//创建过程暂时不述

在函数

void MIconBaseMainwin::CreateActions()

{

//绘制

toolDrawLine = new QAction(QIcon(":/icon/basic\_line.png"),tr("&Line"),this);

Q\_CHECK\_PTR(toolDrawLine);

toolDrawLine->setData(MBaseDefine::Line);

connect(toolDrawLine,SIGNAL(triggered()),this,SLOT(EnterDrawState()));

toolDrawBezier = new QAction(QIcon(":/icon/basic\_bezier.png"),tr("&Bezier"),this);

Q\_CHECK\_PTR(toolDrawBezier);

toolDrawBezier->setData(MBaseDefine::Bezier);

connect(toolDrawBezier,SIGNAL(triggered()),this,SLOT(EnterDrawState()));

toolDrawPolyline = new QAction(QIcon(":/icon/basic\_polyline.png"),tr("&Polyline"),this);

Q\_CHECK\_PTR(toolDrawPolyline);

toolDrawPolyline->setData(MBaseDefine::Polyline);

connect(toolDrawPolyline,SIGNAL(triggered()),this,SLOT(EnterDrawState()));

toolDrawPolygon = new QAction(QIcon(":/icon/basic\_ploygon.png"),tr("&Polygon"),this);

Q\_CHECK\_PTR(toolDrawPolygon);

toolDrawPolygon->setData(MBaseDefine::Polygon);

connect(toolDrawPolygon,SIGNAL(triggered()),this,SLOT(EnterDrawState()));

toolDrawOrtholine = new QAction(QIcon(":/icon/basic\_orthline.png"),tr("&Ortholine"),this);

Q\_CHECK\_PTR(toolDrawOrtholine);

toolDrawOrtholine->setData(MBaseDefine::OrthLine);

connect(toolDrawOrtholine,SIGNAL(triggered()),this,SLOT(EnterDrawState()));

toolDrawArc = new QAction(QIcon(":/icon/basic\_arc.png"),tr("&Arc"),this);

Q\_CHECK\_PTR(toolDrawArc);

toolDrawArc->setData(MBaseDefine::Arc);

connect(toolDrawArc,SIGNAL(triggered()),this,SLOT(EnterDrawState()));

toolDrawEllipse = new QAction(QIcon(":/icon/basic\_ellipse.png"),tr("&Ellipse"),this);

Q\_CHECK\_PTR(toolDrawEllipse);

toolDrawEllipse->setData(MBaseDefine::Ellipse);

connect(toolDrawEllipse,SIGNAL(triggered()),this,SLOT(EnterDrawState()));

toolDrawCircle = new QAction(QIcon(":/icon/basic\_circle.png"),tr("&Circle"),this);

Q\_CHECK\_PTR(toolDrawCircle);

toolDrawCircle->setData(MBaseDefine::Circle);

connect(toolDrawCircle,SIGNAL(triggered()),this,SLOT(EnterDrawState()));

toolDrawRect = new QAction(QIcon(":/icon/basic\_rect.png"),tr("&Rectangle"),this);

Q\_CHECK\_PTR(toolDrawRect);

toolDrawRect->setData(MBaseDefine::Rectangle);

connect(toolDrawRect,SIGNAL(triggered()),this,SLOT(EnterDrawState()));

toolDrawText = new QAction(QIcon(":/icon/basic\_text.png"),tr("&Text"),this);

Q\_CHECK\_PTR(toolDrawText);

toolDrawText->setData(MBaseDefine::Text);

connect(toolDrawText,SIGNAL(triggered()),this,SLOT(EnterDrawState()));

}

模拟选择直线，随后进入EnterDarwState()函数中

void MIconBaseMainwin::EnterDrawState()

{

QAction\* action = qobject\_cast<QAction\*>(sender());

if(!action){

return;

}

if(!m\_pEditMgr){

return;

}

MIconBaseEditOp\* editOp = m\_pEditMgr->EditOp();

if(!editOp){

return;

}

QVariant data = action->data();

DrawObjTypeEnum type = Unknown;

DrawObjTypeEnum shapeType = Unknown;

bool ok;

if(data.type()==QVariant::List){

QVariantList list = data.toList();

if(list.count()!=2){

return;

