VTK/FAQ

General information and availability

What is the Visualization Toolkit?

The **Visualization ToolKit (vtk)** is a software system for 3D Computer Graphics and Visualization.

VTK includes a textbook published by Kitware Inc. (The Visualization Toolkit, An Object-Oriented Approach to 3D Graphics [1]), a C++ class library, and Tcl, Python and Java implementations based on the class library.

For more information, see http://www.vtk.org and http://www.kitware.com.

What is the current release?

The current release of vtk is 5.4.0 (released on 2009-3-26). This release for download available from:

http://www.vtk.org/VTK/resources/software.html

Nightly development releases are available at:

http://www.vtk.org/files/nightly

Can I contribute code or bug fixes?

We encourage people to contribute bug fixes as well as new contributions to the code. We will try to incorporate these into future releases so that the entire user community will benefit from them.

See http://www.vtk.org/contribute.php for information on contributing to VTK.

For some ideas take a look at some of the entries in the "Changes to the VTK API" FAQ section, for example: What changes are being considered for VTK

We now have a bug tracker that allow keeping track of any bug you could find. See BugTracker [2]. You'll need an email to report a bug. To improve the chance of a bug being fixed, do not hesisitate to add as many details as possible, a demo sample code + sample data is always a good idea. Providing a patch almost guarantees that your patch will be incorporated into VTK.

Can I contribute money?

Please don't send money. Not that we think you're going to send in unsolicited money. But if you were thinking about it, stop. It would just complicate our lives and make for all sorts of tax problems.

(Note: if you are a company or funding institution, and would like to fund features or development, please contact Kitware http://www.kitware.com .)

Is there a mailing list or Usenet newsgroup for VTK?

There is a mailing list: vtkusers@vtk.org

To subscribe or unsubscribe to the mailing list, go to: http://www.vtk.org/mailman/listinfo/vtkusers

To search the list archives go to: http://www.kitware.com/search.html

There is also a newsgroup that mirrors the mailinglist. At this point it seems that mirror is down. Mail to the mailinglist used to be posted the newsgroup, but posts on the newsgroup were not sent to the mailinglist. The newsgroup was located at: news://scully.esat.kuleuven.ac.be/vtk.mailinglist

http://www.gmane.org is a bidirectional mail-to-news gateway that carries the vtkusers mailing list. Its located here: news://news.gmane.org/gmane.comp.lib.vtk.user or here: http://news.gmane.org/gmane.comp.lib.vtk.

user.vtkusers mails have been archived since April 2002 and they never expire. You can read and send mails to the vtkusers list but sent mail will bounce back without having subscribed to the list first.

Is the VTK mailing list archived anywhere?

The mailing list is archived at: http://www.vtk.org/pipermail/vtkusers/

You can search the archive at: http://www.kitware.com/search.html

Are answers for the exercises in the VTK book available?

Not anymore.

The answers to the exercises of the textbook used to be maintained by Martin Stoufer (kudos), and will be made available by Kitware in the near future.

Is VTK regression tested on a regular basis? Can I help?

Yes, it is.

You can view the current regression test results at: http://public.kitware.com/dashboard.php?name=vtk

VTK uses Dart to perform builds, run tests, and generate dashboards. You can find more information about Dart at: http://public.kitware.com/Dart/

You can help improve the quality of VTK by supplying the authors with Tcl scripts that can be used as or turned into regression tests. A good regression test will:

1. Cover code that is not already covered.
2. Illustrate a bug that is occuring now or that has occurred in the past.
3. Use data that is on the 2nd Edition book CDROM or use "small" data files or use no data at all.
4. Optionally, produce an interesting result.

Currently almost all regression tests are written in Tcl.

Please send your Tcl regression tests to: mailto:wlorens1@mail.nycap.rr.com

Bill will evaluate them for applicability and integrate them into the nightly test process.

What's the best way to learn VTK?

There are five things you might want to try:

1. Purchase the book The Visualization Toolkit [1] from Kitware Inc.
2. Purchase the book VTK Users Guide [3] from Kitware Inc.
3. Download the source code and/or binaries [4] (available on Windows) and work through the examples (there are 400-500 examples).
4. To learn the innards of VTK, you can attend a VTK course [5] or sponsor a VTK course at your site [6] through Kitware. http://www.kitware.com/products/index.html
5. Buy Bill a beer and get him talking about VTK

How should I ask questions on the mailing lists?

The best online resource for this question is Eric S. Raymond's excellent guide on the topic titled [How to ask questions the smart way [7]]. [Getting Answers [8]] is a good starting point too.

Please do read it and follow his advice. Thanks!

Please also remember the following when you post your messages to the VTK mailing lists.

- Mention the version of VTK you are using and the version of the compiler or scripting language you are using.
- Mention your platform, OS and their versions.
- Include hardware details if relevant.
- Include all relevant error messages (appropriately trimmed of course).
- The lists have a very large number of subscribers (in the thousands), so please keep messages to the point.
- Avoid HTML emails.
- Use a sensible and descriptive subject line.
- Do NOT post large data files or images to the list. Instead put them in your web page and mention the URLs.
- Quote the messages you reply to appropriately. Remove unnecessary details.

When asking a question or reporting a problem try to include a small example program that demonstrates the problem. Make sure that this example program is as small as you can make it, simple (and uses VTK alone), complete and demonstrates the problem adequately. Doing this will go a *long way* towards getting a quick and meaningful response.

Sometimes you might not get any acceptable response. This happens bacause the others think the question has either been already answered elsewhere (the archives, FAQ and google are your friends), or believe that you have not done enough homework to warrant their attention, or they don't know the answer or simply don't have the time to answer. Please do be patient and understanding. Most questions are answered by people volunteering their time to help you.

Happy posting!

How NOT to go about a programming assignment

This is really a link you should read before posting to the mailing list. [This article is an attempt to show these irrational attitudes in an ironical way, intending to make our students aware of bad habits without admonishing them.]

```
http://www.di.uniovi.es/~cernuda/noprog_ENG.html
```

Accessing VTK CVS from behind a firewall

Use the sourceforge project:

http://cvsgrab.sourceforge.net/

Just download the script and type something like:

```
 cvsgrab -rootUrl http://public.kitware.com/cgi-bin/cvsweb.cgi/
 -packagePath VTK -destDir .
         -proxyUser xxx -proxyPassword xxx -proxyHost xxx -proxyPort xx
```

(Thanks to Ingo H. de Boer)

Also cvsgrab support the following option to access a particular branch:

```
-tag <version tag>                 [optional] The version tag of
the files to download
```

For example to get the latest 4.4 branch:

```
 cvsgrab -rootUrl http://public.kitware.com/cgi-bin/cvsweb.cgi/
 -packagePath VTK -destDir .
         -proxyUser xxx -proxyPassword xxx -proxyHost xxx -proxyPort
xxx
         -tag release-4-4
```

Where can I obtain test and sample datasets?

See this page for details on downloading datasets that VTK can read.

Language bindings

Are there bindings to languages other than Tcl?

Aside from C++ (which it's written in) and Tcl, vtk is also bound into Java as of JDK 1.1 and Python 1.5, 1.6 and 2.X. All of the Tcl/Java/Python wrapper code is generated from some LEX and YACC code that parses our classes and extracts the required information to generate the wrapper code.

What version of Tcl/Tk should I use with VTK?

Currently we recommend that you use Tcl/Tk 8.2.3 with VTK. This is the best-supported version combination at this time.

VTK has also been tested with Tcl/Tk 8.3.2 and works well.

Tcl/Tk 8.3.4 has been tested to a limited extent but seems to have more memory leaks that Tcl 8.3.2 has.

Tcl/Tk 8.4.x seems to work well with VTK too, but you might have to change a couple of configuration settings depending on the version of VTK you are using. Check the Does VTK support Tcl/Tk 8.4?.

Where can I find Python 2.x binaries?

All of the Python binaries available on the kitware site are built for Python 1.5.2. This includes the official release VTK3.2 and the nightly builds (as at 2001-07-16).

For Python 2.x binaries, you will have to compile your own from source. It is worth checking the mailing list archives for comments by others who have been through this process.

There are some user-contributed binaries available at other sites. Check the mailing list archives for possible leads. Some win32 binaries for Python 2.1 are available at;

```
http://basic.netmeg.net/godzilla/
```

YMMV...

Why do I get the Python error -- ValueError: method requires a VTK object?

You just built VTK with Python support and everything went smoothly. After you install everything and try running a Python-VTK script you get a traceback with this error:

```
ValueError: method requires a VTK object.
```

This error occurs if you have two copies of the VTK libraries on your system. These copies need not be in your linkers path. The VTK libraries are usually built with an rpath flag (under *nix). This is necessary to be able to test the build in place. When you install VTK into another directory in your linkers path and then run a Python script the Python modules remember the old path and load the libraries in the build directory as well. This triggers the above error since the object you passed the method was instantiated from the other copy.

So how do you fix it? The easiest solution is to simply delete the copy of the libraries inside your build directory or move the build directory to another place. For example, if you build the libraries in VTK/bin then move VTK/bin to VTK/bin1 or remove all the VTK/bin/*.so files. The error should no longer occur.

Another way to fix the error is to turn the CMAKE_SKIP_RPATH boolean to ON in your CMakeCache.txt file and then rebuild VTK. You shouldn't have to rebuild all of VTK, just delete the libraries (*.so files) and then re-run cmake and make. The only trouble with this approach is that you cannot have BUILD_TESTING to ON when you do this.

Alternatively, starting with recent VTK CVS versions (post Dec. 6, 2002) and with VTK versions greater than 4.1 (i.e. 4.2 and beyond) there is a special VTK-Python interpreter built as part of VTK called 'vtkpython' that should eliminate this problem. Simply use vtkpython in place of the usual python interpreter when you use VTK-Python scripts and the problem should not occur. This is because vtkpython uses the libraries inside the build directory.

2002 by Prabhu Ramachandran

Does VTK support Tcl/Tk 8.4 ?

Short answer: yes, but it might require some adjustments, depending on the VTK and CMake versions you are using.

1. The VTK 4.x CVS nightly/development distribution supports Tcl/Tk 8.4 as long as you use a release version of CMake > 1.4.5. Since VTK 4.2 will require CMake 1.6, the next release version will support Tcl/Tk 8.4.
2. The VTK 4.0 release distribution does not support Tcl/Tk 8.4 out-of-the-box.

In either cases, the following solutions will adress the problem. This basically involves setting two definition symbols that will make Tcl/Tk 8.4 backward compatible with previous versions of Tcl/Tk (i.e. discard the "const correctness" and Tk_PhotoPutBlock compositing rule features) :

a) Edit your C/C++ flags:

Run your favorite CMake cache editor (i.e. CMakeSetup, or ccmake), display the advanced values and add the USE_NON_CONST and USE_COMPOSITELESS_PHOTO_PUT_BLOCK definition symbols to the end of any of the following CMake variables (if they exist): CMAKE_CXX_FLAGS, CMAKE_C_FLAGS.

Example: On Unix your CMAKE_CXX_FLAGS will probably look like:

```
-g -O2 -DUSE_NON_CONST -DUSE_COMPOSITELESS_PHOTO_PUT_BLOCK
```

On Windows (Microsoft MSDev nmake mode):

```
/W3 /Zm1000 /GX /GR /YX /DUSE_NON_CONST
/DUSE_COMPOSITELESS_PHOTO_PUT_BLOCK
```

b) or a more intrusive solution:

Edit the top VTK/CMakeList.txt file and the following lines add **at the** top **of this file:**

```
ADD_DEFINITIONS(
  -DUSE_NON_CONST
  -DUSE_COMPOSITELESS_PHOTO_PUT_BLOCK
  )
```

When I try to run my program with Java-wrapped VTK, why do I get "java.lang.NoClassDefFoundError: vtk/vtkSomeClassName"?

The file **vtk.jar** is not in your CLASSPATH in your execution environment.

When I try to run my program with Java-wrapped VTK, why do I get "java.lang.UnsatisfiedLinkError: no vtkSomeLibraryName"?

Some or all of the library (e.g., dll) files cannot be found. Make sure the files exist and that the PATH environment variable of your execution environment points to them.

