|  |  |
| --- | --- |
|  |  |

OneGISdeskop平台地图渲染改进

技术方案

苏州中科蓝迪软件技术有限公司

二○一八年一月十七日

文档版本记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 作者 | 日期 | 版本 | 纪要 |
|  | 姚亮 | 2018年1月17日 | V1.0.0 | 支持多粒度对象渲染方案 |
|  | 李坡 | 2018年1月17日 | V1.0.0 | 支持多粒度对象渲染方案 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目 录

[1. 矢量地图渲染及相关类 4](#_Toc6205)

[1.1 QgsVectorLayer 4](#_Toc6082)

[1.2 QgsFeatureRendererV2 4](#_Toc27743)

[1.3 QgsSymbolV2 5](#_Toc18166)

[1.4 QgsSymbolLayerV2 5](#_Toc30748)

[2. QGIS渲染流程 6](#_Toc861)

[3. 改进方案 8](#_Toc8495)

# 矢量地图渲染及相关类

* QgsVectorLayer
* QgsSymbolV2
* QgsSymbolLayerV2
* QgsFeatureRendererV2

QgsFeatureRendererV2 控制着QgsVectorLayer 的“渲染”样式，而具体用什么样式来“渲染”，则是有 QgsSymbolV2来定义的，QgsSymbolLayerV2 是 QgsSymbolV2 的扩展。搞明白这几个类的关系有助于我们后面的理解。实际上，我们基本不会直接用到这几个类，大多数时候是在用它们的子类。

## QgsVectorLayer

QgsVectorLayer 不必多说，只需要知道使用它的方法“setRendererV2()”来绑定它的 Renderer 就好了。

## QgsFeatureRendererV2

来看看矢量图层都支持那些 Renderer，也就是 QgsFeatureRendererV2 这个类的派生关系，如下图

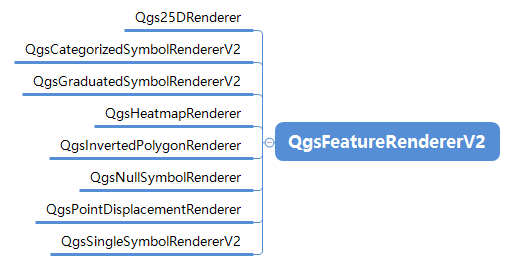


图1.1 QgsFeatureRendererV2类及其派生类

从图上可以看到，这里支持的渲染方式还是蛮多的，GIS的同学看到这些名字应该不会太陌生，就是我们设置图层属性面板时候的那个下拉菜单里面的内容，见下图。

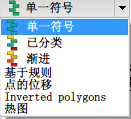


图1.2 层渲染属性设置面板

这里内容太多，就不逐个介绍了，本篇文章中我们只关注一个最简单也是最常用的 QgsSingleSymbolRendererV2 ，也就是我们经常使用在点图层标记上的单个的样式。

## QgsSymbolV2

QgsSymbolV2类就是直接关系到图层显示的“造型”了，也就是在这里设置各种图层样式的属性。还是先来看看它的派生类关系。

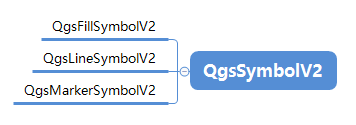


图1.3 QgsSymbolV2类及其派生类

简单说明一下，QgsFillSymbolV2 对应的是多边形矢量图层，QgsLineSymbolV2 对应的是线性矢量图层， 而 QgsMarkerSymbolV2 自然就是对应点矢量图层了。

关于这些类的设置大多集中在“颜色”、“大小”、“透明度”等属性上，每个派生类也都有自己的一些属性，详细的情况还是建议大家看看 API 文档 。

## QgsSymbolLayerV2

QGis扩展了简单的 Symbol 图层样式，将原来单一的 Symbol 变成了“图层”方式，使得样式可定制的自由度开阔了许多， 当然派生关系也多了很多，见下图。



