【qgisapp.h】

PLASmartCardManager\* mSmartCardManager;

PLAMaptoolCardSettingDialog\* mMapToolCardSettingDialog;

QAction\* mActionMapToolCardSetting;

【qgisapp.cpp】

mSmartCardManager = new PLASmartCardManager(mMapCanvas);

PLASmartCardManager\* QgisApp::smartCardManager()

{

Q\_ASSERT( mSmartCardManager );

return mSmartCardManager;

}

mActionMapToolCardSetting(0)

mMapToolCardSettingDialog = new PLAMaptoolCardSettingDialog(this);

void QgisApp::onMaptoolCardSetting()

{

mMapToolCardSettingDialog->show();

}

mActionMapToolCardSetting = new QAction(QgisApp::instance());

mActionMapToolCardSetting->setText(tr("鼠º¨®标À¨º工¡è具?助¨²手º?"));

connect(mActionMapToolCardSetting, SIGNAL(triggered()), this, SLOT(onMaptoolCardSetting()));

mSettingsMenu->addAction(mActionMapToolCardSetting);

【qgsmaptool.h/qgsmaptool.cpp】

public:

void insertSmartTrackCard(PLAMaptoolSmartTrackerCard\* card);

void insertSmartProcessCard(PLAMaptoolSmartProcessCard\* card);

const PLAMaptoolSmartTrackerCard\* smartTrackerCard() const;

const PLAMaptoolSmartProcessCard\* smartProcessCard() const;

protected:

PLAMaptoolSmartTrackerCard\* mSmartTrackerCard;

PLAMaptoolSmartProcessCard\* mSmartProcessCard;

void QgsMapTool::insertSmartTrackCard(PLAMaptoolSmartTrackerCard\* card)

{

mSmartTrackerCard = card;

}

void QgsMapTool::insertSmartProcessCard(PLAMaptoolSmartProcessCard\* card)

{

mSmartProcessCard = card;

}

const PLAMaptoolSmartTrackerCard\* QgsMapTool::smartTrackerCard() const

{

return mSmartTrackerCard;

}

const PLAMaptoolSmartProcessCard\* QgsMapTool::smartProcessCard() const

{

return mSmartProcessCard;

}

【PLASmartCard.h/PLASmartCard.cpp】

1. PLACardSetting类：智能卡的参数的一些信息，有

参数名称，参数描述，参数类型，参数的值，参数的选项五个

1. PLASmartCard类（虚类）：智能卡本身的一些信息，有

1、成员变量

智能卡的唯一标识符：mID

智能卡需要的参数集合（PLACardSetting的Map）：mSettings

2、成员函数

三个实函数，用于更新参数：

addSettings（const QString& key,const PLACardSettings& settings,bool update=true）; 用于添加参数，最后一个参数用于表示如果该参数已存在是否要更新该参数，默认为是

updateSetting(const QString& key,QVariant value); 更新指定参数；

updateSetting(const PLASmartCard& other); 更新智能卡的所有参数

三个纯虚函数：

Name(),describe();category() 分别获得智能卡的名称、描述和类别，其中类别有未知、处理器、跟踪器三类。

1. PLASmartMaptoolCard类（虚类）：继承PLASmartCard，根据id构建一个智能卡（PLASmartCard），并关联一个画布

成员变量：QgsMapCanvas\* mCanvas；

1. PLAMaptoolSmartTrackerCard类（虚类）：继承PLASmartMaptoolCard，根据id和画布构建一个智能卡地图工具（PLASmartMaptoolCard）

成员函数：

虚函数：

SmartCardCategory category（） const； 返回 SCCMaptoolTracker；

纯虚函数：

virtual bool track(const QList<QgsPoint>& sample\_points, QList<QgsPoint>& result) = 0; 跟踪器处理函数

1. PLAMaptoolSmartProcessCard类（虚类）：继承PLASmartMaptoolCard，根据id和画布构建一个智能卡地图工具（PLASmartMaptoolCard）

成员函数：

虚函数：

SmartCardCategory category（） const； 返回 SCCMaptoolProcess；

纯虚函数：

virtual bool process(const QList<QgsPoint>& sample\_points, QVector<QVector<QgsPoint> >& result

