1. Qt里的按钮形状不美观，为此做过初期研究：可以在Qt 设计界面geometry调整其几何位置、按钮的长和宽属性；在font里设置Bold时按钮的内容字体为黑体，Italic为斜体；在styleSheet里用CSS代码可以更改自定义的按钮外观(目前还没学会)，桌面有text文档(CSS控件美化)；
2. 出来的ossimplanet.exe上的经纬度字体太小，可以改变字体的大小：在ossimplanetQtMainWindow里是找不到如”Eye:”这样的关键字的；目前我所获知和猜测的窗口父子关系：theGlWidget :tabwidget; theHudLayer:tabwidget; theGlWidget是ossimplanetQtviewer类的对象，类里有球体的声明；球体是ossimplanetviewer类；theHudLayer是ossimPlanetLatLonHud类，该类可看做是tabwidget窗口的最顶层，类有私有变量定义了出现在我们面前的经纬度的字体大小；因此可以找到ossimPlanetLatLonHud类的定义并修改默认的字体大小，我设为50.
3. Ossimplanet.exe呈现在我们面前的窗口在ossimplanetQt的main里叫mainWindow，在ossimplanetQtViewerWindow里叫theMainWindow，他们引用的是同一个指针。
4. 在实现模型加入和字体标注之后，我觉得还要完善一些：首先是建筑物有些字体没有，有待继续加入文字；第二个点击和鼠标之后弹出个介绍框，一旦鼠标移走该位置，对话框自动消失；第三，考虑到用户的更好地体验，针对以上作如下改善：当鼠标移动到某一建筑物上或建筑物对应的文字上，改文字自动变大，当鼠标点击文字时，弹出一个介绍框，一旦鼠标在另一位置单击时，该对话框自动消失。
5. Exe黑屏上的经纬度字体可能开始会太小，通过2改字体的大小时可能字体会不端正，即出现斜体之类的情况，还有，模型如果加载不上时，可在ossimPlanetQtApplication.cpp里的617行左右改某个变量的路径为自己3ddll相应的路径，以便使地图层支持3d模型。
6. 接下来集体换成了张旭做的需求界面：接下来需要学会下面的事情：1，学会从一个目录里读取一个或一类文件；2，学会并熟练地掌握添加新界面以及在新界面使用当前项目资源的方法；3，逐渐实现需求界面的各个功能；。。。其先后次序是没有严格规定的，可按情况逐一实现。
7. 今天又对有些问题有了新的了解：在新建一个ossimPlanetQtMainWindow类时，执行完构造函数后，并没有把theGlWidget立马加进去二是接着调用一个自带的init()函数，将theGlWidget加进去，并同时对theGlWidget进行相应的初始化，比如新建一个球体viewer并把应用赋给theGlWidget自带的一个球体类theViewer，以后操作的球体就是这个theViewer，不过没直接调用它，二是再将它的引用赋给一个变量viewer进行操作。
8. 在创建一个新的ui后，如果要你自己写的继承这个ui类的一个对话框类，如果你要对此类设置一些slots，则最好不用创建时自带的单重继承，而代替使用多重继承，同时记得删除一些声明，最重要的是，要在此类的构造函数里setupUi(this);否则执行new 此类时会是个空对话框，和ui完全不同。但是可以在此类的.cpp里使用ui里的多个变量，因为此类已经继承了ui,这点和单继承不同，单继承是在执行完构造函数之后才能使用一些ui里的变量，在此之前，是不能直接使用ui里的变量的。
9. 今天有了个突破性的进展：在成功加了一个ui界面后，怎么调用有球体的那个主窗口呢？其实很简单，假如你这个对话框要调用主窗口，那么首先得包含主窗口类的头文件，其次声明一个主窗口指针，叫theMainWindow,通过函数setMainWindow把主窗口指针赋给theMainWindow，以后就可以在这个类里通过theMainWindow调用主窗口了；但这还没完，同时还要在ossimPlanetQtMainWindow里包含你的那个窗口类头文件，然后可以在要显示这个窗口的函数比如： ：：theViewerWindow()里调用子窗口的函数setMainWindow来对theMainWindow进行赋值；当然，如果你希望你的那个子窗口类是主窗口的子类的话，可以这样：ossimPlanetQtViewerWindow \*theViewer = new ossimPlanetQtViewerWindow(this);当然你可以见好几个窗口，只要他们要调用主窗口，就可以这样做；继续发掘中。。。
10. 继续第9条，我发现在主窗口类实现文件里，有个init()函数用来调用theViewerWindow()函数，但是在主窗口类头文件里把这个函数作为public slots 然后用emit theViewerWindow()，说明emit不知可以用来触发信号(signal)也可以触发槽(slot)。
11. Kmz.xml记录模型的路径、经纬度等信息，xml\building.xml记录模型标注、经纬度等信息，Images里有模型的图片链接，点击字体时弹出图片。
12. 在模型和地标的map里，如果键值即名称是一样的话，当鼠标点击左键模型和地标的任何一个，询问是否删除模型和地标的图片对话框都将弹出来，这是因为鼠标位置在两个类里都有监听，鼠标的位置总是可以预先判断这个点的name，这造成模型和地表指针同时被锁定。另外，为了不使对话卡U难过反复弹出，可以对模型和地标占有的多个点组成的数组只选择其中一个点操作，以避免for循环造成的对话框反复弹出现象。
13. 左键单击方案元素，只要单击范围属于该元素，就被选择。右键单击方案元素，只要范围属于该元素，就附带一个功能框，能重命名、删除查看信息。