Using VTK

The C++ compiler cannot convert some pointer type to another pointer type in my little program

For instance, the C++ compiler cannot convert a **`vtkDataSet *`** type to a **`vtkImageData *`** type.

It means the compiler does not know the relationship between a **`vtkDataSet`** and a **`vtkImageData`**. This relationship is actually inheritance: **`vtkImageData`** is a subclass of **`vtkDataSet`**. The only way for the compiler to know this relationship is to include the header file of the subclass, that is:

```
#include "vtkImageData.h"
```

If you wonder why the compiler did not complain about an unknown type, it is because somewhere (probably in a filter header file) there is a forward class declaration, like:

```
class vtkImageData;
```

Accessing a pointer in Python

If you use VTK code with Python and need to pass some VTK data onto them, there are 2 approaches to wrap your code:

1. first, you can use the VTK wrapper (already used for the wrapping of VTK code)
2. you can use SWIG, which results in a light-weight module.

In the second case, you will need to convert some VTK data, say a vtkPolyData, to a void pointer (no, it is not sufficient to just pass the object). For that, you can use the __this__ member variable in Python for the VTK data - see mailing archives:

- vtk, Python and SWIG - 'state of the union' [9]

What object/filter should I use to do ???

Frequently when starting out with a large visualization system people are not sure what object to use to achieve a desired effect.

The most up-to-date information can be found in the VTK User's Guide (http://www.kitware.com/products/vtkguide.html).

Alternative sources for information are the appendix of the book which has nice one line descriptions of what the different objects do and the VTK man pages (http://www.vtk.org/doc/nightly/html/classes.html).

Additionally, the VTK man pages feature a "Related" section that provide links from each class to all the examples or tests using that class (http://www.vtk.org/doc/nightly/html/pages.html).This information is also provided in each class man page under the "Tests" or "Examples" sub-section.

Some useful books are listed at http://www.vtk.org/buy-books.php

What 3D file formats can VTK import and export?

The following table identifies the file formats that VTK can read and write. Importer and Exporter classes move full scene information into or out of VTK. Reader and Writer classes move just geometry.

File Format	Read	Write
3D Studio	vtk3DSImporter	
AVS "UCD" format	vtkAVSucdReader	
Movie BYU	vtkBYUReader	vtkBYUWriter
Renderman		vtkRIBExporter
Open Inventor 2.0		vtkIVExporter/vtkIVWriter
CAD STL	vtkSTLReader	vtkSTLWriter
Fluent GAMBIT ASCII	vtkGAMBITReader	
Unigraphics Facet Files	vtkUGFacetReader	
Marching Cubes	vtkMCubesReader	vtkMCubesWriter
Wavefront OBJ		vtkOBJExporter
VRML 2.0		vtkVRMLExporter
VTK Structured Grid †	vtkStructuredGridReader	vtkStructuredWriter
VTK Poly Data †	vtkPolyDataReader	vtkPolyDataWriter
PLOT3D	vtkPLOT3DReader	
CGM		vtkCGMWriter
OBJ	vtkOBJReader	
Particle	vtkParticleReader	
PDB	vtkPDBReader	
PLY	vtkPLYReader	vtkPLYWriter
Gaussian	vtkGaussianCubeReader	
Facet	vtkFacetReader	vtkFacetWriter
XYZ	vtkXYZMolReader	
Ensight ‡	vtkGenericEnSightReader	

† See the books The Visualization Toolkit, An Object-Oriented Approach to 3D Graphics [1] or the User's Guide [3] for details about structured grid and poly data file formats.

‡ The class vtkGenericEnSightReader allows the user to read an EnSight data set without a priori knowledge of what type of EnSight data set it is (among vtkEnSight6BinaryReader, vtkEnSight6Reader, vtkEnSightGoldBinaryReader, vtkEnSightGoldReader, vtkEnSightMasterServerReader, vtkEnSightReader).

For any other file format you may want to search for a converter to a known VTK file format, more info on: http://www.tech-edv.co.at/lunix/UTILlinks.html

Why can't I find vtktcl (vtktcl.c)?

In versions of VTK prior to 4.0 VTK Tcl scripts would require a:

```
catch {load vtktcl}
```

so that they could be executed directly from wish. In VTK 4.0 the correct mechanism is to use:

```
package require vtk
```

For people using versions earlier than 4.0, vtktcl is a shared library that is built only on the PC. Most examples used the "catch" notation so that they will work on UNIX and on the PC. On UNIX you must use the vtk executable/shell which should be in vtk/tcl/vtk.

Why does this filter not produce any output? eg. GetPoints()==0

This is a very common question for VTK users. VTK uses a pipeline mechanism for rendering, which has multiple benefits, including the fact that filters that aren't used don't get called. This means that when you call a function such as x->GetOutput()->GetPoints() this will return 0 if the filter has not yet been executed. Just call x->Update() beforehand to make the pipeline update everything up to that point and it should work. -timh

Problems with vtkDecimate and vtkDecimatePro

vtkDecimate and *vtkDecimatePro* have been tested fairly heavily so all known bugs have been removed. However, there are three situations where you can encounter weird behavior:

1. The mesh is not all triangles. Solution: use *vtkTriangleFilter* to triangulate polygons.
2. The mesh consists of independent triangles (i.e., not joined at vertices - no decimation occurs). Solution: use *vtkCleanPolyData* to link triangles.
3. Bad triangles are present: e.g., triangles with duplicate vertices such as (1,2,1) or (100,100,112), or (57,57,57), and so on. Solution: use *vtkCleanPolyData*.

How can I read DICOM files ?

Starting with VTK 4.4, you can use the vtkDICOMImageReader class [10] to read DICOM files. Note however that DICOM is a huge protocol, and vtkDICOMImageReader is not able to read every DICOM file out there. If it does not meet your needs, we suggest you look for an existing converter before coding your own. Some of them are listed in the The Medical Image Format FAQ (Part 8) [11].

GDCM

For a more elaborate DICOM library that supports more image format, you might try GDCM [12]. Specifically: vtkGDCMImageReader [13] & vtkGDCMImageWriter [14]

Grassroots DiCoM is a C++ library for DICOM medical files. It is automatically wrapped to python/C#/Java (using swig). It supports RAW,JPEG (lossy/lossless),J2K,JPEG-LS,RLE and deflated. It also comes with DICOM Part 3,6

& 7 of the standard as XML files.

If GDCM is too complex to integrate in your environment you can also consider simply using the command line converter: gdcmconv [15] to convert an unsupported DICOM file into something that vtkDICOMImageReader, can support. Typically you would want:

```
gdcmconv --raw compressed_input.dcom uncompressed_output.dcom
```

dicom2

Sebastien BARRE wrote a free DICOM converter, named dicom2, that can be used to convert medical images to raw format. This tool is a command line program and does not provide any GUI at the moment. http://dicom2.barre.nom.fr/

There is a special section dedicated to the VTK: http://dicom2.barre.nom.fr/how-to.html, then "Convert to raw (vtk)"

The following page also provide links to several other DICOM converters: http://www.barre.nom.fr/medical/samples/index.html#links

vtkVolume16Reader

When searching the vtkusers mailing list a lot of posts are still using vtkVolume16Reader to read in DICOM file. It will works in the following case:

- You know the dimension (cols & rows) of your image
- You know the spacing of your image
- You know the pixel type (pixel type & #components) of your image
- You know Pixel Data (7fe0,0010) is the last element in the image
- You know Pixel Data (7fe0,0010) was sent in uncompressed format (not encapsulated)

All those requirements are a stronger set of requirements than vtkDICOMImageReader, therefore it is encourage to use vtkDICOMImageReader instead.

The spacing in my DICOM files are wrong

Image Position (Patient) (0020,0032) is the only attribute that can be relied on to determine the "reconstruction interval" or "space between the center of slices".

If the distance between Image Position (Patient) (0020,0032) of two parallel slices along the normal to Image Orientation (Patient) (0020,0037) is not the same as whatever happens to be in the DICOM Spacing Between Slices (0018,0088) attribute, then (0018,0088) is incorrect, without question

This is a known bug in some scanners.

When Slice Thickness (0018,0050) + Spacing Between Slices (0018,0088) equals the computed reconstruction interval, then chances are the modality implementor has made the obvious mistake of misinterpreting the definition of (0018,0088) to mean the distance between edges (gap) rather than the distance between centers.

Further, one should never use Slice Location (0020,1041) either, an optional and purely annotative attribute, though chances are that the distance between the Slice Location (0020,1041) values of two slices will match the distance along the normal to the orientation derived from the position.

The GDCM library simply discard any information present in the (0018,0088) tag and instead recompute the spacing by computing the distance in between two consecutive slices (along the normal).

GDCM 1.x:

```
  typedef std::vector<gdcm::File *> FileList;
  FileList l;
```

```
gdcm::SerieHelper sh;
sh.OrderFileList(l); // calls ImagePositionPatientOrdering()
zspacing = sh.GetZSpacing();
```

GDCM 2.x:

```
IPPSorter ipp;
ipp.Sort( filenames );
zspacing = ipp.GetZSpacing();
```

How to handle large data sets in VTK

One of the challenges in VTK is to efficiently handle large datasets. By default VTK is tuned towards smaller datasets. For large datasets there are a couple of changes you can make that should yield a much smaller memory footprint (less swapping) and also improve rendering performance. The solution is to:

1. Use ReleaseDataFlag,
2. Turn on ImmediateModeRendering
3. Use triangle strips via vtkStripper
4. Use a different filter or mapper

Each of these will be discussed below.

Using ReleaseDataFlag

By default VTK keeps a copy of all intermediate results between filters in a pipeline. For a pipeline with five filters this can result in having six copies of the data in memory at once. This can be controlled using ReleaseDataFlag and GlobalReleaseDataFlag. If ReleaseDataFlag is set to one on a data object, then once a filter has finished using that data object, it will release its memory. Likewise, if GlobalReleaseDataFlag is set on ANY data object, all data objects will release their memory once their dependent filter has finished executing. For example in Tcl and C++

```
# Tcl
vtkPolyDataReader reader
[reader GetOutput] ReleaseDataFlagOn
```

```
// C++
vtkPolyDataReader *reader = vtkPolyDataReader::New();
reader->GetOutput()->ReleaseDataFlagOn();
```

or

```
// C++
vtkPolyDataReader *reader = vtkPolyDataReader::New();
reader->GetOutput()->GlobalReleaseDataFlagOn();
```

While turning on the ReleaseDataFlag will reduce your memory footprint, the disadvantage is that none of the intermediate results are kept in memory. So if you interactively change a parameter of a filter (such as the isosurface value), all the filters will have to re-execute to produce the new result. When the intermediate results are stored in memory, only the downstream filters would have to re-execute.

One hint for good interactive performance. If only one stage of the pipeline can have its parameters changed interactively (such as the target reduction in a decimation filter), only retain the data just prior to that step (which is the default) and turn ReleaseDataFlag on for all other steps.

Use ImmediateModeRendering

By default, VTK uses OpenGL display lists which results in another copy of the data being stored in memory. For most large datasets you will be better off saving memory by not using display lists. You can turn off display lists by turning on ImmediateModeRendering. This can be controlled on a mapper by mapper basis using ImmediateModeRendering, or globally for all mappers in a process by using GlobalImmediateModeRendering. For example:

```
# Tcl
vtkPolyDataMapper mapper
mapper ImmediateModeRenderingOn
```

```
// C++
vtkPolyDataMapper *mapper = vtkPolyDataMapper::New();
mapper->ImmediateModeRenderingOn();
```

or

```
// C++
vtkPolyDataMapper *mapper = vtkPolyDataMapper::New();
mapper->GlobalImmediateModeRenderingOn();
```

The disadvantage to using ImmediateModeRendering is that if memory is not a problem, your rendering rates will typically be slower with ImmediateModeRendering turned on.

Use triangle strips via vtkStripper.

