OPENCPN 空间对象(Spatial object)构建

1.1 概述

本文针对 s57 格式海图中空间对象从海图原始文件 ENC 文件读取到写入 SENC 文件之前整个演变过程予以比较详细的分析。

1.2 空间对象从 s57 ENC 到 GDAL 几何对象

首先，谈到空间对象，需要一个空间对象的准确定义。标准给出的空间对象的定义为：一个包含真实世界实体的位置相关信息的对象。它在 s57 标准中对现实世界实体建模中与特征对象（Feature Object）一同构成对现实实体的描述。空间对象不能单独存在，它必须依赖于一个特征对象。而特征对象一般要引用一个或者多个空间对象，也可以不引用空间对象。

目前，标准只定义了向量模型这一种模型来描述二维的空间对象。包括 0,1,2 维的对象：节点（Nodes），边（Edge），面（Face）。利用这些对象，可以采用四种拓扑结构：无拓扑（Cartographic spaghetti），链状拓扑（Chain-node），平面图形拓扑， 全拓扑。而 S57 ENC 采用了链状拓扑结构，也就是说只利用了节点和边对象，而没有面。

S57 ENC 文件格式符合 ISO/IEC 8211 标准数据描述文件格式。标准的具体格式不在此处展开。一个空间对象，包括点， 边，或者面（ENC 中不包括面）在 ENC 文件中的存储结构为：

```
Vector record
 |
 |--0001 (1)   - ISO/IEC 8211 Record Identifier
    |
    |--VRID (4)   - Vector Record Identifier field
       |
       |--<R>--ATTV (*2)  - Vector Record Attribute field
       |
       |--VRPC (3)        - Vector Record Pointer Control field
       |
       |--<R>--VRPT (*5)  - Vector Record Pointer field
       |
       |--SGCC (3)        - Coordinate control field
       |
       |   alternate coordinate representations
       |
       *--<R>--SG2D (*2) - 2-D Coordinate field
       |
       *--<R>--SG3D (*3) - 3-D coordinate (Sounding Array) field
       |
       *--<R>--ARCC (5)  - Arc/Curve definitions field
                  |
                  |   alternate arc/curve definitions
                  |
                  *--<R>--AR2D (3*2)    - Arc coordinates field
                  |
                  *--<R>--EL2D (5*2)    - Ellipse coordinates field
                  |
                  *--<R>--CT2D (*2)     - Curve coordinates field
```

这里说明其中几在 OpenCpn 中使用的关键字段，其它字段或子字段请参考 s57 标准

文档。

VRID：空间对象的标示符字段，该字段中 RCNM 和 RCID 子字段组合在当前 ENC 文件中唯一标示了该空间对象。RVER 子字段为空间对象版本字段，RUIN 为更新指令字段，这两个字段主要在自动更新中使用。

ATTV：空间对象属性字段。

VRPC：自动更新控制字段，该字段只出现在自动更新文件中。

VRPT：指针字段，存储了空间独享之间的引用关系，指向该空间对象引用的其它空间对象。其中最关键的是 NAME 子字段，它给出了所引用的空间对象的唯一标示。其实就是 RCNM 和 RCID 的组合。

SGCC：坐标字段控制字段，只出现在自动更新文件中。

SG2D， SG3D： 2d, 3d 坐标字段，记录了 2d 或者 3d 的坐标值(整数值，考虑到不同平台对浮点数的处理方式不同，所以采用了整数值来记录坐标值，在实际本地处理时要除一个转换因子，该因子在 ENC 文件的对象中有记录)。其中 3d 坐标中 YCOO，XCOO 子字段表示坐标 2d 位置，VE3D 表示在该位置出现的量值，比如水深值。

