3D Počítačová grafika

Matej Lexa
IV121
Informatika pro biology

3D Počítačová Grafika (nebo Geometrie)

- modelování scén

SDL (scene description language)

- vizualizace scén (rendering)

- rasterizace
- „raytracing“

- zajímavé koncepty

CSG (constructive solid geometry)

skriptování scén

příklad generování realistických stromů a keřů

SDL – Scene Description Languages

VRML/X3D

3DMLW

POV-Ray SDL

Renderman shading language

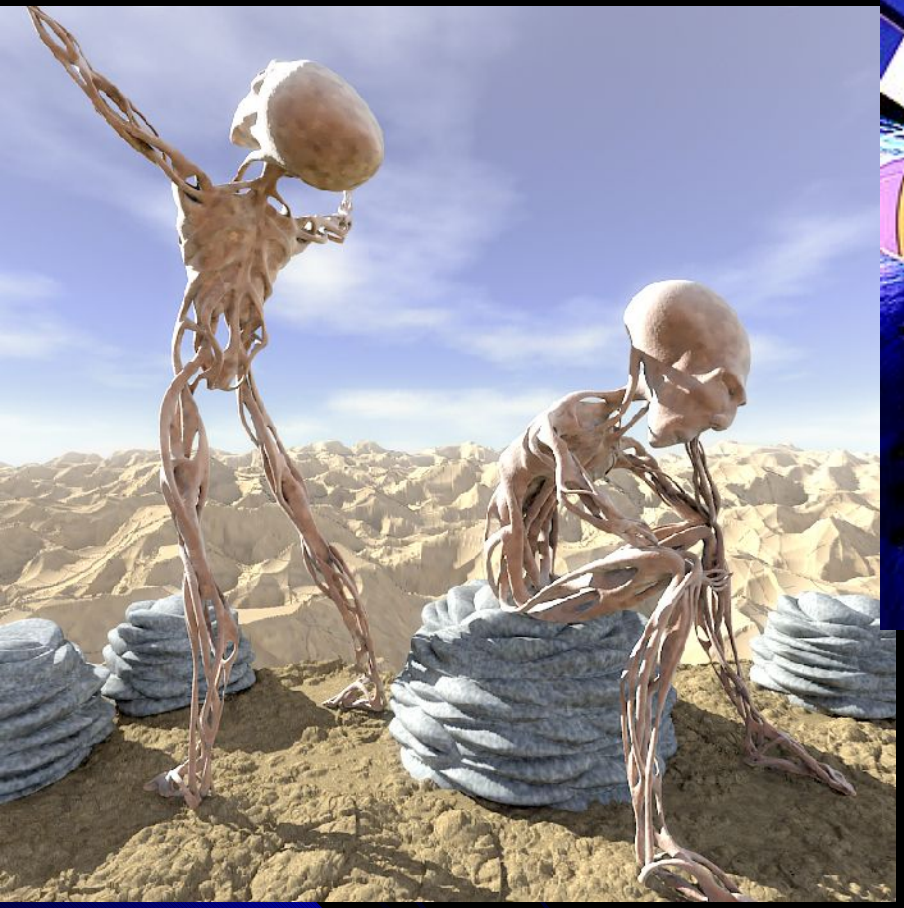
http://en.wikipedia.org/wiki/Scene_description_language

RenderMan Shading Language

Daniel Scherzer

Vienna University of Technology

Co je RenderMan?



Entropy image contest winner by Claude Schitter

MonstersInc by Pixar

A Bug‘s Life by Pixar

Co je RenderMan?

Autorem je společnost Pixar (1987)

Něco jako PostScript pro 3D

- Scene Description Language
  - není modelovacím programem
  - není renderingovým programem
  - Rozhraním mezi modelováním a renderingem

Příklad Bytestream kódu pro RenderMan Interface

```
Display "RenderMan" "framebuffer"
  "rgb"
Format 256 192 1
WorldBegin
Surface "constant"
Polygon "P" [0.5 0.5 0.5 0.5 -0.5
  0.5 -0.5 -0.5 0.5 -0.5 0.5 0.5]
WorldEnd
```

RIB

```
Display "RenderMan" "framebuffer"
  "rgb"
Format 256 192 1
WorldBegin
Surface "constant"
Polygon "P" [0.5 0.5 0.5 0.5 -0.5
  0.5 -0.5 -0.5 0.5 -0.5 0.5 0.5]
WorldEnd
```

API

```
#include <ri.h>
RtPoint Square[4] = { {.5,.5,.5}, {.5,-.5,.5}, {-.5,-.5,.5},
   {-.5,.5,.5} };

main(void){
    RiBegin(RI_NULL);   /* Start the renderer */
        RiDisplay("RenderMan", RI_FRAMEBUFFER, "rgb", RI_NULL);
        RiFormat((RtInt) 256, (RtInt) 192, 1.0);
        RiWorldBegin();
            RiSurface("constant", RI_NULL);
            RiPolygon( (RtInt) 4,      /* Declare the square */
                RI_P, (RtPointer) Square, RI_NULL);
        RiWorldEnd();
    RiEnd();                /* Clean up */
```

RenderMan Shading Language

```
surface clouds(float vfreq = .8 )
{
    float sum ;
    float i;
    color white = color(1.0, 1.0, 1.0);
    point Psh = transform("shader", P);

    sum = 0;
    freq = vfreq;
    for (i = 0; i < 6; i = i + 1) {
        sum = sum + 1/freq * abs(.5 - noise(freq * Psh));
        freq = 2 * freq;
    }
    Ci = mix(Cs, white, sum*4.0);
    Oi = 1.0;              /* Always make the surface opaque */
}
```

RenderMan Shading Language

```
surface clouds(float vfreq = .8 )
{
    float sum ;
    float i;
    color white = color(1.0, 1.0, 1.0);
    point Psh = transform("shader", P);

    sum = 0;
    freq = vfreq;
    for (i = 0; i < 6; i = i + 1) {
        sum = sum + 1/freq * abs(.5 - noise(freq * Psh));
        freq = 2 * freq;
    }
    Ci = mix(Cs, white, sum*4.0);
    Oi = 1.0;              /* Always make the surface opaque */
}
```

RenderMan Shading Language

```
surface clouds(float vfreq = .8 )
{
    float sum ;
    float i;
    color white = color(1.0, 1.0, 1.0);
    point Psh = transform("shader", P);

    sum = 0;
    freq = vfreq;
    for (i = 0; i < 6; i = i + 1) {
        sum = sum + 1/freq * abs(.5 - noise(freq * Psh));
        freq = 2 * freq;
    }
    Ci = mix(Cs, white, sum*4.0);
    Oi = 1.0;              /* Always make the surface opaque */
}
```

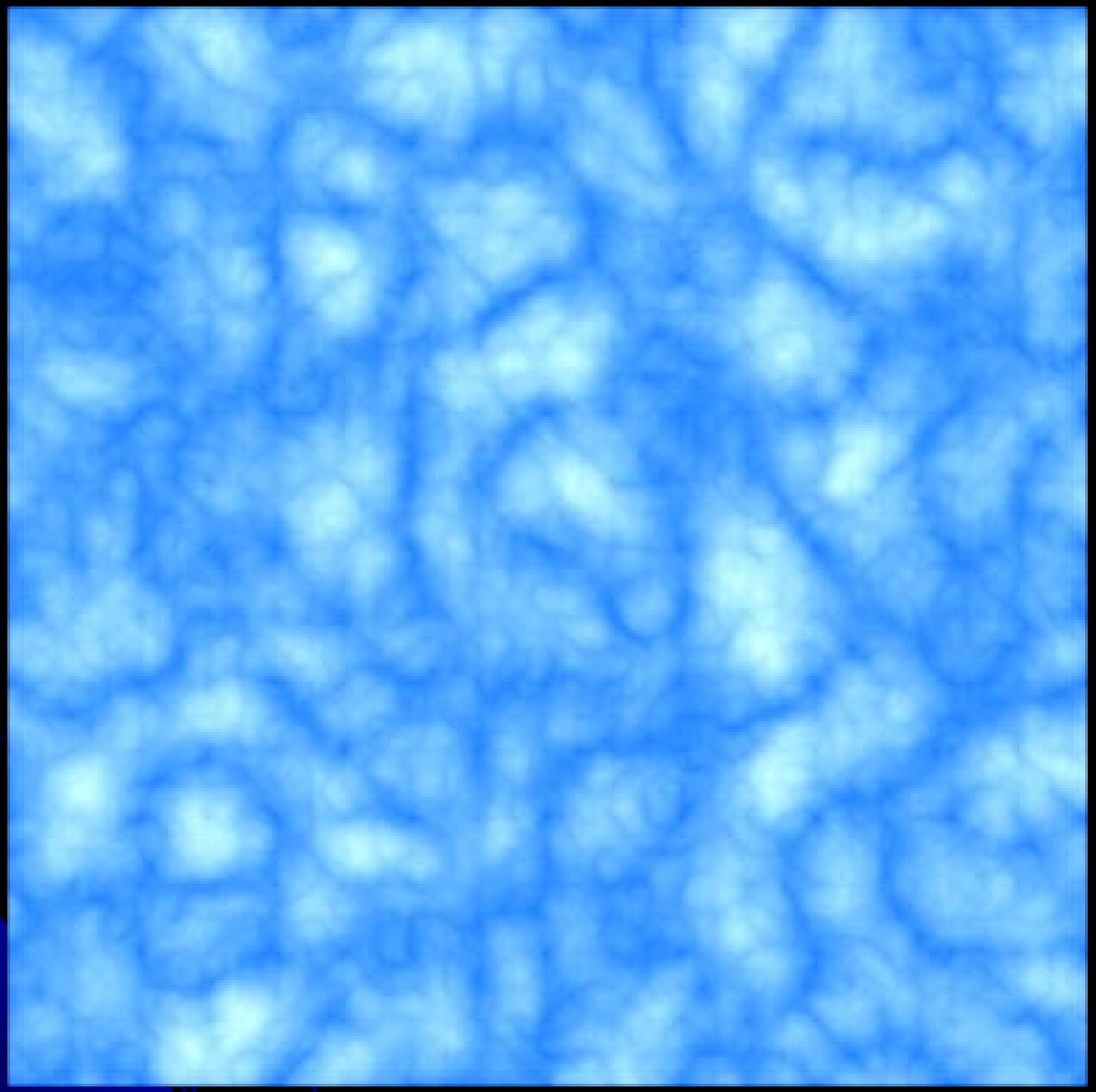
RenderMan Shading Language

```
surface clouds(float vfreq = .8 )
{
    float sum ;
    float i;
    color white = color(1.0, 1.0, 1.0);
    point Psh = transform("shader", P);

    sum = 0;
    freq = vfreq;
    for (i = 0; i < 6; i = i + 1) {
        sum = sum + 1/freq * abs(.5 - noise(freq * Psh));
        freq = 2 * freq;
    }
    Ci = mix(Cs, white, sum*4.0);
    Oi = 1.0;              /* Always make the surface opaque */
}
```

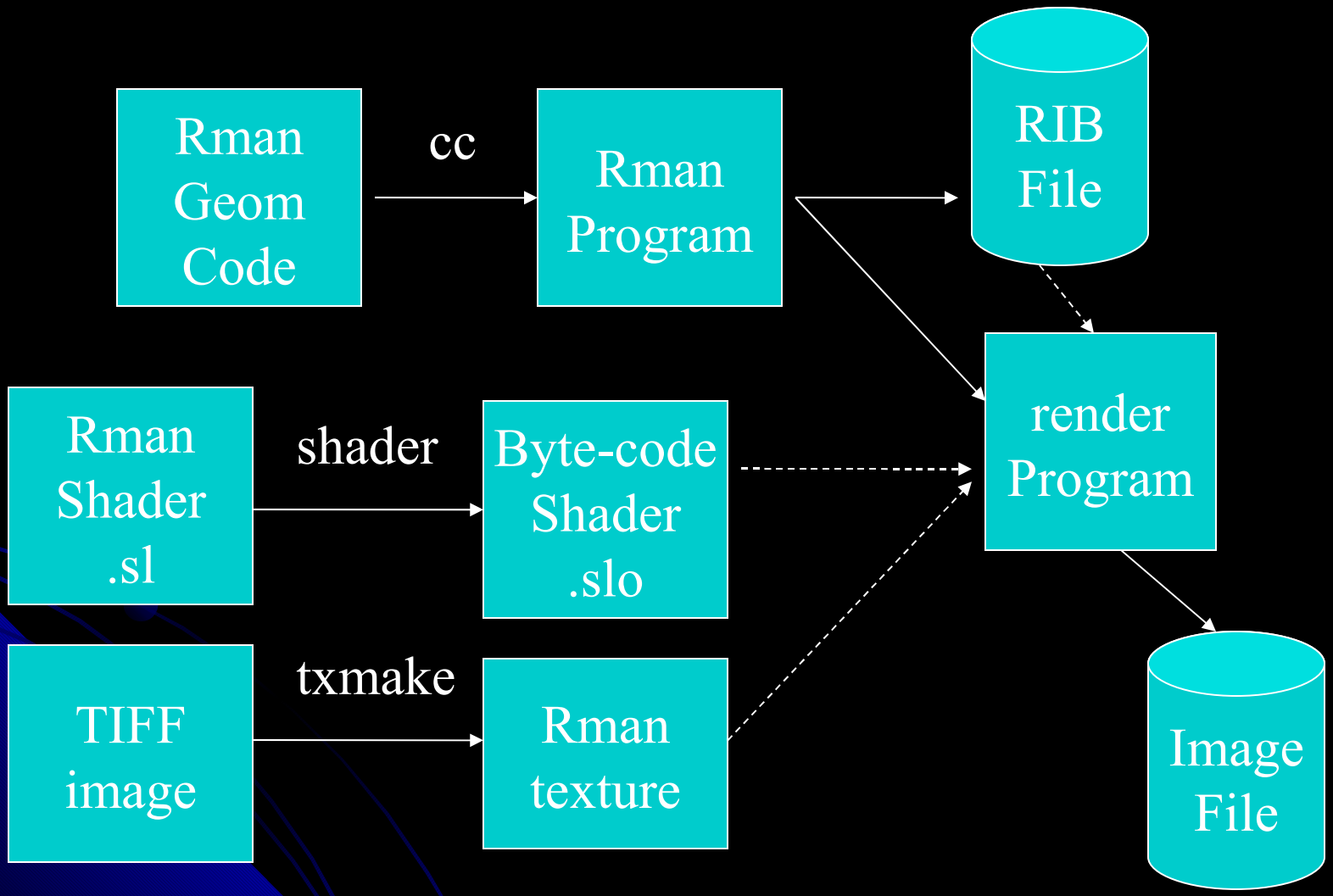
RIB s použitím “Shader” kódu

```
Display "RenderMan" "framebuffer"
  "rgb"
Format 256 192 1
WorldBegin
Surface "clouds"
Polygon "P" [0.5 0.5 0.5 0.5 -0.5
  0.5 -0.5 -0.5 0.5 -0.5 0.5 0.5]
WorldEnd
```

Result

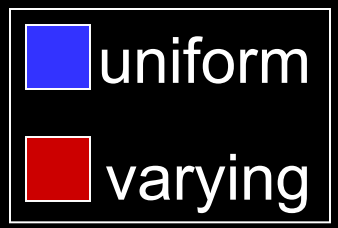


Součásti systému RenderMan



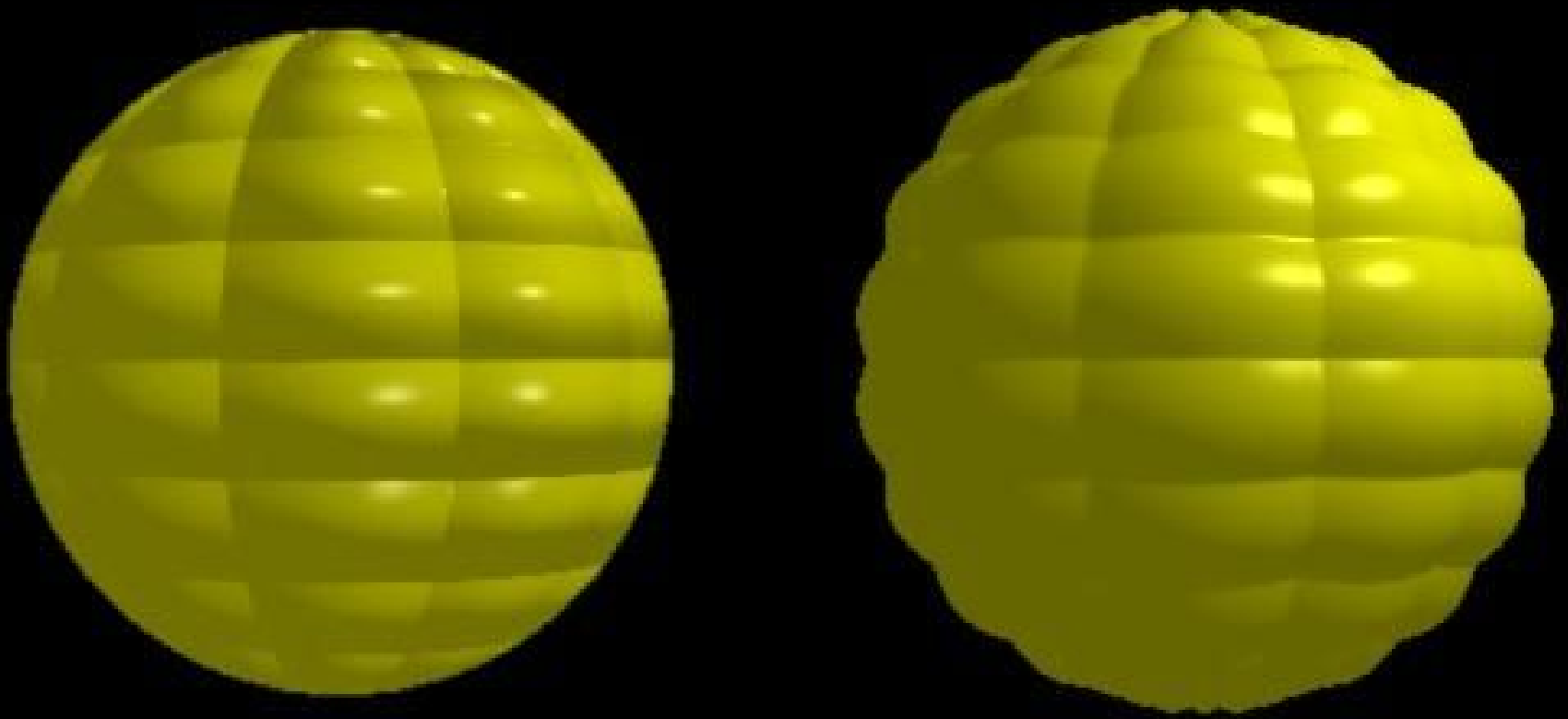
Surface Shader

```
surface plastic( float   Ks = .5, Kd = .5,
                 Ka =  1, roughness = .1;
                 color   specularcolor =  1 )
{
    normal Nf = faceforward(normalize(N), I );
    vector V = normalize(-I);

    Oi = Os;
    Ci = Os*( Cs * (Ka*ambient()+
   Kd*diffuse(Nf))+
      specularcolor * Ks *
   specular(Nf,V,roughness) );
}
```



Light Shader

```
light
pointlight(
        float   intensity  = 1;
        color   lightcolor = 1;
        point   from  = point "camera"
   (0,0,0) )
{
        illuminate( from )
             Cl = intensity * lightcolor / L.L;
}
```



„Rendering“

- rasterizace

Vlastnosti všech bodů v prostoru/modelu jsou lokálně definovány

- “raytracing“

Vlastnosti bodů jsou ovlivněny globálně všemi ostatními body ve scéně/modelu.

Dokáže správně vykreslit tansparetnost, refrakci světla a podobné efekty.

Rendering Pipeline - OpenGL

Aaron Bloomfield

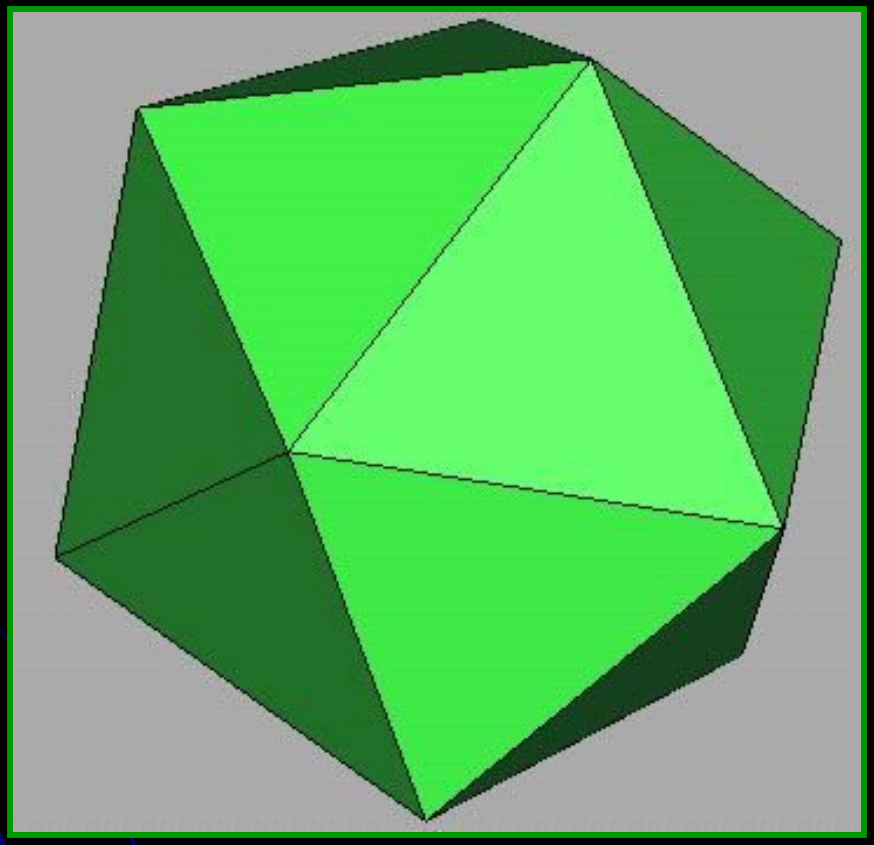
CS 445: Introduction to Graphics

Fall 2006

(Slide set originally by Greg Humphreys)

3D Polygon Rendering

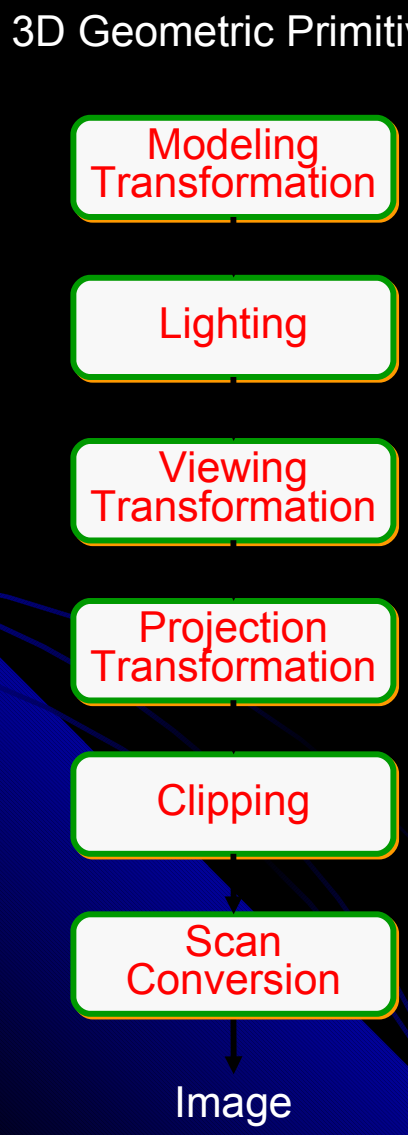
Many applications use rendering of 3D polygons with direct illumination



3D Polygon Rendering

Many applications use rendering of 3D polygons with direct illumination

3D Rendering Pipeline



This is a pipelined sequence of operations to draw a 3D primitive into a 2D image

(this pipeline applies only for direct illumination)

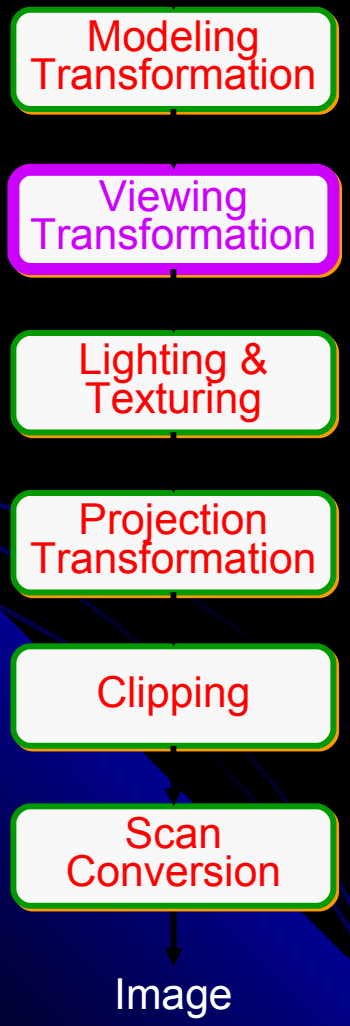
3D Rendering Pipeline

3D Geometric Primitives

Transform into 3D world coordinate system

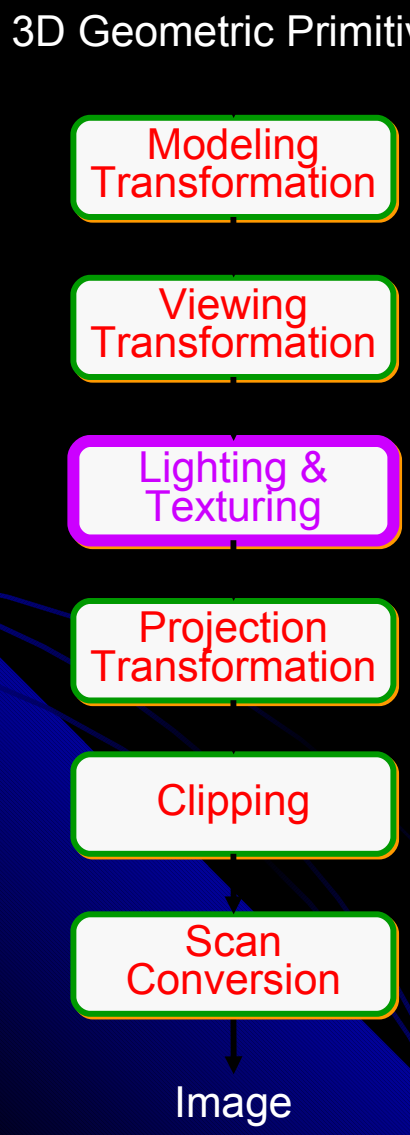
3D Rendering Pipeline

3D Geometric Primitives



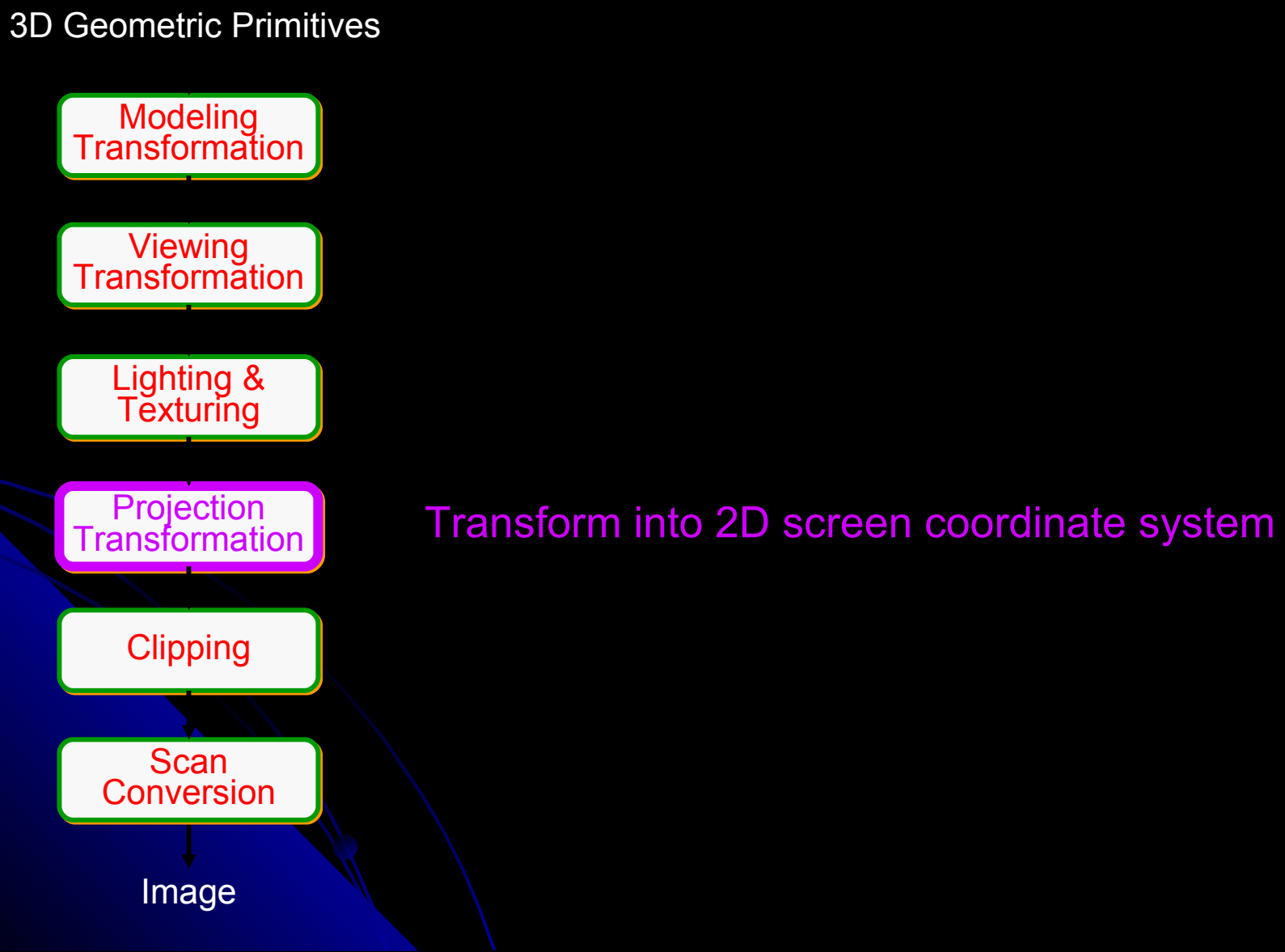
Transform into 3D camera coordinate system
Done with modeling transformation