Most filters in VTK produce independent triangles or polygons which are not the most compact or efficient to render. To create triangle strips from polydata you can first use vtkTriangleFilter to convert any polygons to triangles (not required if you only have triangles to start with) then run it through a vtkStipper to convert the triangles into triangle strips. For example in C++

```
vtkPolyDataReader *reader = vtkPolyDataReader::New();
reader->SetFileName("yourdatafile.vtk");
reader->GetOutput()->ReleaseDataFlagOn();

vtkTriangleFilter *tris = vtkTriangleFilter::New();
tris->SetInput(reader->GetOutput());
tris->GetOutput()->ReleaseDataFlagOn();
```

```
vtkStripper *strip = vtkStripper::New();
strip->SetInput(tris->GetOutput());
strip->GetOutput()->ReleaseDataFlagOn();
```

```
vtkPolyDataMapper *mapper = vtkPolyDataMapper::New();
mapper->ImmediateModeRenderingOn();
mapper->SetInput(tris->GetOutput());
```

The only disadvantage to using triangle strips is that they require time to compute, so if your data is changing every time you render, it could actually be slower.

Use a different filter or mapper

This is a tough issue. In VTK there are typically a couple of ways to solve any problem. For example an image can be rendered as a polygon for each pixel, or it can be rendered as a single polygon with a texture map on it. For almost all cases the second approach will be much faster than the first event though VTK supports both. There isn't a single good answer for how to find the best approach. If you suspect that it is running more slowly than it should, try posting to the mailing list or looking for other ways to achieve the same result.

VTK is slow, what is wrong?

We have heard people say that VTK is really slow. In many of these cases, changing a few parameters can make a huge difference in performance.

If you find that VTK is slower than other visualization systems running the same problem first take a look at the FAQ section dealing with large data: How to handle large data sets in VTK. Many of its suggestions will improve VTK's performance significantly for many datasets.

If you still find VTK slow, please let us know and send us an example (to mailto:kitware@kitware.com). In the past there have been some filters that simply were not written to be fast. When we come across one of these we frequently can make minor changes to the filter that will make it run much more quickly. In fact many changes in the past couple years have been this type of performance improvement.

Is VTK thread-safe ?

The short answer is no.

Many VTK sources and filters cache information and will not perform as expected when used in multiple threads. When writing a multithreaded filter, the developer has to be very careful about how she accesses data.

For example, GetXXX() methods which return a pointer should only be used to read. If the pointer returned by these methods are used to change data in multiple threads (without mutex locks), the result will most probably be wrong and unpredictable. In many cases, there are alternative methods which copy the data referred by the pointer. For example:

```
float* vtkDataArray::GetTuple(const vtkIdType i);
```

is thread-safe only for reading whereas:

```
void vtkDataArray::GetTuple (const vtkIdType i, float * tuple);
```

copies the requested tuple and is thread safe even if tuple is modified afterwards (as long as the same pointer is not passed as the argument tuple simultaneously by different threads).

Unfortunately, only very few methods are clearly marked as thread-(un)safe and, in many situations, the developer has to dig into the source code to figure out whether an accessor is thread safe or not.

vtkDataSet and most of it's sub-classes are well documented and almost all methods are marked thread-safe or not thread-safe. This might be a good place to start. Most of the filters in imaging and some filters in graphics (like *vtkStreamer*) are good examples of how a multi-threaded filter can be written in VTK.

However, if you are not interested in developing multithreaded filters but want to process some data in parallel using the same (or similar) pipeline, your job is much easier. To do this, create a different copy of the pipeline on each thread and execute them in parallel on a different piece of the data. This is best accomplished by using *vtkThreadedController* (instead of *vtkMultiThreader*). See the documentation of *vtkMultiProcessController* and *vtkThreadedController* and the examples in the parallel directory for details on how this can be done.

Also, note that most of the OpenGL libraries are not thread-safe. Therefore, if you are rendering to multiple render windows from different threads, you are likely to get in trouble, even if you have mutex locks around the render

calls.

Can I use STL with VTK?

As of VTK version 4.2, you can use the STL. However, see the VTK Coding Standards for limitations. Here's an example (from vtkInterpolateVelocityField):

In the .h file (the PIMPL) forward declare

```
class vtkInterpolatedVelocityFieldDataSetsType;
//
class VTK_COMMON_EXPORT vtkInterpolatedVelocityField : public
vtkFunctionSet
{
private:
  vtkInterpolatedVelocityFieldDataSetsType* DataSets;
};
```

In the .cxx file define the class (here deriving from the STL vector container)

```
# include <vtkstd/vector>
typedef vtkstd::vector< vtkSmartPointer<vtkDataSet> >
DataSetsTypeBase;
class vtkInterpolatedVelocityFieldDataSetsType: public DataSetsTypeBase
{};
```

In the .cxx file construct and destruct the class:

```
vtkInterpolatedVelocityField::vtkInterpolatedVelocityField()
{
  this->DataSets = new vtkInterpolatedVelocityFieldDataSetsType;
}
vtkInterpolatedVelocityField::~vtkInterpolatedVelocityField()
{
  delete this->DataSets;
}
```

And in the .cxx file use the container as you would any STL container:

```
for ( DataSetsTypeBase::iterator i = this->DataSets->begin();
      i != this->DataSets->end(); ++i)
  {
  ds = i->GetPointer();
  ....
  }
```

What image file formats can VTK read and write?

The following table identifies the image file formats that VTK can read and write.

Image File	Read	Write
AVI		vtkAVIWriter
Bitmap	vtkBMPReader	vtkBMPWriter
Digital Elevation Model (DEM)	vtkDEMReader	
DICOM	vtkDICOMImageReader	
GE Signal	vtkGESignaReader	
JPEG	vtkJPEGReader	vtkJPEGWriter
FFMPEG		vtkFFMPEGWriter
MINC (1.1)	vtkMINCImageReader	vtkMINCImageWriter
MPEG2		vtkMPEG2Writer
Binary UNC meta image data	vtkMetaImageReader	vtkMetaImageWriter
PNG	vtkPNGReader	vtkPNGWriter
PNM	vtkPNMReader	vtkPNMWriter
PostScript		vtkPostScriptWriter
SLC	vtkSLCReader	
TIFF	vtkTIFFReader	vtkTIFFWriter
RAW files †	vtkImageReader, vtkVolumeReader	

† A typical example of use is:

```
# Image pipeline
reader = vtkImageReader()
reader.SetDataByteOrderToBigEndian()
reader.SetDataExtent(0,511,0,511,0,511)
reader.SetFilePrefix("Ser397")
reader.SetFilePattern("%s/I.%03d")
reader.SetDataScalarTypeToUnsignedShort()
reader.SetHeaderSize(5432)
```

Printing an object.

Sometimes when debugging you need to print an object to a string, either for logging purposes, or in the case of windows applications, to a window.

Here is a way to do this:

```
std::ostringstream os;
//
// "SomeVTKObject" could be, for example,
// declared somewhere as: vtkCamera *SomeVTKObject;
//
SomeVTKObject->Print(os);
vtkstd::string str = os.str();
```

```
//
// Process the string as you want
```

Writing a simple CMakeLists.txt.

If you get something that looks like:

undefined reference to `__imp___ZN13vtkTIFFReader3NewEv' collect2: ld returned 1 exit status

You certainly forgot to pass in a library to your executable. The easisest way is to use CMakeLists.txt file.

For example the minimal project is:

```
FIND_PACKAGE(VTK)
IF (VTK_FOUND)
  INCLUDE (${VTK_USE_FILE})
ENDIF (VTK_FOUND)
ADD_EXECUTABLE(tiff tiff.cxx )
TARGET_LINK_LIBRARIES (tiff
    vtkRendering
)
```

Since vtkRendering is link against all other vtk lib. Except if you are building VTK with Hybrid or Parallel in that case you need to explicitely specify which library you want to link against.

Testing for VTK within a configure script

VTK uses CMake as build tool but if you VTK-based application wants to use autoconf and/or automake, then you will find very useful an M4 macro file which detects from your configure script the presence/absence of VTK on the user system. VTK won't add such file into the official distribution but you can always write your own, as I did. Look in VTK_Autoconf page for more info.

How do I get my C++ code editor to do VTK-style indentation?

If you are writing code with VTK, you may want to follow the VTK Coding Standards. This is particularly important if you plan to contribute back to VTK. Most C++ code editors will help you with indenting, but the indenting may differ significantly from that prescribed by the VTK Coding Standards. Fortunately, most editors have enough options to allow you to change the indention enough to get at least close to the VTK-style indentation.

Below is a list of C++ editors and some suggestions on getting the indentation VTK compliant. If you use a popular editor that is not listed here, please feel free to contribute.

Microsoft Visual C++ .NET indentation

Under the "Tools" menu, select "Options". Go to the options under "Text Editor" and then "C/C++". Click the "Tabs" options. Set "Indenting" to "Smart", "Indent Size" to 2, and select "Insert spaces". Click the "Formatting" options enable "Indent braces".

This will make most of the indentation correct. However, it will indent all of the braces. In VTK classes, most of the braces are indented, but those starting a class, method, or function are typically flush left. You will have to correct this on your own.

Emacs indentation

Place the Elisp Code for VTK-Style C Indentation in your .emacs file.

Vim indentation

Andy Cedilnik has some information on following the VTK coding guidelines using vim. You may place the following in your ~/.vimrc file

```
set tabstop=2       " Tabs are two characters
set shiftwidth=2    " Indents are two characes too
set expandtab       " Do not use tabs
set cinoptions={1s,:0,l1,g0,c0,(0,(s,m1
"Keep tabs in makefiles as they are significant:
:autocmd BufRead,BufNewFile [Mm]akefile :set noexpandtab
```

How to display transparent objects?

(keywords: alpha, correct, depth, geometry, object, opacity, opaque, order, ordering, peel, peeling, sorting, translucent, transparent.)

When opaque geometry is rendered, there is no need to sort it because the depth buffer (or z-buffer) is used and the sorting is done automatically by keeping the geometry closest to the viewpoint at a given pixel. (It is easy because it is a MAX/MIN calculation, not a real sorting).

With translucent geometry the final color of a pixel is the contribution of all the geometry primitives visible through the pixel. The color of the pixel is the result of **a** blending operation between the colors of all visible primitives. Blending operations themselves are usually order-dependent (ie not commutative). That's why depth sorting is required. There are two ways to fix the ordering in VTK:

- 1. Append all your polygonal geometry with vtkAppendPolyData [16] and pass it to vtkDepthSortPolyData [17]. See this tcl example [18]. Depth sorting is done per centroid of geometry primitives, not per pixel. For this reason it is not exact but it solves **most** of the ordering and gives result usually good enough.
- 2. If the graphics card supports it, use "depth peeling". It performs per pixel sorting (better result) but it is really slow.

What's the deal with SetInput() vs. SetInputConnection()?

(keywords: SetInput, SetInputConnection, vtkAlgorithm, algorithm, pipeline, source)

In the transition from VTK 4 to VTK 5, the VTK pipeline executive was completely cleaned up and redesigned. The fundamental idea behind the new pipeline is that the "pipeline" should consist of a chain of "algorithm" objects. The algorithms are connected together with the familiar b->SetInputConnection(a->GetOutputPort()) methods.

So how is this different from SetInput()/GetOutput()? The difference between an "OutputPort" and an "Output" is as follows:

- OutputPort (*vtkAlgorithmOutput*): A trivial object that says "I am output port N of algorithm X" (usually N = 0).
- Output (*subclass of vtkDataObject*): A container for data produced by any VTK code.

The "OutputPort" method does not presuppose anything about what data (if any) will pass along the pipeline. It could simply signify that algorithm "a" must execute before algorithm "b". This provides enormous flexibility. Trust the VTK designers here, it's better to do things this way than to have the user grab the actual data object from the output of one algorithm and shove it into the input of another.

However, any newcomer to VTK will quickly notice that the use of SetInputConnection() is not universal. The reason is this: only vtkAlgorithm objects can have a SetInputConnection() or GetOutputPort() method. Some objects that can take inputs are not derived from vtkAlgorithm. For example vtkImageActor is a vtkProp3D and therefore it

cannot be a vtkAlgorithm (VTK never uses multiple inheritance). This is a case of an old VTK object that doesn't "fit" the new VTK 5 pipeline. However, the VTK developers did not want to throw away such useful classes when the pipeline was redesigned. Instead, such classes are still served by the backwards-compatible SetInput() method.

