Unstructured Grids

Hank Childs, University of Oregon
Announcements (1/2)

• Grading:
  – 6A, 6B, 7A graded
  – To do: 5, 7b (both 510 only)

• Current denominators:
  – 410: 65 points
  – 510: 85 points

• Feedback: fantastic job!
  – put them on your resumes...
Announcements (2/2)

• VisIt lecture: posted!

• Projects:
  – ~30 hours
  – Aim for something worthy of your resume

• Pre-defined project: posted!

• Schedule:
  – today: unstructured grids + FAQ on VTK
  – Weds: large data vis
  – Fri: Lagrangian flow (research)
Review
Types of Meshes

- Curvilinear
- Adaptive Mesh Refinement
- Unstructured
Curvilinear Mesh

- Logically rectilinear
  - Each cell has an (i, j)
  - Always left, right, bottom, top neighbor (unless on boundary)

- Points can be anywhere ... as long as the cells don’t overlap

VTK calls this “vtkStructuredGrid”
Curvilinear Mesh

- A curvilinear mesh has 5x5 cells and a cell-centered variable stored.
- Quiz: how many bytes to store this data set if all data is in single precision floating point?

\[
2 \text{ ints } + 6 \times 6 \text{ floats (points) } + 5 \times 5 \text{ floats (variable)} = 2 \times 4 + 36 \times 4 + 25 \times 4 = 63 \times 4 = 272 \text{ bytes}
\]
Example unstructured mesh

- Meshes contain:
  - Cells
  - Points
- This mesh contains 3 cells and 13 vertices
- Pseudonyms:
  - Cell == Element == Zone
  - Point == Vertex == Node
If we stored each cell like this, how many bytes would it take? (assume single precision)

Let’s call this the “explicit” scheme

Cell 0:
{ (0, 0, 0), // 10
  (1, 0, 0), // 11
  (1, 1, 0), // 8
  (0, 1, 0), // 7
  (0, 0, 1), // 4
  (1, 0, 1), // 5
  (1, 1, 1), // 2
  (0, 1, 1) } // 1

A: 1 int (# cells), 3 ints (# pts per cell), 24+24+15 floats = 268 bytes
Example unstructured mesh

• Pts:
  – \{(0, 1, 1), (1, 1, 1), (2,1,1),
    (0,0,1), (1,0,1), (2,0,1),
    (0,1,0),
    (1,1,0),
    (2,1,0),
    (0,0,0), (1,0,0),
    (2,0,0), (2.5, 0.5, 0.5)\}

• Cells:
  – 3 cells
    • 1st cell: hexahedron
      – (10,11,8,7,4,5,2,1)
    • 2nd cell: hexahedron
      – (11,12,9,8,5,6,3,2)
    • 3rd cell: prism
      – (13,3,6,12,9)

If we stored each cell like this, how many bytes would it take? (assume single precision)

A: 1 int (# pts), 1 int (# cells), 3 ints (# cell type), 13*3 floats (pts), 8+8+5 ints = 260 bytes

Let’s call this the “connectivity” scheme
New Material
Comparing unstructured mesh storage schemes

• Hexahedral meshes: each internal point incident to 8 cells
  – Explicit scheme:
    • represent that point 8 times: 24 floats for storage
  – Connectivity scheme:
    • represent that point once in point list, 8 times in connectivity list: 3 floats + 8 ints

• (takeaway: connectivity wins!)

Further benefit to connectivity scheme is in finding exterior faces.
Finding external faces: motivation

• Interval volume, clip:
  – Take data set (rectilinear, unstructured, or other) and produce unstructured mesh
  – When rendering, only want to render the faces on the outside (the inside aren’t visible)

Question: what proportion of faces are exterior?

Question: how to find exterior faces?
Finding external faces: algorithm

• For each face, count how many cells are incident.
  – If “1”, then external
  – If “2”, then interior

Question: why does this work?
Finding exterior faces: algorithm

• Estimate # of faces (ncells * 6 / 2)
• Double that number
• Create a hash table of that size
• For each cell C
  – For each face F of C
    • Create hash index for F based on connectivity indices
    • Search hash table
      – If F already there, remove F from hash
      – If face not there, add F to hash
• All faces in hash are exterior
Interpolation for arbitrary cells: tetrahedrons

- Assume tetrahedron T, point P in T
- Goal: calculate F(P)
Interpolation for arbitrary cells: tetrahedrons

- Assume tetrahedron T, point P in T
- Goal: calculate $F(P)$

Set up parametric coordinate system
Interpolation for arbitrary cells: tetrahedrons

- Assume tetrahedron $T$, point $P$ in $T$
- Goal: calculate $F(P)$

Calculate parametric coordinates $(a, b, c)$

This is a 3x3 matrix solve. This matrix is invertible since $R, S, T$ form a basis.

$P = P0 + aR + bS + cT$
Interpolation for arbitrary cells: tetrahedrons

- Assume tetrahedron T, point P in T
- Goal: calculate $F(P)$

$P = P_0 + aR + bS + cT$

$F(P) = \text{sum}(W_i \cdot F(P_i))$

- $W_0 = 1 - a - b - c$
- $W_1 = a$
- $W_2 = b$
- $W_3 = c$

Calculate $F(P)$ as weighted sum of vertices
General idea

• Set up parametric coordinate system
• Calculate parametric coordinates for P
• Calculate $F(P)$ as $\sum(W_i*F(P_i))$
  – Weights $W_i$ can get pretty tricky.

