Overview of VTK

November 1st, 2013

Hank Childs, University of Oregon
Announcements

• Final project
  – Come in & talk with me
• Quiz #2 delayed to Nov 6th
• OH:
  – Weds 10-11:30
  – Thurs 10-11
Announcements

• Thanksgiving lecture
• Faculty Fireside
• Project 7 & 8 likely to become 7A & 7B
Outline

• 3D data
• VTK
• Extra time: do some cases
Bijective function for rectilinear meshes for this course

```c
int GetPoint(int i, int j, int k, int nX, int nY, int nZ)
{
    return k*nX*nY + j*nX + i;
}
```

![Matrix example](image)
Bijective function for rectilinear meshes for this course

```c
int *GetLogicalPointIndex(int point,
                           int nX, int nY, int nZ)
{
    int rv[3];
    rv[0] = point % nX;
    rv[1] = (point/nX) % nY;
    rv[2] = (point/(nX*nY));
    return rv; // terrible code!!
}
```
Bijective function for rectilinear meshes for this course

```c
int GetCell(int i, int j, int k,
            int nX, int nY, int nZ)
{
    return k*(nX-1)*(nY-1) + j*(nX-1) + i;
}
```
Bijective function for rectilinear meshes for this course

```c
int *GetLogicalCellIndex(int cell, int nX, int nY, int nZ)
{
    int rv[3];
    rv[0] = cell % (nX-1);
    rv[1] = (cell/(nX-1)) % (nY-1);
    rv[2] = (cell/((nX-1)*(nY-1)));
    return rv; // terrible code!!
}
```
Trilinear Interpolation

What is the value of $F(0.9, 0.9, 0.9)$?

Then do bilinear interpolation along plane $X=0.9$.
Visualization with VTK

Content from: Erik Vidholm, Univ of Uppsula, Sweden
David Gobbi, Robarts Research Institute, London, Ontario, Canada
Outline

• What is VTK?
• What can it be used for?
• How do I actually use it?
VTK – The Visualization ToolKit

- Open source, freely available software for 3D computer graphics, image processing, and visualization
- Managed by Kitware Inc.
- Strictly object-oriented design (C++)
- High-level of abstraction
- Use C++, Tcl/Tk, Python, Java
True visualization system

- Visualization techniques for visualizing
  - Scalar fields
  - Vector fields
  - Tensor fields
- Polygon reduction
- Mesh smoothing
- Image processing
- Your own algorithms
Additional features

• Parallel support (message passing, multithreading)
• Stereo support
• Integrates easily with Motif, Qt, Tcl/Tk, Python/Tk, X11, Windows, ...
• Event handling
• 3D widgets
3D graphics

• Surface rendering
• Volume rendering
  – Ray casting
  – Texture mapping (2D)
  – Volume pro support
• Lights and cameras
• Textures
• Save render window to .png, .jpg, ...
  (useful for movie creation)
Objects

• Data objects
  – Next slide

• Process objects
  – Source objects (vtkReader, vtkSphereSource)
  – Filter objects (vtkContourFilter)
  – Mapper objects (vtkPolyDataMapper)
Data model
VTK_QUADRATIC_EDGE  
(=21)

VTK_QUADRATIC_TRIANGLE  
(=22)

VTK_QUADRATIC_QUAD  
(=23)

VTK_QUADRATIC_TETRA  
(=24)

VTK_QUADRATIC_HEXAHEDRON  
(=25)
Visualization continued

• Scalar algorithms
  – Iso-contouring
  – Color mapping

• Vector algorithms
  – Hedgehogs
  – Streamlines / streamtubes

• Tensor algorithms
  – Tensor ellipsoids
The visualization pipeline

DATA

FILTER

MAPPING

Visualization algorithms

Interactive feedback

DISPLAY
from vtkpython import *
cone = vtkConeSource()
cone.SetResolution(10)
coneMapper = vtkPolyDataMapper()
coneMapper.SetInput(cone.GetOutput())
coneActor = vtkActor()
coneActor.SetMapper(coneMapper)
ren = vtkRenderer()
ren.AddActor(coneActor)
renWin = vtkRenderWindow()
renWin.SetWindowName("Cone")
renWin.SetSize(300,300)
renWin.AddRenderer(ren)
iren = vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)
iren.Initialize()
iren.Start()
Imaging