So, use SetInputConnection() whenever you can, but if there is no SetInputConnection(), then go ahead and use SetInput(). There is nothing wrong with doing so. The new pipeline is backwards compatible with the old pipeline methods.

Platform-specific questions

What platforms does vtk run on?

VTK should compile and run on most versions of Unix, Linux, Windows, and Mac OS X. It has been tested on Suns, SGIs, HPs, Alphas, RS6000s and many Windows and Mac workstations.

What Graphics Cards work with VTK

VTK uses OpenGL to perform almost all of its rendering and some graphics cards/drivers have better support for OpenGL than others. This is not a listing of what cards perform well. It is a listing of what cards actually produce correct results. Here is a list of cards and their status roughly in best to worst order.

- Any Nvidia desktop card on Windows -- 100% compatible
- Any ATI desktop cards on Windows -- 100% compatible
- Mesa -- most releases pass all VTK tests
- Microsoft Software OpenGL -- passes all VTK tests but does have a couple bugs
- Mac graphics cards -- these usually pass all VTK tests. Older cards may have some issues, for example, the ATI Rage 128 Pro does not support textures larger than 1024x1024.
- Non-linux UNIX cards (Sun HP SGI) -- These generally work
- Any Nvidia card under linux -- these usually pass all VTK tests but have some issues
- Any ATI card under linux -- these usually pass all VTK tests but have some issues
- Nvidia laptop graphics cards under Windows -- known to have some issues, newer cards pass all tests
- ATI laptop graphics cards under Windows -- known to have some issues, newer cards pass all tests (e.g. ATI Mobility Radeon 9600 [19])
- Intel Extreme Graphics -- fails some VTK tests

How do I build the examples on the PC running Windows?

Since building the C++ examples on the PC isn't all that easy, here are some instructions from Jack McInerney.

Steps for creating a VTK C++ project 8/14/96

This is based on what I learned creating a project to run the Mace example. These steps allowed me to successfully built and run this example.

1. Create a console project (File, New, then select Console application).
2. Add the files of interest to the project. (e.g., Mace.cxx)
3. Under Build, select Update all Dependencies. A long list of .hh files will show up under dependencies
   For this to work, Visual C++ needs to know where to look to find the include files. In my case they are at C:\VTK\VTK12SRC\INCLUDE. To tell Visual C++ to look there, go to Tools, Options. Select the tab Directories. Under the list for Include files add: C:\VTK\VTK12SRC\INCLUDE
4. Compile the file Mace.cxx. This will lead to many warnings about data possibly lost as double variables are converted to float variables. These can be gotten rid of by going to Build, Settings, and select the C++ tab. Under the General catagory, set Warning Level to 1* (instead of 3).

5. Before linking, some additional settings must be modified. Go to Build, Settings, and select the Link tab. In the General catagory, add the libraries opengl32.lib and glaux.lib to the Object/Library Modules. Put a space between each file name. Then select the C++ tab and the Category: Code Generation. Under Use Run-Time Library, select Debug Multithreaded DLL. Select OK to exit the dialog box. The above libraries are available from Microsoft's Web site at: http://www.microsoft.com/softlib/mslfiles/Opengl95.exe or ftp://ftp.microsoft.com/softlib/mslfiles/Opengl95.exe
   This is a self extracting archive which contains these files. Simply place them in your windows system directory.
6. Link the code by selecting Build, Build MaceProject.exe. I still get one warning when I do this, but it appears to be harmless

When you go to run the program, it will bomb out unless it can find 2 DLLs: Opengl32.dll and Glu32.dll. These need to be located either in the project directory or the C:\WINDOWS directory. These files are supplied on the vtk CD-ROM (in the vtk\bin directory).

How do I build the Java examples on the PC running Windows?

One common issue building the examples is missing one or all of vtkPanel, vtkCanvas and AxesActor classes. For whatever reason these are not in the vtk.jar (at least for 4.2.2). But you can get them from the source distribution (just unzip the source and extract these needed .java files, and point your Java-compiler to them).

Another common issue appears to be class loading dependency errors. Make sure the directory with the .dll files is in your classpath when you run (default location is C:\Program Files\vtk42\bin\). Yet this still seems insufficient for some of the libraries. One possible solution is to copy the Java awt.dll to this directory as well.

64-bit System Issues

vtk builds on 64 bit systems, that is, systems where sizeof(void*) is 64 bits. However, parts of the vtk codebase are not 64 bit clean and so runtime problems are likely if that code is used.

General

VTK binary files are not compatible between 32-bit and 64-bit systems. For portability, use the default file type, ASCII, for vtkPolyDataWriter, etc. You may be able to write a binary file on a 64-bit system and read it back in.

Mac OS X Specific

Mac OS X 10.3 and earlier have no support for 64 bit. On Mac OS X 10.4, VTK cannot be built as 64 bit because it requires Carbon, Cocoa, or X11, none of which are available to 64 bit processes. On Mac OS X 10.5, Cocoa is available to 64 bit processes, but Carbon is not. VTK is known to work reasonably with 64 bit Cocoa.

Windows Specific

todo

What size swap space should I use on a PC?

Building vtk on the PC requires a significant amount of memory (at least when using Visual C++)... but the final product is nice and compact. To build vtk on the PC, we recommend setting the min/max swap space to at least 400MB/500MB (depending on how much RAM you have... the sum of RAM and swap space should be roughly 500+ MB).

Are there any benchmarks of VTK and/or the hardware it runs on?

Take a look at the "Simple Sphere Benchmark":

http://www.barre.nom.fr/vtk/bench.html

It is not a "real world" benchmark, but provide synthetic results comparing different hardware running VTK:

http://purl.oclc.org/NET/rriv/vtk/sphere-bench

Why is XtString undefined when using VTK+Python on Unix?

This is a side effect of dynamic linking on (some?) Unix systems. It appears often on Linux with the Mesa libraries at least. The solution is to make sure your Mesa libraries are linked with the Xt library. One way to do this is to add "-lXt" to MESA_LIB in your user.make file.

How do I get the Python bindings to work when building VTK with Borland C++?

If you've built VTK with the freely downloadable Borland C++ 5.5 (or its commercial counterpart) and you're using the Python binaries from http://www.python.org/, you'll note that when you try to run a VTK Python example you get something similar to the following error message:

```
from vtkCommonPython import *
ImportError: dynamic module does not define init function
(initvtkCommonPython)
```

This is because BCC32 prepends an underscore ("_") to all exported functions, so (in this case) the vtkCommonPython.dll contains a symbol _initvtkCommonPython which Python does not find. All kits (e.g. Rendering, Filtering, Patented) will suffer from this problem.

The solution is to create Borland module definition in the VTK binary (output) directory, in my case VTK/bin. You have to do this for all kits that you are planning to use in Python. Each .def file must have the same basename as the DLL, e.g. "vtkCommonPython.def" for vtkCommonPython.dll and it must be present at VTK link time. The def file contains an export alias, e.g.:

```
EXPORTS
initvtkCommonPython=_initvtkCommonPython
```

The Borland compiler will create an underscore-less alias in the DLL file and Python will be able to load it as a module.

How do I build Python bindings on AIX?

There is a problem with dynamic loading on AIX. Old AIX did not have dlopen/dlsym, but they used load mechanism. Python still reflects this. VTK is however not compatible with the old load mechanism.

The following patch to Python 2.2.2 makes python use dlopen/dlsym on AIX 5 or greater.

http://www.vtk.org/files/misc/python_aix.diff

How to build VTK for offscreen rendering?

[this section is obsolete. Mangle Mesa is not supported anymore in VTK>=5.2] (not sure about 5.0)

Struggled a few hours to get VTK to do offscreen rendering. I use it to batch process medical images. Without actually producing output on the screen, I still print resulting images in a report to easily review the results of an experiment.

Here is how I solved this problem for VTK version 4.2.2.

1. Download Mesa-4.0.4 source

Modify Mesa-4.0.4/Make-config in the 'linux:' target the following vars:

```
GL_LIB = libVTKMesaGL.so
GLU_LIB = libVTKMesaGLU.so
GLUT_LIB = libVTKMesaglut.so
GLW_LIB = libVTKMesaGLw.so
OSMESA_LIB = libOSVTKMesa.so
```

In Mesa 6.2.1 you need to edit Mesa/configs/default instead:

```
 # Library names (base name)
 GL_LIB = VTKMesaGL
 GLU_LIB = VTKMesaGLU
 GLUT_LIB = VTKMesaglut
 GLW_LIB = VTKMesaGLw
 OSMESA_LIB = VTKMesaOSMesa
```

And then export this env var:

```
export CFLAGS="-O -g -ansi -pedantic -fPIC
-ffast-math-DUSE_MGL_NAMESPACE -D_POSIX_SOURCE
-D_POSIX_C_SOURCE=199309L-D_SVID_SOURCE -D_BSD_SOURCE -DUSE_XSHM
-DPTHREADS -I/usr/X11R6/include"
```

then

For Mesa 4.0.4

```
  make -f Makefile.X11 linux
  cp Mesa-4.0.4/lib/* /data/usr/mesa404/lib/
```

in Mesa 6.2.1:

```
  make linux-x86
  make install
  (I generally use /opt/VTKMesa/*)
```

I use 'VTKMesa' name extension to avoid conflicts with my RH9.0 libs (especially OSMesa lib in XFree!). I'm using shared libraries, because that allows me to use dynamic libs from VTK and not the vtk program itself without explicitly having to load VTKMesaGL with my app. I copied the 'VTKMesa' libs in /data/usr/mesa404/lib/, but any odd place probably will work. Avoid /usr/lib /usr/local/lib for now.

2. Follow normal instructions to get a proper working vtk, then

```
 ccmake
```

with the following options:

VTK_USE_MANGLED_MESA	ON
MANGLED_MESA_INCLUDE_DIR	/data/usr/mesa404/include
MANGLED_MESA_LIBRARY	/data/usr/mesa404/lib/libVTKMesaGL.so
MANGLED_OSMESA_INCLUDE_DIR	/data/usr/mesa404/include
MANGLED_OSMESA_LIBRARY	/data/usr/mesa404/lib/libVTKMesaOSMesa.so
OPENGL_xmesa_INCLUDE_DIR	/data/usr/mesa404/include

test using /data/prog/VTK-4.2.2/Examples/MangledMesa/Tcl scripts

If you're doing things on UNIX, you should also look at VTK Classes. It has links to RenderWindow objects that are probably easier to use than rebuilding VTK with Mesa.

How to get keyboard events working on Mac OS X?

On Mac OS X, there are (at least) two kinds of executables:

- Application Bundles [20]
- plain UNIX executables

For a program to be able to display a graphical interface (that is, display windows that allow mouse and keyboard interaction) it really should be an Application Bundle. If a plain UNIX executable tries, there will be various bugs, such as keyboard and mouse events not working reliably.

Many, but not all, of the example VTK applications are built as plain UNIX executables, and thus have these problems. This is VTK bug 2025 [21].

When you build your own VTK application, it is best to make it in the form of an Application Bundle. With CMake 2.0.5 or later, simply add the following to your CMakeLists.txt file:

```
IF(APPLE)
  SET(EXECUTABLE_FLAG MACOSX_BUNDLE)
ENDIF(APPLE)
```

If for some reason you cannot build as an Application Bundle (perhaps because your app needs command line parameters) you might be able to avoid the above problems by adding an __info_plist section [22] to your Mach-O executable. If you succeed, please post to the VTK list.

Can VTK be built as a Universal Binary on Mac OS X?

For VTK 5.0.4 and older, the short answer is "no".

For VTK CVS the short answer is "mostly". You need to set CMAKE_OSX_ARCHITECTURES to the architectures you want and CMAKE_OSX_SYSROOT to a Mac OS X SDK that supports Universal builds. The usual settings are:

```
 CMAKE_OSX_ARCHITECTURES=ppc;i386

 CMAKE_OSX_SYSROOT=/Developer/SDKs/MacOSX10.4u.sdk
```

This will result in a Universal build. However, there may be runtime bugs due to VTK's use of TRY_RUN. Work is being done to improve this situation.

How can I stop Java Swing or AWT components from flashing or bouncing between values?