3D Rendering Pipeline



Illuminate according to lighting and reflectance
Apply texture maps

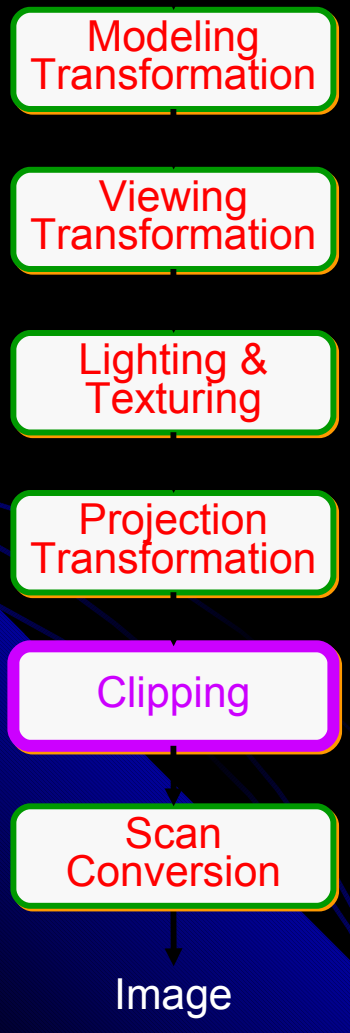
3D Rendering Pipeline



Transform into 2D screen coordinate system

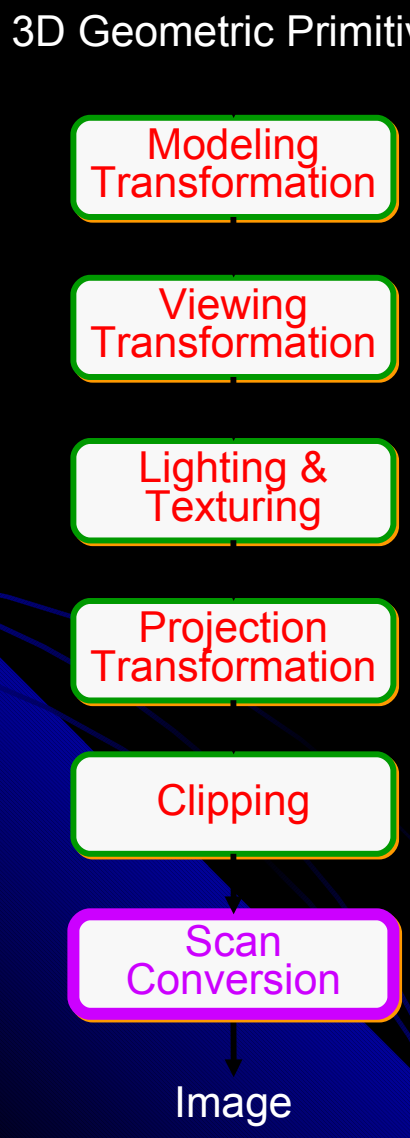
3D Rendering Pipeline

3D Geometric Primitives



Clip primitives outside camera's view

3D Rendering Pipeline

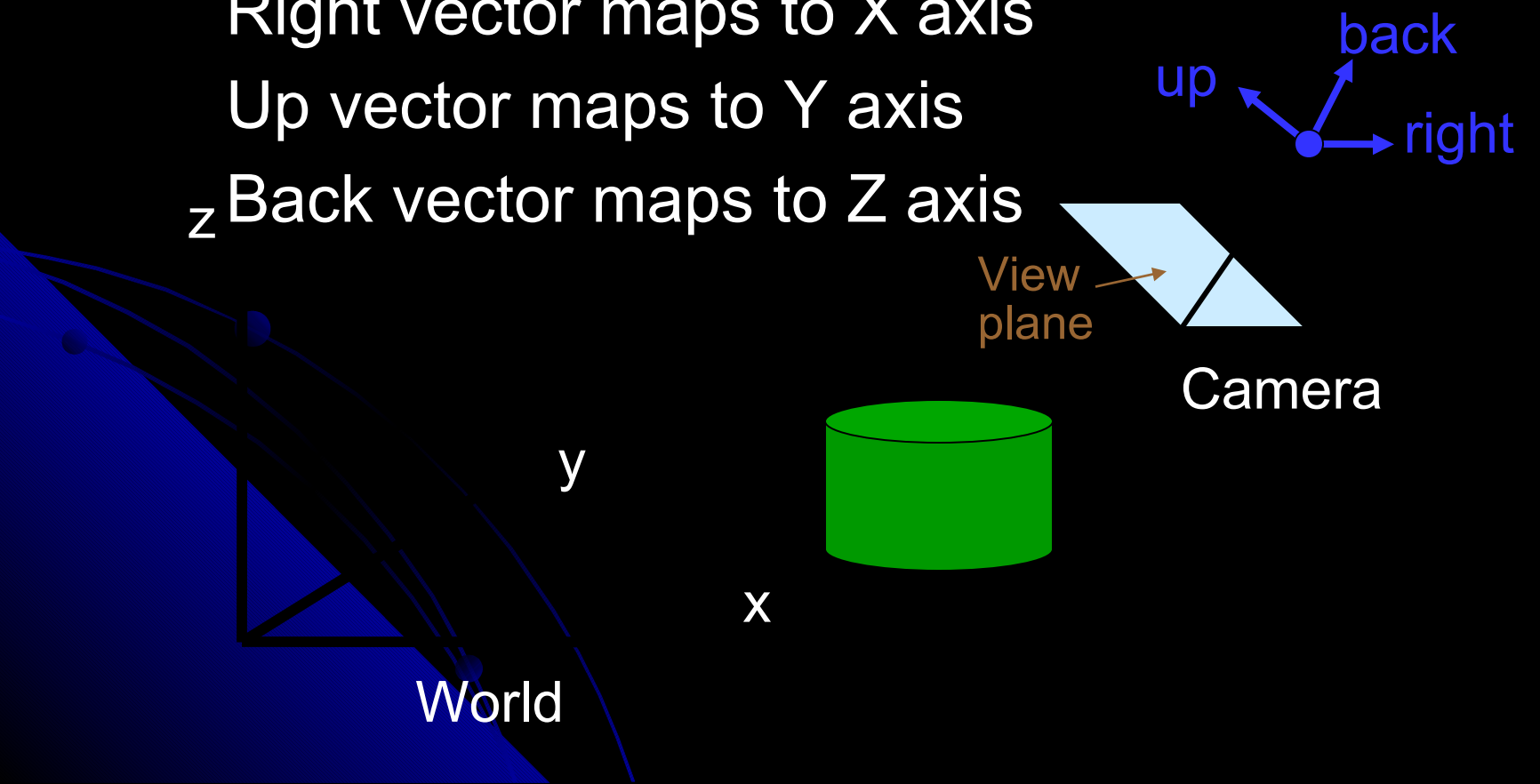


Draw pixels (includes texturing, hidden surface, ...)

Viewing Transformation

Mapping from world to camera coordinates

- Eye position maps to origin
- Right vector maps to X axis
- Up vector maps to Y axis
- Back vector maps to Z axis



Projection

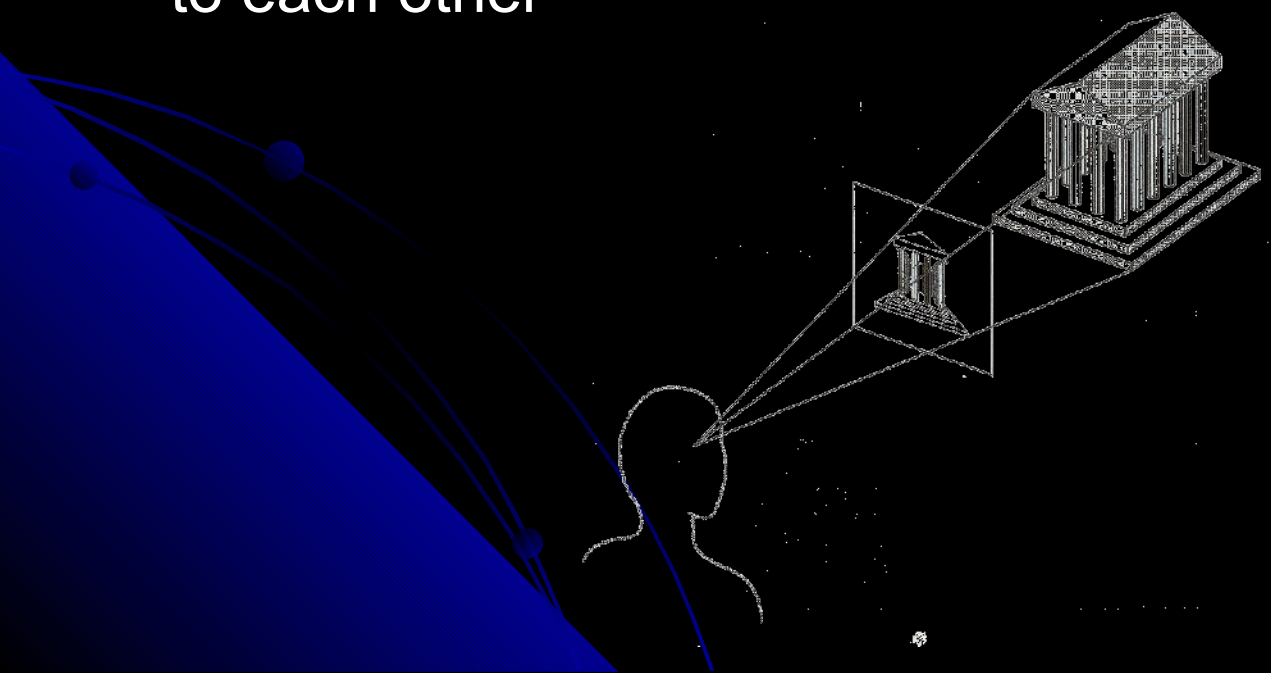
General definition:

- Transform points in *n*-space to *m*-space (*m*<*n*)

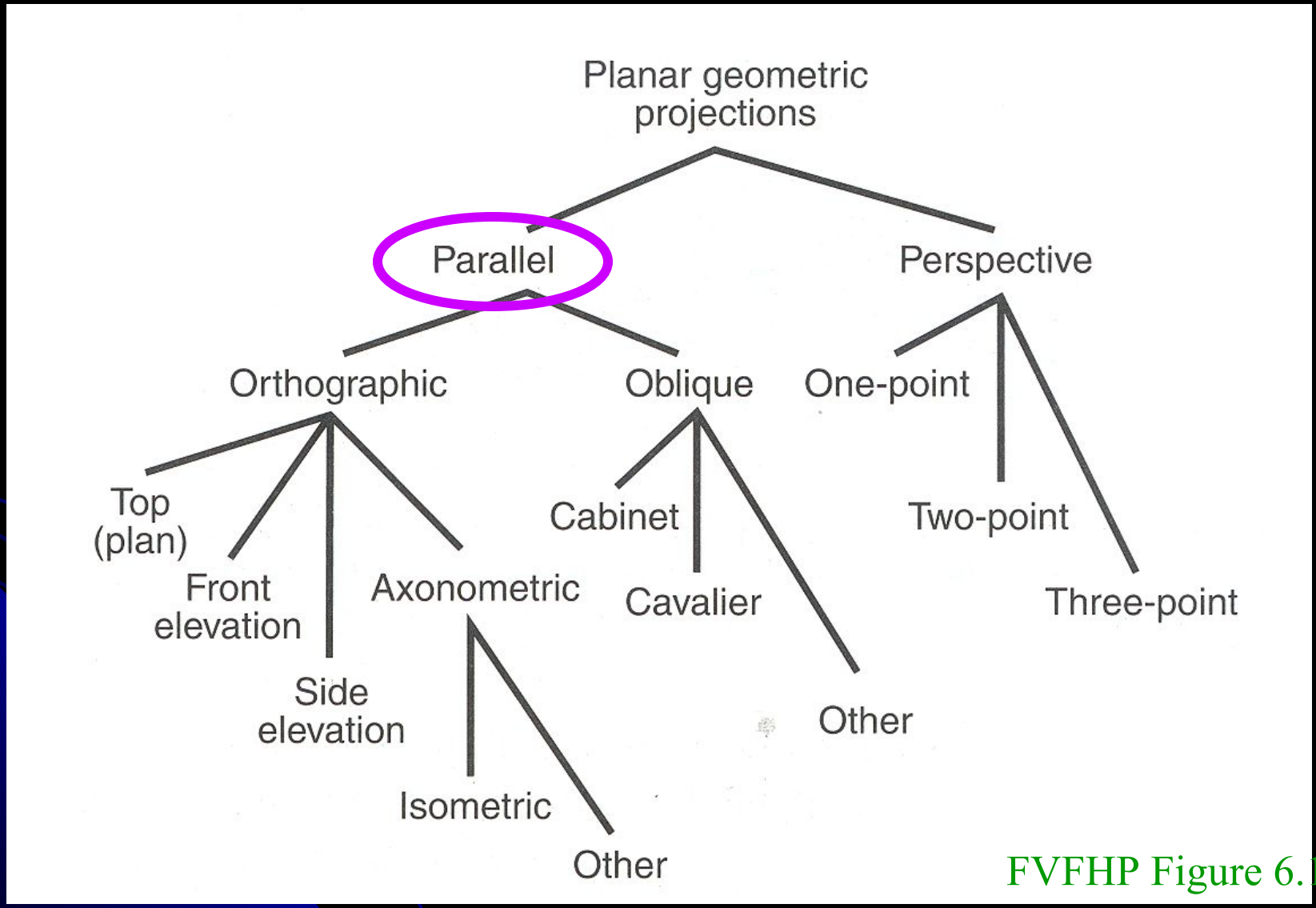
In computer graphics:

- Map 3D camera coordinates to 2D screen coordinates

For perspective transformations, no two “rays” are parallel to each other



Taxonomy of Projections

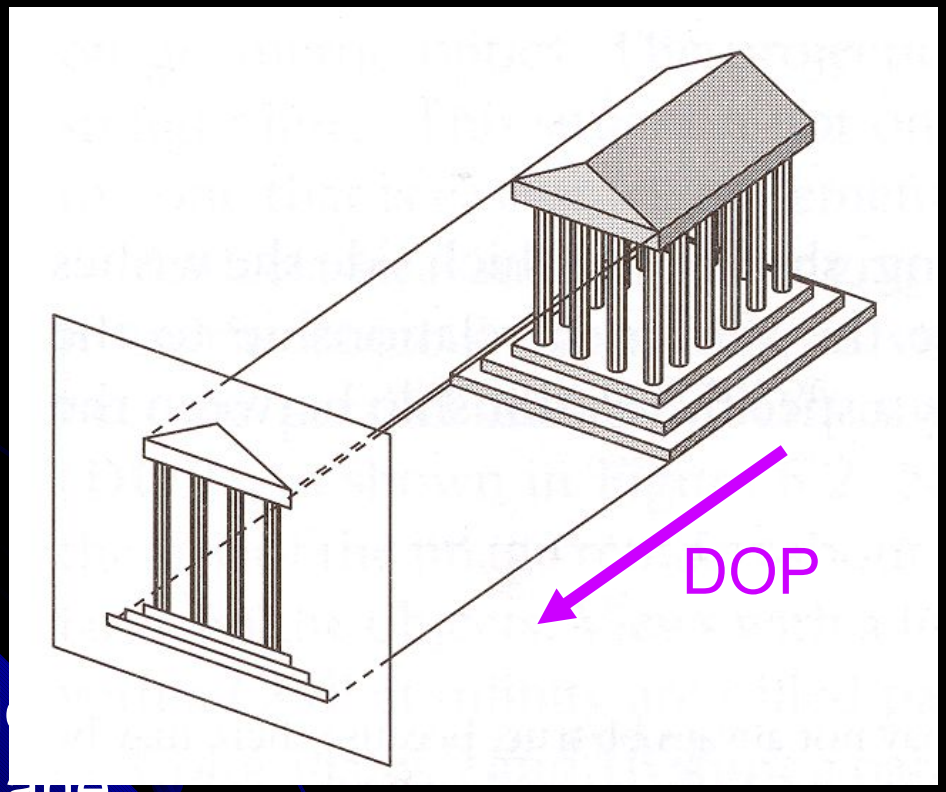


FVFHP Figure 6.10

Parallel Projection

Center of projection is at infinity

Direction of projection (DOP) same for all points

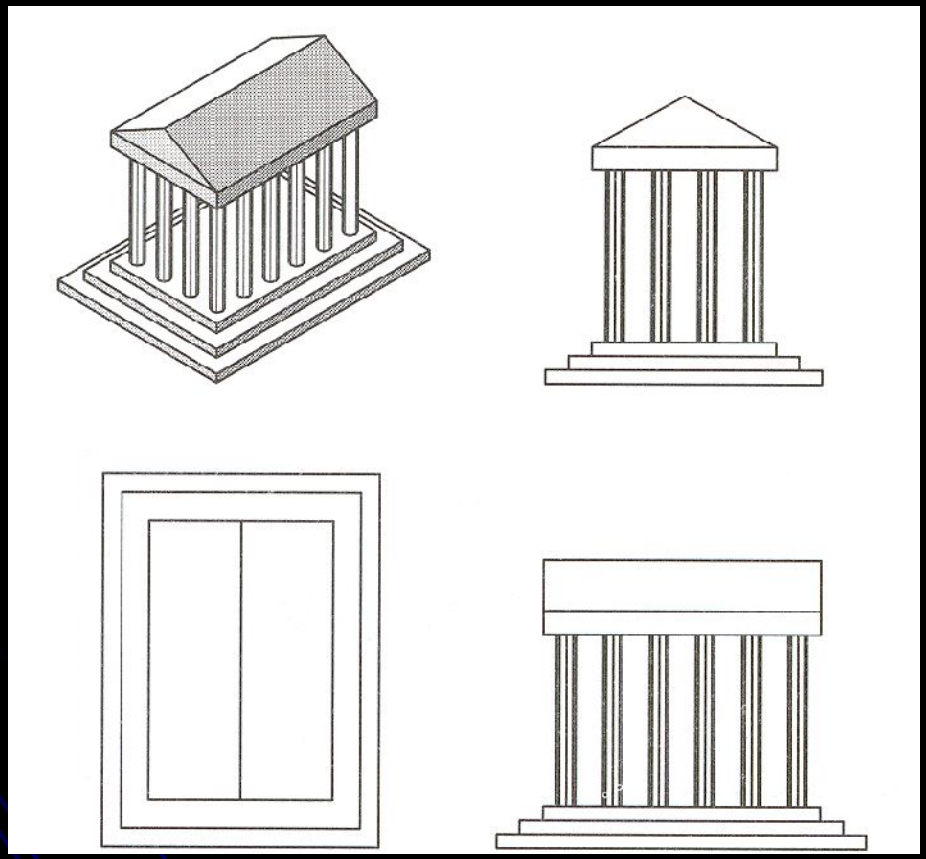


Vi
Plane

Angel Figure 5.4

Orthographic Projections

DOP perpendicular to view plane

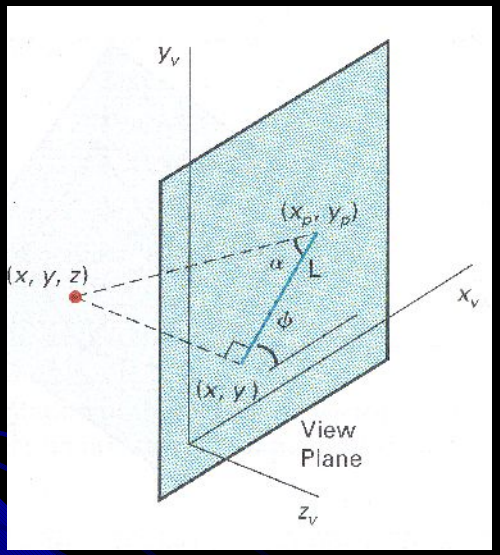


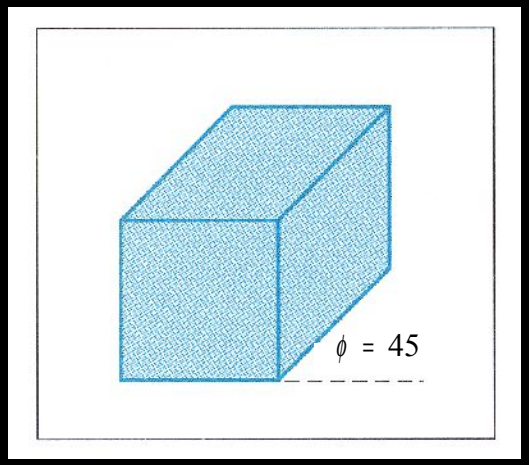
Top Side

Angel Figure 5.5

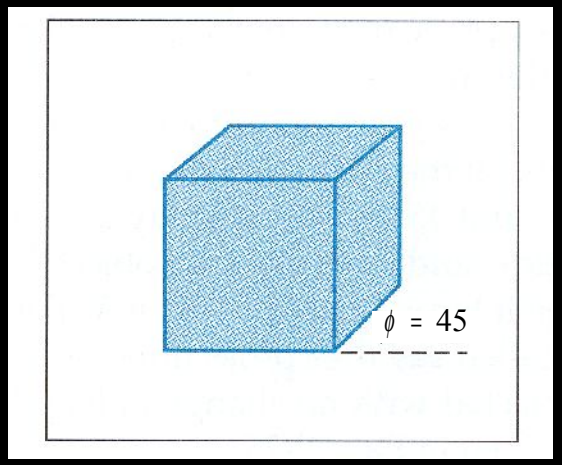
Oblique Projections

DOP not perpendicular to view plane





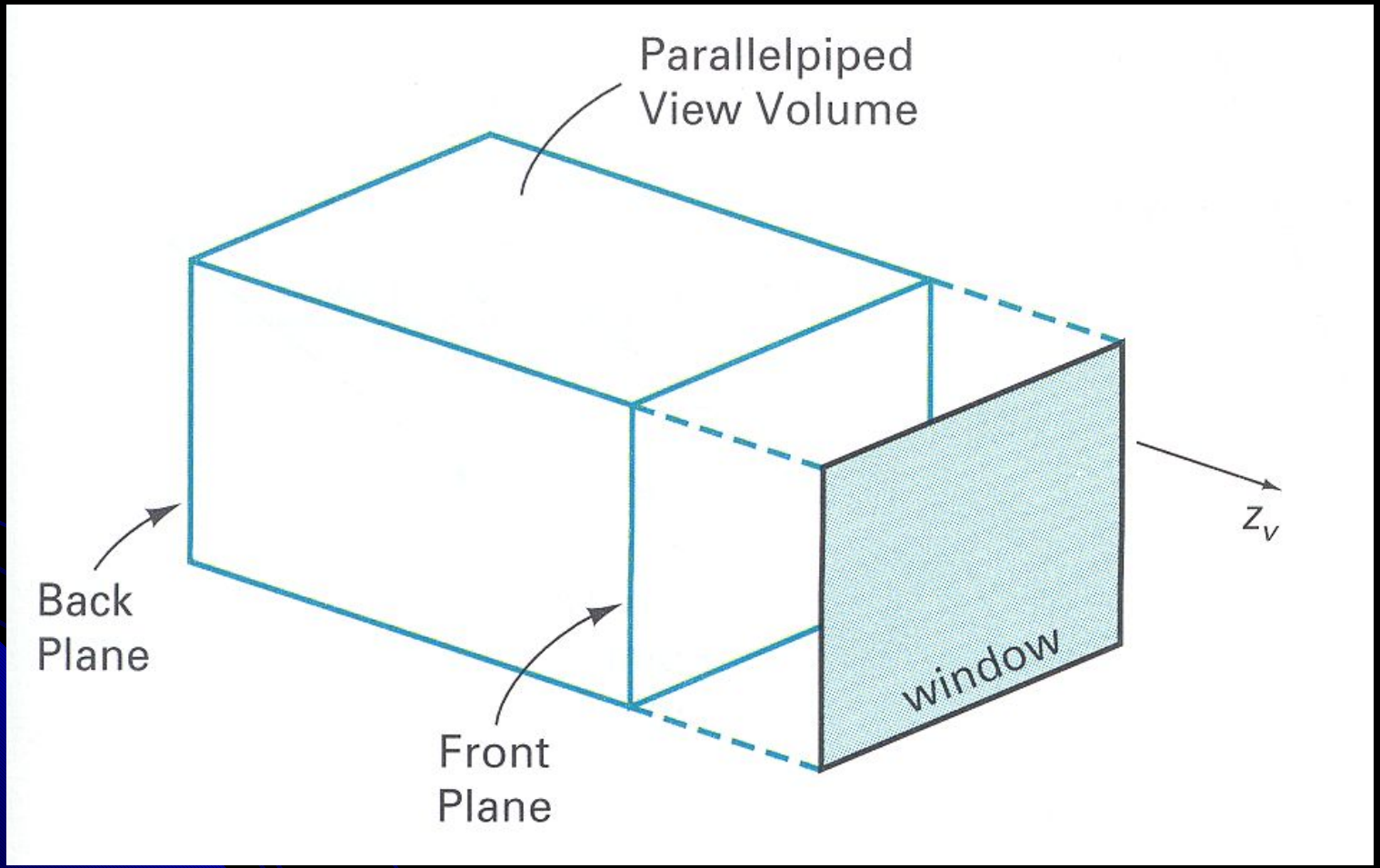
Cavalier
(DOP $\alpha = 45^{\circ}$)



Cabinet
(DOP $\alpha = 63.4^{\circ}$)

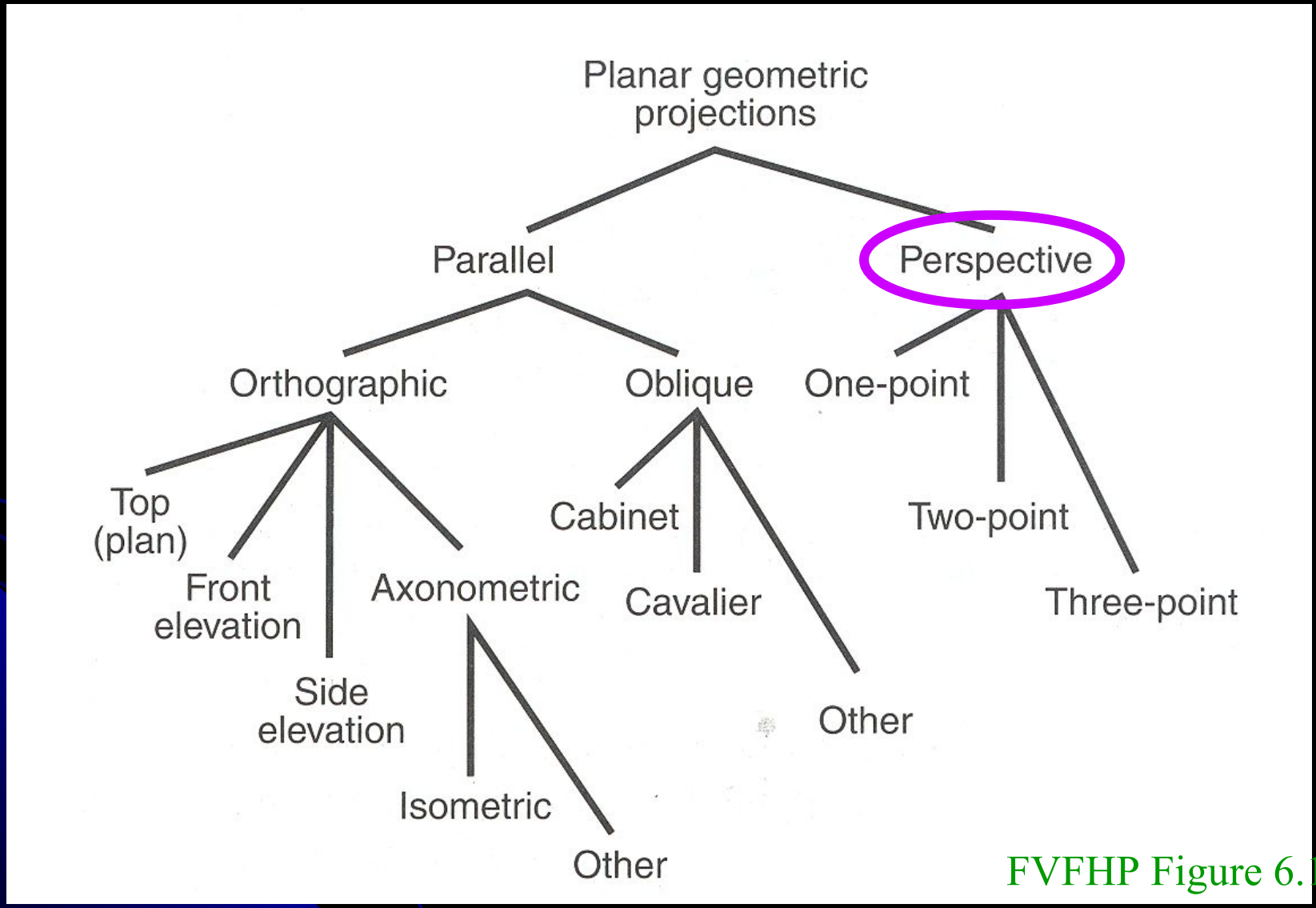
H&B Figure 12.24

Parallel Projection View Volume



H&B Figure 12.30

Taxonomy of Projections

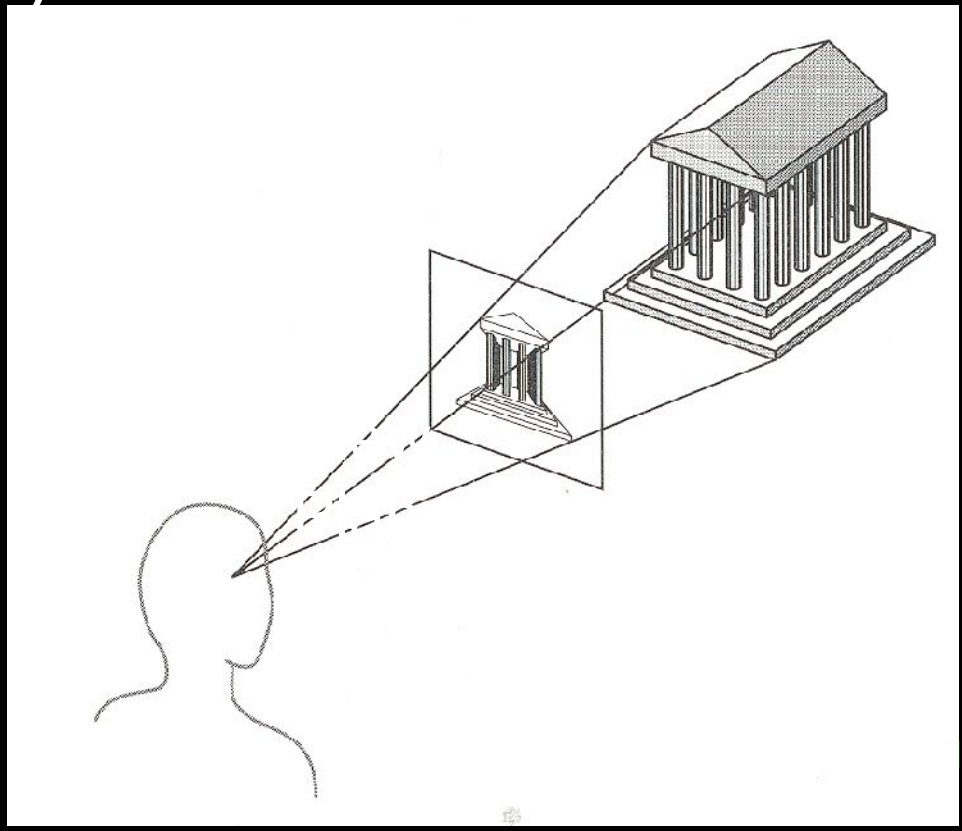


FVFHP Figure 6.10

Perspective Projection

Map points onto “view plane” along “projectors” emanating from “center of projection” (COP)

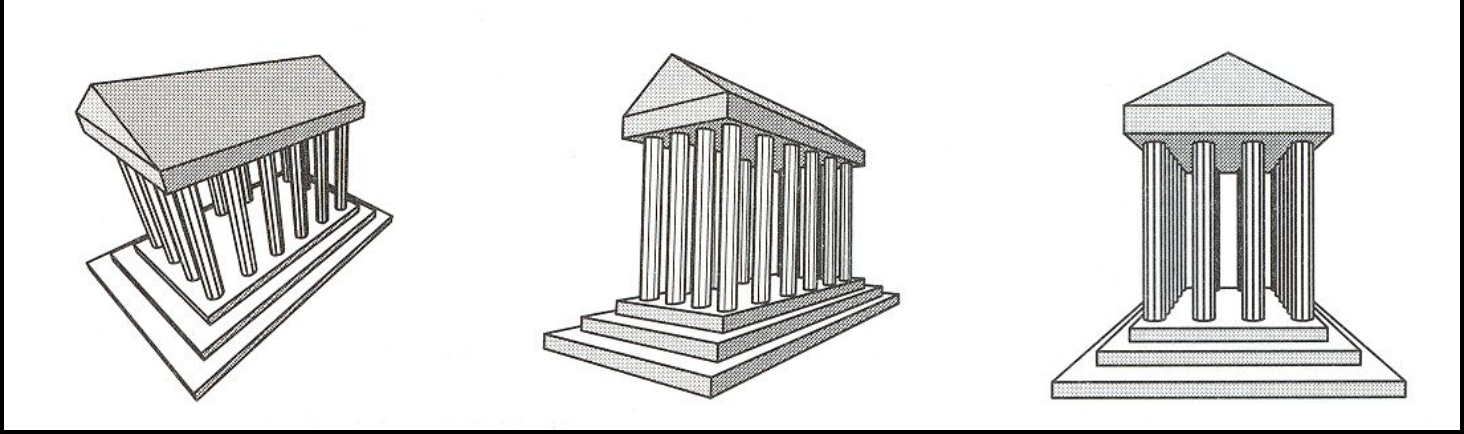
Center of Projection



re 5.9

Perspective Projection

How many vanishing points?

