VTK Volume Visualization Example

VTK Approaches to Volume Rendering

- vtkImage data only
  - Ray Casting
  - 2D Texture Mapping
  - VolumePro Rendering Board

Read & Produce vtkImage Data as Output

- vtkStructuredPointsReader
  - Reads VTK formatted files
    - Same process as previously used when vtkImage data was read and used for geometric rendering applications
- vtkBMPReader
- vtkJPEGReader
- vtkVolumeReader
- Many other readers provided by VTK that produce vtkImage data as output...

Example: Volume Reader

```tcl
# Read a series of 2D slices (images) that compose the volume.
# The slice dimensions are set, and the pixel spacing.
# Endianness must also be specified.
# The reader uses the FilePrefix in combination with the slice number
# to construct filenames using the format FilePrefix.%d.
# In this example the FilePrefix is the root name of the file: quarter.

vtkVolume16Reader v16
v16 SetDataDimensions 64 64
v16 SetDataByteOrderToLittleEndian
v16 SetFilePrefix "$VTK_DATA_ROOT/Data/headsq/quarter"
v16 SetImageRange 1 93
v16 SetDataSpacing 3.2 3.2 1.5
```

Ray Casting Example

```tcl
# Create the reader for the data
vtkStructuredPointsReader reader
reader SetFileName "$VTK_DATA_ROOT/Data/ironProt.vtk"
```
Define Transfer Functions

- Transfer function definition is a challenge
  - Define transfer functions for opacity, color, and sometimes the magnitude of the scalar gradient
  - Must have an understanding of the meaning of the underlying data values

```python
# Define transfer function mapping scalar value to opacity
vtkPiecewiseFunction opacityTransferFunction
opacityTransferFunction AddPoint 20 0.0
opacityTransferFunction AddPoint 255 0.2
```

Clamping is on by default; if off, then out of range values map to 0

- Can add and remove points via methods
- Can define <point, value> segments
- Uses linear interpolation

# Define transfer function mapping scalar value to color
vtkColorTransferFunction colorTransferFunction
```
colorTransferFunction AddRGBPoint 0.0 0.0 0.0 0.0
colorTransferFunction AddRGBPoint 64.0 1.0 0.0 0.0
colorTransferFunction AddRGBPoint 128.0 0.0 0.0 1.0
colorTransferFunction AddRGBPoint 192.0 0.0 1.0 0.0
colorTransferFunction AddRGBPoint 255.0 0.0 0.2 0.0
```

Similar principle to vtkPiecewiseFunction
- Can specify either RGB or HSV values

Control Shading

- Similar to controlling Actor properties
  - Ambient coefficient
  - Diffuse coefficient
  - Specular coefficient
  - Specular power coefficient

- In general, how these parameters are used depends on how the illumination equations are implemented by the specific render

Lighting Parameters

- If ambient term dominates, the volume will appear unshaded
- If diffuse term dominates, the volume will appear rough (e.g., concrete)
- If specular term dominates, the volume will appear smooth (e.g., glass)
- Note that lighting effects are often turned off in volume visualization applications since many traditionalists feel that lighting impacts the interpretation of the image

```python
# The property describes how the data will look based on transfer function and other properties.
vtkVolumeProperty volumeProperty
```

Shade off results in:
- ambient coefficient = 1, diffuse = 0, specular = 0
- Tradeoff in speed versus quality
Volumetric Ray Casting

- vtkVolumeRayCastMapper employs a software ray casting technique (function) to perform volume rendering
  - compositing function along the ray
  - MIP function
  - IsoSurface function

# This instance will perform the compositing of samples along the ray
vtkVolumeRayCastCompositeFunction rayFunction

vtkVolumeRayCastMapper volumeMapper
volumeMapper SetVolumeRayCastFunction rayFunction
volumeMapper SetInput [reader GetOutput]

- vtkVolumeRayCastCompositeFunction performs compositing along the ray
- vtkVolumeRayCastMipFunction performs MIP function
- vtkVolumeRayCastIsosurfaceFunction is a volume ray cast function that intersects a ray with an analytic isosurface in a scalar field

Compositing

vtkVolumeRayCastCompositeFunction
Important Methods
- SetCompositeMethodToInterpolateFirst
- SetCompositeMethodToClassifyFirst

Interpolate or Classify First?

