，如点的形状、点线的宽度等。