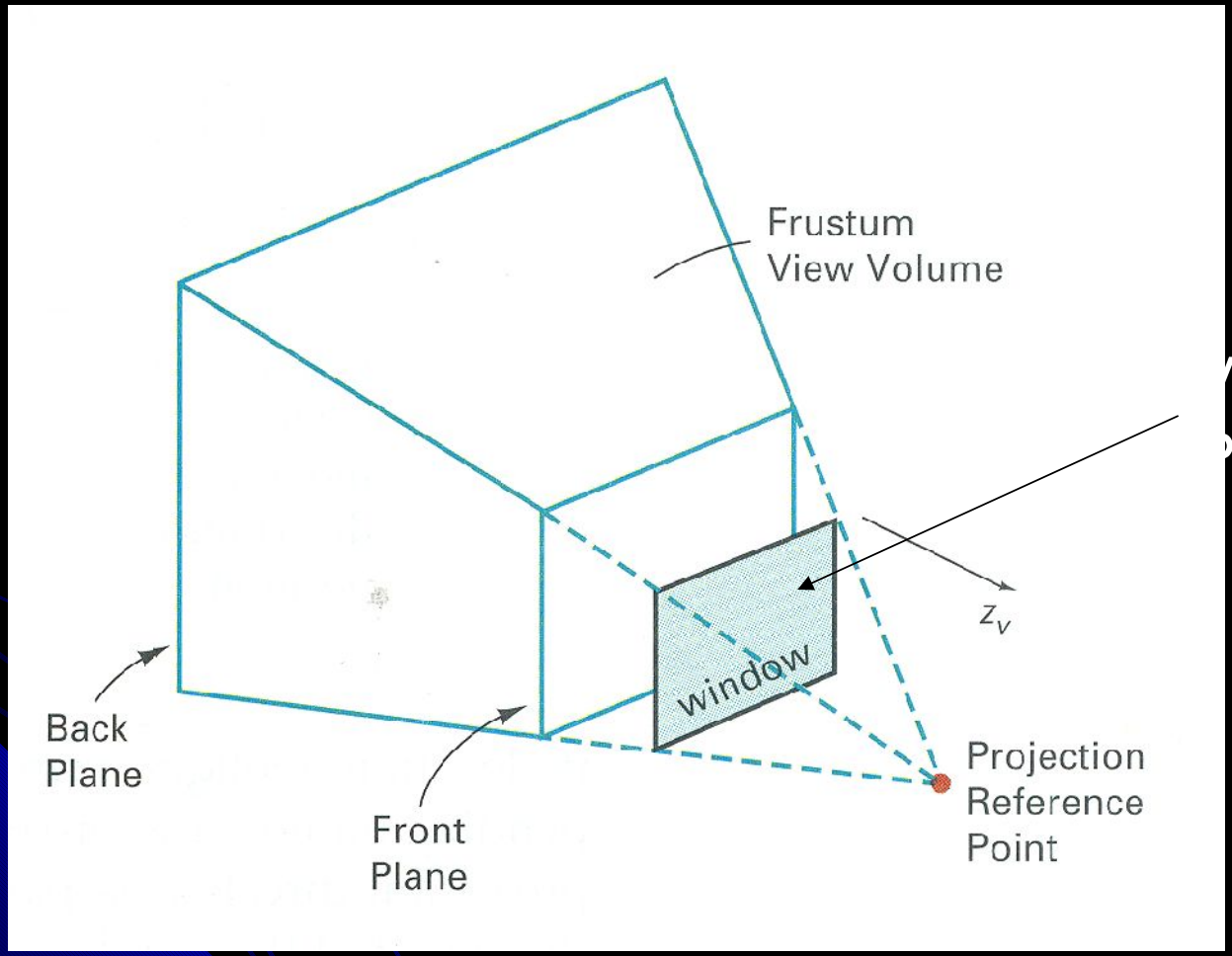


3-Point Perspective | 2-Point Perspective | 1-Point Perspective

- The difference is how many of the three principle directions are parallel/orthogonal to the projection plane

Angel Figure 5.10

Perspective Projection View Volume



H&B Figure 12.30

Camera to Screen

Remember: Object → Camera → Screen

Just like raytracer

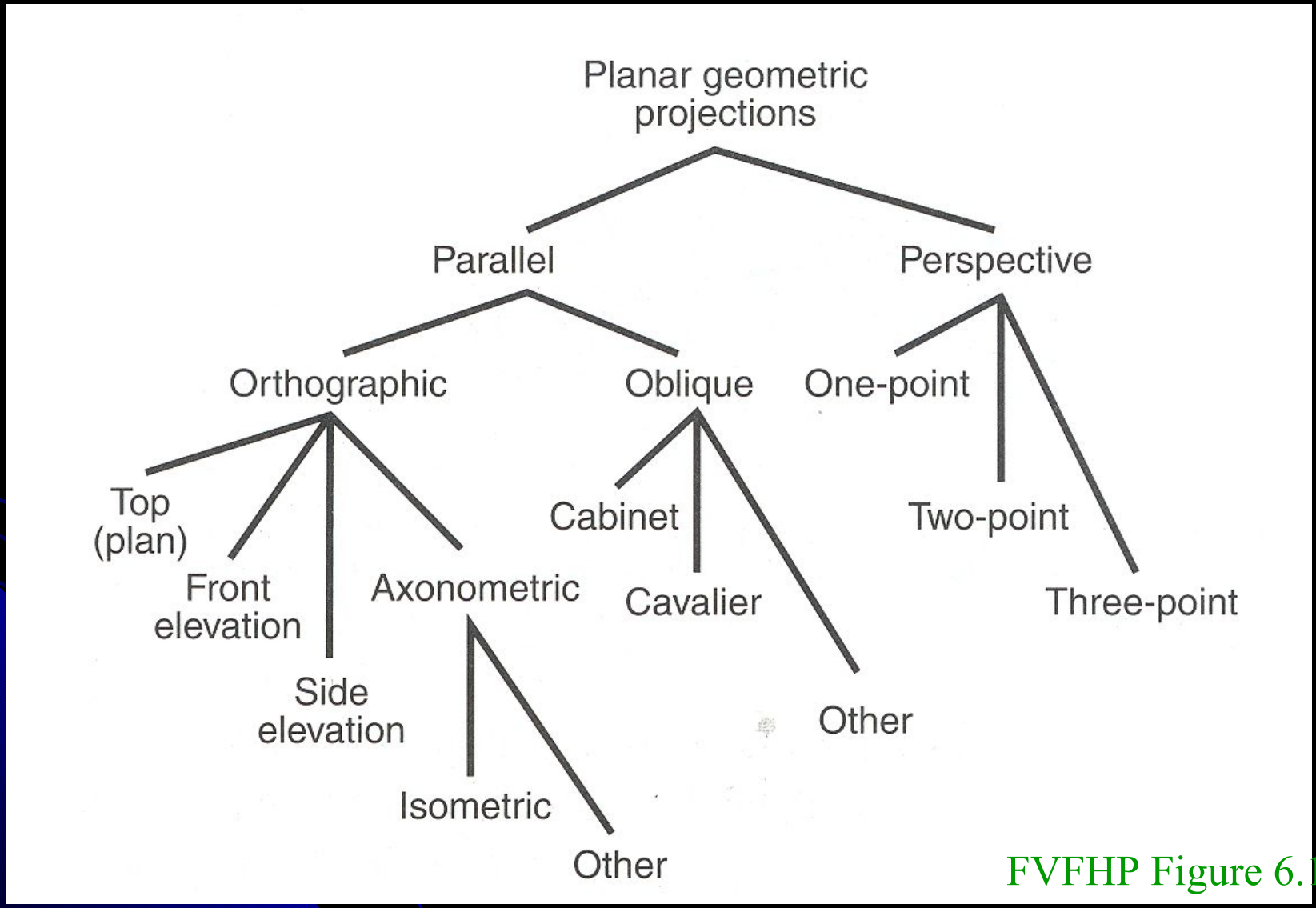
"screen" is the $z=d$ plane for some constant d

Origin of screen coordinates is (0,0,d)

Its *x* and *y* axes are parallel to the *x* and *y* axes of the eye coordinate system

- All these coordinates are in camera space now

Taxonomy of Projections

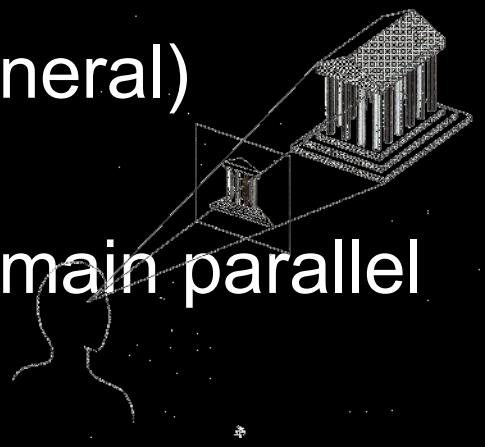


FVFHP Figure 6.10

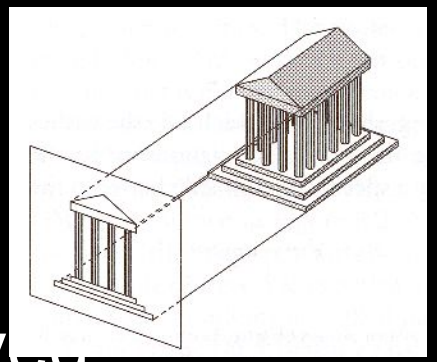
Perspective vs. Parallel

Perspective projection

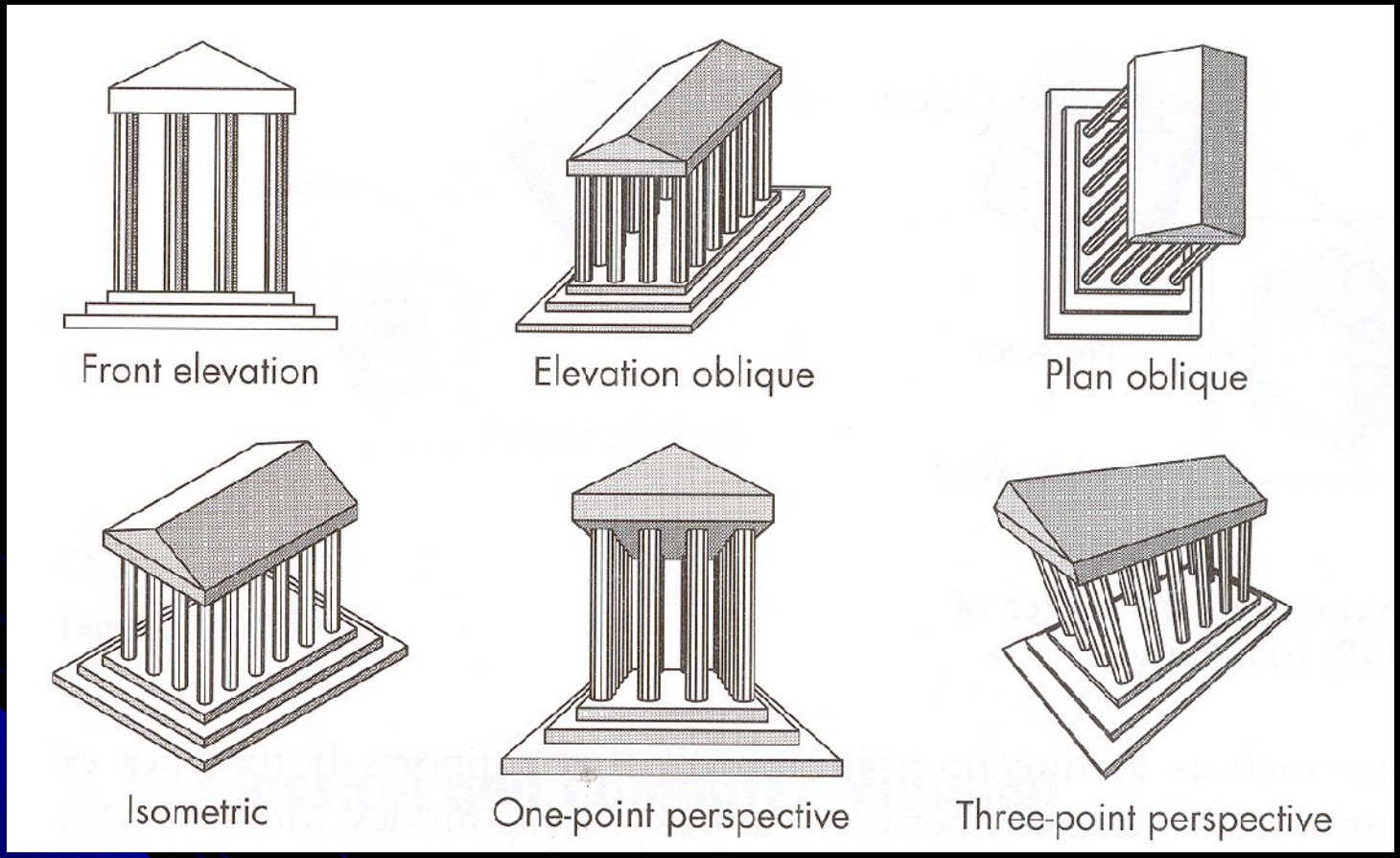
- Size varies inversely with distance - looks realistic
- Distance and angles are not (in general) preserved
- Parallel lines do not (in general) remain parallel



- Parallel projection
  - Good for exact measurements
  - Parallel lines remain parallel
  - Angles are not (in general) preserved



Classical Projections



Angel Figure 5.3

CSC 480 / 580

Computer Graphics

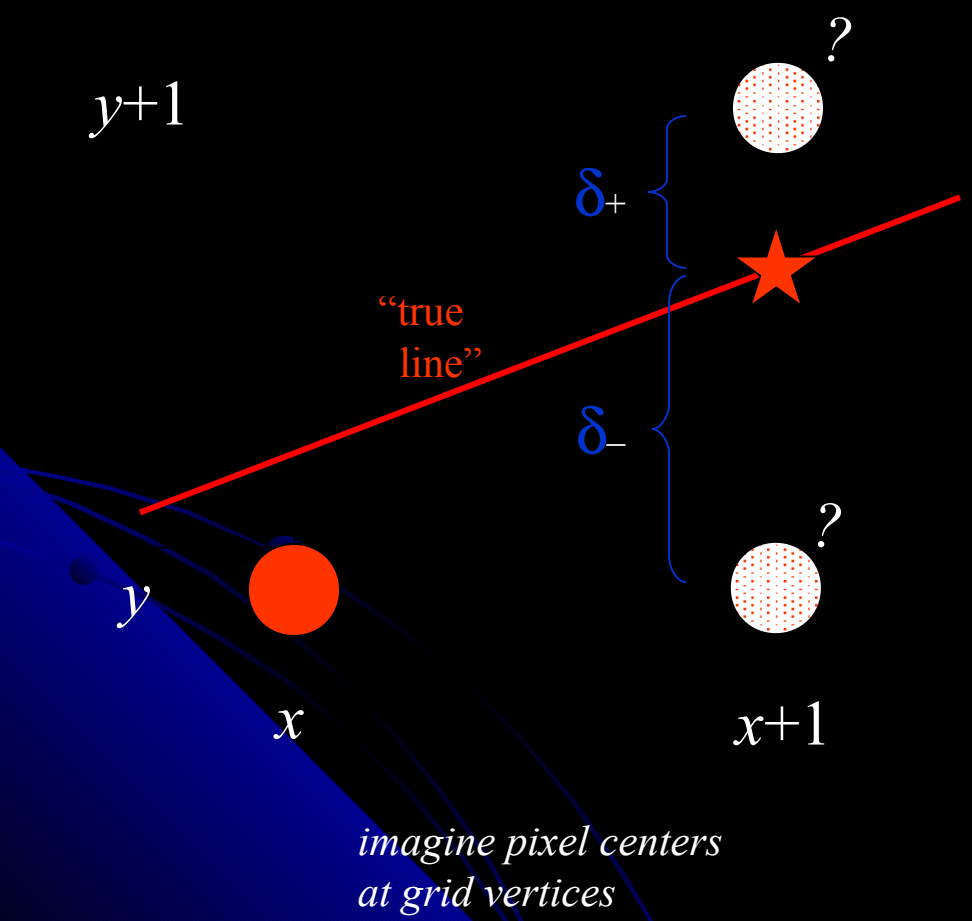
K. Kirby

Scan Conversion

Filling

Scan-converting Lines - Toward the Bresenham Algorithm

Special case: $0 < m < 1$, $\Delta x > 0$



At each step:

```
x++ ;
if ( δ- > δ+)
    y++ ;
```

Let $p = \Delta x\,(\,\delta^- - \delta^+\,)$. Then:

At each step:

```
x++ ;
if ( p > 0 )
    y++ ;
update p ;
```

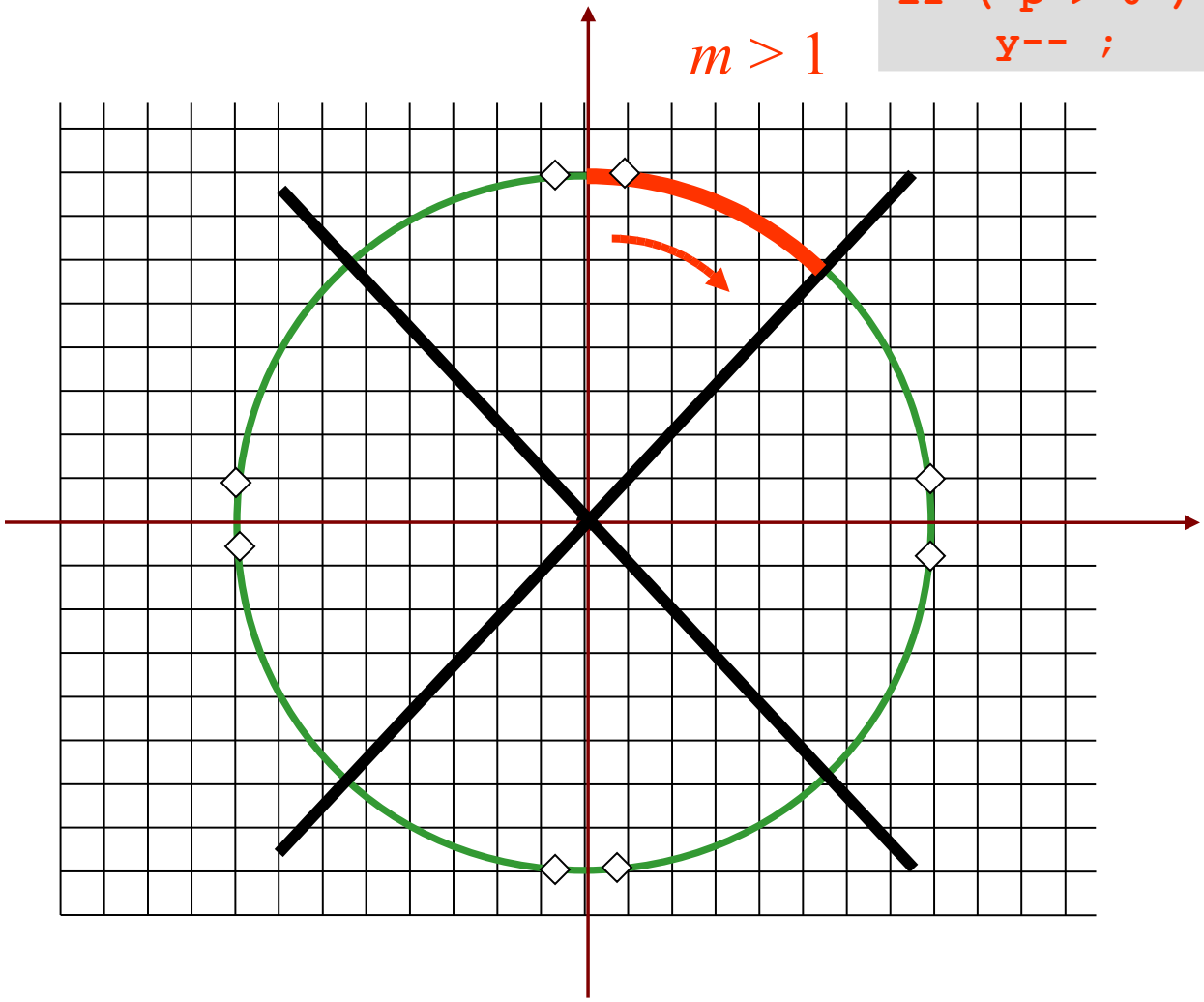
Can we do this with simple all-int arithmetic? Yes!

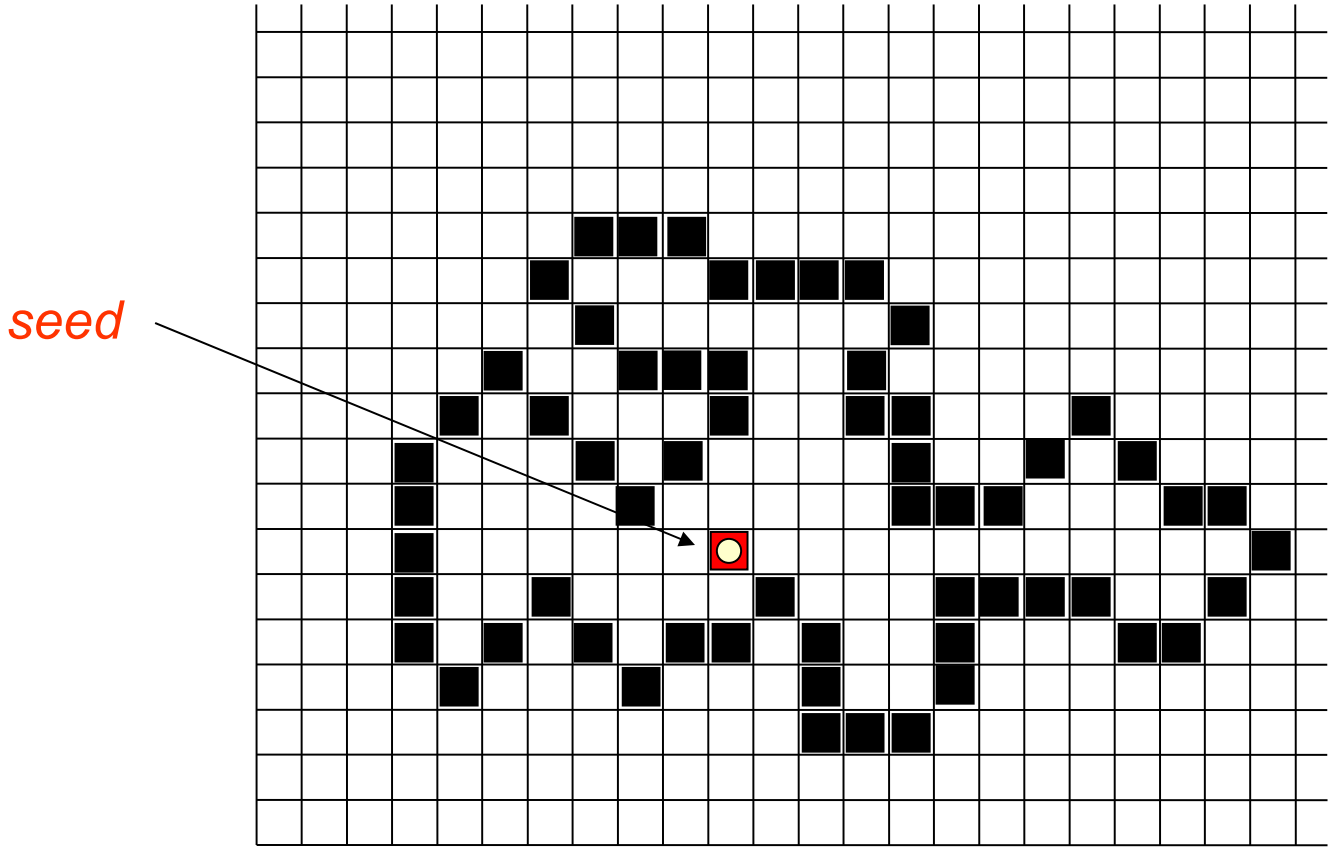
The Bresenham Algorithm

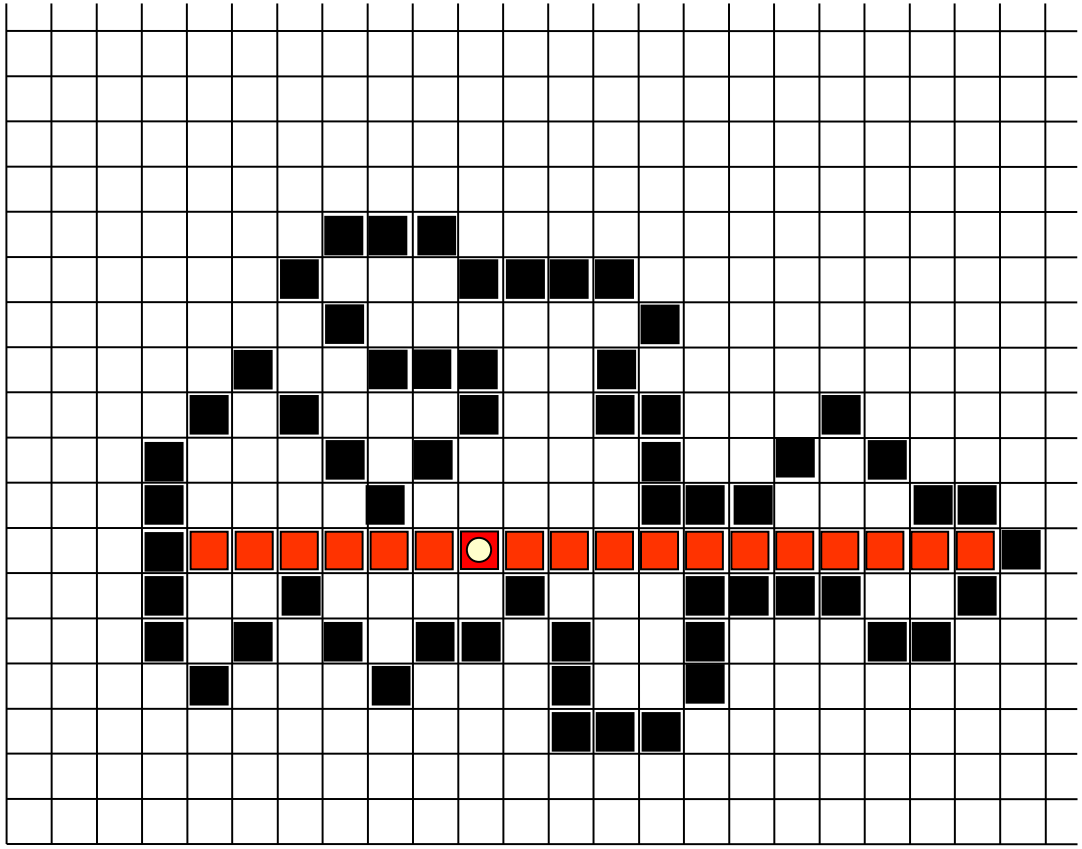
```
void line( int xA, int yA, int xB, int yB )
// Special case: shallow increasing slope.
{
    const int DX= xB - xA ;
    const int DY= yB - yA ;
    const int DP_FLAT= 2*DY ;
    const int DP_JUMP= DP_FLAT - 2*DX ;
    assert( 0 < DY && DY < DX ) ;

    int x= xA ;
    int y= yA ;
    int p= 2*DY - DX ;
    pix( xA, yA ) ;
    while ( x < xB )
    {
        x++ ;
        if ( p > 0 )
        {
            ++y ;
            p+= DP_JUMP ;
        }
        else
            p+= DP_FLAT ;
        pix( x, y ) ;
    }
}
```