- Interpolate
  1. Determine scalar value for sampled point via interpolation
  2. Use the interpolated value of the sampled point for classification (i.e., apply color and opacity transfer functions)

Interpolate-Classify Problem

- There is a consequence of interpolation before classification
- Assume scalar value 50 (e.g., soft tissue) lies between 20 (e.g., air) and 80 (e.g., bone)
- Results in images in which bone can never be adjacent to air
  - Layer of skin will be introduced on teeth inside mouth

Interpolate or Classify First?

- Classify - Interpolate
  1. Classify 8 vertices of voxel
  2. Interpolate RGB and opacity from vertices
Classify – Interpolate Problem

- There is a danger in interpolation after classification
- Naïve color interpolation would assign 3 parts yellow, 1 part blue to center point ...

\[ \alpha=1 \quad \alpha=1 \quad \alpha=0 \quad \alpha=1 \]

... but what if opacity of bottom left is zero?
Correct approach is to weight according to opacity, so color at center is yellow!

\[ \alpha = 1 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]

\[ \alpha = 1 \]

\[ \alpha = 0 \]
Step size = 0.1
Step size = 1.0
Step size = 2.0
20X faster than 0.1

• Voxelized vase with a 1x1x1 spacing between points in dataset
• Color transfer function has a rapid change from black to white
• Even with step size of 1.0 artifacts are introduced since color changes significantly within that distance.

Other Transfer Functions

• Map scalar gradient magnitude to opacity to enhance transition regions of a volume
  • may be more effective than using scalar values alone to do classification (good for edge detection)
  • Define transfer function as done with scalar to opacity and replace scalar value with gradient magnitude
  • Often used to eliminate homogeneous regions

Gradient Vector

• Is often used as a measure of orientation
  • In volume rendering, the gradient at a location is used to determine the surface normal at that location
  • Surface normal is need for illumination equations (if illumination is used)

Gradient Vector

• Finite difference technique to approximate gradient

\[ g_x = \frac{df}{dx} = 0.5 \left( \frac{f_{i+1,j,k} - f_{i-1,j,k}}{2} \right) \]

\[ g_y = \frac{df}{dy} = 0.5 \left( \frac{f_{i,j+1,k} - f_{i,j-1,k}}{2} \right) \]

\[ g_z = \frac{df}{dz} = 0.5 \left( \frac{f_{i,j,k+1} - f_{i,j,k-1}}{2} \right) \]

• Finite difference technique to approximate gradient
  • \( g_x \), \( g_y \), and \( g_z \) are the partial derivatives of this function along the x, y, and z axes

Tradeoff in step size:
Too large then we may lose small features in the dataset
Too small then may have too much high-frequency content

Gradient Opacity

vtkVolumeProperty volumeProperty

• VTK separated geometric and volume rendering actors into separate classes. Some methods of one class do not apply to the other. For example, SetRepresentationToWireframe makes no sense for volume rendering.

Gradient Vector

Similar to an Actor used in geometric rendering

Add volume to the renderer

VTK separated geometric and volume rendering actors into separate classes. Some methods of one class do not apply to the other. For example, SetRepresentationToWireframe makes no sense for volume rendering.

Gradient Opacity

vtkVolumeProperty volumeProperty

• VTK separated geometric and volume rendering actors into separate classes. Some methods of one class do not apply to the other. For example, SetRepresentationToWireframe makes no sense for volume rendering.
vtkVolumeProperty Summary

• Used to represent common properties associated with volume rendering including the interpolation method to use when sampling a volume, the color of a volume, the scalar opacity of a volume, the gradient opacity of a volume, and the shading parameters of a volume

• When a scalar and gradient opacity are both set, then the opacity is defined to be the product of the scalar opacity and gradient opacity transfer functions