}

type = static\_cast<DrawObjTypeEnum>(list.at(0).toInt(&ok));

if(!ok){

return;

}

shapeType = static\_cast<DrawObjTypeEnum>(list.at(1).toInt(&ok));

if(!ok){

return;

}

}

else{

type = shapeType = static\_cast<DrawObjTypeEnum>(data.toInt(&ok));

if(!ok){

return;

}

}

editOp->EnterDrawState(type,shapeType);

if(m\_pEditor&&m\_pEditor->View()){

m\_pEditor->View()->setInteractive(false);

}

}

进入到MIconBaseEditOp中调用了 EnterDrawState(xx,xx)进入了绘制状态

主要工具就选择绘制工具

bool MIconBaseEditOp::EnterDrawState(DrawObjTypeEnum type,DrawObjTypeEnum shapeType)

{

MStateMgr\* stateMgr = m\_pEditMgr->m\_pStateMgr;

if(!stateMgr){

return false;

}

MIconBaseDrawState\* drawState = (MIconBaseDrawState\*)stateMgr->SwitchState("ICONDRAWSTATE");

if(!drawState){

return false;

}

//根据图元类型,控制绘制的图符类型

QList<MDrawObj\*> objs = FindObjsByType(type,shapeType);

if(m\_pEditMgr->m\_nCatalogId==5){//bus

if(type!=LinkShapeLineOrth&&type!=LinkShapeOrthLine){

type = Unknown;

}

else{

if(objs.count()>0){

type = Unknown;

}

}

}

else if(m\_pEditMgr->m\_nCatalogId==15){//tieline

if((!(type==LinkLine&&(shapeType==Polyline||shapeType==OrthLine)))&& (type!=Text))

{

type = Unknown;

}

else{

if(objs.count()>0){

type = Unknown;

}

}

}

else if(m\_pEditMgr->m\_nCatalogId==31){//value

if(type!=Text){

type = Unknown;

}

}

drawState->ChangeTool(type,shapeType);

OnEndDraw();

QCursor cursor = drawState->Cursor();

if(m\_pEditMgr->m\_pEditor){

m\_pEditMgr->m\_pEditor->CursorChanged(cursor);

}

return true;

}

绘制工具选择好之后,进入绘制部分

绘制工具就在带有view的MIconBaseEditor(QWidget)中进行的

bool MIconBaseEditor::eventFilter(QObject\* obj, QEvent\* event)

{

static bool bRight = false;

MEditor::eventFilter(obj,event);

if(obj==m\_pView->viewport()){

switch(event->type()){

case QEvent::MouseButtonDblClick:

case QEvent::MouseButtonPress:

case QEvent::MouseButtonRelease:

case QEvent::MouseMove:

{

if(!m\_pEditMgr||!m\_pEditMgr->StateMgr()||

!m\_pEditMgr->StateMgr()->CurState()||!m\_pView){

return false;

}

//带右键

if(event->type()==QEvent::MouseButtonRelease)

{

QMouseEvent\* me = (QMouseEvent\*)event;

if(me->button()==Qt::RightButton){

bool isRotating = false;

if(m\_pEditMgr->StateMgr()->CurState()->Type()=="ICONSELECTSTATE")

{

MIconBaseSelectState\*selectState= (MIconBaseSelectState\*)m\_pEditMgr->StateMgr()->CurState();

if(selectState&&selectState->SubState()==MState::ROTATE){isRotating = true;}}

if(m\_pEditMgr->StateMgr()->CurState()->Type()=="ICONDRAWSTATE"

|| m\_pEditMgr->StateMgr()->CurState()->Type()=="ZOOM"|| isRotating){

m\_pEditMgr->EditOp()->OnSwitchToSelectState();

bRight = true;

return true;}}}

QPointF pos = m\_pView->mapToScene(((QMouseEvent\*)event)->pos());

if(m\_pEditMgr->StateMgr()->CurState()->Type()=="ICONDRAWSTATE"){

m\_pEditMgr->EditOp()->AdhereToGrid(pos);}

//很重要，发送MEvent到当前状态中

**MEvent mEvent(event,QVariant(pos));**

**m\_pEditMgr->StateMgr()->CurState()->OnEvent(mEvent);**

return false;