While not strictly a VTK problem, this comes up fairly often when using Java-wrapped VTK. Try the following two JRE arguments to stop the Swing/AWT components flashing:

```
-Dsun.java2d.ddoffscreen=false -Dsun.java2d.gdiblit=false
```

Note that these are classified as "unsupported properties," so may not work on all platform or installations (in particular, ddoffscreen refers to DirectDraw and, as such, is specific to Windows).

How can a user process access more than 2 GB of ram in 32-bit Windows?

By default on Windows, the most memory that a user process can access is 2 GB, no matter how much RAM you have installed in your system. With Windows XP Professional you can make it possible for a process to use up to 3 GB of memory by doing two things:

1) Modify the boot parameters in boot.ini (on my 32 bit WinXP Pro machine, it's in: "C:\boot.ini") to tell the operating system that you want user processes to have access to up to 3GB of RAM (This is a really important file, and if you don't know what you are doing, stop reading this and go back to work!). This is done by adding the /3GB flag to the line of the file that tells the boot loader where the operating system is. My boot.ini file looks like:

```
  [boot loader]
  timeout=30
  default=multi(0)disk(0)rdisk(0)partition(1)\WINDOWS
  [operating systems]
  multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Microsoft Windows XP
Professional" /3GB
```

This is a very bad file to make mistakes on, so don't - it may be very difficult to repair your computer to boot if you mess up this file. There is a nice description of this in the Microsoft article Memory Support and Windows Operating Systems [23].

2) The other thing that you need to do is make your executable LARGEADDRESSAWARE. Assuming that you have a Windows binary that you want to try this on, you can use the 'editbin' utility that comes with Visual Studio to change the setting of one bit (the IMAGE_FILE_LARGE_ADDRESS_AWARE bit) in the image header of the executable. For a program 'prog.exe' you can make the change by

```
  editbin /LARGEADDRESSAWARE prog.exe
```

Of course, depending on how your program handles memory you might find that it crashes when you try to use the extra memory, but that's a separate issue. If you are compiling your program with a version of Visual Studio you should be able to find the switch to make your program /LARGEADDRESSAWARE.

Shared builds of VTK and debugging QVTKWidget using Visual Studio

Assuming that you have built a shared build of VTK and you may or may not have a set it up such that there is a path to the release version of VTK in your PATH statement.

Then if you debug a project that is using QVTKWidget, you will come across a problem in that if you are debugging a debug version; the application depends upon the debug version of QVTK.dll which will depend upon QtGui4d.dll (among others) and load it. But, because the release version of QVTK.dll is in the path, QtGiu4.dll will also be loaded preventing the application from running. You will get a

```
"QWidget: Must construct a QApplication before a QPaintDevice"
```

message.

The solution to this problem is to set the path to the correct build of VTK on the "**Debugging**" properties of your project. Right click on your project, bring up the properties dialog, and select "**Debugging**" from the list on the left. There should be an "**Environment**" line. You can add variables here using key=value pairs. For example, add the following line:

```
PATH=<Path To VTK>\bin\$(OutDir);%PATH%
```

You can then add the same line to other configurations, such as the release one, by selecting them from the top left drop down box labelled **Configuration**.

$(OutDir) will be set by Visual Studio to either Debug or Release, depending upon what configuration you have selected. Make sure that ;%PATH% is appended so that Qt and other files can be appended to the PATH statement.

Changes to the VTK API

What is the policy on Changes to the API

Between patch releases maintain the API unless there is a really strong reason not to.

Between regular releases maintain backwards compatibility to the API with prior releases of VTK when doing so does not increase the complexity or readability of the current VTK or when the benefits of breaking the API are negligible.

Clearly these statements have a lot of wiggle room. For example in vtkLightKit BackLight and Headlight were used and released. Now BackLight and HeadLight might make more sense and probably will be easier for non-native English speakers, but is it worth breaking the API for it, probably not. Another factor is how long the API has been around and how widely used it is. These all indicate how painful it will be to change the API which is half of the cost/benefit decision.

Change to vtkIdList::IsId()

vtkIdList::IsId(int id) used to return a 0 or 1 to indicate whether the specified id is in the list. Now it returns -1 if the id is not in the list; or a non-negative number indicating the position in the list.

Changes vtkEdgeTable

vtkEdgeTable had two changes. The constructor now takes no arguments, and you use InitEdgeInsertion() to tell the class how many points are in the dataset. Also, IsEdge(p1,p2) now returns a -1 if the edge (defined by points p1,p2) is not defined. otherwise a non-negative integer value is returned.

These changes were made to support the association of attributes with edges.

Changes between VTK 4.2 and VTK 4.4 (and how to update)

We have removed the CVS date, revision, and the language from the copyright on all the files. This information wasn't being used much and it created extra work for developers. For example you edit vtkObject.h rebuild all of VTK, check in you change, then you must rebuild all of VTK again because commiting the header file causes it to be changed by CVS (because the revision number changed) This change will also make it easier to compare different branches of VTK since these revision number differences will no longer show up. The CVS revision number is still in the cxx file in the RevisionMacro. You don't need to make any changes to your code for this.

The DataArray classes now use a templated intermediate class to share their implementation. Again there is no need for you to make changes to your code.

Legacy code has been removed. Specifically none of the old style callbacks are supported and observers should be used instead. So where you used a filter->SetStartMethod(myFunc) you should do a filter->AddObserver(vtkCommand::StartEvent,myCommand) Usually this will require you to create a small class for the observer. vtkImageOpenClose3D.cxx has an example of using an observer and there are a few other examples in VTK. If you switch to using Observers your code should also work with versions of VTK from 3.2 or later since the Observers have been in VTK since VTK 3.2.

Many functions that previously took or returned float now take or return double. To change your code to work with VTK 4.4 or later you can just replace float with double for the appropriate calls and variables. If you want your code to work with both old and new versions of VTK you can use vtkFloatingPointType which is defined to be double in VTK 4.4 and later and it is float in vtk 4.2.5. In versions of VTK prior to 4.2.5 you can use something like:

```
#ifndef vtkFloatingPointType
#define vtkFloatingPointType vtkFloatingPointType
typedef float vtkFloatingPointType;
#endif
```

at the beginning of your code. That will set it to the correct value for all versions of VTK old and new.

Use of New() and Delete() now enforced (vs. new & delete)

Constructors and destructors in VTK are now protected. This means you can no longer use little "new" or "delete" to create VTK instances. You'll have to use the methods ::New() and ::Delete() (as has been standard practice for some time).

The reason for this is to enforce the use of New() and Delete(). Not using New() and Delete() can lead to bad mojo, mainly reference counting problems or not taking advantage of special procedures incorporated into the New() method (e.g., selecting the appropriate hardware interface during instance creation time).

If you've used New() and Delete() in your code, these changes will not affect you at all. If you're using little "new" or "delete", your code will no longer and compile and you'll have to switch to New() and Delete().

Changes between VTK 4.4 and VTK 4.6

Collection Changes

Collections have had some small changes (originally started by Chris Volpe) to better support reentrant iteration. Specifically all the collection have an InitTraversal(sit) and GetNextFoobar(sit) methods. (where Foobar is what the collection contains, for example GetNextActor(sit)) The argument to both of these methods is a vtkCollectionSimpleIterator. Most of the collection use in VTK has been modified to use these new methods. The advantage is that these new methods support having the same collection be iterated through in a reentrant safe manner. In the past this was not true and led to a number of problems. In the future for C++ class development please use this approach to iterating through a collection. These changes are fully backwards compatible and no old APIs were harmed in the making of these changes. So in summary, for the future, where you would have written:

```
for (actors->InitTraversal();
     (actor = actors->GetNextActor());)
```

you would now have:

```
 vtkCollectionSimpleIterator actorIt;
 for (actors->InitTraversal(actorIt);
      (actor = actors->GetNextActor(actorIt));)
```

Changes in VTK between 3.2 and 4.0

- Changes to vtkDataSetAttributes, vtkFieldData and vtkDataArray: All attributes (scalars, vectors...) are now stored in the field data as vtkDataArray's. vtkDataSetAttributes became a sub-class of vtkFieldData. For backwards compatibility, the interface which allows setting/getting the attributes the old way (by passing in a sub-class of vtkAttributeData such as vtkScalars) is still supported but it will be removed in the future. Therefore, the developers should use the new interface which requires passing in a vtkDataArray to set an attribute. vtkAttributeData and it's sub-classes (vtkScalars, vtkVectors...) will be deprectated in the near future; developers should use vtkDataArray and it's sub-classes instead. We are in the process of removing the use of these classes from vtk filters.
- Subclasses of vtkAttributeData (vtkScalars, vtkVectors, vtkNormals, vtkTCoords, vtkTensors) were removed. As of VTK 4.0, vtkDataArray and it's sub-classes should be used to represent attributes and fields. Detailed description of the changes and utilities for upgrading from 3.2 to 4.0 can be found in the package: http://www.vtk.org/files/misc/Upgrading.zip
- Added special methods to data arrays to replace methods like

```
tc SetTCoord i x y 0
```

or

```
vc SetVector i vx vy vz
```

in interpreted languages (Tcl, Python, Java). Use:

```
tc SetTuple2 i x y
```

or

```
vc SetTuple3 i vx vy vz
```

- Improved support for parallel visualization: vtkMultiProcessController and it's sub-classes have been re-structured and mostly re-written. The functionality of vtkMultiProcessController have been re-distributed between vtkMultiProcessController and vtkCommunicator. vtkCommunicator is responsible of sending/receiving messages whereas vtkMultiProcessController (and it's subclasses) is responsible of program flow/control (for example processing rmi's). New classes have been added to the Parallel directory. These include vtkCommunicator, vtkMPIGroup, vtkMPICommunicator, vtkSharedMemoryCommunicator, vtkMPIEventLog... There is now a tcl interpreter which supports parallel scripts. It is called pvtk and can be build on Windows and Unix. Examples for both Tcl and C++ can be found in the examples directories.
- vtkSocketCommunicator and vtkSocketController have been added. These support message passing via BSD sockets. Best used together with input-output ports.
- Since it was causing very long compile times (it essentially includes every vtk header file) and it was hard to maintain (you had to add a line whenever you added a class to VTK) vtk.h was removed. You will have to identify the header files needed by your application and include them one by one.
- vtkIterativeClosestPointTransform has been added. This class is an implementation of the ICP algorithm. It matches two surfaces using the iterative closest point (ICP) algorithm. The core of the algorithm is to match each vertex in one surface with the closest surface point on the other, then apply the transformation that modify one surface to best match the other (in a least square sense).
- The SetFileName, SaveImageAsPPM and related methods in vtkRenderWindow have been removed. vtkWindowToImageFilter combined with any of the image writers provides greater functionality.
- Support for reading and writing PGM and JPEG images has been included.

- Methods with parameters of the form "type param[n]" are wrapped. Previously, these methods were only wrapped if the array was declared 'const'. The python wrappers will allow values to be returned in the array.
- The directory structure was completely reorganized. There are now subdirectories for Common (core common classes) Filtering (superclasses for filtering operations) Imaging (filters and sources that produce images or structured points) Graphics (filters or sources that produce data types other than ImageData and StructuredPoints) IO (file IO classes that do not require Rendering support) Rendering (all actors mappers annotation and rendering classes) Hybrid (typically filters and sources that require support from Rendering or both Imaging and Graphics) Parallel (parallel visualization support classes) Patented (patented classes) Examples (documented examples) Wrapping (support for the language wrappers). In many directories you will see a Testing subdirectory. The Testing subdirectories contain tests used to validate VTKs operation. Some tests may be useful as examples but they are not well documented.
- The Build process for VTK now uses CMake (found at www.cmake.org) This replaces pcmaker on windows and configure on UNIX. This resolves some longstanding problems and limitation we were having with pcmaker and configure, and unifies the build process into one place.

