Lecture 8: CS6250 Graphics & Visualization

Data Representation for Visualization

- Data Abstractions in General
- VTK specifics
- Examples

 ©2013 David M. Chelberg

Other Models for Data

AVS

The first large-scale commercial visualization system.

- Data flow architecture
- Explicit executive

 ©2013 David M. Chelberg

Data Model:

- Primitive data
  - Byte
  - Real
  - Integer
  - String
- Aggregate data
  - Fields
  - Color maps
  - Geometries (points, lines, polylines, …)
  - Pixel maps (output)

 ©2013 David M. Chelberg

Fields

A field can be thought of as an n-dimensional data model. (can be represented as an array with a scalar or vector for each entry).

Users define a mapping from the nD array to coordinate points.

- Uniform (structured)
- Rectilinear
- Irregular (unstructured)

 ©2013 David M. Chelberg

The Data Explorer

Their data model is based on a general mathematical model (fiber bundles).

The mathematics involves representing fields as patches of regular and irregular grids stitched together.

It is a very abstract model of data.

 ©2013 David M. Chelberg

VTK's implementation

Contiguous arrays.

Arrays are not arrays of objects. Why?

 ©2013 David M. Chelberg

Abstract vs. Concrete Class Structure

 ©2013 David M. Chelberg

Dataset Type Representation

Each type has a different internal data representation.

Why?

Some types require explicit specification of the cell topology. These types make use of a `vtkCellArray` for this.

 ©2013 David M. Chelberg

vtkUnstructuredGrid

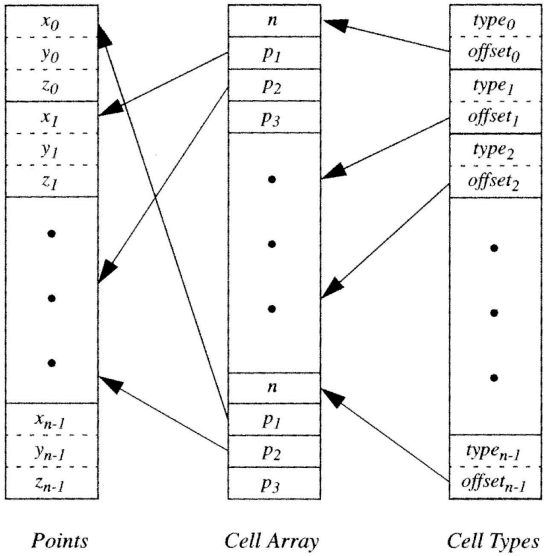


Figure 5–13 The data structure of the class vtkUnstructuredGrid. (This is a subset of the complete structure. See Chapter 8 for complete details.)

 ©2013 David M. Chelberg

Example: Cube

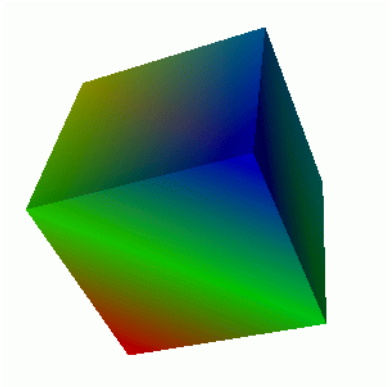
```
int main( int argc, char *argv[] ){
  int i;
  static float x[8][3]={{0,0,0}, {1,0,0}, {1,1,0}, {0,1,0},
                        {0,0,1}, {1,0,1}, {1,1,1}, {0,1,1}};
  static int pts[6][4]={{0,1,2,3}, {4,5,6,7}, {0,1,5,4},
                        {1,2,6,5}, {2,3,7,6}, {3,0,4,7}};

  vtkRenderer *renderer = vtkRenderer::New();
  vtkRenderWindow *renWin = vtkRenderWindow::New();
    renWin->AddRenderer(renderer);

  vtkRenderWindowInteractor *iren = vtkRenderWindowInteractor::New();
    iren->SetRenderWindow(renWin);

  vtkPolyData *cube = vtkPolyData::New();
  vtkPoints *points = vtkPoints::New();
  vtkCellArray *polys = vtkCellArray::New();
  vtkFloatArray *scalars = vtkFloatArray::New();

  for (i=0; i<8; i++) points->InsertPoint(i,x[i]);
  for (i=0; i<6; i++) polys->InsertNextCell(4,pts[i]);
  for (i=0; i<8; i++) scalars->InsertTuple1(i,i);

  cube->SetPoints(points);
  points->Delete();
```

