Dancing Proteins: 3-D Visualization of Protein Structure and Dynamics on Next-Generation Graphics Hardware

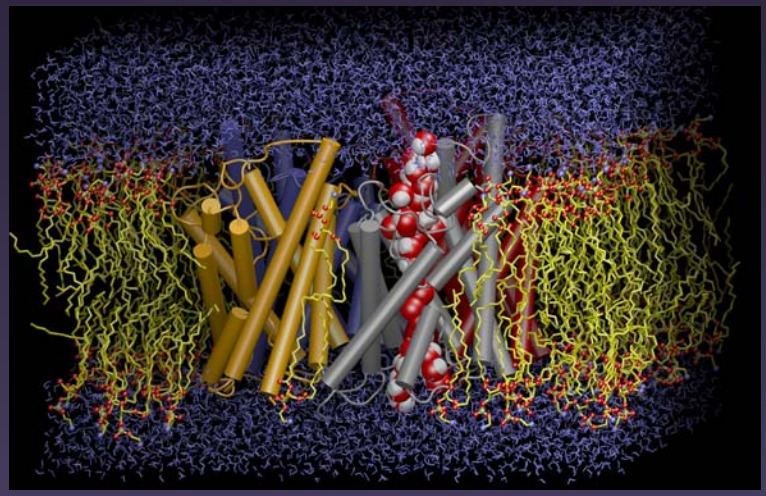
John E. Stone

Beckman Institute

University of Illinois

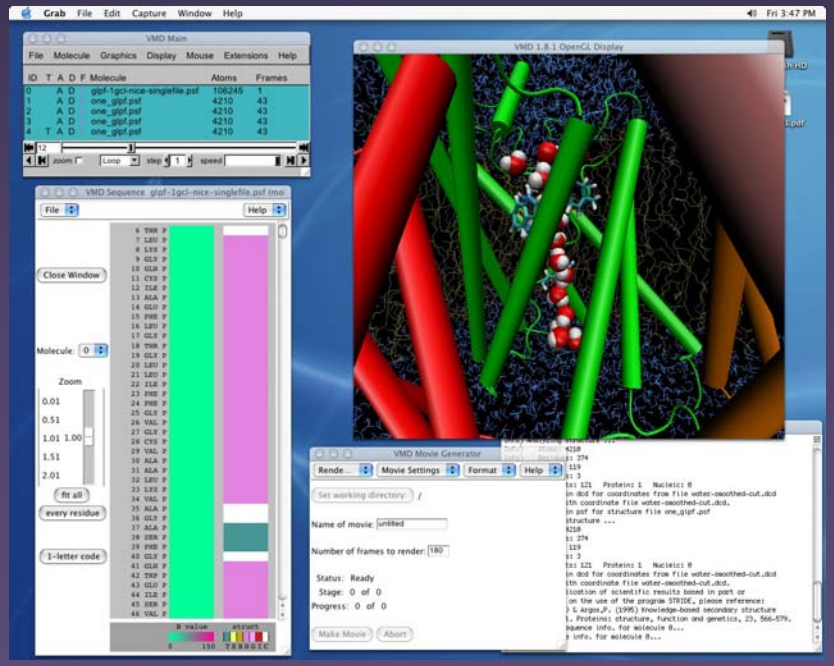
Overview

- Intro to biomolecular visualization with VMD
- Brief VMD development history
- VMD on present-day graphics accelerators
- Next-generation graphics accelerators and future opportunities



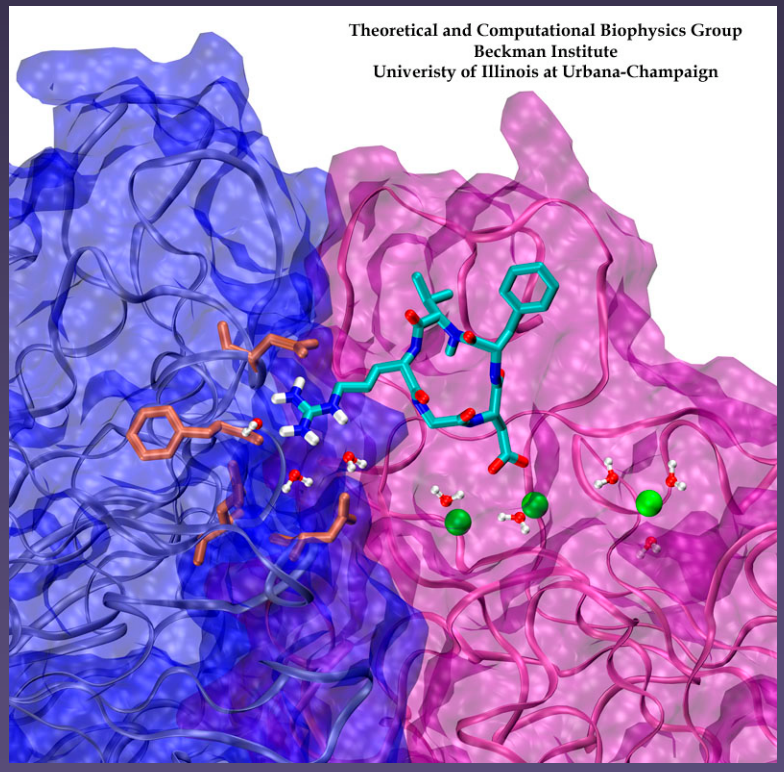
VMD

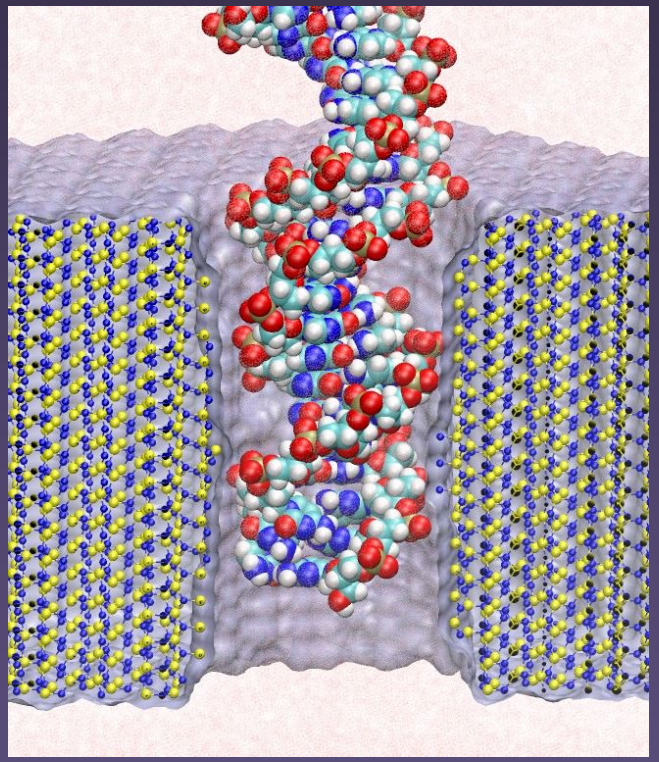
- VMD – “Visual Molecular Dynamics”
- Primarily for visualizing molecular dynamics simulations of proteins
- Developed by the Theoretical and Computational Biophysics Group @ UIUC
- Over 19,000 registered users
- http://www.ks.uiuc.edu/Research/vmd/



Biomolecular Visualization

- View proteins, DNA, nanodevices
- Explore structure and function of biological molecules





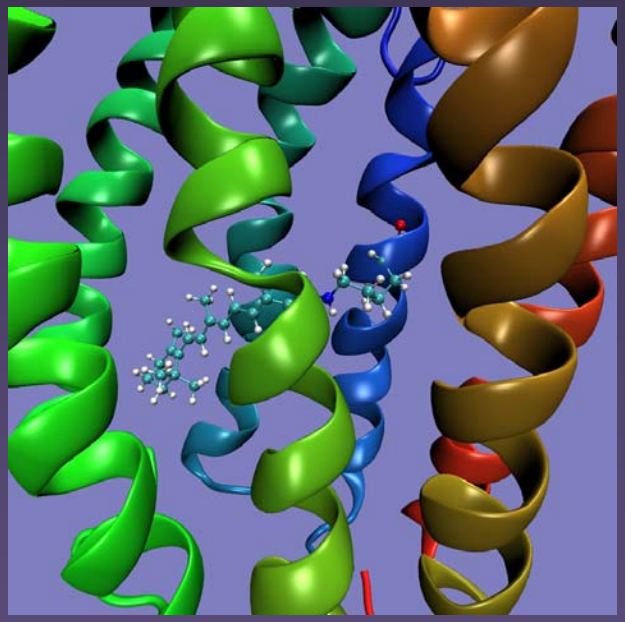
National Center for Research Resources

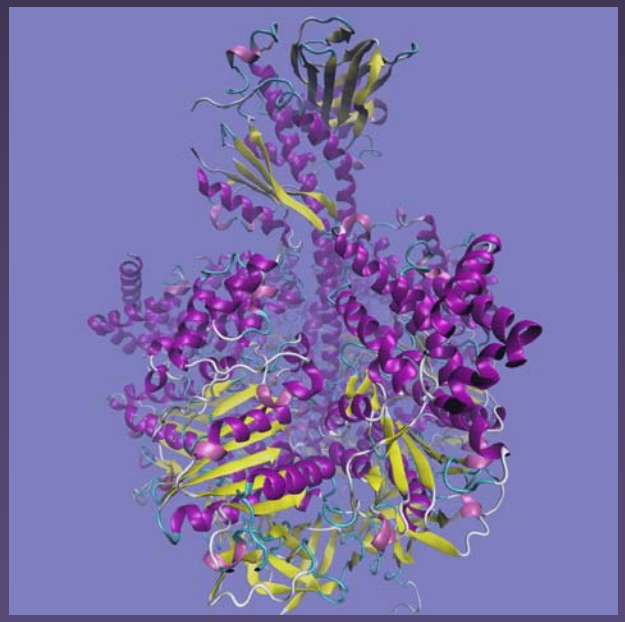
NIH Resource for Macromolecular Modeling and Bioinformatics
http://www.ks.uiuc.edu/

