Revised Schedule for Part 1

1. Introduce VTK  
2. Introduce Tcl/Tk  
3. Simple Visualization Tasks  
4. Surface and Image Manipulation  
5. Image Manipulation and Display  
6. Local Extensions  
7. Tk and GUIs  
8. [Incr] Tcl and other extensions

Revision -- VTK Pipeline (I)

Data Representation (vtkDataSet)

Images are Simpler

vtkImageData

- vtkImageData is the basic VTK class for storing images.
- It is defined by 4 key elements
  - Dimensions -- these define the size of the image
  - Origin -- the position in 3D space of point 0 0 0
  - Spacing -- the voxel dimensions
  - Scalar Type -- the type of the image (e.g. float, short etc)
- An 4x4x4 image has 4x4x4=64 points and 3x3x3=27 cubic cells (both are implicitly defined)
Visualizing Images

- Images are displayed as textures mapped on polygons, uses special feature (texture memory) on most decent (>$75) graphics boards.
- In OpenGL all textures must have dimensions that are powers of two.
- Images are interpolated before display, hence some (small) loss of sharpness takes place (only visible in small images).
  - E.g. an 100x50 image will be resampled to 128x64 before display.
- Similar issues for color display i.e. scalars vs lookup tables as in surfaces.
- We will examine image display next week.

An Aside: Imaging Conventions

- Images are 3D Functions with proper X,Y,Z axes.
- Caveats:
  - Radiologists are used to looking at images in certain orientations regardless of underlying axes.
  - Images can be acquired in one of three (excluding obliques) orientations: Axial, Coronal and Sagittal, the X,Y,Z axes depend on the acquisition protocol (All non-MR are usually acquired using axial orientation).
  - Images are typically left/right flipped (radiology convention – looking from the feet up!)
  - Neurosurgery applications require proper left/right orientation.
  - Brain is mostly left/right symmetric, really easy to make very costly mistakes.

Image Acquisition Orientations

Axial(0) – Bottom to Top
Coronal(1) – Front to Back
Sagittal(2) – Left to Right

Imaging Slices

Coronal
Sagittal
Axial

The slices should look the same regardless of the original acquisition orientation!

Display Problems

- If the images are displayed in 3D space as they are acquired they look wrong!
- The contrast of some images can be bad!
- If the images are reformatted to look right:
  - we loose track of the original data locations
  - often this is not possible for all three types of planes
  - images are often acquired using the wrong slice order.
- (My adopted) Solution
  - Ensure slice order adheres to predefined protocol, if not flip the slice order (this is the only modification to the original data).
  - Preserve original data and perform ‘camera’ acrobatics in 3D to make them look right.
  - Use lookup tables to adjust the contrast, but never modify the original intensities.

A 3D View (Axial Acquisition)
Image Data

This is a dummy slide to make available links for downloading the images used in the examples of this presentation.

- 2D Tiff Image
  - examples\brain.tif
- 3D low resolution brain image in VTK internal format
  - examples\brain.vt

Adding a Smoothing Filter

```tcl
vtkTIFFReader tr
tr SetFileName brain.tif
tr Update
vtkImageGaussianSmooth smooth
smooth SetInput [ tr GetOutput ]
smooth SetStandardDeviations 3 3 3
vtkImageViewer vw
vw SetInput [ tr GetOutput ]
vw SetZSlice 0
vw SetColorLevel 128
vw SetColorWindow 255
vw Render
wm withdraw .
vwait forever
```
Adding an Image Flip Filter
examples/example5_3.tcl

```tcl
vtkTIFFReader tr; tr SetFileName brain.tif
vtkImageGaussianSmooth smooth
smooth SetInput [ tr GetOutput ]
smooth SetStandardDeviations 3 3 3
vtkImageFlip flip
flip SetInput [ smooth GetOutput ]
flip SetFilteredAxis 1
vtkImageViewer vw
vw SetInput [ tr GetOutput ]
vw SetZSlice 0;
vw SetColorLevel 128; vw SetColorWindow 255
vw Render

wm withdraw .; vwait forever
```