Changes to VTK between 4.0 and 4.2

- Use of macros to support serialization, standardize the New method, and provide the Superclass typedef.
- Subclassing of VTK classes in the python wrappers (virtual method hooks are not provided).
- vtkImageWindow, vtkImager, vtkTkImageWindowWidget and their subclasses have been removed to reduce duplicated code and enable interation in ImageWindows. Now people should use vtkRenderer and vtkRenderWindow instead. vtkImageViewer still works as a turn key image viewing class although it now uses vtkRenderWindow and vtkRenderer internally instead of vtkImageWindow and vtkImager.
- New class: vtkBandedPolyDataContourFilter. Creates solid colored bands (like you find on maps) of scalar value.
- Event processing: Several new events to VTK were added (see vtkCommand.h). Also event processing can now be prioritized and aborted. This allows applications to manage who processes which events, and terminates the processing of a particular event if desired.
- 3D Widgets: A new class vtkInteractorObserver was added to observe events on vtkRenderWindowInteractor. Using the new event processing infrastructure, multiple 3D widgets (subclasses of vtkInteractorObserver) can be used simultaneously to process interactions. Several new 3D widgets have been added including:
  - vtkLineWidget
  - vtkPlaneWidget
  - vtkImagePlaneWidget
  - vtkBoxWidget
  - vtkSphereWidget
- Besides providing a representation, widgets also provide auxiliary functionality such as providing transforms, implicit functions, plane normals, sphere radius and center, etc.
- New class: vtkInstantiator provides a means by which one can create an instance of a VTK class using only the name of the class as a string.
- New class: vtkXMLParser provides a wrapper around the Expat XML parsing library. A new parser can be written by subclassing from vtkXMLParser and providing a few simple virtual method implementations.
- TIFF reader is now implemented using libtiff, which makes it capable of reading almost all available TIFF formats. The libtiff is also available internally as vtktiff.
- New method (all sub-classes of vtkObject): Added a virtual function called NewInstance to vtkTypeMacro. NewInstance creates and returns an object of the same type as the current one. It does not copy any properties.

The returned pointer is of the same type as the pointer the method was invoked with. This method should replace all the MakeObject methods scattered through VTK.

- vtkSetObject macro is depricated for use inside the VTK. It is still a valid construct in projects that use VTK. Instead use vtkCxxSetObjectMacro which does the same thing.
- vtkPLOT3DReader have been improved. It now supports:
  - multigrid (each block is one output)
  - ascii
  - fortran-style byte counts
  - little/big endian
  - i-blanking (partial)
- A new vtkTextProperty class has been created, and duplicated text API s have been obsoleted accordingly. Check the

Text properties in VTK 4.2 FAQ entry for a full description of the change.

How do I upgrade my existing C++ code from 3.2 to 4.x?

This is (a corrected version of) an email that was posted to vtkusers. Please feel free to correct or add anything.

I've just ported my medium-sized (40K lines) application from vtk3.2 to vtk4.x. I thought I would share my experiences with you, in case there were people out there contemplating it but a bit scared.

One source of information for upgrading code is:

http://www.vtk.org/files/misc/Upgrading.zip

I'm using VC++6 + MFC on Win2K and was unable/unwilling to run the script in the zip file.

So,

I switched all my include directories to the new VTK ones and recompiled. 337 errors, not unexpectedly. Most concerned vtkScalars and vtkTCoords which have both been removed. Where I was using single value scalars, like this:

```
vtkScalars *scalars = vtkScalars::New();
scalars->SetNumberOfScalars(N_POINTS);
...
polydata->GetPointData()->SetScalars(scalars);
...
scalars->SetScalar(i,2.3);
...
```

I replaced with:

```
vtkFloatArray *scalars = vtkFloatArray::New();
scalars->SetNumberOfComponents(1);
scalars->SetNumberOfTuples(N_POINTS);
...
polydata->GetPointData()->SetScalars(scalars);
...
scalars->SetTuple1(i,2.3);
...
```

OK so far, far fewer errors.

Where I had 2D texture coordinates:

```
vtkTCoords *tcoords = vtkTCoords::New();
tcoords->SetNumberOfTCoords(N);
...
float p[3];
p[0]=x; p[1]=y;
tcoords->SetTCoord(i,p);
...
```

I replaced with:

```
vtkFloatArray *tcoords = vtkFloatArray::New();
tcoords->SetNumberOfComponents(2);
tcoords->SetNumberOfTuples(N);
...
float p[2];
p[0]=x; p[1]=y;
tcoords->SetTuple(i,p);
....
```

All well and good, still fewer errors. Make sure you call SetNumberOfComponents *before* SetNumberOfTuples else you'll get problems (I did!).

Where I was creating 0-255 image data and had been using:

```
vtkScalars* scalars = vtkScalars::New();
scalars->SetDataTypeToUnsignedChar();
...
```

I replaced with:

```
vtkUnsignedCharArray *scalars = vtkUnsignedCharArray::New()
...
```

Going well!

When creating RGB images, I had been using:

```
vtkScalars *scalars = vtkScalars::New();
scalars->SetDataTypeToUnsignedChar();
scalars->SetNumberOfComponents(3);
scalars->SetNumberOfScalars(X*Y);
...
scalars->SetActiveComponent(0);
scalars->SetScalar(i,val1);
scalars->SetActiveComponent(1);
scalars->SetScalar(i,val2);
scalars->SetActiveComponent(2);
scalars->SetScalar(i,val3);
...
```

I replaced with:

```
vtkUnsignedCharArray *scalars = vtkUnsignedCharArray::New()
scalars->SetNumberOfComponents(3);
```

```
scalars->SetNumberOfTuples(X*Y);
...
scalars->SetComponent(i,0,val1);
scalars->SetComponent(i,1,val2);
scalars->SetComponent(i,2,val3);
...
```

My remaining errors concerned vtkWin32OffscreenRenderWindow that has been removed. Where I had been using:

```
vtkWin32OffscreenRenderWindow *offscreen =
vtkWin32OffscreenRenderWindow::New();
...
```

I replaced with:

```
vtkWin32OpenGLRenderWindow *offscreen =
vtkWin32OpenGLRenderWindow::New();
offscreen->SetOffScreenRendering(1);
...
```

All done. I'd had to throw in some #include "vtkFloatArray.h" and things like that of course. Zero compile errors.

Had to remember to link against the new vtk lib files, so I removed

```
vtkdll.lib
```

and added

```
vtkCommon.lib
vtkGraphics.lib
```

etc.

Zero link errors. My program is up and running again, no apparant problems. Plus now I can use all the new features of vtk4. (And I'm sure it's faster but maybe that's my imagination.)

All this took me about three hours.

Bye!

Tim.

What is the release schedule for VTK

VTK has a formal release every eight to sixteen months. VTK 4.0 was cut in December 2001 and released in March 2002. VTK 4.2 was releaseed in February 2003. VTK 4.4 (which was an interim release) was released at the end of 2003. VTK 5.0 was released in January 2006, 5.0.1 in July 2006, 5.0.2 in September 2006, 5.0.3 in March 2007, and 5.0.4 in January 2008.

Roadmap: What changes are being considered for VTK

This is a list of changes that are being considered for inclusion into VTK. Some of these changes will happen while other changes we would like to see happen but may not due to funding or time issues. For each change we try to list what the change is, when we hope to complete it, if it is actively being developed. Detailed discussion on changes is limited to the vtk-developers mailing list.

1. Modify existing image filters to use the new vtkImageIterator etc. Most simple filters have been modified to use ithe iterator in VTK 4.2. It would be nice to have some sort of efficient neighborhood iterators but so far we

haven't come up with any.

2. Rework the polydata and unstructured grid structures (vtkMesh ??). Related ideas include:
   - Make UnstructuredGrid more compact by removing the cell point count from the vtkCellArray. This will reduce the storage required by each cell by 4 bytes.
   - Make vtkPolyData an empty subclass of vtkUnstructuredGrid. There are a number of good reasons for this but it is a tricky task and backwards compatibility needs to be maintained.
3. More parallel support, including parallel compositing algorithms
4. Algorithms like LIC (http://www-courses.cs.uiuc.edu/~cs419/lic.pdf), maybe a couple terrain-decimation algorithms
5. Further integration of STL and other important C++ constructs (like templates)

VTK 4.4 (intermediate release, end of 2003)

- convert APIs to double (done)
- remove old callbacks (done)
- blanking
- ref count observers (done)
- switch collections to use iterators (done)
- improve copyright (done)

VTK 5.0 (major release, early 2005)

- new pipeline mechanism (see Pipeline.pdf)
- time support
- true AMR support

Changes to Interactors

The Interactors have been updated to use the Command/Observer events of vtk. The vtkRenderWindowInteractor now has ivars for all the event information. There is a new class called vtkGenericRenderWindowInteractor that can be used to set up the bindings from other languages like python, Java or TCl.

A new class vtkInteractorObserver was also added. It has a SetInteractor() method. It observes the keypress and delete events invoked by the render window interactor. The keypress activation value for a widget is now 'i' (although this can be programmed). vtkInteractorObserver has the state ivar Enabled. All subclasses must have the SetEnabled(int) method. Convenience methods like On(), Off(), EnabledOn(), and EnabledOff() are available. The state of the interactor observer is obtained using GetEnabled(). The SetEnabled(1) method adds observers to watch the interactor (appropriate to the particular interactor observer) ; SetEnabled(0) removes the observers . There are two new events: EnableEvent and DisableEvent which are invoked by the SetEnabled() method.

The events also support the idea of priority now. When you add an observer, you can specify a priority from 0 to 1. Higher values will be called back first. An observer can also tell the object not to call any more observers. This way you can handle an event, and stop further processing. In this way you can add handlers to InteractorStyles without sub-classing and from wrapped languages.

For more information see: vtkGenericRenderWindowInteractor, vtkRenderWindowInteractor, vtkInteractorObserver.

Header files and vtkSetObjectMacro

On some platforms such as MS Visual Studio .NET, compiler is not capable of handling too big input files. Some VTK files with all includes do become big enough to overwhelm the compiler. The solution is to minimize the amount of includes. This especially goes for header files, because they propagate to other files. Every class header file should include only the parent class header file. If there is no other alternative, you should put a comment next to include file explaining why the file has to be included.

The related issue is with vtkSetObjectMacro. This file calles some methods on an argument class, which implies that the argument class header file has to be included. The result is bloat on the header files. The solution is to not use vtkSetObjectMacro but vtkCxxSetObjectMacro. The difference is that vtkCxxSetObjectMacro goes in the Cxx file and not in the header file.

Example: Instead of

```
#include "vtkBar.h"
class vtkFoo : public vtkObject
{ ...
  vtkSetObjectMacro(Bar, vtkBar);
  ...
};
```

Do:

```
class vtkBar;
class vtkFoo : public vtkObject
{
  ...
  virtual void SetBar(vtkBar*);
  ...
};
```

and add the following line to vtkFoo.cxx

```
vtkCxxSetObjectMacro(vtkFoo,Bar,vtkBar);
```

Text properties in VTK 4.2

A new vtkTextProperty [24] class has been added to VTK 4.2.

This class factorizes text attributes that used to be spread out and duplicated in many different classes (mostly 2D actors and text mappers). Among those attributes, font family, font size, bold/italic/shadow properties, horizontal and vertical justification, line spacing and offset have been retained, whereas new attributes like color and opacity have been introduced.

We tried to make sure that you can use a vtkTextProperty to modify text properties in the same way a vtkProperty can be used to modify the surface properties of a geometric object. In that regard, you should be able to share a vtkTextProperty between different actors or assign the same vtkTextProperty to an actor that offers multiple vtkTextProperty attributes (vtkXYPlot [25] for example).

Here is a quick example:

```
vtkTextActor *actor0 = vtkTextActor::New();
actor0->GetTextProperty()->SetItalic(1);
                                                    //
vtkTextProperty *tprop = vtkTextProperty::New();
tprop->SetBold(1);
                                                    //
vtkTextActor *actor1 = vtkTextActor::New();
actor1->SetTextProperty(tprop);
                                                    //
vtkTextActor *actor2 = vtkTextActor::New();
```

```
actor2->SetTextProperty(tprop);
```

- Backward compatibility issues*:

1) Color and Opacity:

The text color and text opacity settings are now controlled by the vtkTextProperty Color and Opacity attributes instead of the corresponding actor's color and opacity attributes. In the following example, those settings were controlled by the attributes of the vtkProperty2D attached to the vtkActor2D (vtkTextActor). The vtkTextProperty attributes should be used instead:

```
vtkTextActor *actor = vtkActor::New();
actor->GetProperty()->SetColor(...);
actor->GetProperty()->SetOpacity(...);
```

becomes:

```
actor->GetTextProperty()->SetColor(...);
actor->GetTextProperty()->SetOpacity(...);
```

To make migration easier for a while, we have set the vtkTextProperty default color to *(-1.0, -1.0, -1.0)* and the default opacity to *-1.0*. These "magic" values are checked by the underlying text mappers at rendering time. If they are found, the color and opacity of the 2D actor's vtkProperty2D are used, just as it was in VTK 4.1.