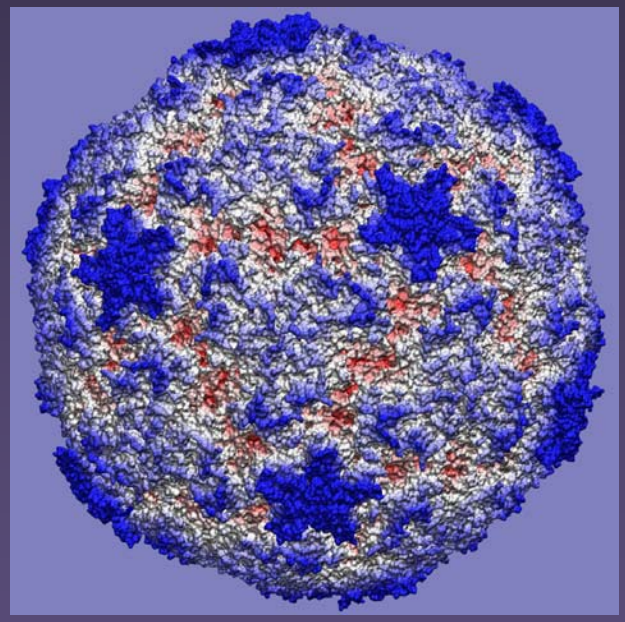
Beckman Institute, UIUC

Structure Visualization

- Draw atomic structure, protein backbone, secondary structure, solvent-accessible surface
- Color by per-atom or per-residue info, position, time, electrostatic potential, or user-defined properties

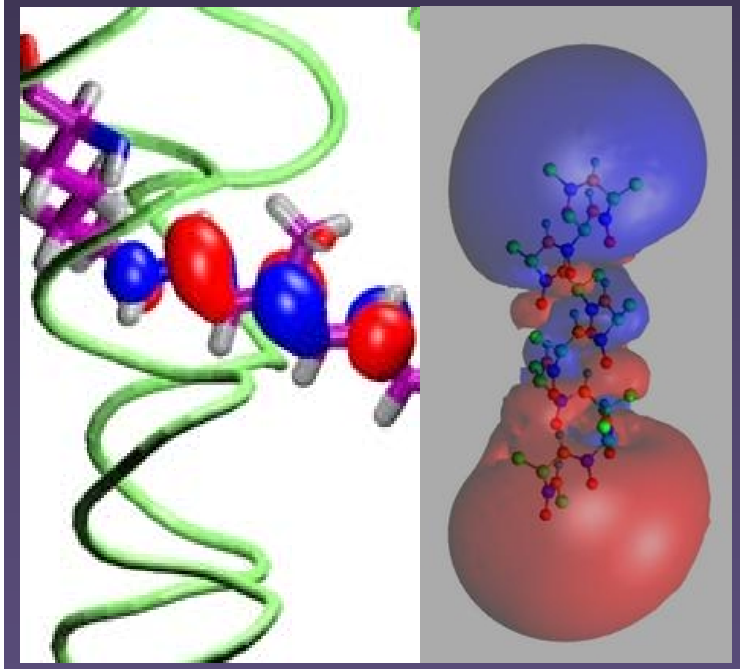


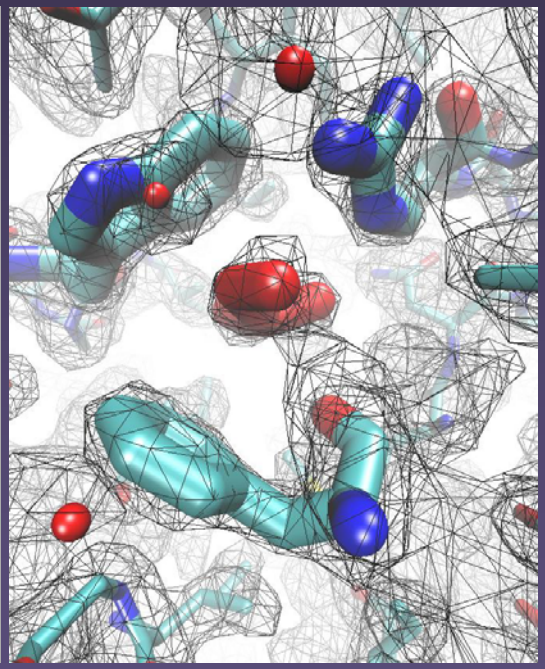


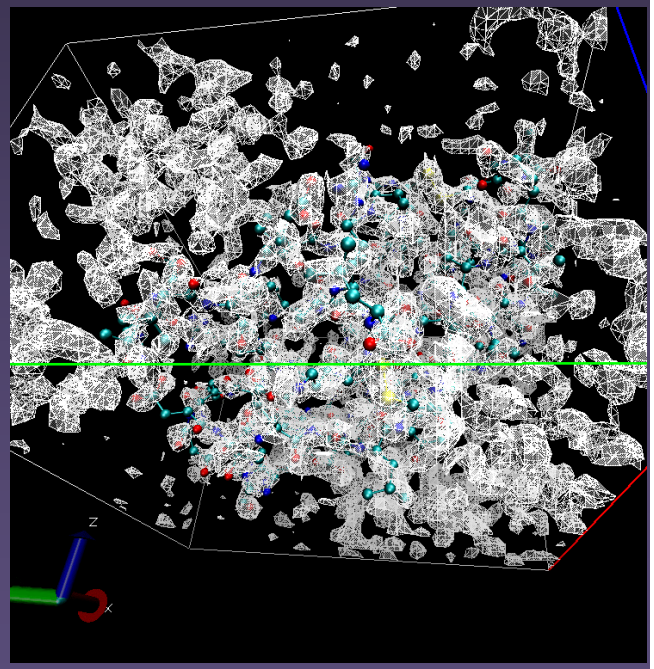


Visualizing Volumetric Data

- Display environment surrounding molecular structure, fields that affect structure and function
- Electron orbitals, electron density, electrostatic potential, temporal occupancy maps