图1.4 QgsSymbolLayerV2类及其派生类

从上图里面可以看到，分别对应这多边形、线和点都有各自的样式图层，将一个矢量图形的图标拆分，基本每个地方都有可定制的余地，都有专门的样式图层类来控制。

# QGIS渲染流程

由于当mapcanvas刷新时，进行重新的渲染工作。所以以类Mapcanvas的refresh()方法为入口进行矢量数据渲染工作的梳理。

1. 调用类Mapcanvas的refresh()方法。通过判断QgsMapSettings(存储地图渲染工作的所有必要信息)中是否存储有渲染工作所必需的设置、是否用户请求暂停渲染、画布是否被冻结、是否已经在进行渲染否的话将该参数设置为true。满足渲染条件进行下一步，否则不进行渲染。
2. 信号连接Mapcanvas的refreshMap()方法。在该方法中，首先，停止所有的在后台进行的地图渲染工作。QgsMapRendererQImageJob该渲染方法在渲染工作正在进行时就可以获取渲染图片；然后，从全局的配置文件、工程文件和QgsMapSetting中获取表达式上下文，并全部赋值给QgsMapSetting。再后，选择并行渲染还是顺序渲染。并将处于编辑状态的图层进行运存的缓冲区存储。

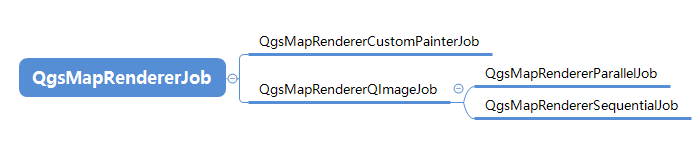


图2.1 渲染类及其集成关系

1. 多线程渲染->QgsMapRendererParallelJob(QgsMapSettings)；单线程渲染->QgsMapRendererSwquentialJob(QgsMapSettings)。
2. 进行改进的初步想法主要针对单线程渲染流程的改进。下面是单线程渲染流程QgsMapRendererSwquentialJob(QgsMapSettings)->start()开始渲染在start函数中创建QgsMapRendererCustomPainterJob(QgsMapSettings,Qpainter)调用QgsMapRendererCustomPainterJob->start()。
3. 在4步骤中调用QgsMapRendererJob->prepareJobs()函数获取渲染工作所需要的各个图层渲染所需的上下文信息、图片、渲染器、painter类型、是否缓存、图层id、渲染时间等；然后调用QgsMapRendererCustomPainterJob->doRender();进行渲染工作。在该函数中通过遍历LayerRenderJobs集合，根据各个图层的渲染器进行渲染进行渲染。该工作关注QgsVectorlayerRenderer。
4. QgsVectorLayerRenderer( QgsVectorLayer\* layer, QgsRenderContext& context )调用QgsVectorlayerRenderer->render()做一些准备工作之后调用QgsVectorLayerRenderer::drawRendererV2( QgsFeatureIterator& fit )循环遍历每一个要素，调用QgsFeatureRendererV2::renderFeature( QgsFeature& feature, QgsRenderContext& context, int layer, bool selected, bool drawVertexMarker )；根据QgsSingleSymbolRendererV2::symbolForFeature(QgsFeature,context)获取要素的样式，然后调用QgsFeatureRendererV2::renderFeatureWithSymbol( QgsFeature& feature, QgsSymbolV2\* symbol, QgsRenderContext& context, int layer, bool selected, bool drawVertexMarker )。
5. 在QgsFeatureRendererV2::renderFeatureWithSymbol调用QgsSymbolV2::renderFeature( const QgsFeature& feature, QgsRenderContext& context, int layer, bool selected, bool drawVertexMarker, int currentVertexMarkerType, int currentVertexMarkerSize )对每个geometry进行渲染。

# 改进方案

1. 定义QgsHybirdFeature继承于QgsFeature,增加geometry容器成员变量，用于存储每个要素的不同几何形态，以及每个几何形态显示的比例尺范围。
2. 定义QgsHybirdSymbolV2继承于QgsSymbolV2,混合符号类包括了QgsFillSymbolV2、QgsLineSymbolV2、QgsMarkerSymbolV2类型的成员变量，可以实现点、线、面符号的渲染。
3. 扩展枚举类型定义，QGis::WkbType中增加WKBHybirdGeometry混合几何体，用于provider提供文件中包含的集合体类型。