所有的空间对象都以这种结构存储在 ENC 文件中。下面我们看 Opencpn 如何从 ENC 中读取这些数据。

Opencpn 中的 S57Reader 类正是用来读取 ENC 文件的。S57Reader 中封装了 DDFModule 类，该类负责实际 ENC 文件的实际读取操作。在 Opencpn 中读取 ENC 其实非常简单，用 ENC 的文件路径构造一个 S57Reader 对象，然后调用 S57Reader 的 Open 方法，此时文件指针其实已经指向了 ENC 中的 DR（已经完成了对 DDR 的读取（DDR 相当于 ENC 文件的头，包含一些描述性的内容，不包含实际的海图数据)），每一个 DR（Data Record）对应于 57 模型的一个 Object，所以实际的海图数据都包含在 DR 中。Open 函数调用结束之后，文件指正便指向了第一个 DR 的起始处。然后调用 S57Reader 的 Ingest 函数，该函数执行完毕之后所有的 ENC 数据实际上已经全部读取完毕。按照不同的类型存放在 S57Reader 的 oVI_Index, oVC_Index, oVE_Index, oVF_index 以及 oFE_Index 成员中。每一条 DR 在内存中以 DDFRecord 对象存在。

如此，ENC 已读取完毕，所有的空间对象都已被读入内存。下面的任务是将内存中的空间对象转换存储至 SENC。在 Opencpn 中 SENC 文件的后缀为.s57。

空间对象存储至 SENC 的过程在 s57chart 的 BuildSENCFile 函数中完成。

我们在前面提到过，空间对象不能单独存在，必须依赖于特征对象而存在，也就是说一个空间对象肯定被某个特征对象所引用。考虑到文件传输效率，在 ENC 中任何空间对象都不会出现重复，在当前这种建模方式下，必然会出现相同的空间对象在多个特征对象中出现的情况。在 ENC 中通过建立引用关系来实现一个空间对象被多个特征对象引用的状况。

而 Opencpn 在构建 SENC 文件之前将这种引用关系都进行了解引用。将空间对象纳入特征对象，还原了一个完整的现实世界对象（Object）。这个被还原的对象在 Opencpn 中用类 OGRFeature 来描述。OGRFeature 中类型为 OGRGeometry 的成员 poGeometry 即描述 Object 中的空间对象部分。这个 OGRGeometry 对象如何构造出来？Opencpn 中的

AssembleFeature 函数完成了这个工作。

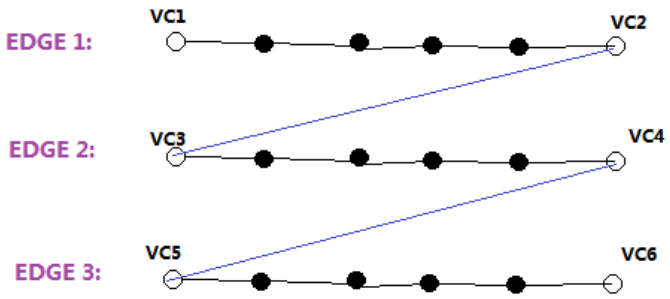
AssembleFeature 所做的事情就是利用 Ingest 函数的主要输出：oVI_Index, oVC_Index, oVE_Index, oVF_index 以及 oFE_Index （存放了 DDRRecord 对象数组），来生成一个 OGRFeature 对象。我们将视线转向 OGRFeature 对象的空间对象构造部分。在 AssembleFeature 函数中调用 AssembleSoundingGeometry，AssemblePointGeometry，AssembleLineGeometry，AssembleAreaGeometry 这四个函数来构造空间对象部分。

我们分析一下 AssemblePointGeometry 和相对复杂的 AssembleAreaGeometry，AssembleLineGeometry。

首先来看 AssemblePointGeometry：我们需要为某个特征对象构建其对应的点空间对象，第一步要知道特征对象对应的空间对象是哪一个，这个信息在特征对象的 FSPT 字段中指出，这个我们在前面有介绍。通过 DDFRecord（一个特征对象的数据目前也存储在 DDFRecord 中）类提供的便捷的方法可以获取 DDFRecord 中的任何一个字段和子字段的信息。有了 FSPT 字段的数据，Opencpn 遍历 oVI_Index 包含的数组（单独的点空间对象包含在 oVI_Index 中，所谓的 isolated node），找到对应的 DDFRecord，然后获取到记录中的 SG2D 字段，其子字段 XCOO，YCOO 即为点的坐标值数据。用这两个坐标数据构造一个 GDAL 的点几何对象 OGRPoint，这便是 OGRFeature 的空间对象部分。这样一个点对象的构建就完成了。