initial values

```
x++
if ( p > 0 )
    y-- ;
```

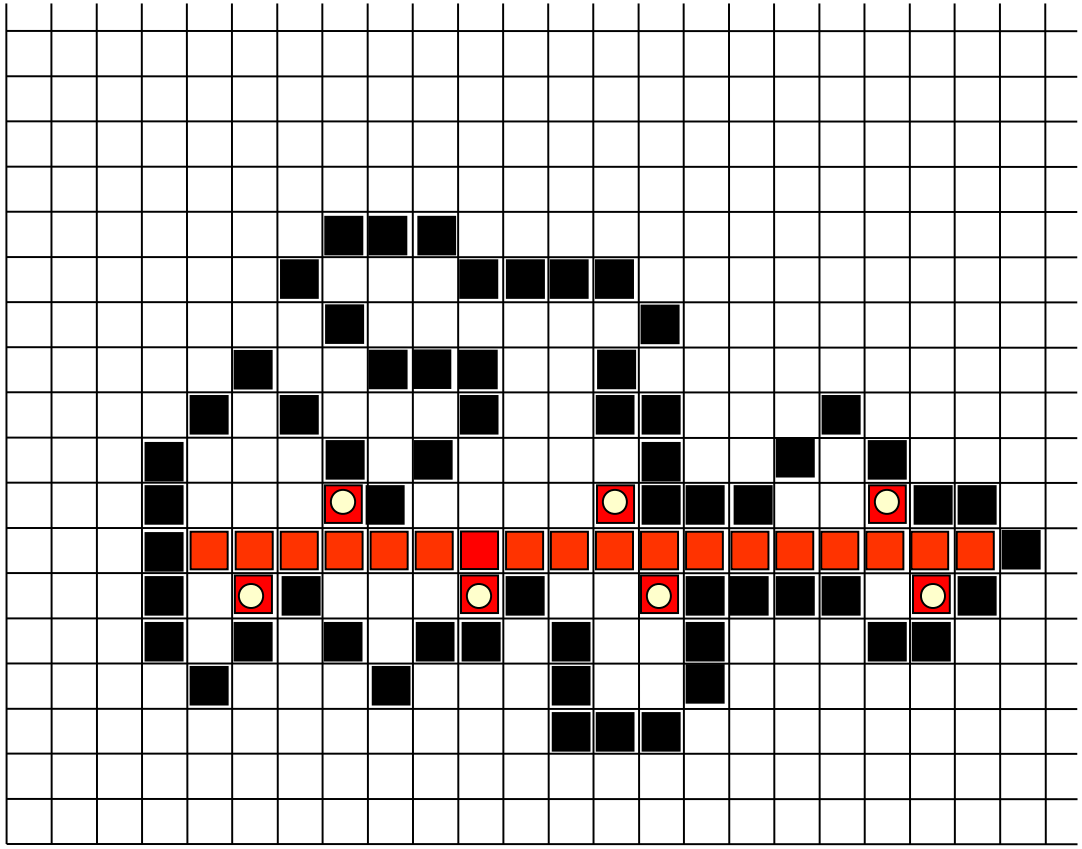



seed

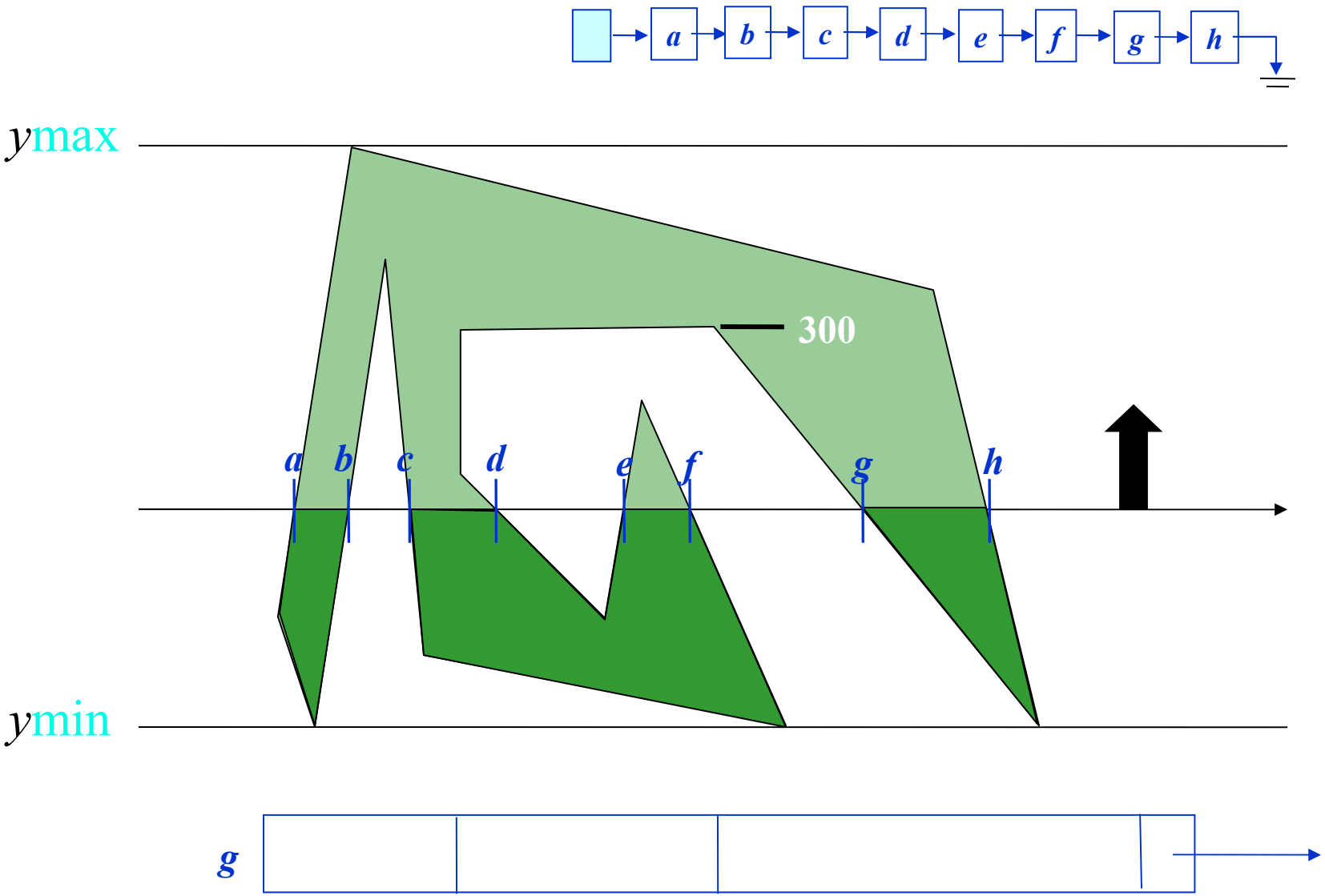


Fill the scan line

Push seeds for the next step

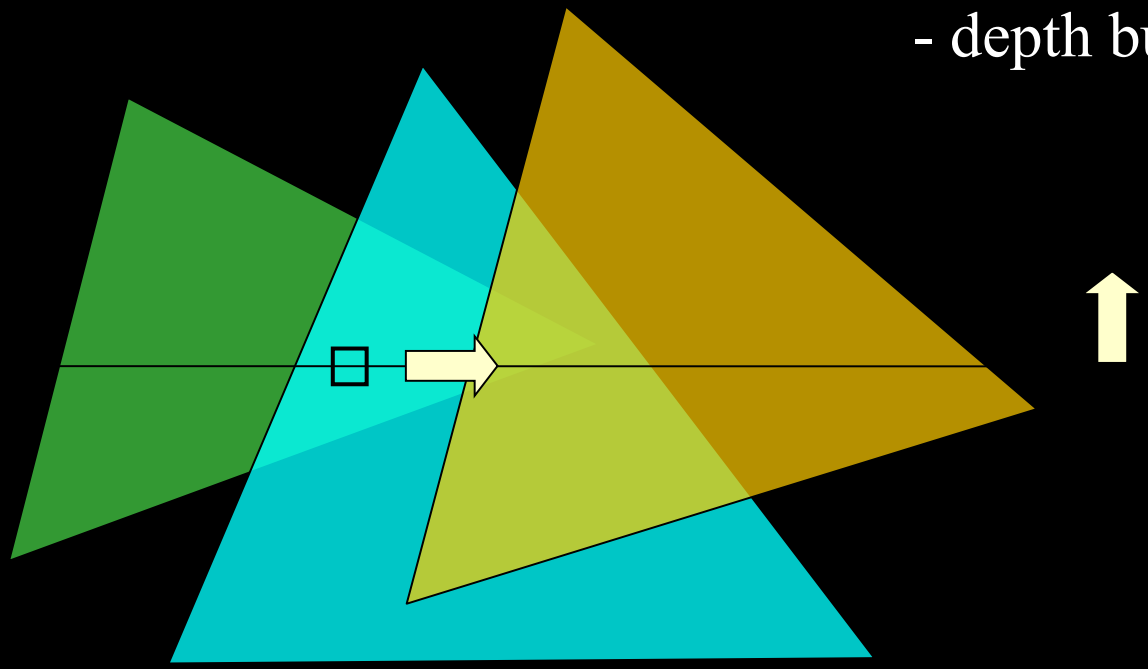


rightmost left pixels, above & below


a
b
c
d
e
f
g
h
ymax
300
a
b
c
d
e
f
g
h
ymin
g

Z-Buffering

Two buffers
- screen buffer (color)
- depth buffer

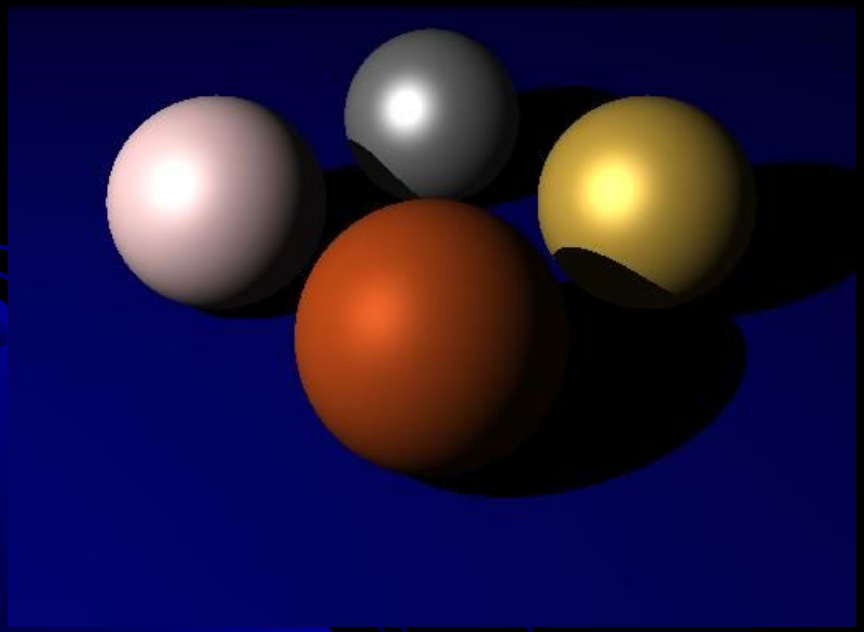


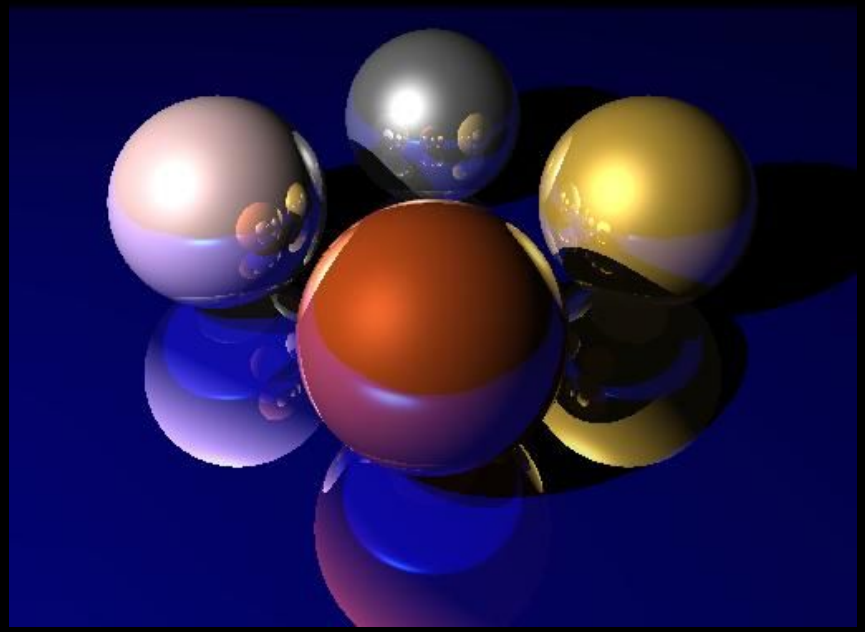
```
for each polygon
    for each scanline
        for each pixel in scanline
            update depth at pixel
            if pixel depth < buffered depth
                write to screen & depth buffers
```

Ray Tracing

Mani Thomas
CISC 440/640
Computer Graphics

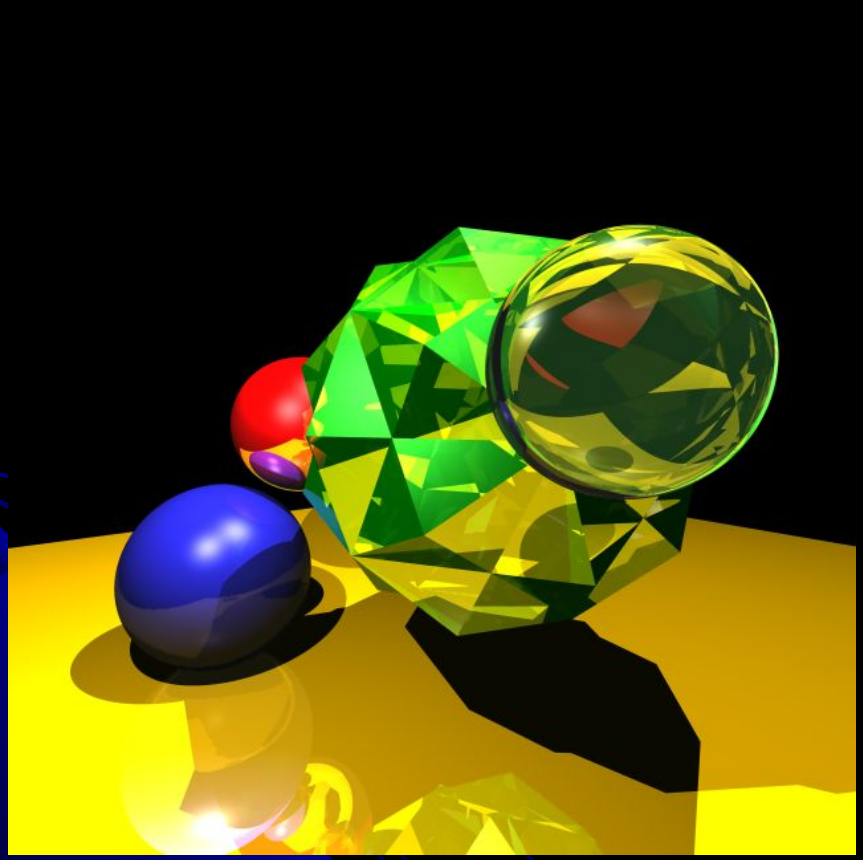
Fotorealismus

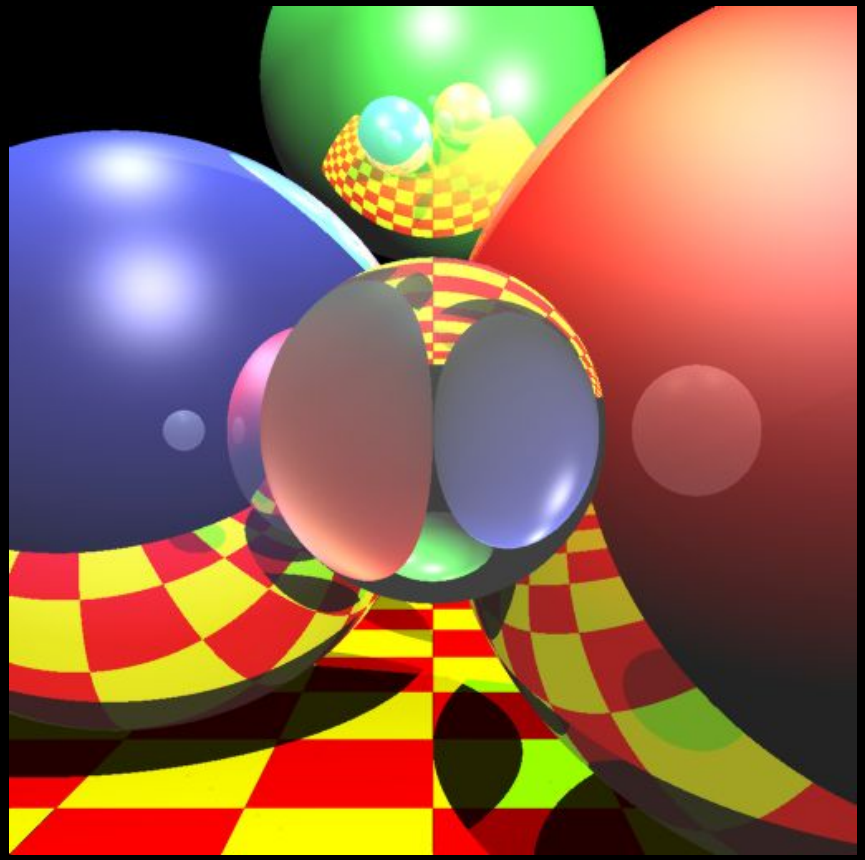




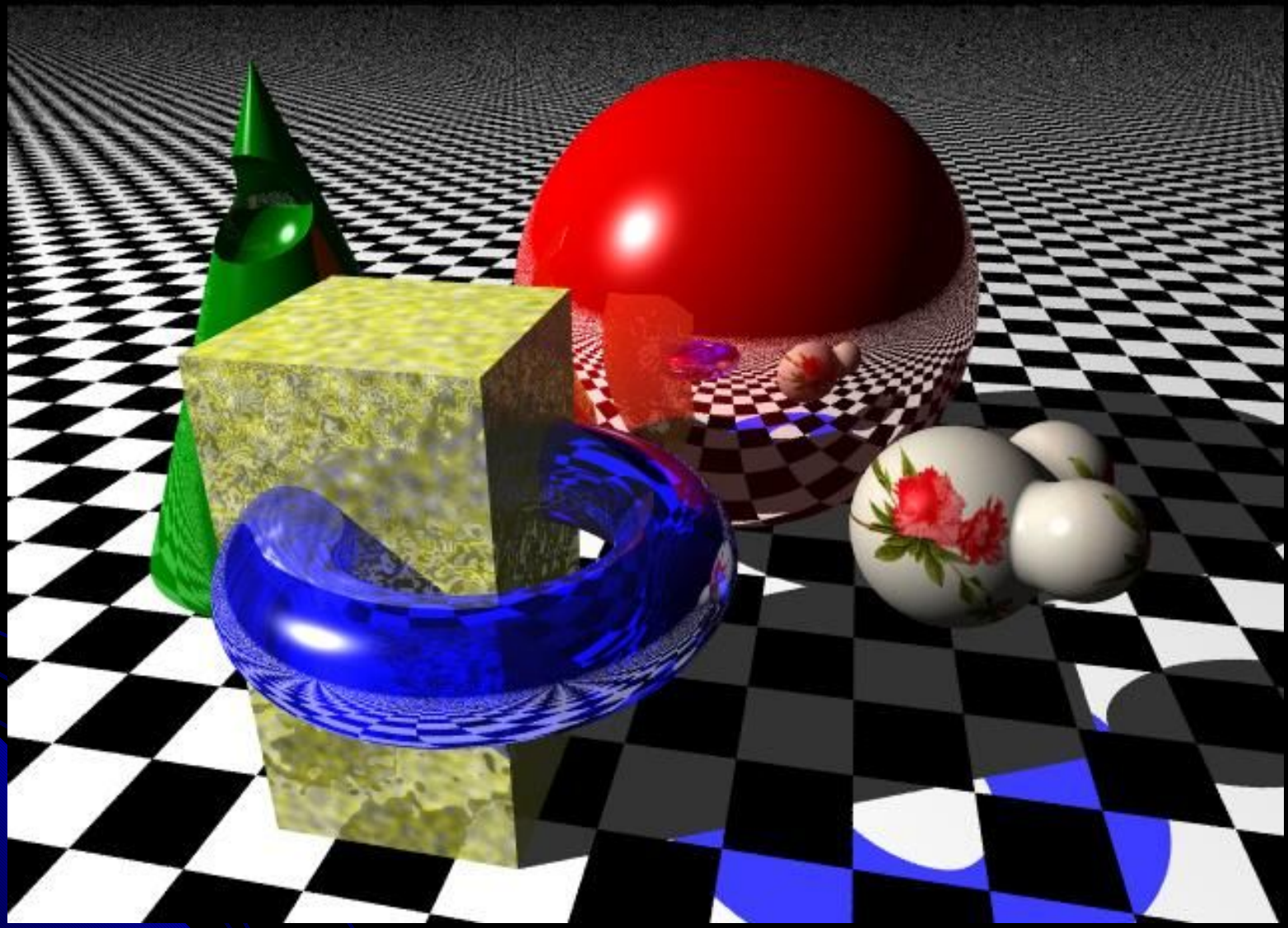
Created by David Derman – CISC 440

Fotorealismus





Created by Jan Oberlaender – CISC 640

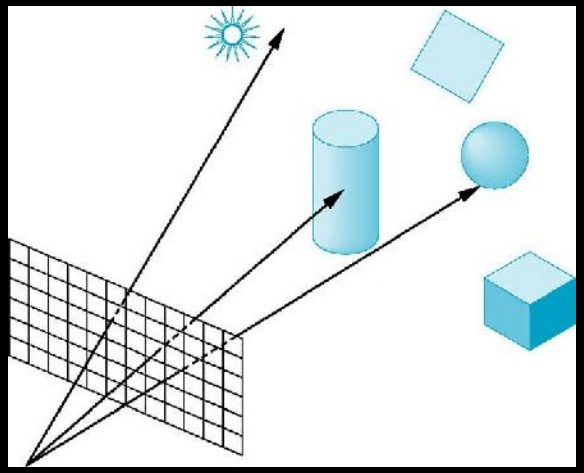


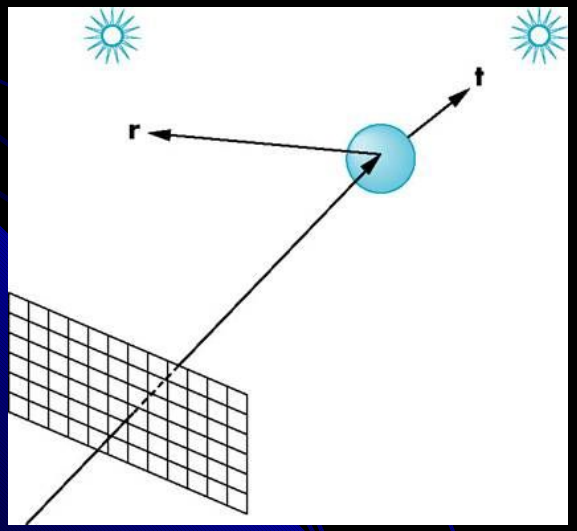
Created by Donald Hyatt
http://www.tjhsst.edu/~dhyatt/superap/povray.html

Úvod

- Co je Ray Tracing?
  - Ray Tracing je renderingová metoda založena na globálním osvětlení scény, která generuje realistické obrazy pomocí počítače.
  - V ray tracing-u, paprsek světla je sledován podél své dráhy v opačném směru.
    - Začínáme od kamery směrem ke zdroji světla a zjišťujeme stav objektů protínajících dráhu paprsku
    - Daný obrazový bod je nastaven na barvu odpovídající danému paprsku.
    - Pokud paprsek nenarazí na žádný předmět je bod nastaven na barvu pozadí.

Ray Casting/Tracing

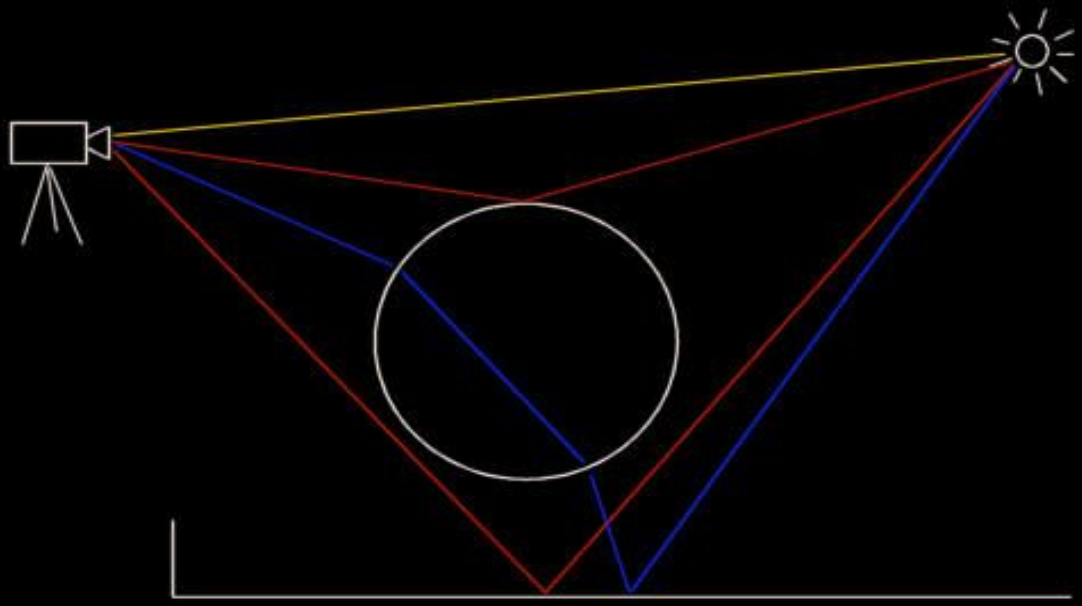




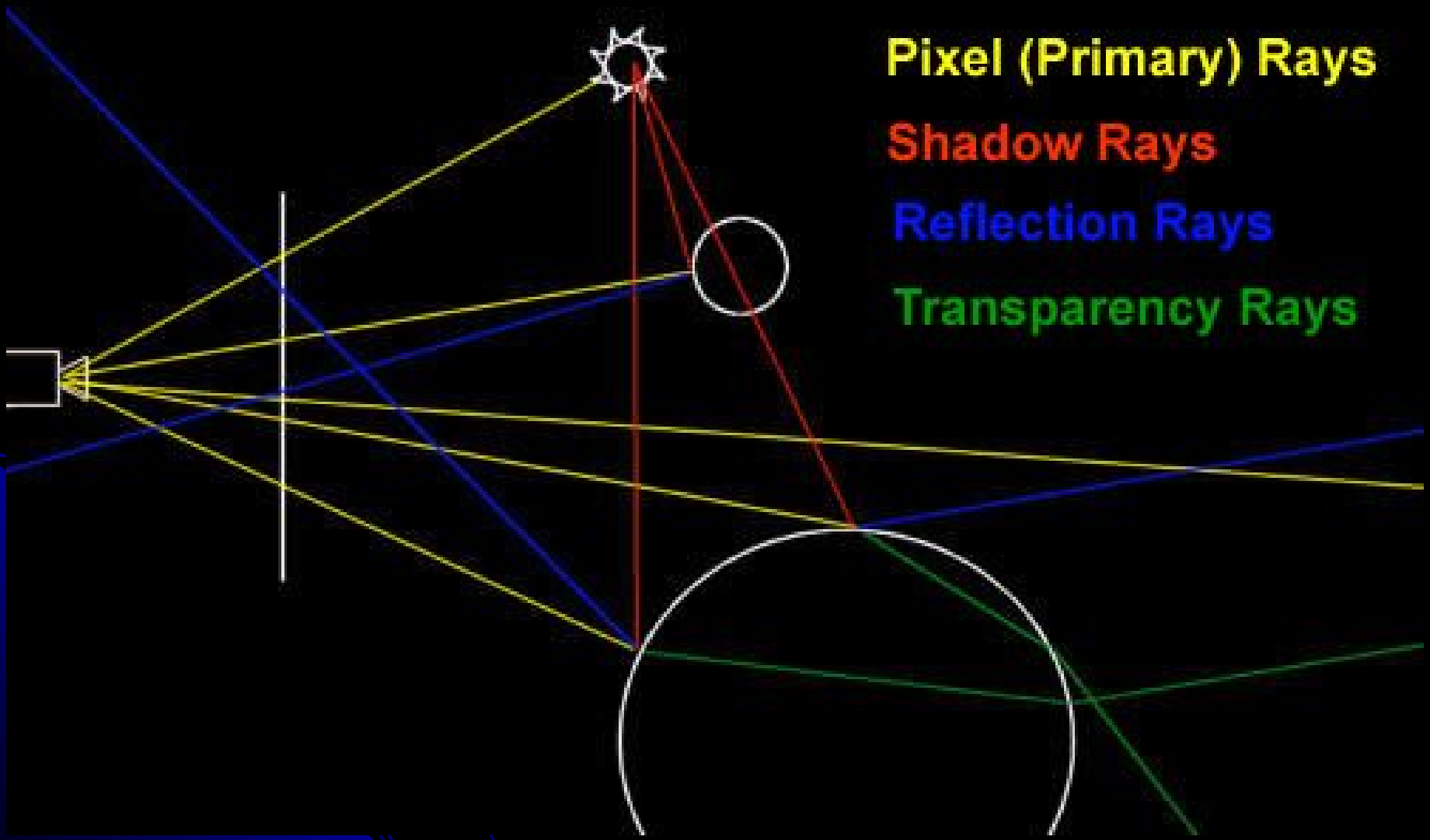
Courtesy: Angel

- Ray Casting
  - Paprsky se zastaví na prvním objektu
- Ray Tracing
  - Rekurze předcházejícího principu

Šíření světla



Typy paprsků



Algoritmus – Ray casting

define the objects and light sources in the scene
set up the camera

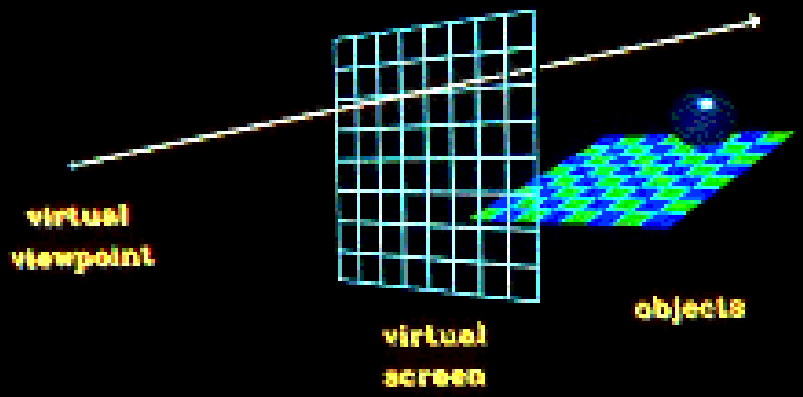
```
for(int r = 0; r < nRows; r++)
  for(int c = 0; c < nCols; c++)
  {
```

1. *Build the rc-th ray*
2. *Find all intersections of the rc-th ray with objects in the scene*
3. *Identify the intersection that lies closest to, and in front of, the eye*
4. *Compute the "hit point" where the ray hits this object, and the normal vector at that point*
5. *Find the color of the light returning to the eye along the ray from the point of intersection*
6. *Place the color in the rc-th pixel.*

```
  }
```

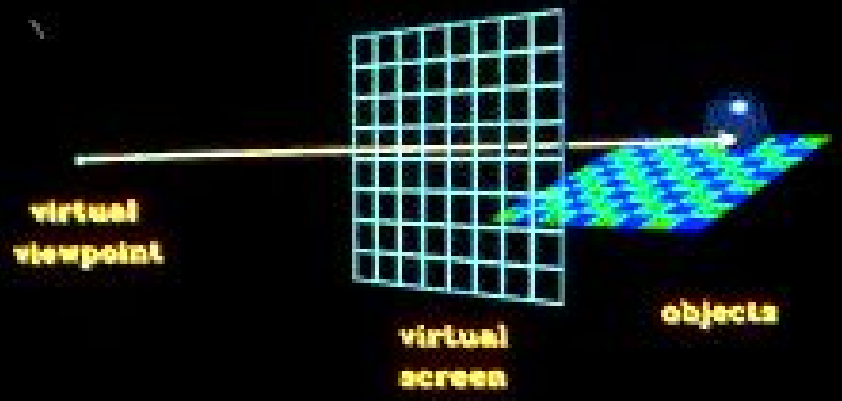
Courtesy F.S. Hill, "Computer Graphics using OpenGL"

Ray Tracing



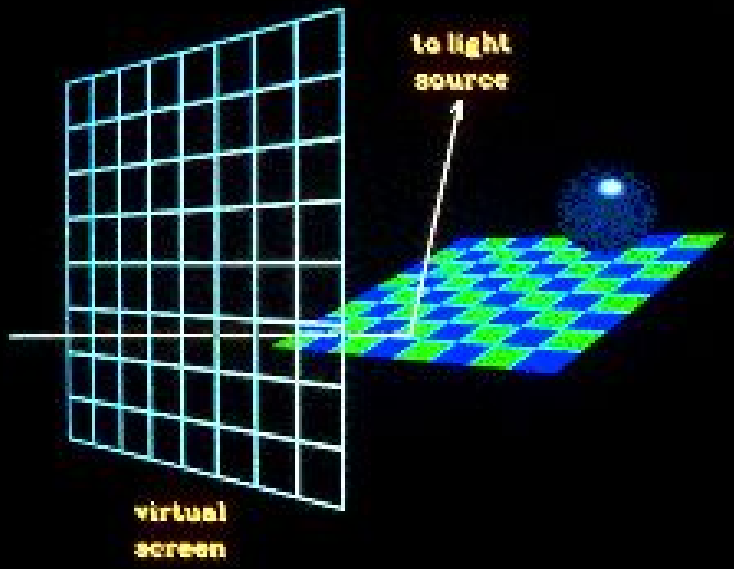
created by Michael Sweeny, et al for ACM SIGGRAPH Education slide set 1991

Ray Tracing



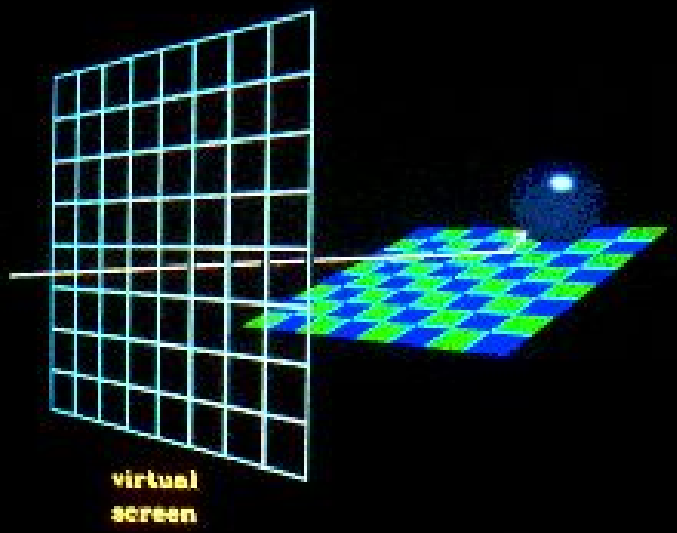
created by Michael Sweeny, et al for ACM SIGGRAPH Education slide set 1991

Ray Tracing



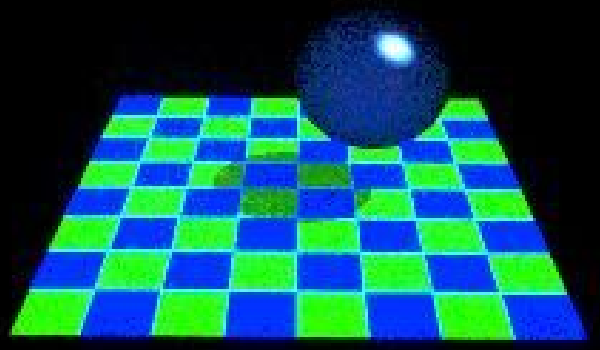
created by Michael Sweeny, et al for ACM SIGGRAPH Education slide set 1991