Displaying Image Animation
examples/example5_4.tcl

```tcl
vtkStructuredPointsReader tr; tr SetFileName brain.vt
vtkImageMagnify magn; magn SetMagnificationFactors 4 4 1
magn SetInput [ tr GetOutput ]
vtkImageGaussianSmooth sm; sm SetInput [ magn GetOutput ]
sm SetStandardDeviations 3 3 3
vtkImageFlip flip; flip SetInput [ sm GetOutput ]
flip SetFilteredAxis 1
vtkImageViewer vw; vw SetInput [ flip GetOutput ]
vw SetColorLevel 240; vw SetColorWindow 256; vw Render

set dim [ [ tr GetOutput ] GetDimensions ]
set numslices [ lindex $dim 2 ]
for { set i 0 } { $i < $numslices } { incr i } {
puts stdout "Displaying slice $i/$numslices"
after 100;
vw SetZSlice $i;
vw Render
}
ex
```

A Seven Line File Converter
examples/example5_5.tcl

```tcl
Convert TIFF to JPEG
vtkTIFFReader tr
tr SetFileName brain.tif
vtkJPEGWriter jp
jp SetInput [ tr GetOutput ]
jp SetFileName brain.jpg
jp Write
exit
```

Texture Mapping – the Pipeline

Texture Mapping I – Displaying the Plane
examples/example5_6.tcl

```tcl
vtkPlaneSource imageplane
imageplane SetXResolution 1
imageplane SetYResolution 1
imageplane SetOrigin -0.5 -0.5 30
imageplane SetPoint1 40.5 -0.5 30
imageplane SetPoint2 -0.5 46.5 30
vtkPolyDataMapper map
map SetInput [ imageplane GetOutput ]
vtkActor imactor
imactor SetMapper map
vtkRenderer ren
ren AddActor imactor
```

Texture Mapping II – Displaying the Plane
examples/example5_6.tcl

```tcl
vtkRenderWindow renwin
renwin AddRenderer ren
renwin SetSize 300 300
ren ResetCamera
renwin Render

# Interactor
vtkRenderWindowInteractor iren
iren SetRenderWindow renwin
iren Initialize
iren AddObserver SetExitMethod { exit }
wm withdraw .; vwait forever
```
Texture Mapping III – Texturing the Plane
examples\example5_7.tcl

```tcl
vtkStructuredPointsReader tr
tr SetFileName examples/brain.vt
tr Update

tvtkExtractVOI voi
voi SetInput [ tr GetOutput ]
voi SetVOI 0 41 0 47 30 30

tvtkTexture texture
texture SetInput [ voi GetOutput ]
texture InterpolateOn

imactor SetTexture texture
```

Texture Mapping IV – Adding a Colormap
examples\example5_8.tcl

```tcl
vtkLookupTable colormap
colormap SetNumberOfColors 256
colormap SetTableRange 0 255

for { set i 0 } { $i < 256 } { incr i } {
    set v [ expr $i /255.0 ]
    colormap SetTableValue $i $v $v $v 1.0
}

vtkTexture texture
texture SetInput [ voi GetOutput ]
texture SetLookupTable colormap
```

Texture Mapping V – An Axial Slice
examples\example5_9.tcl

```
Replace (Note Coronal = XY, Axial = XZ)
imageplane SetOrigin -0.5 -0.5 30
imageplane SetPoint1 40.5 -0.5 30
imageplane SetPoint2 -0.5 46.5 30

with
imageplane SetYResolution 1
imageplane SetOrigin -0.5 24 -0.5
imageplane SetPoint1 -0.5 24 52.5
imageplane SetPoint2 40.5 24 -0.5

and
voi SetVOI 0 41 0 47 30 30
with
voi SetVOI 0 41 24 24 0 53
```

Texture Mapping IV – Mapping to a Sphere
examples\example5_10.tcl

```
• Texture Mapping is a general technique for mapping textures(images) to polygonal data
• Non-planar images require proper texture coordinates to be generated
• See example for mapping an image to a sphere!
```