• Supports streaming => huge datasets
• \texttt{vtkImageToImageFilter}
  – Diffusion
  – High-pass / Low-pass \text{(Fourier)}
  – Convolution
  – Gradient \text{(magnitude)}
  – Distance map
  – Morphology
  – Skeletons
Summary +

- Free and open source
- Create graphics/visualization applications fairly fast
- Object oriented - easy to derive new classes
- Build applications using "interpretive" languages Tcl, Python, and Java
- Many (state of the art) algorithms
- Heavily tested in real-world applications
- Large user base provides decent support
- Commercial support and consulting available
Summary -

• Not a super-fast graphics engine due to portability and C++ dynamic binding – you need a decent workstation

• Very large class hierarchy => learning threshold might be steep

• Many subtleties in usage
  – Pipeline execution model
  – Memory management
reader = vtkBMPReader()
reader.SetFileName("image.bmp")

blur = vtkImageGaussianSmooth()
blur.SetInput(reader.GetOutput())
blur.SetDimensionality(2)
blur.SetStandardDeviations(5.0, 5.0)
blur.SetRadiusFactors(10.0, 10.0)

subtract = vtkImageMathematics()
subtract.SetOperationToSubtract()
subtract.SetInput1(reader.GetOutput())
subtract.SetInput2(blur.GetOutput())

writer = vtkBMPWriter()
writer.SetInput(subtract.GetOutput())
writer.SetFileName("image2.bmp")
writer.Write()

viewer = vtkImageViewer()
viewer.SetInput(subtract.GetOutput())
viewer.SetColorWindow(255)
viewer.SetColorLevel(127.5)
viewer.Render()
Example – Vector field visualization

```python
vtkStructuredGridReader reader
    reader SetFileName "office.binary.vtk"

# Create source for streamtubes
vtkPointSource seeds
    seeds SetRadius 0.15
eval seeds SetCenter 0.1 2.1 0.5
    seeds SetNumberOfPoints 6
vtkRungeKutta4 integ
vtkStreamLine streamer
    streamer SetInput [reader GetOutput]
    streamer SetSource [seeds GetOutput]
    streamer SetMaximumPropagationTime 500
    streamer SetStepLength 0.5
    streamer SetIntegrationStepLength 0.05
    streamer SetIntegrationDirectionToIntegrateBothDirections
    streamer SetIntegrator integ
...```
The visualization pipeline - example

- \texttt{vtkStructuredPointsReader} 
  "hydrogen.vtk"
- \texttt{vtkMarchingCubes}
- \texttt{vtkPolyDataMapper}
  \texttt{vtkActor}
- \texttt{vtkRenderWindowInteractor}
- \texttt{vtkRenderer}
Python example: visualization hydrogen molecule

```python
# File: isosurface.py
import vtk

# image reader
reader = vtk.vtkStructuredPointsReader()
reader.SetFileName("hydrogen.vtk")
reader.Update()

# bounding box
outline = vtk.vtkOutlineFilter()
outline.SetInput( reader.GetOutput() )
outlineMapper = vtk.vtkPolyDataMapper()
outlineMapper.SetInput( outline.GetOutput() )
outlineActor = vtk.vtkActor()
outlineActor.SetMapper( outlineMapper )
outlineActor.GetProperty().SetColor(0.0,0.0,1.0)
```

Must call update to read!

Pipeline connections

---

Note: The code snippet provided is a simple example of reading a VTK file and visualizing it. In a real-world scenario, you might need to handle errors, add more complex visualizations, or perform additional data analysis.
Example continued

```python
# iso surface
isosurface = vtk.vtkContourFilter()
isosurface.SetInput( reader.GetOutput() )
isosurface.SetValue( 0, .2 )
isosurfaceMapper = vtk.vtkPolyDataMapper()
isosurfaceMapper.SetInput( isosurface.GetOutput() )
isosurfaceMapper.SetColorModeToMapScalars()
isosurfaceActor = vtk.vtkActor()
isosurfaceActor.SetMapper( isosurfaceMapper )

