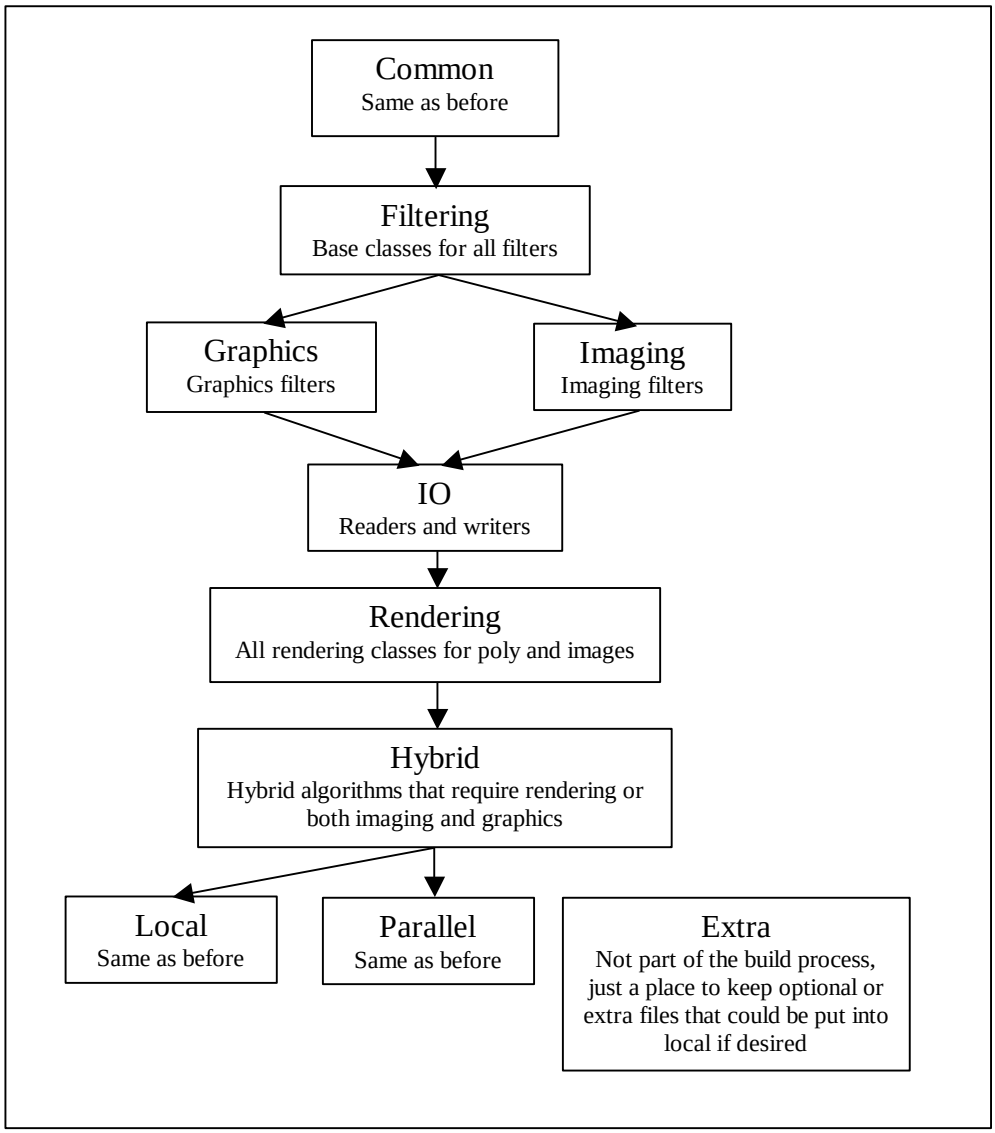
I'd like to toss around the idea of a directory reorganization to address a few problems we've been having with VTK. The problems are:

1. Graphics is so large that the Microsoft compiler cannot incrementally link it. So the build process break it into multiple libraries Graphics0, 1, 2… which is a nasty process and leads random link errors when a new class is added because it caused other classes to "jump" from one library to another. It also leads to problems where someone creates a project that links against all the current graphics libraries 0-4 but then a new class is added creating graphics5 and then all their project files must be modified.
2. Graphics is so large that a MSVC project file cannot be created for it that also wraps Tcl/ Java/ Python. This is because of the number of custom rules required to wrap all the classes in graphics is too large for MSVC.
3. Contrib is a confused directory. It mainly holds classes that require both graphics and imaging because there is no other place to put them. There are also a few "optional" classes that tend to be rarely used. I'd rather break this into two directories, one called hybrid which is designed to hold hybrid algorithms and a second directory called extra which holds a few extra classes that people, if they want to, can add to their local directory.
4. Switching to Cmake (a cross platform build tool developed for the Insight project) requires a smaller graphics directory.
5. Graphics was getting to be a bit of a zoo with over 300 classes in it. Ideally I'd like to break that down into pieces with more like 100 classes or less.

Here is one possible structure that would work. I have done a build with the following structure and everything works. The new graphics directory ends up with about 115 classes in it, with the balance going into filtering, rendering, and IO. All of the directories between (and including) common and IO would always be compiled. The other directories would be optional to build but dependent on other directories as shown. (e.g. you cannot build hybrid without rendering.)


Common
Same as before
Filtering
Base classes for all filters
Graphics
Graphics filters
Imaging
Imaging filters
IO
Readers and writers
Rendering
All rendering classes for poly and images
Hybrid
Hybrid algorithms that require rendering or both imaging and graphics
Local
Same as before
Parallel
Same as before
Extra
Not part of the build process, just a place to keep optional or extra files that could be put into local if desired