

Streaming Renderer Maintainability

[David Blasby](#) sent in the following request....

(sending this out quickly for the IRC meeting)

Okay, I've been fixing up the StreamingRenderer the last little while, and here's what I think needs to be done. Most of these are just little clean-up to get rid of the evidence of a long period of people slapping stuff into it.

A) The path through the renderer is a bit confused. For example, where is CRS transformation done? Its actually done by wrapping the FeatureReader in a CRS-transforming feature reader. But, there's also code (now in LiteShape2 - it used to be partially in LiteShape2 and partially in the Graphics2D segment Iterators) that takes a geometry, decimates it, transforms it, then does a pixel-level generalization. The CRS transformation code should be done here since you can save a bunch of complex math by generalizing first. Its also just really confusing to read since there's 3 sets of generalization code! There's other cases of confusion.

This is mostly a clean-up than writing new code - just make the path through the renderer obvious and itemize the steps (AND WRITE DOX). Currently its confusing, mostly because the original renderer was dead-simple and the new one was made by bolting tonnes of extras on.

B) All the Graphics2D Iterators need to be re-written. I just rewrote the LiteIterator (which can get re-used by all the other Iterators). The Iterators are extremely difficult to read.

Once (A) is done, this should be much easier since they seem to be "doing too much" right now.

C) The renderer should be segmented into two classes. The first one should be concerned with "setting things up". These are things like dealing with the MapContext, optimizing the styles, getting optimized FeatureReaders. The second will basically take the FeatureTypeStyle[] (which has the Graphics2D and actual Styles) and FeatureReader and do the actual drawing. This break is currently in the Renderer approximately at the function "processStylers()"/"processSymbolizers()".

This will allow for more code-reuse, and simplify the current (giant) class. Its not too much work since the renderer is conceptually already broken up like this. It will make it very easy to write a parallel renderer that would render more than one layer at a time (good for multiprocessor machines and/or datasets that block while reading from the disk/network/socket).

The end result will be a renderer that is much easier to understand and maintain. I'd bet there will be minor speed improvements too.

Anyone got any time to help in this?