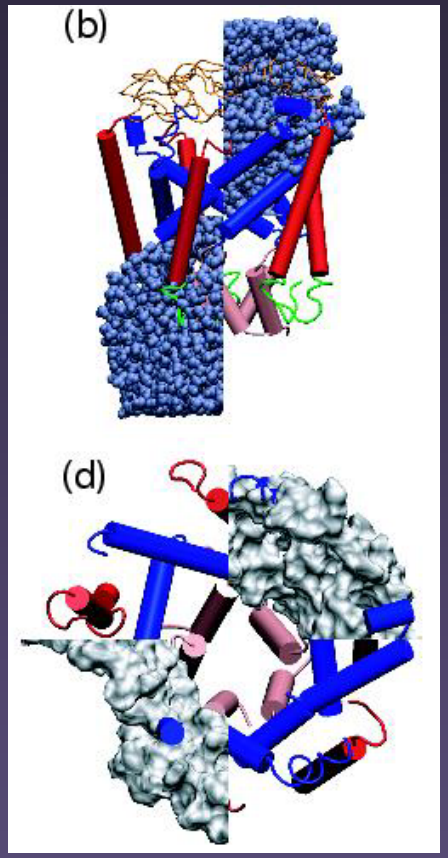


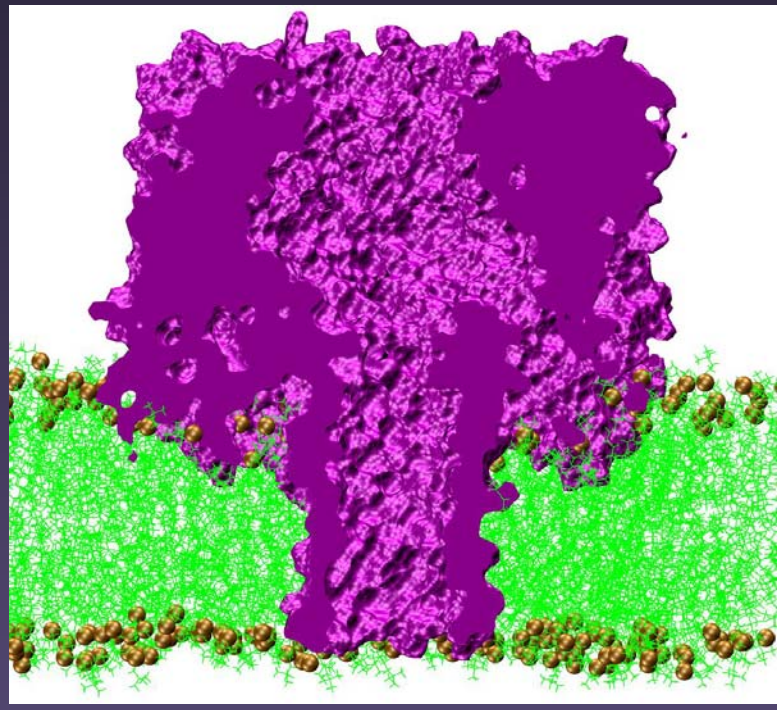




Multiple Representations, Cut-away Views

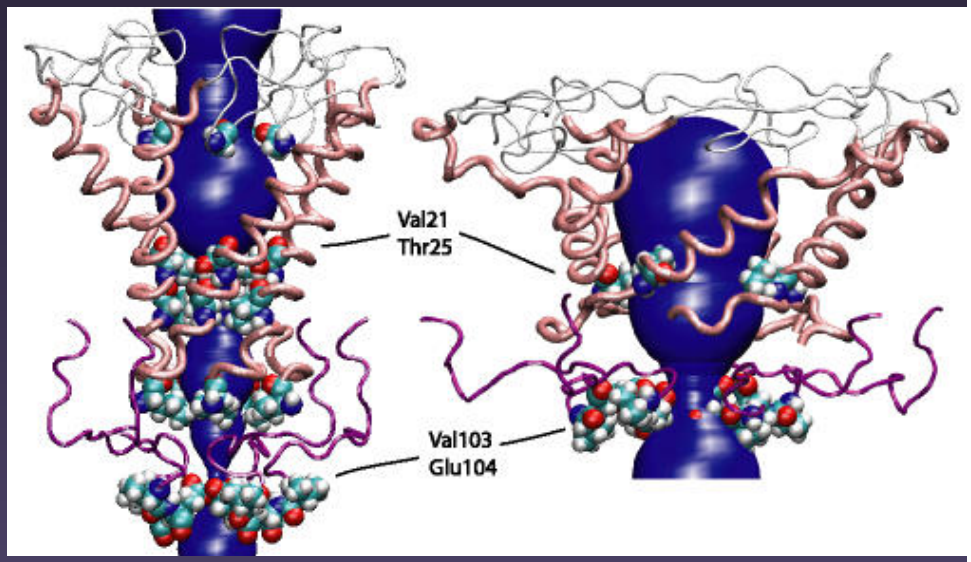
- Multiple reps are often used concurrently
  - Show selected regions in full atomic detail
  - Simplified cartoon-like or schematic form
- Clipping planes can slice away structure obscuring interesting features

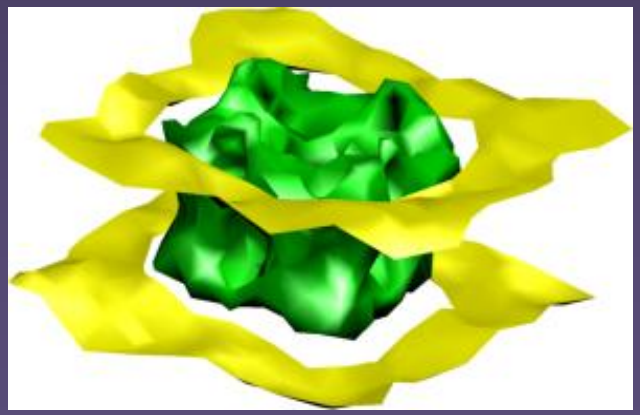


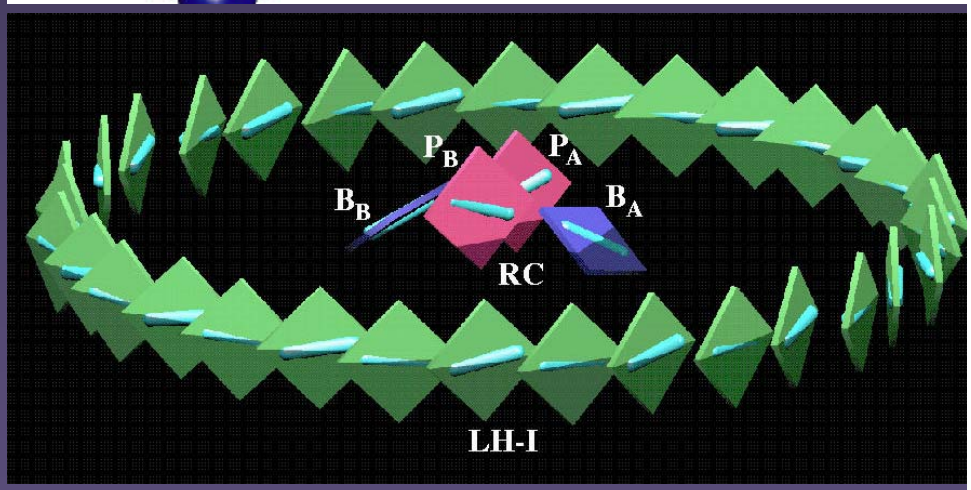


Other Representations

- Extract and render pores, cavities, indentations
- Simplified representations of large structures

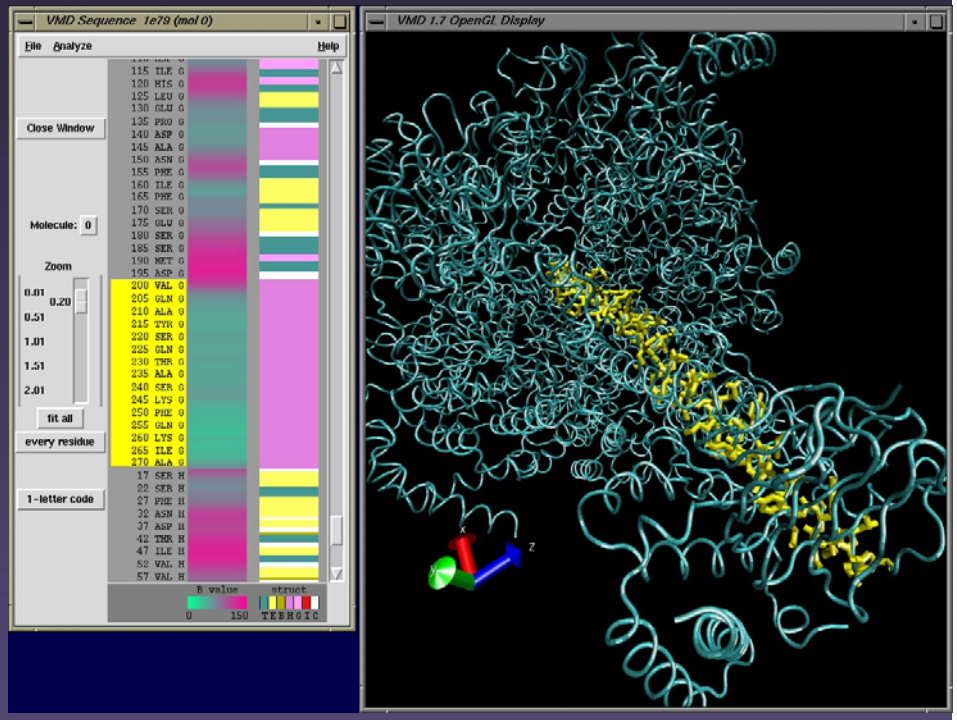






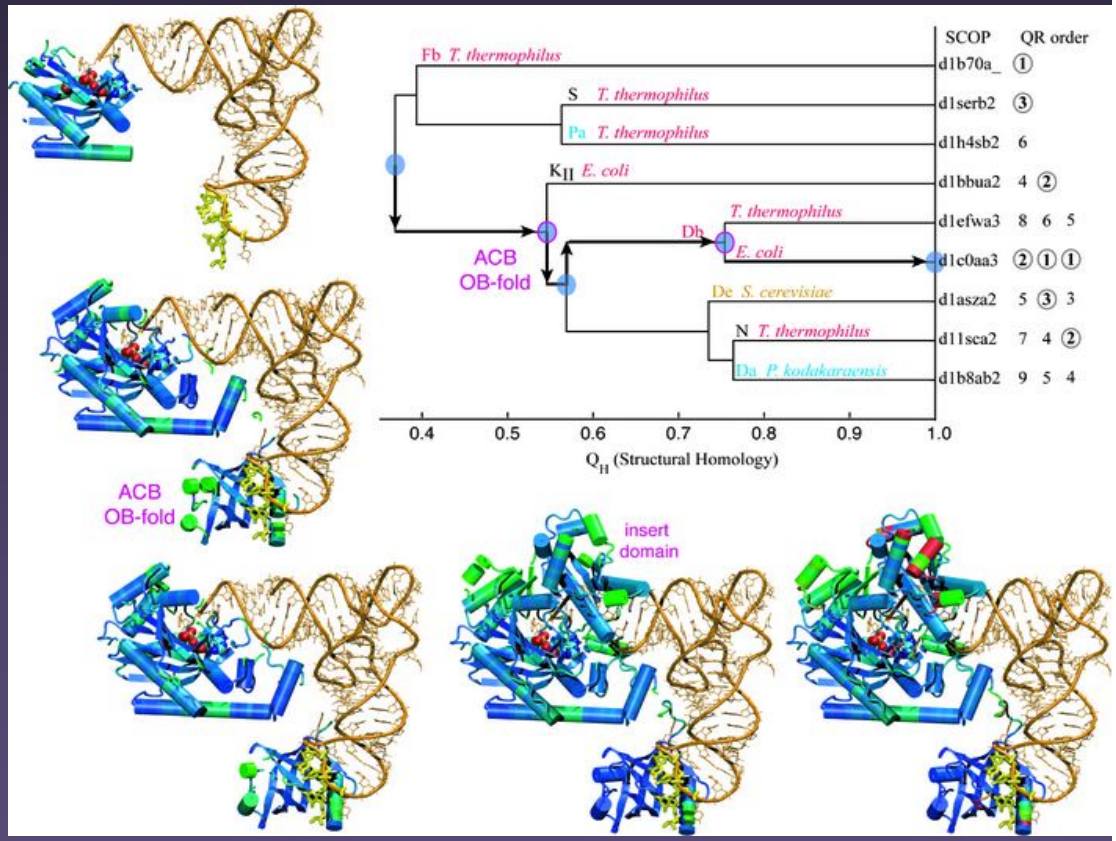
Integrating Protein Sequence Data

- View sequence and structure information side-by-side
- Highlight selected residues
- Color sequence chart by secondary structure and temperature factor



