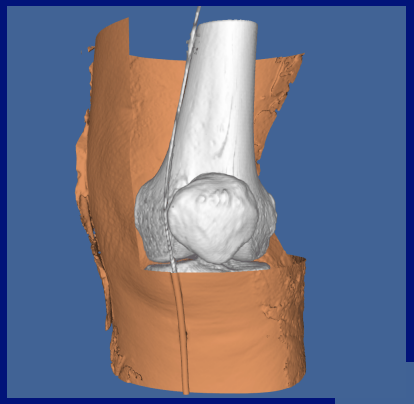
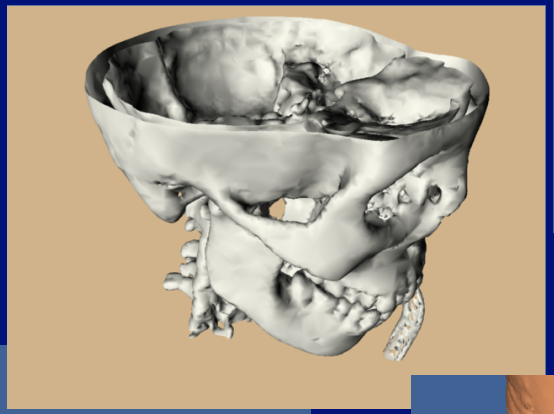
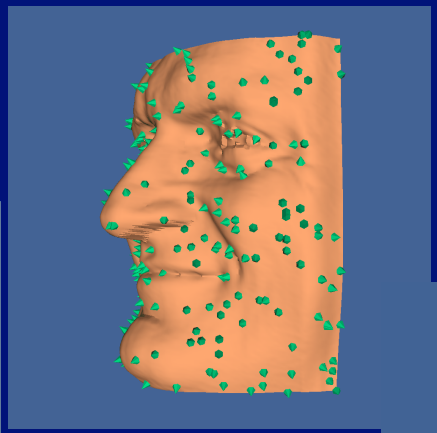
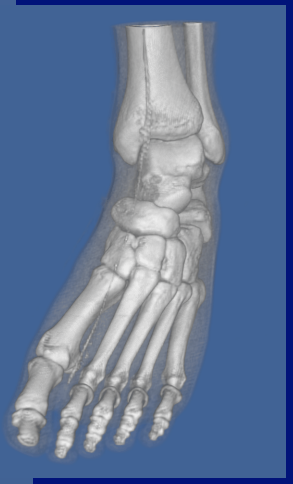
The Visualization Toolkit (VTK)

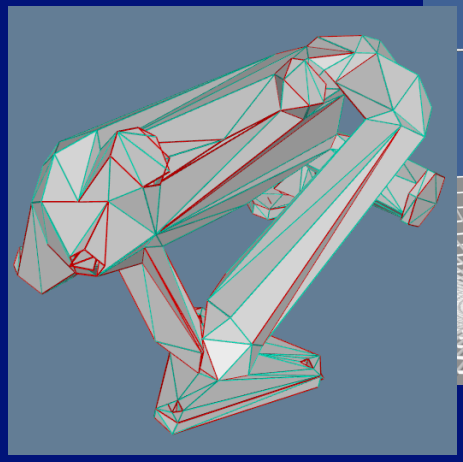
Overview

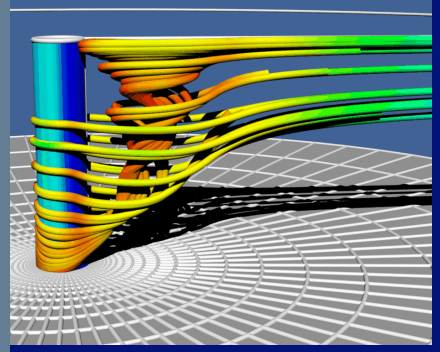


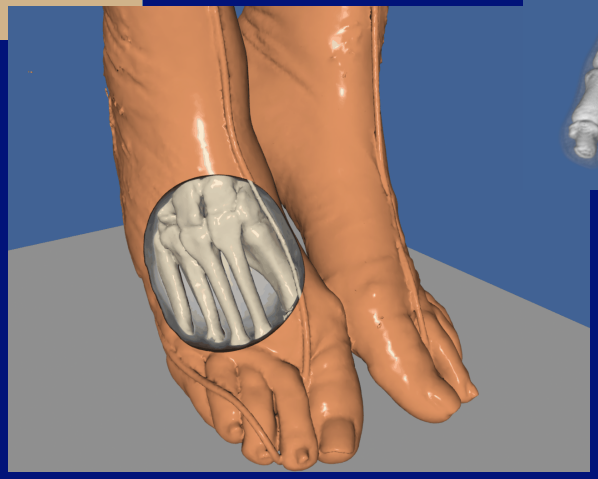












The Visualization Toolkit
An Overview

William J. Schroeder

Kitware, Inc.

Agenda

- VTK Technology
  - Background
  - The Graphics Model
  - The Visualization Model
  - Volume Rendering
- VTK Process
  - Open Source
  - Development Process

Agenda

- VTK Technology
  - Background
  - The Graphics Model
  - The Visualization Model
  - Volume Rendering
- VTK Process
  - Open Source
  - Development Process

Visualization

A definition for visualization

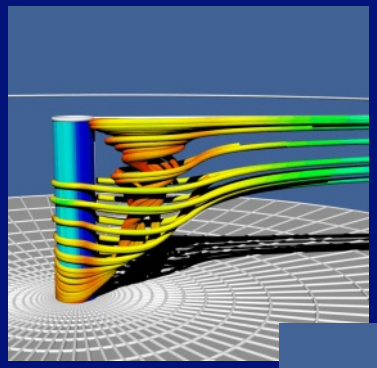
- Map data or information into images or other sensory input (touch, sound, smell, taste)
- Engages human perception system
- Simple, effective powerful
  - Complex data
  - Voluminous data

Related Fields

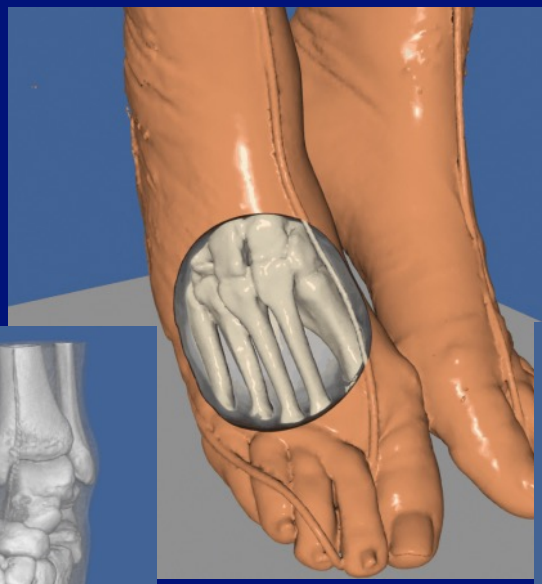
- Visualization
  - Scientific
  - Information
  - Financial
  - Data
  - Multivariate
- Image Processing
- 2D Graphics
- Statistical Graphics
  - Mapping
  - 3D Graphics
  - Volume Rendering
  - Haptics (touch)
  - Gaming / Entertainment
  - 3D GUI's
  - Virtual Reality

Example Applications

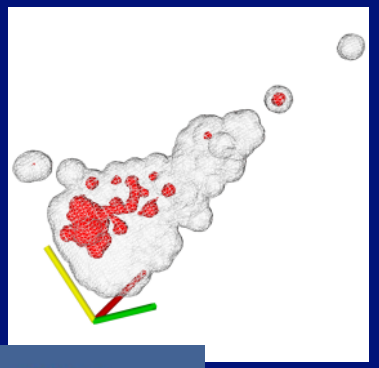
Scientific, medical, financial, information, ...



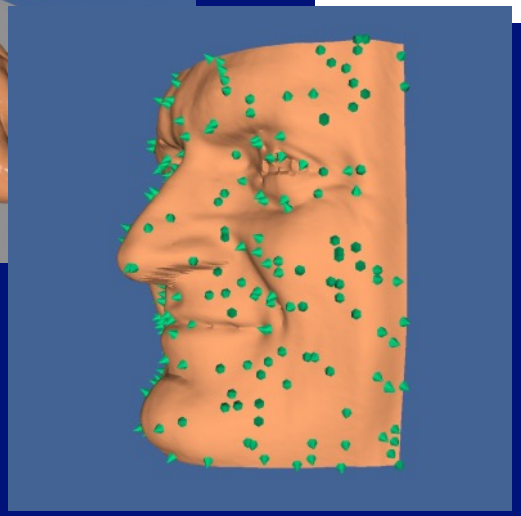
Simulation



Medical
CT / MRI / Ultrasound



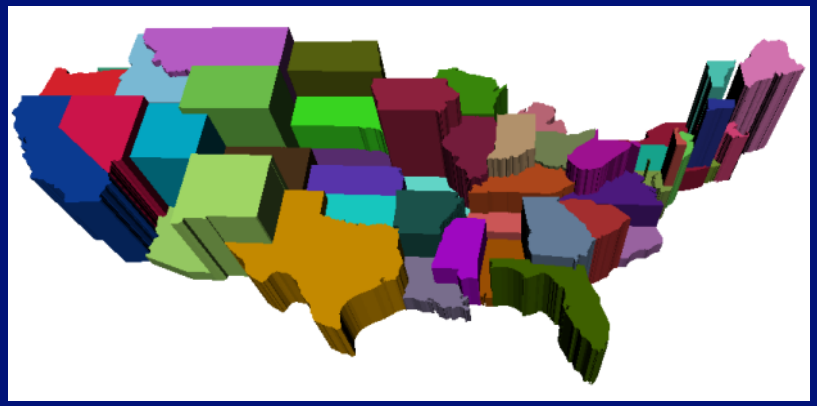
Business

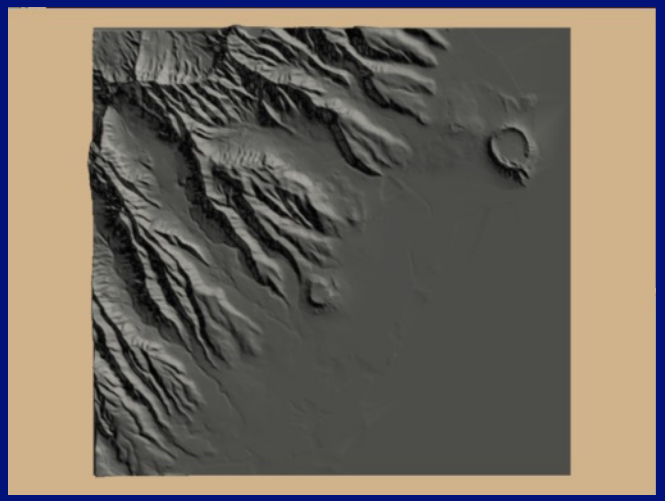


Modeling

Example Applications

Geophysical / Mapping



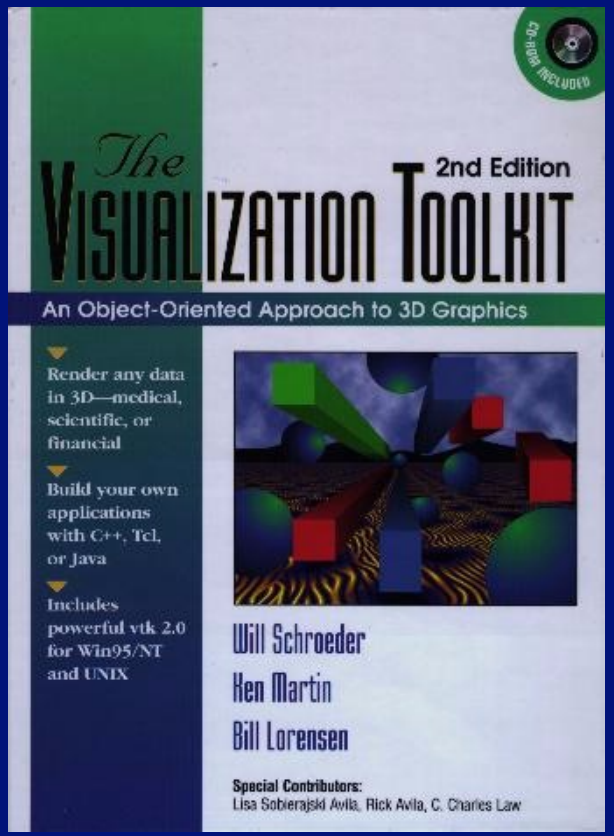


Trends

- The 3D Graphics Market is Exploding
  - Gaming driving the hardware
  - VRML, Web applications
  - Virtual Reality
  - Graphics standards (OpenGL, Direct3D)
  - Digital / 3D Data
- Data is overwhelming users
  - Visualization is a proven solution
- Open Source systems are credible
  - Even respectable

Textbook

Now in Second Edition



The Visualization Toolkit
An Object-Oriented Approach To 3D Graphics
Will Schroeder, Ken Martin, Bill Lorensen
ISBN 0-13-954694-4
Prentice Hall

Work on first edition began in 1994

What Is VTK?