Ray Tracing



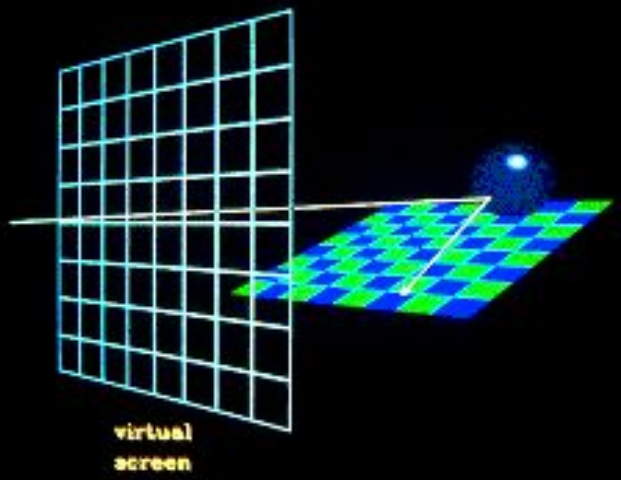
created by Michael Sweeny, et al for ACM SIGGRAPH Education slide set 1991

Ray Tracing



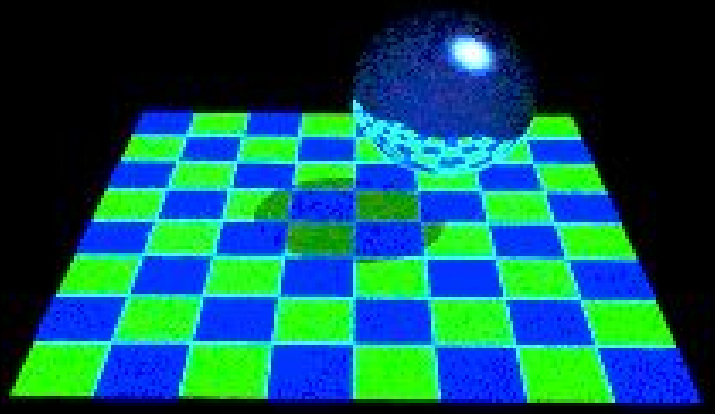
created by Michael Sweeny, et al for ACM SIGGRAPH Education slide set 1991

Ray Tracing



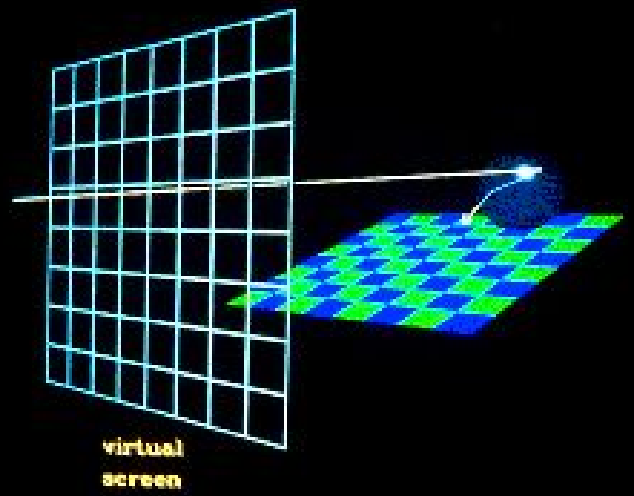
created by Michael Sweeny, et al for ACM SIGGRAPH Education slide set 1991

Ray Tracing



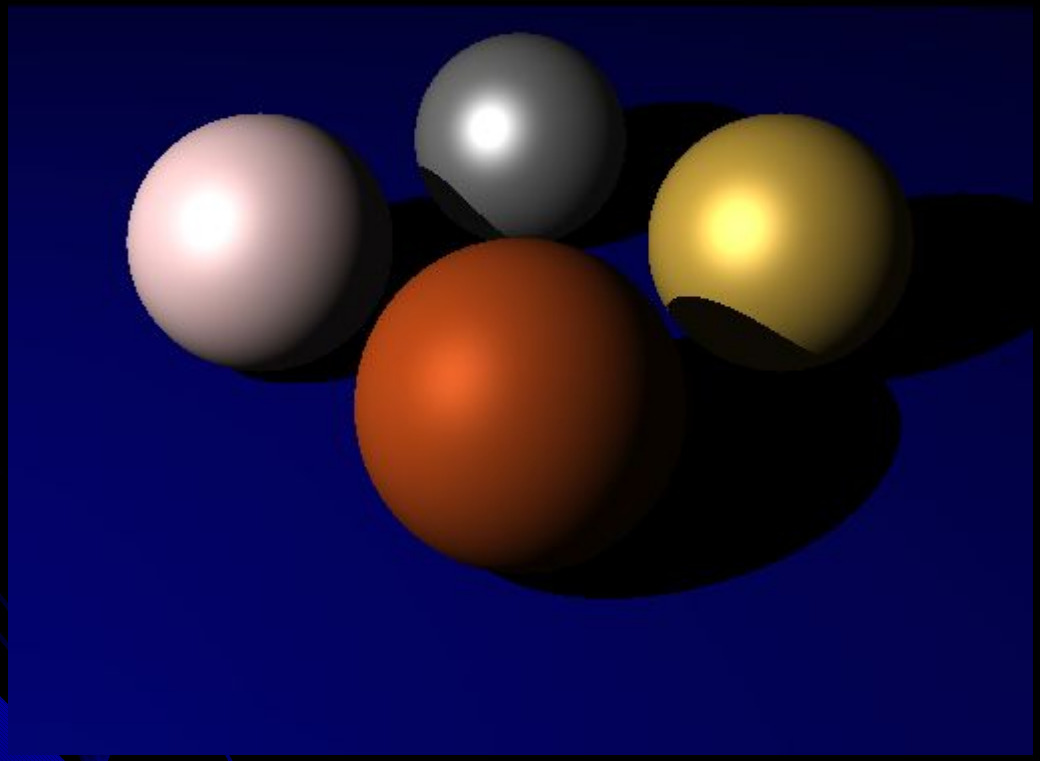
created by Michael Sweeny, et al for ACM SIGGRAPH Education slide set 1991

Ray Tracing

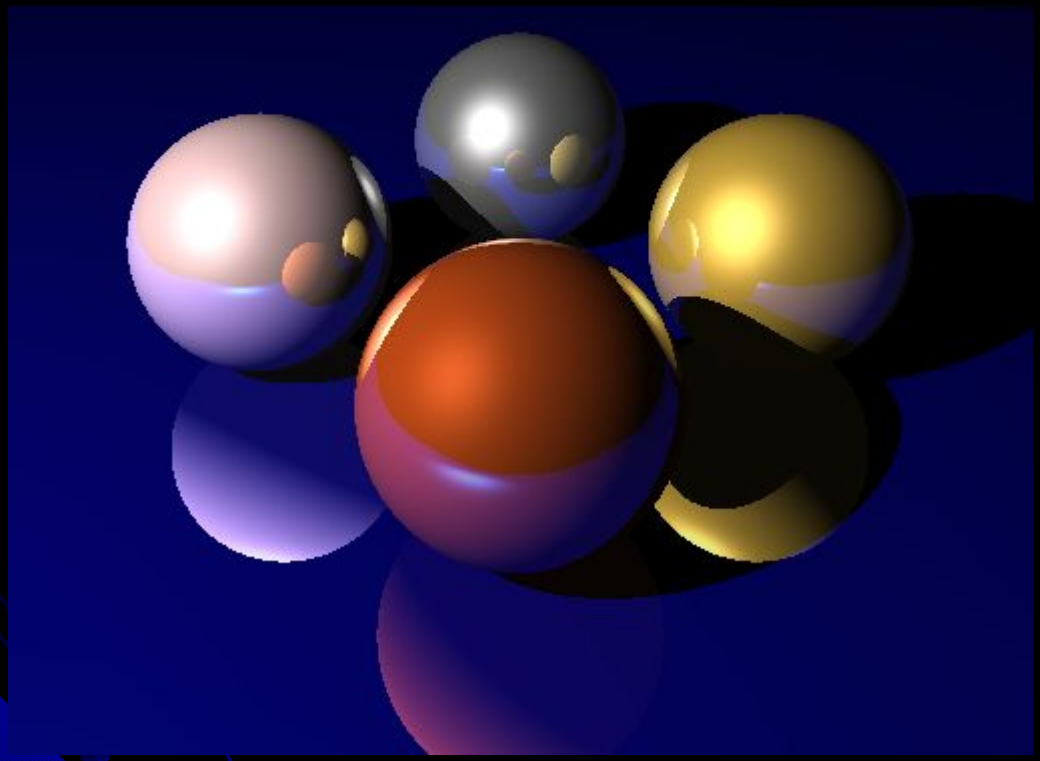


created by Michael Sweeny, et al for ACM SIGGRAPH Education slide set 1991

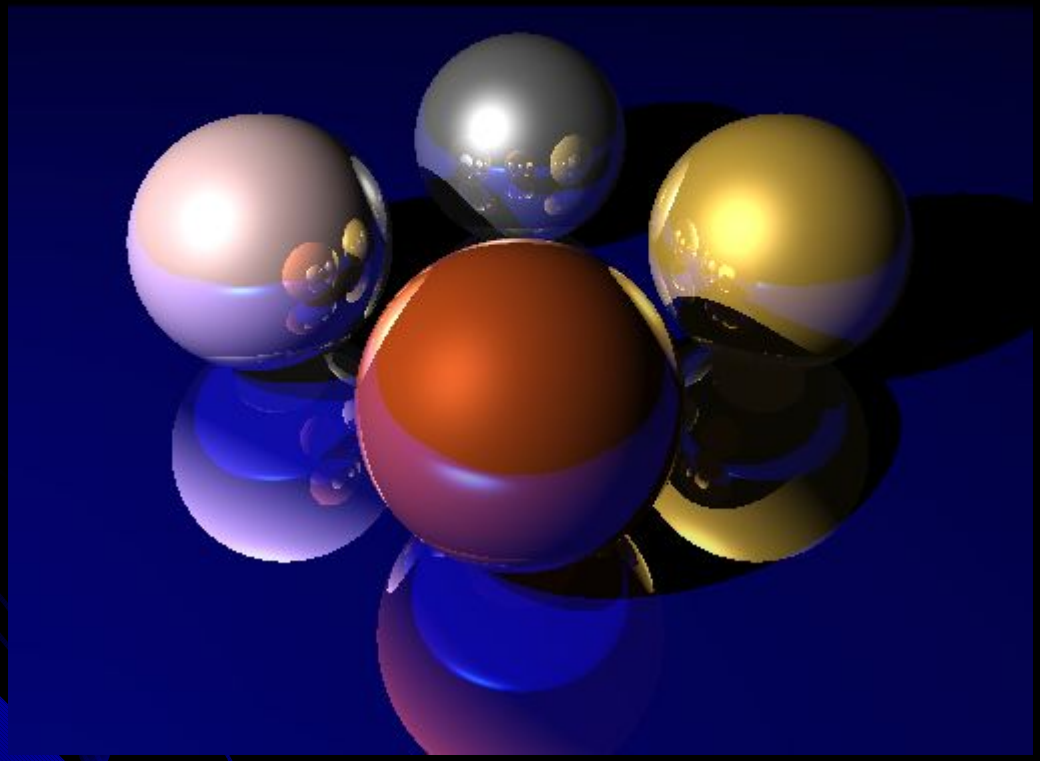
Odraz



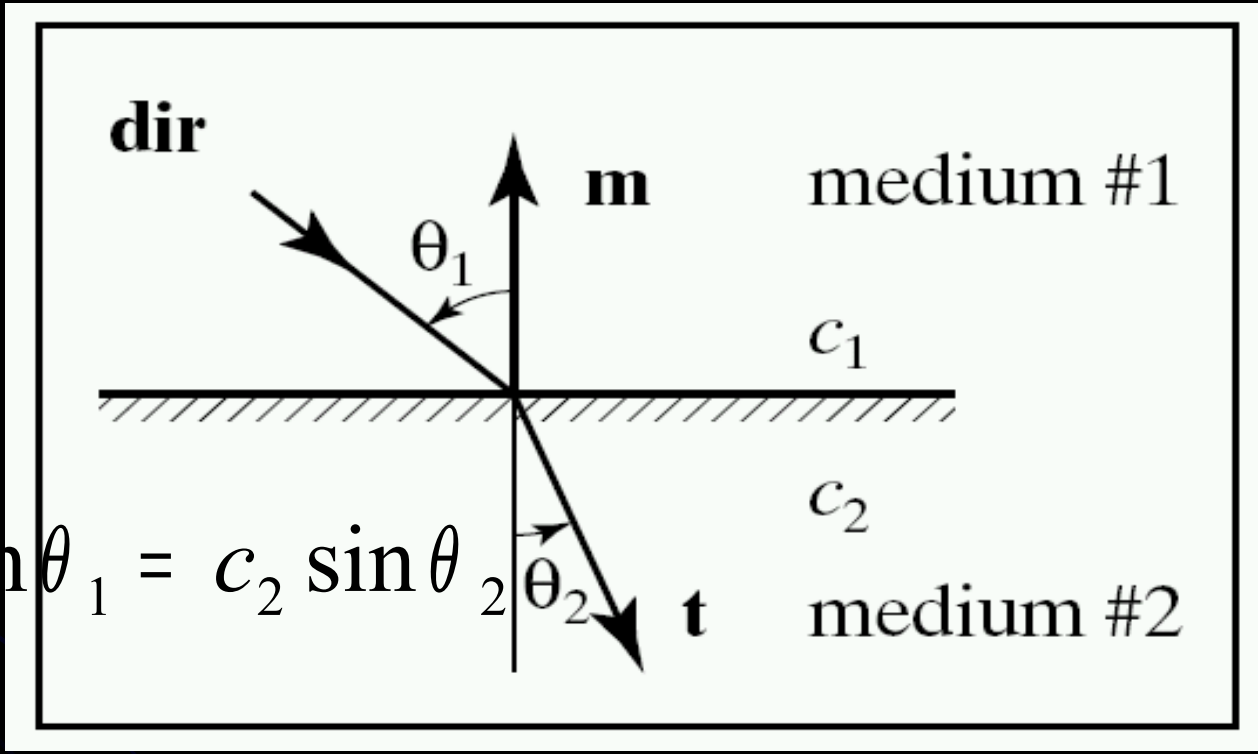
Odraz



Odraz



Lom světla

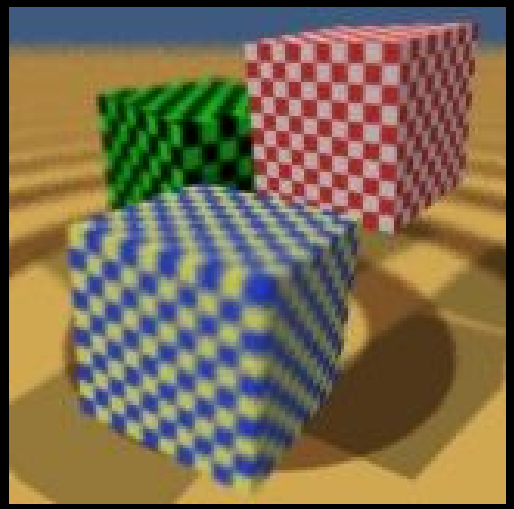


$$\sin\theta_1 = c_2 \sin\theta_2$$

Courtesy F.S. Hill, "Computer Graphics using OpenGL"

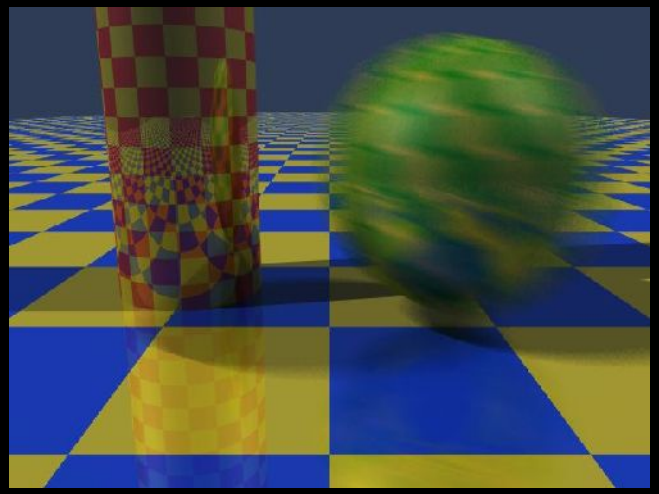
Jiné efekty

Hloubka ostrosti



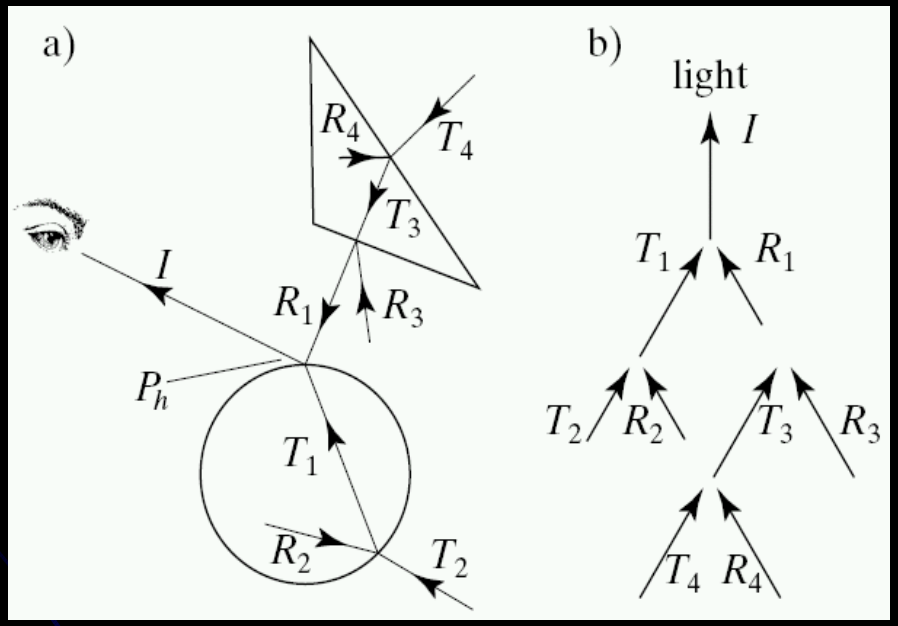
Jiné efekty

Rozmazání pohybem



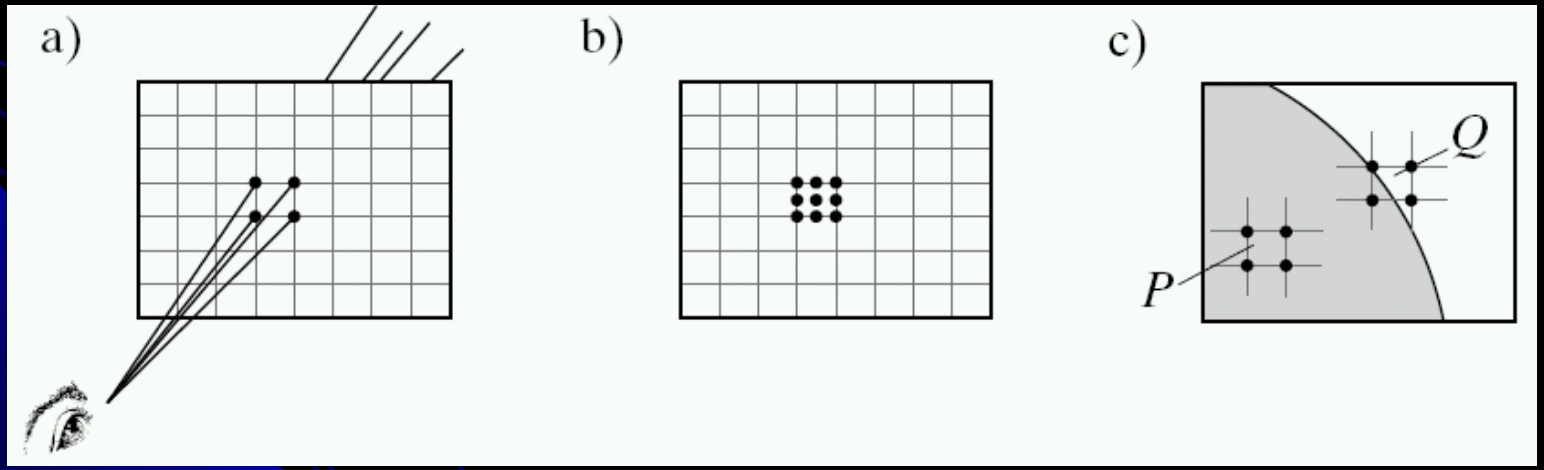
Strom světla

- Informace o paprsku sčítají



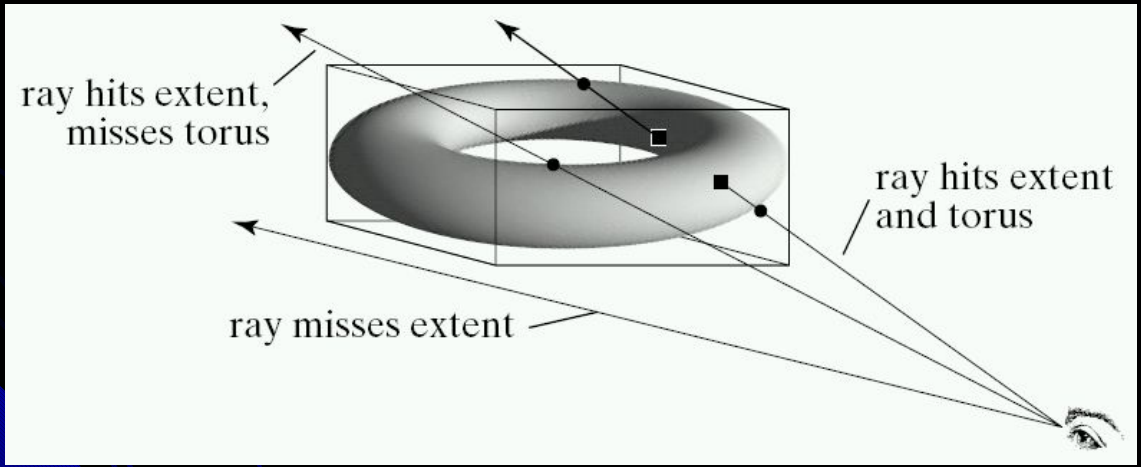
Super-sampling

- Vyhlazení hran

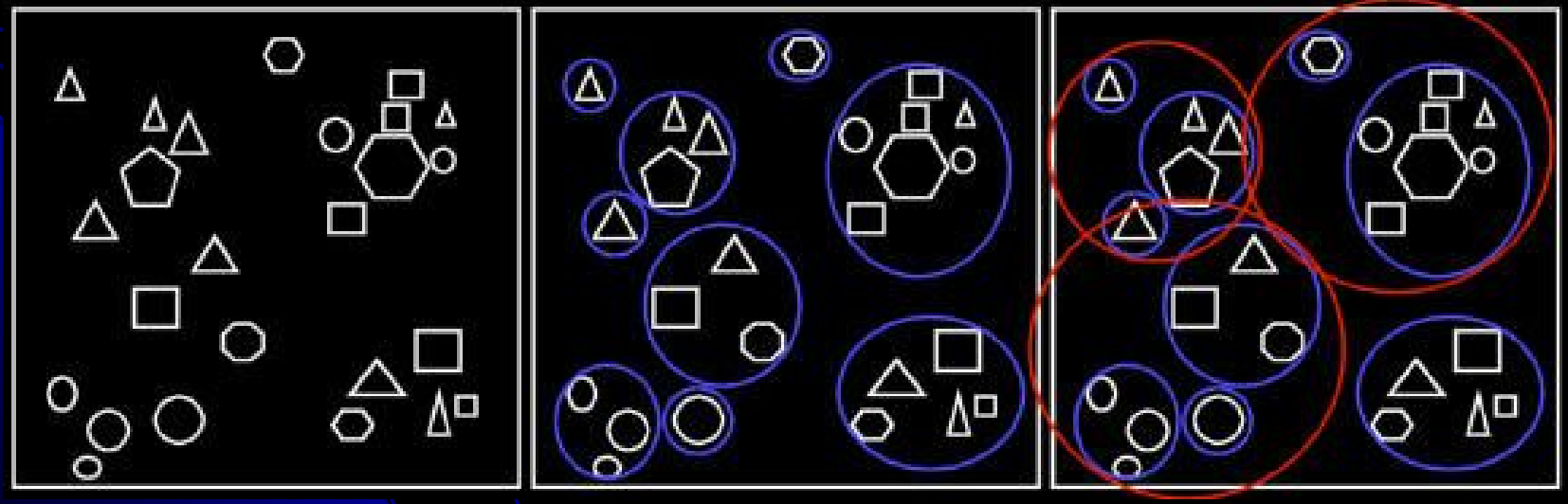


Courtesy F.S. Hill, "Computer Graphics using OpenGL"

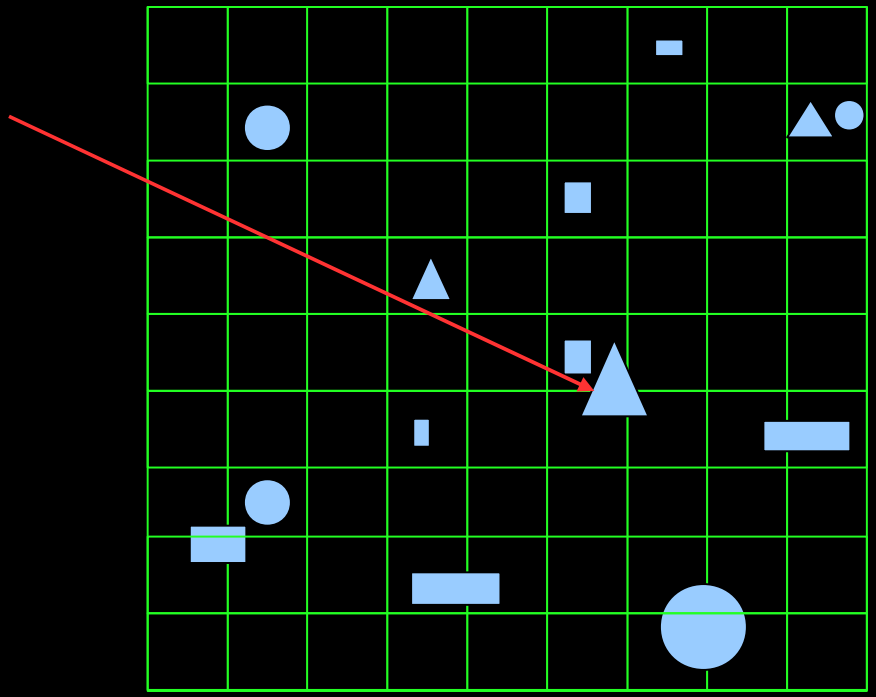
Obal


ray hits extent,
misses torus
ray hits extent
and torus
ray misses extent

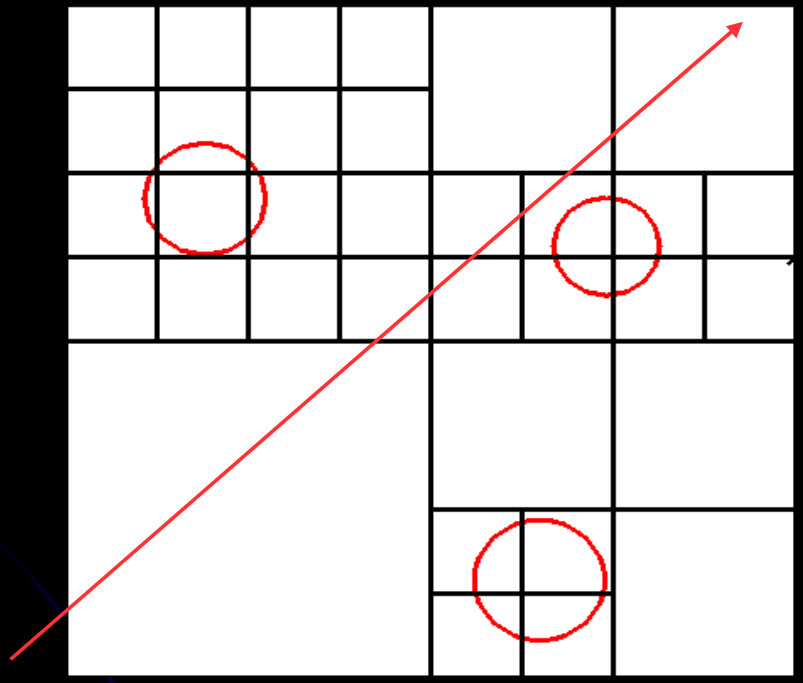
Obal skupiny objektů



Prostorové rozdělení úloh



Nerovnoměrné rozdělení na podprostory



References

- Textbooks
  - F. S. Hill, “Computer Graphics Using OpenGL”
- Commonly used ray tracing program (completely free and available for most platforms)
  - http://www.povray.org/
- Interesting Links
  - Interactive Ray Tracer – Alyosha Efros
    - http://www.cs.berkeley.edu/~efros/java/tracer/tracer.html
- Ray Tracing explained
  - http://www.geocities.com/jamisbuck/raytracing.html
  - http://www.siggraph.org/education/materials/HyperGra

Structure Visualization Tools

Written by James Coleman
Presented by Xiang Zhou

Structure Visualization

- One of the primary activities in proteomics R&D is determining and Visualizing the 3D structure of proteins in order to find where drugs might modulate their activity.
- Other activities include identifying all of the proteins produced by a given cell or tissue and determining how these proteins interact.