VTK book has weights & good description in Ch. 8.2.
How to do contouring

• Basically the same:
  – Iterate over cells
  – Identify case
  – Lookup case in table
  – Create resulting geometry

• Difference:
  – New tables for each cell type
How to do ray casting

• Basically the same:
  – Cast rays for every pixel on the screen
  – Sample along rays
  – Apply transfer function
  – Composite front to back
  – Assign color to image

• Differences:
  – Sampling gets hard!
    • Which cell contains a sample point?
      – Need smart data structures ....
      – .... Or a way to transform data to make it easy
How to do particle advection

• Basically the same:
  – Start with a seed location
  – Evaluate velocity
  – Displace particle
  – Repeat until termination criteria reached

• Differences:
  – Evaluating velocity different x2:
    • Now a harder proposition: which cell contains particle?
    • Now more math: how to LERP velocity?
Adaptive Mesh Refinement

• Put resolution where you want it...
• ... but simpler than unstructured (indexing & interpolation) and cheaper (memory)

Problems:
  – Everything great, except at the boundaries....
And now...

• You know everything I think you should know after having taken a sci-vis class.
  – Still need unstructured mesh lectures
• I am proud of this class.
• Next few lectures should be fun.
• Projects will hopefully be fun too.
Textures
First OpenGL programs

class vtk441PolyDataMapper : public vtkOpenGLPolyDataMapper
{
  public:
    static vtk441PolyDataMapper *New();
    virtual void RenderPiece(vtkRenderer *ren, vtkActor *act)
    {
      glEnable(GL_COLOR_MATERIAL);
      float ambient[3] = { 1, 1, 1 };
      glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, ambient);
      glBegin(GL_TRIANGLES);
      glColor3ub(0, 0, 255);
      glVertex3f(0,0,0);
      glColor3ub(0, 255, 0);
      glVertex3f(0,1,0);
      glColor3ub(255, 0, 0);
      glVertex3f(1,1,0);
      glEnd();
    }
};
Why is there purple in this picture?
Textures: a better way to specify a color map

There is no purple when we use textures
Textures

• “Textures” are a mechanism for adding “texture” to surfaces.
  – Think of texture of a cloth being applied to a surface
  – Typically used in 2D form

• Previous examples were 1D.
1D textures: basic idea

- Store color map on GPU as a texture
  - An array of colors

- Old color interpolation of fragment on a scanline:
  - For (int j = 0 ; j < 3 ; j++)
    - RGB[j] = leftRGB[j] + proportion*(rightRGB[j]-leftRGB[j])

- New color interpolation of fragment on a scanline:
  - textureVal = leftTextureVal
    - + proportion*(rightTextureVal-leftTextureVal)
  - RGB ← textureLookup[textureVal]
2D Texture Program

```c++
class vtk441PolyDataMapper : public vtkOpenGLPolyDataMapper
{
    public:
    static vtk441PolyDataMapper *New();

    virtual void RenderPiece(vtkRenderer *ren, vtkActor *act)
    {
        vtkJPEGReader *rdr = vtkJPEGReader::New();
        rdr->SetFileName("HankChilds_345.jpg");
        rdr->Update();
        vtkImageData *img = rdr->GetOutput();
        int dims[3];
        img->GetDimensions(dims);
        unsigned char *buffer = (unsigned char *)img->GetScalarPointer(0,0,0);
        glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, dims[0], dims[1], 0, GL_RGB,
                     GL_UNSIGNED_BYTE, buffer);
        glEnable(GL_COLOR_MATERIAL);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
        glEnable(GL_TEXTURE_2D);
        float ambient[3] = { 1, 1, 1 };
        glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, ambient);
        glBegin(GL_TRIANGLES);
        glVertex3f(0,0);
        glVertex3f(0,0);
        glVertex3f(1, 0);
        glVertex3f(0,1,0);
        glVertex3f(0,1,0);
        glVertex3f(1,1,0);
        glEnd();
    }
};
```

What do we expect the output to be?
vtkSmartPointer<vtkFloatArray> textureCoordinates = vtkSmartPointer<vtkFloatArray>::New();
textureCoordinates->SetNumberOfComponents(3);
textureCoordinates->SetName("TextureCoordinates");

float tuple[3] = {0.0, 0.0, 0.0};
textureCoordinates->InsertNextTuple(tuple);
tuple[0] = 1.0; tuple[1] = 0.0; tuple[2] = 0.0;
textureCoordinates->InsertNextTuple(tuple);
tuple[0] = 1.0; tuple[1] = 1.0; tuple[2] = 0.0;
textureCoordinates->InsertNextTuple(tuple);
tuple[0] = 0.0; tuple[1] = 2.0; tuple[2] = 0.0;
textureCoordinates->InsertNextTuple(tuple);

quad->GetPointData()->SetTCoords(textureCoordinates);

// Apply the texture
vtkSmartPointer<vtkTexture> texture = vtkSmartPointer<vtkTexture>::New();
texture->SetInputConnection(jPEGReader->GetOutputPort());

vtkSmartPointer<vtkPolyDataMapper> mapper = vtkSmartPointer<vtkPolyDataMapper>::New();
if VTK_MAJOR_VERSION <= 5
mapper->SetInput(quad);
else
mapper->SetInputData(quad);
endif

vtkSmartPointer<vtkActor> texturedQuad = vtkSmartPointer<vtkActor>::New();
texturedQuad->SetMapper(mapper);
texturedQuad->SetTexture(texture);
Anti-aliasing

Aliased

Anti-Aliased
VTK + Anti-aliasing

```cpp
virtual int vtkRenderWindow::GetAAFrames ( ) [virtual]

Set the number of frames for doing antialiasing. The default is zero. Typically five or six will yield reasonable results without taking too long.
```

```cpp
virtual void vtkRenderWindow::SetAAFrames ( int ) [virtual]

Set the number of frames for doing antialiasing. The default is zero. Typically five or six will yield reasonable results without taking too long.
```