}

case QEvent::ContextMenu:

{

if(bRight){

bRight = false;

return true;

}

if(!m\_pEditMgr||!m\_pEditMgr->StateMgr()||

!m\_pEditMgr->StateMgr()->CurState()){

return false;

}

if(m\_pEditMgr->StateMgr()->CurState()->Type()=="ICONSELECTSTATE"){

MIconBaseSelectState\* selectState = (MIconBaseSelectState\*)m\_pEditMgr->StateMgr()->CurState();

if(selectState&&selectState->SubState()==MState::ROTATE){

m\_pEditMgr->EditOp()->OnSwitchToSelectState();

return false;

}

}

MEvent mEvent(event,QVariant());

m\_pEditMgr->StateMgr()->CurState()->OnEvent(mEvent);

return false;

}

case QEvent::Wheel:

case QEvent::Drop:

{

return false;

}

default:

return false;

}

}

else if(obj==m\_pView){

switch(event->type()){

case QEvent::KeyPress:

case QEvent::KeyRelease:

{

if(!m\_pEditMgr||!m\_pEditMgr->StateMgr()||

!m\_pEditMgr->StateMgr()->CurState()){

return false;

}

MEvent mEvent(event,QVariant());

m\_pEditMgr->StateMgr()->CurState()->OnEvent(mEvent);

return false;

}

default:

{

return false;

}

}

}

else{

return false;

}

}

此时进入绘制状态里面调用绘制工具进行绘制

void MDrawState::OnEvent(MEvent& e)

{

if(m\_pCurTool){

m\_pCurTool->OnEvent(e);

}

}

//举例 矩形的工具就是如此，主要是OnMouseRelease

void MRectTool::OnEvent(MEvent& e)

{

MDrawTool::OnEvent(e);

if(e.event->type()==QEvent::MouseButtonPress){

OnMousePress((QMouseEvent\*)e.event,e.data);

}

else if(e.event->type()==QEvent::MouseMove){

OnMouseMove((QMouseEvent\*)e.event,e.data);

}

else if(e.event->type()==QEvent::MouseButtonRelease){

OnMouseRelease((QMouseEvent\*)e.event,e.data);

}

}

//因为是绘制结束 所以创建对象的工作必须要在此处完成。

void MRectTool::OnMouseRelease(QMouseEvent\* e,QVariant& data)

{

QPointF point = data.toPointF();

m\_vCurLogPoint = point;

if(e->button()!=Qt::LeftButton){

return;

}

if(mFuzzyCompare(m\_vCurLogPoint.x(),m\_vPreLogPoint.x())&&mFuzzyCompare(m\_vCurLogPoint.y(),m\_vPreLogPoint.y())){

if(m\_eObjType!=LinkNode){

return;

}

}

MDrawObj\* obj = NULL;

switch(m\_eObjType) {

case IconObj:

{

MIconTemplate\* iconTemplate = GetTemplate(m\_sCatalog, m\_sUuid);

if(!iconTemplate){

return;

}

obj = iconTemplate->NewObj("");

break;

}

case ComponentObj:

{

obj = MDrawObjFactory::Instance()->Product(m\_eObjType,false,m\_sUuid);

break;

}

default:

{

bool isPhoto = m\_eObjType==Photo;

if(m\_eObjType==Photo){

m\_eObjType = Rectangle;

}

obj = MDrawObjFactory::Instance()->Product(m\_eObjType);

if(isPhoto){

MRectangle\* rect = (MRectangle\*)obj;

rect->SetBackgroundImagePtr(new QImage());

rect->bFill = true;

rect->bBorderVisible = false;

}

break;

}

}

if(obj){

QPolygonF points;

QRectF bounding;

if(m\_eObjType==Circle){

qreal radius = sqrt(pow((m\_vCurLogPoint.x()-m\_vPreLogPoint.x()),2)+pow((m\_vCurLogPoint.y()-m\_vPreLogPoint.y()),2));

bounding.setSize(QSizeF(radius\*2,radius\*2));

bounding.moveCenter(m\_vPreLogPoint);

}

else{

bounding = QRectF(m\_vPreLogPoint,m\_vCurLogPoint).normalized();