 ©2013 David M. Chelberg

```
cube->SetPolys(polys);
polys->Delete();
cube->GetPointData()->SetScalars(scalars);
scalars->Delete();

vtkPolyDataMapper *cubeMapper = vtkPolyDataMapper::New();
    cubeMapper->SetInput(cube);
    cubeMapper->SetScalarRange(0,7);
vtkActor *cubeActor = vtkActor::New();
    cubeActor->SetMapper(cubeMapper);

vtkCamera *camera = vtkCamera::New();
    camera->SetPosition(1,1,1);
    camera->SetFocalPoint(0,0,0);
    camera->ComputeViewPlaneNormal();
renderer->AddActor(cubeActor);
    renderer->SetActiveCamera(camera);
    renderer->ResetCamera();
    renderer->SetBackground(1,1,1);
```

 ©2013 David M. Chelberg

Cube Output



 ©2013 David M. Chelberg

Modified Cube:

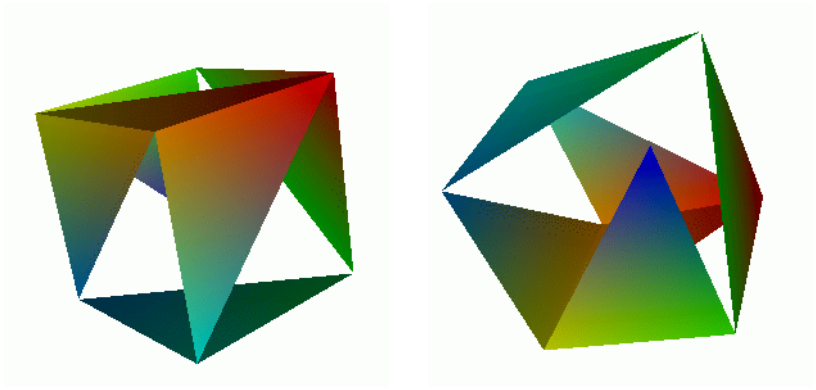
```
int main( int argc, char *argv[] ){
  int i;
  static float x[8][3]={{0,0,0}, {1,0,0}, {1,1,0}, {0,1,0},
                        {0,0,1}, {1,0,1}, {1,1,1}, {0,1,1}};
  static int pts[6][4]={{0,1,2,3}, {4,5,6,7}, {0,1,5,4},
                        {1,2,6,5}, {2,3,7,6}, {3,0,4,7}};

  vtkRenderer *renderer = vtkRenderer::New();
  vtkRenderWindow *renWin = vtkRenderWindow::New();
    renWin->AddRenderer(renderer);

  vtkRenderWindowInteractor *iren = vtkRenderWindowInteractor::New();
    iren->SetRenderWindow(renWin);

  vtkPolyData *cube = vtkPolyData::New();
  vtkPoints *points = vtkPoints::New();
  vtkCellArray *polys = vtkCellArray::New();
  vtkFloatArray *scalars = vtkFloatArray::New();

  for (i=0; i<8; i++) points->InsertPoint(i,x[i]);
  for (i=0; i<6; i++) polys->InsertNextCell(3,pts[i]);
  for (i=0; i<8; i++) scalars->InsertTuple1(i,i);

  cube->SetPoints(points);
```

