

# 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](http://en.wikipedia.org/wiki/Scene_description_language)

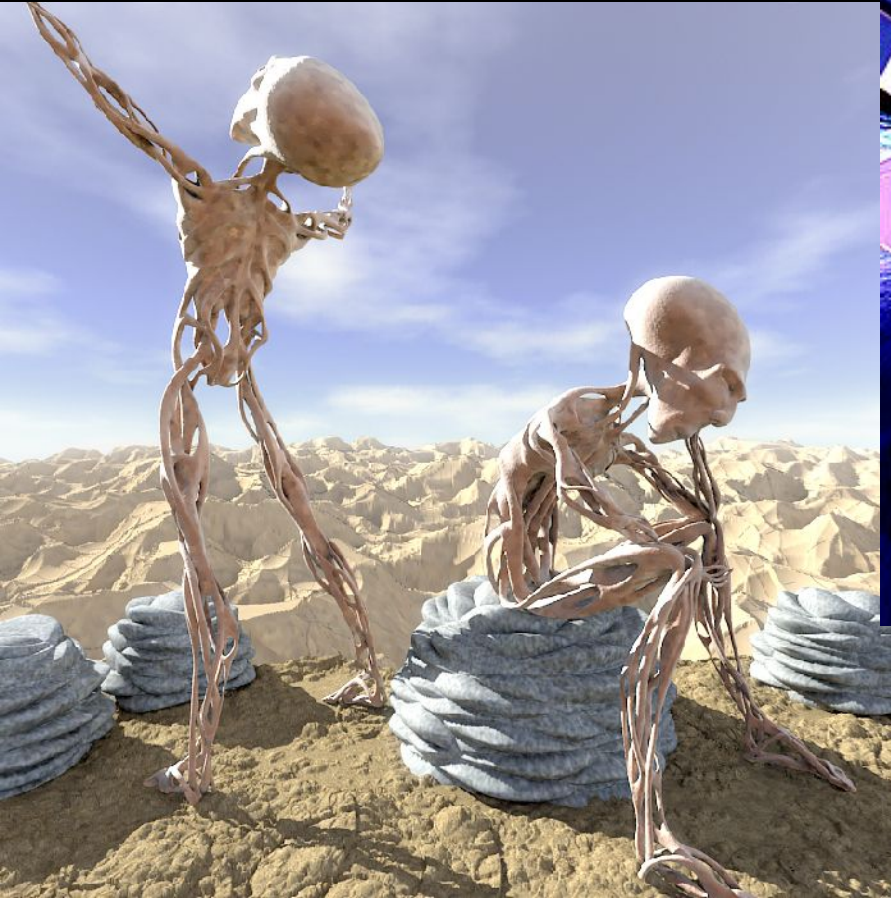
# RenderMan Shading Language

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
```

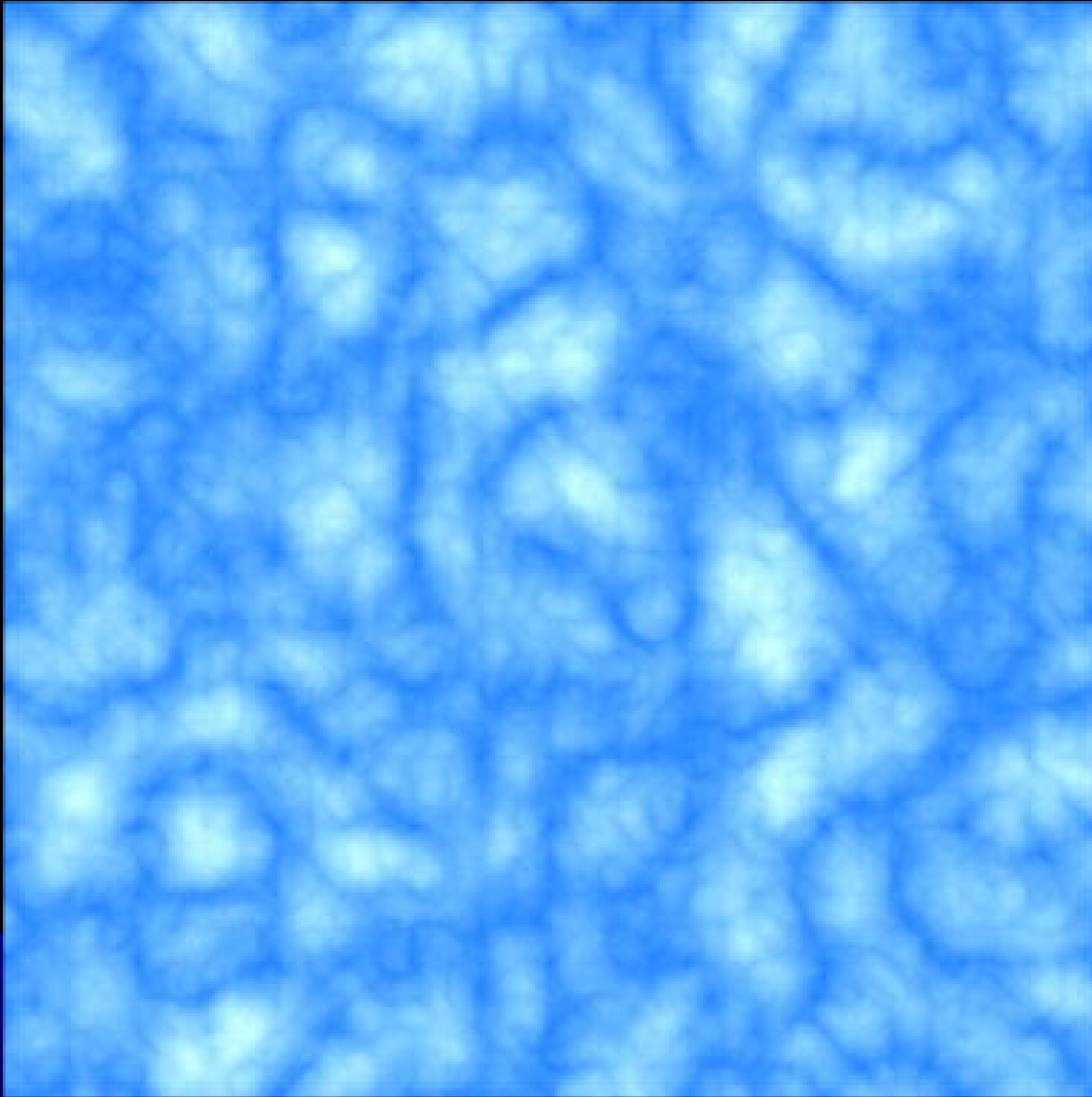
```
WorldBegin
```

```
Surface "clouds"
```

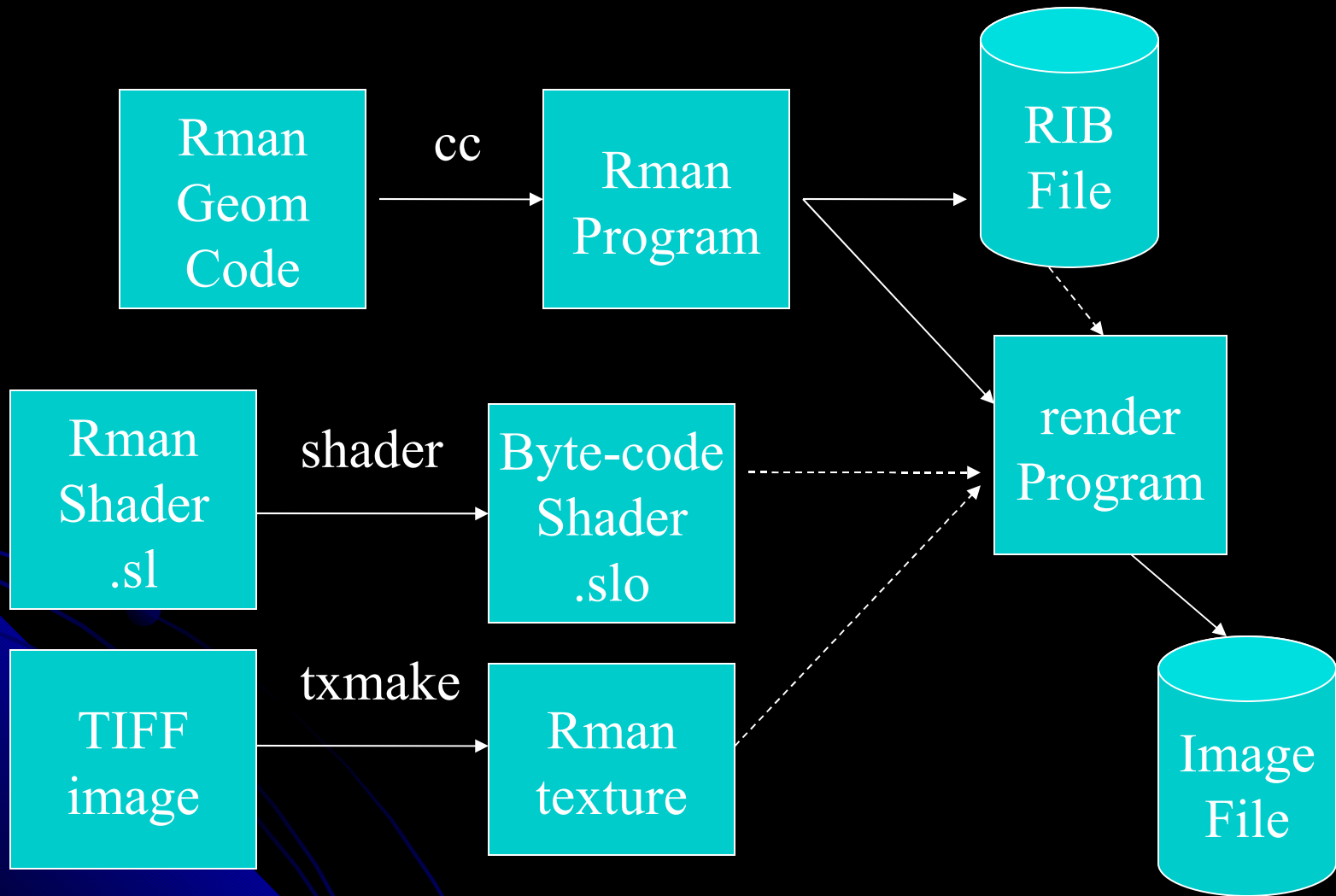
```
Polygon "P" [0.5 0.5 0.5 0.5 -0.5  
0.5 -0.5 -0.5 0.5 -0.5 0.5 0.5]
```