- BIOINFORMATICS COMPUTING, p.186, Bryon Bergeron, M.D., Prentice Hall 2002

Some Common Tools

- 100's of visualization tools have been developed in bioinformatics.
- Many are specific to hardware such as microarray devices.
- Shareware utilities for PC's
  - PDB Viewer, WebMol, RasMol, Protein Explorer, Cn3D
  - VMD, MolMol, MidasPlus, Pymol, Chime, Chimera

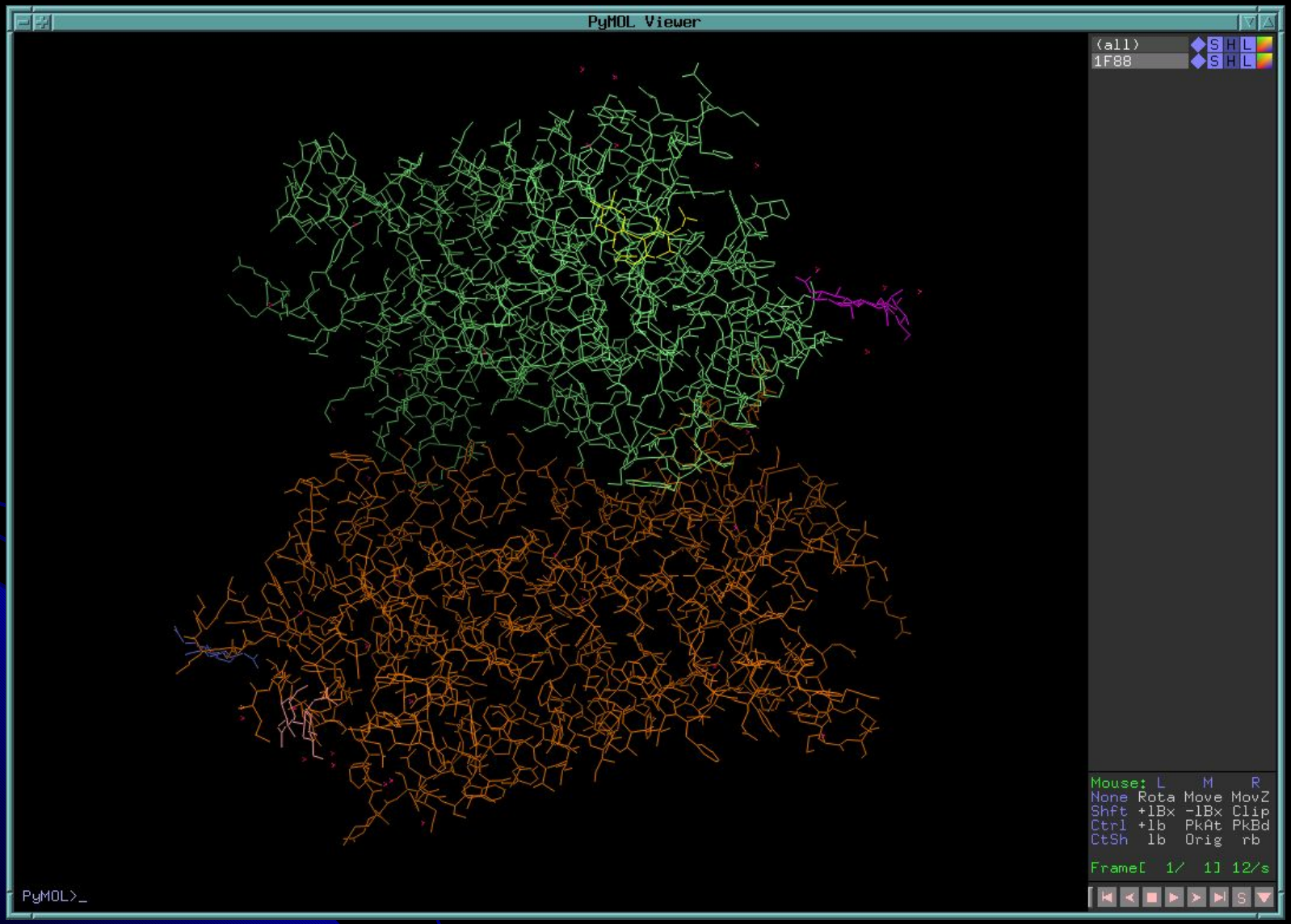
Application Feature Summary

Feature	RasMol	Cn3D	PyMol	SWISS-PDBViewer	Chimera
Architecture	Stand-Alone	Plug-in	Web-Enabled	Web-enabled	Web-enabled
Manipulation Power	Low	High	High	High	High
Hardware Requirements	Low/Moderate	High	High	Moderate	High
Ease of Use	High; command line	Moderate	Moderate	High	Moderate;GUI +command line
Special Features	Small Size; easy install	Powerful GUI	GUI; ray tracing	Powerful GUI	GUI; collaboration
Output Quality	Moderate	Very high	High	High	Very high
Documentation	Good	Good	Limited	Good	Very good
Support	Online; Users groups	Online; Users groups	Online; Users groups	Online; Users groups	Online; Users groups
Speed	High	Moderate	Moderate	Moderate	Moderate/Slow
OpenGL Support	Yes	Yes	Yes	Yes	Yes

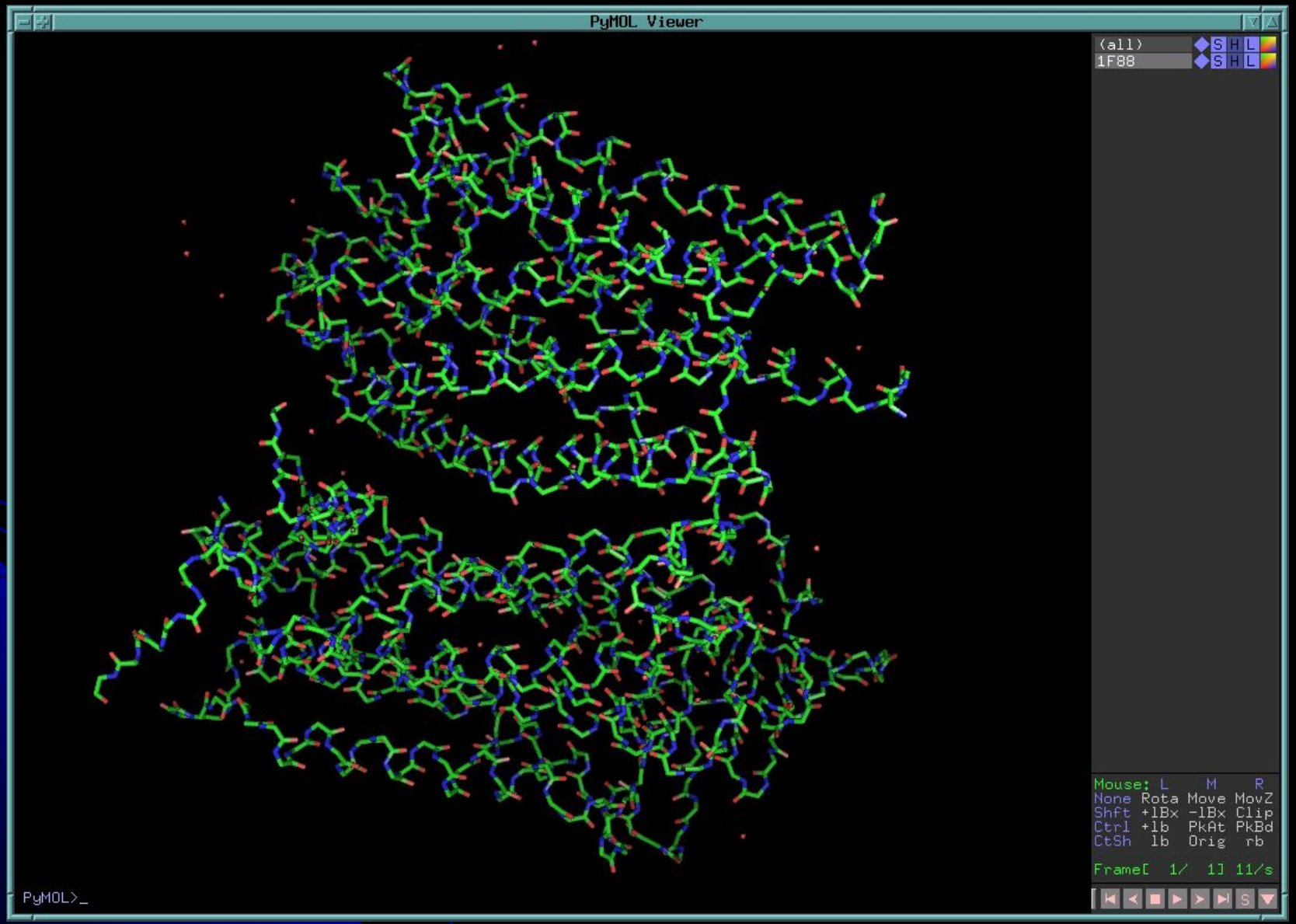
Molecule Representations

Wireframe	Bonds and Bond Angles	
Ball and Stick	Shows Atoms, Bonds and Bonds Angles	
Ribbon diagrams	Shows Secondary Structure	
Van der Waals surface Diagram	Shows Atomic Volumes	
Backbone	Shows Overall Molecular Structure	

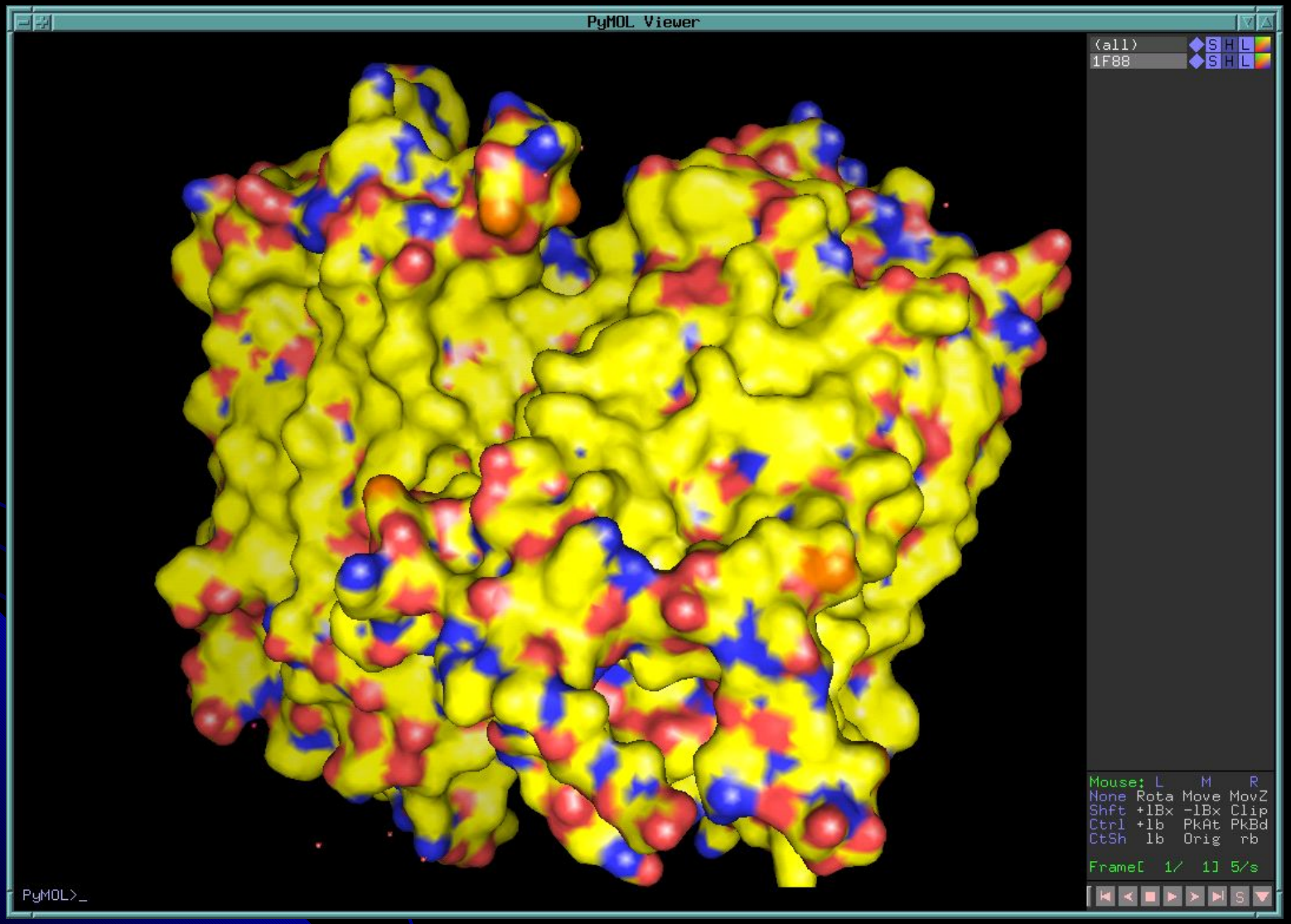
Wireframe used to show individual chains:



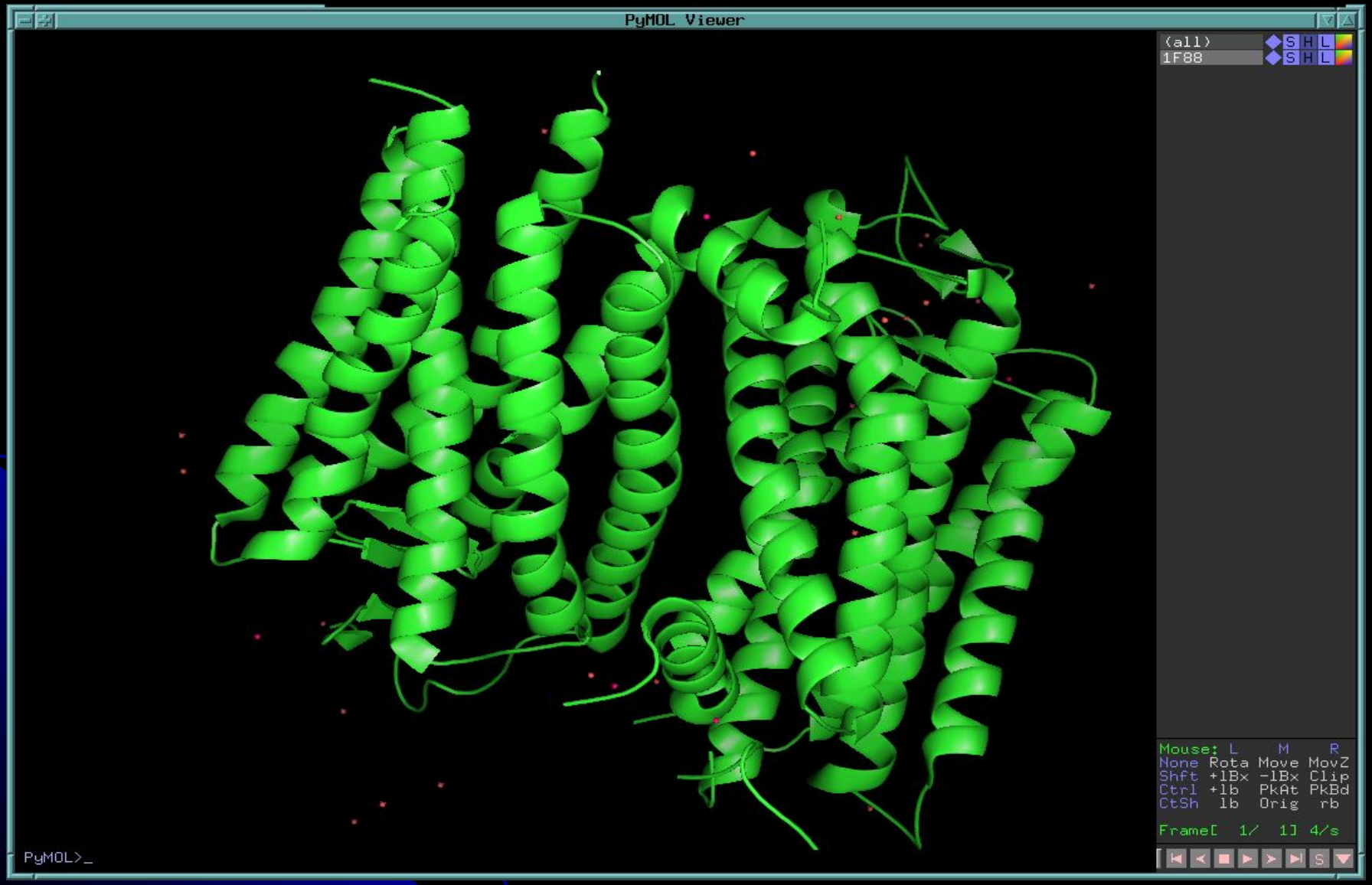
Stick view showing atoms and bonds:



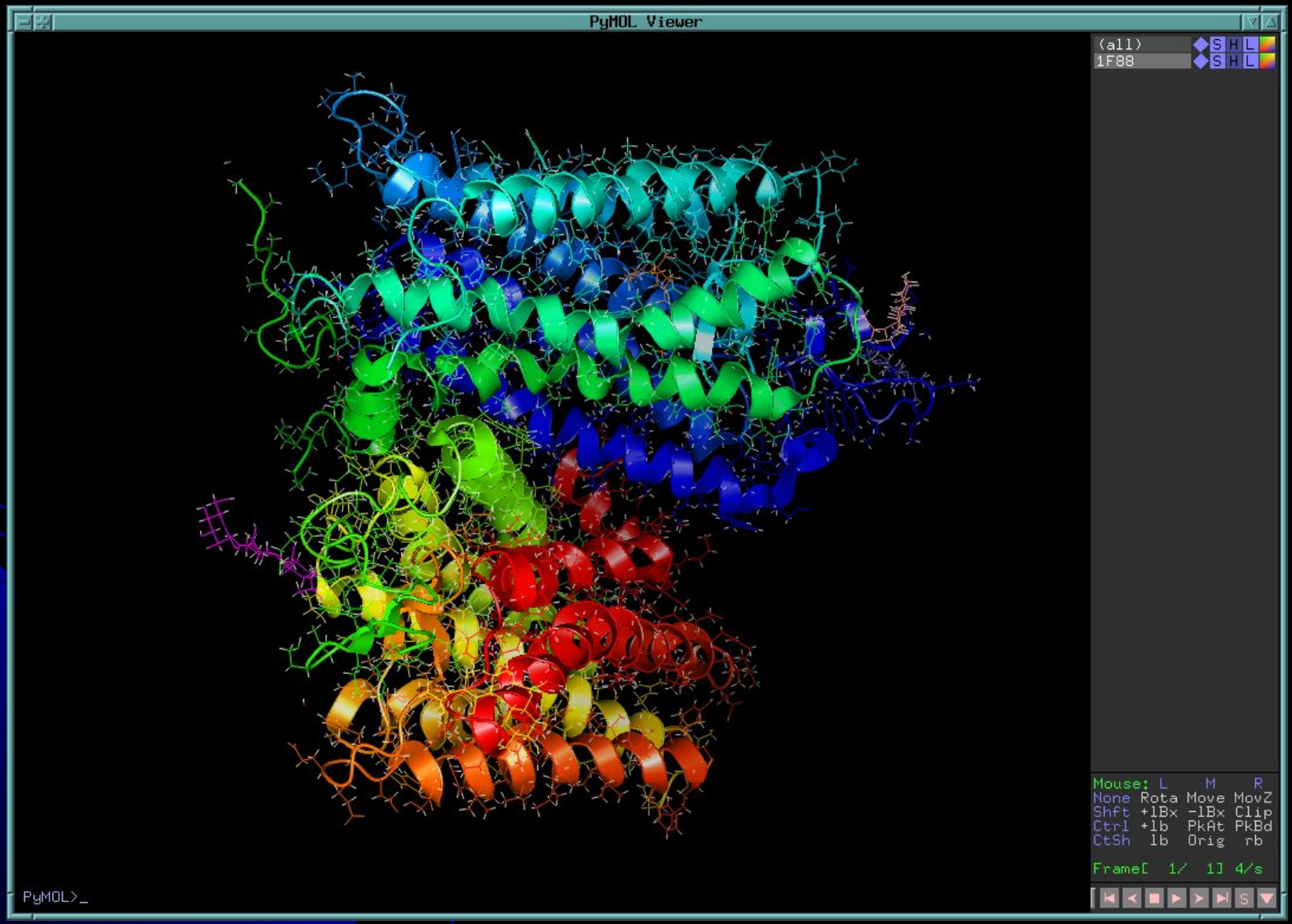
Surface View showing surface fields:



Ribbon view of secondary structure:

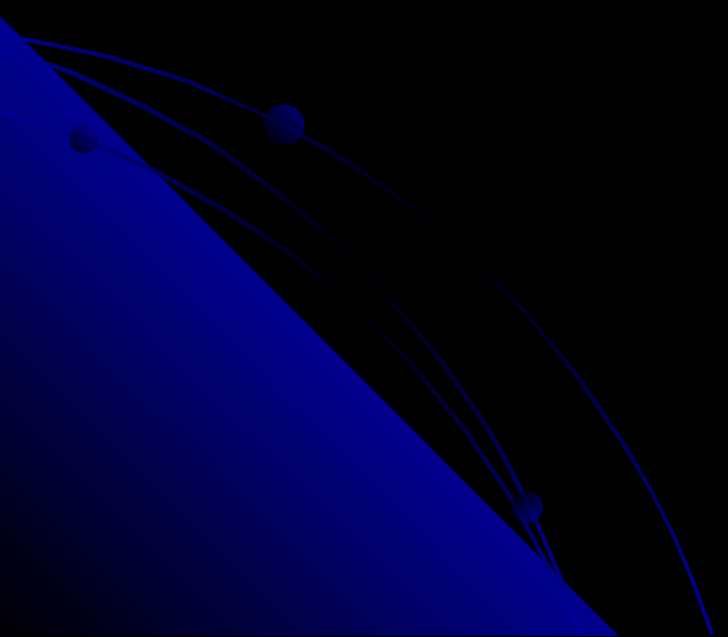


Distinct geometrical features by color:



Other properties that can be Visualized

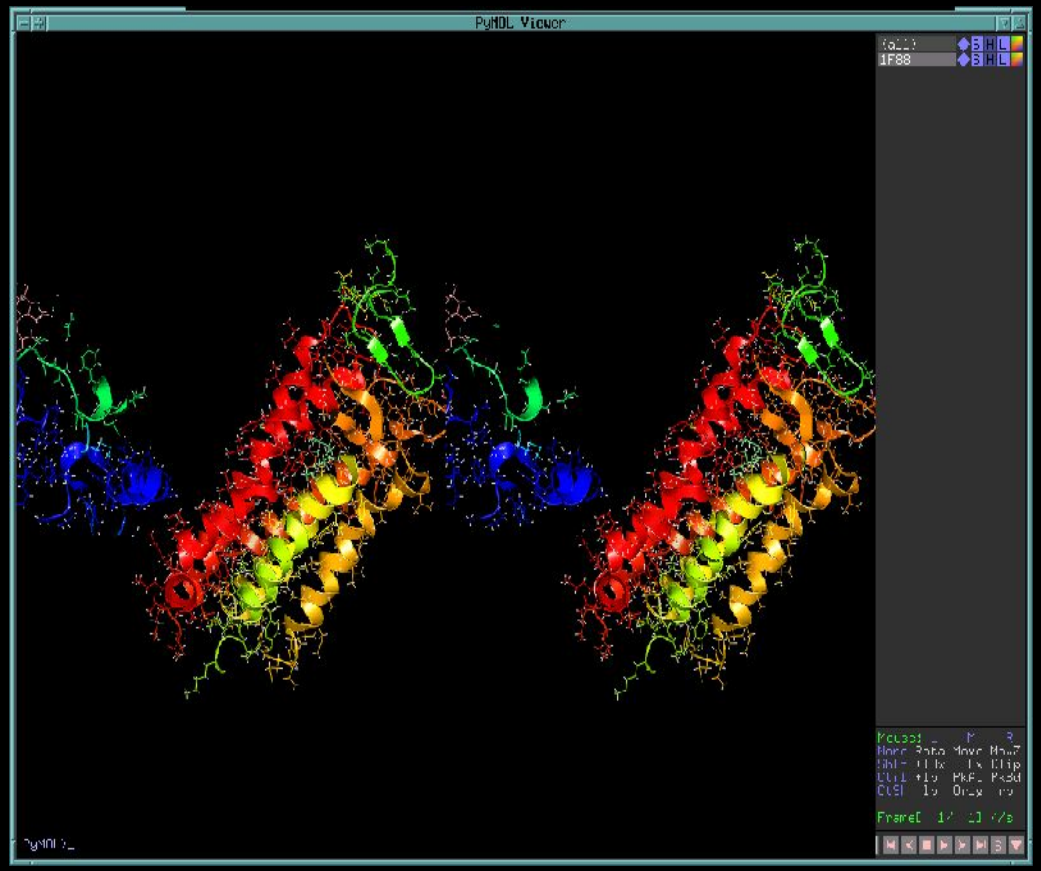
- MolMol supports the display of electrostatic potentials across a protein molecule.
- MidasPlus (a predecessor of Chimera) allows for the editing of sequences visually to see the effects of point mutations.



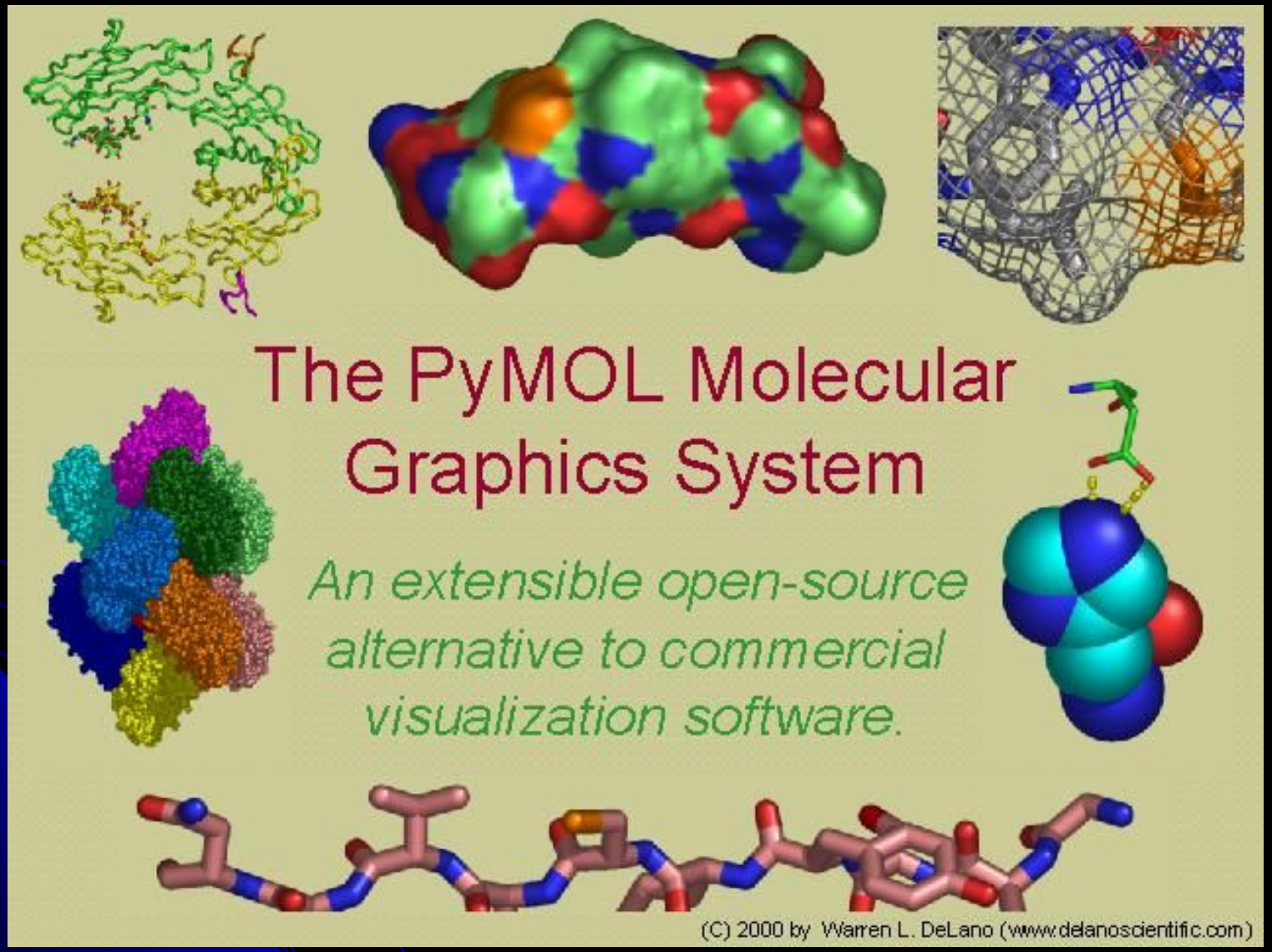
For Protein interactions, we need a metaphor that reveals dynamics

- Haptic Joystick: Provides force feedback when user manipulates a molecule near another one.
- 3D Goggles combined with haptic gloves to feel electrostatic potentials and see tertiary structure dynamics.
- PyMol provides scripting that can produce a movie in 3D of the geometrical relationship between multiple proteins.

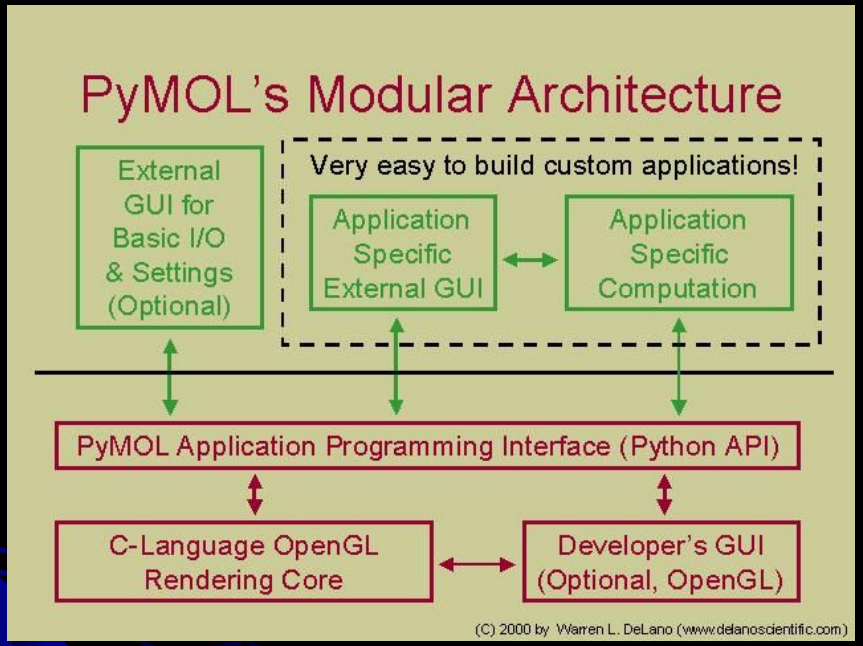
Stereo view of interaction of two proteins. Scripting allows for the movement of individual molecules creating a movie.