A visualization toolkit

- Designed and implemented using object-oriented principles
- C++ class library (250,000 LOC, <100,000 executable lines)
- Automated Java, TCL, Python bindings
- Portable across Unix, Windows9x/NT
- Supports 3D/2D graphics, visualization, image processing, volume rendering

Other Features

- Reference Counting
- Abstract / Virtual Method Access
  - data stored in native type (byte, short, int, etc.)
  - data accessed in native type (with templates) or using generic interface (float)
- Data In Memory
  - some objects read pieces (vtkSliceCubes)
  - data pipeline can stream pieces based on memory limitations (more later)

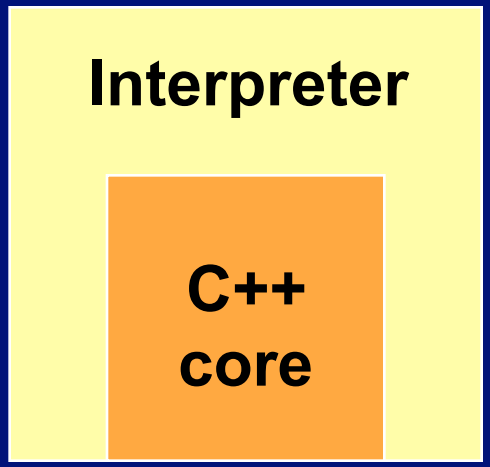
VTK Is Not a System

- Embeddable
  - Plays with other software
- Separable
  - Can pull out “pieces”
- Adaptable
  - Not dependent on GUI
  - Not dependent on rendering library

VTK Architecture

Hybrid approach

- compiled C++ (faster algorithms)
- interpreted applications (rapid development) (Java, Tcl, Python)
- A *toolkit*



Interpreted layer generated automatically

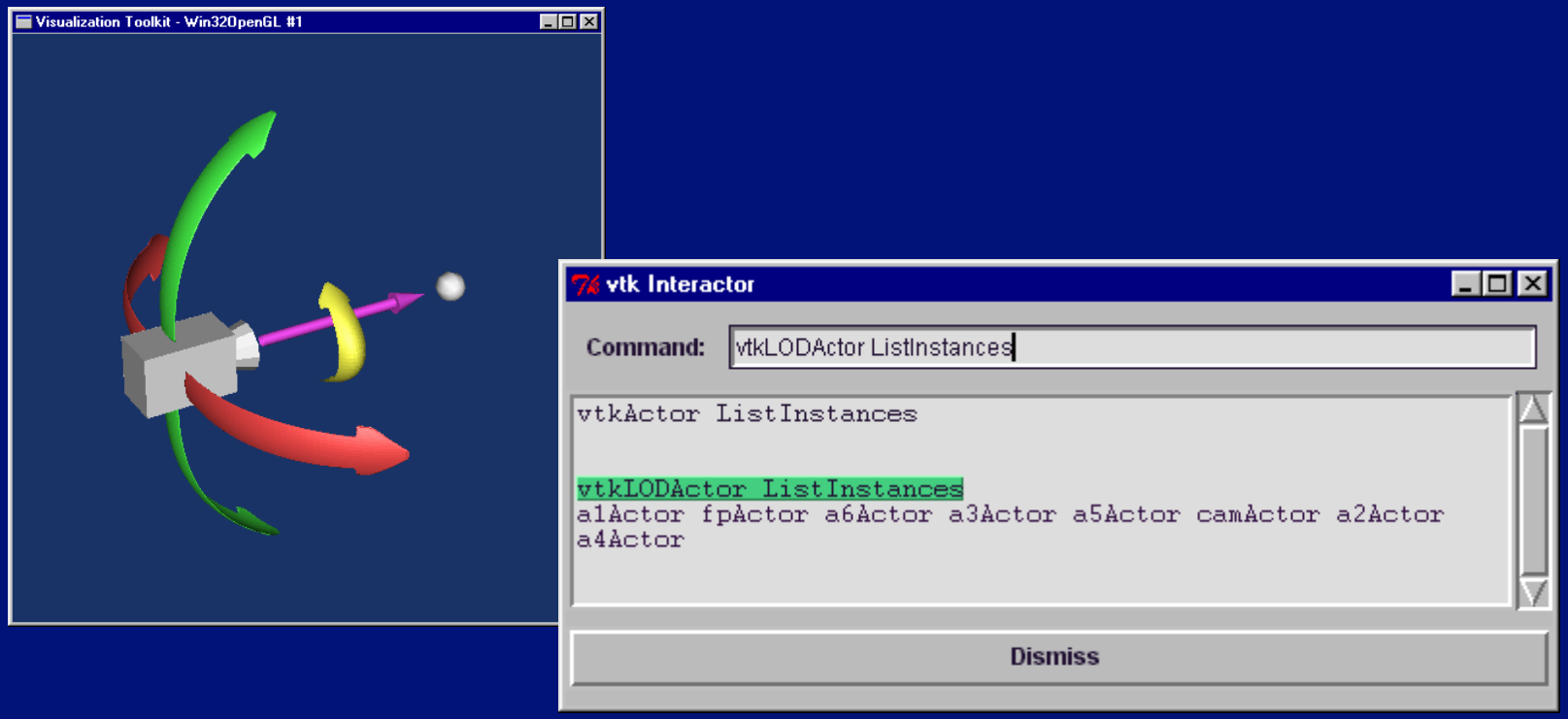
Interpreters

- Tcl
- Java
- Python

Interpreters provide faster turn-around;
suffer from slower execution

Tcl Interpreter

- source vtkInt.tcl (define interpreter GUI)

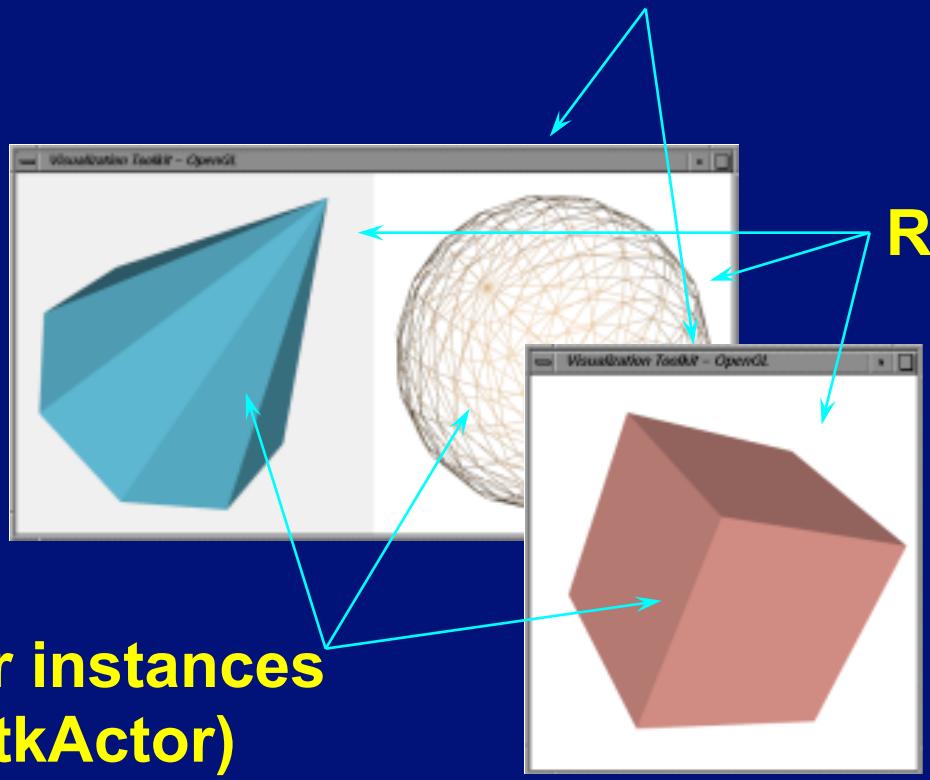


Agenda

- VTK Technology
  - Background
  - The Graphics Model
  - The Visualization Model
  - Volume Rendering
- VTK Process
  - Open Source
  - Development Process

Graphics Model

Instances of render window (vtkRenderWindow)



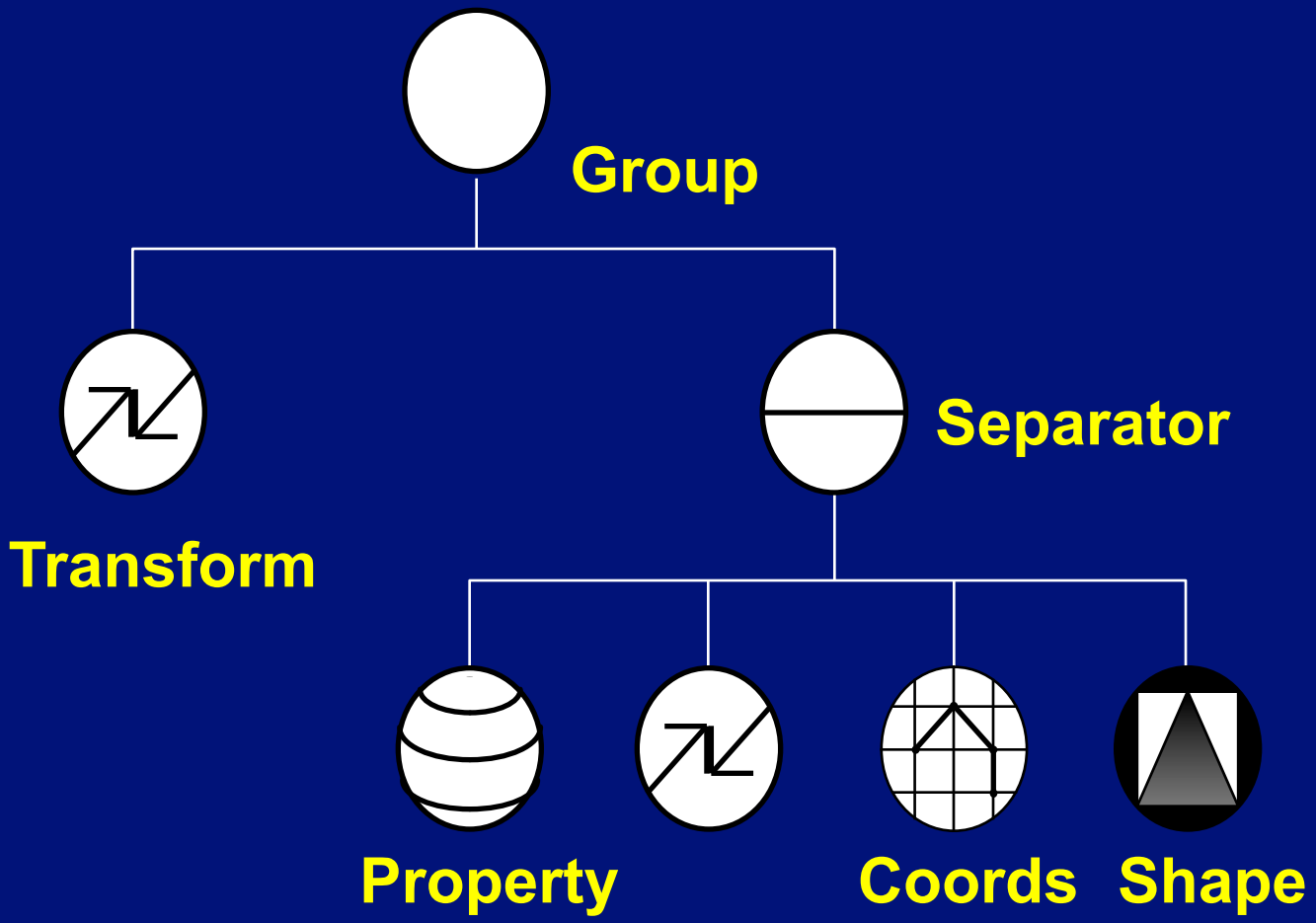
Renderer instances (vtkRenderer)

Actor instances (vtkActor)

Graphics Model

- RenderWindow - contains final image
- Renderer - *draws into render window*
- Actor - *combines properties / geometry*
- Lights - illuminate actors
- Camera - renders scene
- Mappers - represent geometry
- Transformations - position actors

In Context: The Scene Graph

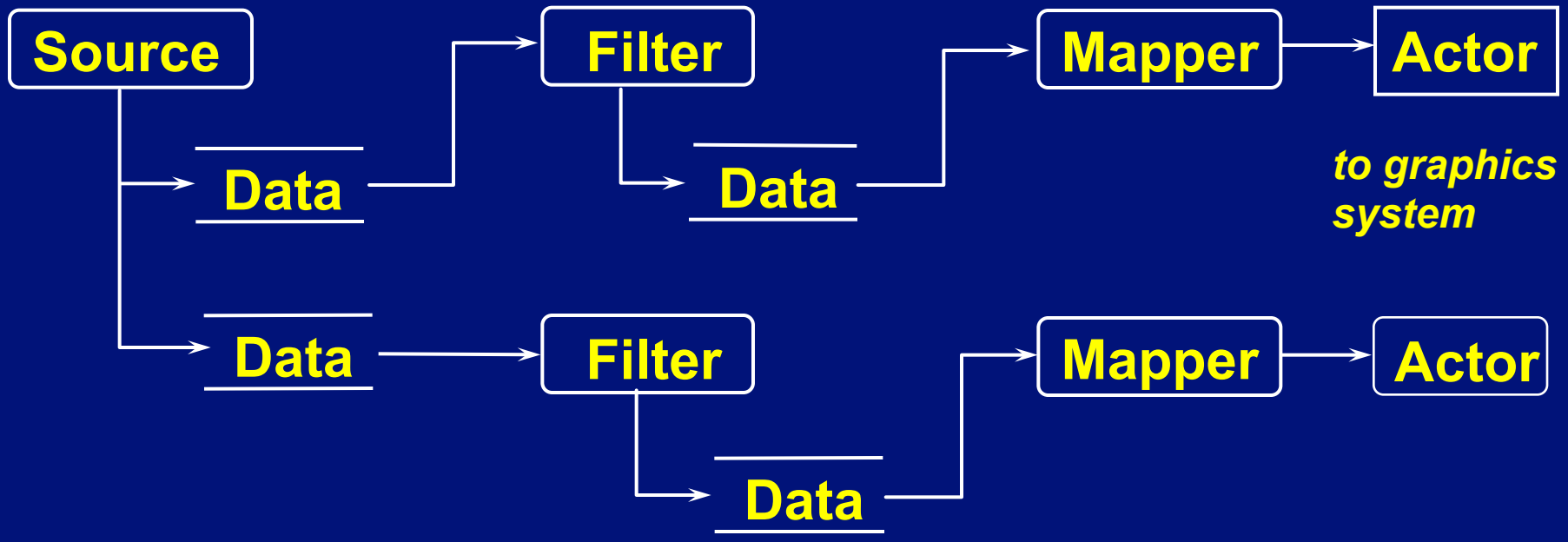


Agenda

- VTK Technology
  - Background
  - The Graphics Model
  - The Visualization Model
  - Volume Rendering
- VTK Process
  - Open Source
  - Development Process

What Is The Visualization Pipeline?