```
WorldEnd
```

# Result



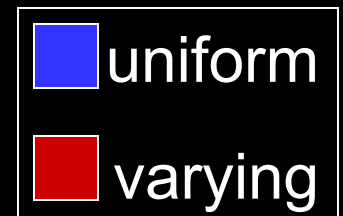
# Součásti systému RenderMan





# 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 transparentnost, refrakci světla a podobné efekty.

# Rendering Pipeline - OpenGL

Aaron Bloomfield

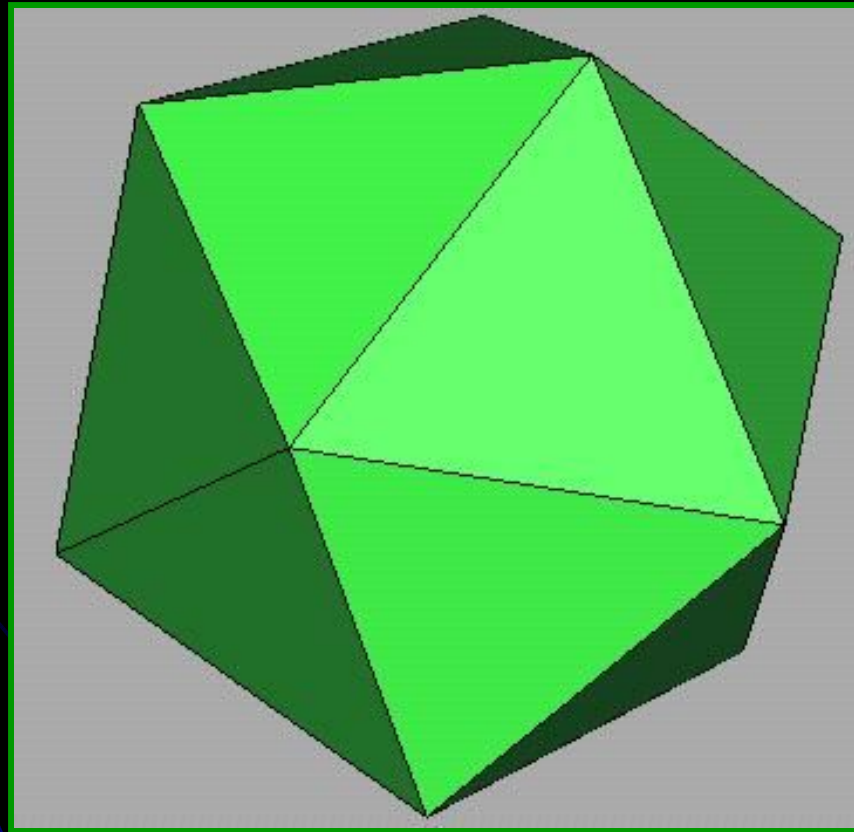
CS 445: Introduction to Graphics

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

## 3D Geometric Primitives

Modeling  
Transformation

Lighting

Viewing  
Transformation

Projection  
Transformation

Clipping

Scan  
Conversion

Image

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

Modeling Transformation

Transform into 3D world coordinate system

Viewing Transformation

Lighting & Texturing

Projection Transformation

Clipping

Scan Conversion

Image

# 3D Rendering Pipeline

## 3D Geometric Primitives

Modeling  
Transformation

Viewing  
Transformation

Lighting &  
Texturing

Projection  
Transformation

Clipping

Scan  
Conversion

Image

Transform into 3D camera coordinate system  
Done with modeling transformation

# 3D Rendering Pipeline

## 3D Geometric Primitives

Modeling  
Transformation

Viewing  
Transformation

Lighting &  
Texturing

Projection  
Transformation

Clipping

Scan  
Conversion

Image

Illuminate according to lighting and reflectance  
Apply texture maps

# 3D Rendering Pipeline

## 3D Geometric Primitives

Modeling  
Transformation

Viewing  
Transformation

Lighting &  
Texturing

Projection  
Transformation

Clipping

Scan  
Conversion

Image

Transform into 2D screen coordinate system

# 3D Rendering Pipeline

## 3D Geometric Primitives

Modeling  
Transformation

Viewing  
Transformation

Lighting &  
Texturing

Projection  
Transformation

Clipping

Scan  
Conversion

Image

Clip primitives outside camera's view

# 3D Rendering Pipeline

## 3D Geometric Primitives

Modeling  
Transformation

Viewing  
Transformation

Lighting &  
Texturing

Projection  
Transformation

Clipping

Scan  
Conversion

Image

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

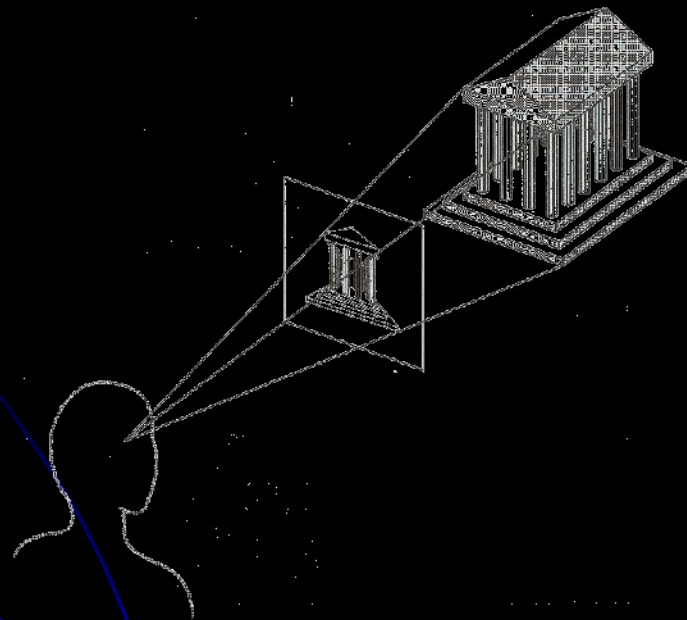
General definition:

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

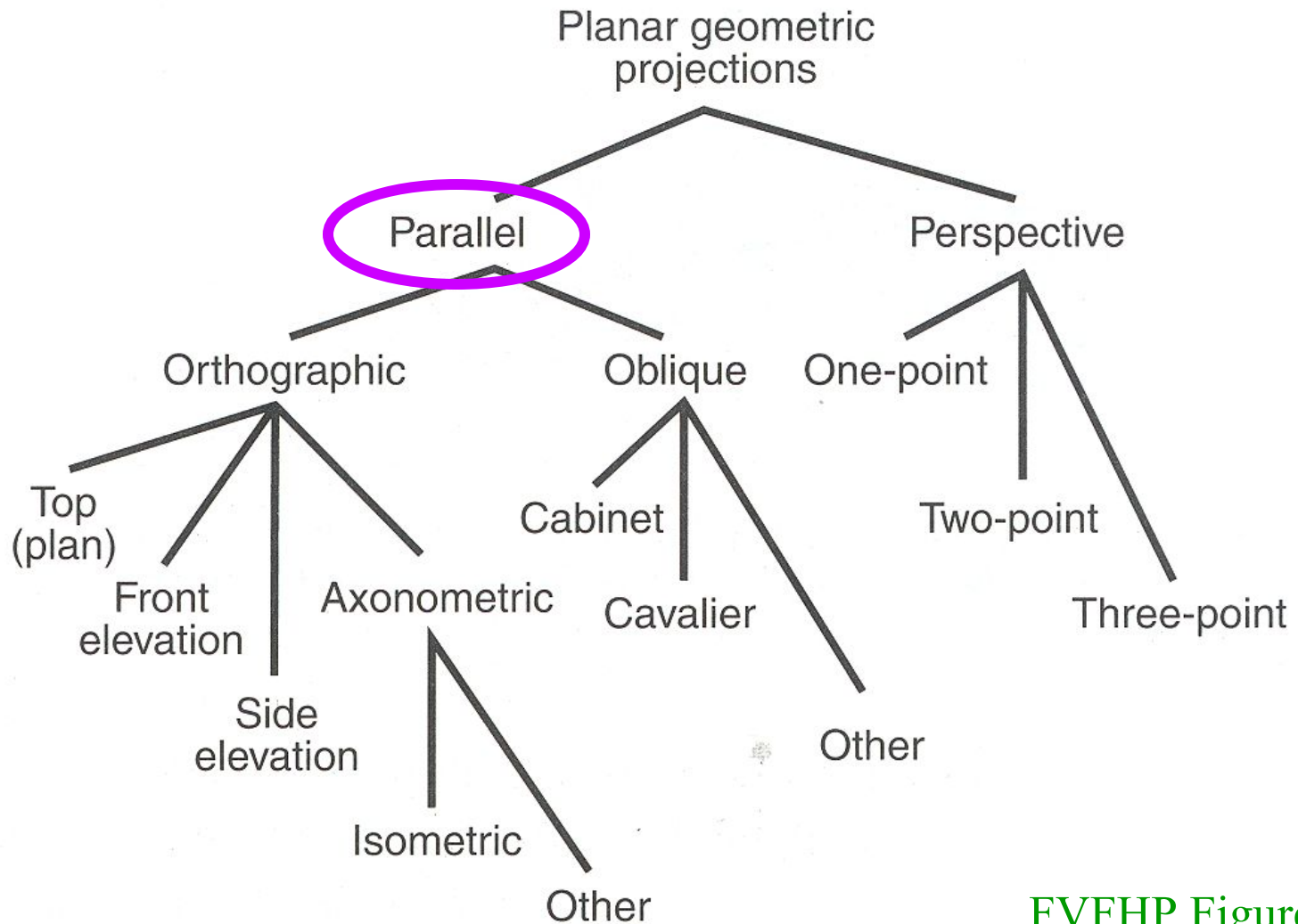
In computer graphics:

Map 3D camera coordinates to 2D screen coordinates

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



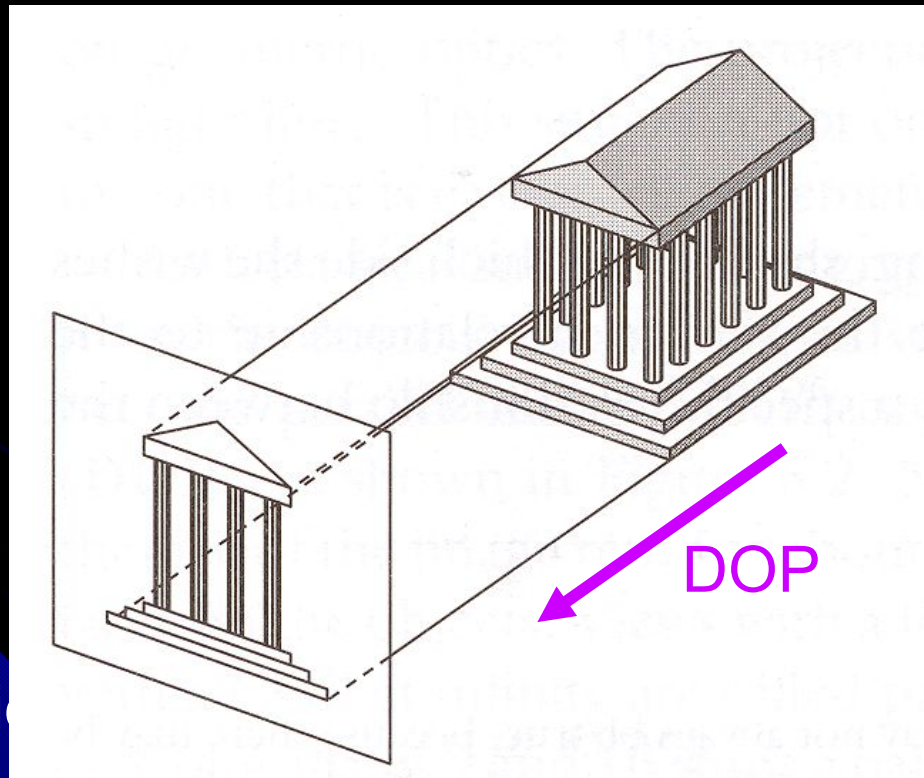
# Taxonomy of Projections



# Parallel Projection

Center of projection is at infinity

