几个主要的部分

1. 左边的文件列表信息
2. 右边的图元展示信息
3. 中间部分的绘制区域
4. 工具栏的部分(主要是各种控件的展示)

左边文件列表

文件夹名称就叫五防画面，包含Tree和TreeItem两项

右边图元展示信息

图元展示信息包含两个部分:基本图元Tab页、相关控件Tab页

1. 基本图元Tab页

Tab页包含一个view(QWidget)

包含两个部分：1.树形图元结构图 2.列表图元展示图

树形图元结构图：

HIconxxxTreeWidget(树) HIconxxxTreeWigetItem(树项)

列表图元展示图:

HIconXXXListWidget(列表) HIconXXXListWidgetItem(列表项)

//需要提供接口来获取已经绘制好的图元信息然后进行展示

1. 相关控件Tab页

控件模式情况下，就采用List形式进行展示即可，减除图元预览的功能。

中间部分的绘制区域

绘制的基本图元包括(直线，矩形，文字) 其他都不要

绘制的组合图元就是包含自定义的部分

绘图文件管理

HGraphManagerHGraphManagerItem

HGraphViewHGraphScene

HIconObj类就是组合图元的部分继承HBaseObj类

注意该类需要Resize功能，主要是实现将图元大小的转换。

整个过程：(iconeditor,grapheditor,graphonline)

1. 打开画面编辑部分，最基本的两个操作过程：
2. 画面管理部分读取已绘制好的画面信息
3. 图元管理部分读取已绘制好的模板信息
4. 中间部分绘制出现的绘制view和scene,可以将组合图元拖拽到view里面实现绘制。
5. 组合图元HIconObj,HIconSymbolItem为数据基本类和图元显示类。(放到H5IconGui)

解决问题: 图元模板文件里面存储的数据都是基于较大的scene坐标绘制的。如果将比较大的坐标系统数字转换成较小的坐标系统展示。

这种类型的图元还需要有关联或者存储动态数据的功能。

遗留问题：主要是存储在模板里面元素(矩形)是topleft点，以及宽度和高度。宽度和高度可以按照比例缩放。然后topleft点是如何移动到小矩形里面对应的位置？

重点是动态数据的问题！

各种类

图形文件管理总类HGraphEditorMgr（主要管理图形文件管理类，图形系统框架类）

图形文件管理类HGraphEditorDoc (专门负责管理所有绘制好的图形)

图形文件存储在pic文件夹下面，每个图形一个文件夹？

//单个图形文件类HGraph(负责记录单个画面信息里面所有元素)

图形系统框架类HGraphEditorFrame

图形文件显示类HGraphEditorView

图形文件场景类HGraphEditorScene(负责增加/拖拽各种图元)

----------------------------------------------------------------------------------------------

图形窗口类HGraphEditorMainWindow

包含以下三个部分:

左边图形文件树型管理部分

HGraphTreeWidget, HGraphCatagoryTreeWidgetItem, HGraphFileTreeWidgetItem

图形文件树总框架，图形文件类别树结构，图文文件树结构

系统画面 ----HGraphTreeWigetItem (rootItem) ,新增的功能

|------110kVxx变主画面 ----(child:graphID)删除，刷新的功能setData(graphID)

|----最新版本 ----无

|-----110kVxx变#1主变主画面

|----最新画面

0，1，2

中间图形系统框架部分

右边图元显示管理部分

HIconTabWidget(图元管理Tab页)

里面包含两个部分:

树形图元结构图：

HIconTreeWidget

HIconTreeWigetItem(树项)

列表图元展示图:

HIconListWidget(列表) HIconListWidgetItem(列表项)

增加HIconViewEditor主要是放在Tab上面的里面存放tree和list

构建tree和list关联起来

流程分析:

tree主要是读取icon下面的所有目录，list主要是显示目录下面的图符内容

目录如何获取doc?还是自己读？应该是从doc里面读取

doc启动的时候，读取目录（保存各种目录的类型），读取所有目录下面的所有icontemplate形成一个list,这只是读取icon这一块的部分。

还有读取graph部分，待定

主要类及结构如下

HIconObj(图元组合类继承HBaseObj)

{

//二进制读写

virtual void readData(QDataStream\* data);

virtual void writeData(QDataStream\* data);

//xml文件读写

virtual void readXml(QDomElement\* dom);

virtual void writeXml(QDomElement\* dom);

virtual QStringTagName();

//拷贝克隆

virtual void copyTo(HBaseObj\* obj);

virtual void clone(HBaseObj\* obj);

virtual DRAWSHAPE getShapeType();

virtual void moveBy(qrealdx,qrealdy);

virtual void resize(qrealw,qreal h);

}

//带有动态信息的对象

HSymbolObj(HBaseObj+HDynamicObj复合数据对象)

HBaseObj🡪HIconTemplate

HDynamicObj🡪HDynamicTemplate

HSymbolTemplate？？是graph保存的主要对象。

是不是复合图元是采用HIconTemplate+HDynamicTemplate来组合完成了

HIconObj(HIconTemplate\* pIconTemplate,HDynamicTemplate\* pDynamicTemplate)

HIconObj<--->HSymbolItem

Xml结构图

<GraphEditorDoc> //画面里面的基本属性（长宽高填充颜色刷新时间等等）

<IconTemplates> //模板，只要在里面用了都增加到模板里面

<IconTemplate>

</IconTemplate>

…

<IconTemplate>

</IconTemplate>

</IconTemplates>

<Relations>

<SymbolObj> //属性遥测/遥信uuid,name,字体，id,线型,位置等等

<IconSymbol> //记录实际显示图元信息，红绿灯，高度，宽度，颜色，类似上面的

<Relation Name=””Station=””Point=””Attrib=”” />

</IconSymbol>

</Symbol>

<SymbolObj>

…

</SymbolObj>

<Circle>…..