QPointF oldTopLeft = bounding.topLeft();

if(e->modifiers()&Qt::ControlModifier){

double xDelta = qAbs(m\_vCurLogPoint.x()-m\_vPreLogPoint.x());

double yDelta = qAbs(m\_vCurLogPoint.y()-m\_vPreLogPoint.y());

bounding.setWidth(qMin(xDelta,yDelta));

bounding.setHeight(qMin(xDelta,yDelta));

bounding.moveTopLeft(oldTopLeft);

}

}

bounding = bounding.normalized();

QPointF center = bounding.center();

if(bounding.width()<5){

bounding.setWidth(5);

}

if(bounding.height()<5){

bounding.setHeight(5);

}

bounding.moveCenter(center);

points.append(bounding.topLeft());

points.append(bounding.bottomRight());

if( m\_eShapeType>Unknown )

{

obj->SetShapeType(m\_eShapeType);

}

obj->SetPointList(points);

SetProperty(obj);

m\_pState->AppendObj(obj);

}

m\_pState->OnEndDraw();

}

实际上绘制类是MIconBaseDrawState

void MIconBaseDrawState::AppendObj(MDrawObj\* obj)

{

if(!m\_pEditMgr){

return;

}

if(m\_pEditMgr->Template()&&m\_pEditMgr->Template()->pSymbol){

m\_pEditMgr->Template()->pSymbol->AddChild(obj);

}

if(m\_pEditMgr->EditOp()){

m\_pEditMgr->EditOp()->OnObjCreated(obj,false); //

//终于进入创建对象部分，

}

MIconBaseNewCommand\* newCommand = new MIconBaseNewCommand(m\_pEditMgr,obj);

Q\_CHECK\_PTR(newCommand);

m\_pEditMgr->CommandHistory()->push(newCommand);

}

//在创建过程中进行了多种信号关联，

void MIconBaseEditor::ObjCreated(MDrawObj\* obj,bool isPaste)

{

if(!m\_pView||!m\_pView->scene()){

return;

}

MIconBaseScene\* scene = (MIconBaseScene\*)m\_pView->scene();

if(!scene){

return;

}

MDrawObjItem\* item = new MDrawObjItem(obj);

Q\_CHECK\_PTR(item);

if(isPaste){

obj->nStackOrder = m\_pEditMgr->m\_nCurZValue + 1 + obj->nStackOrder;

item->setZValue(obj->nStackOrder);

}

scene->addItem(item);

//进行了关联，注意这里主要绘图基本元素，不是组合图形

connect(item,SIGNAL(SelectChanged(MDrawObj\*,bool)),this,SLOT(OnSelectChanged(MDrawObj\*,bool))); //[1]

connect(item,SIGNAL(RecalcSelect()),this,SLOT(OnRecalcSelect()));//[2]

}

//此处是当点击对象时进行的选择过程：

当在scene中进行点击，进入到Item(继承QGraphicsItem)中，发射SelectChanged信号如下：

void MIconBaseScene::mousePressEvent(QGraphicsSceneMouseEvent\* mouseEvent)

{

bool multiSelect = (mouseEvent->modifiers()&Qt::ControlModifier)!=0;

if(multiSelect){

if(!itemAt(mouseEvent->scenePos())){

mouseEvent->ignore();

return;

}

}

QGraphicsScene::mousePressEvent(mouseEvent);

}

//scene的mousePressEvent完成后,只有有选中的item，会自动进入item的mousePressEvent

Item没有mousePressEvent只有itemChange函数，所以进入如下函数：

QVariant MDrawObjItem::itemChange(GraphicsItemChange change, const QVariant & value)

{

if(m\_pDrawObj&&!m\_pDrawObj->bDeleted)

{

if(change==QGraphicsItem::ItemSelectedChange && m\_pDrawObj->IsVisible() )

{

emit SelectChanged(m\_pDrawObj,value.toBool()); //看[1]处

}

else if(change==QGraphicsItem::ItemPositionHasChanged)

{

QPointF pos = this->pos();

m\_pDrawObj->Move(pos.x(),pos.y());

emit RecalcSelect();

}

}

return QGraphicsItem::itemChange(change,value);