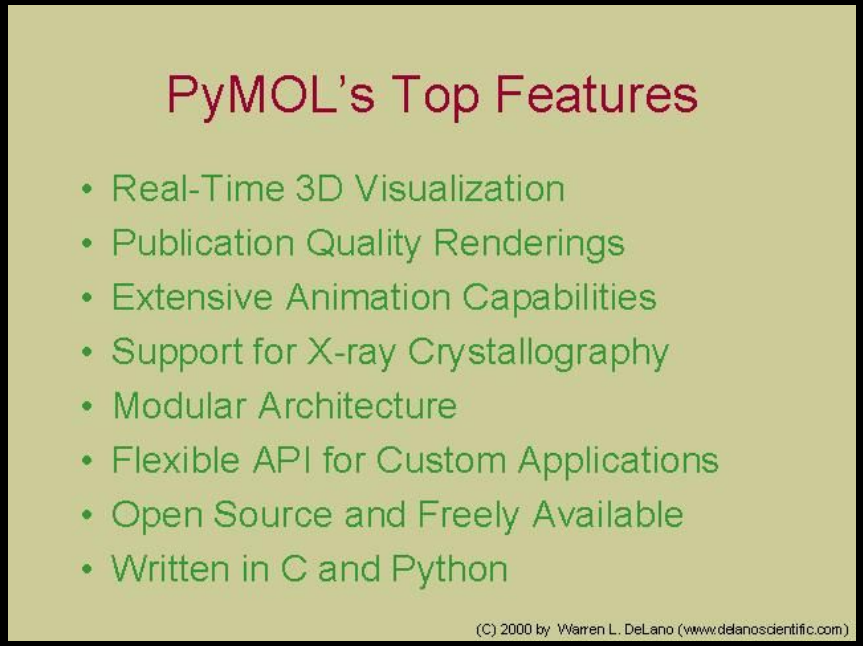


The PYMOL Molecular Graphic System



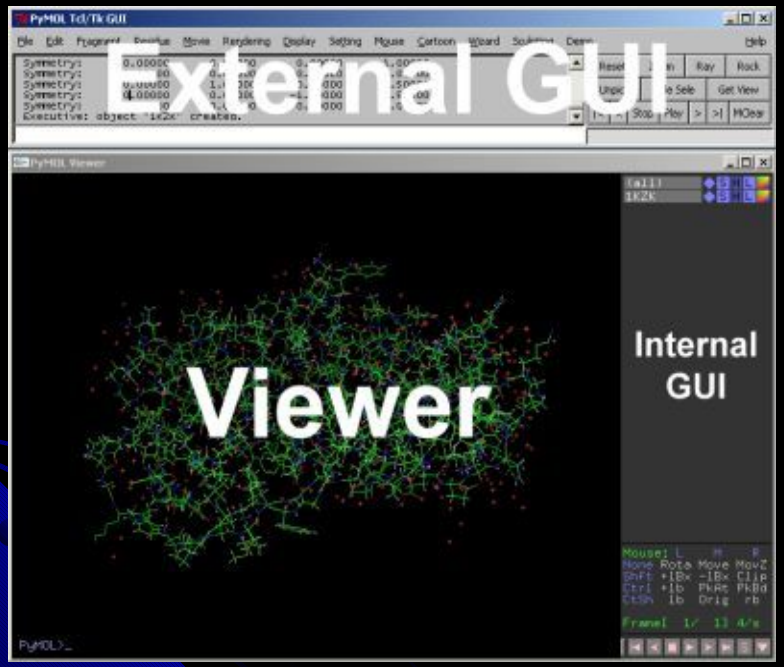
PYMOL

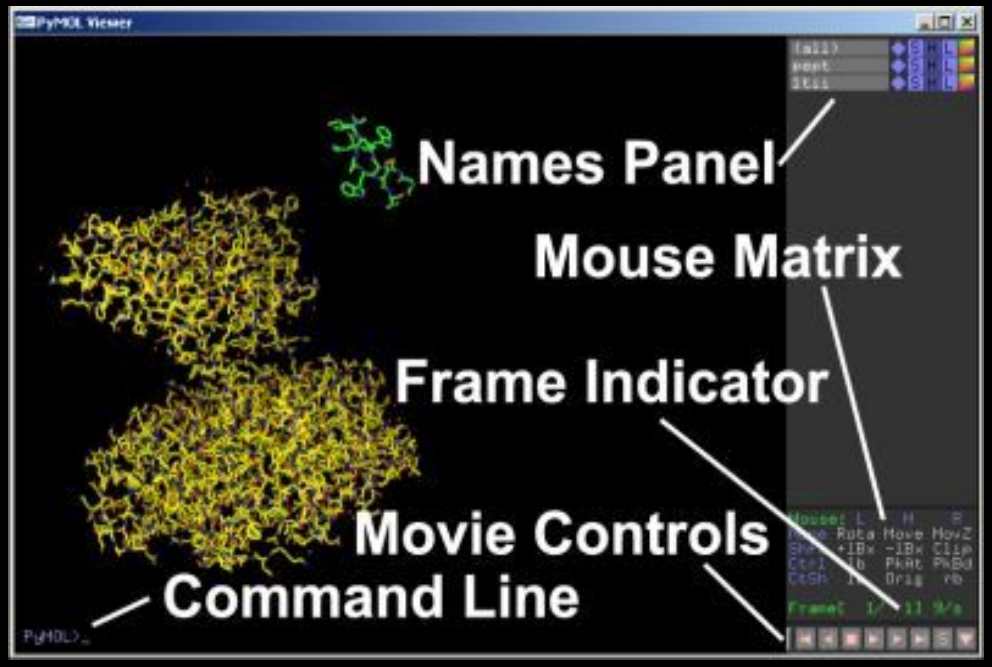




- It supports Windows, Macintosh, Linux, Solaris, IRIX
- Freely available @ `http://www.pymol.org/`

Basic modules





PDB file

Residue #

Coordinates

Temp. factor

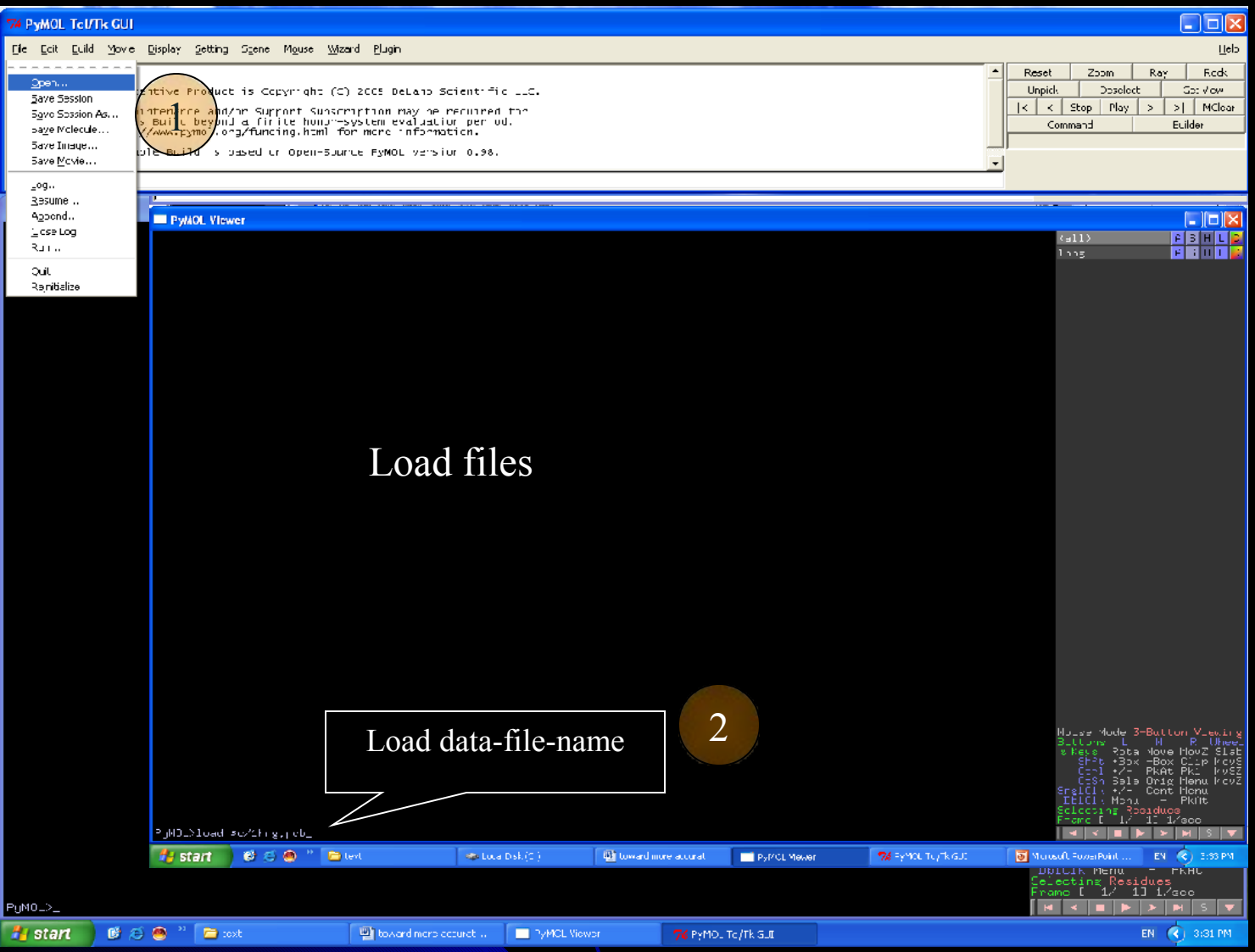
```
ATOM      3   C    ASP  A    2      28.867 144.134  11.673  1.00 98.50
ATOM      4   O    ASP  A    2      28.431 143.093  11.149  1.00 98.95
ATOM      5   CB   ASP  A    2      27.061 145.944  12.178  1.00 98.95
ATOM      6   CG   ASP  A    2      27.416 146.580  13.526  1.00 98.24
ATOM      7   OD1  ASP  A    2      28.567 146.469  14.018  1.00 97.02
ATOM      8   OD2  ASP  A    2      26.500 147.204  14.110  1.00 95.61
ATOM      9   N    SER  A    3      29.866 144.144  12.570  1.00 96.71
ATOM     10   CA   SER  A    3      30.623 142.954  13.011  1.00 91.16
ATOM     11   C    SER  A    3      30.990 142.916  14.503  1.00 87.50
ATOM     12   O    SER  A    3      31.008 143.981  15.129  1.00 92.21
ATOM     13   CB   SER  A    3      31.935 142.827  12.176  1.00 88.43
ATOM     14   OG   SER  A    3      32.387 144.045  11.589  1.00 85.92
ATOM     15   N    GLY  A    4      31.281 141.750  15.093  1.00 82.44
ATOM     16   CA   GLY  A    4      31.687 141.634  16.495  1.00 62.79
ATOM     17   C    GLY  A    4      32.756 140.563  16.628  1.00 49.98
```

Chain #

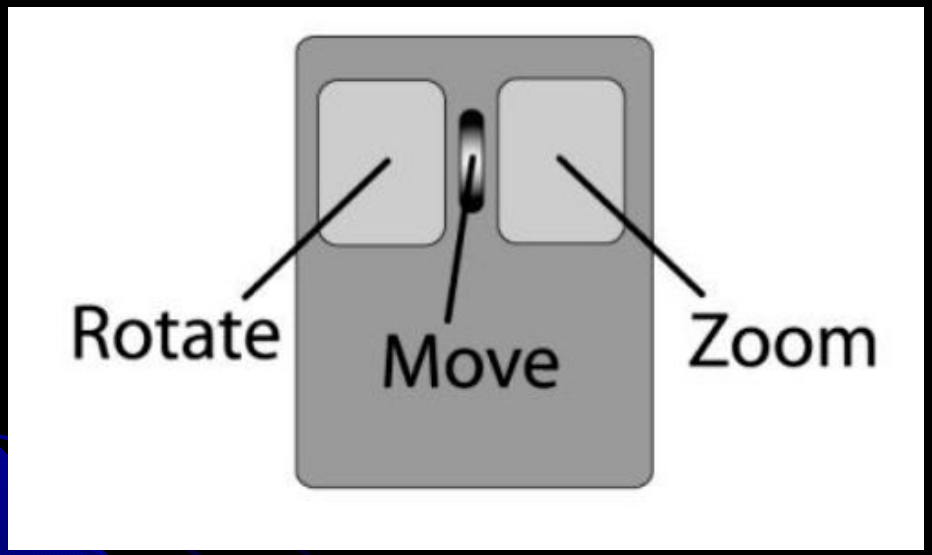
Load your pdb file

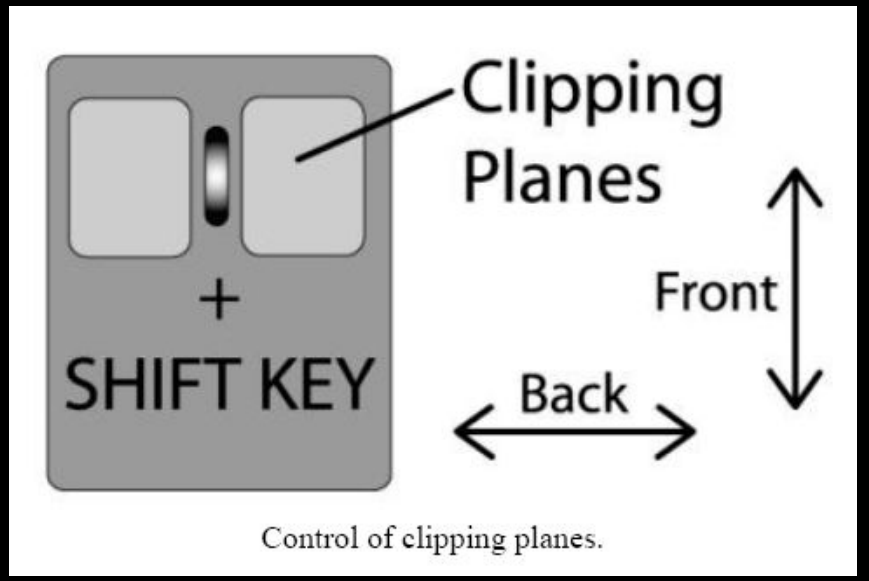
External GUI: file-open

Command lines: load file-path (e.g., **`load $c/1hng.pdb`**)



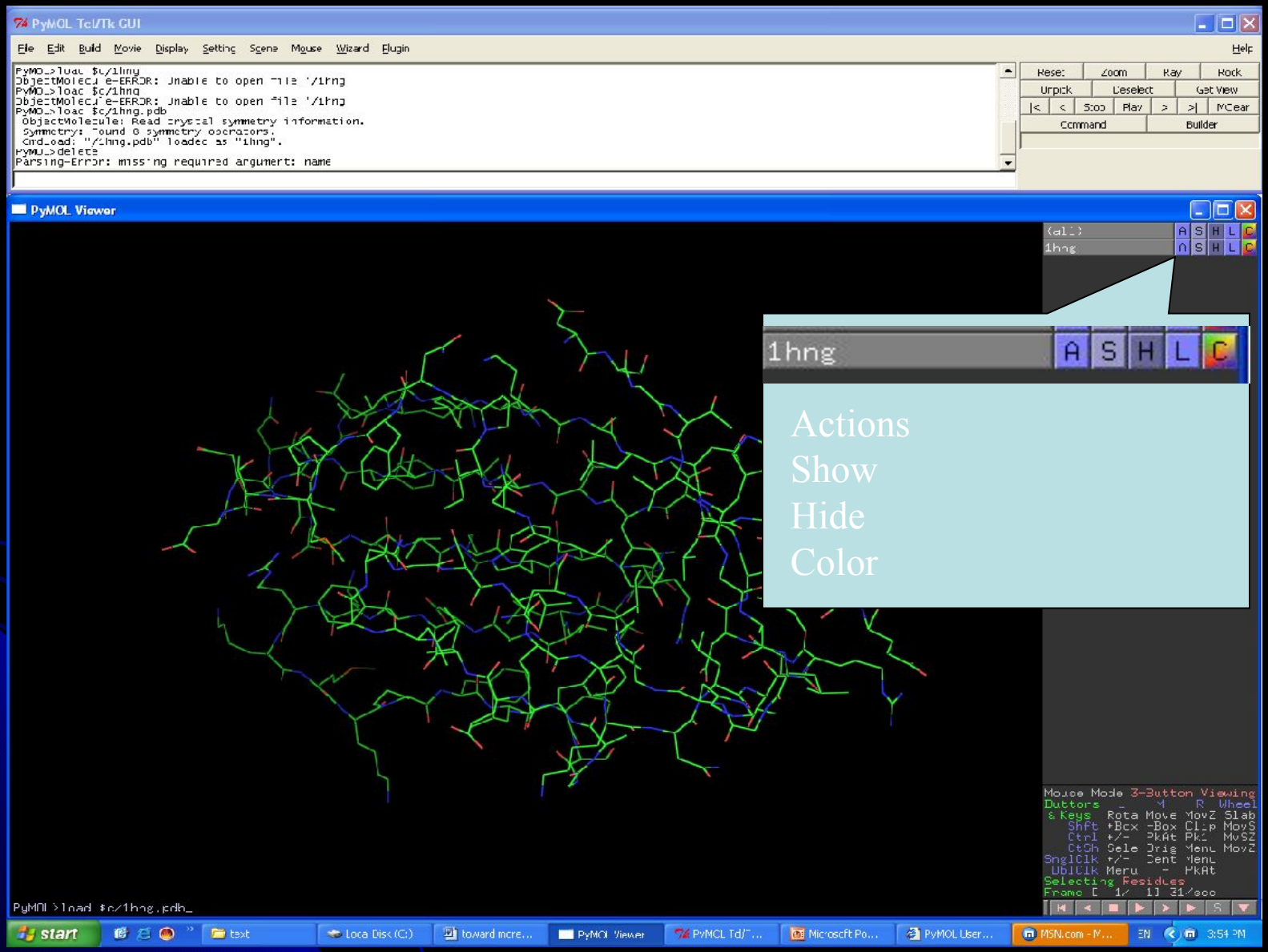
Manipulate the view by mouse



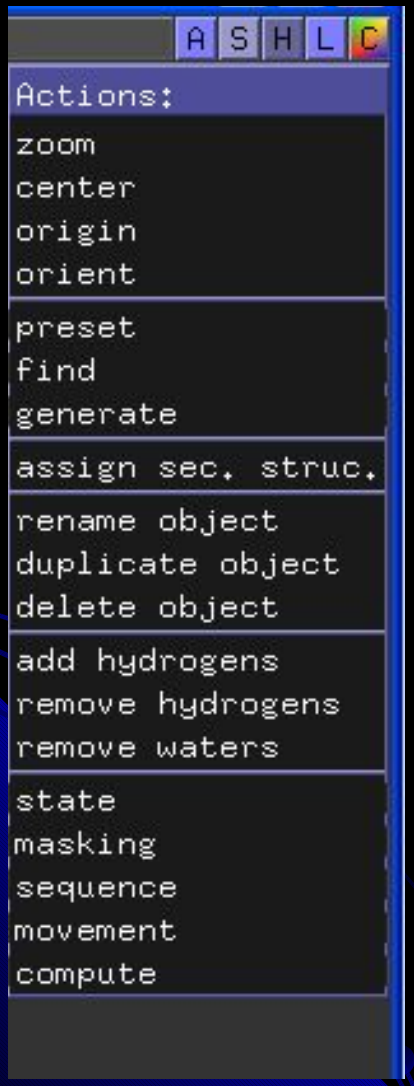


Control of clipping planes.

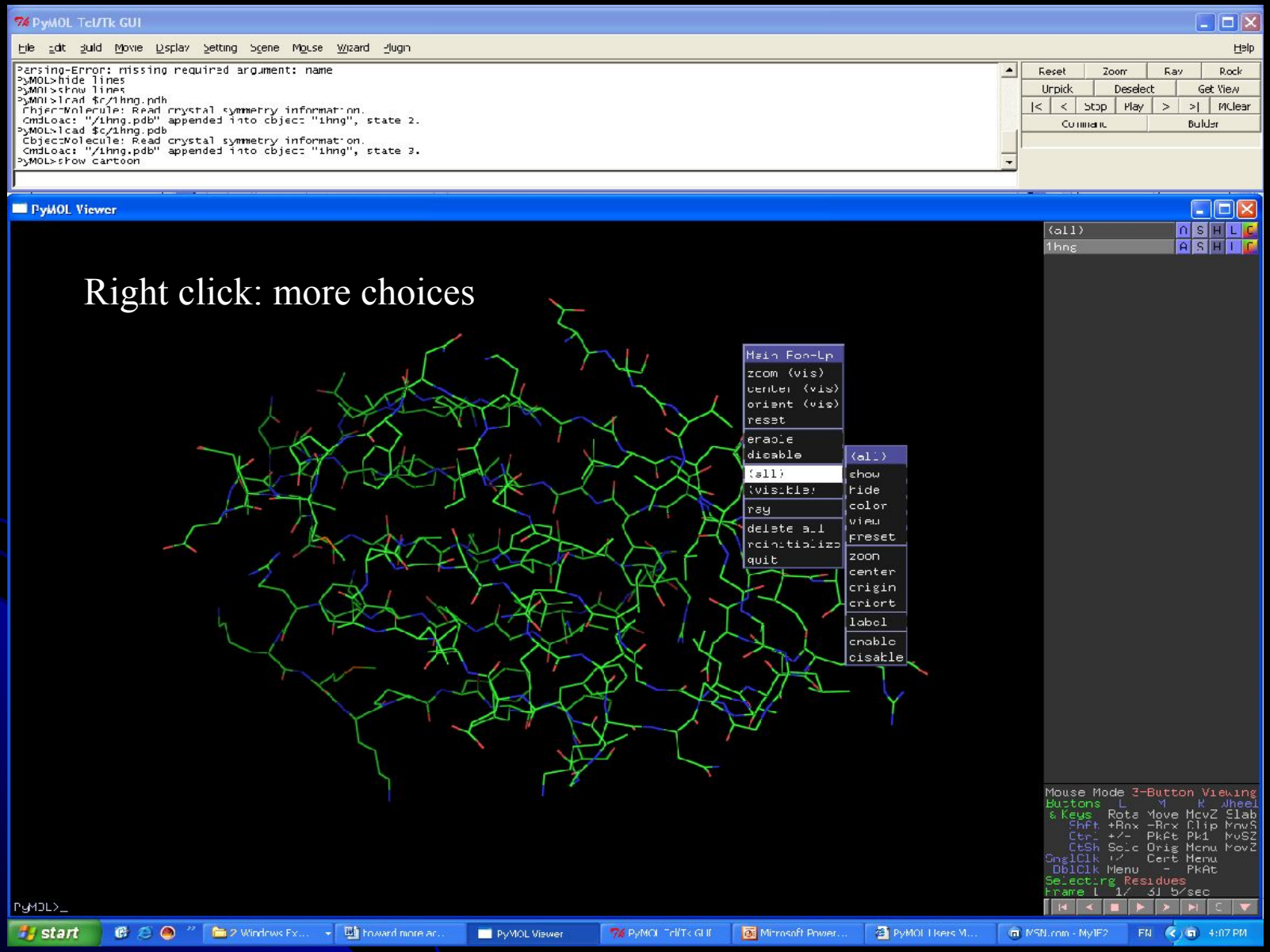
Important panels



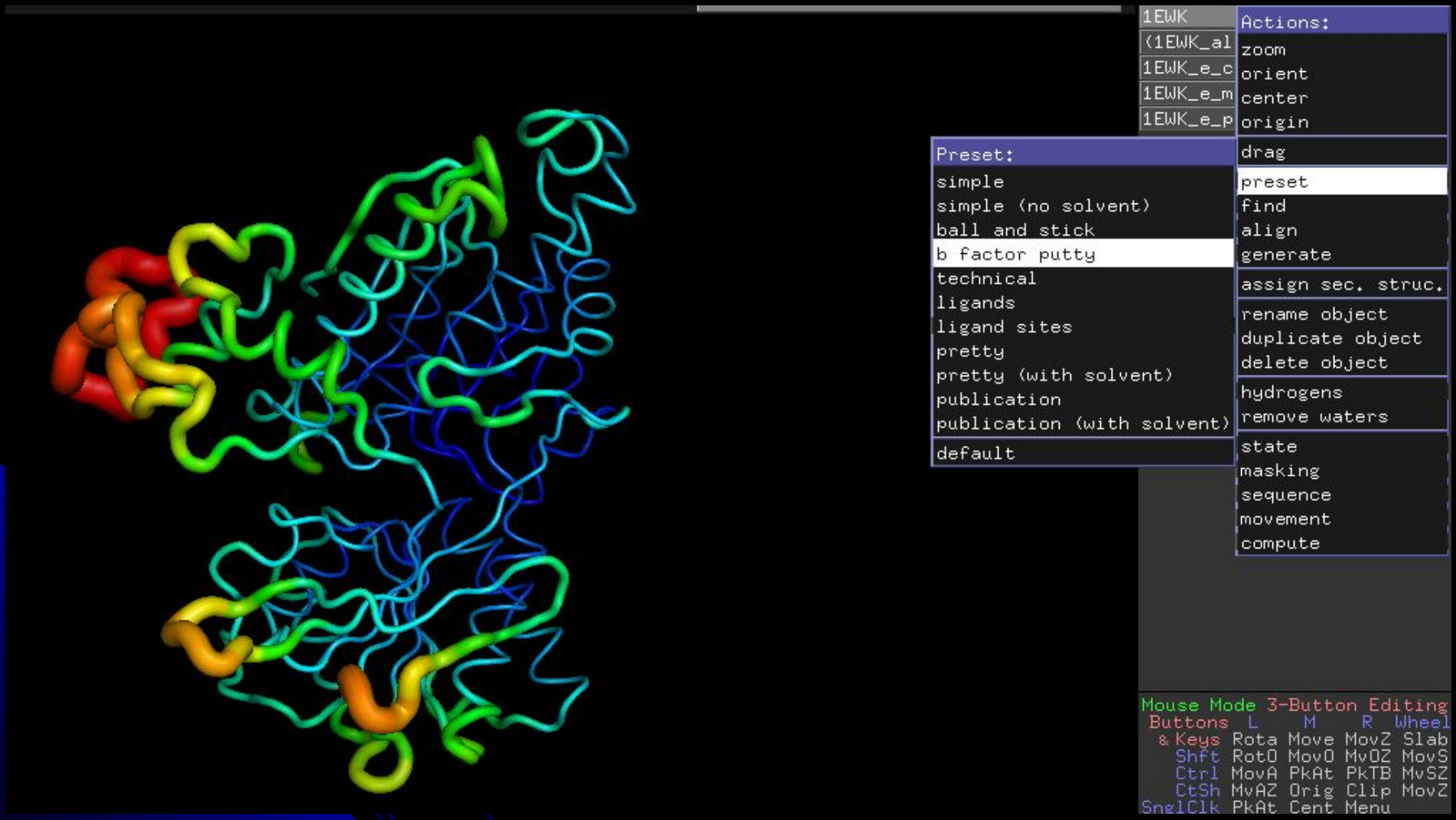
Manipulate your objects



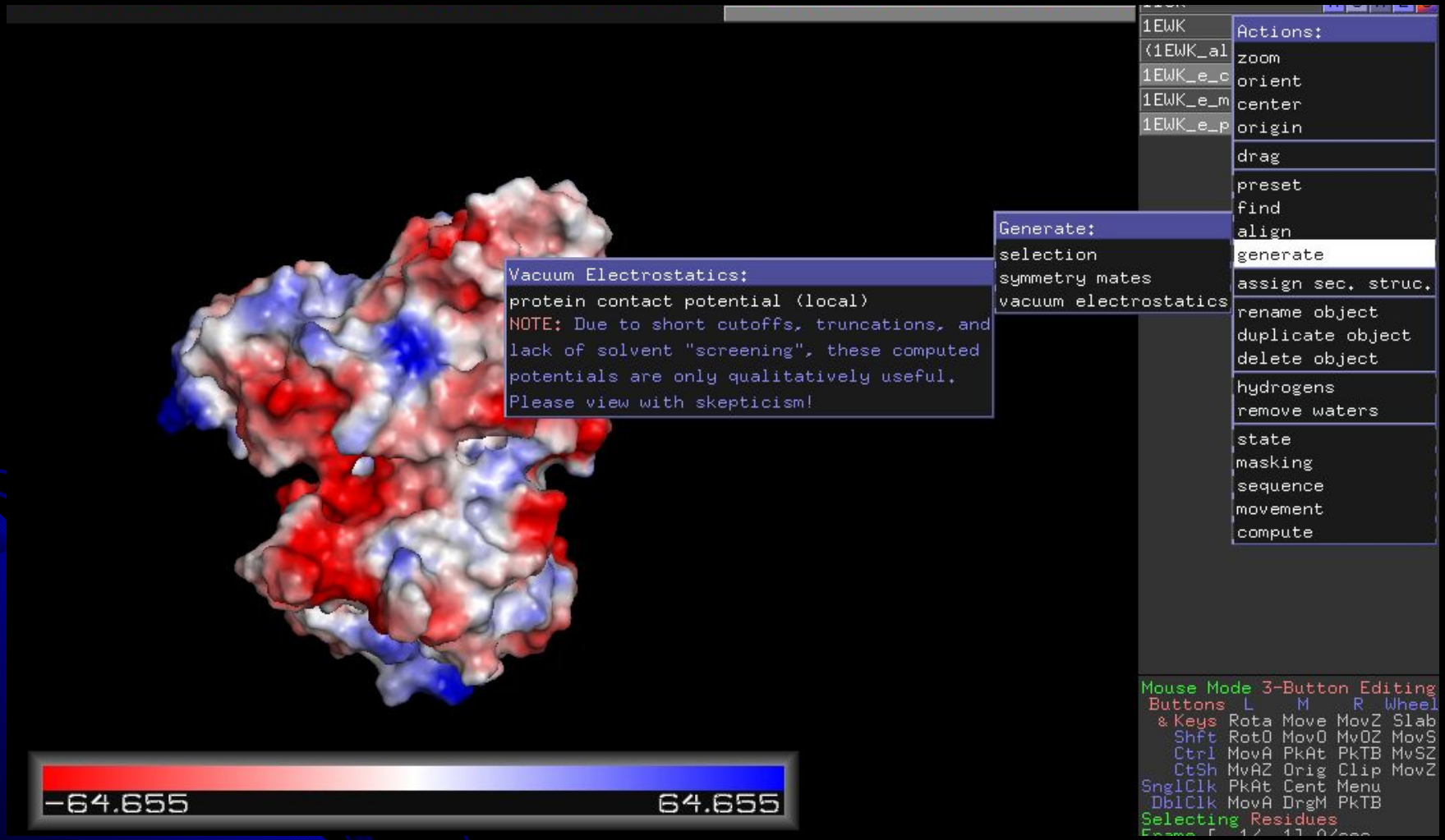
Right click your mouse: more options



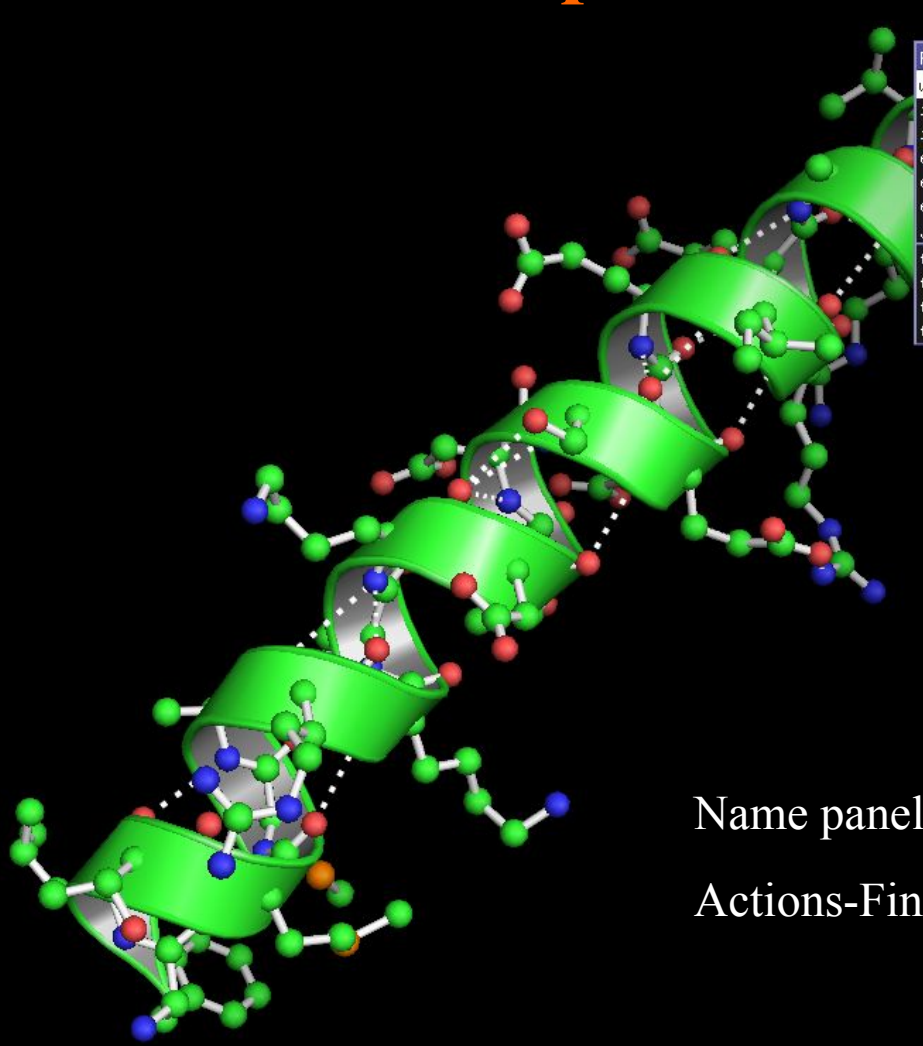
Actions: show B-factor putty



Actions: generating electrostatics map



Actions: Find polar contacts



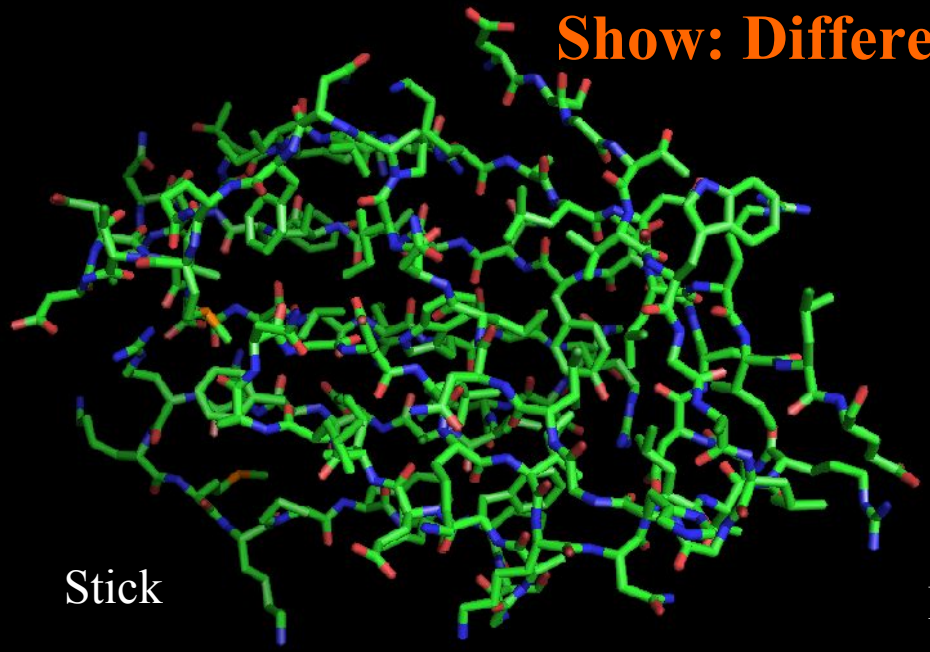
Name panel: your selected residues

Actions-Find-Polar contacts-…

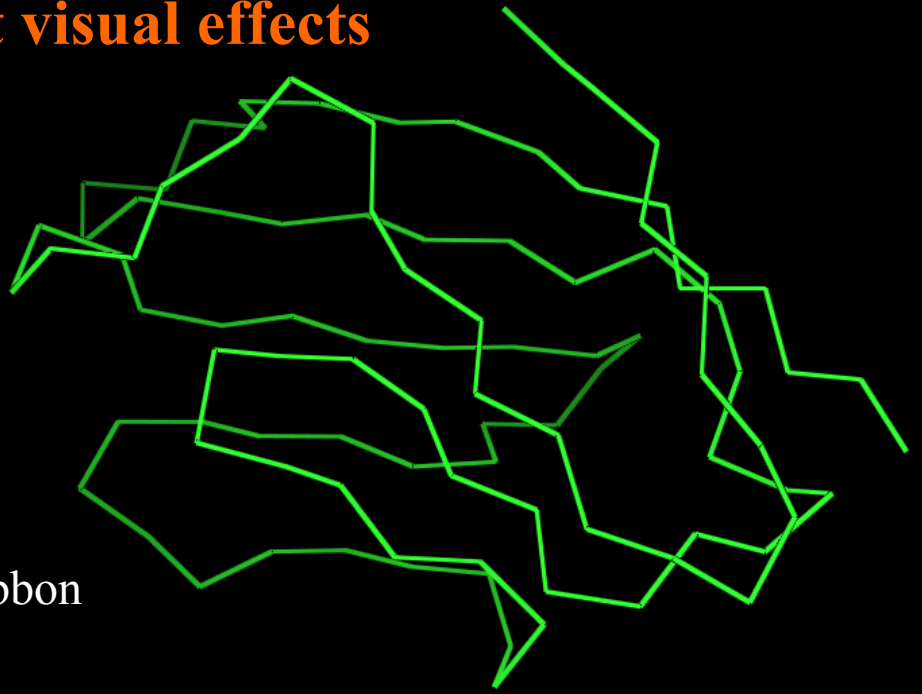
Actions: Others

- Adding or removing Hydrogen atoms
- Counting atom numbers or net charges
- Masking or unmasking residues
- Objects operations