) = 0; 处理器处理函数

【PLASmartCardManager.h/PLASmartCardManager.cpp】

1. PLASmartCardManager类：用于管理所有的智能卡

成员变量：

QList<PLASmartCard\*> mCardList; 所有注册的智能卡

QgsMapCanvas\* mCanvas； 智能卡作用的画布

成员函数：

1. 构造函数：用一个画布构造一个PLASmartCardManager对象
2. 注册卡：registerCard（）

如果当前ID的卡不在mCardList中，则添加进来，并发出信号cardRegistered()，在PLAMaptoolCardSettingDialog中有相应的槽函数

void PLAMaptoolCardSettingDialog::onCardRegistered(PLASmartCard\* card)

{

cbTrackers->blockSignals(true); 禁止cboBox发送信号

cbTrackers->addItem(card->name(), (qlonglong)card);

cbTrackers->blockSignals(false); 取消禁止

}

Qlonglong这里，直接存储了card对象，相当有用，不用去二次解析。

1. 卸载卡：UNregisterCard（const QString& id）；

在mCardList中删除id对应的卡，释放beforeCareUnregister信号，该信号没有槽对应。

1. unregisterAll（）；

清空mCardList，释放cardsChanged信号，该信号没有槽对应。

1. QList<PLASmartCard\*> listCards(SmartCardCategory c = SCCUnkonwn);

过滤出对应类型的卡

1. PLASmartCard\* getCard(const QString& id);

获得指定id的卡

【plamaptoolcardsettingdialog.h/plamaptoolcardsettingdialog.cpp】

处理一些ui事件，结果是得到一个智能卡

关于comboBox中存储元数据：

1）存储： cbTrackers->addItem(card->name(), (qlonglong)card);

2）读取： const PLAMaptoolSmartTrackerCard\* c = (const PLAMaptoolSmartTrackerCard\*)cbProcessors->itemData(i).toLongLong();

【在画布中画的点如何获取】

跟踪卡：在qgsMapToolCapture.cpp中

if((e->modifiers() & Qt::ControlModifier) && e->button() == Qt::LeftButton && mSmartTrackerCard && points().size() > 0)

{

QList<QgsPoint> result;

QList<QgsPoint> sample;

sample.append(points().last());

sample.append(e->mapPoint());

if(mSmartTrackerCard->track(sample, result))

{

foreach(const QgsPoint& p, result)

addVertex(p);

}

}

1）e->modifiers()判断是否按下某个键盘键，其余QT::ControlModifiers求逻辑与，就可以判断是否摁下Ctrl键

2）points()返回的是橡皮条中的点数，不包含当前点

3）用points()中的最后一个点和当前鼠标点击的点作为样本进行计算，捕捉点，将生成的点加到橡皮条中

处理器：在qgsMapToolAddFeature.cpp qgsmaptooladdring.cpp qgsmaptooladdpart.cpp中

if ( mSmartProcessCard )

mSmartProcessCard->process( points(),result );

if ( !result.isEmpty() )

{

QgsPolygon polygon;

foreach(const QVector<QgsPoint>& ps, result)

polygon << ps;

g = QgsGeometry::fromPolygon( polygon );

}

即是将要素简单替换为计算完产生的要素

【智能卡是什么时候加入智能卡列表的】

每个智能卡都是一个插件，在加载插件时initGui函数中有如下代码，将插件加入智能卡列表：

mQGisIface->smartCardManager()->registerCard( PLASmartTrackInnerCard(mQGisIface->mapCanvas()) );

mQGisIface->smartCardManager()->registerCard( PLASmoothProcessCard(mQGisIface->mapCanvas()) );

【MapTool是如何知道现在用的是什么卡的】

mCurrentTool = mApp->mapCanvas()->mapTool();

void PLAMaptoolCardSettingDialog::onProcessChanged()

{

int curId = cbProcessors->currentIndex();

if(curId>=0)

{

PLAMaptoolSmartProcessCard\* card = (PLAMaptoolSmartProcessCard\*)cbProcessors->itemData(curId).toLongLong();

if(mCurrentTool)

{

mCurrentTool->insertSmartProcessCard(card);

}

updateProcessSettings(card);

updateMaptoolInfo();

}

}