A sequence of *process objects* that operate on *data objects* to generate geometry that can be rendered by the graphics engine



Visualization Model

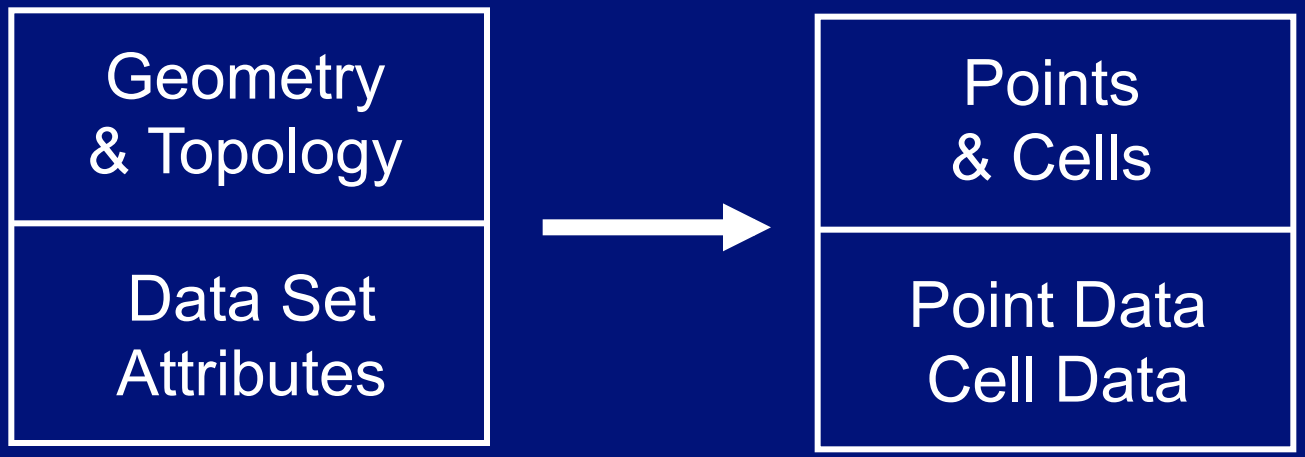
- Data Objects
  - represent data
  - provide access to data
  - compute information particular to data (e.g., bounding box, derivatives)
- Process Objects
  - Ingest, transform, and output data objects
  - represent visualization algorithms

Data Objects

- Represent a “blob” of data
  - contain instance of vtkFieldData
  - an array of arrays
  - no geometric/topological structure
  - typically not used in pipelines (but its subclasses such as vtkDataSet are)
- Can be converted to vtkDataSet
  - vtkDataObjectToDataSetFilter
  - advanced topic (see financialField.tcl)

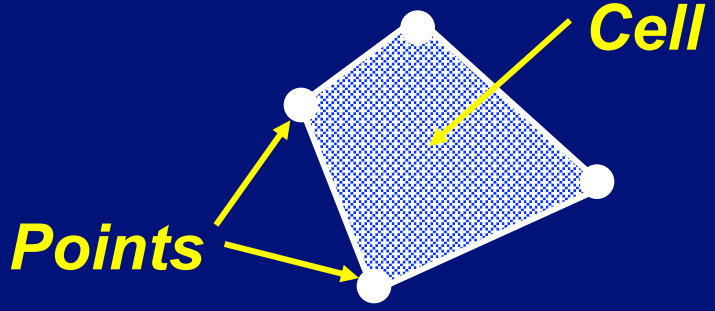
Data Objects / Data Sets

- vtkDataObject is a “blob” of data
  - Contains an instance of vtkFieldData
- vtkDataSet is data with geometric & topological structure; and with attribute data



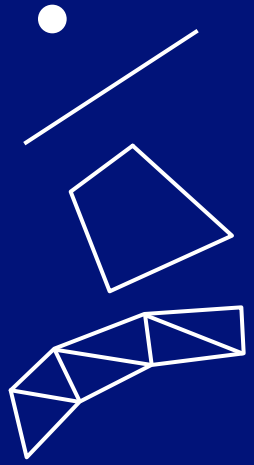
Dataset Model

- A dataset is a data object with structure
- Structure consists of
  - cells (e.g., polygons, lines, voxels)
  - points (x-y-z coordinates)
  - cells defined by connectivity list referring to points
  - implicit representations
  - explicit representations

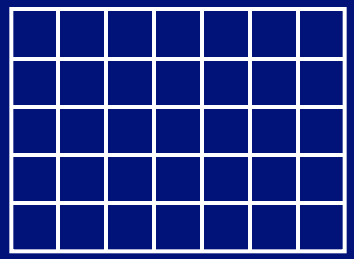


Dataset Types

vtkPolyData

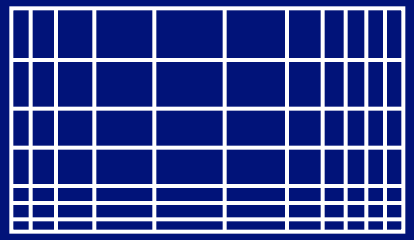


vtkStructuredPoints

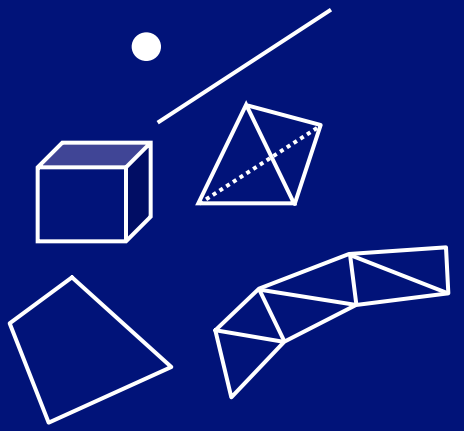


vtkStructuredGrid

vtkRectilinearGrid



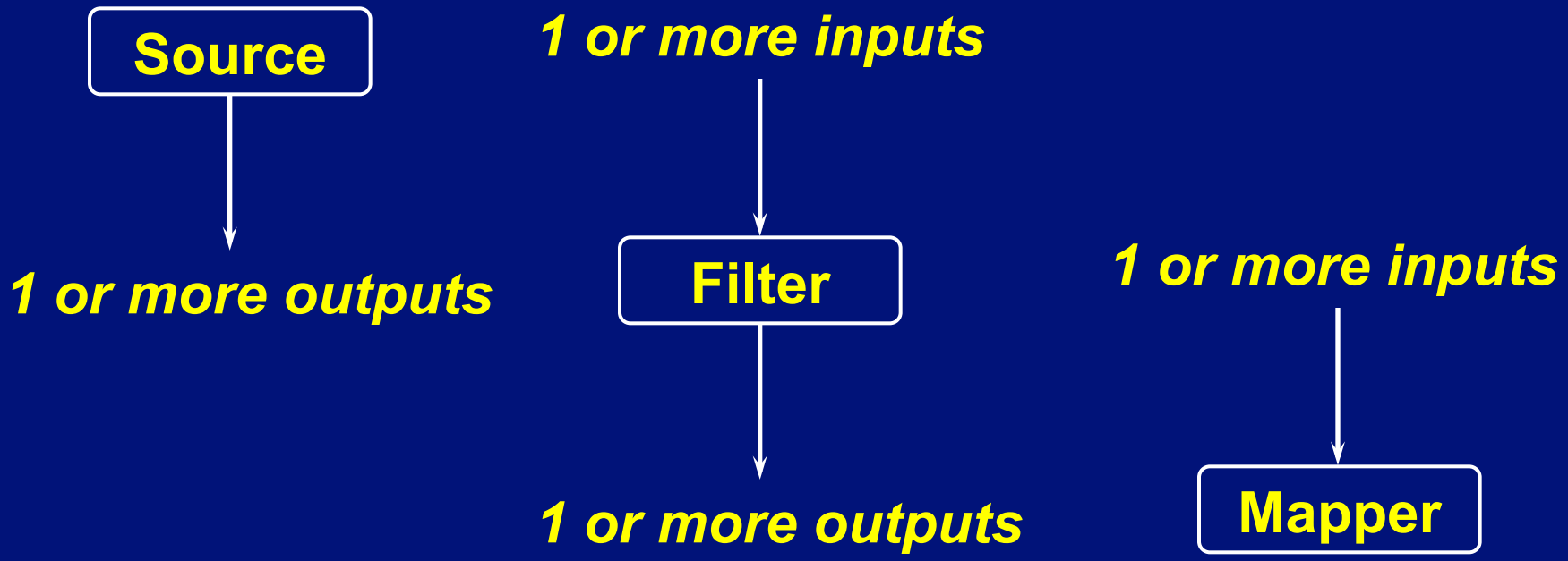
vtkUnstructuredGrid



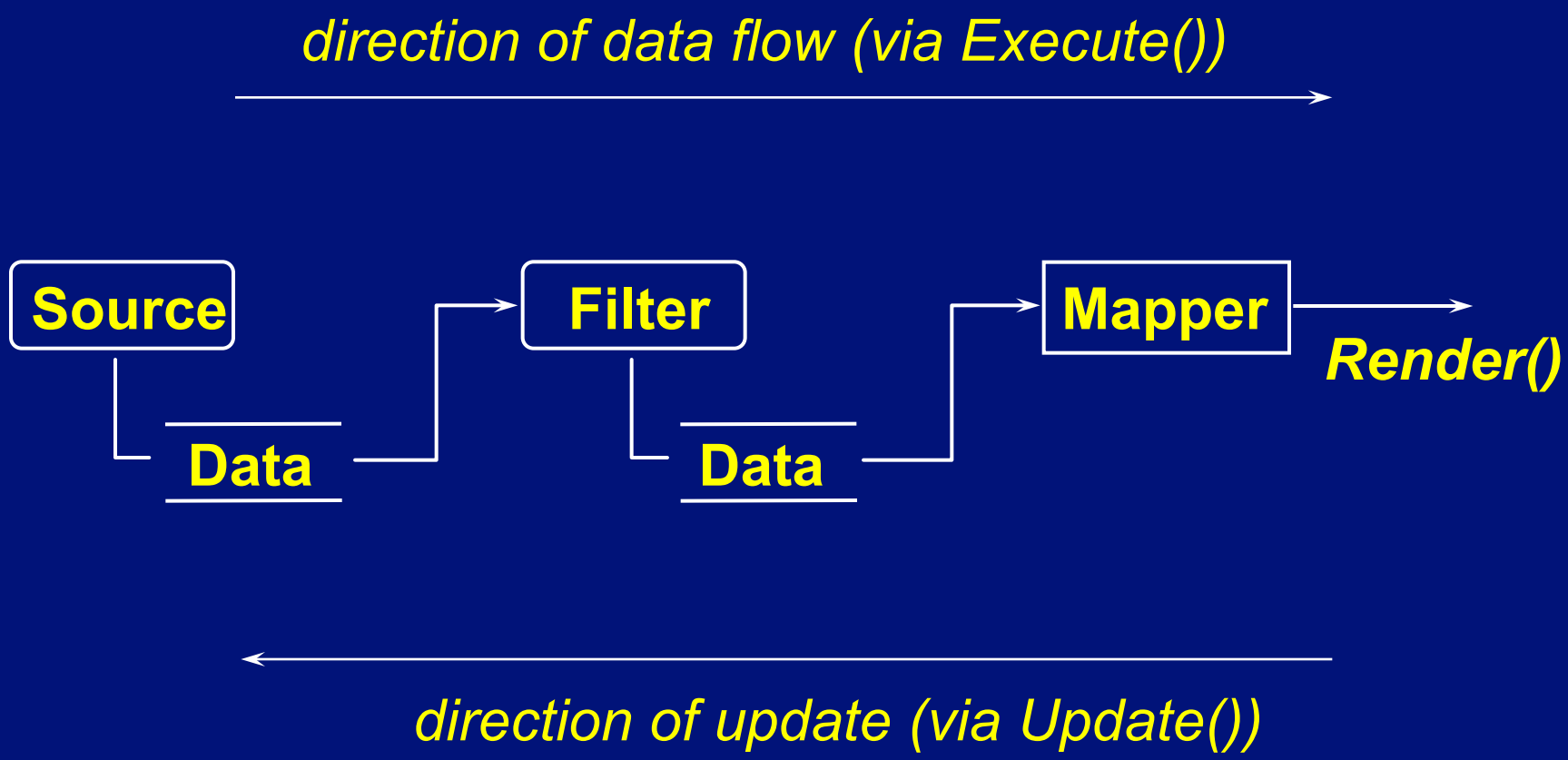
Data Set Attributes

- **Scalars** - 1-4 values (*vtkScalars*)
  - single value ranging to RGBA color
- **Vectors** - 3-vector (*vtkVectors*)
- **Tensors** - 3x3 symmetric matrix (*vtkTensors*)
- **Normals** - unit vector (*vtkNormals*)
- **Texture Coordinates** 1-3D (*vtkTCoords*)
- **Field Data** (an array of arrays) (*vtkFieldData*)

Process Objects



Pipeline Execution Model



Creating Pipeline Topology

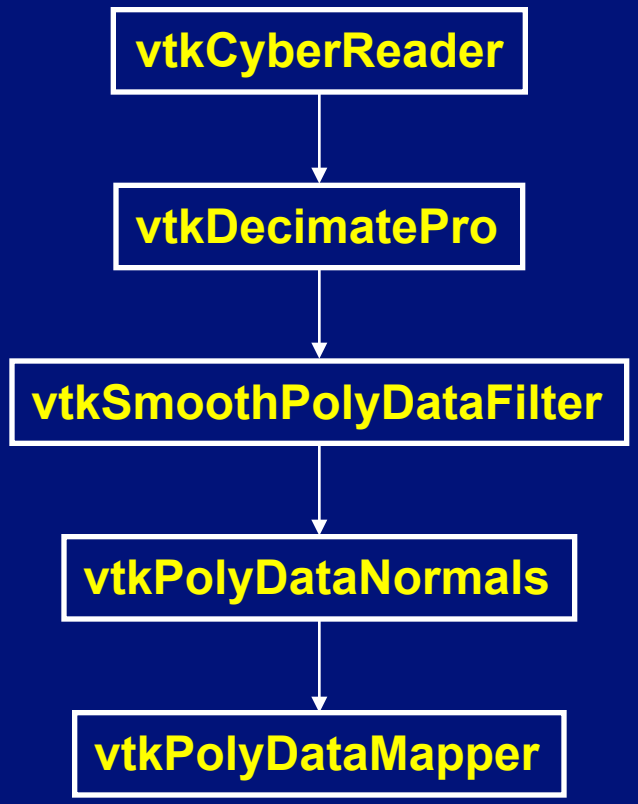
- aFilter->SetInput(bFilter->GetOutput());

- The Role of Type-Checking
  - SetInput() accepts dataset type or subclass
  - C++ compile-time checking
  - Interpreter run-time checking

Example Pipeline

- Decimation, smoothing, normals
- Implemented in C++

Note: data objects are not shown -> they are implied from the output type of the filter



Create Reader & Decimator

```
vtkCyberReader *cyber = vtkCyberReader::New();
    cyber->SetFileName("../../vtkdata/fran_cut”);

vtkDecimatePro *deci = vtkDecimatePro::New();
    deci->SetInput( cyber->GetOutput() );
    deci->SetTargetReduction( 0.9 );
    deci->PreserveTopologyOn();
    deci->SetMaximumError( 0.0002 );
```