Direction of projection (DOP) same for all points

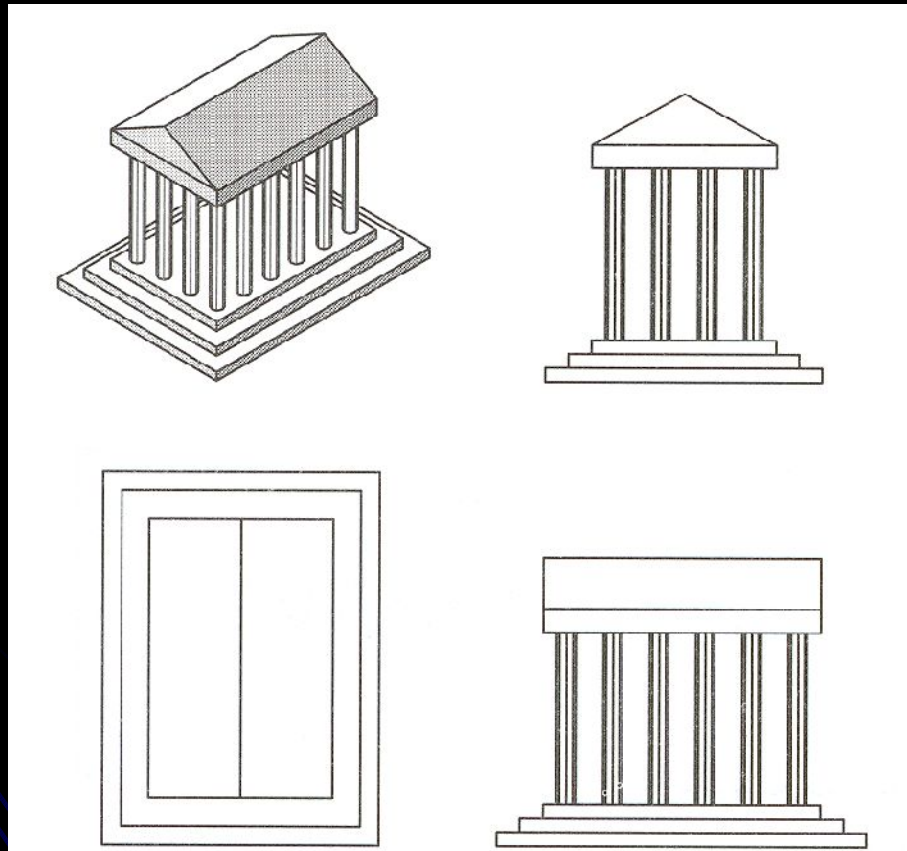


Vi  
Plane

Angel Figure 5.4

# Orthographic Projections

DOP perpendicular to view plane

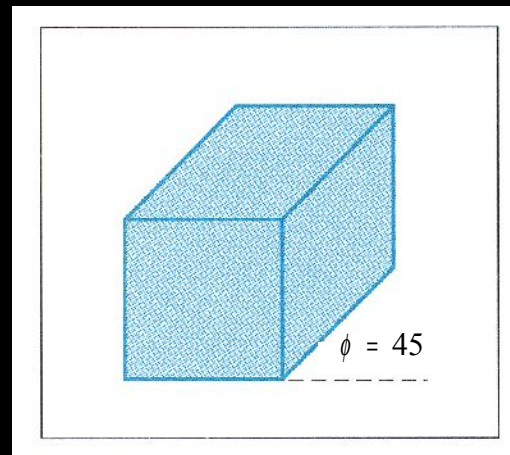
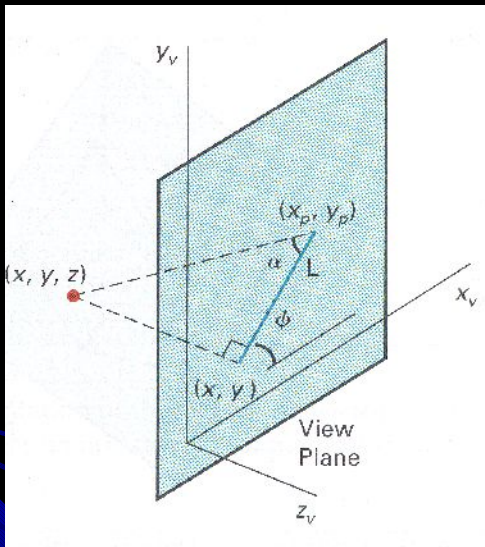


Top

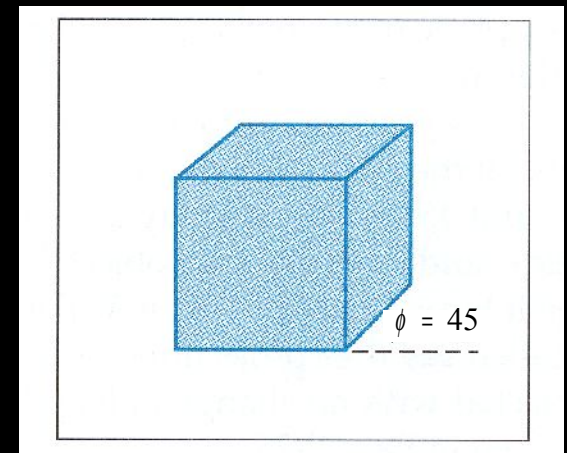
Side

# Oblique Projections

DOP **not** perpendicular to view plane

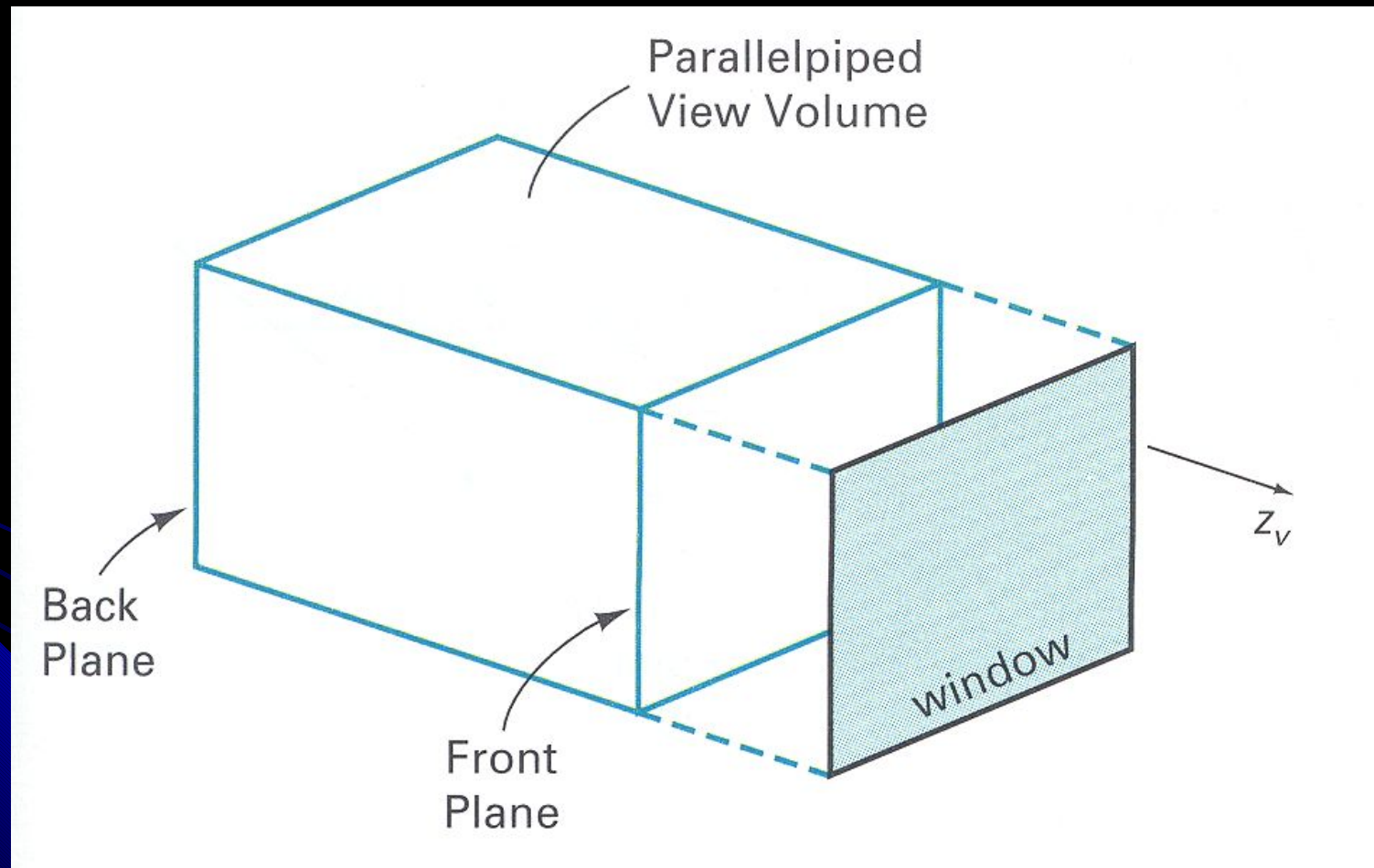


Cavalier  
(DOP  $\alpha = 45^\circ$ )

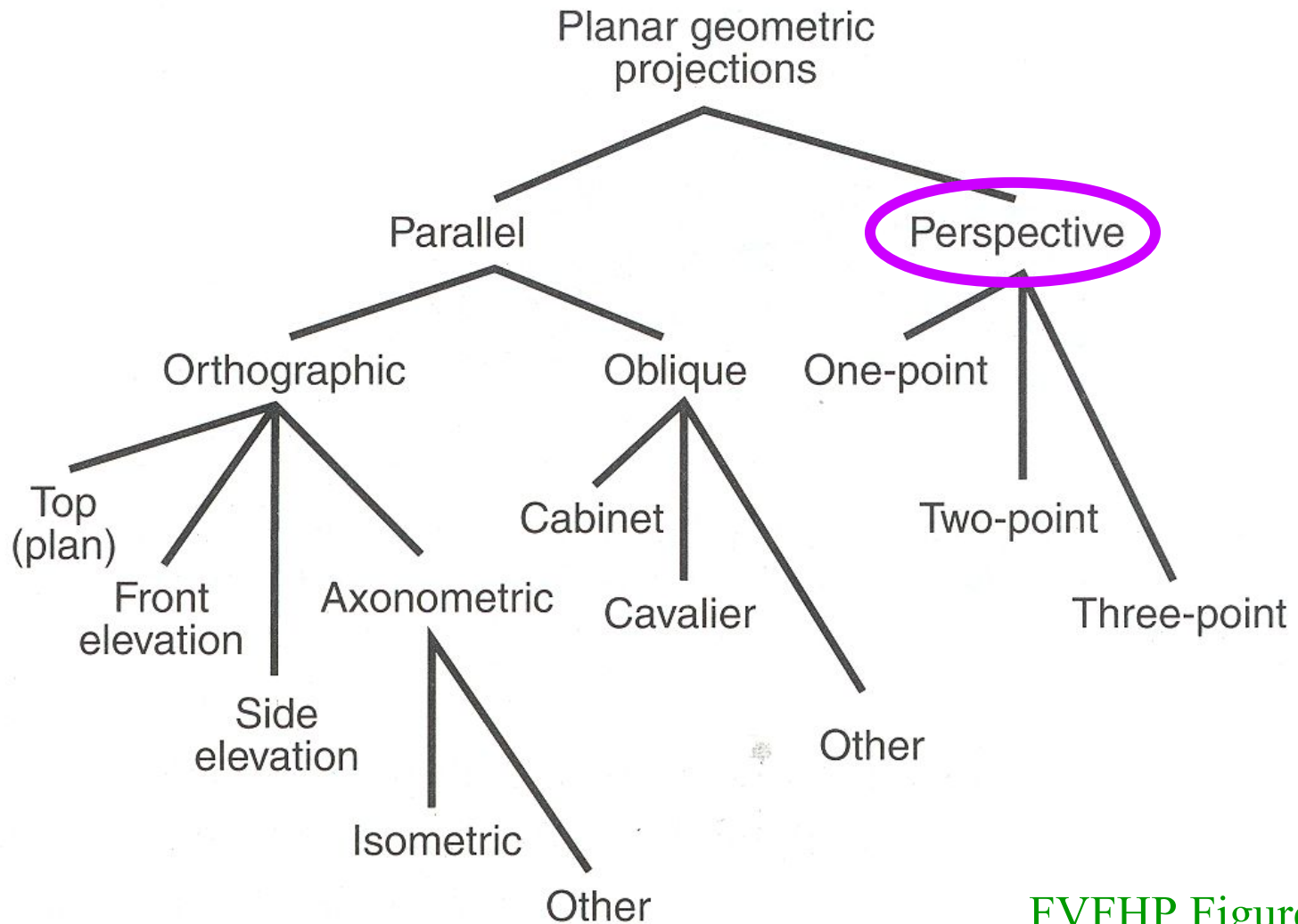


Cabinet  
(DOP  $\alpha = 63.4^\circ$ )

# Parallel Projection View Volume



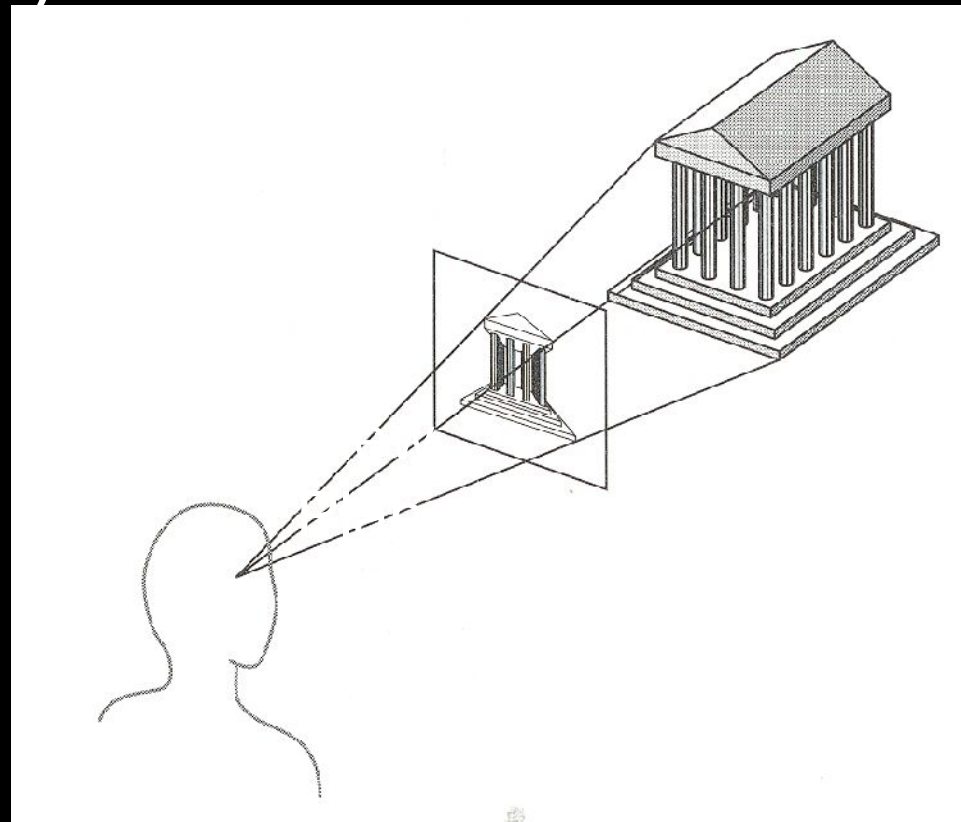
# Taxonomy of Projections



# Perspective Projection

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

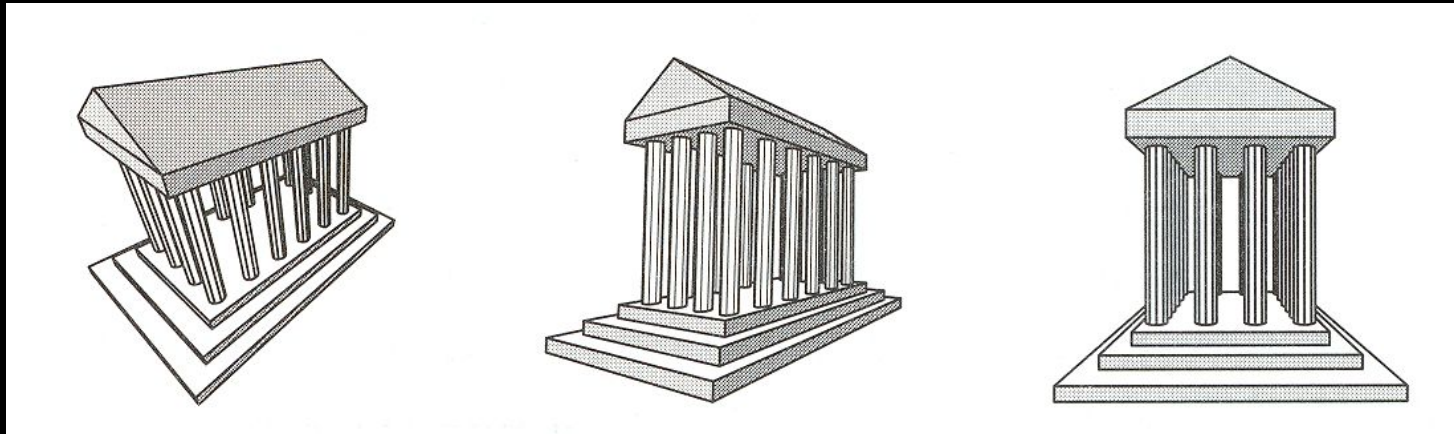
Center of  
Projection





# Perspective Projection

How many vanishing points?



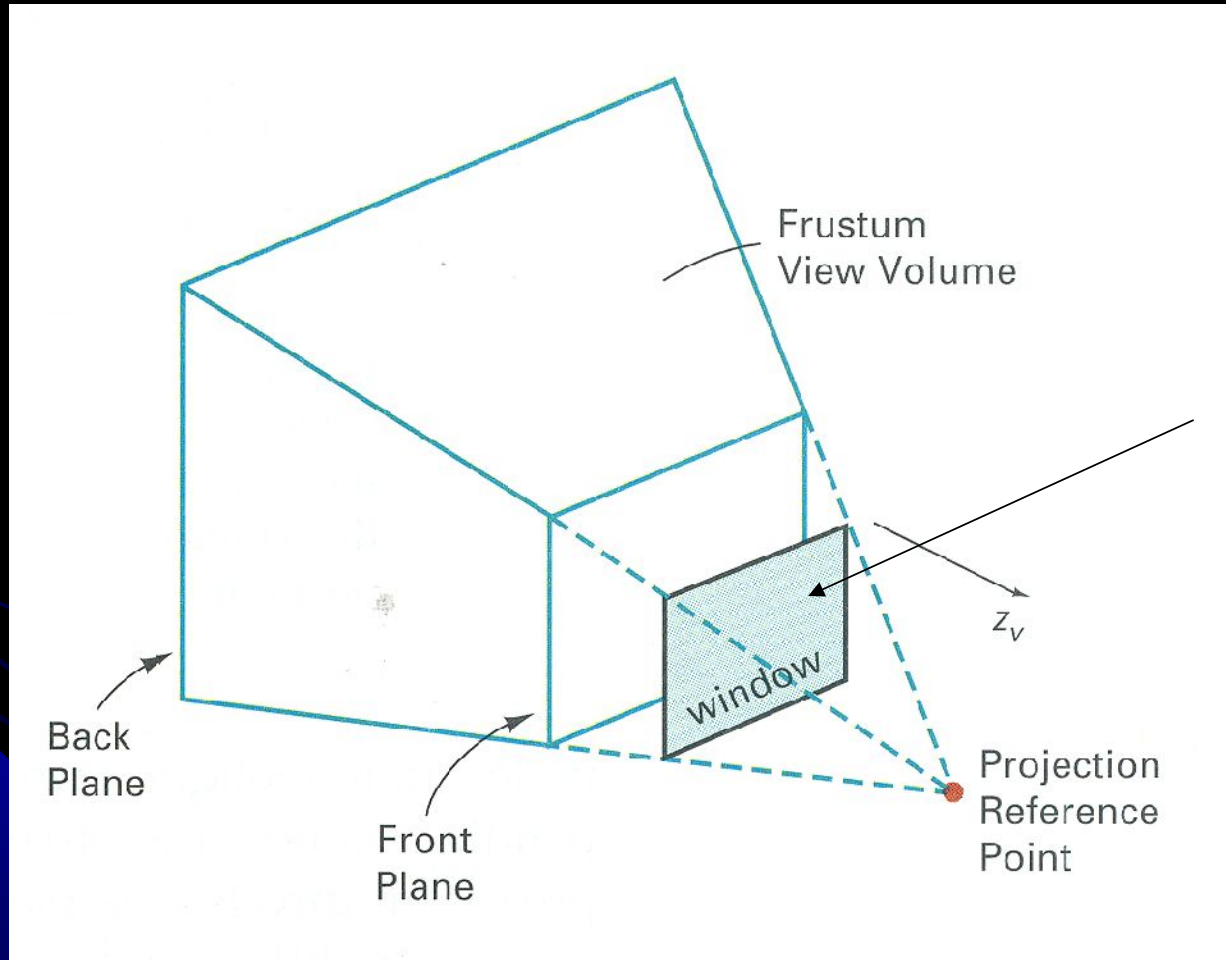
3-Point  
Perspective

2-Point  
Perspective

1-Point  
Perspective

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

# Perspective Projection View Volume



View  
Plane

# Camera to Screen

Remember: Object  $\rightarrow$  Camera  $\rightarrow$  Screen

Just like raytracer

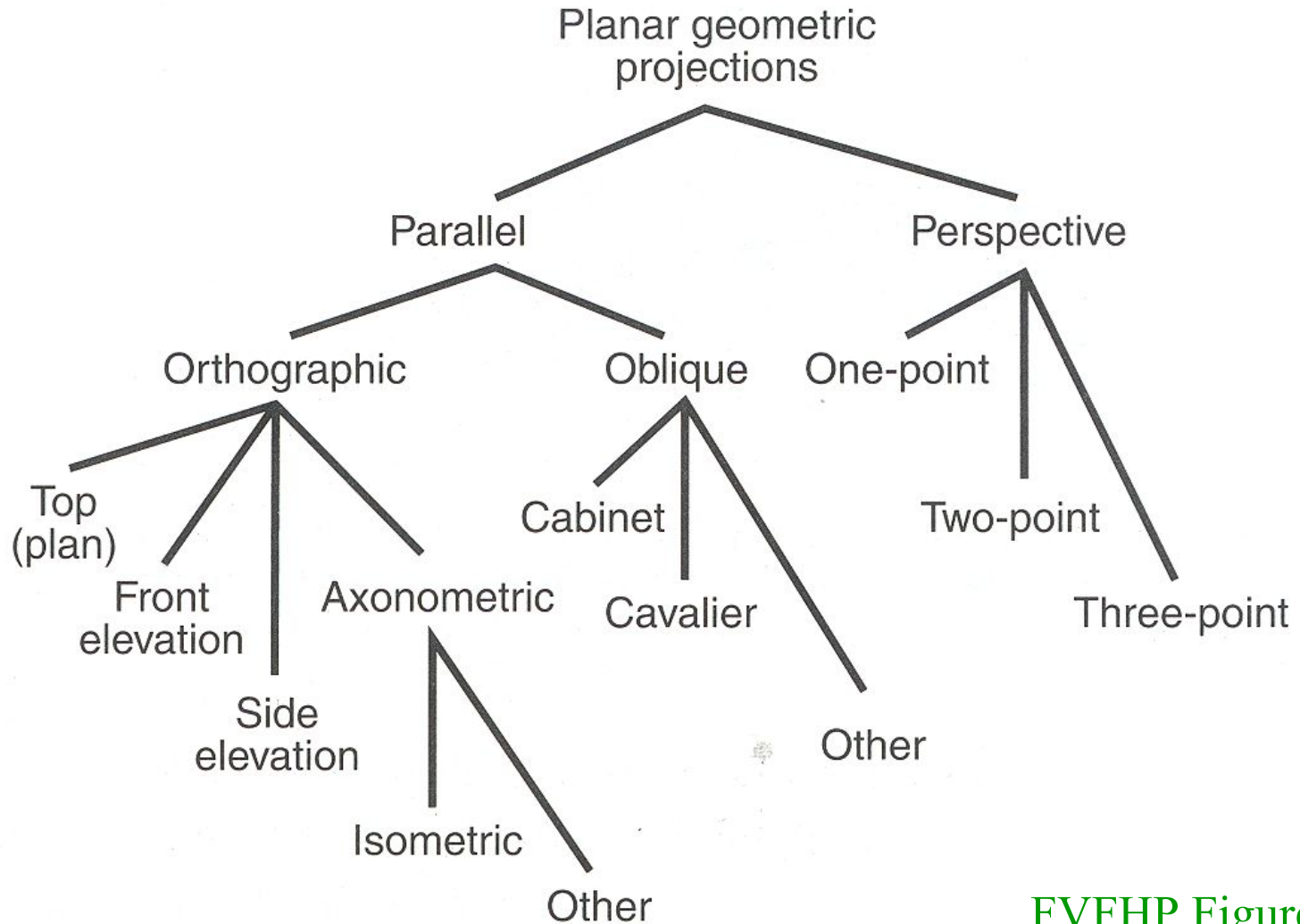
“screen” is the  $z=d$  plane for some constant  $d$

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

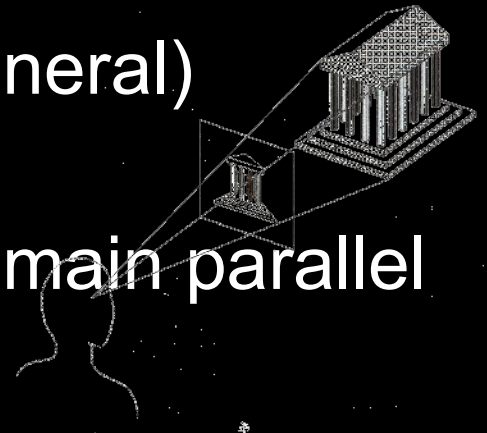
# Taxonomy of Projections



# Perspective vs. Parallel

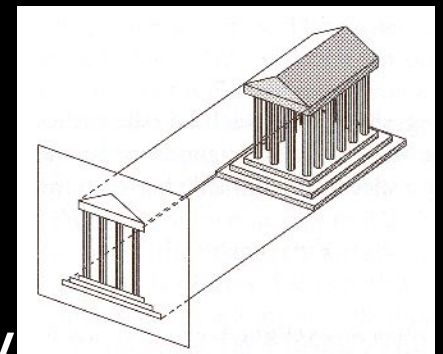
## Perspective projection

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

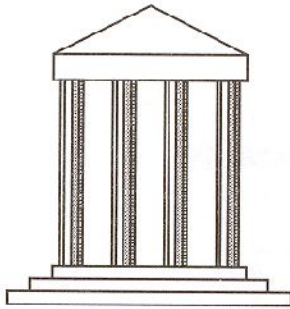


## ■ Parallel projection

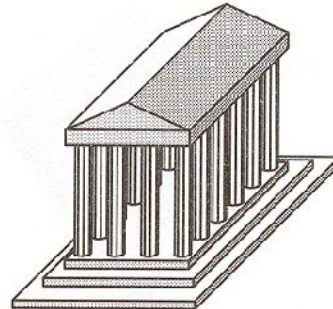
- + Good for exact measurements
- + Parallel lines remain parallel
- Angles are not (in general) preserved



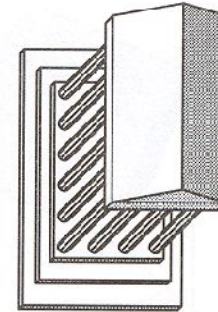
# Classical Projections



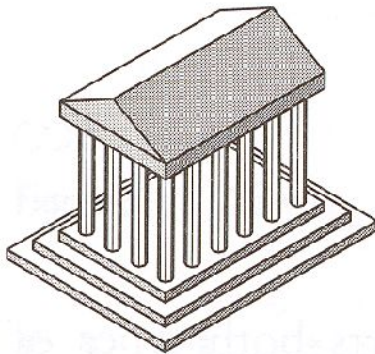
Front elevation



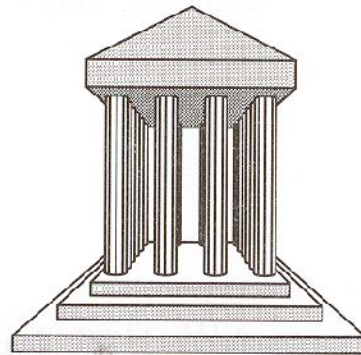
Elevation oblique



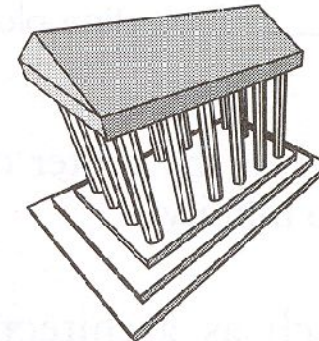
Plan oblique



Isometric



One-point perspective



Three-point perspective



CSC 480 / 580

# Computer Graphics

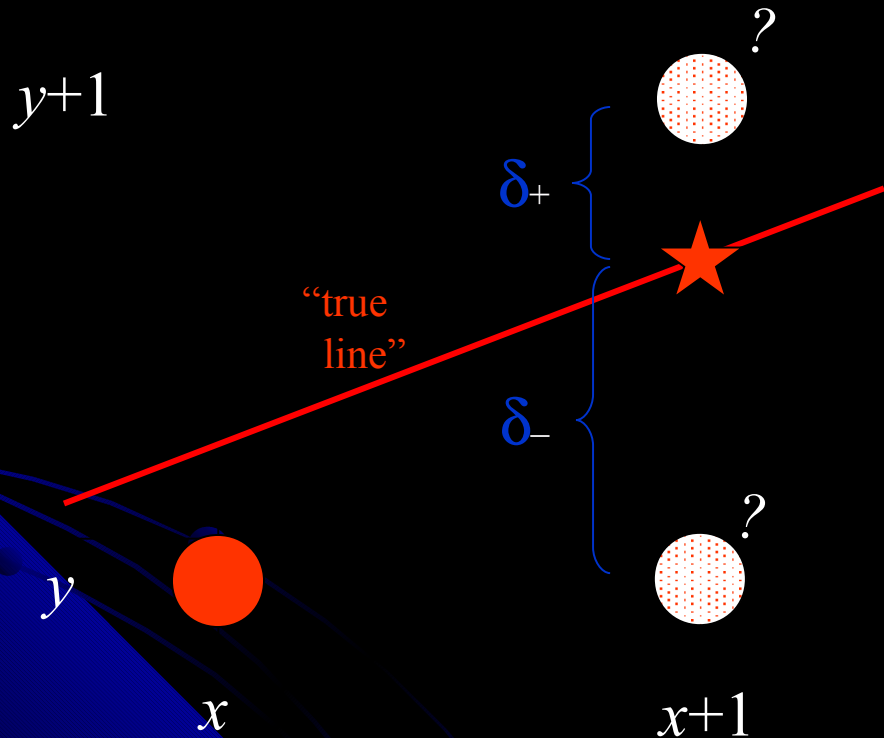
K. Kirby

Scan Conversion

Filling

# Scan-converting Lines - Toward the Bresenham Algorithm

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



imagine pixel centers  
at grid vertices

*At each step:*