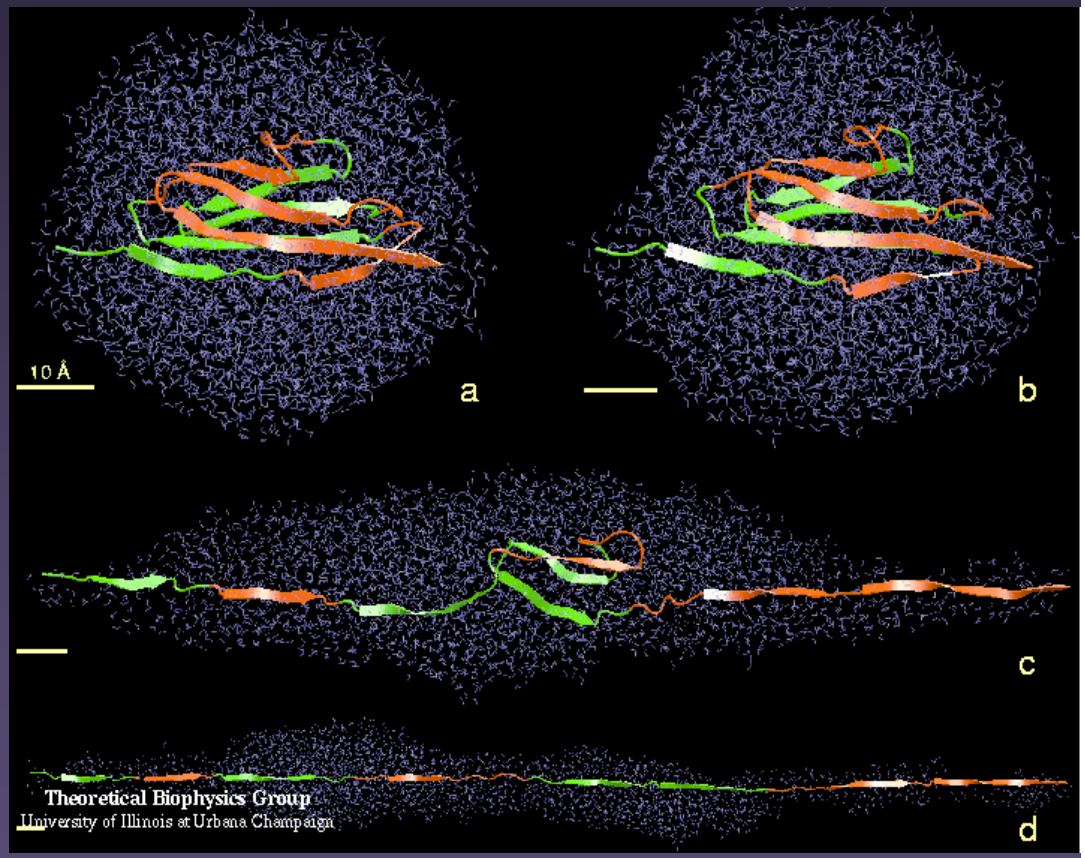
Multiple Structure Alignment

- Study evolutionary changes in sequence and structure of proteins
- Align and superimpose multiple structures
- Color by structural conservation
- Color by sequence conservation
- Display phylogenetic tree, cluster biological form by similarity



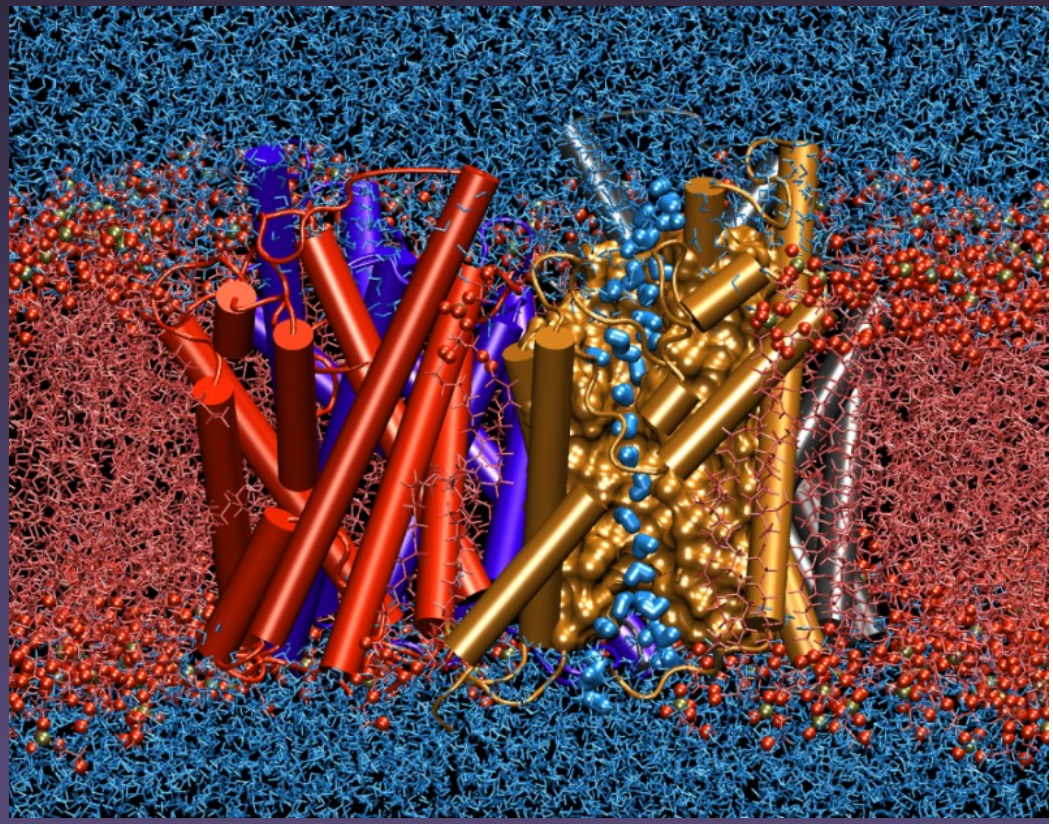
Molecular Dynamics

- Classical mechanical simulation of atomic motions (F=ma)
- Molecular dynamics calculations save trajectories of atomic coordinates as the simulation progresses
- Researchers study trajectories by analyzing force profiles, energies, structural changes etc



Structure Building, Simulation Preparation

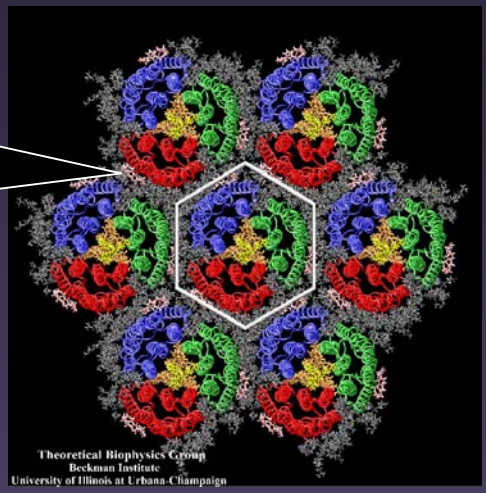
- Obtain protein structure
- Setup simulated environment:
  - Add water
  - Add ions
  - Add membrane
- Display and analyze prepared structure

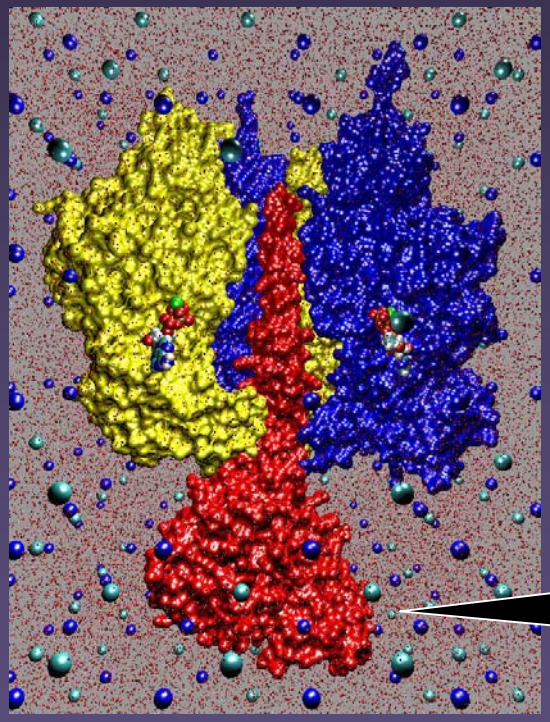


Large Scale Molecular Visualization

- Large structures: 300,000 atoms and up
- Complex representations
- Long trajectories: thousands of timesteps
- A 10 ns simulation of 100K atoms produces a 12GB trajectory
- Multi-gigabyte data sets break 32-bit addressing barriers