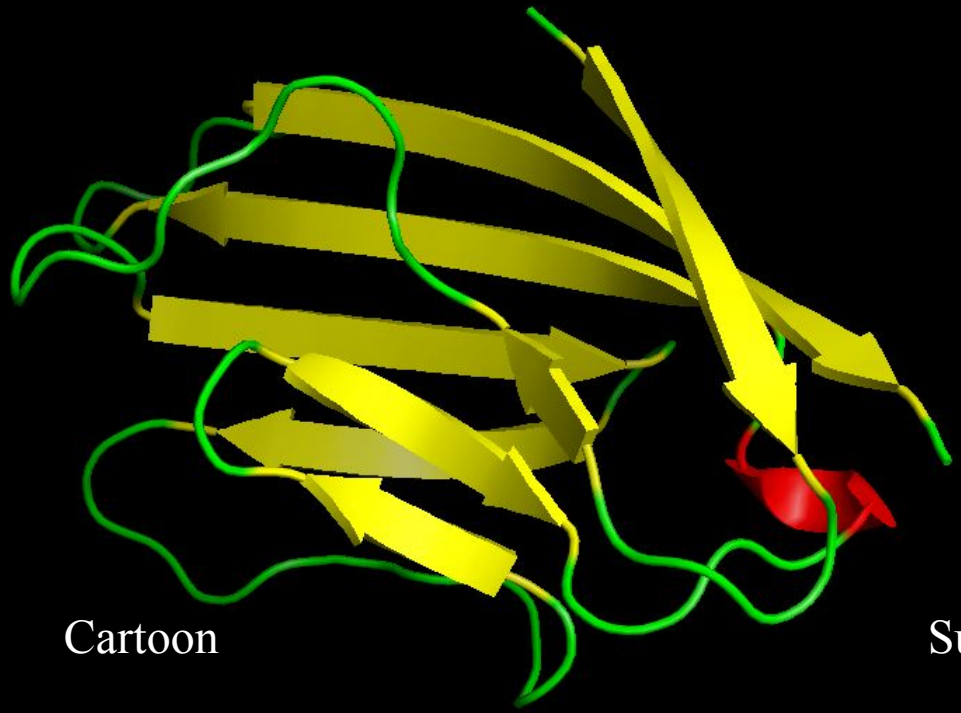
Show: Different visual effects



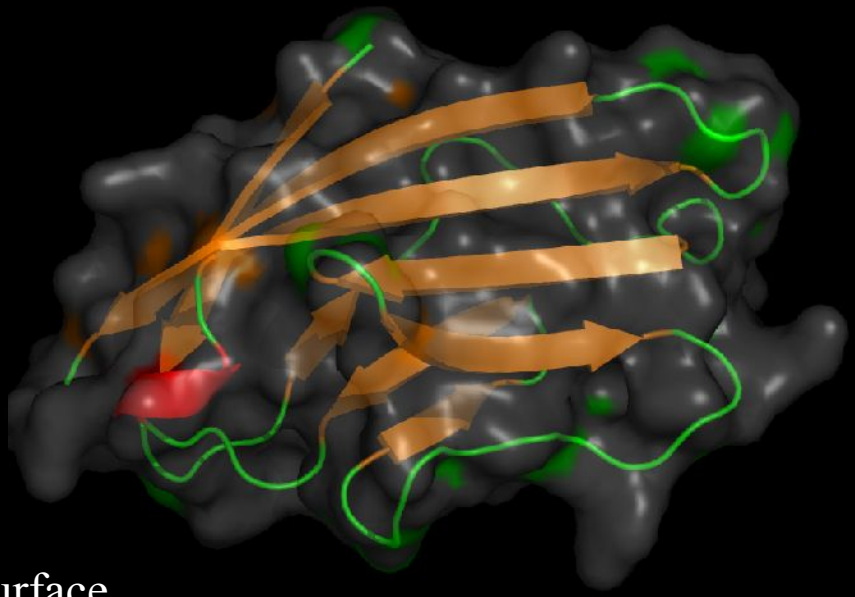
Stick



Ribbon

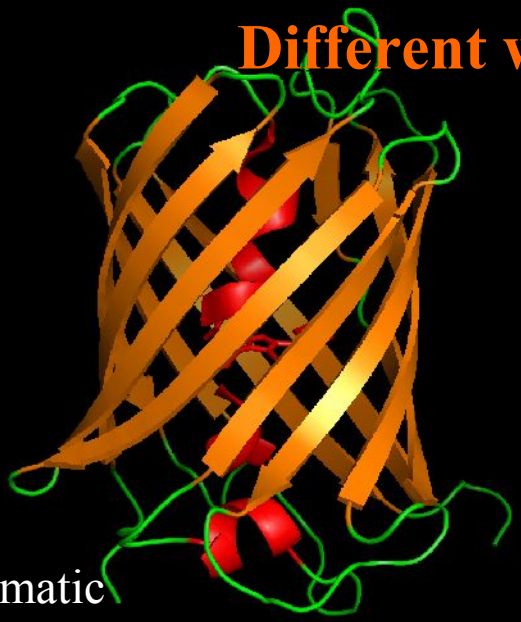


Cartoon

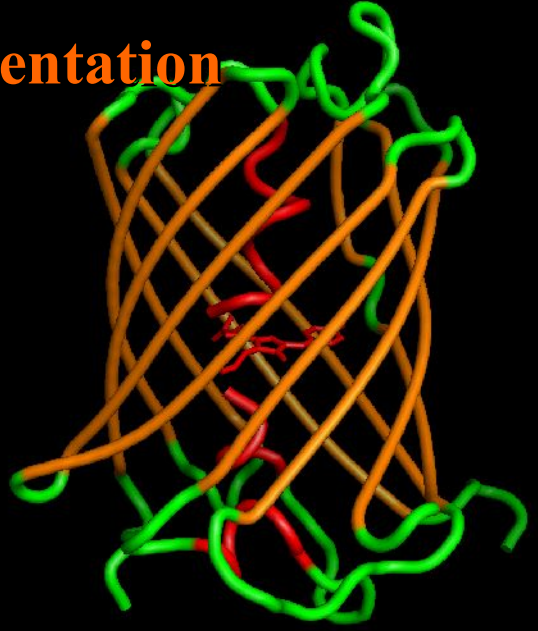


Surface

Different way of cartoon presentation

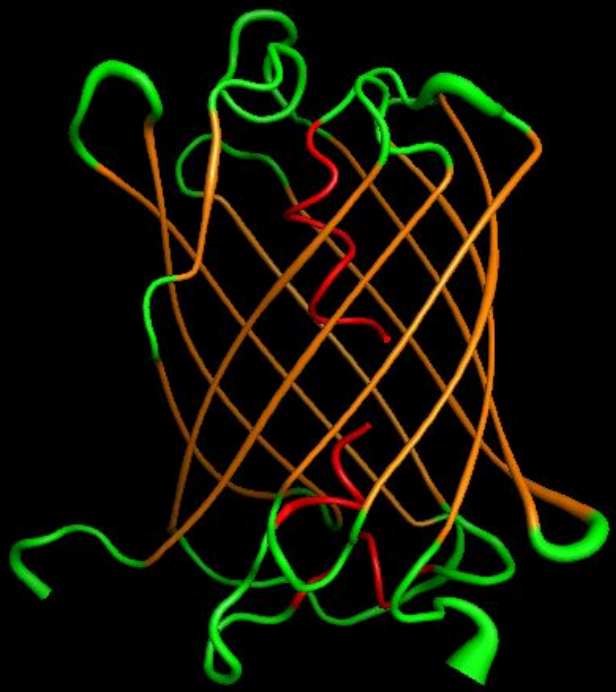


automatic



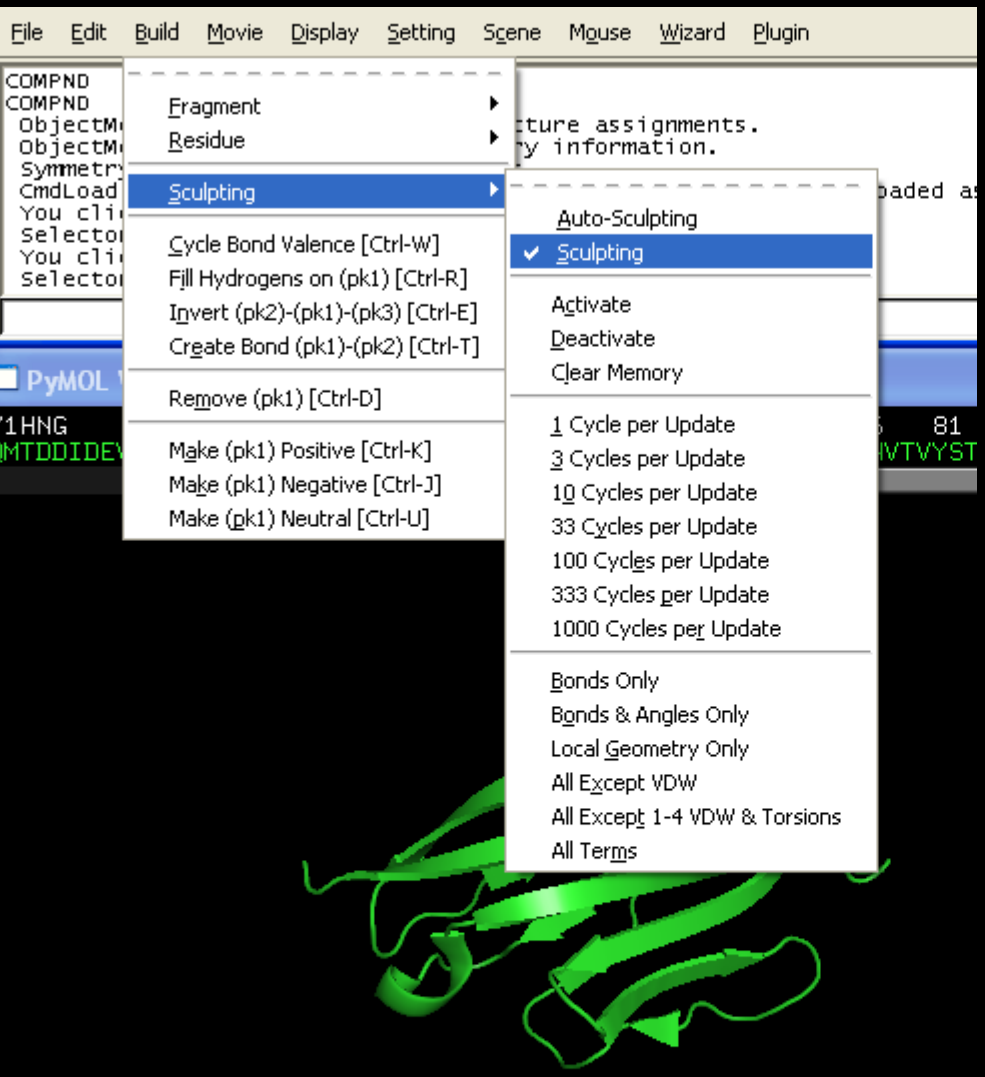
tube

loop



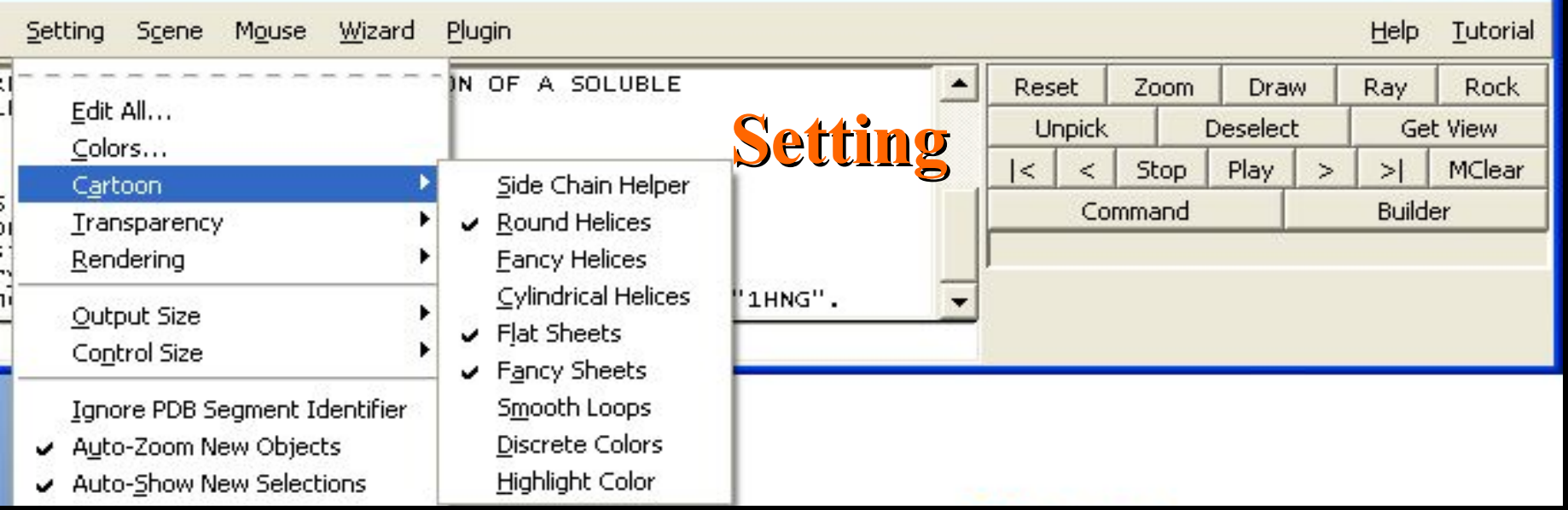
putty

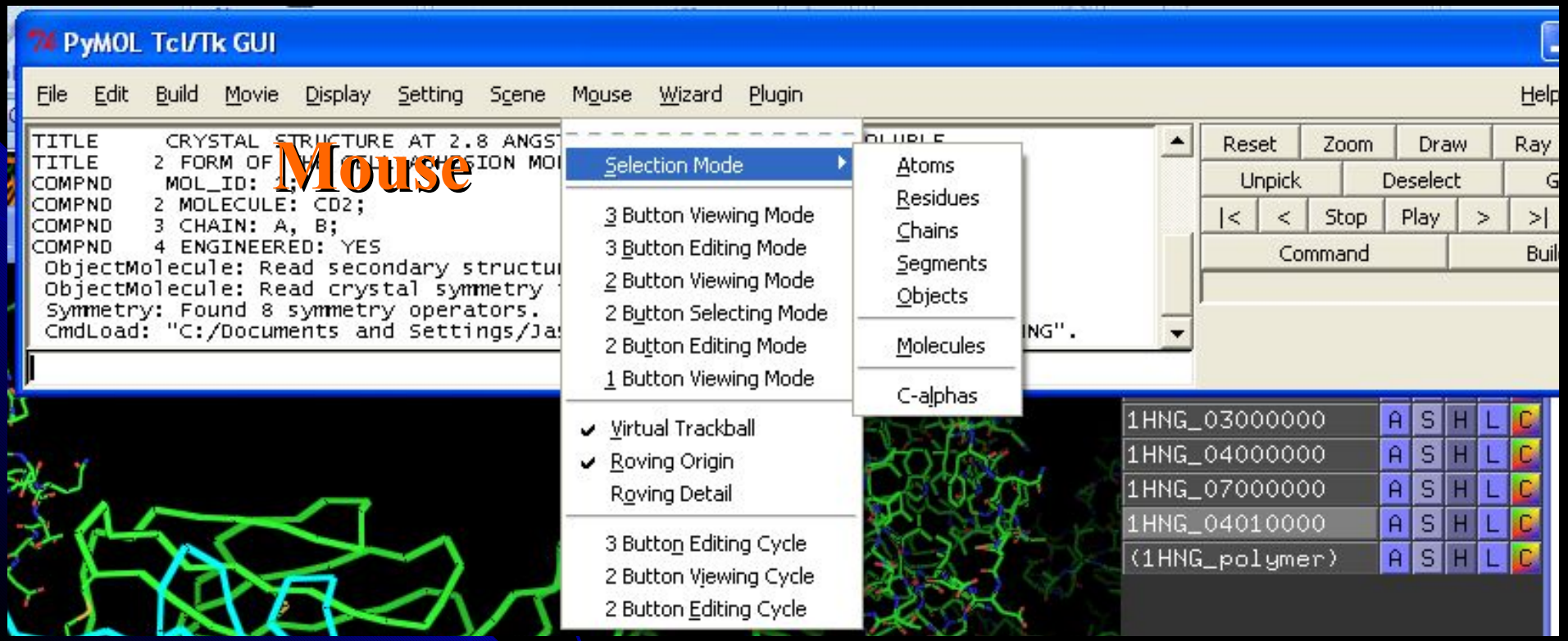
Build: sculpting

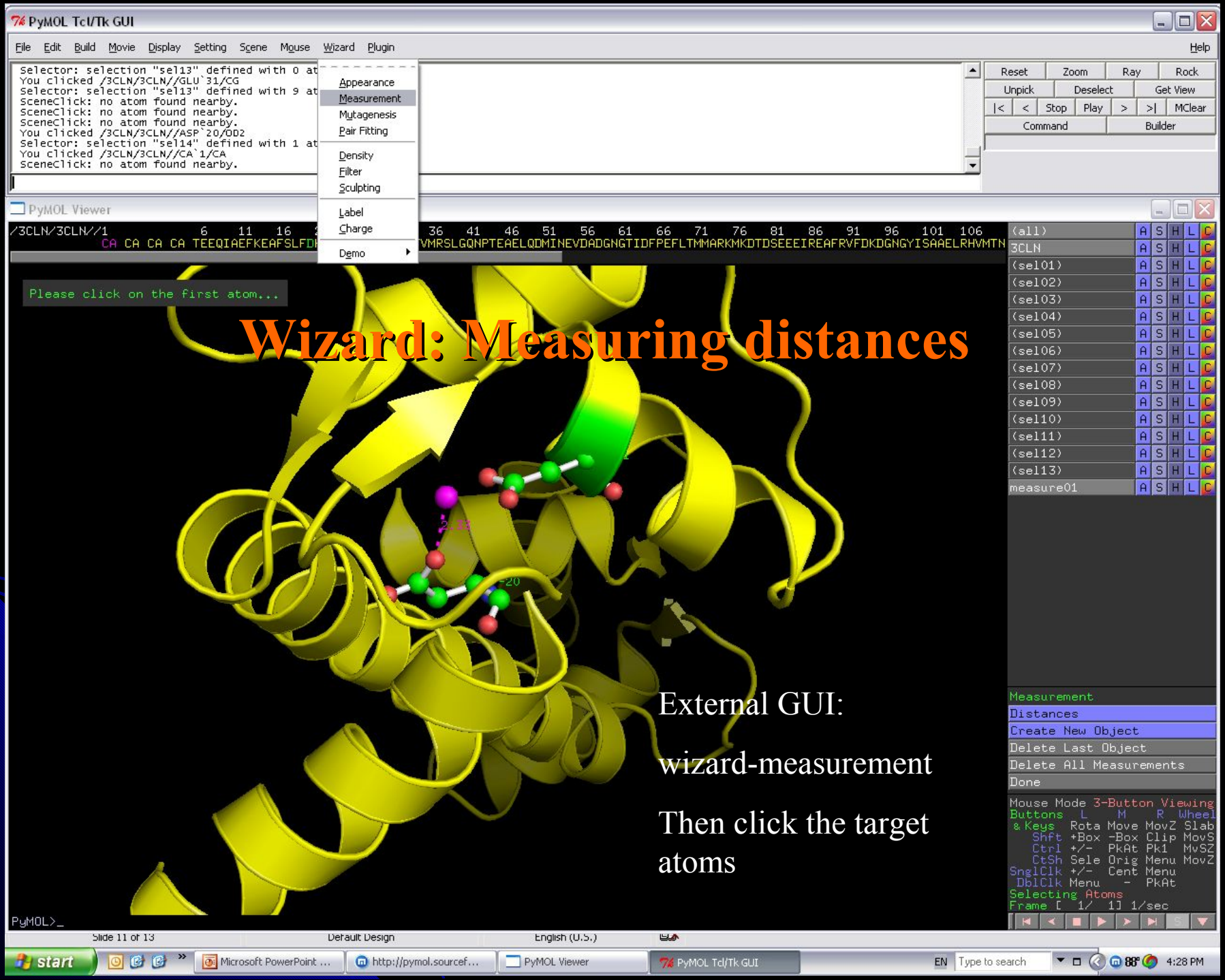


Ctrl+ left click

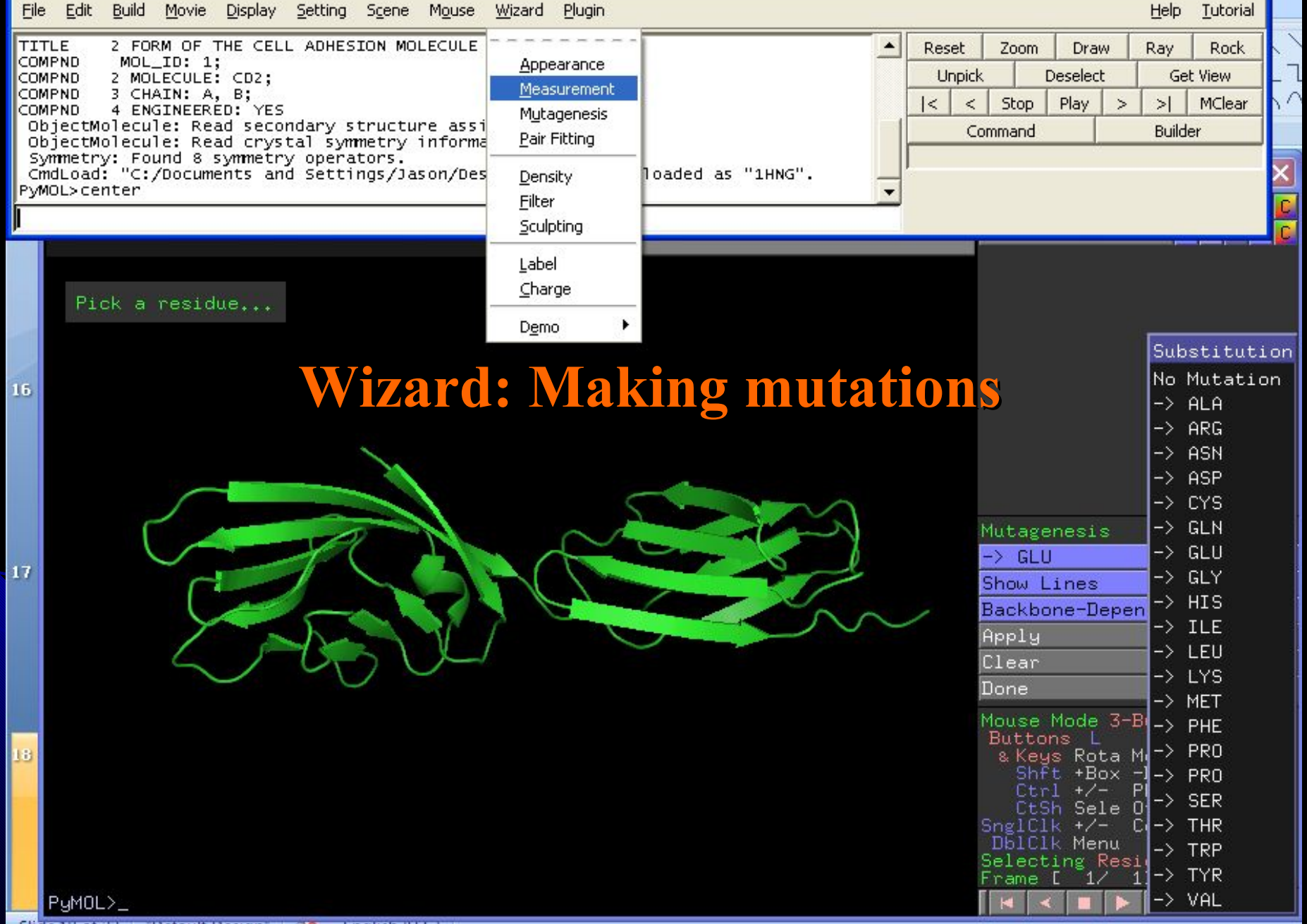
- **Also can build structures by adding or deleting atoms, bonds and amino acids**


Setting
Setting Scene Mouse Wizard Plugin Help Tutorial
Edit All...
Colors...
Cartoon
Transparency
Rendering
Output Size
Control Size
Ignore PDB Segment Identifier
Auto-Zoom New Objects
Auto-Show New Selections
Side Chain Helper
Round Helices
Fancy Helices
Cylindrical Helices
Flat Sheets
Fancy Sheets
Smooth Loops
Discrete Colors
Highlight Color
Reset Zoom Draw Ray Rock
Unpick Deselect Get View
|< < Stop Play > >| MClear
Command Builder


PyMOL TcI/Tk GUI
File Edit Build Movie Display Setting Scene Mouse Wizard Plugin Help
Mouse
TITLE CRYSTAL STRUCTURE AT 2.8 ANGST
TITLE 2 FORM OF THE CELL ADHESION MOL
COMPND MOL_ID: 1;
COMPND 2 MOLECULE: CD2;
COMPND 3 CHAIN: A, B;
COMPND 4 ENGINEERED: YES
ObjectMolecule: Read secondary structu
ObjectMolecule: Read crystal symmetry
Symmetry: Found 8 symmetry operators.
CmdLoad: "C:/Documents and Settings/Ja
Selection Mode
3 Button Viewing Mode
3 Button Editing Mode
2 Button Viewing Mode
2 Button Selecting Mode
2 Button Editing Mode
1 Button Viewing Mode
Virtual Trackball
Roving Origin
Roving Detail
3 Button Editing Cycle
2 Button Viewing Cycle
2 Button Editing Cycle
Atoms
Residues
Chains
Segments
Objects
Molecules
C-alphas
Reset Zoom Draw Ray
Unpick Deselect
|< < Stop Play > >|
Command
1HNG_03000000 A S H L C
1HNG_04000000 A S H L C
1HNG_07000000 A S H L C
1HNG_04010000 A S H L C
(1HNG_polymer) A S H L C

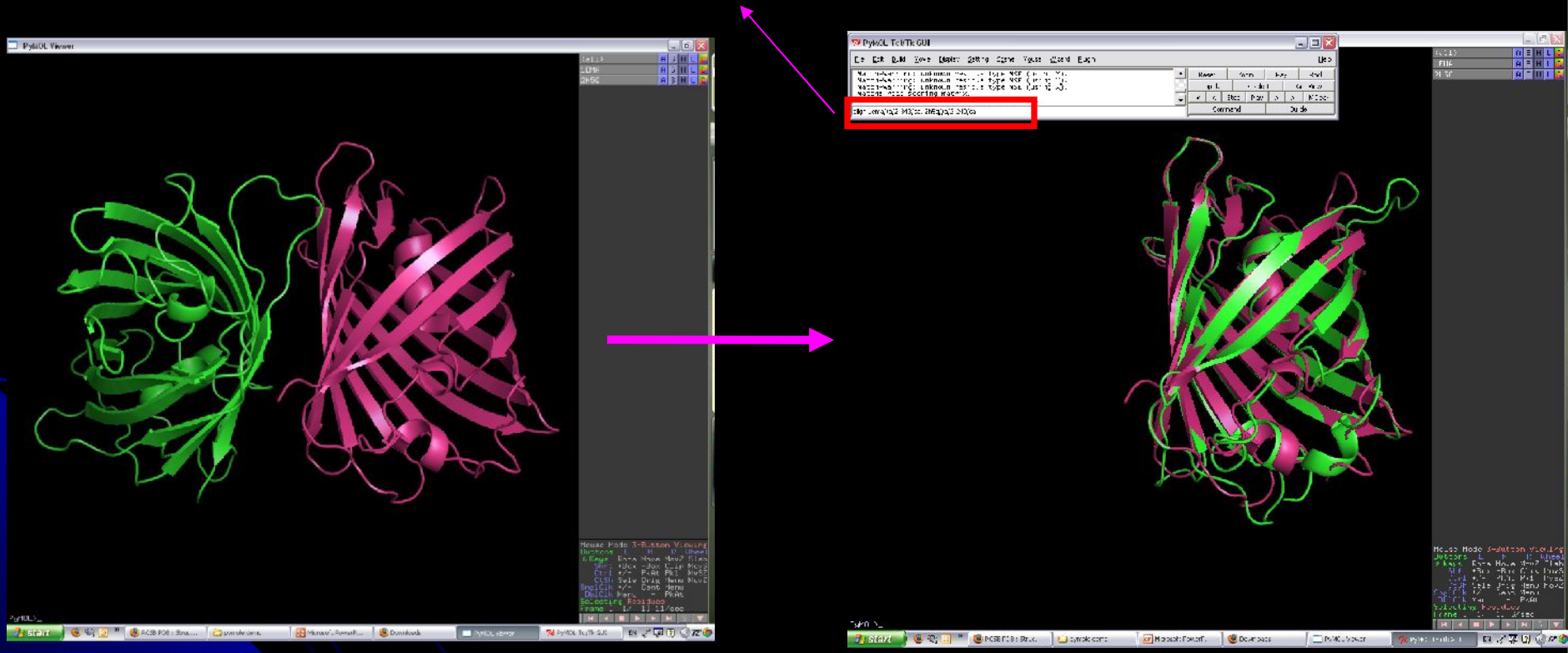

PyMOL Tcl/Tk GUI
File Edit Build Movie Display Setting Scene Mouse Wizard Plugin Help
Appearance
Measurement
Mutagenesis
Pair Fitting
Density
Filter
Sculpting
Label
Charge
Demo
Reset Zoom Ray Rock
Unpick Deselect Get View
Stop Play MClear
Command Builder
PyMOL Viewer
Please click on the first atom...
Wizard: Measuring distances
External GUI:
wizard-measurement
Then click the target atoms
Measurement
Distances
Create New Object
Delete Last Object
Delete All Measurements
Done
PyMOL>_

Wizard: Making mutations



Overlay two structures

Align filenameA//2-240/CA, filenameB//2-240/CA



Make movies

Simplest way: Fetch filename, mplay

```
Util.mrock(start,finish,angle,phase,loop-flag)
Util.mroll(start,finish,loop-flag)
```

e.g.,

```
load $c/3cln.pdb
mset 1 x60
util.mrock 1,60,180
```

Create a 60 frame movie with +/- 90 deg. rock

```
load $c/3cln.pdb
mset 1 x60
util.mrock 1,60
```

Create full rotation around the Y axis over 60 frames

ImageJ, or Xnview and UnFREEz to generate movies

Scripts animation in Pymol