```
x++;  
if (  $\delta_- > \delta_+$  )  
    y++;
```

Let  $p = \Delta x (\delta_- - \delta_+)$ . Then:

*At each step:*

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

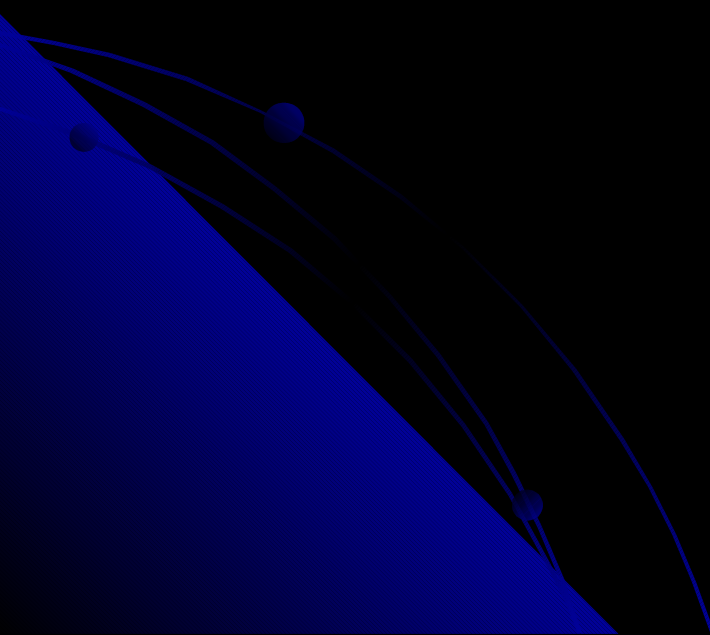
*Can we do this with simple all-int arithmetic? Yes!*



# The Bresenham Algorithm

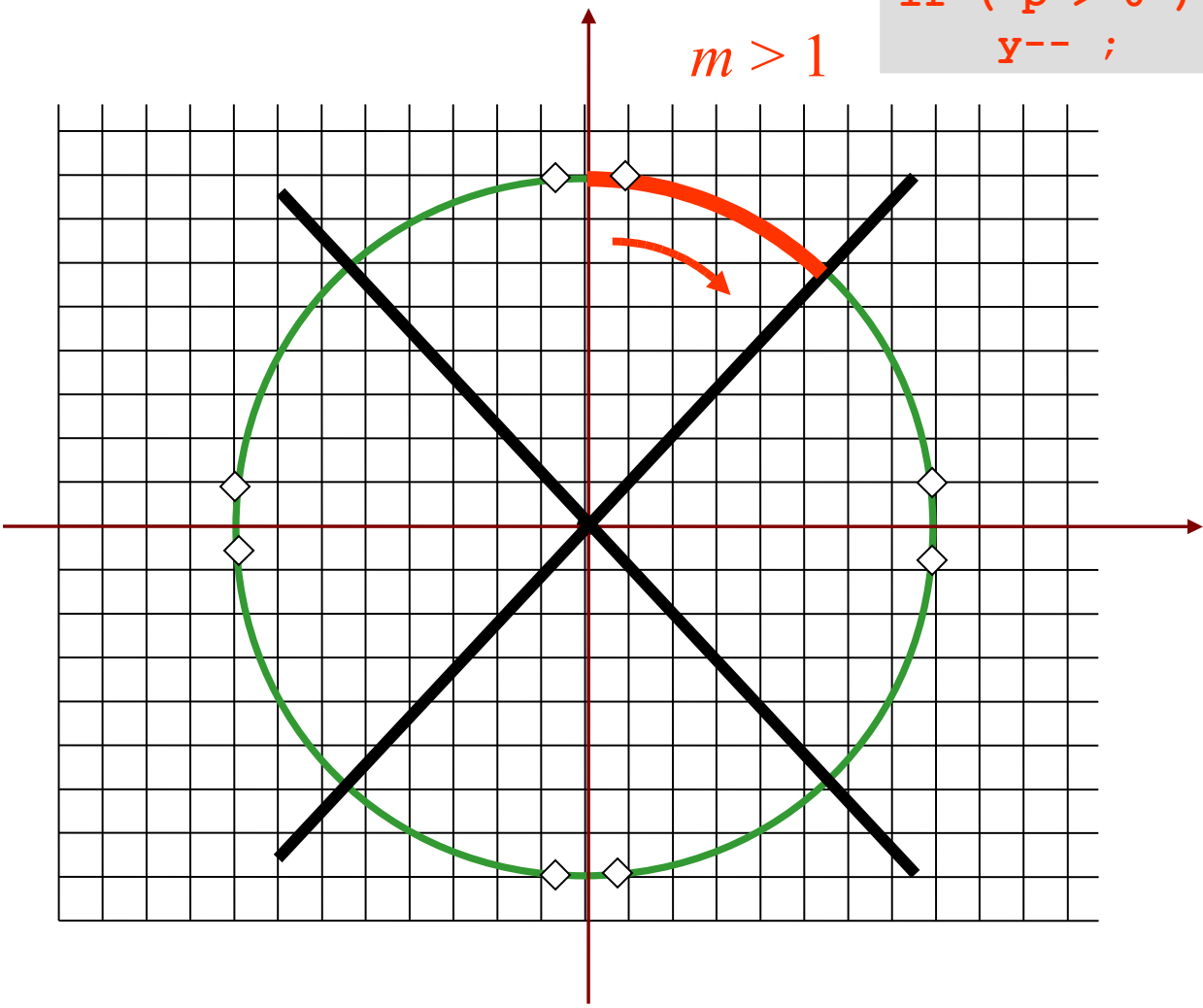
```
void line( int xA, int yA, int xB, int yB )  
// Special case: shallow increasing slope.  
{  
    const int DX= xB - xA ;  
    const int DY= yB - yA ;  
    const int DP_FLAT= 2*DY ;  
    const int DP_JUMP= DP_FLAT - 2*DX ;  
    assert( 0 < DY && DY < DX ) ;  
  