在选中结果一栏，左键单击4个顶级元素能展开；展开后的每个元素右键单击能查看信息。pS.学习QTreeWidget的QPoint、index；并和event的QPoint比较。

上面QPoint任务仅供学习，在QTreeWidget里没有实战意义，因为mouse事件是基于Widgetl类的，而QTreeWidget是嵌套在ViewerWindow类里的，而事件虚函数只在ViewerWindow里重载，点击ViewerWindow会捕获到鼠标事件，点击QTreeWidget时是捕获不到的！！（应该说是能被QTreeWidget捕获到但是没反应，因为只是调用了基类的鼠标事件虚函数）。所以点击QTreeWidget的时候没有点击在ViewerWindow的表层上，故而捕获不到鼠标的QMouseEvent\* event。到这里只能通过itemClicked()消息来选择item了。现在这种情况，我们不知道itemClicked是怎样被触发的，但肯定是通过鼠标事件虚函数起作用的，这点只在QTreeWidget类里定义。

treeWidget\_selectList->setStyleSheet(QString::fromUtf8("QTreeView{\n"

" background: #c8dcf1;\n"

" border-left: 2px solid gray;\n"

" border-top: 2px solid gray;\n"

" border-bottom: 2px solid gray;\n"

" border-right: 2px solid gray;\n"

" border-top-left-radius: 4px;\n"

" border-top-right-radius: 4px;\n"

" color: rgb(0,0,127);\n"

" font: Bold;\n"

" show-decoration-selected: 1;\n"

"}\n"

"\n"

"\n"

"QTreeView::branch:has-siblings:!adjoins-item {\n"

" border-image: url(D:/ossim\_image/vline.png) 0;\n"

" }\n"

"\n"

" QTreeView::branch:has-siblings:adjoins-item {\n"

" border-image: url(D:/ossim\_image/branch-more.png) 0;\n"

" }\n"

"\n"

" QTreeView::branch:!has-children:!has-siblings:adjoins-item {\n"

" border-image: url(D:/ossim\_image/branch-end.png) 0;\n"

" }\n"

"\n"

" QTreeView::branch:has-children:!has-siblings:closed,\n"

" QTreeView::branch:closed:has-children:has-siblings {\n"

" border-image: none;\n"

" image: url(D:/ossim\_image/branch-closed.png);\n"

" }\n"

"\n"

" QTreeView::branch:open:has-children:!has-siblings,\n"

" QTreeV"

"iew::branch:open:has-children:has-siblings {\n"

" border-image: none;\n"

" image: url(D:/ossim\_image/branch-open.png);\n"

" }\n"

" QTreeView::item:selected:!active {\n"

" background: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1, stop: 0 #6b9be8, stop: 1 #577fbf);\n"

" }\n"

"QTreeView {\n"

" show-decoration-selected: 1;\n"

" }\n"

"\n"

