**ViewPort与Window概念**

以下的信息我也是查找资料和验证代码得出的结论，不一定非常准确。

在Qt中，ViewPort描述实际显示数据在客户区或者屏幕的哪个地方，描述它坐标是物理坐标(一般单位是像素)。Window是把实际显示的区域映射成一个绘制的逻辑区域，里面的坐标为逻辑坐标。逻辑Window大小可以随意更改，但是ViewPort大小却有限制，就是在有些情况设置的大小不能超过实际的大小。

一般情况下也不会去修改ViewPort，在某些特殊情况下修改它反而是简洁的办法，例如分屏，显示区域整体移动等。

# 控件或者窗体的viewPort与Window

Qt在缺省情况下，viewPort与Window大小相等，对于控件或者窗体默认为640\*480像素大小。但是在实际情况下，根据不同的Widget类型可能会有不同的缺省大小。实现者一般会在实现Widget的构造函数使用resize函数设置大小(注意该函数实际上是设置window大小)。**注意：在构造函数里面使用resize函数设置大小并不会立即生效，只是设置了参数，因为此时Widget显示设备还没有初始化。**

在Widget初始化时，会使用设置的大小把viewport和window设置为实际的大小(或者不同类型Widget的初始值，设置大小不能超过屏幕分辨率)。例如在调用Widget的Show()函数就会初始化数据。所以如果在初始化之前获取ViewPort和Window的大小，只是一个登记的数字(该数字也许是缺省的，也许是人为设置的)，并不代表实际的区域。

通过resize函数修改控件或者窗体的大小，如果要求修改的区域超出了实际的值，则最后使用的实际的大小。例如在构造函数里面调用resize(4000,4000)设置为，但是最后的值可能是1920\*1080相近的数字(跟显示器分辨率有关)。

# 图形绘制区域的viewPort与Window

对于图形绘制，一般使用QPainter对象在客户区里面绘制，设置ViewPort大小并不受实际的分辨率或者所在的控件实际大小限制，ViewPort大小就可以随意修改。同样setWindow函数也可以随意修改，不管你设置你的逻辑区域多大，都会对应着设置的ViewPort大小。有了这个特性，我们对于部分的缩放和平移可以通过ViewPort大小实现。如果把ViewPort设置为原来的一半，则显示的效果就是整体图形缩小到原先的1/2。不过修改ViewPort的大小并不是万能的办法，Qt还提供其他设置转换的更方便的接口。

setWindow函数设置的起始点坐标对应的是设置后ViewPort的起始点。所以如果你绘制的图形坐标不在setWindow函数的区域里面，则不会显示，或者显示部分(当然如果设置ViewPort只是客户区的一部分，则有可能显示完整图形)。

绘图函数所绘制的坐标都是window里面的逻辑坐标，有了viewPort和window的概念，这样可以让图形绘制与显示独立。尽管如此，但是我们在有些情况下还是需要把两者关联起来，例如最大化或者手动拖动边框修改的Widget大小。我们应该始终保持两者的长宽比是相同的，否则图元可能发生扭曲或者显示不出意料的结果。

控件的ViewPort与Window和客户区的ViewPort与Window是两个不同的对象。Widget本身的viewport()函数获取的是控件本身的。通过QPainter对象setViewport设置的是控件客户区的。所以如果设置setWindow和setViewport是通过QPainter对象来设置客户区绘制区域，获取设置后的大小也要通过QPainter对象来获取。如果通过QPainter对象设置了大小，但是最后验证却是通过Widget本身的viewport()函数来获取大小，则两者数据不同的。

# 附加

Qt的viewPort与Window坐标系统都是X轴向右，Y轴向下。所以跟人的习惯性的坐标系统相违背。在绘制时，需要把Y坐标做处理，X坐标不需要处理。由于从Window坐标到viewport坐标起始就是矩阵变换，如果对矩阵熟悉，可以直接设置矩阵(如平移到新的原点，调用scale(1,-1)把Y轴转向，这个是理论)。以下是处理的部分办法。

1、为了简单，假设viewPort与Window相等，他们的高度为H，则新的坐标转换公式如下：

Ynew = H – Yold。

2、直接使用setWindow(0,H,w,-H)把Y轴改成向上