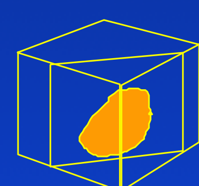
Volume Visualization

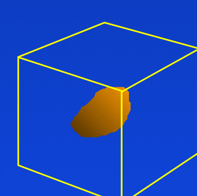
R. Daniel Bergeron
Department of Computer Science
University of New Hampshire
rdb@unh.edu

Volume Visualization Techniques

- Planar Slicing
  - move slice through space



- Isosurface —surface from equal valued cells
  - change value over time



- Direct volume rendering
  - viewing "gas" using color/opacity
  - ray casting and splatting

Volume Data

- Volume data is a set of data points in 3D
  - regularly spaced sampling is common from medicine
  - irregular sampling sometime occurs with finite element analysis problems
- Assume sampling from a *continuous* phenomena
- Regular sampling leads to division of volume into rectilinear *voxels* (volume data elements)
  - sometimes view the sample value as the center of a voxel, sometimes as a corner

Isosurface Rendering

Often useful to construct a surface within a volume that represents a constant value, k

Three common algorithms

- Connectivity
- Marching Cubes [Lorenson&Cline 87]
- Dividing Cubes [Cline et al. 88]

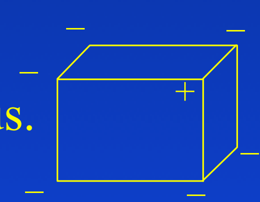
Connectivity Isosurface Algorithm

- Start with a "seed" voxel, recursively find neighboring voxels with same value

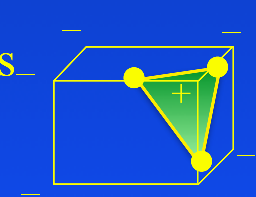
  connect(voxel(x,y,z)):
  - if voxel(x,y,z) intersects surface & is not marked
    mark (x,y,z)
    connect (x+1, y, z)
    connect (x-1, y, z)
    connect (x, y+1, z)
    connect (x, y-1, z)
    connect (x, y, z+1)
    connect (x, y, z -1)
- Allows separation of surfaces with same value

Marching Cubes Overview

1. Label each voxel vertex + (>=k) or – (<k)
2. Assign index to each voxel based on vertices. 64 cases, 15 unique ones, but some ambiguous.



3. For each voxel edge with +/– end points, linearly interpolate along edge to get estimate of position where value is k
4. For each voxel with +/– edges, connect points to get polygon.