Purple Membrane 150,000 Atoms

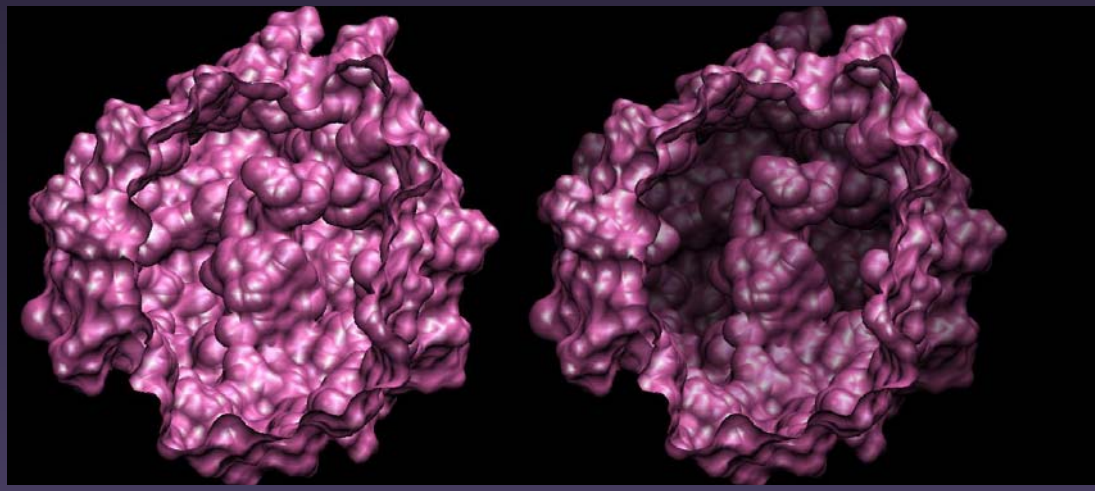


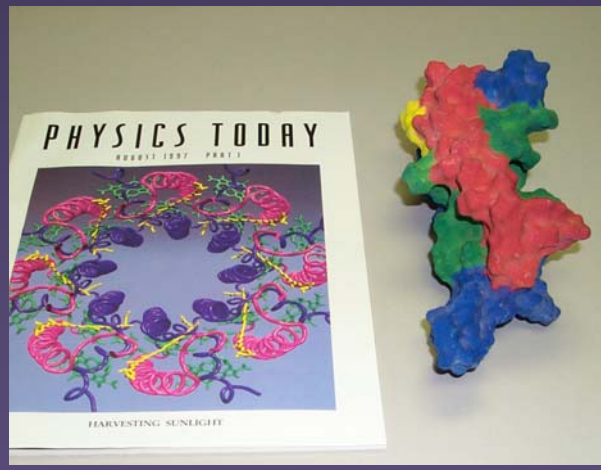


F1 ATPase 327,000 Atoms

3-D Depth and Shape Perception

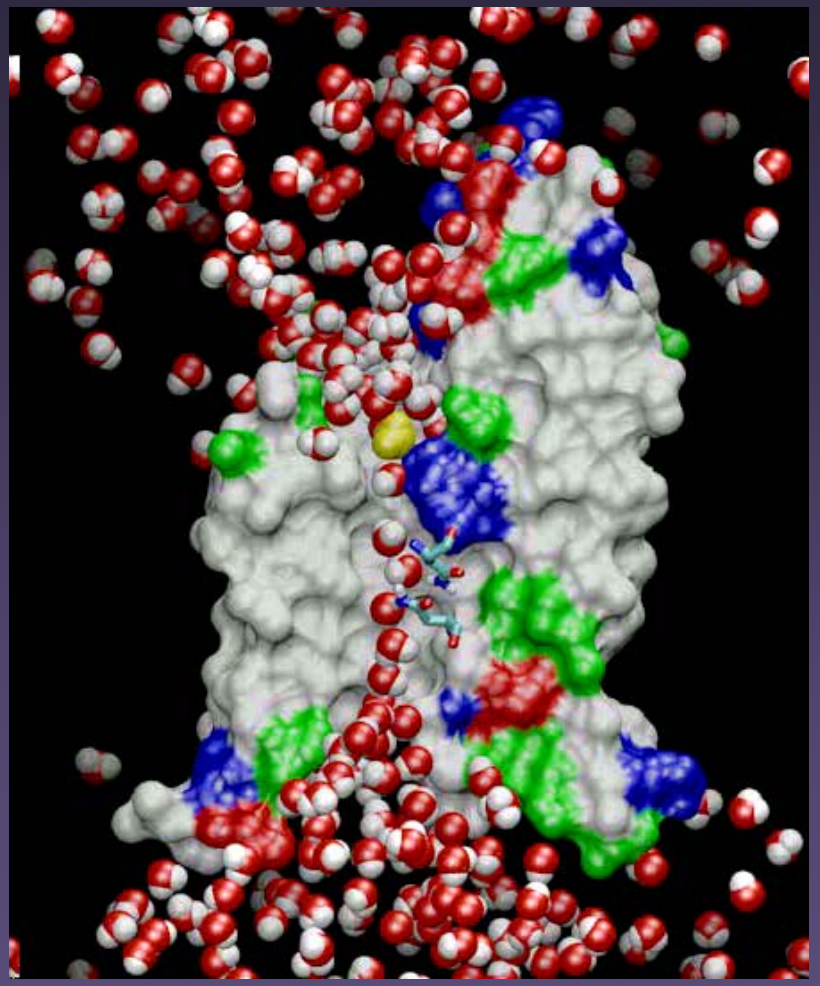
- Perspective projection
- Depth cueing:
  - Great when orthographic (parallel) projection is required
- Stereoscopic rendering:
  - Twice the work (half the speed)
  - More expensive graphics hardware
  - LCD panels are costly
- Solid Models:
  - 3-D printers are becoming more common





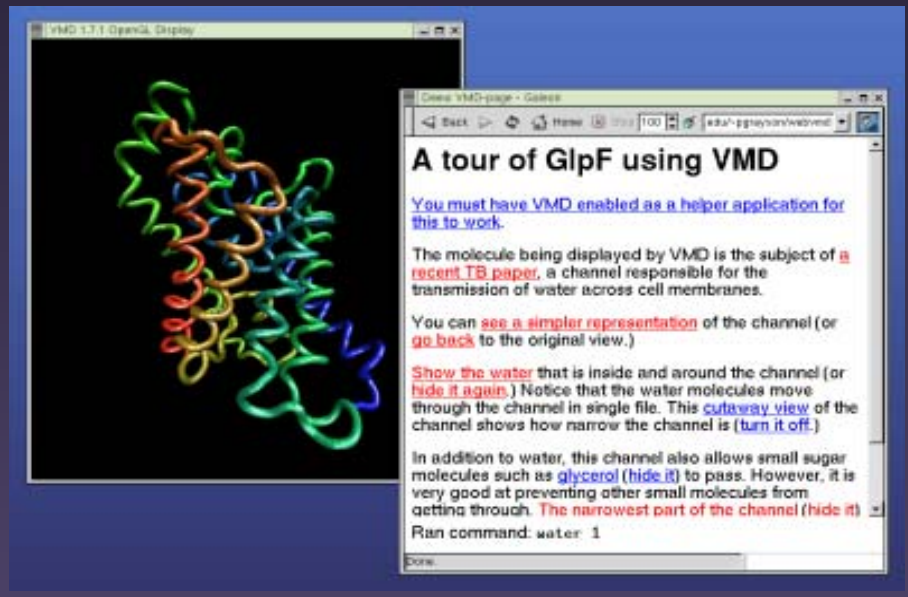
Easy-to-use Movie Making

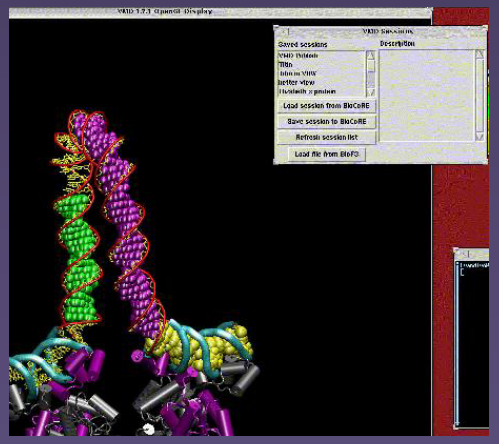
- Generate simple movies in a few minutes
- Uses readily available video compression tools
- Several movie types:
  - Rotation
  - Rocking
  - Trajectory animation
  - Trajectory rock
  - Development continues…



Web and Collaboration Features

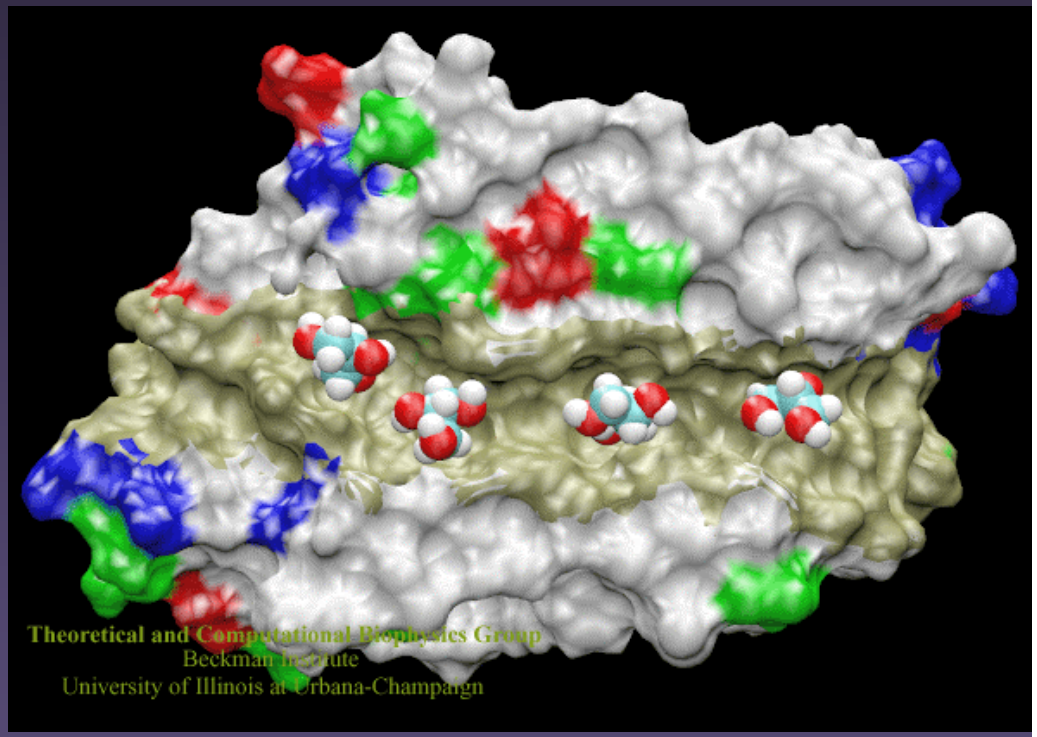
- BioCoRE
  - Publish VMD sessions to collaboratory
  - Load structures from BioFS
  - http://www.ks.uiuc.edu/Research/biocore/
- Web-based VMD scripting
  - Clickable links execute script commands





