Welcome to UPPMAX Workshop on
Scientific Visualization
with

Ingela Nyström ingela@cb.uu.se

UPPMAX
Uppsala Multidisciplinary Center for
Advanced Computational Science
http://www.uppmax.uu.se/
"UPPMAX provides resources and a common
environment for an extensive community of
researchers in a wide range of high performance
computing (HPC) research fields."

UPPMAX
Director Sverker Holmgren
Application experts:
- Quantum Chemistry: Hans Karlsson
- Algorithm & Code Development: Jarmo Rantakokko
- Scientific Visualization: Ingela Nyström
- Molecular Dynamics: Daniel Spångberg
- Bioinformatics: Ann-Charlotte Sonnhammer
System experts

Dictionary

visualize
- To form a mental image of;
envisage: tried to visualize the scene
as it was described
- To make visible

"Visualization offers a way to see the unseen"

Interpreting data in visual terms
- When data is complex: Collected/Computed
- When numerous data
- Visualization is not a substitute to,
but in addition to, statistical analysis
and other quantitative methods
- Visualization takes advantage of human
sensory abilities
- Pattern recognition, Trend discovery, etc.
Graphs are one type of visualization

Some more sophisticated examples

Computer Graphics
- Creating images with a computer – 3D

Visualization is more than computer graphics!

Scientific Visualization
- Scientific visualization is the process of exploring, transforming, and viewing data as images
- The dimensionality of the data is generally larger than or equal to 3
- Visualization is often interactive
- We are not trying to create realistic images, but to visualize the data in an informative way
- Dependent on the task given

Visualization serves many purposes
- “Pretty pictures”
- For further analysis
- Debugging
- ...
Visualization can be used in every step of most processes

- Problem formulation
- Mathematical modelling
- Software/Hardware
- Simulation
- Result
- Interpretation

General development of visualization

- Rather new discipline that still is developing into sub-areas
- Tool users vs tool developers
- Collaboration among computer scientists and computational scientists
- Faster computers, high-speed networks, new user-interfaces

VTK – The Visualization ToolKit

- What is VTK?
- What can VTK be used for?
- How to actually use VTK?

VTK – The Visualization ToolKit

- Open source, freely available software for
  3D computer graphics
  image processing
  visualization
- Managed by Kitware, Inc.
- 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
  multi-threading
- Stereo support
- Integrates 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)

### Data Representation
- **Cells & Points**
- **Topology**
  - Shape such as triangle, tetrahedron
  - Geometry
  - Point coordinates assigned to a topology
- **Data attributes**
  - Data associated with topology or geometry

### Cells specify Topology
- Vertex
- Poly-vertex
- Line
- Poly-line
- Triangle
- Triangle strip
- Quadrilateral
- Polygon
- Tetrahedron
- Hexahedron
- Voxel

### Cells
- A Cell is defined by an ordered list of points
  - Triangle and quadrilateral vertices specified counterclockwise

### Meshes consist of Cells
- Cells can have different shapes and sizes
  - 2D: Triangles, Quadrilaterals, etc.
  - 3D: Tetrahedra, Hexahedra, Pyramids, etc.
- Meshes can consist of one or more types of cells

### VTK Dataset Types
- `vtkStructuredPoints`, `vtkImageData`
- `vtkRectilinearGrid`
- `vtkStructuredGrid`
- `vtkPolyData`
- `vtkUnstructuredGrid`
- Methods for reading and writing
Datasets
- Organizing structure plus attributes
  - Structured Points
  - Rectilinear Grid
  - Structured Grid

Unstructured Grid
A collection of vertices, edges, faces, and cells whose connectivity information must be explicitly stored

How are unstructured meshes different than regular grids?
- Regular Grids
  - mesh info accessed implicitly using grid point indices
  - efficient in both computation and storage
  - typically use finite difference (FD) discretization
  - Cartesian grids or logically rectangular grids

How are unstructured meshes different than regular grids?
- Unstructured Meshes
  - mesh connectivity information must be stored
  - handles complex geometries and grid adaptivity
  - typically use finite volume or finite element (FE) discretization
  - mesh quality becomes a concern