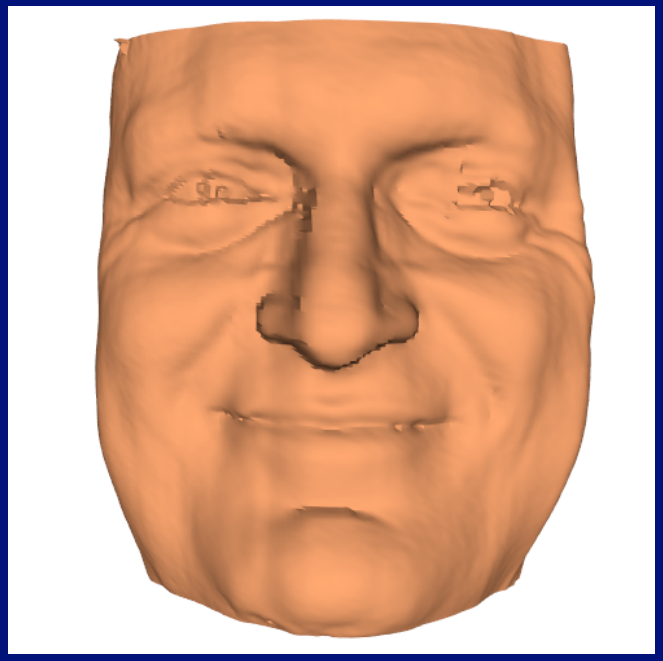
Smoother & Graphics Objects

```
vtkSmoothPolyDataFilter *smooth = vtkSmoothPolyDataFilter::New();
    smooth->SetInput(deci->GetOutput());
    smooth->SetNumberOfIterations( 20 );
    smooth->SetRelaxationFactor( 0.05 );

vtkPolyDataNormals *normals = vtkPolyDataNormals::New();
    normals->SetInput( smooth->GetOutput() );

vtkPolyDataMapper *cyberMapper = vtkPolyDataMapper::New();
    cyberMapper->SetInput( normals->GetOutput() );

vtkActor *cyberActor = vtkActor::New();
    cyberActor->SetMapper (cyberMapper);
    cyberActor->GetProperty()->SetColor ( 1.0, 0.49, 0.25 );
    cyberActor->GetProperty()->SetRepresentationToWireframe();
```

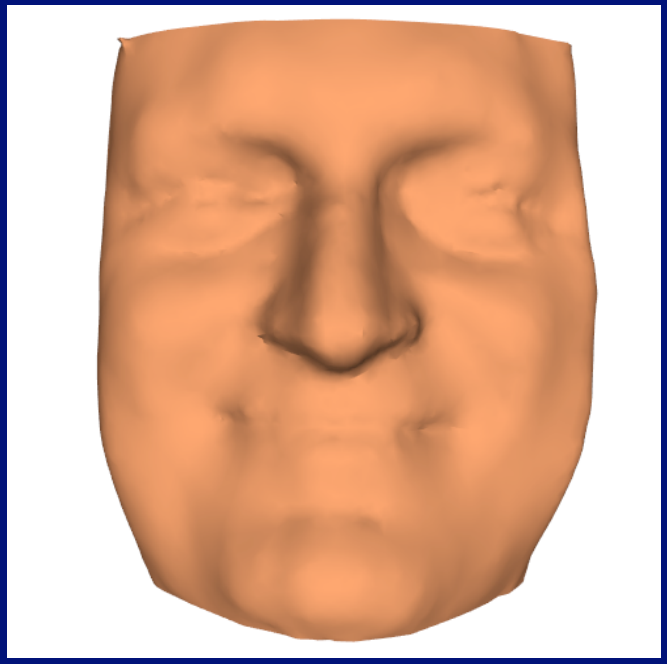
More Graphics Objects

```
vtkRenderer *ren1 = vtkRenderer::New();

vtkRenderWindow *renWin = vtkRenderWindow::New();
   renWin->AddRenderer( ren1 );

vtkRenderWindowInteractor *iren =
          vtkRenderWindowInteractor ::New();
   iren->SetRenderWindow( renWin );

ren1->AddActor( cyberActor );
ren1->SetBackground( 1, 1, 1 );
renWin->SetSize( 500, 500 );

iren->Start();
```

Results



Before
(52,260 triangles)



After Decimation
and Smoothing
(7,477 triangles)

Filter Overview: Sources

Readers

- vtkOBJReader
- vtkBYUReader
- vtkCyberReader
- vtkDataSetReader
- vtkMCubesReader
- vtkPLOT3DReader
- vtkPolyDataReader
- vtkRectilinearGridReader
- vtkSLCReader
- vtkSTLReader
- vtkStructuredGridReader
- vtkStructuredPointsReader
- vtkUnstructuredGridReader
- vtkVolume16Reader
- vtkFieldDataReader
- vtkBMPReader
- vtkPNMReader
- vtkTIFFReader

Sources

Procedural Sources

- vtkEarthSource
- vtkConeSource
- vtkCylinderSource
- vtkDiskSource
- vtkLineSource
- vtkOutlineSource
- vtkPlaneSource
- vtkPointSource
- vtkTextSource
- vtkVectorText
- vtkSphereSource
- vtkTexturedSphereSource
- vtkAxes
- vtkCursor3D
- vtkProgrammableSource
- vtkPointLoad

Filters

- vtkAppendFilter
- vtkAppendPolyData
- vtkBooleanTexture
- vtkBrownianPoints
- vtkCastToConcrete
- vtkCellCenters
- vtkCellDataToPointData
- vtkCullVisiblePoints
- vtkCleanPolyData
- vtkClipPolyData
- vtkClipVolume
- vtkConnectivityFilter
- vtkContourFilter
- vtkCutter
- vtkDashedStreamLine
- vtkDecimate
- vtkDecimatePro
- vtkDelaunay2D
- vtkDelaunay3D
- vtkDicers

Filters (2)

- vtkEdgePoints
- vtkElevationFilter
- vtkExtractEdges
- vtkExtractGeometry
- vtkExtractGrid
- vtkExtractTensorComponents
- vtkExtractUnstructuredGrid
- vtkExtractVOI
- vtkExtractVectorComponents
- vtkFeatureEdges
- vtkGaussianSplatter
- vtkGeometryFilter
- vtkGlyph3D
- vtkHedgeHog
- vtkHyperStreamline
- vtkIdFilter
- vtkLinearExtrusionFilter
- vtkMaskPolyData
- vtkOutlineFilter
- vtkPointDataToCellData

Filters (3)

- vtkProgrammableFilter
- vtkProjectedTexture
- vtkRecursiveDividingCubes
- vtkReverseSense
- vtkRibbonFilter
- vtkRotationalExtrusionFilter
- vtkShepardMethod
- vtkShrinkFilter
- vtkShrinkPolyData
- vtkSmoothPolyDataFilter
- vtkMaskPoints
- vtkMaskPolyData
- vtkMergeFilter
- vtkMergePoints
- vtkPolyDataNormals
- vtkProbeFilter
- vtkProgrammableAttributeDataFilter
- vtkSelectVisiblePoints
- vtkSpatialRepresentationFilter
- vtkStreamLine

Filters (4)

- vtkStreamPoints
- vtkStripper
- vtkStructuredGridGeometryFilter
- vtkStructuredGridOutlineFilter
- vtkStructuredPointsGeometryFilter
- vtkTensorGlyph
- vtkTextureMapToBox
- vtkTextureMapToCylinder
- vtkTextureMapToPlane
- vtkTextureMapToSphere
- vtkTexturedSphereSource
- vtkThreshold
- vtkThresholdPoints
- vtkThresholdTextureCoords
- vtkTransformFilter
- vtkTransformPolyDataFilter
- vtkTransformTextureCoords
- vtkTriangleFilter
- vtkTriangularTCoords
- vtkTriangularTexture

Filters (5)

- vtkTubeFilter
- vtkVectorDot
- vtkVectorNorm
- vtkVectorTopology
- vtkVoxelModeller
- vtkWarpScalar
- vtkWarpTo
- vtkWarpVector

Mappers

Writers

- vtkIVWriter
- vtkBYUWriter
- vtkSTLWriter
- vtkMCubesWriter
- vtkPolyDataWriter
- vtkRectilinearGridWriter
- vtkStructuredGridWriter
- vtkStructuredPointsWriter
- vtkUnstructuredGridWriter
- vtkFieldDataWriter
- vtkBMPWriter
- vtkPNMWriter
- vtkTIFFWriter

Graphics Mappers

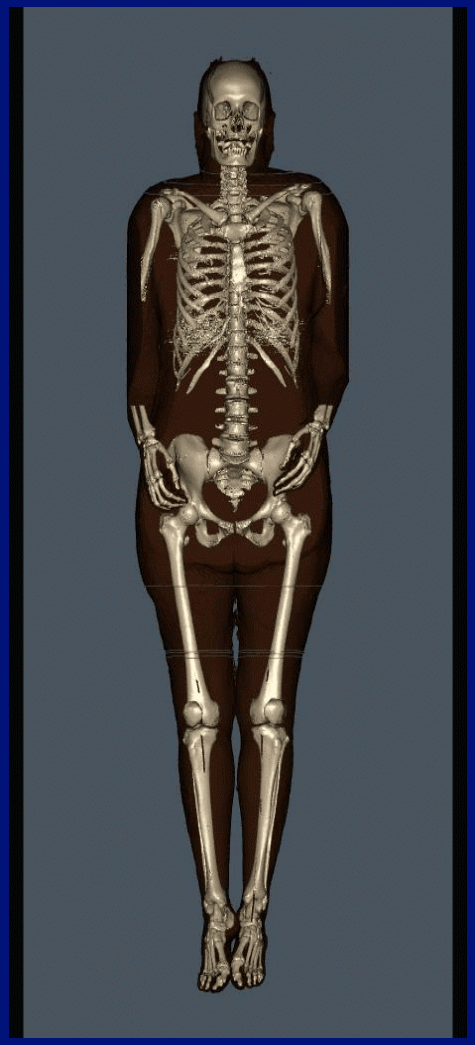
- vtkPolyDataMapper
- vtkDataSetMapper
- (volume mappers - later)
- (image mappers - later)

Data Management

Why Data Management is Critical - Size

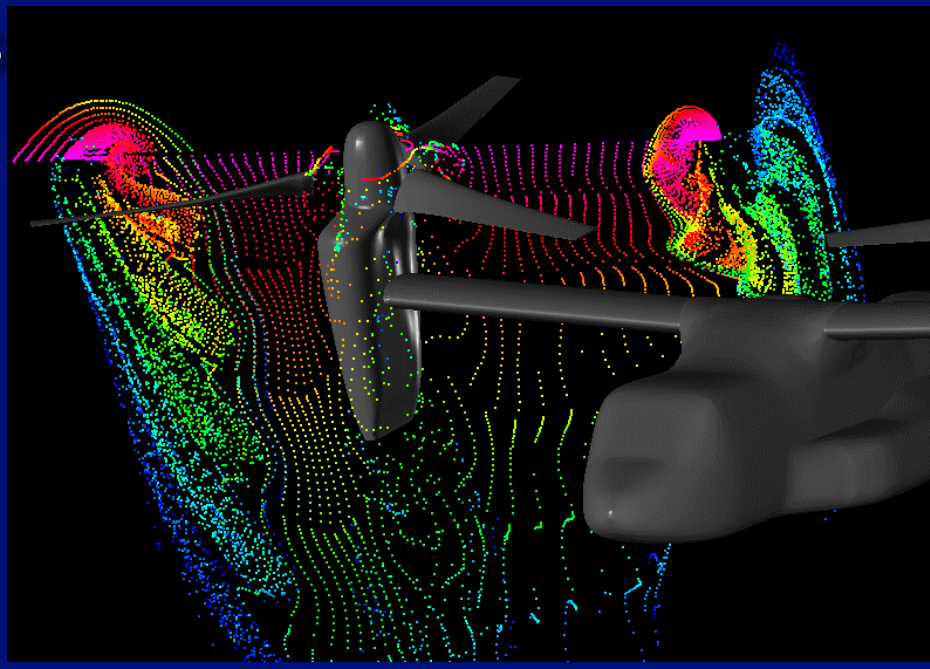
- Medical imaging (512x512x100x2 = 50Meg)
- Scientific visualization
- Video data (fluoroscope, ultrasound)
- Digital X-ray
- Satellite imagery

Data Management



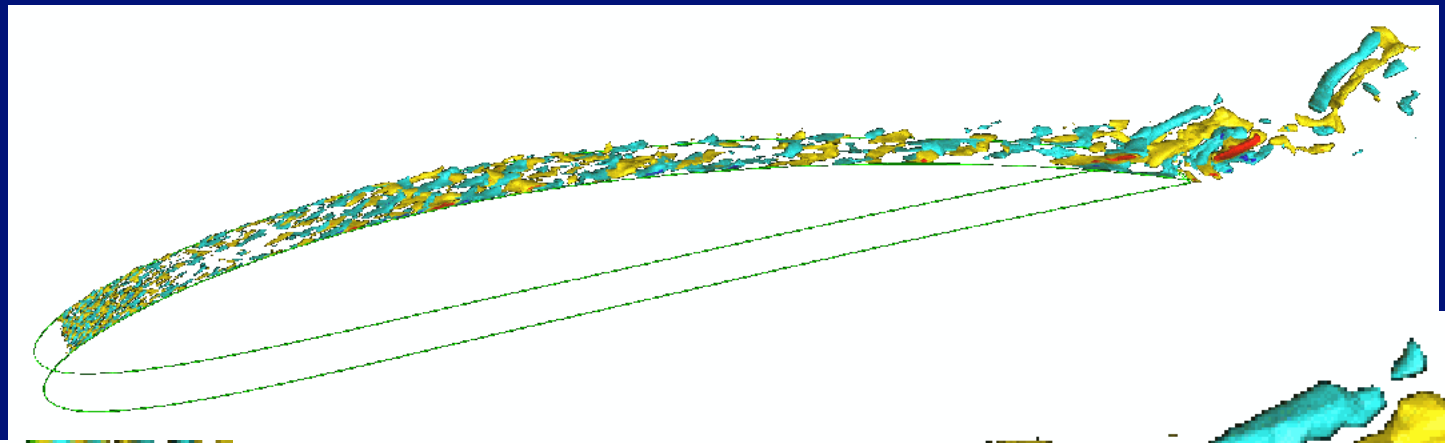
- Visible Woman CT Data
  870 MBytes 1734 Slices at 512x512x2
- Bell-Boeing V-2 2 tiltrotor
  140 Gbytes

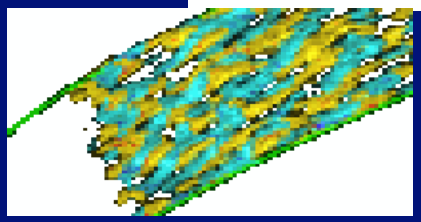
Flow Computation:
Robert Meakin
Visualization:
David Kenwright and
David Lane
Numerical
Aerodynamic Simulation
Division at NASA Ames
Research Center

