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

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

HIconCatagoryTreeWidgetItem(树目录) HIconTypeTreeWigetItem(树项)

列表图元展示图:

HIconListWidget(列表) HIconListWidgetItem(列表项)

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>

</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，3.根据默认的大小创建设置item的大小，4根据template里面的比例关系，自动计算出在item大小之内的所有绘制图元的比例。 问题:就算出来的每个template里面的元素是如何保存的？是每次绘图都计算一次还是计算好了保存起来呢？

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）。

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);

}