" QTreeView::item {\n"

" border: 1px solid #d9d9d9;\n"

" border-top-color: transparent;\n"

" border-bottom-color: transparent;\n"

" }\n"

"\n"

" QTreeView::item:hover {\n"

" background: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1, stop: 0 #e7effd, stop: 1 #cbdaf1);\n"

" border: 1px solid #bfcde4;\n"

" }\n"

"\n"

" QTreeView::item:selected {\n"

" border: 1px solid #567dbc;\n"

" }\n"

"\n"

" QTreeView::item:selected:active{\n"

" background: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1, stop: 0 #6ea1f1, stop: 1 #567dbc);\n"

" }\n"

"\n"

" QTreeView::item:selected:!active {\n"

" background: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1, st"

"op: 0 #6b9be8, stop: 1 #577fbf);\n"

" }\n"

"QTreeView QHeaderView:section\n"

"{\n"

"\n"

" color: rgb(0,0,127);\n"

" font: Bold;\n"

" text-align: center;\n"

" background: qradialgradient(cx: 0.4, cy: -0.4,\n"

" fx: 0.3, fy: -0.4,\n"

" radius: 1.35, stop: 0 #f7fff7, stop: 1 #6895e5);\n"

"}\n"

" QScrollBar:vertical {\n"

" border: 2px solid grey;\n"

" background: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1,\n"

" stop: 0 #E1E1E1, stop: 0.4 #DDDDDD,\n"

" stop: 0.5 #D8D8D8, stop: 1.0 #D3D3D3);\n"

" width: 15px;\n"

" margin: 22px 0 22px 0;\n"

" }\n"

" QScrollBar::handle:vertical {\n"

" background: qradialgradient(cx: 0.4, cy: -0.4,\n"

" fx: 0.3, fy: -0.4,\n"

" radius: 1.35, stop: 0 #f7fff7, stop: 1 #6895e5);\n"

" min-height: 20px;\n"

" }\n"

" QScrollBar::add-line:vertical {\n"

" border: 2px solid grey;\n"

" background: qradialgradient(cx: 0.4, cy: -0.4,\n"

" fx: 0.3, fy: -0.4,\n"

" radius: 1.35, stop: 0 #"

"f7fff7, stop: 1 #6895e5);\n"

" height: 20px;\n"

" subcontrol-position: bottom;\n"

" subcontrol-origin: margin;\n"

" }\n"

"\n"

" QScrollBar::sub-line:vertical {\n"

" border: 2px solid gray;\n"

" background: qradialgradient(cx: 0.4, cy: -0.4,\n"

" fx: 0.3, fy: -0.4,\n"

" radius: 1.35, stop: 0 #f7fff7, stop: 1 #6895e5);\n"

" height: 20px;\n"

" subcontrol-position: top;\n"

" subcontrol-origin: margin;\n"

" }\n"

" QScrollBar::up-arrow:vertical, QScrollBar::down-arrow:vertical {\n"

" border: 2px solid grey;\n"

" width: 3px;\n"

" height: 3px;\n"

" background: white;\n"

" }\n"

"\n"

" QScrollBar::add-page:vertical, QScrollBar::sub-page:vertical {\n"

" background: none;\n"

" }\n"

""));

14.最后还是写了个新类，叫ossimPlanetQtTreeWidget，继承QTreeWidget，并且在这个类里重载鼠标事件虚函数mouseReleaseEvent(QMouseEvent)，这样当点击ossimPlanetQtTreeWidget的item时候ossimPlanetQtTreeWidget的itemClicked(QTreeWidgetItem\* item,int column)消息不会发出，该消息来自于父类QTreeWidget。在新类ossimPlanetQtTreeWidget的mouseReleaseEvent(QMouseEvent)里鼠标左键释放的时候emit itemClicked(QTreeWidgetItem\* item,int column)消息。而且写了个新的signal，contextMenu(QTreeWidgetItem\* item,int column))来代替QWidget的一个右键菜单事件消息(消息参数是const QPoint&)。所以当右键释放的时候有2中消息发出，一种是QWidget的一个右键菜单事件消息customContextMenuRequested ( const QPoint & pos )，一种是contextMenu(QTreeWidgetItem\* item,int column))。