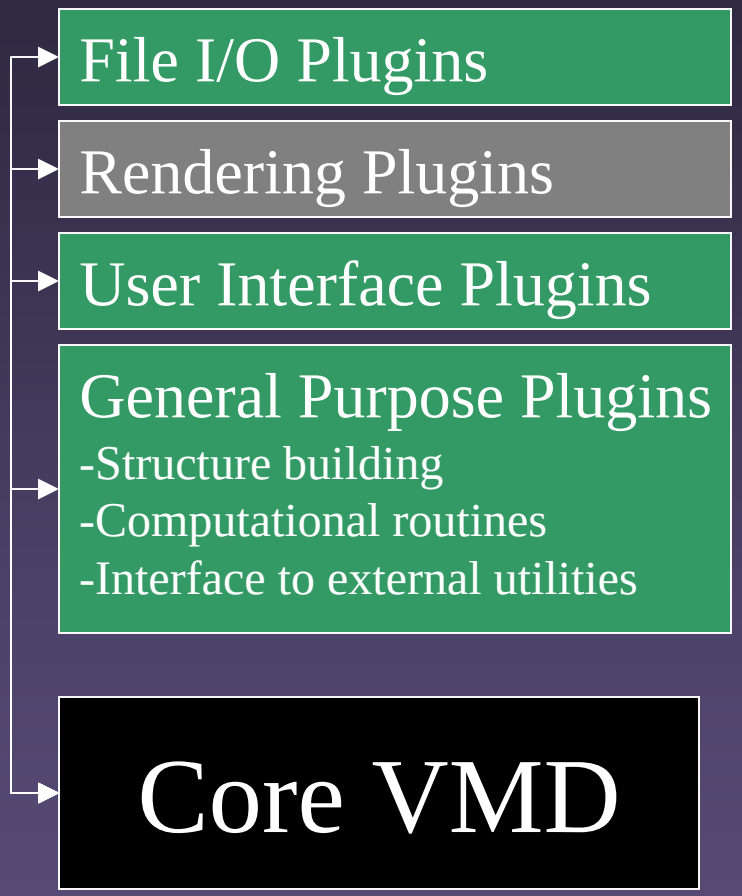
Scripting and Analysis

- Built-in Tcl and Python scripting interfaces
- Interactive command shell
- New commands and routines can be added by users
- Analysis scripts can also be run in batch mode without graphics



VMD "Plugin" Extensions

- Similar in concept to web browser plugins
- Written in C, C++, Python, or Tcl
- Can extend VMD command language
- Potential for higher performance than pure scripts
- Ease creation and distribution of 3rd party VMD extensions and improvements



VMD Technological History

1994 - Original version used Silicon Graphics IRIS GL, CAVE
1996 - Tcl scripting added for user-extensibility
1998 - Ported to OpenGL
- Running on all major Unix platforms
1999 - First stable 64-bit version (DEC Alpha)
2000 - Python scripting added
- Ported to Microsoft Windows
2001 - Advanced input devices: Haptic feedback, Spaceball
- Dynamic use of OpenGL extensions
2002 - Ported to MacOS X /w native OpenGL
- VMD I/O and scripting interfaces support plug-ins
2003 - Initial work towards use of programmable shading:
experiments with custom Sun microcode + drivers,
3DLabs first generation programmable graphics boards
2004 - OpenGL Shading Language becomes a standard
2005 - First release of VMD using programmable shading

Timeline: Graphics Hardware Used for Molecular Visualization

60's and 70's:

- Mainframe-based vector graphics on Tektronix terminals
- Evans & Sutherland graphics machines

80's:

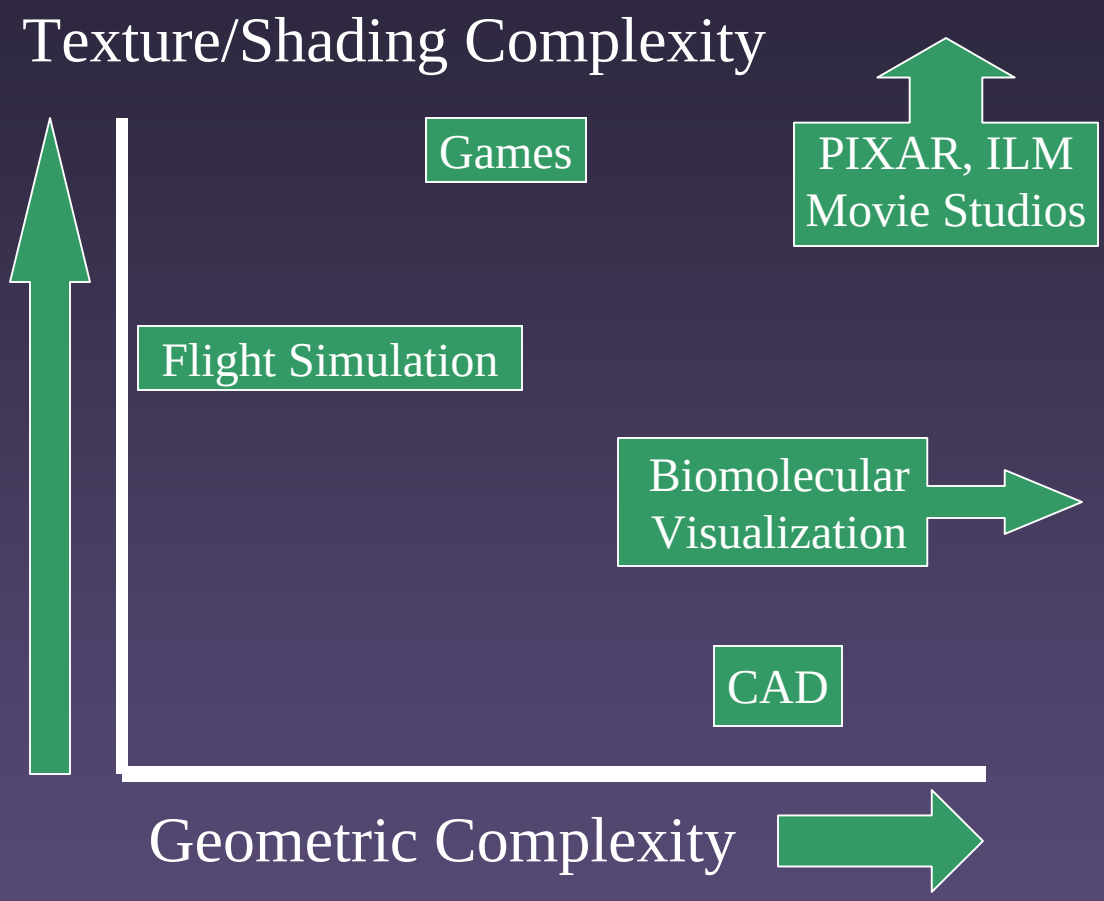
- Transition to raster graphics on Unix workstations, Mac, PC
- Space-filling molecular representations
- Stereoscopic rendering

90's - 2002:

- 3rd-generation raster graphics systems
- Depth-cueing
- Texture mapping: coloring by potential, density, etc
- Full-scene antialiasing

Comparison of Molecular Visualization with Other Applications

- Geometric complexity limits molecular visualization performance
- All atoms move every simulation timestep, thwarts many simplification techniques
- Commodity graphics hardware is tuned for requirements of games
- Solution: Use sophisticated shading instead of geometry where possible

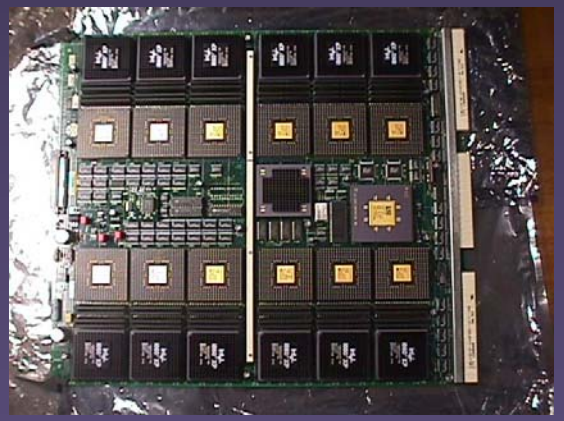


Programmable Graphics Hardware

Groundbreaking research systems:

- AT&T Pixel Machine (1989):
  - 82 x DSP32 processors
- UNC PixelFlow (1992-98):
  - 64 x (PA-8000 +
    - 8,192 bit-serial SIMD)
- SGI RealityEngine (1990s):
  - Up to 12 i860-XP processors perform vertex operations (*u*code), fixed-func fragment hardware
- Most graphics boards now incorporate programmable processors at some level