# slice plane
plane = vtk.vtkImageDataGeometryFilter()
plane.SetInput( reader.GetOutput() )
planeMapper = vtk.vtkPolyDataMapper()
planeMapper.SetInput( plane.GetOutput() )
planeActor = vtk.vtkActor()
planeActor.SetMapper( planeMapper )
```

`vtkContourFilter` chooses the appropriate method for the data set.
Example continued

# a colorbar
scalarBar = vtk.vtkScalarBarActor()
scalarBar.SetTitle("Iso value")

# renderer and render window
ren = vtk.vtkRenderer()
ren.SetBackground(.8, .8, .8)
renWin = vtk.vtkRenderWindow()
renWin.SetSize( 400, 400 )
renWin.AddRenderer( ren )
Example continued

The `RenderWindowInteractor` contains functions for mouse/keyboard interaction

```python
# render window interactor
iren = vtk.vtkRenderWindowInteractor()
iren.SetRenderWindow( renWin )

# add the actors
ren.AddActor( outlineActor )
ren.AddActor( isosurfaceActor )
ren.AddActor( planeActor )
ren.AddActor( scalarBar )

# this causes the pipeline to "execute"
renWin.Render()

# initialize and start the interactor
iren.Initialize()
iren.Start()
```

The `renWin.Render()` calls `Update()` on the renderer, which calls `Update()` for all its actors, which calls...
The VTK file format

- Many modules to write VTK files

```plaintext
# vtk DataFile Version 2.0
Hydrogen orbital
ASCII
DATASET STRUCTURED_POINTS
DIMENSIONS 64 64 64
ORIGIN 32.5 32.5 32.5
SPACING 1.0 1.0 1.0
POINT_DATA 262144
SCALARS probability float
LOOKUP_TABLE default
0.0 0.0 0.01 0.01 ..... 
```
VTK and C++

- Build with CMake and your favorite compiler
- CMake generates makefiles or project files for your environment
- Use the resulting file(s) to build your executable
- With C++ you have full control and can derive own classes, but you need to write many lines of code...
Development Environment Choices

• Languages: Python, Java, C++, Tcl
• Tradeoffs:
  – productivity - low-level (C++) vs high-level (Python)
  – reusability and maintainability - depends mainly on you
  – portability - can your code be used by other groups?
  – speed - because most computations will be done by the VTK classes (which are written in C++), speed will not depend on the application language
Development Environment Choices

- **C++**
  - good C++ programming involves many difficult concepts
  - unless you are experienced, you won't be very productive and your code will not be portable or reusable

- **Python**
  - involves few difficult concepts
  - code is almost guaranteed to be portable
VTK resources

- [www.vtk.org](http://www.vtk.org)
  - Download (source and binaries)
  - Documentation
  - Mailing lists
  - Links
  - FAQ, Search

- [www.kitware.com](http://www.kitware.com)
  - VTK Textbook
  - VTK User’s guide
  - Mastering CMake
3 major components of VTK

• Data Model
  → grid types, data types

• Execution Model
  → data flow

• Algorithms
  → filters
Data flow networks

- Work is performed by a **pipeline**
- A pipeline consists of **data objects** and **components** (*sources, filters, and sinks*)
- Pipeline execution begins with a “pull”, which starts **Update** phase
- Data flows from component to component during the **Execute** phase
Data flow networks: strengths

- Flexible usage
  - Networks can be multi-input / multi-output

- Interoperability of modules

- Easy to extend
  - New derived types of filters
Data flow networks: weaknesses

- Execution of modules happens in stages
  - Algorithms are executed at one time
    - Cache inefficient
  - Memory footprint concerns
- Some implementations fix the data model.
Strategy for memory bloat: reference counting

• Arrays can often be shared between input and output of a filter
  – So re-use and don’t bloat memory
  – But how to prevent memory leaks?
    • Answer: reference count
Reference Counting

• Can’t call allocate objects on the stack
• All objects instantiated with “New”
• Add references explicitly
  – “Register()”
• Remove references explicitly
  – “Delete()”
• Object deleted when its reference count drops to zero.
Data flow networks: observations

- Source for managing flow of data is small and in one place
- Majority of code investment is in algorithms (derived types of filters), not in base classes (which manage data flow).

Algorithms don’t care about data processing paradigm ... they only care about operating on inputs and outputs.
Closing thoughts

• Project 7: out on Monday
• Will review Proj. 7 in class on Weds
• If time: do some cases
• Next time: Quiz #2, more VTK