}

根据信号关联关系

调用如下函数//看[1]处：

void MIconBaseEditor::OnSelectChanged(MDrawObj\* obj,bool isSelected)

{

if(!m\_pEditMgr||!m\_pEditMgr->EditOp()){

return;

}

m\_pEditMgr->EditOp()->SelectChanged(obj,isSelected);

}

调用 SelectionMgr中的两个函数，同时发射信号

void MIconBaseEditOp::SelectChanged(MDrawObj\* obj,bool isSelected)

{

if(m\_pEditMgr&&m\_pEditMgr->m\_pSelectionMgr){

m\_pEditMgr->m\_pSelectionMgr->SelectChanged(obj,isSelected); //[3]

m\_pEditMgr->m\_pSelectionMgr->RecalcSelect();//[4]

}

emit SelectChanged();//[5]

}

上述信号关联在

void MIconBaseMainwinMgr::New()

{

//新建编辑管理并初始化相应的状态

MIconBaseEditMgr\* editMgr = new MIconBaseEditMgr;

Q\_CHECK\_PTR(editMgr);

MIconBaseEditor\* editor = new MIconBaseEditor(editMgr);

Q\_CHECK\_PTR(editor);

editMgr->EditOp()->New(catalogName,catalogId,shapeType,iconName);

MZoomState\* zoomState = new MZoomState;

Q\_CHECK\_PTR(zoomState);

MBaseMeasureState\* measureState = new MBaseMeasureState;

Q\_CHECK\_PTR(measureState);

MIconBaseDrawState\* drawState = new MIconBaseDrawState(editMgr);

Q\_CHECK\_PTR(drawState);

MIconBaseSelectState\* selectState = new MIconBaseSelectState(editMgr);

Q\_CHECK\_PTR(selectState);

editMgr->StateMgr()->AddState(zoomState);

editMgr->StateMgr()->AddState(measureState);

editMgr->StateMgr()->AddState(drawState);

editMgr->StateMgr()->AddState(selectState);

connect(zoomState,SIGNAL(PointZoom(const QPointF&,bool)),editMgr->EditOp(),SLOT(OnPointZoom(const QPointF&,bool)));

connect(zoomState,SIGNAL(RectZoom(const QRectF&,bool)),editMgr->EditOp(),SLOT(OnRectZoom(const QRectF&,bool)));

connect(zoomState,SIGNAL(FreeZoom(bool)),editMgr->EditOp(),SLOT(OnFreeZoom(bool)));

connect(zoomState,SIGNAL(DrawPath(const QList<Path>&)),editMgr->EditOp(),SLOT(OnDrawPath(const QList<Path>&)));

connect(zoomState,SIGNAL(EndDraw()),editMgr->EditOp(),SLOT(OnEndDraw()));

connect(measureState,SIGNAL(DrawPath(const QList<Path>&)),editMgr->EditOp(),SLOT(OnDrawPath(const QList<Path>&)));

connect(measureState,SIGNAL(EndDraw()),editMgr->EditOp(),SLOT(OnEndDraw()));

connect(drawState,SIGNAL(DrawPath(const QList<Path>&)),editMgr->EditOp(),SLOT(OnDrawPath(const QList<Path>&)));

connect(drawState,SIGNAL(EndDraw()),editMgr->EditOp(),SLOT(OnEndDraw()));

connect(selectState,SIGNAL(EndDraw()),editMgr->EditOp(),SLOT(OnEndDraw()));

connect(selectState,SIGNAL(RefreshSelect(const QRectF&)),editMgr->EditOp(),SLOT(OnRefreshSelect(const QRectF&)));//[6]

connect(editMgr->EditOp(),SIGNAL(SelectChanged()),selectState,SLOT(OnSelectChanged()));//[7]

}

先看SelectionMgr中的SelectChanged、RecalcSelect两个函数

SelectChanged中就是添加/删除函数

//看[3]、[4]

void MSelectionMgr::SelectChanged(MDrawObj\* obj,bool isSelected)

{

if(isSelected){

AddObj(obj);

}

else{

RemoveObj(obj);

}

}

AddObj如下：

void MSelectionMgr::AddObj(MDrawObj\* obj)

{

//注意添加到的对象

m\_pTempGroup->AddMember(obj,m\_bIsCalcOnChange);//[aa]