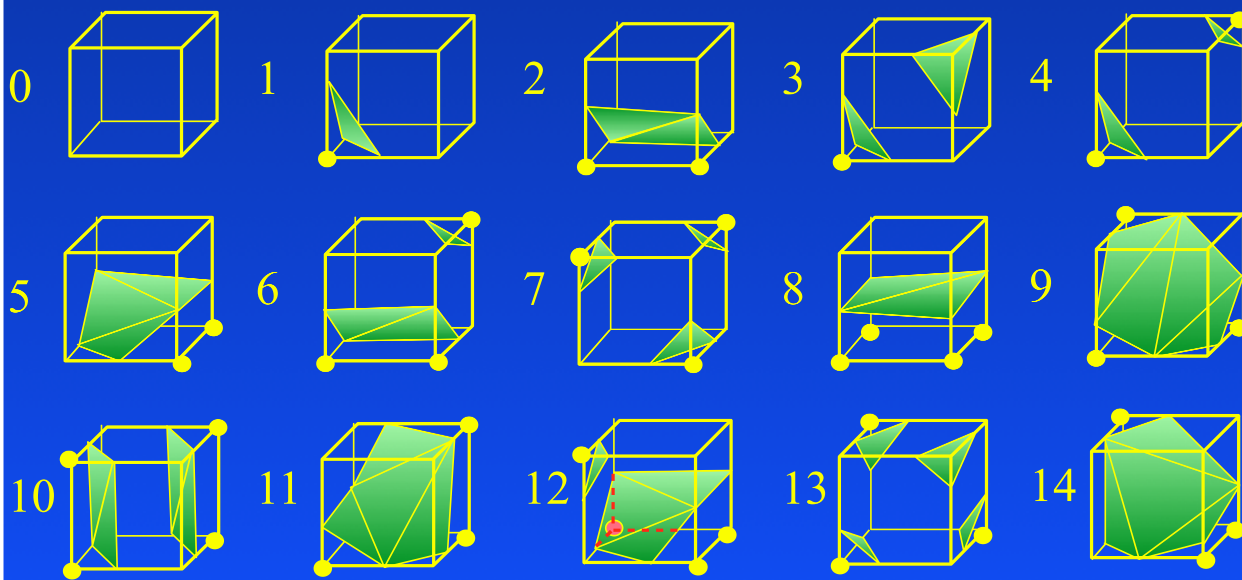


5. Triangulate and display all such polygons

Marching Cubes

Basic Cases

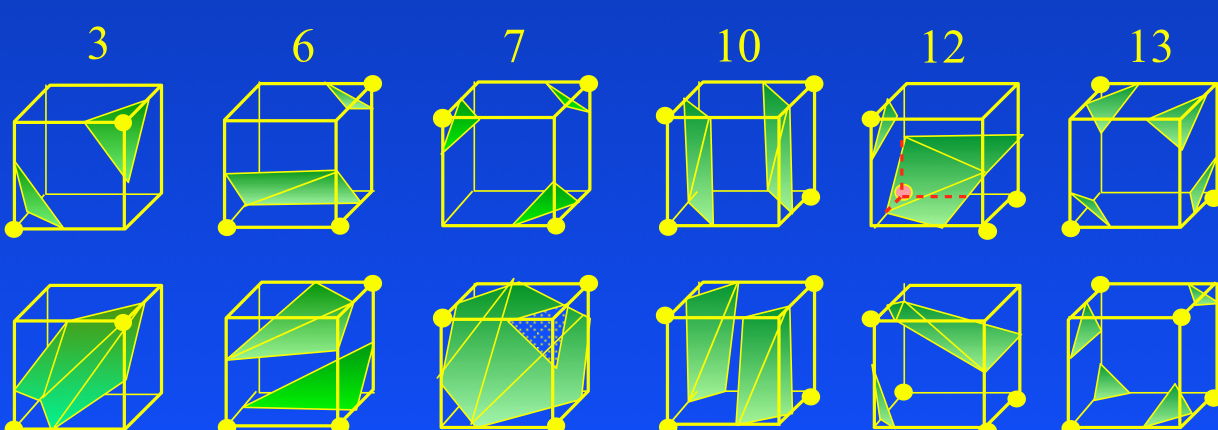
- There are only 15 truly different cases



Marching Cubes

Ambiguous Cases

- Cube face with adjacent different vertices and diagonally opposite same vertices — 6 cases
- Inconsistent neighbor choices yields holes



Computing Normals

- For each vertex, estimate a vector normal using forward differences:
  - dx[i, j, k] = x[i+1, j, k]-x[i, j, k]
  - dy[i, j, k] = y[i, j+1, k]-y[i, j, k]
  - dz[i, j, k] = z[i, j, k+1]-z[i, j, k]

Other Isosurface Algorithms

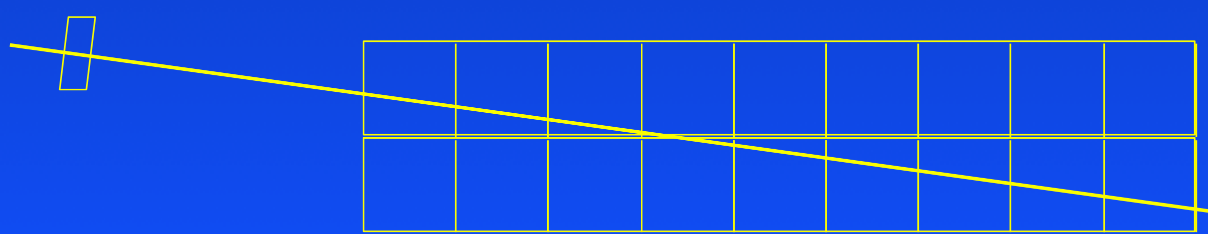
- Dividing cubes - never generate triangles
  - if cube contains isosurface, project it to screen

    if projection is smaller than a pixel, render it
    else subdivide cube and recurse
- Marching tetrahedra
  - Divide each cube into 5 tetrahedra
  - Surface can only pass through a tetrahedron in 2 unique ways: both of which yield 1 triangle
  - 5 times as many objects, but each is much simpler

Direct Volume Rendering

Ray Casting

- Figure out how each pixel is built from data values
- Ray intersects each voxel
  – value inside voxel is a *density*
  – distance ray travels through voxel determines *opacity* that is added to pixel's opacity value
  – if pixel opacity reaches 1, ray traversal terminates



Direct Volume Rendering

Splatting

- Figure out how each data value contributes to each pixel
- Treat each voxel as a solid object, project its faces onto the display area (from back to front)
  - the color and opacity of the projected polygons are determined from the voxel's values
  - the projected polygons are *composited* according to their depths, opacities and colors