# 菜单栏模块

## 思路

顶部的导航窗格可以用qtab做，但是里面的图片的选择我觉得可以用QGraphics做。然后sceneView可以有一些比方说设置横隔线、添加组件的函数和功能。对于内部的组件，需要添加点击事件，点击可唤醒dock widget，当离开那么就把dock widget 的生命给结束。

## 拓展

一个是优化tab栏和主窗口的边界，不要让这个分界太突兀

一个是添加一个样式，可以指向当前所在的tab。

# 地图

地图实际上是一个widget，里面有scene和view成员。

要想实现和图层管理器的数据共享，最好的方式是在客户端中加载这两个类，然后通过图层管理器定义的dump方法加载数据。

从图层到地图还有一个重要的流程，就是投影。但这是后面的工作了

# 要素、图层、图层管理器

## 思路

要素类需要自己修改数据。提供开放的接口来供修改需求。包括修改数据原本的坐标信息（移动，缩放，旋转，编辑折点）。这里还需要能通过对话框编辑几何图元的其他属性。

要素在图层中，图层在图层管理器中，他们分别是多对1，多对1的关系。

我这里这么做的原因时如果通过父类指针调用子类的方式，那么就无法调用到子类的成员。

所以只能这么做。

这里的要素需要以指针的方式存在。

图层负责容纳数据，有多少中要素，对应的就有多少种图层。

图层管理器也是一样的，有多少种图层，对应的就应该有多少种图层容器。

要素实现自己的方法，图层通过实现批量修改和单个修改开放接口，这里还要包括批量修改zValue。实际上这是一种**中介模式**。

图层管理器只有一个功能，那就是按照既定的顺序给每隔图层分配zvalue，并通知做出改变。

# 菜单栏 编辑模块

## 思路

编辑实际上是一种命令，但是编辑又分为很多种类，编辑的前置条件是选择。

选择是将所有我们想选中的对象标识为选中的过程。选择可以选择一个，也可以选择很多个。

当选择一个时，我们可以对选中项进行任意操作。当选中多个时，我们不能进行编辑操作，但是可以移动、缩放、删除选中项。选中实际上是将所有符合条件的选项纳入一个mapobject\*的数组中，编辑的特点是我们不需要增加选项，但是需要快速的随机访问操作。

编辑过程中会有取消编辑操作，因为我们传入的是指针，直接对指针编辑会导致数据改变无法复原。一个办法是利用**原型模式**对目标项进行复制。

编辑结束后

编辑结束后如果想撤回编辑，那么我们需要一个备忘录模式，可以撤回我们的修改。

对于编辑，一次选中可以让很多item都被选中，但是只能进行移动 缩放 删除 这些功能，要实现这个功能，我们需要重写itemAt和选择函数。重写的方向时改变返回值，实际上这已经不是一种重写了。

QGraphicsItem \*QGraphicsScene::itemAt(const QPointF &position, const QTransform &deviceTransform) const

{

const QList<QGraphicsItem \*> itemsAtPoint = items(position, Qt::IntersectsItemShape,

Qt::DescendingOrder, deviceTransform);

return itemsAtPoint.isEmpty() ? nullptr : itemsAtPoint.first();

}

选择在地图模块这里的选择实际上也是一种命令，信号在鼠标松开时触发。

这里的对对象的修改需要命令模式。

因为我们使用的是对象类的指针，任何直接的修改都会导致内容的改变，就无法实现取消修改的功能，所以我们需要**原型模式**对选择到的对象进行拷贝。