MDrawObjHelper\* helper = MDrawObjHelper::Instance();

if(helper){

if(m\_pTempGroup->pChild.Size()==0){

helper->SetObj(0);

}

else if(m\_pTempGroup->pChild.Size()==1){

helper->SetObj(m\_pTempGroup->pChild.At(0));

}

else{

helper->SetObj(m\_pTempGroup);

}

}

}

//看[4]

void MSelectionMgr::RecalcSelect()

{

QRectF oldBounding = m\_vBounding;

CalcPoints();

for(int i=0; i<m\_pTempGroup->pChild.Size(); i++){

if(m\_pTempGroup->pChild.At(i)&&m\_pTempGroup->pChild.At(i)->GetViewItem()){

m\_pTempGroup->pChild.At(i)->GetViewItem()->update();

}

}

emit RefreshSelect(m\_vBounding.united(oldBounding));// [8]

}

再接着分析[5]、[7]处

void MIconBaseSelectState::OnSelectChanged()

{

}

无任何处理

分析[8]、[6]处

可以发现对scene中的rect进行了刷新

void MIconBaseEditOp::OnRefreshSelect(const QRectF& rect)

{

if(m\_pEditMgr->m\_pEditor){

m\_pEditMgr->m\_pEditor->RefreshSelect(rect);

}

}

void MIconBaseEditor::RefreshSelect(const QRectF& rect)

{

if(!m\_pView||!m\_pView->scene()){

return;

}

//[bb]

MIconBaseScene\* scene = (MIconBaseScene\*)m\_pView->scene();

if(!scene){

return;

}

if(rect.isNull()){

scene->invalidate(scene->sceneRect(),QGraphicsScene::ForegroundLayer);

}

else{

scene->invalidate(rect,QGraphicsScene::ForegroundLayer);

}

}

至此分析出了当按住ctrl或者没有按住整过流程，选择的对象都是存在[aa]处，画面更新在[bb]处

相关类分析：

图形部分总的分为四个部分

Graph->H5Base、H5IconEditor、H5GraphEditor、H5GraphOnline

H5Base->H5IconLib(图元基本部分、复杂图元等)

->H5IconHelp(辅助信息)

->H5IconGui(绘制部分)

工具类包含

HToolManager、HRectTool、HLineTool...

状态类

HState HDrawState

HIconDrawState (选择、)

组合图形部分流程：

当画面上绘制2个矩形，按照组合键之后，两个矩形(Rectange)合并成一个组合图形(Group)

具体分析2个矩形和组合后的Group之间位置的关系。

当绘制一个矩形Rectange1是，根据鼠标在scene的移动位置得到Rectange1的左上角、右下角的点信息。通过取Center中心点，可以得到Rectange1在scene的pos位置。这个pos位置非常重要，放大缩小移动变换等操作都是基于此pos位置。

Rectange1对应的HRectange1Item，应该是将(0,0)的位置对应到Rectange1在scene上的pos位置，Rectange1的位置应该通过pos设为绝对坐标。。。。。。

什么时候用绝对坐标 什么时候用相对坐标？

绘制的时候只要相对坐标就行了加上旋转、翻转等操作就可以了，这是因为绘制都是基于pos的，只要pos位置定下来就基本上没有任何问题。

单个图元、或者多个图元选择后移动，也是相对坐标，因为移动的都是图元的pos位置。

**多个图元选择后的改变大小问题就需要进行坐标变换了。**

坐标变换主要是针对组合的图形非常重要。

当两个图形组合之后就要变化了，当两个图形组合成一个组合图形之后，组合图形就是这个两个图形的父图形。所以位置要相对于父图形变化，而不再是scene了

原来当两个图形的位置是pos1,pos2,这两个位置是相对于scene的绝对坐标，当合并后父图形位置为pos,当合并之后，应该是pos1 = pos1 - pos,pos2 = pos2-pos计算出相对于pos的位置两个点。此时的pos1位置和pos2位置变化了，变成相对于父图像pos的位置。所以再求pos1和pos2画面位置的值应该是pos1+ pos和pos2+pos.

当然了如果发生了坐标系的变化还需要将pos1和pos2转换到变化坐标系下对应的位置