下面是 AssembleAreaGeometry，在这之前我们需要对线和面对象的构造做一些说明。

其实是链状拓扑的相关内容。ENC 中只是用了点，和边两种对象，是用了 VI，VC，和 VE，并没有使用 VF 向量模型。单独的点对象比较直观简单，此处不赘述（水深点相对特殊一些，只是把多个点当做一个点对象看待）。



VI 和 VC 都是节点，两者的区别在于 VI 为孤立点，或者水深点。孤立点用来表示单独的类似于物标的对象。 VC 是 connected node，在 Chain-node 拓扑中，VC 不能独立存在，它只能被 VE 引用用来构成线对象。

VE 用来表示边（Edge）对象， 一个 Edge 由 2 部分组成，起始终止点，和中间坐标点。起始点和终止点的坐标值并没有直接放置在 Edge 对象中，而是通过 VRPT 指针来引用 VC 对象。如上图所示。同时上图展示了一个线空间对象在 Opencpn 中的构成方式，也即是 AssembleLineGeometry 函数中构建线空间对象的过程：

1. 由特征对象的 FSPT 字段，可能存在多个，在特征对象为一条线（PRIM 字段为 2）或面（PRIM 为 3）时，FSPT 字段指向的肯定是一个或者多个 EDGE。

2. 找到 FSPT 字段中的 NAME 字段，其中记录的被引用的 EDGE 对象的唯一标示。
3. 通过该唯一标示在 oVE_Index 中找到所有的 EDGE DDFRecord。
4. 按照 FSPT 字段在特征对象中的出现顺序，找到第一个 EDGE DDFRecord 的 VRPT 字段，该字段一般存在两个，分别为起始或者终止点，至于具体哪个为起始点，哪个为终止点，由特征对象的 FSPT 字段的 ORNT 子字段决定。
5. 将第一个 EDGE 的起始点 VRPT 字段指向的 VC 点作为线空间对象的第一个点。
6. 之后是第一个 EDGE 对象直接包含的坐标点（可能为空），按照其在文件中的顺序添加到线空间对象中。
7. 将第一个 EDGE 的终止点 VRPT 字段指向的 VC 点追加至线空间对象。
8. 取下一个 EDGE 对象，只将其直接包含的坐标点以及终止点 VRPT 字段指向的 VC 追加至线空间对象。（上图中 VC2 和 VC3 是同一个点， VC4 和 VC5 是同一个点，因此该 EDGE 对象不需要追加起始点 VRPT 字段指向的 VC）
9. 重复步骤 8 直至最后一个 EDGE 对象。

经过以上步骤便获得了该线空间对象的所有点，并且按照顺序追加到一个 GDAL 的 OGRLineString 对象中。一个 OGRLineString 对象生成，作为现实对象的空间对象部分。即 OGRFeature 的 poGeometry 成员。

至于 AssembleareaGeometry 与上述过程的差别在于 Area 对象的第一个 Edge 对象的起始点和最后一个 EDGE 对象的终止点是同一个点。另外，Opencpn 在构造 GDAL 的 OGRPolygon 对象（Opencpn 的面对象轮廓）时，只是单纯的将各个 EDGE 对象构成一个 OGRLineString 对象，然后将多个 OGRLineString 对象构成的 OGRGeometryCollection 对象传递给 BuildOGRPolygonFromEdges 函数来构建一个 OGRPolygon 对象，作为 OGRFeatrue 的空间对象部分。

最终的 SENC 文件中存储的空间对象就是将 GDAL 的几何对象（OGRPoint，OGRLineString，OGRPOlygon 等等）以二进制方式直接写入文件 。下面的过程我们将在以后的文章中进行详尽的描述。

西安融成科技有限公司出品

摘选自：www dot opencpn dot cn/OpenCPNDocs/OPENCPNS57 标准介绍.pdf