    int x= xA ;  
    int y=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

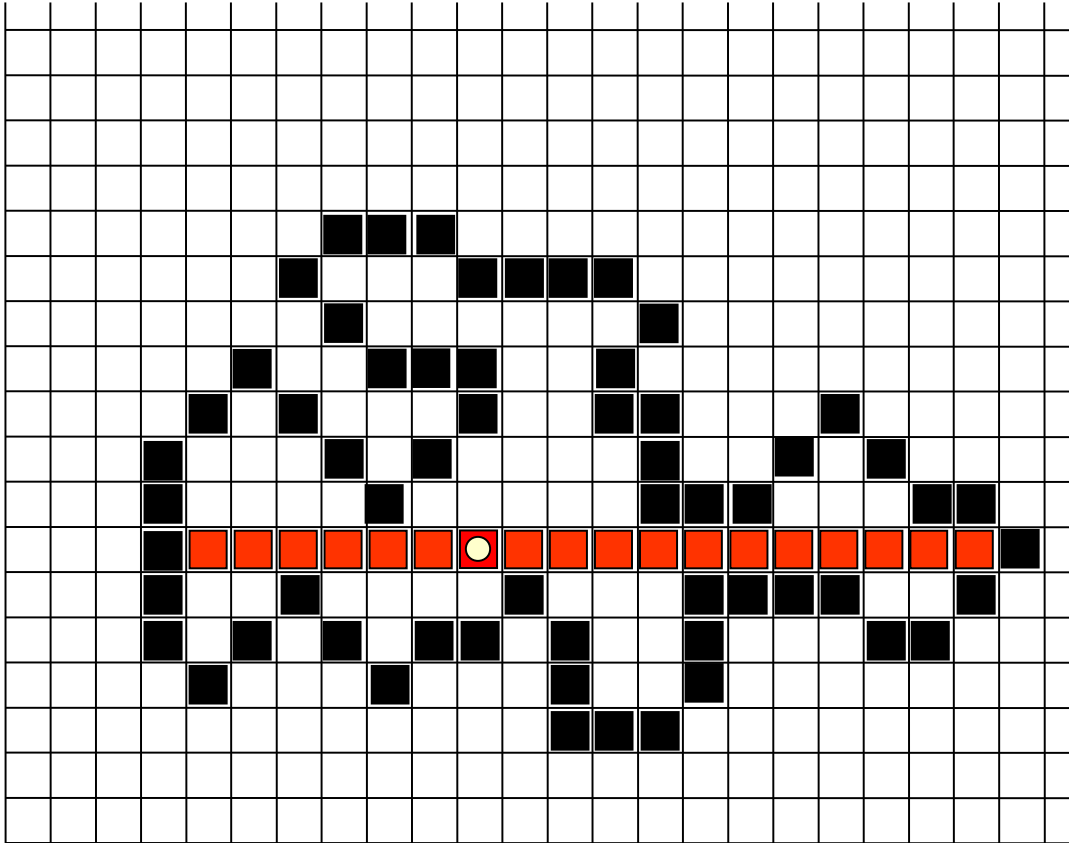


```
x++  
if ( p > 0 )  
    y-- ;
```

$m > 1$

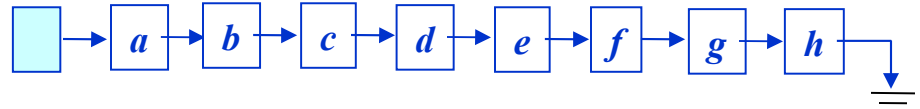




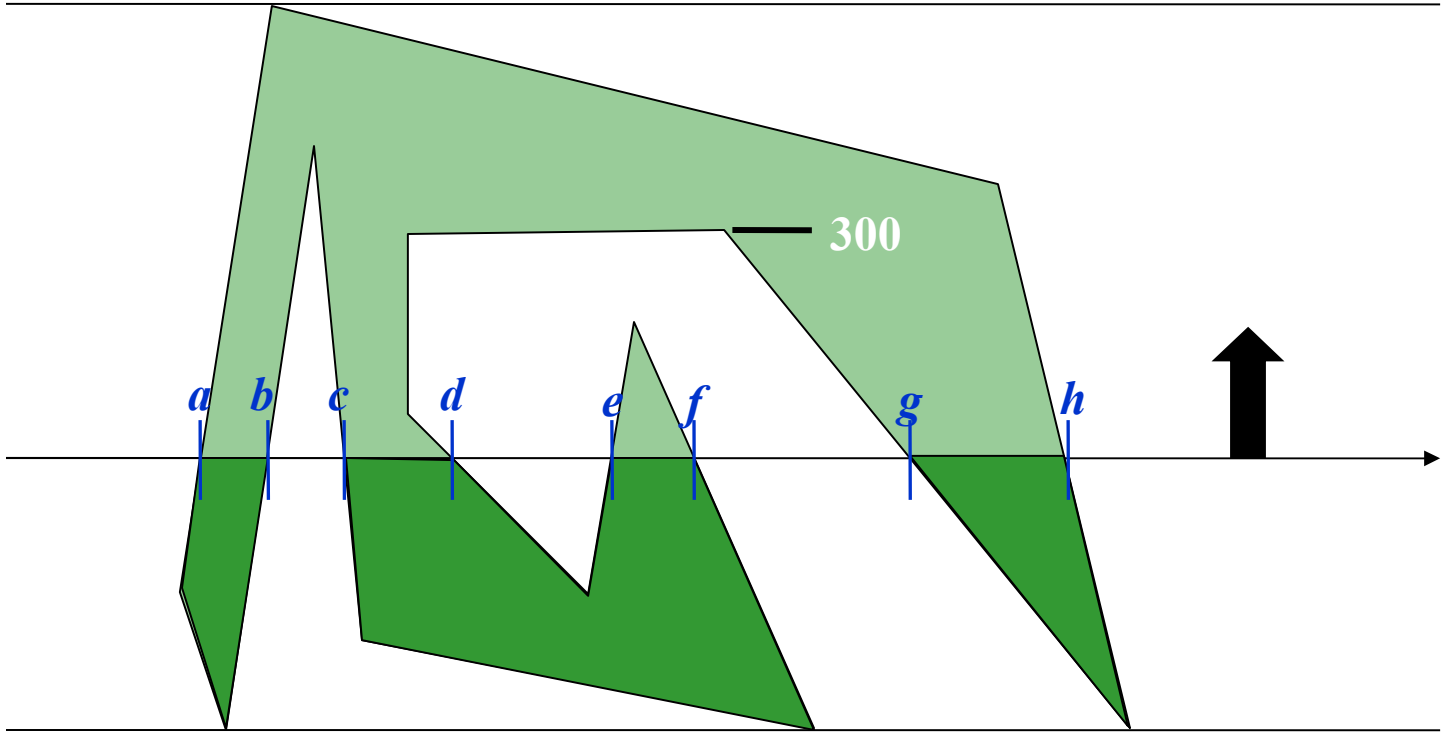


*Fill the  
scan line*





$y_{\max}$



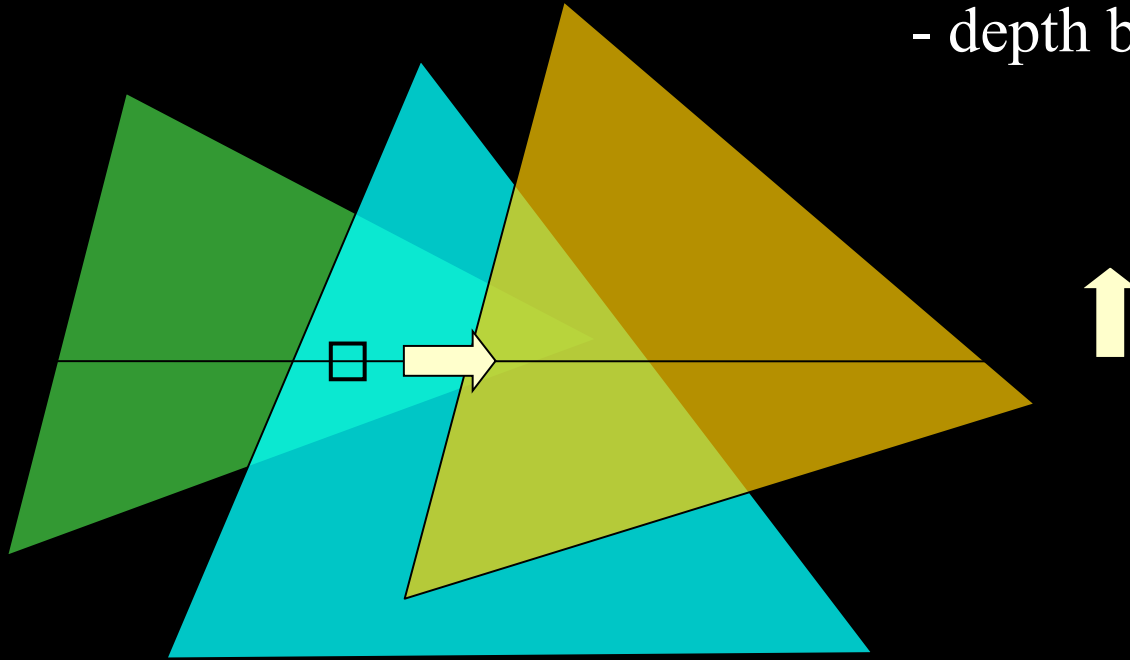
$y_{\min}$



# Z-Buffering

## Two buffers

- screen buffer (color)
- depth buffer



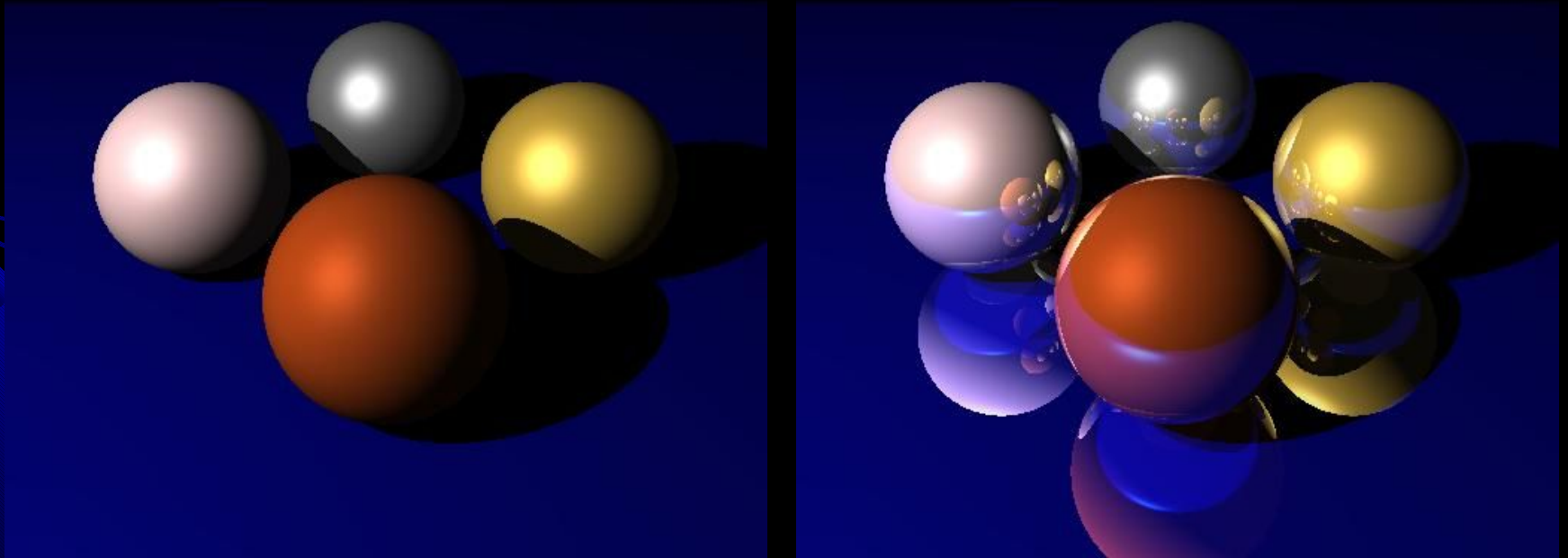
```
for each polygon
  for each scanline
    for each pixel in scanline
      update depth at pixel
      if pixel depth < buffered depth
        write to screen & depth buffers
```