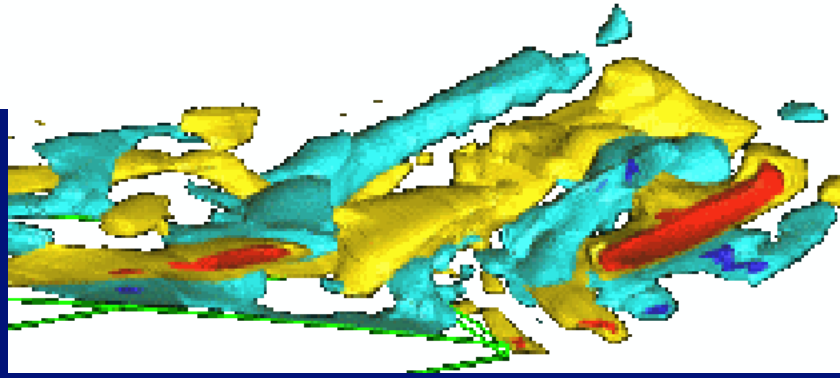


Examples

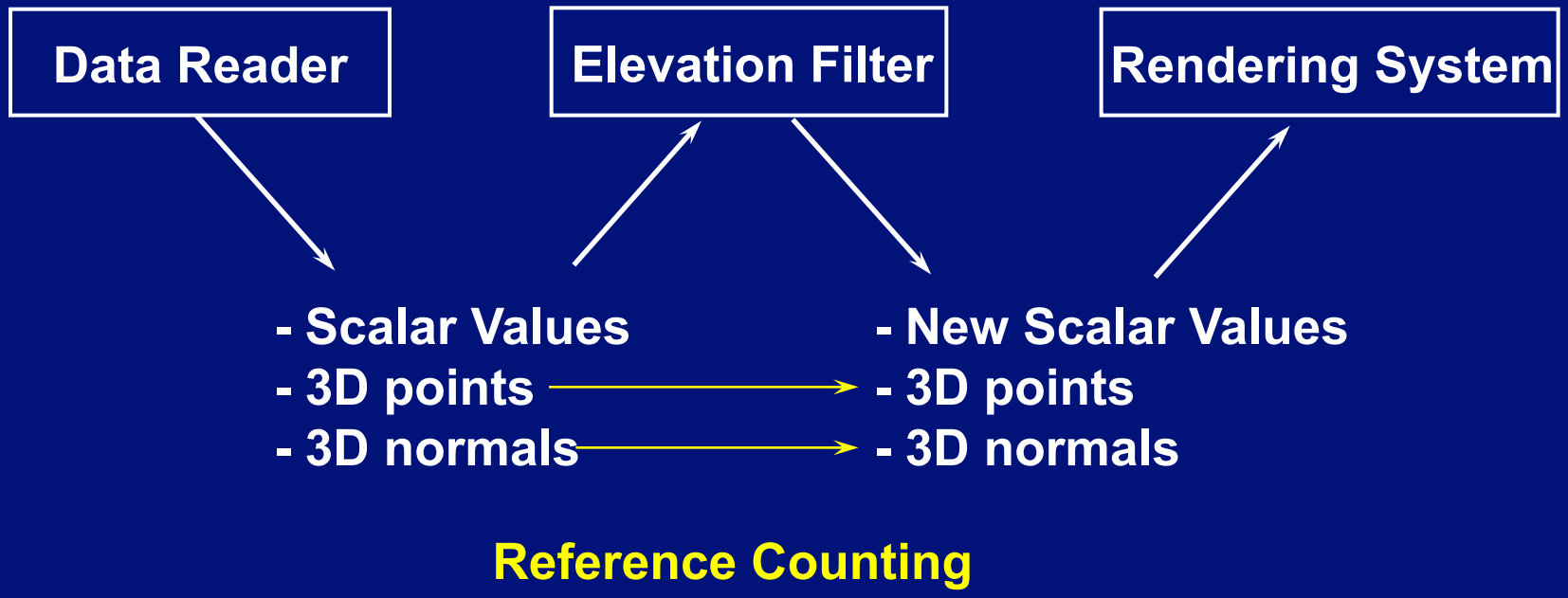
- Modeling turbulence (Ken Jansen RPI)
- 8.5 million tetrahedra, 200 time steps
- 150 million tetrahedra, 2000 time steps (soon)



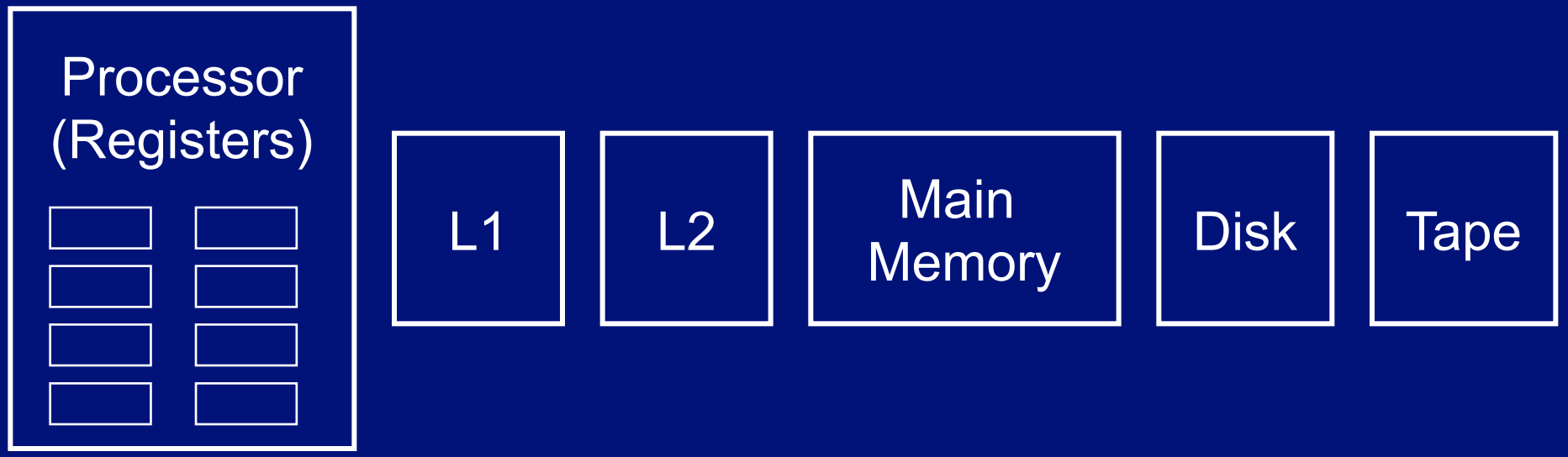




Reference Counting

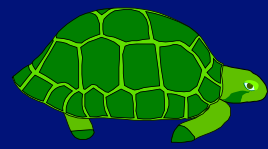


Memory Hierarchy



Access Time

0.5 Clock	1 Clock	5 Clock	10-50 Clock	10^5 Clock	10^7 Clock



Memory Hierarchy

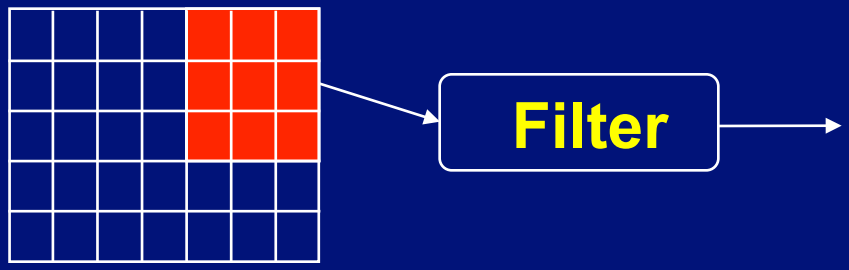
- Depending on disk and tape is to be avoided
- A lot of instructions can be carried out in the time a single disk access occurs
- Better through-put occurs when data is kept in high-levels of memory hierarchy

Enter (The) Streaming Pipeline

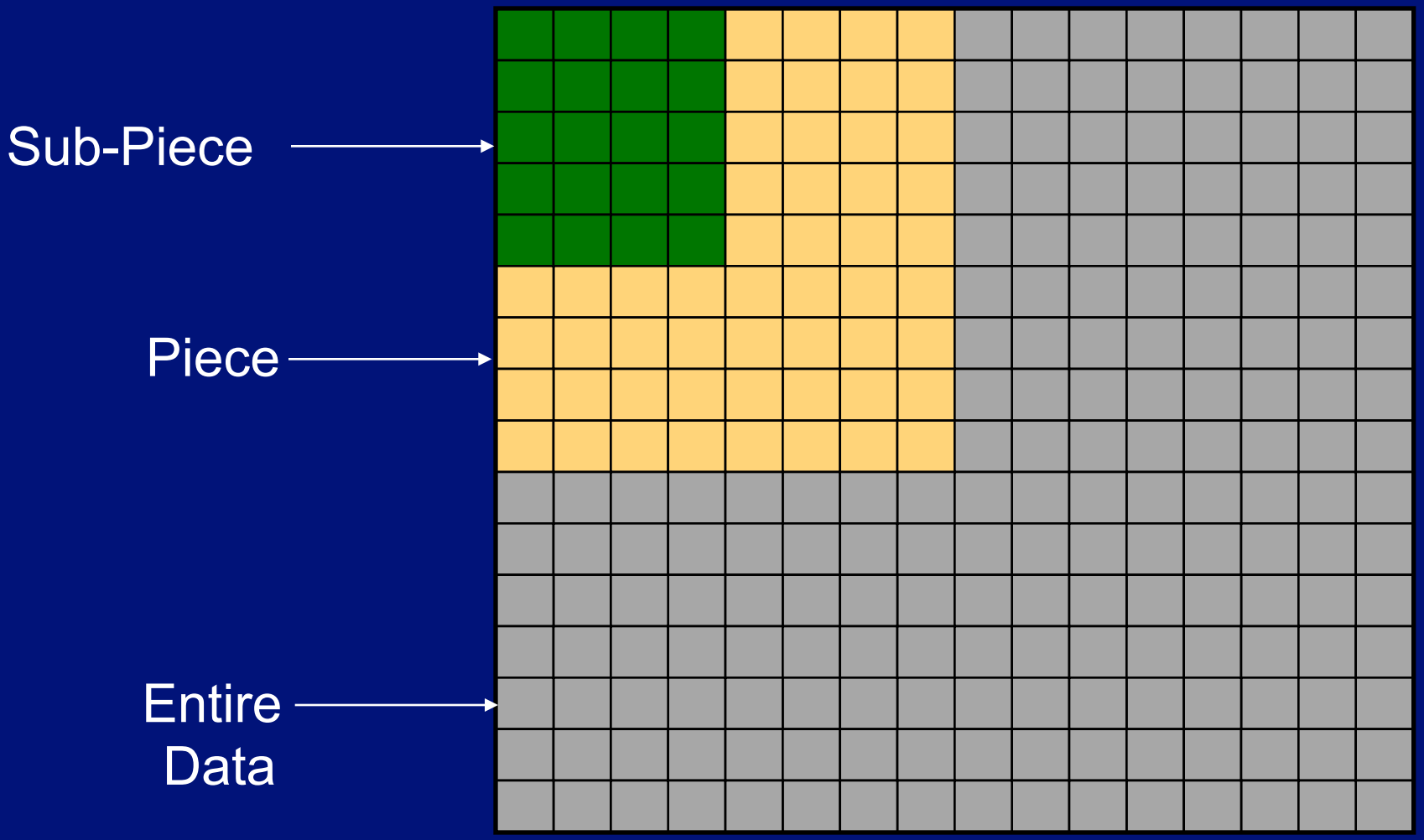
- Read data from disk/tape in pieces
- Process piece through series of filters (supports a data flow architecture)
- Only when processing is complete is data written back to disk/tape
- Control piece size to make best use of memory hierarchy (e.g., avoid swapping)

Streaming Data

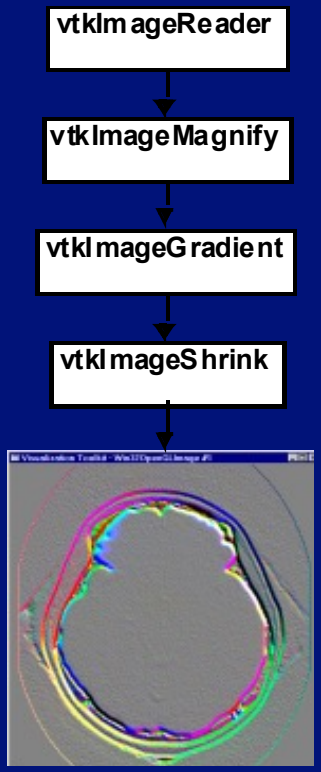
- Data is broken into pieces, and pieces processed one at a time
  - Piece size based on memory limits
  - Can avoid system swap
  - Supports parallel processing
  - Issues
    - How to create pieces
    - Mapping output from input
    - Results invariance