Volume Rendering

```
• Volume Rendering is a method for direct rendering of images without mapping to an underlying geometrical object.
• Lots of different methods
  – key graphics element is transparency, often lower intensities are made more transparent to reveal higher intensities behind them in space
• We will look at
  1. Ray-casting, generating 2D images from 3D using an X-ray like technique
  2. Volume Texture Mapping
```

Ray Casting

```
Set of Rays (need not be parallel)
3D Object
2D Image formed by e.g.
Integral of F(intensity) x G(transparency) along each ray

Major Difficulty: Selection of Intensity and Transparency Transfer Functions F,G, this is highly data dependent.
```
Ray Mapping – the Pipeline

- vtkVolumeRayCastFunction (generates line integration procedure)
- Mapper: vtkVolumeRayCastMapper
- vtkVolume

Image Source
- Ensure Unsigned Char Or Short Texture
  vtkImageCast or vtkImageShiftScale

Generate opacity transfer function using vtkPiecewiseFunction
Generate intensity transfer function using vtkPiecewiseFunction

Ray Casting -- example

```tcl
# Create the reader for the data
vtkStructuredPointsReader reader
reader SetFileName examples/brain.vt
vtkImageCast cast
cast SetInput [ reader GetOutput ]
cast SetOutputScalarTypeToUnsignedChar

# Create transfer mapping scalar value to opacity
vtkPiecewiseFunction opacityTransferFunction
opacityTransferFunction AddPoint 40 0.0
opacityTransferFunction AddPoint 255 1.0

# Create transfer mapping scalar value to color
vtkColorTransferFunction colorTransferFunction
colorTransferFunction AddRGBPoint 0.0 0.0 0.0 0.0
colorTransferFunction AddRGBPoint 255.0 1.0 1.0 1.0
```

Volume Rendering via Texture Mapping

```tcl
# The property describes how the data will look
vtkVolumeProperty volumeProperty
volumeProperty SetColor colorTransferFunction
volumeProperty SetScalarOpacity opacityTransferFunction
volumeProperty SetInterpolationTypeToLinear

# The mapper / ray cast function know how to render the data
vtkVolumeRayCastCompositeFunction compositeFunction
vtkVolumeRayCastMapper volumeMapper
volumeMapper SetVolumeRayCastFunction compositeFunction
volumeMapper SetInput [ cast GetOutput ]

# The volume holds the mapper and the property and
# can be used to position/orient the volume
vtkVolume volume
volume SetMapper volumeMapper
volume SetProperty volumeProperty
vtkRenderer ren1; ren1 AddVolume volume

# Create window/interactor etc as before
```

Ray Casting – example (cnt)
examples\example5_11.tcl

```tcl
# The property describes how the data will look
vtkVolumeProperty volumeProperty
volumeProperty SetColor colorTransferFunction
volumeProperty SetScalarOpacity opacityTransferFunction
volumeProperty SetInterpolationTypeToLinear

# The mapper / ray cast function know how to render the data
vtkVolumeRayCastCompositeFunction compositeFunction
vtkVolumeRayCastMapper volumeMapper
volumeMapper SetVolumeRayCastFunction compositeFunction
volumeMapper SetInput [ cast GetOutput ]

# The volume holds the mapper and the property and
# can be used to position/orient the volume
vtkVolume volume
volume SetMapper volumeMapper
volume SetProperty volumeProperty
vtkRenderer ren1; ren1 AddVolume volume

# Create window/interactor etc as before
```

Volume Rendering via Texture Mapping
examples\example5_12.tcl

Replace
```
vtkVolumeRayCastCompositeFunction compositeFunction
vtkVolumeRayCastMapper volumeMapper
```
with
```
vtkVolumeTextureMapper2D volumeMapper
```

In Volume Texture Mapping the Composite Function is implicit and
determined by the hardware, in general raycasting is more flexible but
texture mapping on modern graphics cards is much much faster!