# Ray Tracing

Mani Thomas  
CISC 440/640  
Computer Graphics

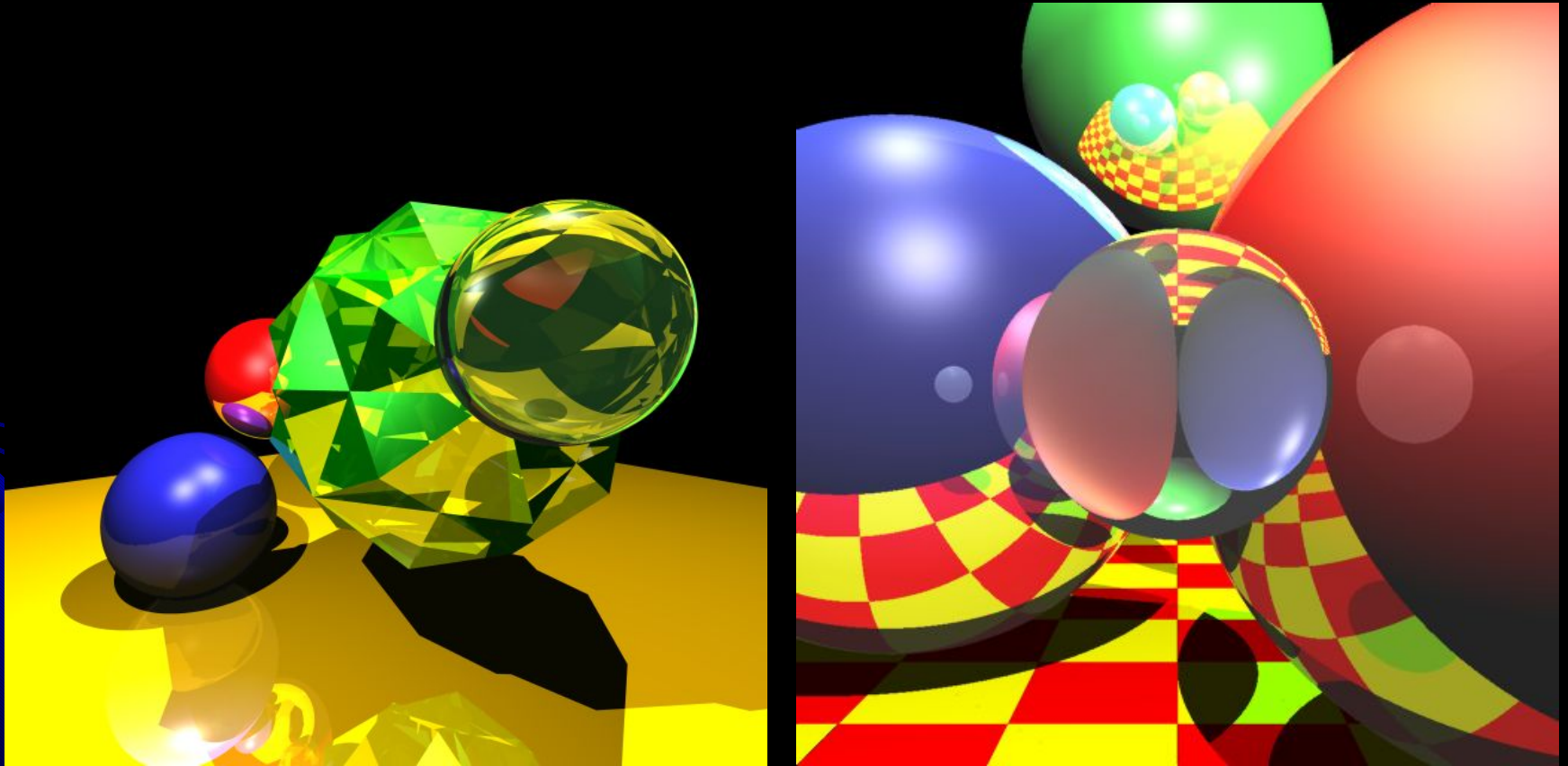


# Fotorealismus

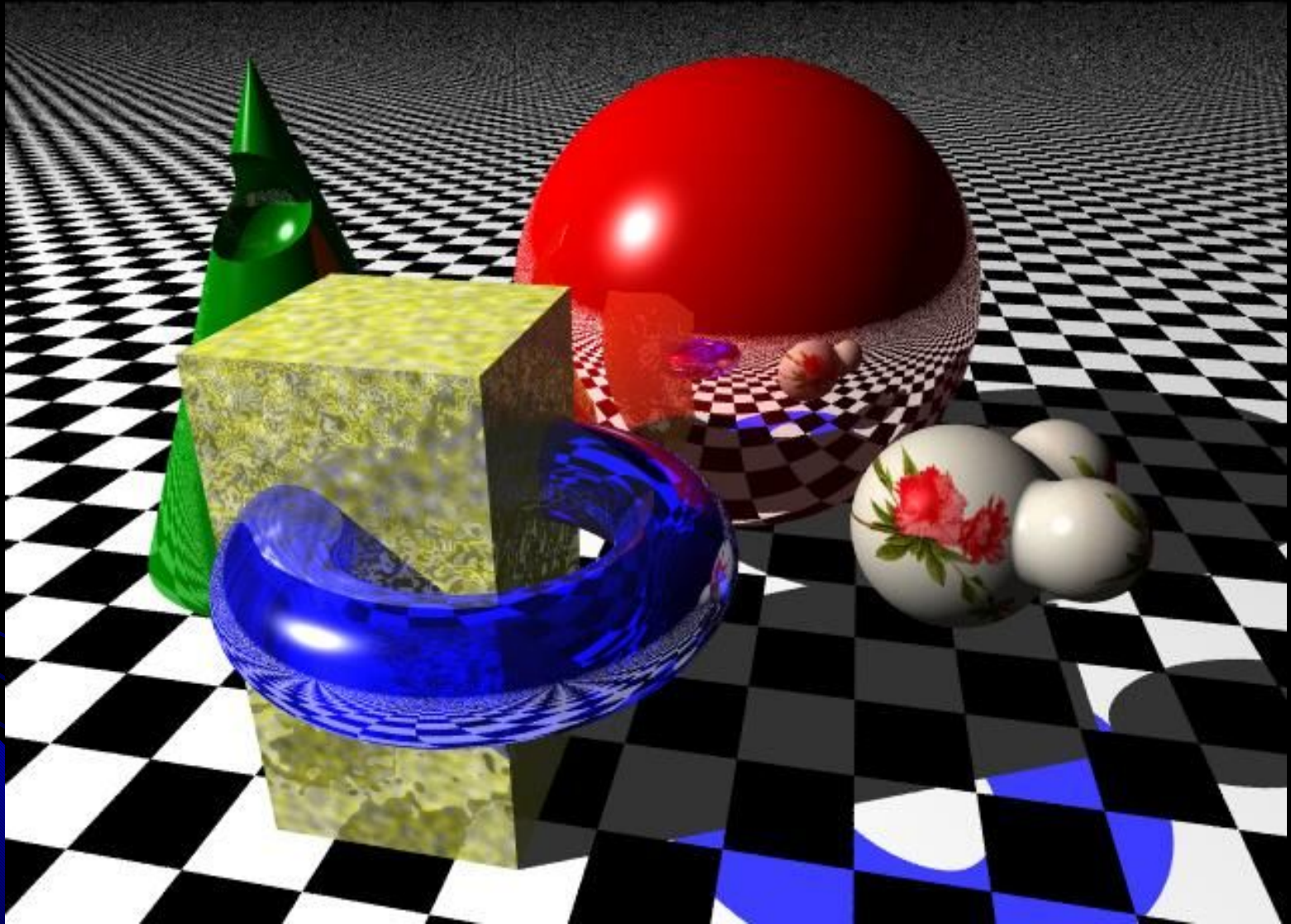


*Created by David Derman – CISC 440*

# Fotorealismus



*Created by Jan Oberlaender – CISC 640*

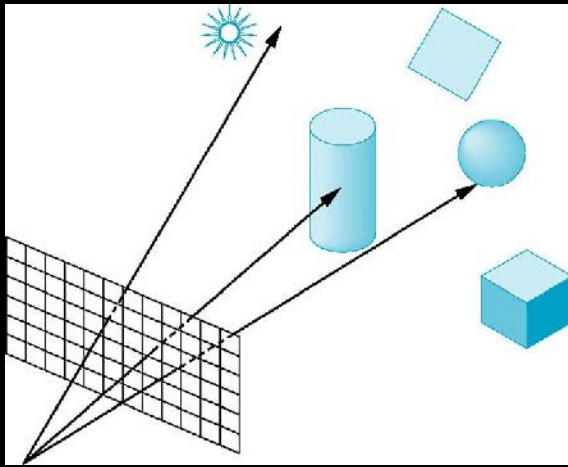


Created by Donald Hyatt  
<http://www.tjhsst.edu/~dhyatt/superap/povray.html>

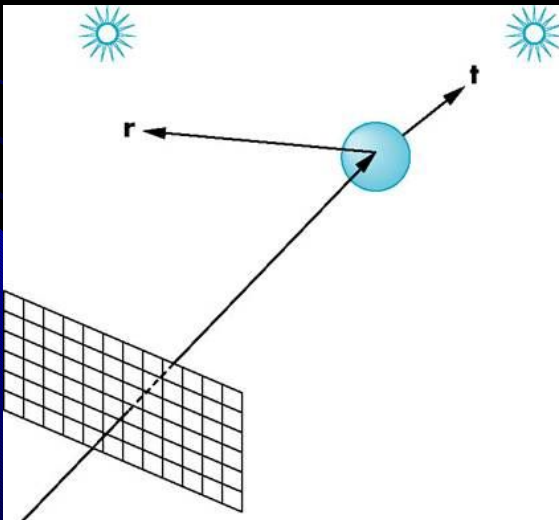
# Úvod

- 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

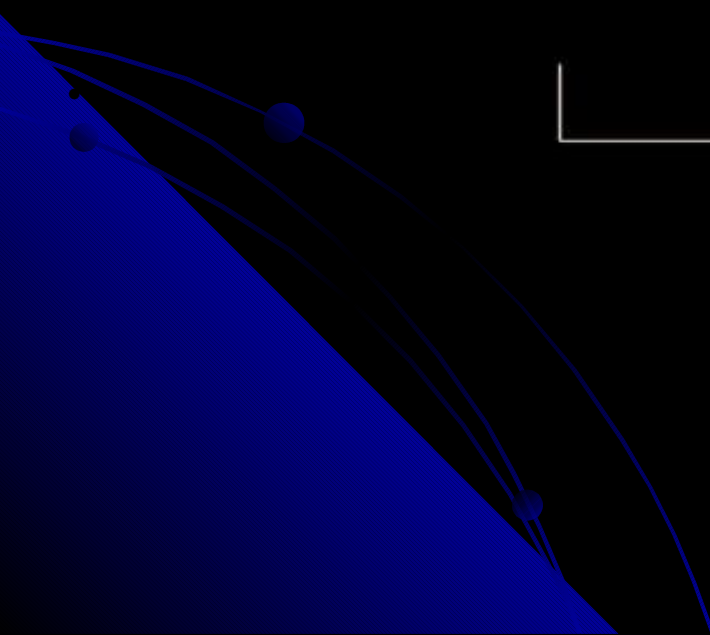
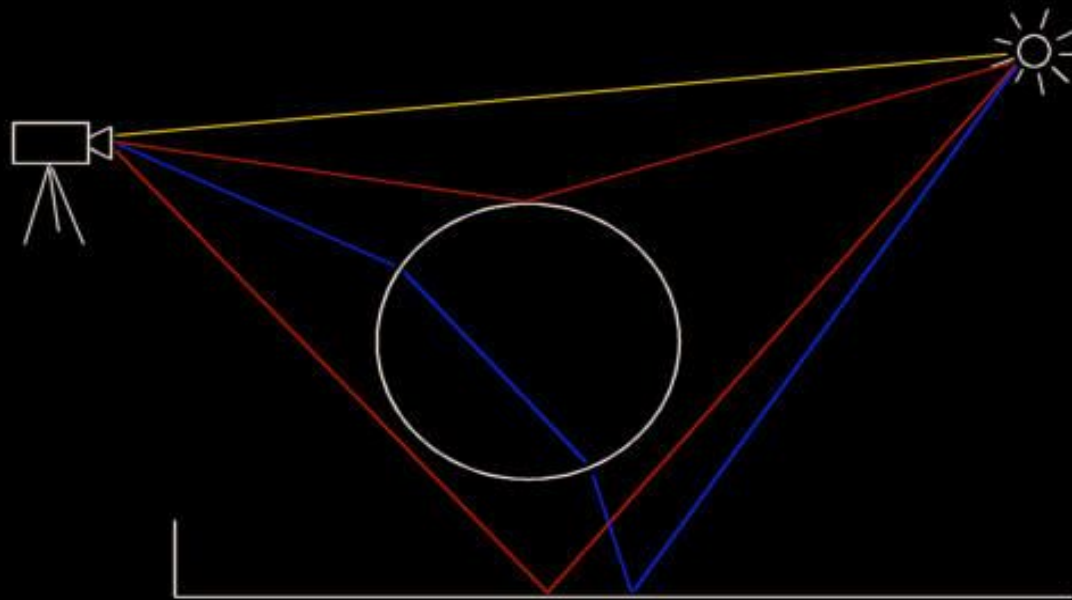