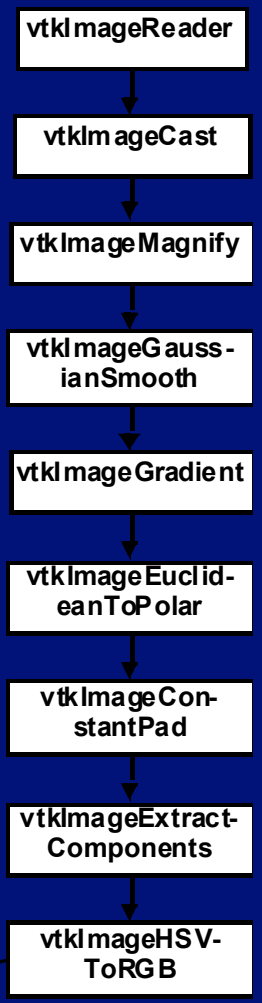


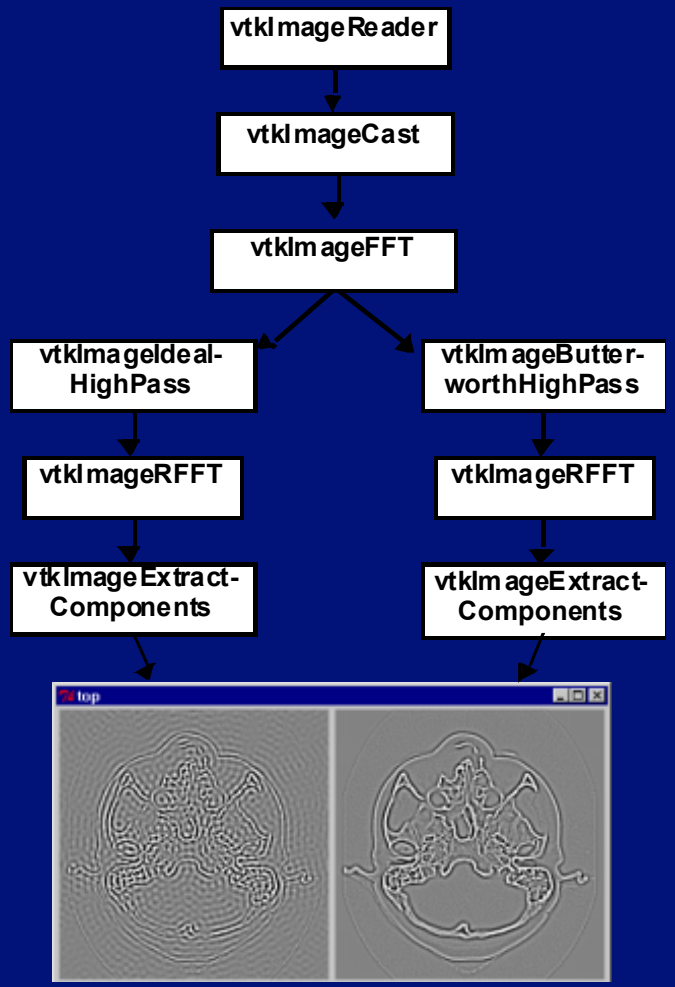
Multi-Threading



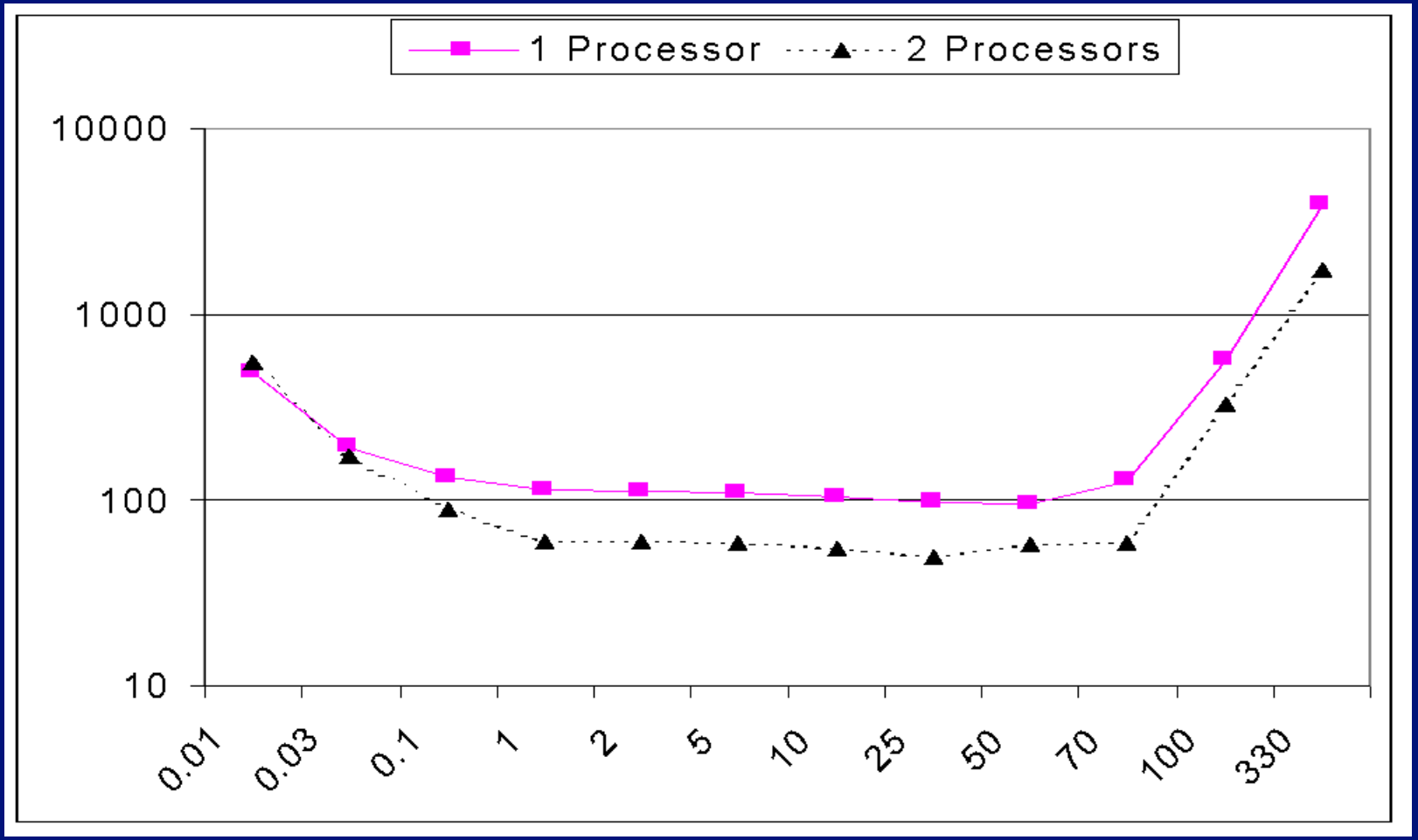
Numerical Experimants







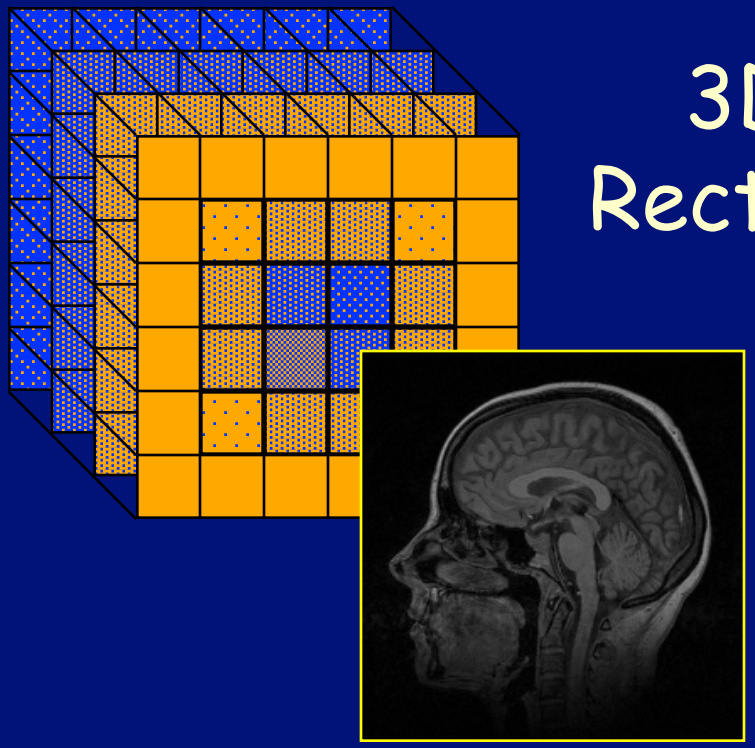
The Effect of Cache Size



Agenda

- VTK Technology
  - Background
  - The Graphics Model
  - The Visualization Model
  - Volume Rendering
- VTK Process
  - Open Source
  - Development Process

The Volume Data Structure

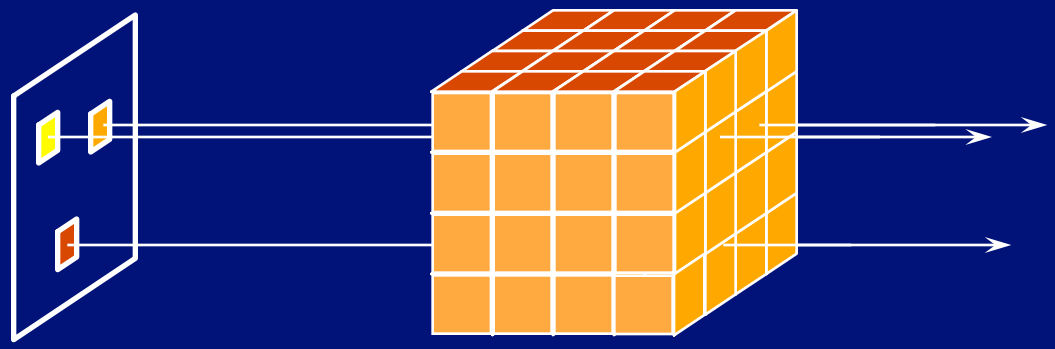


3D Regular
Rectilinear Grid

vtkStructuredPoints:
Dimensions = (D_x, D_y, D_z)
Spacing = (S_x, S_y, S_z)

Volume Rendering Strategies

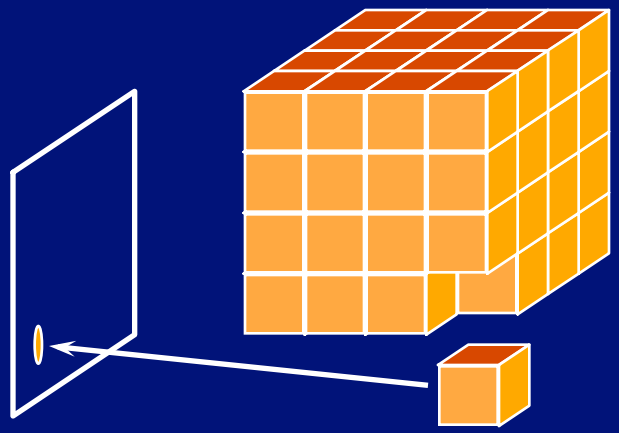
Image-Order Approach: Traverse the image pixel-by-pixel and sample the volume via ray-casting.



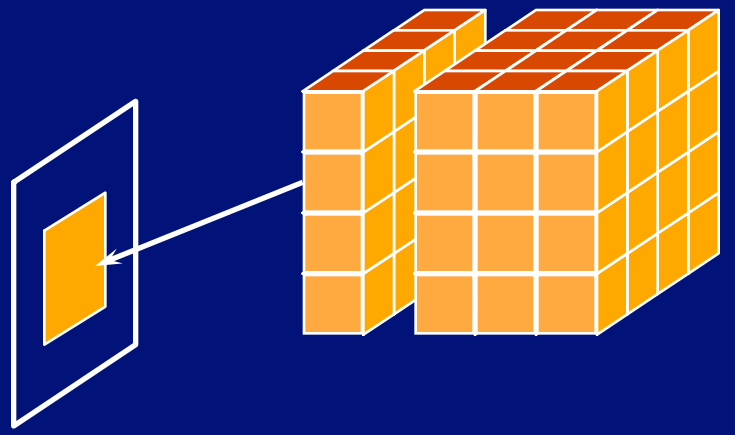
Ray Casting

Volume Rendering Strategies

Object-Order Approach: Traverse the volume, and project to the image plane.