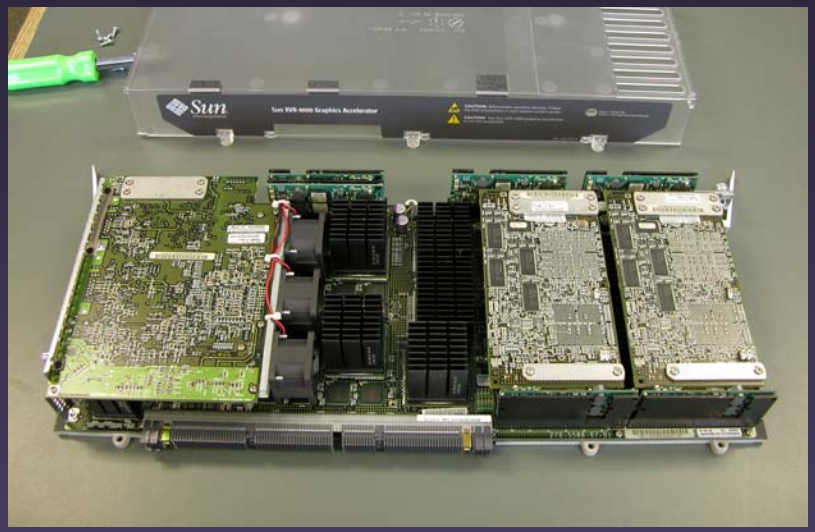
UNC PixelFlow Rack



Reality Engine Vertex Processors

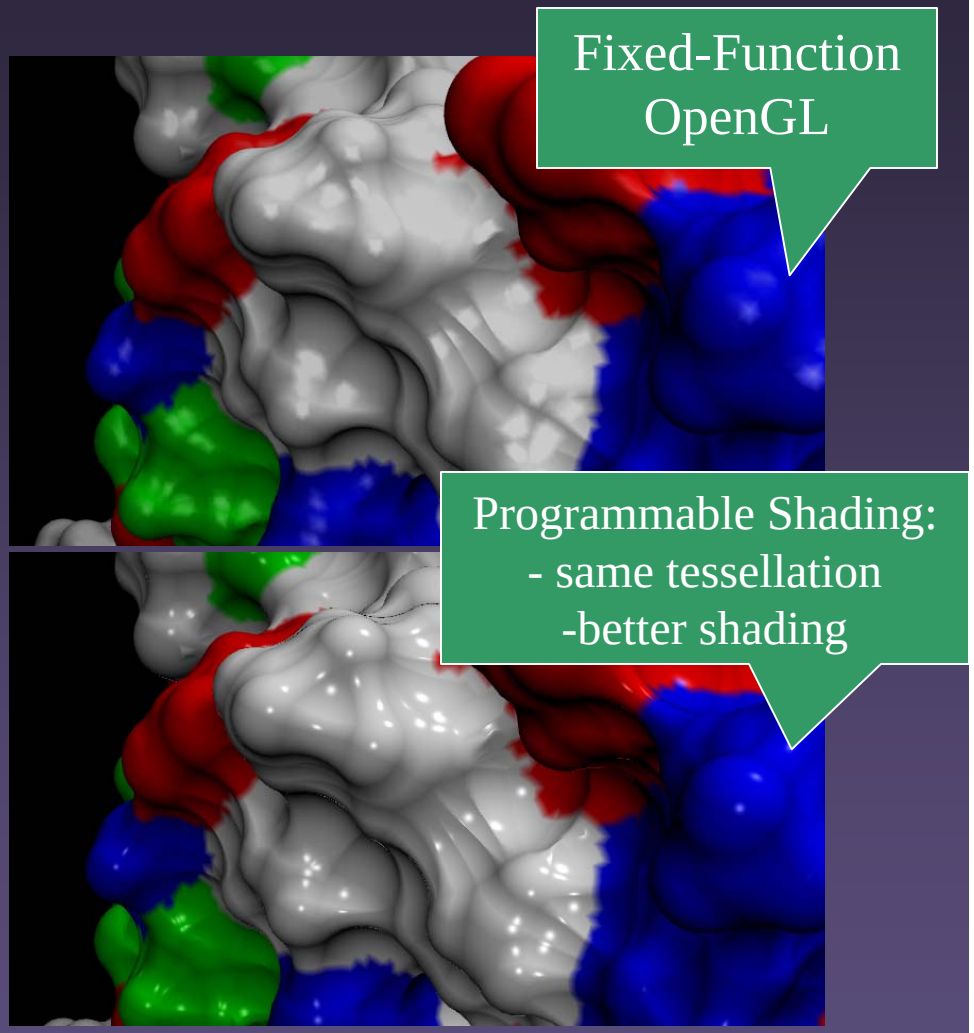
First Experiments with Programmable Graphics Hardware in VMD

- Sun XVR-4000
  - 4xMAJC-5200 CPUs
  - 1GB Texture RAM
  - 32MB *u*code RAM
  - 1 Teraflop Antialiasing Filter Pipeline
- Custom *u*code and OpenGL extension for rendering spheres
  - Draw only half-spheres, with solid side facing the viewer
  - 1-sided lighting
  - Host CPU only sends arrays of radii, positions, colors
  - fast DMA engines copy arrays from system memory to GPU
  - Overall performance twice as fast, host CPU load significantly decreased



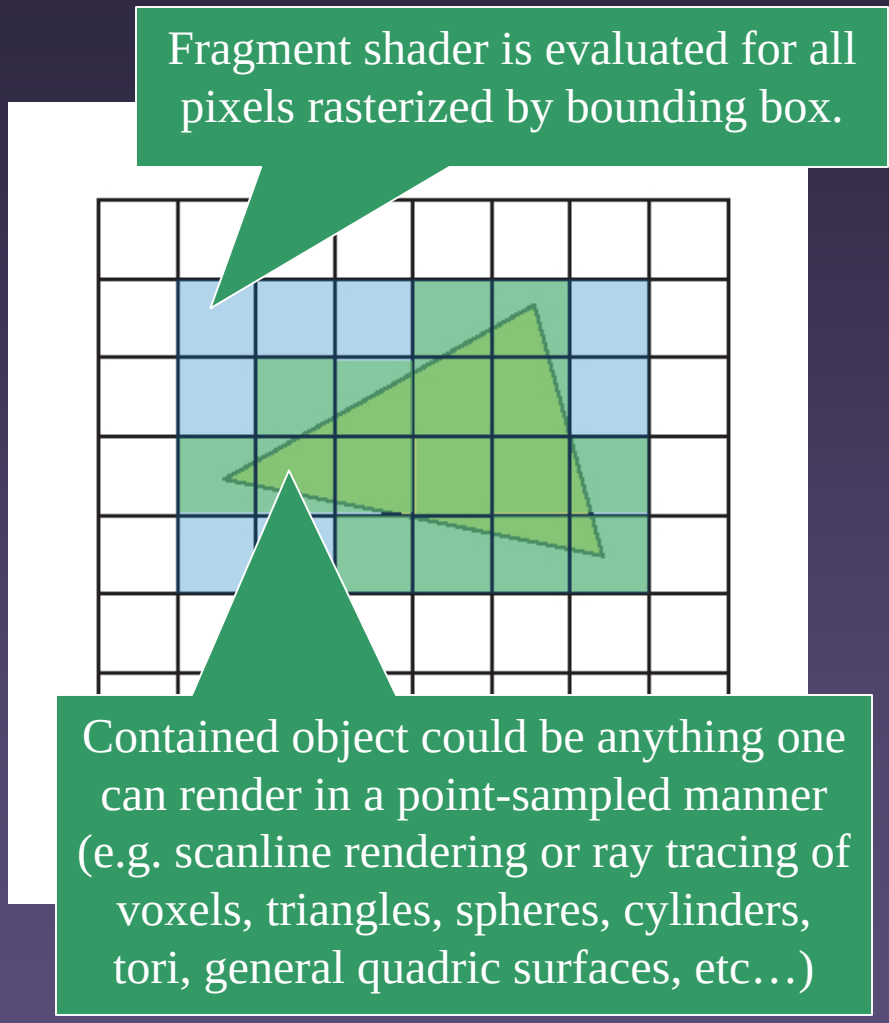
Benefits of Programmable Shading for Molecular Visualization

- Potential for superior image quality with better shading algorithms
- Direct rendering of curved surfaces
- Render density map data, solvent surfaces
- Offload work from host CPU to GPU