Data attributes assigned to points or cells
- Scalar
- Vector
  - magnitude and direction
- Normal
  - a vector of magnitude 1
  - used for lighting
- Texture coordinate
  - mapping data points into a texture space
- Tensor

Visualization of attributes
- Scalar
  - Color Mapping
- Contouring
  - 3D isosurface

Contour value of 5
Visualization of attributes

- Vector
  - Oriented Line
  - Oriented Glyph
- Streamline

Visualization continued

- Scalar algorithms
  - Iso-contouring
  - Colour mapping
- Vector algorithms
  - Hedgehogs
  - Streamlines / streamtubes
- Tensor algorithms
  - Tensor ellipsoids

Basic VTK objects to render a scene:

1. `vtkRenderWindow`
2. `vtkRenderer`
3. `vtkLight`
4. `vtkCamera`
5. `vtkActor`
6. `vtkProperty`
7. `vtkMapper`

The purpose is to render the geometry (volume) on the screen.

Example Program

```cpp
#include <vtkRenderWindow.h>
#include <vtkRenderer.h>
#include <vtkLight.h>
#include <vtkCamera.h>
#include <vtkActor.h>
#include <vtkProperty.h>
#include <vtkMapper.h>

int main()
{
  // Create a window
  Window
  // Create a renderer; give the renderer to the window
  Renderer
  // Create a light
  Light
  // Create a camera
  Camera
  // Create an actor
  Actor
  // Create a mapper; give the geometry to the mapper
  Mapper
  // Create a mapper; give the mapper to the actor
  Actor
  // Give the actor to the renderer
  Actor
  // Render the window
  RenderWindowInteractor
  return 0;
}
```

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.

The Visualization Pipeline

- DATA
- Visualization algorithms
- FILTER
- MAPPING
- Interactive feedback
- DISPLAY

The Visualization Pipeline - An Example

- Objects
  - Data objects
    - vtkPolyData
    - vtkImageData
  - Process objects
    - Source objects (vtkReader, vtkSphereSource)
    - Filter objects (vtkContourFilter)
    - Mapper objects (vtkPolyDataMapper)

The Hydrogen example - Python

```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)

# 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 )
```

Example continued

```python
# iso surface
isosurface = vtk.vtkContourFilter()
isosurface.SetInput( reader.GetOutput() )
isosurface.SetInputConnection( "isosurfaceReader" )
isosurfaceMapper = vtk.vtkPolyDataMapper()
isosurfaceMapper.SetInputConnection( "isosurfaceReader" )
isosurfaceMapper.SetColorModeToMapScalars()
isosurfaceActor = vtk.vtkActor()
isosurfaceActor.SetMapper( isosurfaceMapper )
```
Example continued

create a legend from the data and a lookup table

X is colorbar.
scalarBar = vtk.vtkScalarBarActor()
scalarBar.SetTitle("Iso value")

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

create a legend from the data and a lookup table

# 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()

Renditions

Would you like an icecream?

User interaction

- vtkRenderWindowInteractor
  - allows the user to interact with the objects
- Try the following key presses
  - w wireframe mode
  - s surface mode
  - j joystick mode
  - t trackball mode
  - button1 rotate
  - button2 translate
  - button3 scale
  - r reset camera view
  - e, q exit

The VTK file format

# 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 ....

Example continued

vtkRenderWindowInteractor contains functions for mouse/keyboard interaction

# render window interactor
calls Update() on the renderer, which calls Update() for all the actors, which calls...
renWin.Render()
renWin.Render() # this causes the pipeline to "execute"

vtkRenderWindowInteractor contains functions for mouse/keyboard interaction

# render window interactor
calls Update() on the renderer, which calls Update() for all the actors, which calls...
renWin.Render() # this causes the pipeline to "execute"

Would you like an icecream? Would you like an icecream?
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...

VTK resources

- www.vtk.org
- Download (source and binaries)
- Documentation
- Mailing lists
- Links
- FAQ, Search
- www.kitware.com
- VTK Textbook
- VTK User’s guide
- Mastering CMake

Questions?

Let’s try VTK