 ©2013 David M. Chelberg

```
  points->Delete();
  cube->SetStrips(polys);
  polys->Delete();
  cube->GetPointData()->SetScalars(scalars);
  scalars->Delete();

  vtkPolyDataMapper *cubeMapper = vtkPolyDataMapper::New();
      cubeMapper->SetInput(cube);
      cubeMapper->SetScalarRange(0,7);
  vtkActor *cubeActor = vtkActor::New();
      cubeActor->SetMapper(cubeMapper);

  vtkCamera *camera = vtkCamera::New();
      camera->SetPosition(1,1,1);
      camera->SetFocalPoint(0,0,0);
      camera->ComputeViewPlaneNormal();
  renderer->AddActor(cubeActor);
      renderer->SetActiveCamera(camera);
      renderer->ResetCamera();
      renderer->SetBackground(1,1,1);
```

 ©2013 David M. Chelberg

Modified Cube Output:



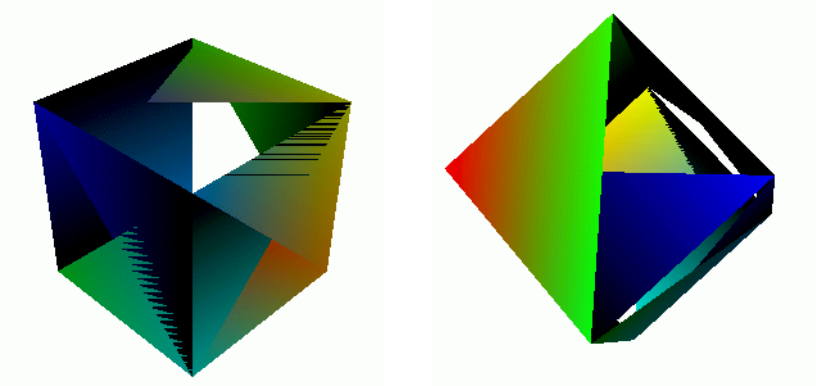
 ©2013 David M. Chelberg

```
int main( int argc, char *argv[] ){
  int i;
  static float x[8][3]={{0,0,0}, {1,0,0}, {1,1,0}, {0,1,0},
                        {0,0,1}, {1,0,1}, {1,1,1}, {0,1,1}};
  static int pts[6][4]={{0,1,2,3}, {4,5,6,7}, {0,1,5,4},
                        {1,2,6,5}, {2,3,7,6}, {3,0,4,7}};

  vtkRenderer *renderer = vtkRenderer::New();
  vtkRenderWindow *renWin = vtkRenderWindow::New();
    renWin->AddRenderer(renderer);

  vtkRenderWindowInteractor *iren = vtkRenderWindowInteractor::New();
    iren->SetRenderWindow(renWin);

  vtkPolyData *cube = vtkPolyData::New();
  vtkPoints *points = vtkPoints::New();
  vtkCellArray *polys = vtkCellArray::New();
  vtkFloatArray *scalars = vtkFloatArray::New();

  for (i=0; i<8; i++) points->InsertPoint(i,x[i]);
  for (i=0; i<6; i++) polys->InsertNextCell(4,pts[i]);
  for (i=0; i<8; i++) scalars->InsertTuple1(i,i);

  cube->SetPoints(points);
  points->Delete();
  cube->SetStrips(polys);
  polys->Delete();
```

 ©2013 David M. Chelberg

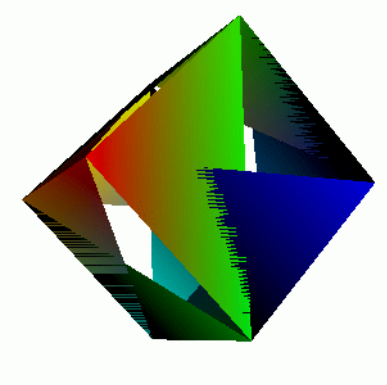
```
cube->GetPointData()->SetScalars(scalars);
scalars->Delete();
```

4-Strips Output



 ©2013 David M. Chelberg

What is wrong with these images?



 ©2013 David M. Chelberg

Algorithms

We will now consider a large number of visualization algorithms. We will be looking at both how they work, as well as what they are good for.

 ©2013 David M. Chelberg