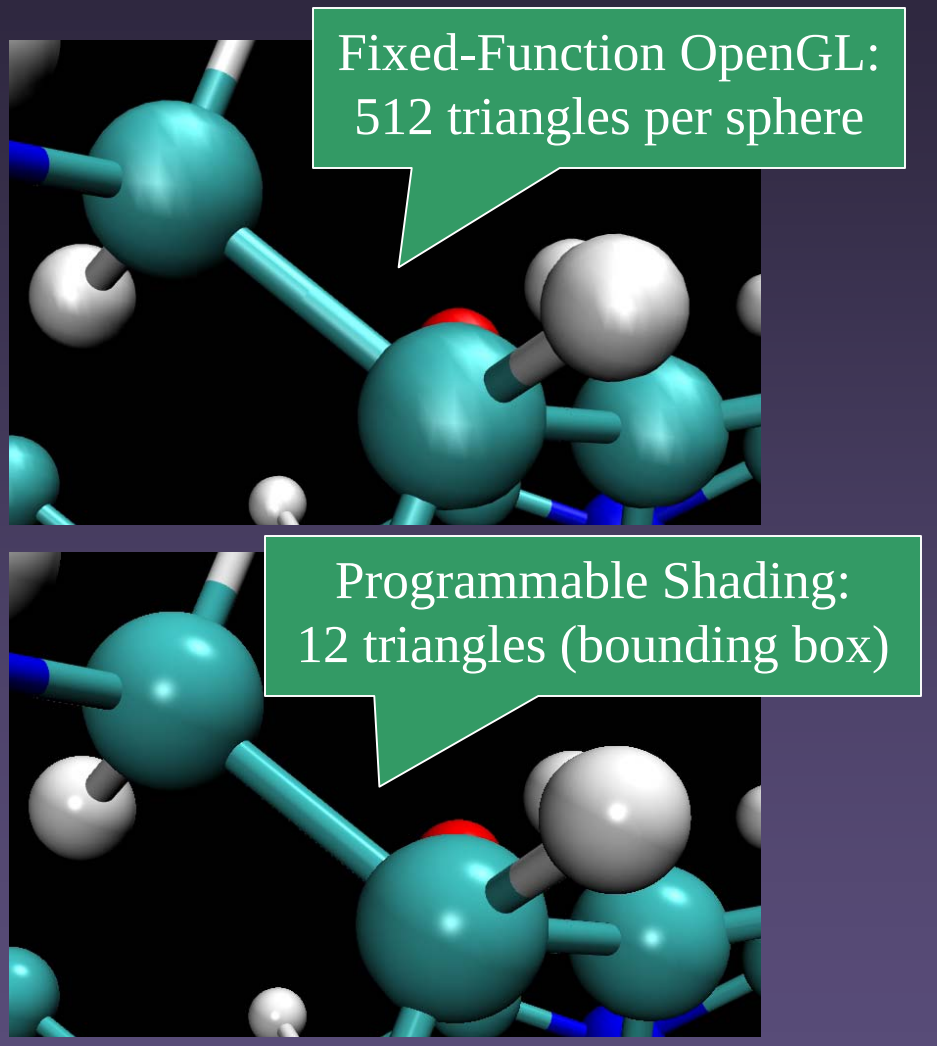
Rendering Non-polygonal Data with Present-day Programmable Shading

- Must express algorithms in terms of vertex/fragment shading model available in current hardware
- Render by drawing bounding box containing shape/data
- Vertex shader sets up state needed for fragment shader
- Fragment shader performs all the work

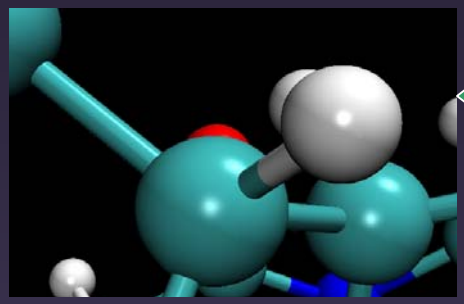


Ray Traced Sphere Rendering with Programmable Shading

- Fixed-function OpenGL requires curved surfaces to be tessellated with triangles
- Fine tessellation required for good results with Gouraud shading; performance suffers
- Static tessellations look bad when one zooms in
- Dynamic tessellation too costly when animating huge molecules
- Solution:
  - Ray trace spheres in fragment shader
  - GPU does all the work
  - Spheres look good at all zoom levels
  - Rendering time is proportional to pixel area covered by sphere



Sphere Fragment Shader



- Written in OpenGL Shading Language
- High-level C-like language with vector types and operations
- Compiled dynamically by the graphics driver at *runtime*
- Compiled machine code executes on GPU

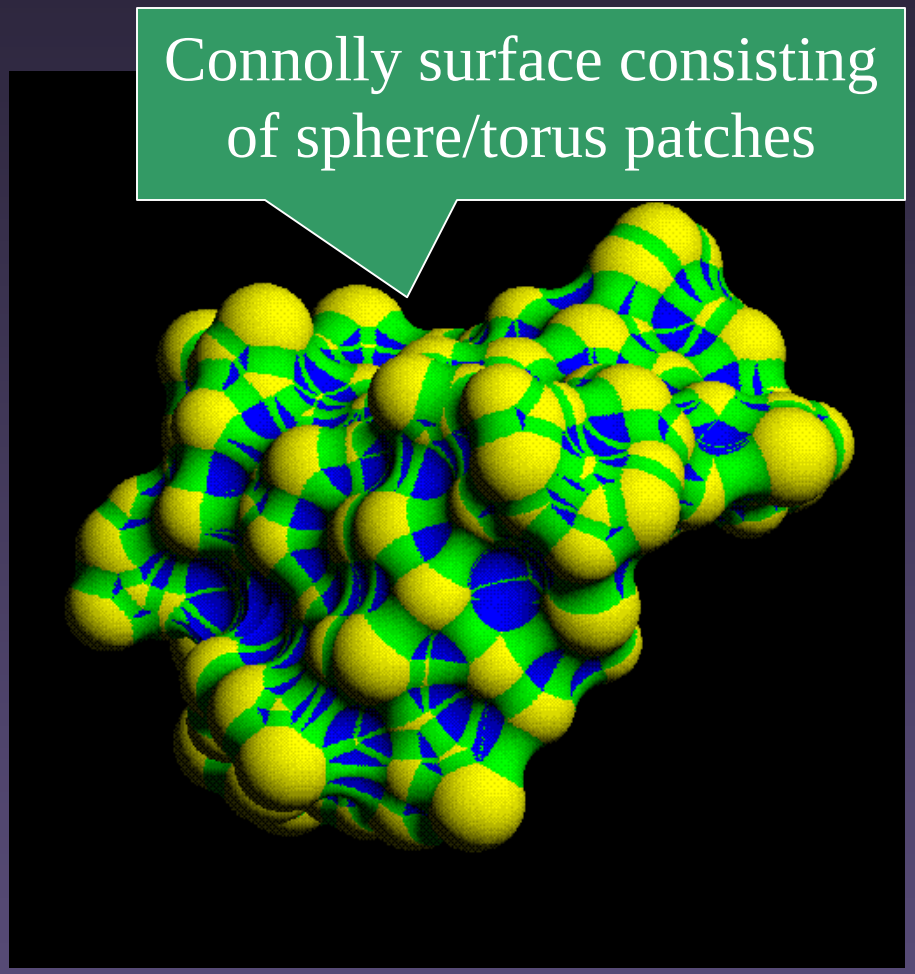
```
//
// VMD Sphere Fragment Shader (not for normal geometry)
//
void main(void) {
  vec3 raydir = normalize(V);
  vec3 spheredir = spherepos - rayorigin;

  // Perform ray-sphere intersection tests based on the code in Tachyon
  float b = dot(raydir, spheredir);
  float temp = dot(spheredir, spheredir);
  float disc = b*b + sphereradsq - temp;

  // only calculate the nearest intersection, for speed
  if (disc <= 0.0)
    discard; // ray missed sphere entirely, discard fragment

  // calculate closest intersection
  float tnear = b - sqrt(disc);

  if (tnear < 0.0)
    discard;

  // calculate hit point and resulting surface normal
  vec3 pnt = rayorigin + tnear * raydir;
  vec3 N = normalize(pnt - spherepos);

  // Output the ray-sphere intersection point as the fragment depth
  // rather than the depth of the bounding box polygons.
  // The eye coordinate Z value must be transformed to normalized device
  // coordinates before being assigned as the final fragment depth.
  if (vmdprojectionmode == 1) {
    // perspective projection = 0.5 + (hfpn + (f * n / pnt.z)) / diff
    gl_FragDepth = 0.5 + (vmdprojparms[2] + (vmdprojparms[1] * vmdprojparms[0
3];
  } else {
    // orthographic projection = 0.5 + (-hfpn - pnt.z) / diff
    gl_FragDepth = 0.5 + (-vmdprojparms[2] - pnt.z) / vmdprojparms[3];
  }

#ifdef TEXTURE
  // perform texturing operations for volumetric data
  // The only texturing mode that applies to the sphere shader
```

Efficient 3-D Texturing of Large Datasets

- Non-power-of-two 3-D texture map dimensions
  - Reduce texture size by a factor of 8 for worst-case scenario (e.g. 2^N-1 dimensions on potential map)
- Perform volumetric color transfer functions on GPU rather than on the host CPU
  - use scalar rather than RGB texture map, reduce texture size
  - perform all range clamping and mapping on GPU
  - update color transfer function without re-downloading large texture maps (e.g. 256^3 potential maps, etc)

The Wheel of Reincarnation: Revival of Old Rendering Techniques?

- Graphics hardware is making another trip around Myer and Sutherland's wheel (CACM '68)
- Visualization techniques that weren't triangle-friendly lost favor in the 90's may return
- Some algorithms that mapped poorly to the OpenGL pipeline are trivial to implement with programmable shading
- Non-polygonal methods get their first shot at running on graphics accelerator hardware rather than the host CPU
  - increased parallelism
  - higher memory bandwidth



Future Possibilities with More Flexible / Powerful GPUs

- Atomic representation tessellation and spline calculations done entirely on GPU?
- Direct rendering of isosurfaces from volumetric data via ray casting (e.g. electron density surfaces)
- Direct rendering of metaball (“Blob”) approximation of molecular surfaces via ray casting:
  - May allow interactive animation of large surface representations not possible with other methods (Connolly, Surf, MSMS, AlphaShapes, Marching cubes, etc..)

Next-Gen Hardware

- Increased parallelism in GPUs
  - Fragment processors
  - Multiple boards (NVIDIA SLI)
- IBM / Sony Cell chip
  - General purpose stream processor
  - 256 GFlop/sec single-precision FP
  - 30 GFlop/sec double-precision FP
- Further OpenGL extensions
- Continued hardware evolution:
  - Additional programmable pipeline stages