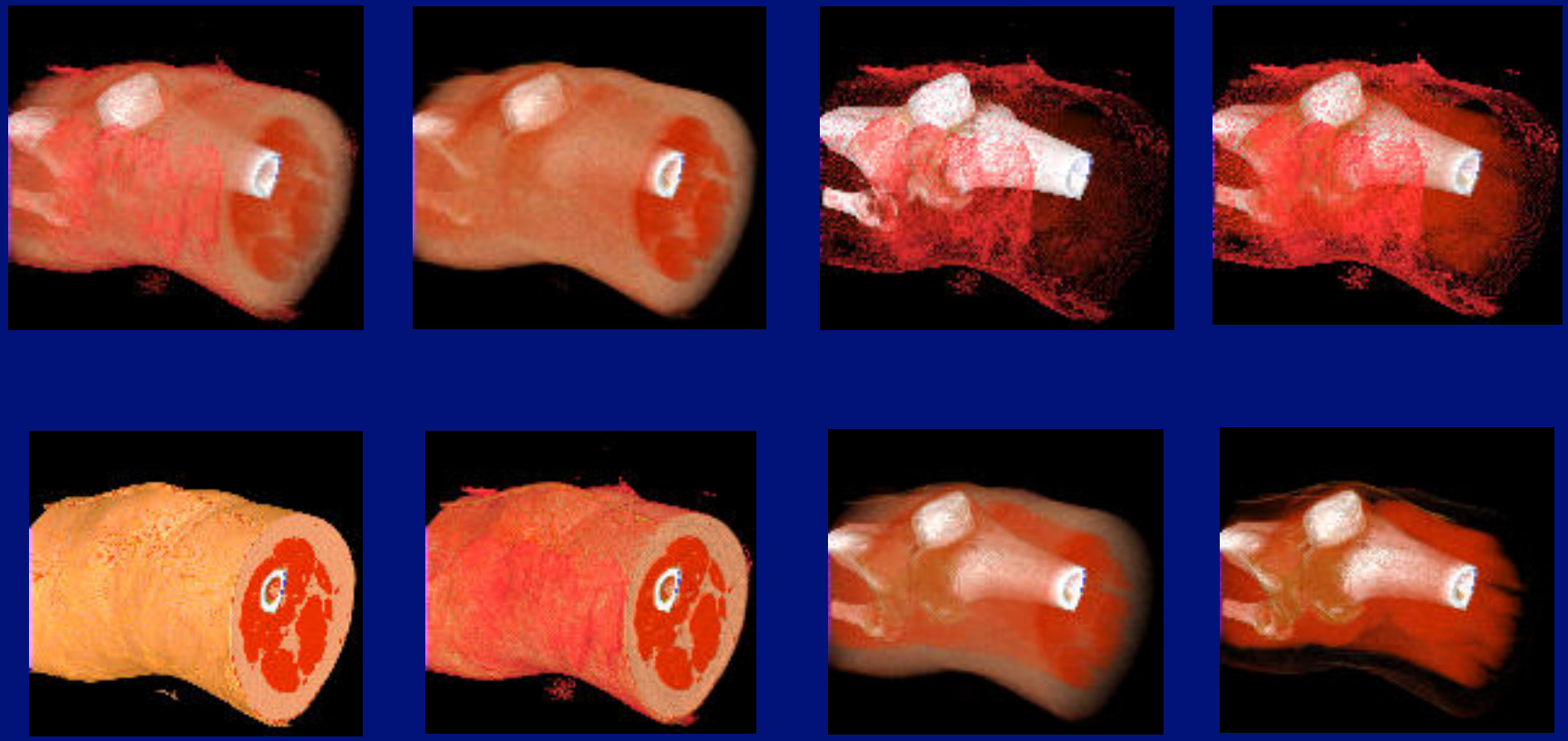
Splatting
cell-by-cell



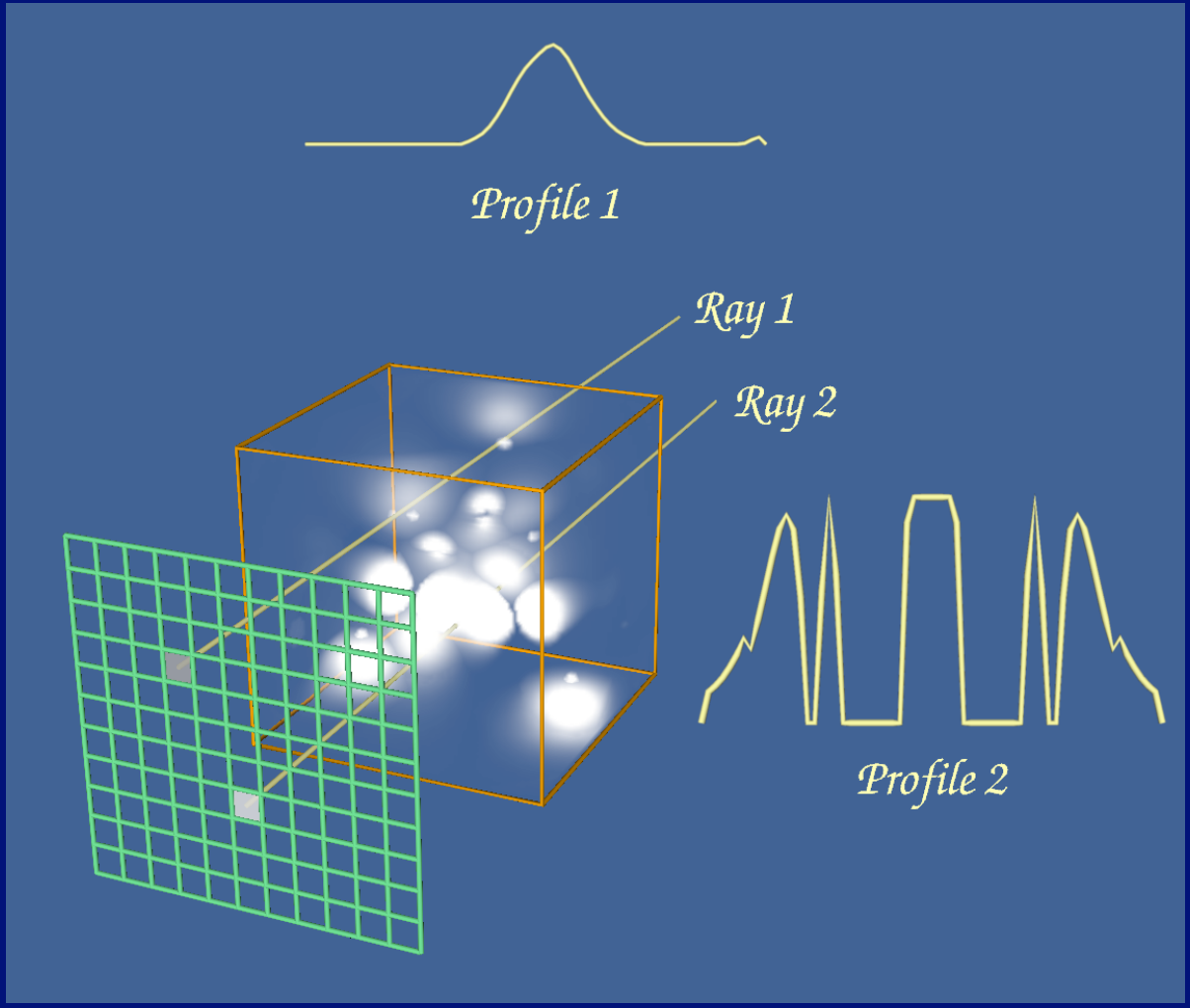
Texture Mapping
plane-by-plane

Transfer Functions

Transfer functions are the key to volume renderings

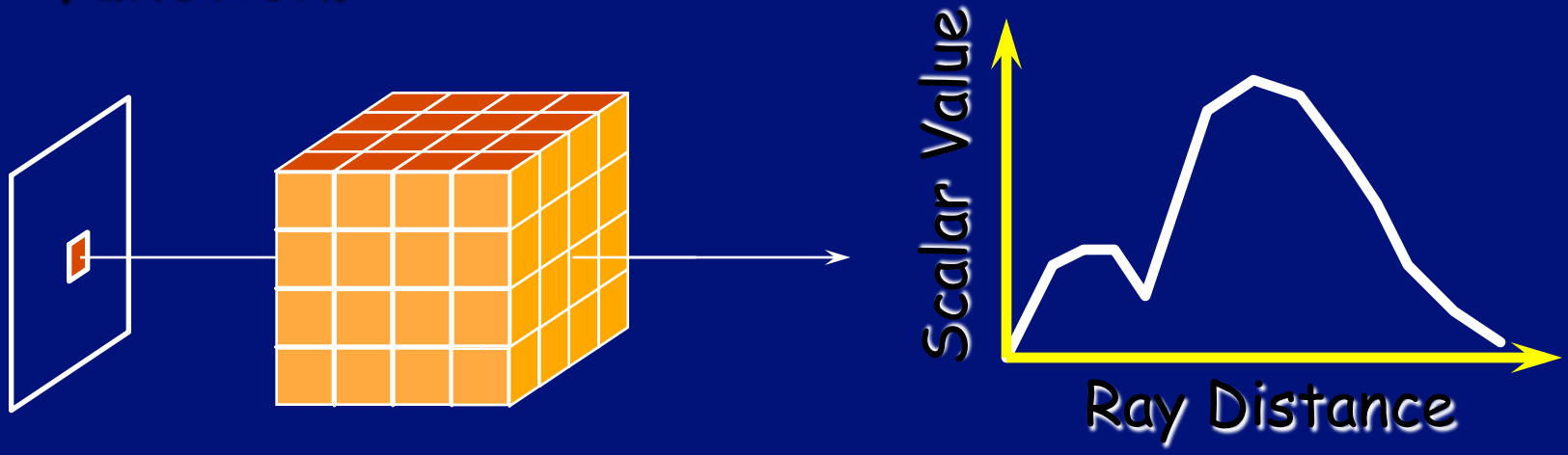


Ray Casting Process

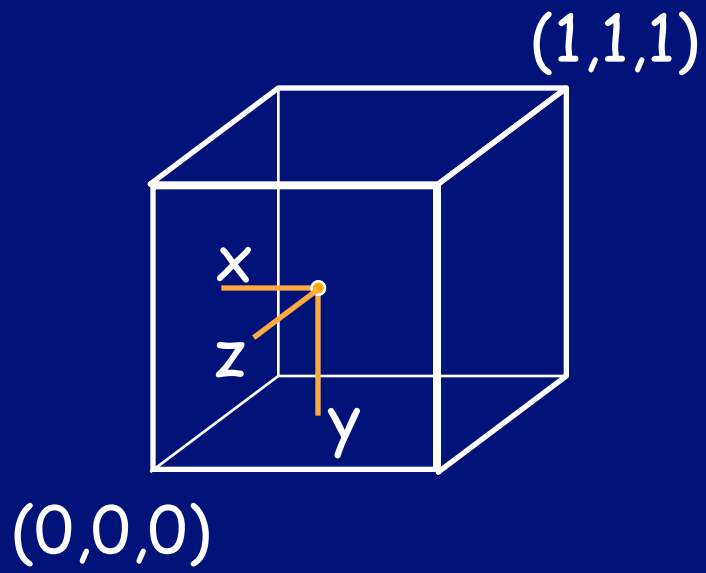


Ray Cast Functions

A Ray Function examines the scalar values encountered along a ray, and produces a final pixel value according to the volume properties, and the specific transfer function.



Scalar Value Interpolation



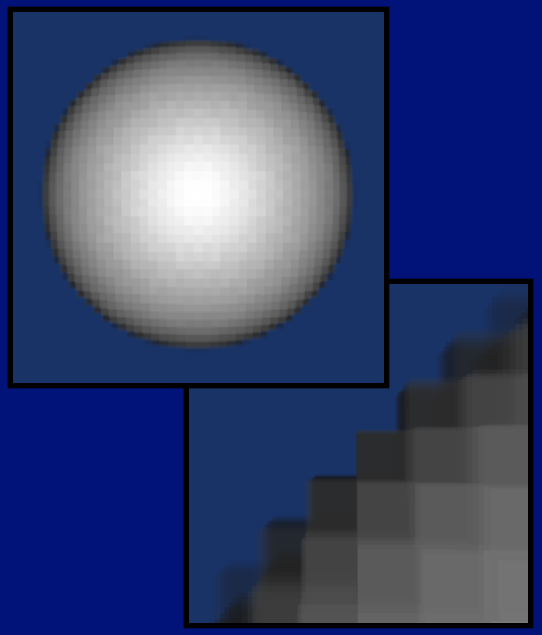
v = S(rnd(x), rnd(y), rnd(z))

Nearest Neighbor

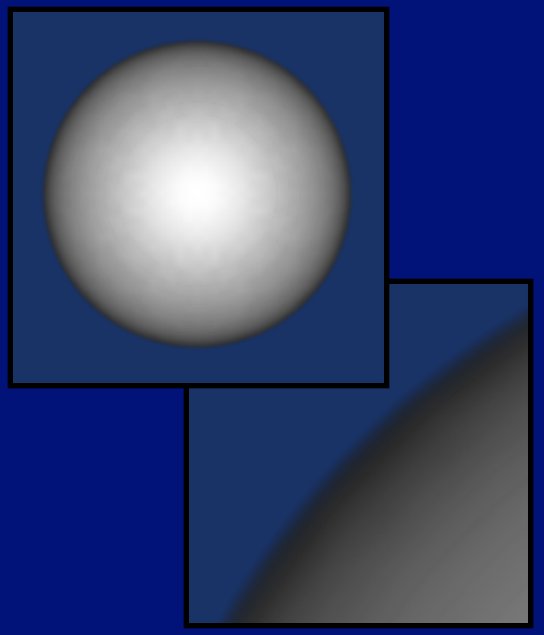
$$
\begin{aligned}
v = {} & (1-x)(1-y)(1-z)S(0,0,0) + \\
& (x)(1-y)(1-z)S(1,0,0) + \\
& (1-x)(y)(1-z)S(0,1,0) + \\
& (x)(y)(1-z)S(1,1,0) + \\
& (1-x)(1-y)(z)S(0,0,1) + \\
& (x)(1-y)(z)S(1,0,1) + \\
& (1-x)(y)(z)S(0,1,1) + \\
& (x)(y)(z)S(1,1,1)
\end{aligned}
$$

Trilinear

Scalar Value Interpolation

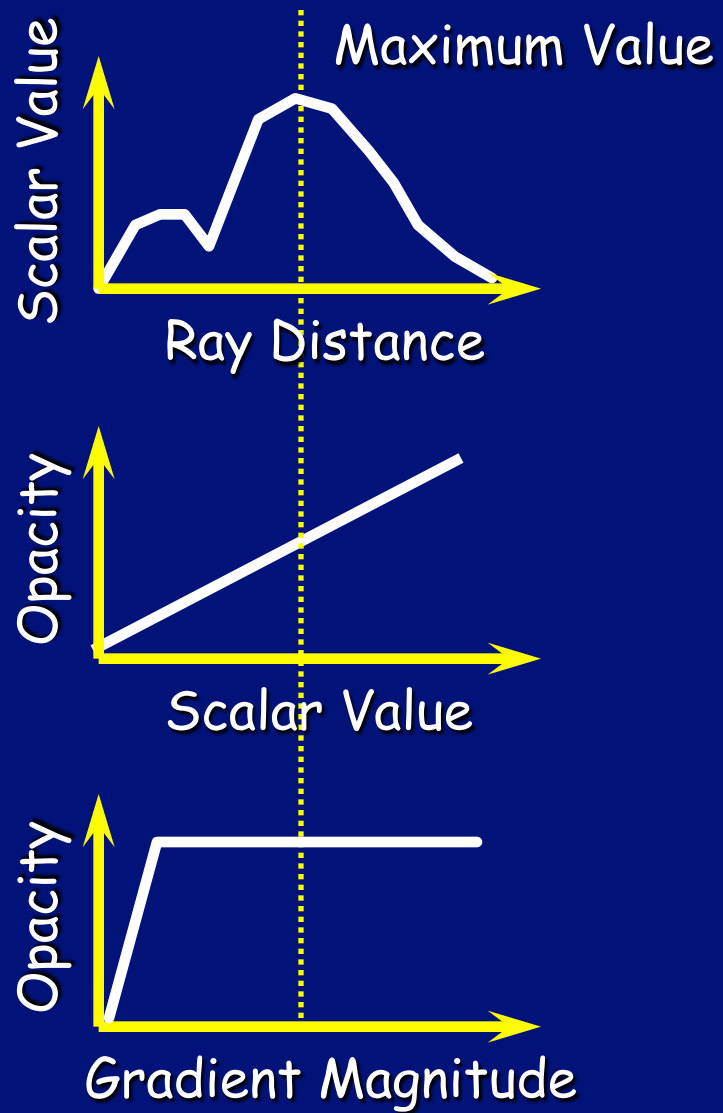


Nearest Neighbor Interpolation



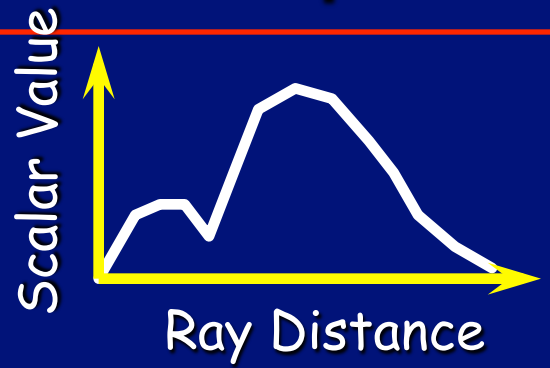
Trilinear Interpolation

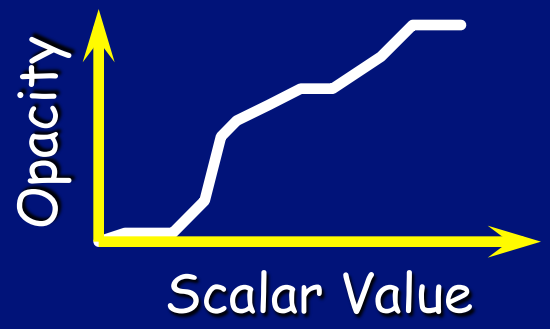
Maximum Intensity Function

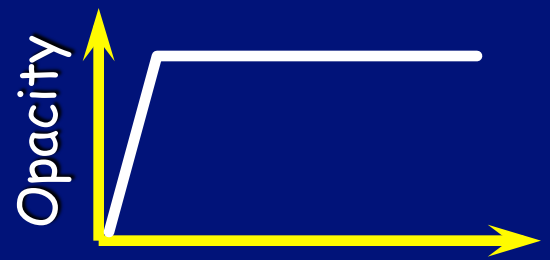


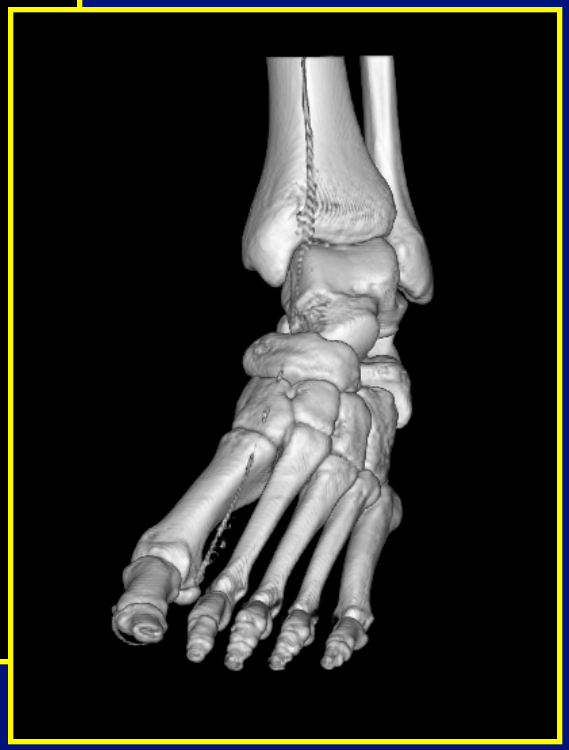
Maximize Scalar Value

Composite Function



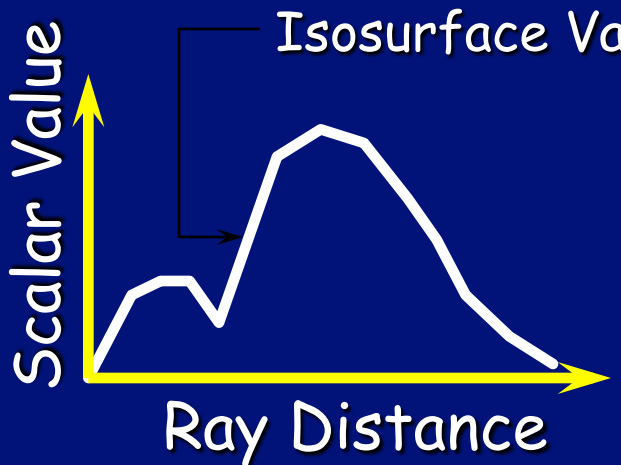






Use α-blending along the ray to produce final RGBA value for each pixel.