2) API (i.e. SetBold(), SetItalic(), etc)

Most of the VTK classes involving text used to provide their own text attributes like Bold, Italic, Shadow, FontFamily. Thus, each of those classes would duplicate the vtkTextMapper API through methods like SetItalic(), SetBold(), SetFontFamily(), etc.

Moreover, if one class had different text elements (say, for example, the title and the labels of a scalar bar), there was no way to modify the text properties of these elements separately.

The vtkTextProperty class has been created to address both issues, by obsoleting those duplicated attributes and methods and providing a unified way to access text properties, and by allowing each class to associate different vtkTextProperty to different text elements.

Migrating your code basically involves using the old API on your actor's vtkTextProperty instead of the actor itself. For example:

```
actor->SetBold(1);
```

becomes:

```
actor->GetTextProperty()->SetBold(1);
```

When a class provides different vtkTextProperty for different text elements, the TextProperty attribute is usually prefixed with that element type. Example: AxisTitleTextProperty, or AxisLabelTextProperty. This allows you to set different aspect for each text elements. If you want to use the same properties, you can either set the same values on each vtkTextProperty, or make both vtkTextProperty point to the same vtkTextProperty object. Example:

```
actor->GetAxisLabelTextProperty()->SetBold(1);
actor->GetAxisTitleTextProperty()->SetBold(1);
```

or:

```
vtkTextProperty *tprop = vtkTextProperty::New();
tprop->SetBold(1);
actor->SetAxisLabelTextProperty(tprop);
```

```
actor->SetAxisTitleTextProperty(tprop);
```

or:

```
actor->SetAxisLabelTextProperty(actor->GetAxisTitleTextProperty());
actor->GetAxisTitleTextProperty()->SetBold(1);
```

The following list specifies the name of the text properties used in the VTK classes involving text.

vtkTextMapper [26]:

- you can still use the vtkTextMapper + vtkActor2D combination, but we would advise you to use a single vtkTextActor instead, this will give you maximum flexibility.
- has 1 text prop: TextProperty, but although you have access to it, do not twwak it unless you are using vtkTextMapper with a vtkActor2D. In all other cases, use the text prop provided by the actor (see below).

vtkTextActor [27]:

- has 1 text prop: TextProperty.

vtkLabeledDataMapper [28]:

- has 1 text prop: LabelTextProperty.

vtkCaptionActor2D [29]:

- has 1 text prop: CaptionTextProperty.

vtkLegendBoxActor [30]:

- has 1 text prop: EntryTextProperty.

vtkAxisActor2D [31], vtkParallelCoordinatesActor [32], and vtkScalarBarActor [33]:

- have 2 text props: TitleTextProperty, LabelTextProperty.

vtkXYPlotActor [25]:

- has 3 text prop: TitleTextProperty (plot title), AxisTitleTextProperty, AxisLabelTextProperty (title and labels of all axes)
- the legend box text prop (i.e. entry text prop) can be retrieved through actor->GetLegendBoxActor()->GetEntryTextProperty()
- the X (or Y) axis text props (i.e. title and label text props) can be retrieved through actor->GetX/YAxisActor2D->GetTitle/LabelTextProperty(), and will override the corresponding AxisTitleTextProperty or AxisLabelTextProperty props as long as they remain untouched.

vtkCubeAxesActor2D [34]:

- has 2 text props: AxisTitleTextProperty, AxisLabelTextProperty (title and label of all axes)
- the X (Y or Z) axis text props (i.e. title and label text props) can be retrieved through actor->GetX/Y/ZAxisActor2D->GetTitle/LabelTextProperty(), and will override the corresponding AxisTitleTextProperty or AxisLabelTextProperty props as long as they remain untouched.

Forward declaration in VTK 4.x

Since VTK 4.x all classes have been carefully inspected to only include the necessesary headers, and do what is called 'forward declaration' for all other needed classes. Thus, when you compile your projects using a filter that takes as input a dataset and you are passing an imagedata: you need to explicitely include imagedata within your implementation file. This is true for all data types.

For example, if you get this error:

```
no matching function for call to
`vtkContourFilter::SetInput(vtkImageData*)'
VTK/Filtering/vtkDataSetToPolyDataFilter.h:44:
candidates are: virtual void
vtkDataSetToPolyDataFilter::SetInput(vtkDataSet*)
```

This means you need to add in your code : #include "vtkImageData.h"

Using Volume Rendering in VTK

I recently updated my VTK CVS version. And my c++ code that use to work fine are now complaining about:

undefined reference to `vtkUnstructuredGridAlgorithm::SetInput(vtkDataObject*)' undefined reference to `vtkUnstructuredGridAlgorithm::GetOutput()'

There is now a new subfolder and a new option to enable building the VolumeRendering library. You have to turn VTK_USE_VOLUMERENDERING to ON in order to use it. Also make sure that your executable is linking properly to this new library:

```
ADD_EXECUTABLE(foo foo.cxx)
TARGET_LINK_LIBRARIES(foo vtkVolumeRendering)
```

API Changes in VTK 5.2

`vtkProp::RenderTranslucentGeometry()` is gone

`vtkProp::RenderTranslucentGeometry()` is gone and has been broken down into 3 methods:

- `HasTranslucentPolygonalGeometry()`
- `RenderTranslucentPolygonalGeometry()`
- `RenderVolumetricGeometry()`

Here is what to change in a vtkProp subclass:

- If `RenderTranslucentGeometry()` was used to render translucent polygonal geometry only, override `HasTranslucentPolygonalGeometry()` and `RenderTranslucentPolygonalGeometry()`. **Just renaming `RenderTranslucentGeometry()` as `RenderTranslucentPolygonalGeometry()` is not enough!**
- If `RenderTranslucentGeometry()` was used to render translucent volumetric geometry only, override `RenderVolumetricGeometry()`. In this case, just renaming `RenderTranslucentGeometry()` as `RenderVolumetricGeometry()` is OK.
- If `RenderTranslucentGeometry()` was used to render translucent polygonal geometry and translucent volumetric geometry, override all 3 methods.

The reason of this change is that `HasTranslucentPolygonalGeometry()` is used to decide if an expensive initialization of the new rendering algorithm of translucent polygonal geometry (depth peeling) is necessary. `RenderTranslucentPolygonalGeometry()` is called multiple times during the rendering of the translucent polygonal geometry of the scene. `RenderVolumetricGeometry()` is called in an additional pass, after depth

peeling. For this reason, **`RenderTranslucentGeometry()` cannot just be marked as deprecated but had to be removed from the API**.

`vtkImagePlaneWidget` has action names changed

from:

```
enum
{
  CURSOR_ACTION       = 0,
  SLICE_MOTION_ACTION = 1,
  WINDOW_LEVEL_ACTION = 2
};
```

to:

```
enum
{
  VTK_CURSOR_ACTION       = 0,
  VTK_SLICE_MOTION_ACTION = 1,
  VTK_WINDOW_LEVEL_ACTION = 2
};
```

`GetOutput()` now returns `vtkDataObject` for some algorithms

The following algorithms now work on `vtkGraph` as well as `vtkDataSet`, so no `GetOutput()` longer returns `vtkDataSet`. To obtain the dataset, use `vtkDataSet::SafeDownCast(filter->GetOutput())`

- `vtkArrayCalculator`
- `vtkAssignAttribute`
- `vtkProgrammableFilter`

API Changes in VTK 5.4

- empty right now.

API Changes in VTK 5.5

- vtkStreamTracer

Changed

```
enum Units
 {
 TIME_UNIT,
 LENGTH_UNIT,
 CELL_LENGTH_UNIT
 }
```

to

```
enum Units
 {
 LENGTH_UNIT = 1,
 CELL_LENGTH_UNIT = 2
```

```
}
```

Changed

- OUT_OF_TIME = 4

to

- OUT_OF_LENGTH = 4

in enum *ReasonForTermination*

Changed

- LastUsedTimeStep

to

- LastUsedStepSize

Changed

- MaximumPropagation
- MaximumIntegrationStep
- MinimumIntegrationStep
- InitialIntegrationStep

from type *IntervalInformation* to type *double*.

Added a member variable to the class

- int IntegrationStepUnit

The following APIs were **removed** from the class:

- void SetMaximumProgration(int unit, double max)
- void SetMaximumProgrationUnit(int unit)
- int GetMaximumPropagationUnit()
- void SetMaximumPropagationUnitToTimeUnit()
- void SetMaximumPropagationUnitToLengthUnit()
- void SetMaximumPropagationUnitToCellLengthUnit()
- void SetMinimumIntegrationStep(int unit, double step)
- void SetMinimumIntegrationStepUnit(int unit)
- int GetMinimumIntegrationStepUnit()
- void SetMinimumIntegrationStepUnitToTimeUnit()
- void SetMinimumIntegrationStepUnitToLengthUnit()
- void SetMinimumIntegrationStepUnitToCellLengthUnit()
- void SetMaximumIntegrationStep(int unit, double step)
- void SetMaximumIntegrationStepUnit(int unit)
- int GetMaximumIntegrationStepUnit()
- void SetMaximumIntegrationStepUnitToTimeUnit()
- void SetMaximumIntegrationStepUnitToLengthUnit()
- void SetMaximumIntegrationStepUnitToCellLengthUnit()
- void SetInitialIntegrationStep(int unit, double step)
- void SetInitialIntegrationStepUnit(int unit)
- int GetInitialIntegrationStepUnit()
- void SetInitialIntegrationStepUnitToTimeUnit()
- void SetInitialIntegrationStepUnitToLengthUnit()
- void SetInitialIntegrationStepUnitToCellLengthUnit()
- void SetIntervalInformation(int unit, double interval, IntervalInformation& currentValues)

- void SetIntervalInformation(int unit,IntervalInformation& currentValues)
- void ConvertIntervals(double& step, double& minStep, double& maxStep, int direction, double cellLength, double speed)
- static double ConvertToTime(IntervalInformation& interval, double cellLength, double speed)
- static double ConvertToLength(IntervalInformation& interval, double cellLength, double speed)
- static double ConvertToCellLength(IntervalInformation& interval, double cellLength, double speed)
- static double ConvertToUnit(IntervalInformation& interval, int unit, double cellLength, double speed)

The following APIs were added to the class:

- int GetIntegrationStepUnit()
- void SetIntegrationStepUnit(int unit)
- void ConvertIntervals(double& step, double& minStep, double& maxStep, int direction, double cellLength)
- static double ConvertToLength(double interval, int unit, double cellLength)
- static double ConvertToLength(IntervalInformation& interval, double cellLength)
- vtkInterpolatedVelocityField

Added a new member variable and two associated functions:

- bool NormalizeVector
- vtkSetMacro(NormalizeVector, bool)
- vtkGetMacro(NormalizeVector, bool)

OpenGL requirements

Terminology

- a software component using OpenGL (like VTK) **requires** some minimal version of OpenGL and some minimal set of OpenGL extensions at runtime. At compile time, it **requires** an OpenGL header file (`gl.h`) compatible with some minimal version of the OpenGL API.
- an OpenGL implementation (software (like Mesa) or hardware (combination of a graphic card and its driver)) **supports** some OpenGL versions and a set of extensions.

How do I check which OpenGL versions or extensions are supported by my graphic card or OpenGL implementation?

Linux/Unix

Two ways:

- General method

```
$ glxinfo
```

- vendor specific tool

if you have an nVidia card and nvidia-settings installed on it, run it and go to the OpenGL/GLX Information item under the X Screen 0 item.

Windows

You can download and use GLview http://www.realtech-vr.com/glview

Mac OS X

With Xcode installed, Macintosh HD->Developer->Applications->Graphic Tools->OpenGL Driver Monitor.app->Monitors->Renderer Info-><name of the OpenGL driver>->OpenGL Extensions

VTK 5.0

What is the minimal OpenGL version of the API required to compile VTK5.0?

