Isosurfacing (Part 2)
Announcements

• Final project
  – Pre-defined project: Volume rendering
  – Finding data
    • http://sciviscontest.ieeevis.org/
    • Talk with me
• I may delay Nov 1\textsuperscript{st} quiz to Nov 6\textsuperscript{th} (stay tuned)
• OH: Fri 12-1 is a bad time
  – How would Thurs 2-3 be?
Project 5

• What is the ground truth?
Height field over a terrain
Isolines vs Isosurfaces

• Isolines:
  – Input: scalar field over 2D space
  – Output: lines

• Isosurfaces:
  – Input: scalar field over 3D space
  – Output: surface

• Commonalities:
  – Reduce topological dimension by 1
  – Produce output where scalar field is constant
Iterating Over Cells

• For isosurface/isoline calculation, we can iterate over the cells.
• At the end, we take the results from each cell and put them into a single scene.
Reality Check: context for isolines

- We have millions of cells
- If one cell can produce tens or hundreds of lines, then the isolines could take much more memory than the input data set
Big picture: what do we want from our isosurface?

• Tractable computation
  – Can’t create 100s or 1000s of line segments per cell for super-accurate representation

• Continuous surface
  – Triangles (or lines for an isosurface) need to connect up ... no gaps.
Big idea #1: approximate the isolines / isosurface

• Isolines: represent them with a minimal # of segments
• Isosurface: represent them with a minimal # of triangles
Quiz: how to approximate our “quarter circle”?
Big picture: what do we want from our isosurface?

- Tractable computation
  - Can’t create 100s or 1000s of line segments per cell for super-accurate representation

- Quiz: did we accomplish this?
  - Yes: very few per cell

- Continuous surface
  - Triangles (or lines for an isosurface) need to connect up ... no gaps.

- Quiz: did we accomplish this?

Answer: we got the answer exactly right at the edge of the cell ... hence no gaps.
The 16 cases

Case 0

Case 1-5

Case 6-10

Case 11-15
Big idea #2: do pre-computation to make it fast

If you knew which case you had, then you would know how to proceed

• Pre-compute correct answers for all 16 cases and store them in a lookup table
• For each cell, identify which case it is in
• Then use corresponding lookup table to generate isosurface
Big idea #2: pre-computation of all possible cases

If you knew which case you had, then you would know how to proceed

- Pre-compute correct answers for all 16 cases and store them in a lookup table
- For each cell, identify which case it is in
- Then use corresponding lookup table to generate isosurface
Pre-compute correct answers for all 16 cases and store them in a lookup table

• Observations about correct answers for a case:
  – It contains one or two line segments
  – The ends of the line segments are always along edges.
Pre-compute correct answers for all 16 cases and store them in a lookup table

• The ends of the line segments are always along edges.
  – We will need to number the edges
Big idea #2: do pre-computation to make it fast

If you knew which case you had, then you would know how to proceed

- Pre-compute correct answers for all 16 cases and store them in a lookup table
- For each cell, identify which case it is in
- Then use corresponding lookup table to generate isosurface
For each cell, identify which case it is in

- 4 vertices
- Each has one of 2 possible classification values
  - Lower than isovalue
    - Call this “0”
  - Higher than isovalue
    - Call this “1”
  - (ignore equality case)
For each cell, identify which case it is in:

- **Goal:** turn classification values into a number
  - Number should be between 0 and 15
- **Idea:** use binary numbers
  - $V3V2V1V0 \rightarrow 1110 \rightarrow 14$

This is case 14
The 16 cases

<table>
<thead>
<tr>
<th>Case</th>
<th>Case 0</th>
<th>Case 1-5</th>
<th>Case 6-10</th>
<th>Case 11-15</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>


Big idea #2: do pre-computation to make it fast

If you knew which case you had, then you would know how to proceed

• Pre-compute correct answers for all 16 cases and store them in a lookup table
• For each cell, identify which case it is in
• Then use corresponding lookup table to generate isosurface
Then use corresponding lookup table to generate isosurface

```c
int numSegments[16];
numSegments[0] = 0;
...
numSegments[6] = 2;
...
numSegments[14] = 1;
umSegments[15] = 0;
```
Then use corresponding lookup table to generate isosurface

```c
int lup[16][4]; // lup == lookup
lup[0][0] = lup[0][1] = lup[0][2] = lup[0][3] = -1;
...
lup[6][0] = 0; lup[6][1] = 1; lup[6][2] = 2; lup[6][3] = 3;
...
lup[14][0] = 0; lup[14][1] = 3; lup[14][2] = lup[14][3] = -1;
```
Then use corresponding lookup table to generate isosurface

```cpp
int icase = IdentifyCase(cell); // case is a reserved word in C++
int nsegments = numSegments[icase];
for (int i = 0 ; i < nsegments ; i++)
{
    int edge1 = lup[icase][2*i];
    float pt1[2] = // Interpolate position along edge1
    int edge2 = lup[icase][2*i+1];
    float pt2[2] = // Interpolate position along edge2
    AddLineSegmentToOutput(pt1, pt2);
}
```
Isosurfacing

• Will follow a very similar game plan.
  – Pre-compute correct answers for all cases and store them in a lookup table
  – For each cell, identify which case it is in
  – Then use corresponding lookup table to generate isosurface
Isosurfacing

Quiz: where should the isosurface go?
Isosurfacing

Quiz: where should the isosurface go?
Isosurfacing

Quiz: where should the isosurface go?
Isosurfacing

Quiz: where should the isosurface go?
Isosurfacing

Quiz: where should the isosurface go?
Isosurfacing

Quiz: where should the isosurface go?
Isosurfacing

Quiz: where should the isosurface go?
Isosurfacing

Quiz: where should the isosurface go?
Isosurfacing

Quiz: where should the isosurface go?
Isosurfacing

We need conventions!
Isosurfacing

We need conventions!
Isosurfacing

We need conventions!
Isosurfacing

We need conventions!
Isosurfacing

static int edges[12][2] =
{ {0,1}, {1,3}, {2,3}, {0,2},
  {4,5}, {5,7}, {6,7}, {4,6},
  {0,4}, {1,5}, {2,6}, {3,7} };
Different cell types

- We know how to contour cubes
- How do we do tetrahedrons? Prisms? Wedges?
Our isosurface approximation is improved by computer graphics
Project 6

• 6A: implement marching quads (Weds lecture), including all 16 cases
• 6B: implement marching cubes, but only a subset of the cases
  – We will use SVN to share our cases
  – There will be a case checker
    • Ambiguities are hard
• Everyone does 6A, some do 6B
• Please email if you want to do 6B