Isosurface Function



Stop ray traversal at isosurface value. Use cubic equation solver if interpolation is trilinear.

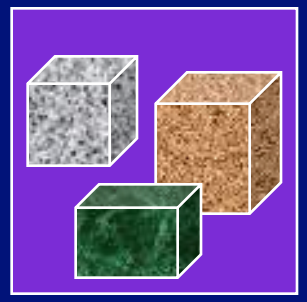
Sampling Distance

0.1 Unit Step Size

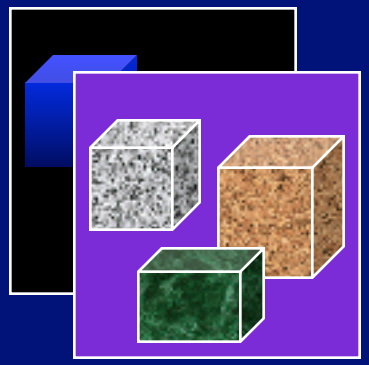
1.0 Unit Step Size

2.0 Unit Step Size

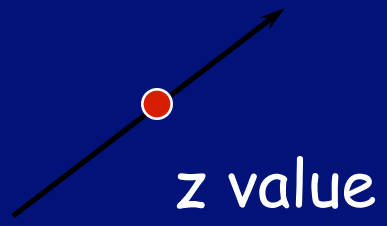
Intermixing Geometry



Render the geometry (opaque, then translucent) using the graphics hardware



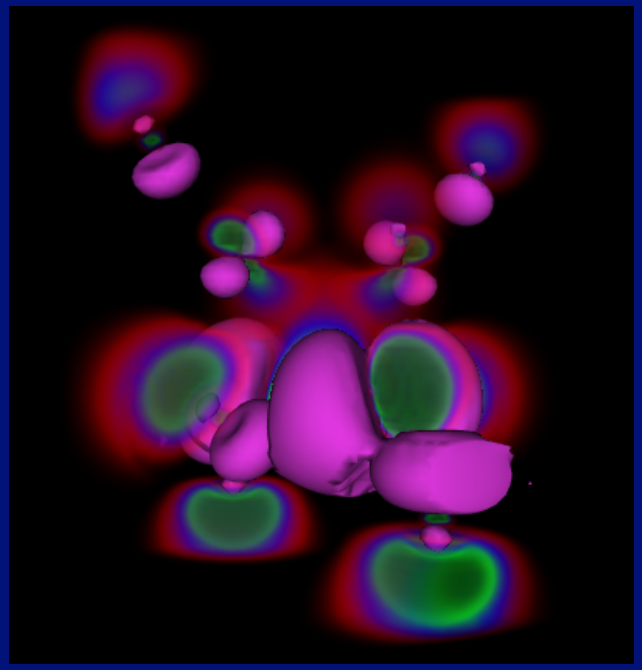
Capture the hardware color and depth buffers



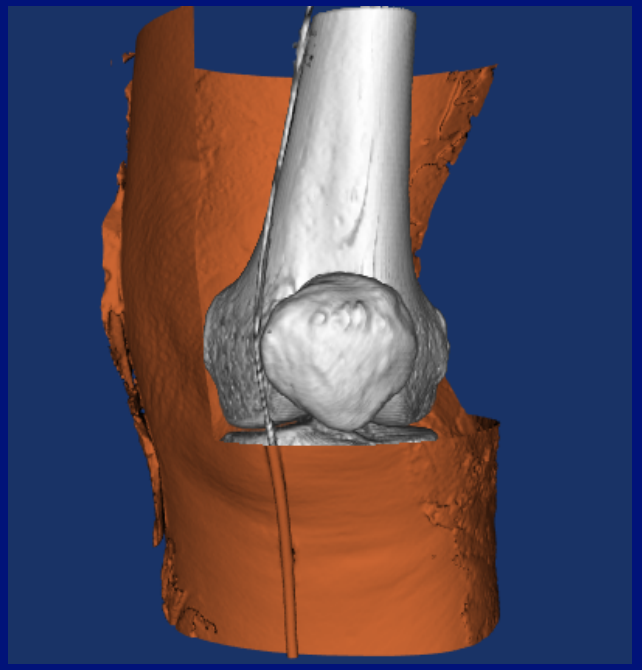
Cast the rays, stopping at the captured depth value for that pixel

Blend the ray RGBA value into the color image and draw to the hardware color buffer

Intermixed Geometry



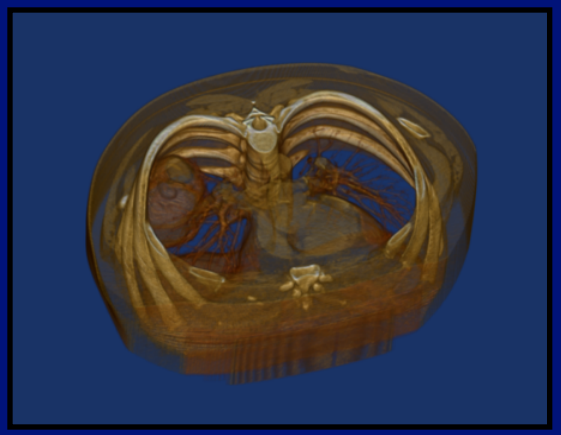
High potential iron protein



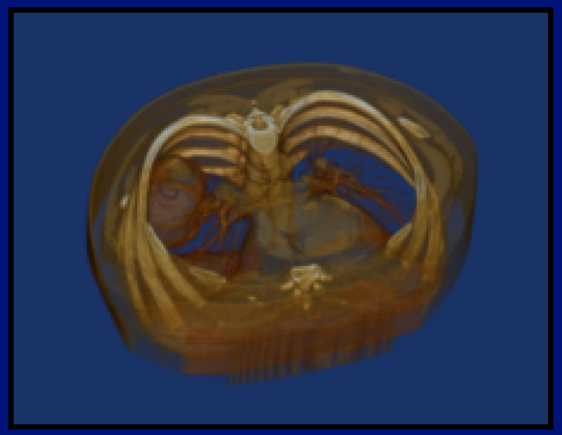
CT scan of the visible woman's knee

Speed / Accuracy Trade-Off

Multi-resolution ray casting:



1x1 Sampling



2x2 Sampling

4x4 Sampling

Combined approached: vtkLODProp3D can be used to hold mappers of various types. A volume can be represented at multiple levels-of-detail using a geometric isosurface, texture mapping, and ray casting. A decision between levels-of-detail is made based on allocated render time.

Agenda

- VTK Technology
  - Background
  - The Graphics Model
  - The Visualization Model
  - Volume Rendering
- VTK Process
  - Open Source
  - Development Process

Why Open Source?

- Fun
  - form communities
  - engage in a hobby / learning experience
- Profitable - successful business models
  - Red Hat, Inc. (Linux Support)
  - Cygnus Solutions
  - Use Value versus Sale Value (service-oriented business model)
- Altruism - serve the planet

Why Open Source?

- Kick M$'s Butt

- Intellectual freedom
  - Ideas are property
  - The territory is being claimed
  - Freedom of expression is being controlled by others
  - Don't give up your freedom!

Why Open Source?

- Scalable Software Development
  - Eric Raymond *The Cathedral & The Bazaar*

"open-source peer review is the only scalable method for achieving high reliability and quality."

- Microsoft Halloween Documents
  - *www.*opensource.org

Agenda

- VTK Technology
  - Background
  - The Graphics Model
  - The Visualization Model
  - Volume Rendering
- VTK Process
  - Open Source
  - Development Process

Development Process

- Standard C++ Style / Methodology
- Documentation embedded in code (use Doxygen to generated HTML)
- Use CVS source code control system
  - Allows simultaneous edits
  - Merges files, flags conflicts
- Automatic wrapper generator (look at SWIG)
- Daily regression testing

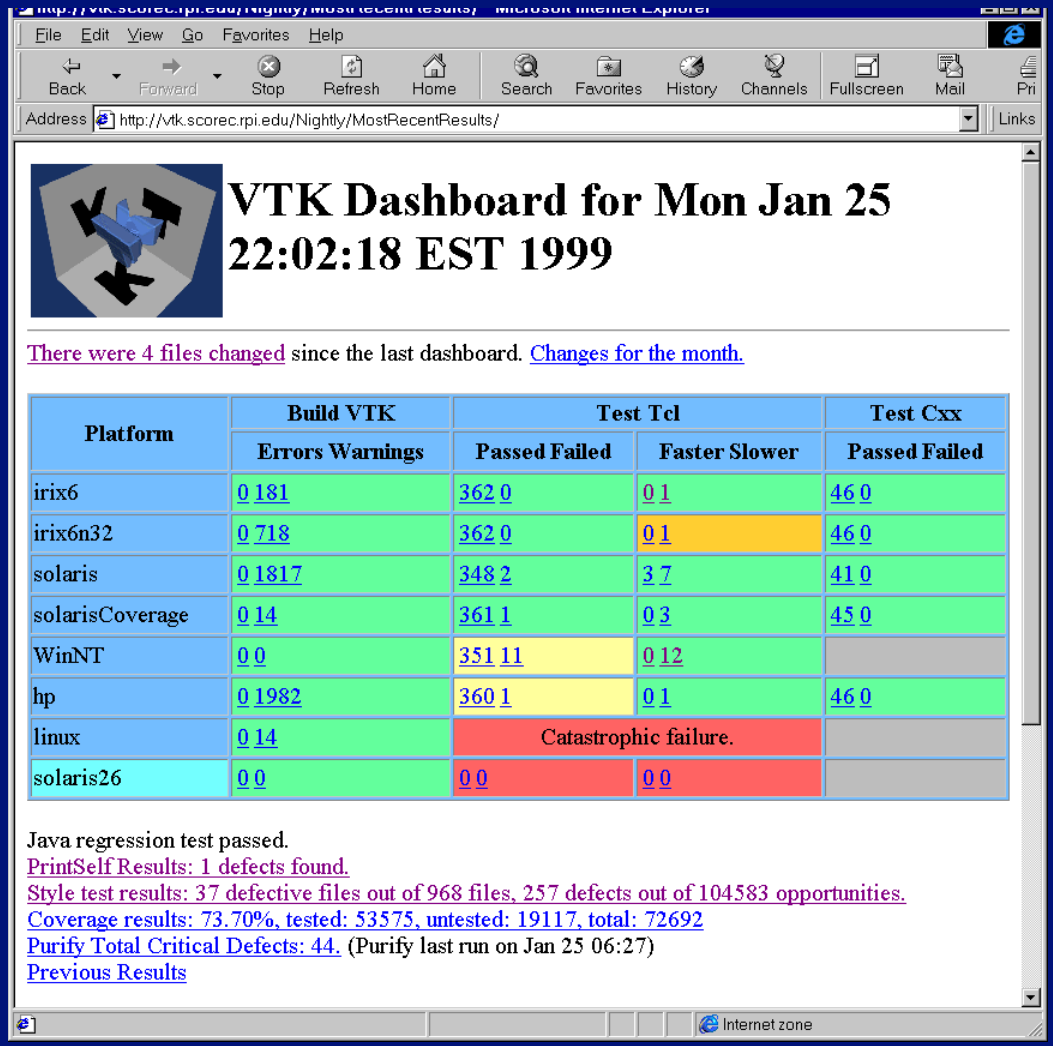
Testing Process (Nightly)

- Regression test ~500 examples
- Check style
- Check PrintSelf()
- Check memory problems (Purify)
  - Read/write beyond memory bounds
  - Memory leaks
- Check Coverage
  - Running around 75%
  - If it ain't covered, it don't work

Regression Testing

- Compare generated image against standard "correct" image
  - pixel-by-pixel comparison
  - can use a threshold metric
  - adjusted for effects like dithering
  - OpenGL is rather loose about image quality

VTK Quality Dashboard



File Edit View Go Favorites Help

Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Pri

Address http://vtk.scorec.rpi.edu/Nightly/MostRecentResults/ Links

VTK Dashboard for Mon Jan 25 22:02:18 EST 1999

There were 4 files changed since the last dashboard. Changes for the month.

Platform	Build VTK	Test Tcl		Test Cxx
	Errors Warnings	Passed Failed	Faster Slower	Passed Failed
irix6	0 181	362 0	0 1	46 0
irix6n32	0 718	362 0	0 1	46 0
solaris	0 1817	348 2	3 7	41 0
solarisCoverage	0 14	361 1	0 3	45 0
WinNT	0 0	351 11	0 12	
hp	0 1982	360 1	0 1	46 0
linux	0 14	Catastrophic failure.		
solaris26	0 0	0 0	0 0	

Java regression test passed.
PrintSelf Results: 1 defects found.
Style test results: 37 defective files out of 968 files, 257 defects out of 104583 opportunities.
Coverage results: 73.70%, tested: 53575, untested: 19117, total: 72692
Purify Total Critical Defects: 44. (Purify last run on Jan 25 06:27)
Previous Results

Internet zone

Why Test Daily?

- Large code base too large for any single developer to understand
- Developers distributed around the world
- Identify problems as they occur
- Insure that object API remains unchanged
- Provide feedback to developers as they experiment with new implementations

Resources

- VTK
  - www.kitware.com/vtk.html
  - vtkusers mailing list
  - other resources (Sebastien Barre's web pages)
- Eric Raymond *The Cathedral and the Bazaar*
- CVS Documentation
- www.opensource.org