The `gl.h` file provided by your compiler/system/SDK has to define at least the OpenGL 1.1 API.

(Note: the functions and macros defined in higher OpenGL API versions or in other OpenGL extensions are provided by `glext.h`, `glxext.h` and `wglext.h`. Those 3 files are official files taken from http://opengl.org/registry/and already part of the VTK source tree).

What is the minimal OpenGL version required by VTK5.0 at runtime?

All the VTK classes using OpenGL require an OpenGL implementation (software or hardware) >=1.1 except for `vtkVolumeTextureMapper3D`.

If you want to use `vtkVolumeTextureMapper3D`, the following extensions or OpenGL versions are required (at runtime):

- extension `GL_EXT_texture3D` or OpenGL>=1.2

and

- extension `GL_ARB_multitexture` or OpenGL>=1.3

and either:

- extensions `GL_ARB_fragment_program` and `GL_ARB_vertex_program`

or:

- extensions `GL_NV_texture_shader2` and `GL_NV_register_combiners` and `GL_NV_register_combiners2`

VTK 5.2

What is the minimal OpenGL version of the API required to compile VTK5.2?

Same answer than for VTK 5.0.

What is the minimal OpenGL version required by VTK5.2 at runtime?

All the VTK classes using OpenGL require an OpenGL implementation (software or hardware) >=1.1 except for `vtkVolumeTextureMapper3D`, `vtkHAVSVolumeMapper`, `vtkGLSLShaderProgram`, depth peeling and some hardware offscreen rendering using framebuffer objects (FBO).

If you want to use `vtkVolumeTextureMapper3D`, the following extensions or OpenGL versions are required (at runtime):

- extension `GL_EXT_texture3D` or OpenGL>=1.2

and

- extension `GL_ARB_multitexture` or OpenGL>=1.3

and either:

- extensions GL_ARB_fragment_program and GL_ARB_vertex_program

or:

- extensions GL_NV_texture_shader2 and GL_NV_register_combiners and GL_NV_register_combiners2

If you want to use vtkHAVSVolumeMapper, the following extensions or OpenGL versions are required (at runtime):

- OpenGL>=1.3
- GL_ARB_draw_buffers or OpenGL>=2.0
- GL_ARB_fragment_program
- GL_ARB_vertex_program
- GL_EXT_framebuffer_object
- either GL_ARB_texture_float or GL_ATI_texture_float

The following extension or OpenGL version is used by vtkHAVSVolumeMapper if provided (at runtime), but it is optional:

- GL_ARB_vertex_buffer_object or OpenGL>=1.5

If you want to use vtkGLSLShaderProgram, the following extensions or OpenGL versions are required (at runtime):

- OpenGL>=1.3
- GL_ARB_shading_language_100 or OpenGL>=2.0,
- GL_ARB_shader_objects or OpenGL>=2.0
- GL_ARB_vertex_shader or OpenGL>=2.0
- GL_ARB_fragment_shader or OpenGL>=2.0.

Depth peeling (see VTK Depth Peeling for more information) requires (at runtime):

- GL_ARB_depth_texture or OpenGL>=1.4
- GL_ARB_shadow or OpenGL>=1.4
- GL_EXT_shadow_funcs or OpenGL>=1.5
- GL_ARB_vertex_shader or OpenGL>=2.0
- GL_ARB_fragment_shader or OpenGL>=2.0
- GL_ARB_shader_objects or OpenGL>=2.0
- GL_ARB_occlusion_query or OpenGL>=1.5
- GL_ARB_multitexture or OpenGL>=1.3
- GL_ARB_texture_rectangle
- GL_SGIS_texture_edge_clamp or GL_EXT_texture_edge_clamp or OpenGL>=1.2

Hardware-based offscreen rendering using framebuffer object (FBO) will be used as the default offscreen method if the following extensions or OpenGL version are available (at runtime):

- GL_EXT_framebuffer_object

and either

- GL_ARB_texture_non_power_of_two or OpenGL>=2.0

or

- GL_ARB_texture_rectangle

In addition, if the the framebuffer needs a stencil buffer, extension GL_EXT_packed_depth_stencil is required. Even if all those extensions are supported, the chosen FBO format might not be supported by the card; in this case, this method of offscreen rendering is not used.

Miscellaneous questions

Can't you split up the data file?

The data is now in one file that is about 15 Megabytes. This is smaller than the original data files VTK used and we hope that this size is not a problem for people anymore. If it is please let us know.

Do you have any shared library tips?

VTK version 4.0 and later supports both shared and static libraries on most all platforms. For development we typically use shared libraries since they are faster to link when making small changes. You can control how VTK builds by setting the BUILD_SHARED_LIBS option in CMake.

Legal issues

Is VTK FDA-Approved ?

Given the fact that VTK is a software toolkit, it cannot be the subject of FDA approval as a medical device. We have discussed this topic in several occasions and received advice from FDA representatives, that can be summarized as follow:

```
VTK is to be considered as an off-the-shelf (OTS) product that
is used for supporting a higher level medical application/product.
The developer of such application/product will be responsible for
performing the validation processes described in FDA published
guidelines for the development of software-related medical devices.
```

For mode details see the page FDA Guidelines for Software Development

What are the legal issues?

The Visualization Toolkit software is provided under the following copyright. We think it's pretty reasonable. We do restrict the distribution of modified code. This is primarily a revision control issue. We don't want a bunch of renegade vtks running around without us having some idea why the changes were made and giving us a chance to incorporate them into the general release.

The text of the VTK copyright is available here [35].

What is the deal with the patents

As the copyright mentions there are some patents used in VTK. If you use any code in the Patented/ directory for commercial application you should contact the patent holder and obtain a license.

As of VTK4.0 the following classes are known to use algorithms patented by General Electric Company: vtkDecimate, vtkMarchingCubes, vtkMarchingSquares, vtkDividingCubes, vtkSliceCubes and vtkSweptSurface. The GE contact is:

```
      Carl B. Horton
      Sr. Counsel, Intellectual Property
      3000 N. Grandview Blvd., W-710
      Waukesha, WI  53188
      Phone:  (262) 513-4022
      E-Mail: mailto:Carl.Horton@med.ge.com
```

As of VTK4.0 the following classes are known to use algorithms patented by Kitware, Inc.: vtkGridSynchronizedTemplates3D, vtkKitwareContourFilter.h, vtkSynchronizedTemplates2D, and vtkSynchronizedTemplates3D. The Kitware contact is:

```
Ken Martin
Kitware
28 Corporate Drive, Suite 204,
Clifton Park, NY 12065
Phone:1-518-371-3971
E-Mail: mailto:kitware@kitware.com
```

Can VTK be used as part of a project distributed under a GPL License ?

Short Answer

Yes, it is fine to take VTK code and to include it in a project that is distributed under a GPL license.

Long Answer

Terms

Let's call project X the larger project that:

1. Will include source code from VTK (in part or as a whole)
2. Will be distributed under GPL license

Note in particular that:

1. The copyright notices in VTK files must be kept.
2. If VTK files are modified by the developers of project X, that fact must be clearly indicated.
3. Only the modifications of VTK files made by the developers of project X will be covered by a GPL license. The original VTK code remains covered by the VTK license.
4. The collection of copyrighted works (project X in this case), that includes VTK (in part or as a whole) and their software will be covered by a GPL license.

Details

As the VTK license [35] is a variation of the Modified BSD license [36], to which only the following term has been added:

```
Modified source versions must be plainly marked as such,
and must not be misrepresented as being the original software.
```

and that the Modified BSD license is itself compatible with the GPL

http://www.gnu.org/philosophy/license-list.html (Modified BSD license)

Then the VTK license is also compatible with the GPL license. Since the terms of the GPL license do not preclude the additional term of the VTK license from being followed.

NOTE: The licenses are only **one way compatible**.

- You can use VTK code inside a GPL licensed project.
- You **can not** use GPL licensed code inside VTK.

That is the reason why there are no GPL third party libraries in VTK. Having GPL third party libraries in VTK would prevent closed source projects from being built against VTK.

VTK: [Welcome | Site Map [37]]

References

[1] http://www.kitware.com/products/vtktextbook.html
[2] http://www.vtk.org/Bug
[3] http://www.kitware.com/products/vtkguide.html
[4] http://www.vtk.org/get-software.php
[5] http://www.kitware.com/products/proftrain.html#VTKCourse
[6] http://www.kitware.com/products/proftrain.html
[7] http://www.catb.org/~esr/faqs/smart-questions.html
[8] http://www.mikeash.com/getting_answers.html
[9] http://public.kitware.com/pipermail/vtkusers/2003-October/070054.html
[10] http://www.vtk.org/doc/nightly/html/classvtkDICOMImageReader.html
[11] http://www.dclunie.com/medical-image-faq/html/part8.html
[12] http://gdcm.sourceforge.net
[13] http://gdcm.sourceforge.net/html/classvtkGDCMImageReader.html
[14] http://gdcm.sourceforge.net/html/classvtkGDCMImageWriter.html
[15] http://apps.sourceforge.net/mediawiki/gdcm/index.php?title=Gdcmconv
[16] http://www.vtk.org/doc/nightly/html/classvtkAppendPolyData.html
[17] http://www.vtk.org/doc/nightly/html/classvtkDepthSortPolyData.html
[18] http://public.kitware.com/cgi-bin/viewcvs.cgi/*checkout*/Hybrid/Testing/Tcl/depthSort.tcl?root=VTK&content-type=text/plain
[19] http://public.kitware.com/pipermail/vtkusers/2004-August/075966.html
[20] http://developer.apple.com/documentation/MacOSX/Conceptual/BPInternational/Articles/InternatSupport.html#//apple_ref/doc/uid/20000278-73764
[21] http://www.vtk.org/Bug/bug.php?op=show&bugid=2025
[22] http://developer.apple.com/documentation/MacOSX/Conceptual/BPRuntimeConfig/Articles/ConfigFiles.html#//apple_ref/doc/uid/20002091-SW1
[23] http://www.microsoft.com/whdc/system/platform/server/PAE/PAEmem.mspx
[24] http://public.kitware.com/VTK/doc/nightly/html/classvtkTextProperty.html
[25] http://www.vtk.org/doc/nightly/html/classvtkXYPlotActor.html
[26] http://www.vtk.org/doc/nightly/html/classvtkTextMapper.html
[27] http://www.vtk.org/doc/nightly/html/classvtkTextActor.html
[28] http://www.vtk.org/doc/nightly/html/classvtkLabeledDataMapper.html
[29] http://www.vtk.org/doc/nightly/html/classvtkCaptionActor2D.html
[30] http://www.vtk.org/doc/nightly/html/classvtkLegendBoxActor.html
[31] http://www.vtk.org/doc/nightly/html/classvtkAxisActor2D.html
[32] http://www.vtk.org/doc/nightly/html/classvtkParallelCoordinatesActor.html
[33] http://www.vtk.org/doc/nightly/html/classvtkScalarBarActor.html
[34] http://www.vtk.org/doc/nightly/html/classvtkCubeAxesActor2D.html
[35] http://www.vtk.org/copyright.php
[36] http://www.opensource.org/licenses/bsd-license.php
[37] http://www.vtk.org/Wiki/Category:VTK

Article Sources and Contributors

VTK/FAQ *Source*: http://www.vtk.org/Wiki/index.php?oldid=19102 *Contributors*: Akochanowska, Amaclean, Amoebahop, Amy, Andy, Andyisaweasle, Berk, Cowell, David.cole, Daviddoria, Davidmarshburn, Dcthomp, Dgobbi, DssaSad, DssaSadasdf, Fbertel, Francoisbertel, Frm, Gheorghe, Goodwin, Ibanez, Idb, Jasonphysics, Jean-pierre.roux, Jeff, Jonj, Juergen Mueller, Ken.martin@kitware.com, Kmorel, M Little, Mathieu, Mofolllom, Oncualtuntas, Pengo, Pschmitt, Quim, Seanm, Seanmcbride, SsSa, SssSss1, Tfogal, Tim.hutton, Tuner, Turner, Wschroed, Zack, Zhanping.liu

License