Programming for Image Processing/Analysis and Visualization using The Visualization Toolkit

http://noodle.med.yale.edu/seminar/seminar.html

Xenios Papademetris
papad@noodle.med.yale.edu
BML 325, 5-7294

Course Structure

1. Using VTK using scripting languages
   • Understand Toolkit Structure
   • Use existing algorithms (incl. local extensions)
   • Learn Tcl/Tk
2. Extending VTK
   • C++/Object Oriented Programming/Design
   • Cross-Platform Issues
   • Coding Guidelines

Schedule – Part 1

1. Introduce VTK (today)
2. Introduce Tcl/Tk
3. Simple Visualization Tasks
4. Simple Image/Surface Manipulation
5. More on Images and Volume Rendering
6. Local Extensions
   • (Marcel: Itcl/Itk/Iwidgets, more advanced GUI generation etc.)

Schedule – Part 2

1. C++ Fundamentals, Pointers/Classes
2. Object Oriented Programming
3. Adding new VTK Commands/Cmake
4. More on Images and Surfaces
5. Case Study I – Computing t-maps in fMRI
6. Case Study II -- Iterative Closest Point surface matching
7. Case Study III – Linear Mutual Information Multimodal Registration

VTK Pipeline (I)

VTK Pipeline II

Sources Filters Mappers

File Output

Props

Renderer Render Window

Props

Props

 vtkDataSet vtkDataSet

vtkCamera, vtkLight

vtkProperty

vtkRenderWindowInteractor
Data Representation (vtkDataSet)

Points (vtkPoints) Define Location

Cells (vtkCellArray) Define Topology

Point Attributes (vtkPointData)
Point Properties (e.g. Intensity)

Arrays of Numbers (per point or cell)
vtkDataArray

Cell Attributes (vtkPointData)
Point Properties (e.g. normal)

Cells specify Topology

- Polygon
- Tetrahedron
- Hexahedron
- Triangle
- Line
- etc

Cells

- Cell is defined by an ordered list of points
  - Triangle, quadrilateral points specified counter clockwise
  - Others as shown

VTK Dataset Types

- vtkStructuredPoints
- vtkRectilinearGrid
- vtkStructuredGrid
- vtkPolyData
- vtkUnstructuredGrid

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
Data Attributes Assigned to points or cells

- Scalars
- Vector
  - Magnitude and direction
- Normal
  - A vector of magnitude 1
  - Used for lighting
- Texture Coordinate
  - Mapping data points into a texture space
- Tensor

Object Oriented Programming

- Impossible to Cover in 10 minutes … but
- Traditional programming (i.e. C, Fortran, Matlab) is procedural:
  - If we have matrix 'a'
  - To print it we do: print(a), disp(a) etc,
- Data/Procedures are separate, data is dumb

Object Oriented Programming II

- In OOP data is “intelligent” i.e. data structures encapsulate procedures i.e.
- If we have matrix 'a'
  - To print it we do: a.print()
- Procedures are embedded within the data structure definition (called methods)
- Lots of good reasons for this …

Object Oriented Programming III

- Data structures are classes: e.g. in C++
  ```cpp
class Matrix3x3 {
    float a[3][3];
    public:
    void Print();
    void Invert();
    void Load(char* filename);
    etc.
}
```
  - To use:
    ```cpp```
    ```cpp```

OOP IV -- Inheritance

- Consider now need to change file format for matrix:
- INHERIT new class myMatrix3x3 from Matrix3x3 and override Print()
  ```cpp```
  ```cpp```
  ```cpp```
  ```cpp```
  ```cpp```
To use:
```cpp```
```cpp```
```cpp```
```cpp```
This calls the Print function from Matrix3x3 class

VTK Hierarchy

- To understand VTK structure need to look at class hierarchy
- Common functionality implemented in base-level (parent) classes
- Specialized functionality implemented in lower-level (children) classes
VTK Hierarchy

- To understand man pages
  http://www.vtk.org/doc/release/4.0/html/classes.html