<Rectangle>

</Relations>

</GraphEditorDoc>

首先保存基础模板信息。然后具体图元的矩形大小需要保存。

需要通过实际的矩形大小来计算出模板里面每个图元的在这个矩形位置的大小就可以了。

如何绘制HIconTemplate模板的图形呢？

当拖拽到view上释放之后，

1.根据获取的template获取对象，然后新建一个HSymbolObj,

2.创建一个HIconSymbolItem(addItem),必然会关联一个HSymbolObj的对象，从而将template的对象传递到item里面，item获取到template后的对象后，obj将新建一个iconSymbol将template里面的iconSymbol复制到Obj里面,然后按照比例进行缩放设置，把Obj里面的iconSymbol缩放，缩放比例应该是10倍（待确定），topLeft需要按照比例对应缩放（待确定）

3.根据默认的大小创建设置item的大小，然后addItem

4. Item的painter 根据不同icon进行不同的绘制

5.移动item 是否需要重新计算，需要看topLeft这个东西如何缩放

6.改变item大小，肯定要重新计算，只要按照新的比例缩放重新计算就可以了

重新修改icon部分

主要是将绘制函数painter改到obj里面

可以这样解决，能不能在item里面构建一个IconSymbol，在IconSymbol里面resize（或者HBaseObj来实现resize）。

1. 修改HIconSymbolItem的名字HIconComplexItem(复杂图符)、HIconComplexObj

2.mgr创建一个icontemplate,然后item获取传递的对象icontemplate,里面有size可以作为item的大小

3.当拖拽一个Icon到scene上面时，首先创建HIconComplexItem到scene上，将iconTemplate复制到Item里面，同时创建一个HIconComplexObj对象，将Item里面的rect拷贝过来，注意rect的width和height来自于iconTemplate.注意这个是绘制的矩形大小

注意HIconComplexObj的需要有个新的iconSymbol，创建obj后需要将icontemplate里面的iconSymbol拷贝过去，然后resize改变里面icon的大小和位置。进行绘制

HGraph内容的构建

HGraph主要是由基本图元和复杂图元组成的。复杂图元包含图元本身(图元模板)还有动态数据两个部分。涉及的问题就是新增图元，打开图元，保存图元，删除图元等功能。

保存图元重点：

当从图元模板拖动一个复杂图元后，首先mgr创建一个icontemplate对象，这个对象要增加到HGraph画面的hicontemplate的list中，保存。这个是模板部分的保存。

具体到ComplexObj的保存，分为两个部分：

1. 复杂图元的关联属性包含 icontemp的部分。就是隶属模板的uuid
2. 保存ComplexObj本身的属性。最好iconsymbol的属性也保存一下。

总而言之ComplexObj 有icontemp,iconsymbol,加上自身的属性三个静态属性

HComlexObj分析

HComplexObj主要包含了HIconTemplate,和HIconSymbol两个对象

其中HIconSymbol是新建的，这个iconSymbol对象是从icontemplate拷贝过来的呢，还是从graph文件中读取过来呢？取决于这个iconsymbo是否进行了改变，如果改变了就要从graph文件里面读取，如果不改变可以从icontemplate里面拷贝。

但是不是需要改变呢？IconSymbol是不是需要这个变量?还要值得研究一下。就是在绘制复杂图符。应该是不需要的待定吧。

HMainWindow

{

initMainWindow();//初始化窗口

initGraphMgr();//初始化画面管理

}

initMainWindow

{

addWidget(graphicsView);

pGraphMgr = null;

//初始化右边tab结构

//初始化左边的树结构

HGraphTreeWidget,

HGraphCatagoryTreeWidgetItem,

HGraphFileTreeWidgetItem

//创建各种actions和工具栏;

//创建状态栏

}

initGraphMgr(HGraphEditorMgr—画面相关操作最后都是归结到此处完成)

{

new HGraphEditorMgr

}

HGraphEditorMgr

{

new HGraphEditorDoc //存储画面信息

new QUndoStack

new HGraphicEditorScene

void loadGraphFile();//读取所有图形文件

void createTemplateObj(constQString&name,constQString&sUuid);//添加模板图元

}

HGraphEditorDoc

{

readXmlFile

writeXmlFile

readData

writeData

AddIcon;//增加图元

AddIconObj;//增加icon

…

}

HGraphEditorView

{

}

HGraphEditorScene

{

virtual void drawBackground(QPainter \*painter, constQRectF&rect);

virtual void[dragEnterEvent](qgraphicsscene.html#dragEnterEvent)(QGraphicsSceneDragDropEvent \*event);

virtual void[dragMoveEvent](qgraphicsscene.html#dragMoveEvent)(QGraphicsSceneDragDropEvent \*event);

virtual void[dropEvent](qgraphicsscene.html#dropEvent)(QGraphicsSceneDragDropEvent \*event);

}

HIconComplexItem和HIconComplexObj的问题

注意要保存动态数据没有处理

另外初始化开始用的到底是那一个pattern要确定

10.24完成了HIconComplexItem,HIconComplexObj的基本工作，尚遗留动态数据没有保存。还有保存拷贝工作。

预计后面完成

完成station

画面保存，模板读取等工作最好也是提供lib来实现，避免在不同地方进行相同的读写工作

1. 读取文件必须有文件名。
2. 打开文件读取文件，有模板信息(模板列表)，有图元信息(复杂/单个图元列表)，
3. 应为需要关联厂站信息，所有要有厂站列表