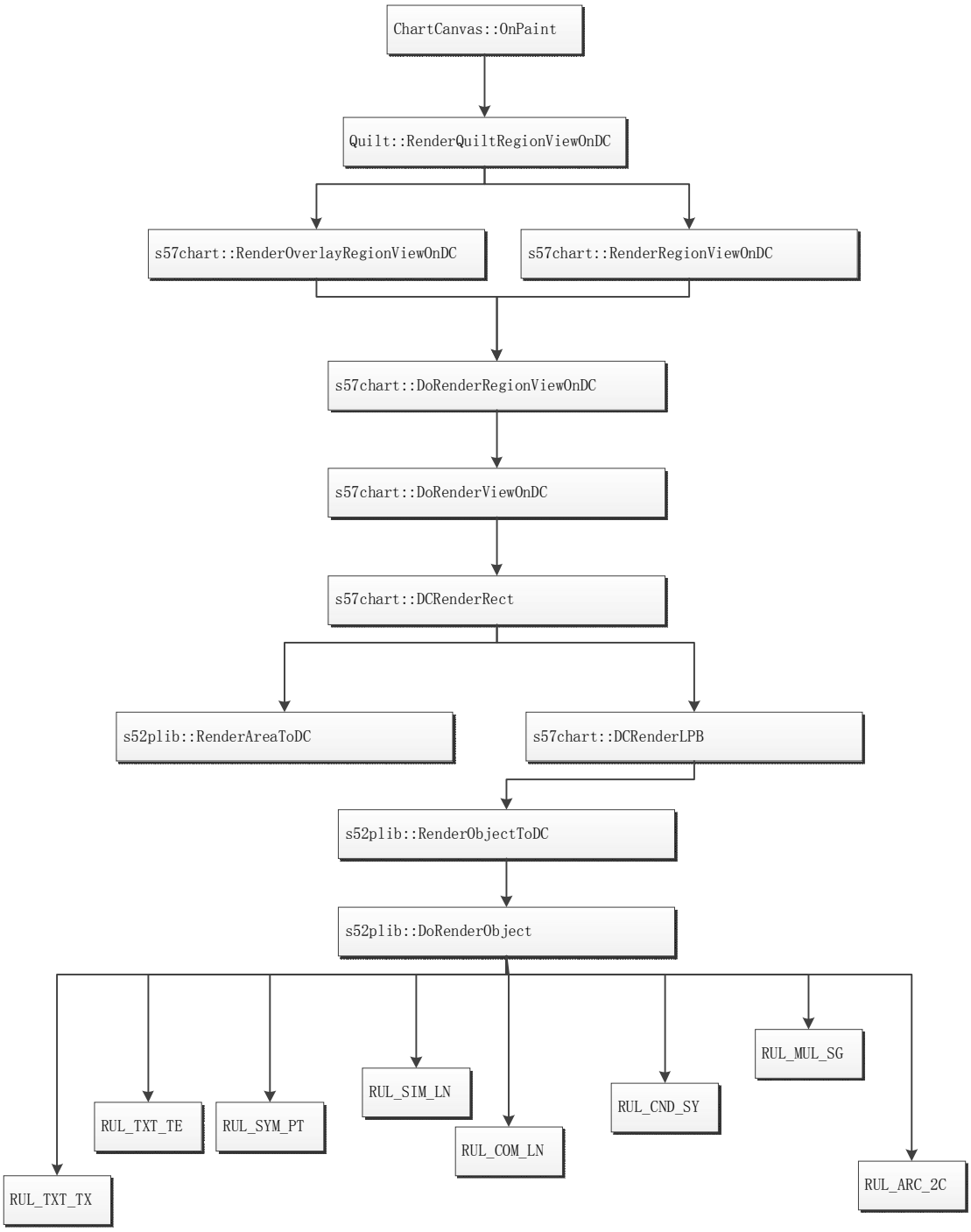
OpenCPN 探索—海图显示

前面几章我们依次介绍了 S57 文件、SENC 转换、SENC 读取、S52 介绍和海图拼接。经过前面几张的数据积累，本章将基于前面的数据阐述海图显示过程。海图显示是一个相对靠后的复杂绘图过程，它依赖于 S57 文件的海图数据和 S52 文件的显示标准。在 OpenCPN 中，这一个复杂的绘图过程如同流水线一般从上之下依次排开，中间没有设立动态库，也没有设定线程。在这样一个流水线处理过程中，处理效率显得极其重要。下面我们来分析一下，它的具体实现过程。

OpenCPN 的显示流程—以 S57 显示为例

海图显示流程图：



绘图的起始点为什么会是ChartCanvas::OnPaint呢，因为OpenCPN的绘图时流水线式的，绘制海图的触发条件是OnPaint消息。下面让我们详细分析一下这些函数以及这些函数的实现方法。分析中将显示作为重点，对其中算法比较感兴趣的可以查看源代码自行分析。