- Ray Casting
  - Paprsky se zastaví na prvním objektu



- Ray Tracing
  - Rekurze předcházejícího principu

# Šíření světla



# Typy paprsků

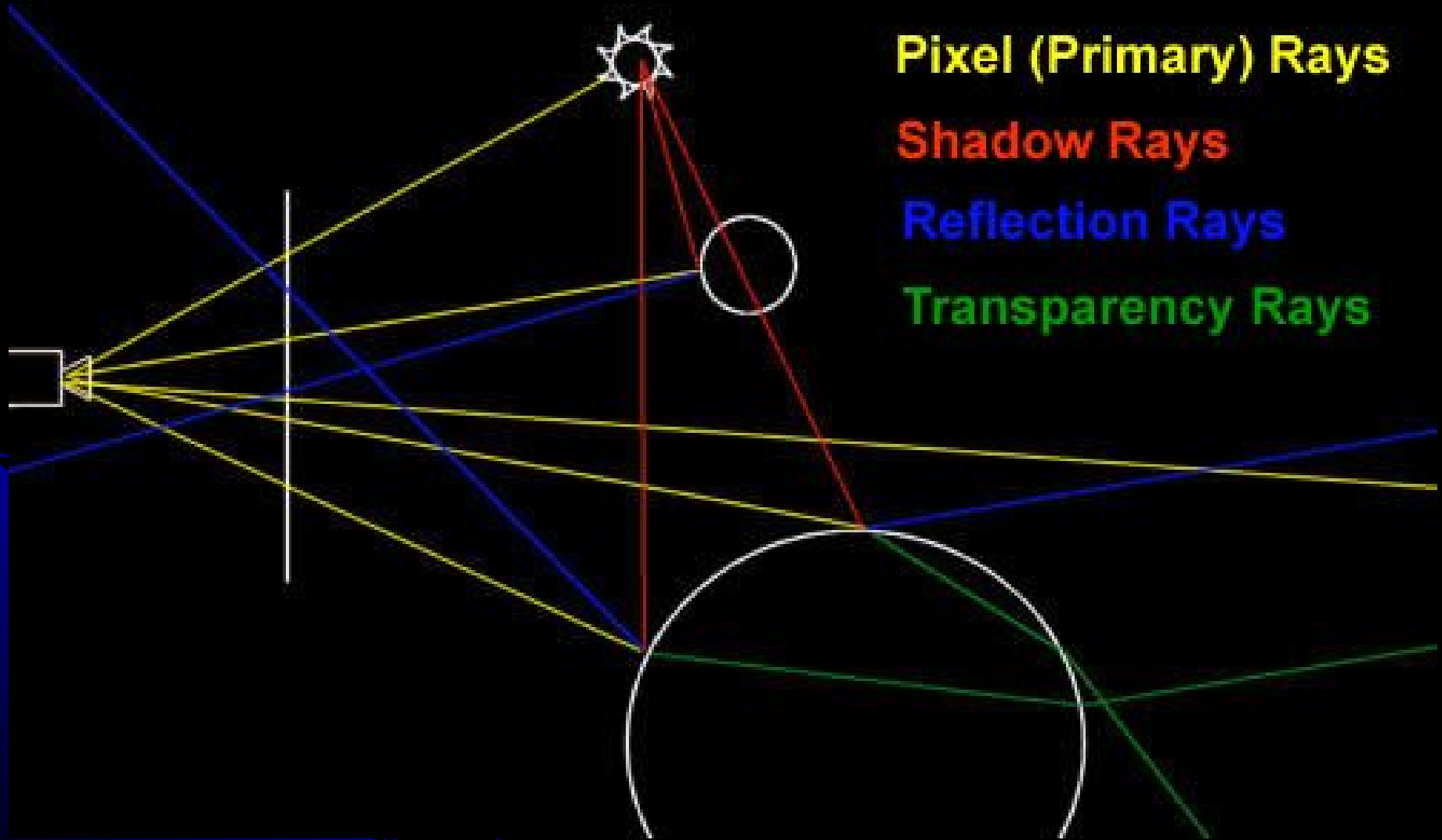


Image copyright Jacco Bikker.