各个函数解析：

1：ChartCanvas::OnPaint函数中代码非常长，但是具体和绘制海图相关的只有中间调用Quilt的一段

（此处说的绘制方式为Quilt方式，非Quilt方式不太一样，感兴趣的可以研究一下）。函数开头一大堆代码用于更新区域的计算，这里是OpenCPN的对计算区域的控制，是它提高效率的一个体现。如果我们仅仅是先达到显示，这里可以抛弃前面的计算，直接将屏幕区域传递到下面进行绘图；函数的末尾一大堆除了拷贝结果到图上以外别的都是OpenCPN的界面显示代码，可以不管。

所以这个函数中只需要注意m_pQuilt->RenderQuiltRegionViewOnDC(temp_dc, svp, chart_get_region)和dc.Blit(rect.x, rect.y, rect.width, rect.height, &mscratch_dc, rect.x, rect.y)。

2：Quilt::RenderQuiltRegionViewOnDC函数中，遍历当前视口中的所有图信息（通过海图拼接得到的数据）。代码中可以看到有两个循环，第一个循环用于绘制非overlay的图，第二个循环用于绘制overlay的图。这样做是为了区分叠加顺序，在S57图中二者的绘制是一样的。该函数后面一系列的拼接和区域处理可以不用管，前面的遍历过程都是先判断海图文件与更新区域的相交区域，然后将相交区域传递给海图对象，后面海图对象将根据更新区域重新绘制。

3：s57chart::RenderOverlayRegionViewOnDC和s57chart::RenderRegionViewOnDC。前者只有在S57类中才有，而且与后者一样。两个函数，一个用于绘制overlay一个绘制非overlay。

4：s57chart::DoRenderRegionViewOnDC 该函数是海图对象内部的绘制函数。各个海图对象将根据海图数据符号化形成的二维对象数据进行绘图。该函数中前面有一些显示设置，后面有一些图形的处理和参数的保存，这些对于显示没有太多影响，主要用于显示的函数是bool bnew_view = DoRenderViewOnDC(dc, VPoint, DC_RENDER_ONLY, force_new_view)。

5：s57chart::DoRenderViewOnDC函数中，前面有很长一段用于显示中心的判断和计算，计算的目的是为了形成更新区域和新的dc，从代码中可以看出新的dc是由旧的dc拷贝而来，这个拷贝是为了弥补更新区域较小的缺陷。如果前面你直接传递过来的是更新全屏，这个地方则也不需要考虑拷贝旧图像信息的问题。

6：s57chart::DCRenderRect函数中，前面一段用于预设更新区域的背景，可以直接拷贝过来使用。

后面分为两块：

一块用于绘制区域型对象：中间用到了符号化的数据。

```
for( i = 0; i < PRIO_NUM; ++i ) {
        if( ps52plib->m_nBoundaryStyle == SYMBOLIZED_BOUNDARIES ) top = razRules[i][4]; // Area Symbolized Boundaries
        else
            top = razRules[i][3];           // Area Plain Boundaries

        while( top != NULL ) {
            crnt = top;
            top = top->next;                // next object
            ps52plib->RenderAreaToDC( &dcinput, crnt, &tvp, &pb_spec );
        }
    }
```

一块用于绘制物标型对象：

```
DCRenderLPB( dcinput, vp, rect );
```

中间的一些拷贝都是为了填充背景。

7：s52plib::RenderAreaToDC函数用于绘制区域。区域绘制中包含三种类型的区域（AC颜色填充、AP模式填充、条件符号化）；AC颜色填充和AP模式填充在显示颜色上有些区别，绘制AC和AP的函数分别是RenderToBufferAC(rzRules, rules, vp, pb_spec)和RenderToBufferAP(rzRules, rules, vp, pb_spec)。条件符号化则是根据条件判断是否显示AC和AP。

8：s57chart::DCRenderLPB函数用户绘制非区域对象。razRules数组中第一维数组表示的是类型，第二维数组表示的是基准数据。函数中两层循环，外层循环是遍历所有类型，内层有三个循环，分别是绘制区域、线和点的。其中区域有两种基准数据，点也有两种基准数据。

9：s52plib::RenderObjectToDC函数用于绘制对象，里面就一调用，具体实现在调用函数中。

10：s52plib::DoRenderObject函数用于实现绘制功能。函数前面是一段判断条件，主要判断是否显示用的，判断是否在可视区域内，避免在非显示区浪费绘图动作。最后根据类型分别转向各个有效绘制函数中。

11：各个绘图函数：

RUL_TXT_TX和RUL_TXT_TE----文本绘制
RUL_SYM_PT----------点绘制
RUL_SIM_LN----------简单线
RUL_COM_LN----------复杂线
RUL_MUL_SG----------水深点
RUL_ARC_2C----------弧线
RUL_CND_SY----------条件符号化，根据条件绘制上述信息

底层的复杂绘制采用HPGL实现，其他的都是WX的绘制函数，感兴趣的可以仔细查看，只为显示的可以直接拷贝OpenCPN的绘制方法，无需修改。