# Algorithmus – 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++)  
  {
```

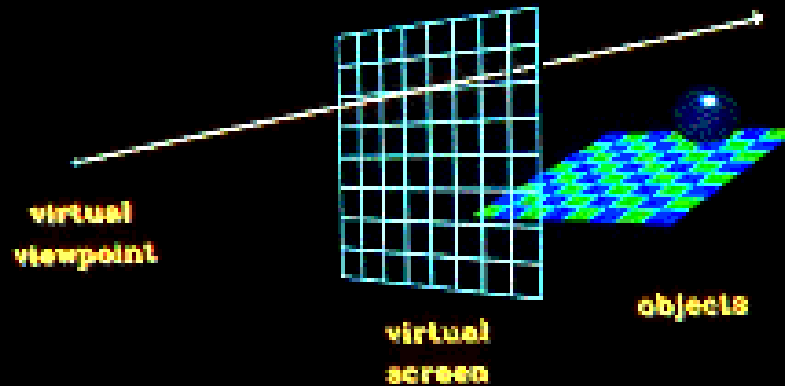
- 1. Build the rc-th ray*
- 2. Find all intersections of the rc-th ray with objects in the scene*
- 3. Identify the intersection that lies closest to, and in front of, the eye*
- 4. Compute the "hit point" where the ray hits this object, and the normal vector at that point*
- 5. Find the color of the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

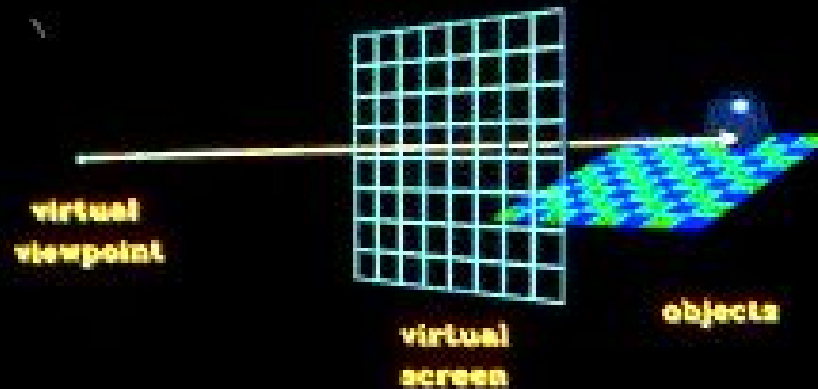


# Ray Tracing



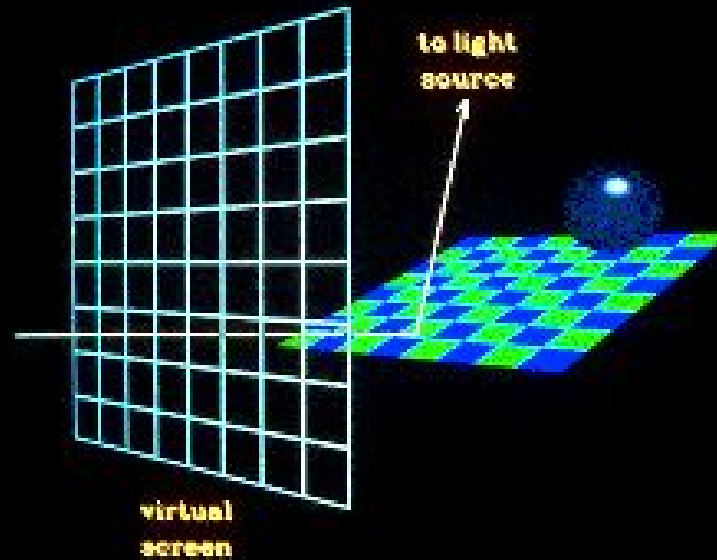
*created by Michael Sweeny, et al for ACM SIGGRAPH Education slide set 1991*

# Ray Tracing



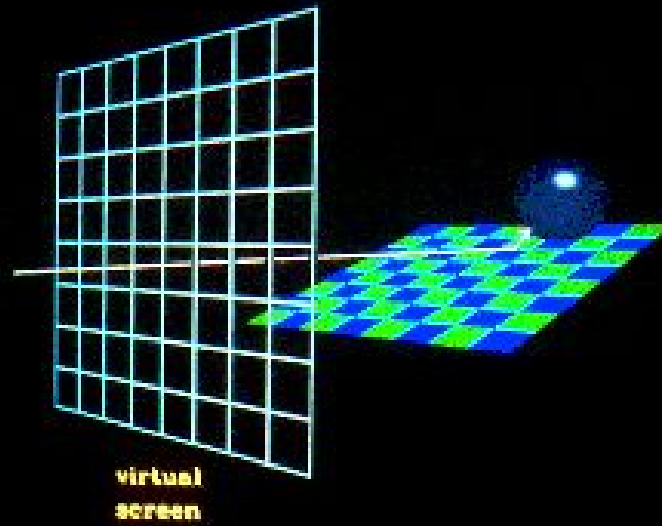
*created by Michael Sweeny, et al for ACM SIGGRAPH Education slide set 1991*

# Ray Tracing



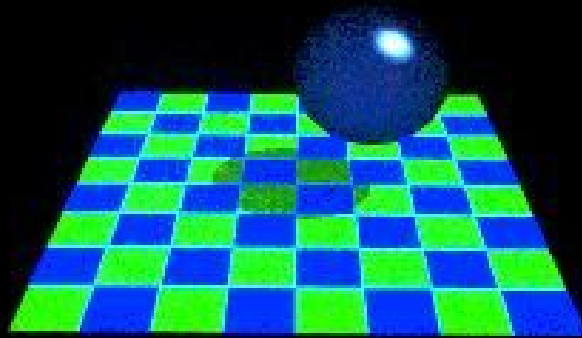
*created by Michael Sweeny, et al for ACM SIGGRAPH Education slide set 1991*

# Ray Tracing



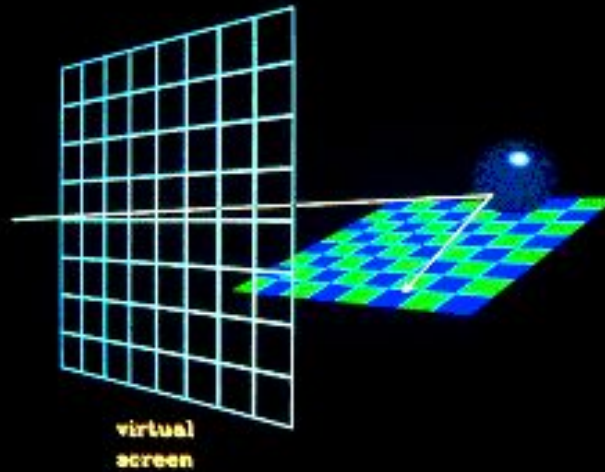
*created by Michael Sweeny, et al for ACM SIGGRAPH Education slide set 1991*

# Ray Tracing



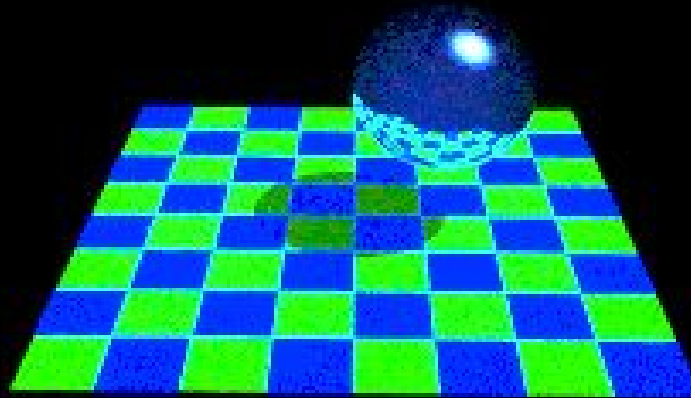
*created by Michael Sweeny, et al for ACM SIGGRAPH Education slide set 1991*

# Ray Tracing



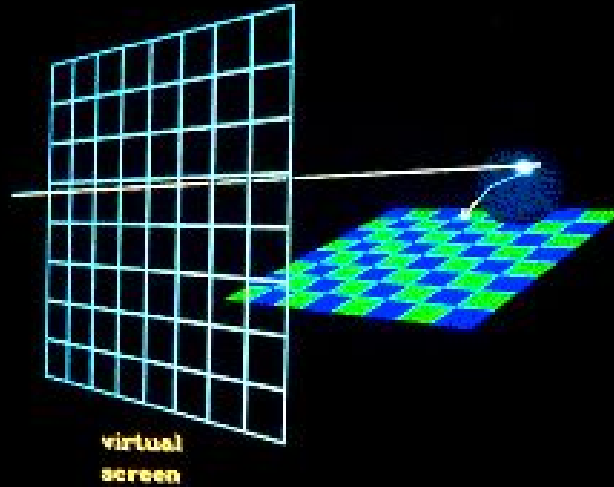
*created by Michael Sweeny, et al for ACM SIGGRAPH Education slide set 1991*

# Ray Tracing



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

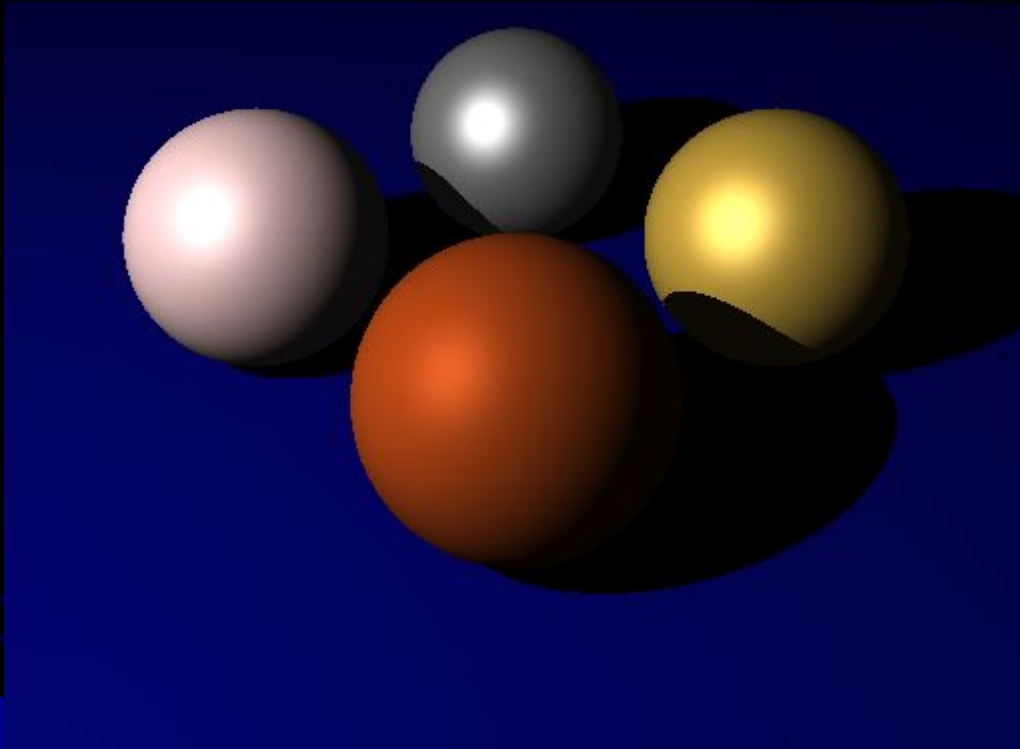
# Ray Tracing



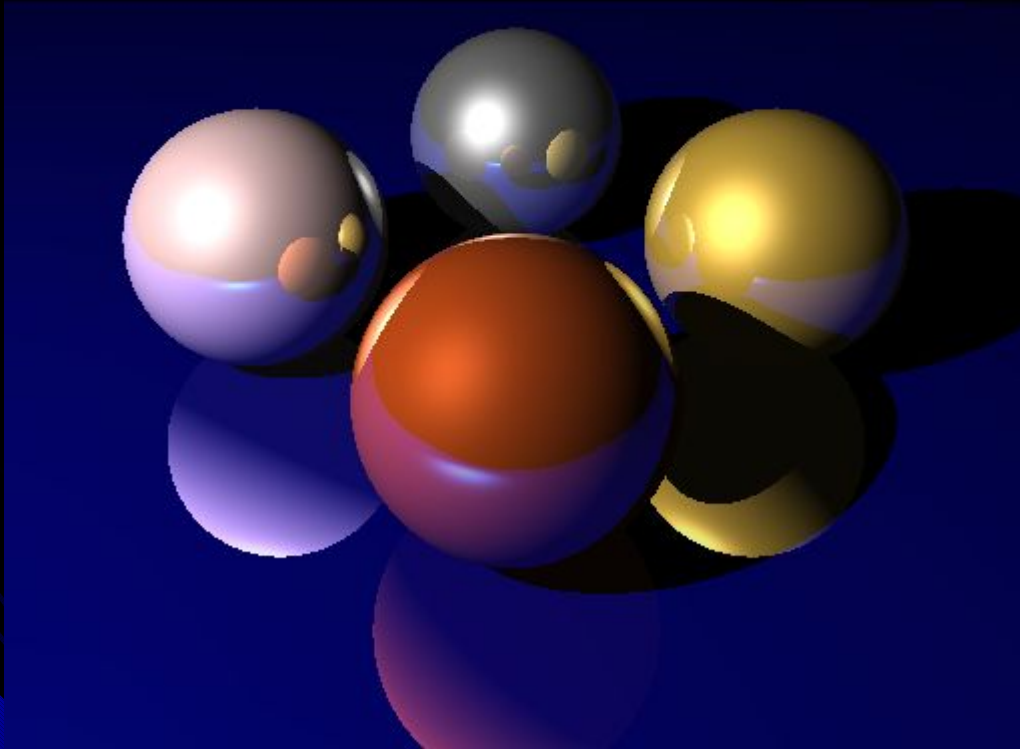
*created by Michael Sweeny, et al for ACM SIGGRAPH Education slide set 1991*



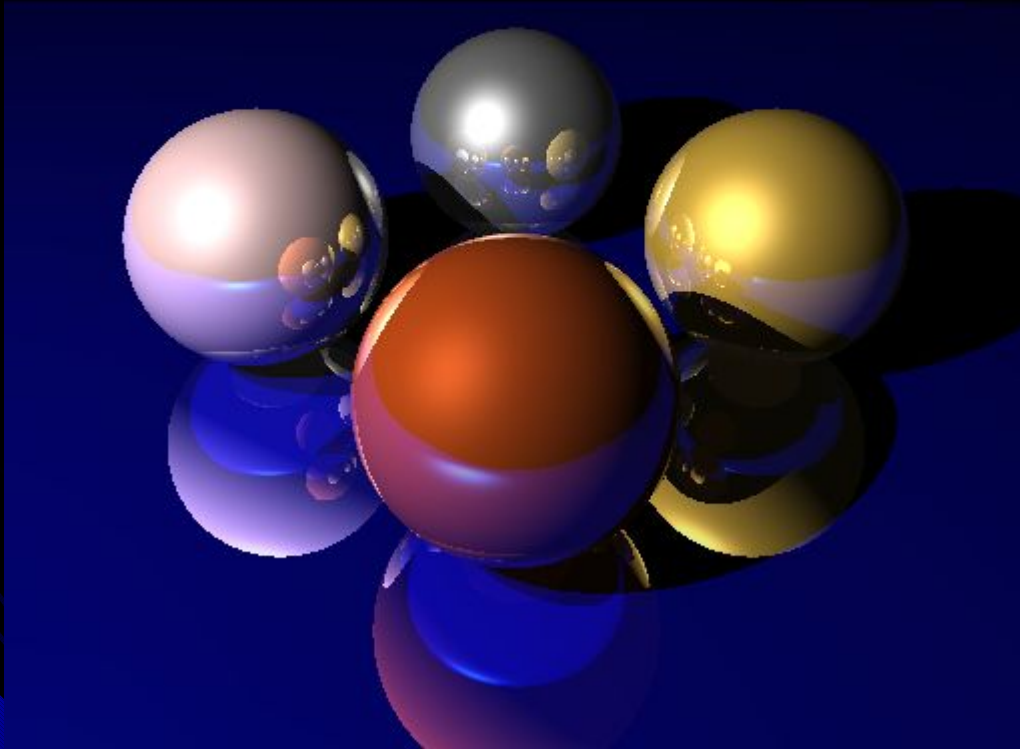
# Odraz



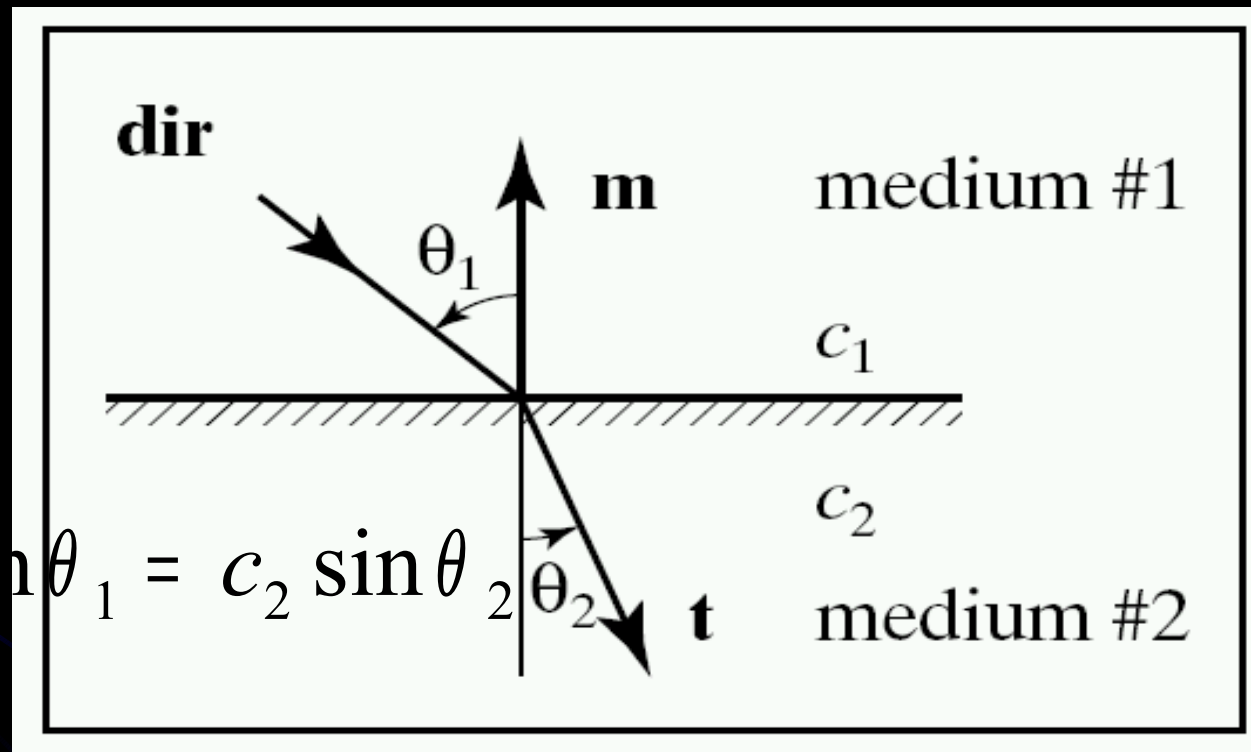
# Odraz



# Odraz



# Lom světla



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

# Jiné efekty

## Hloubka ostrosti

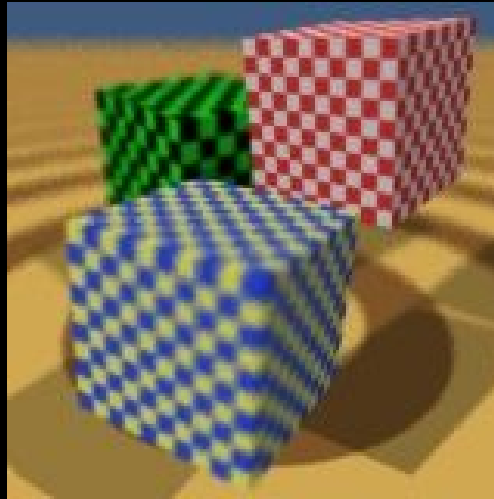


Image copyright  
Josef Pelikan  
<http://cgg.ms.mff.cuni.cz/gallery/>

# Jiné efekty

## Rozmazání pohybem

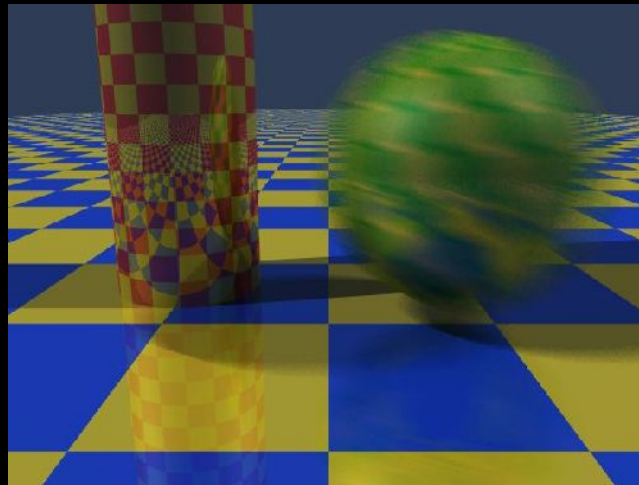
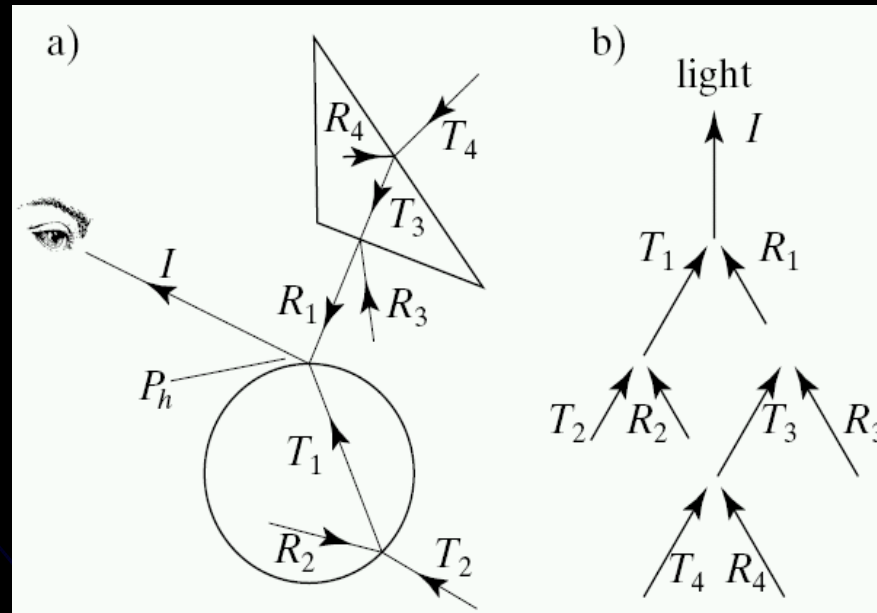


Image copyright  
Josef Pelikan  
<http://cgg.ms.mff.cuni.cz/gallery/>

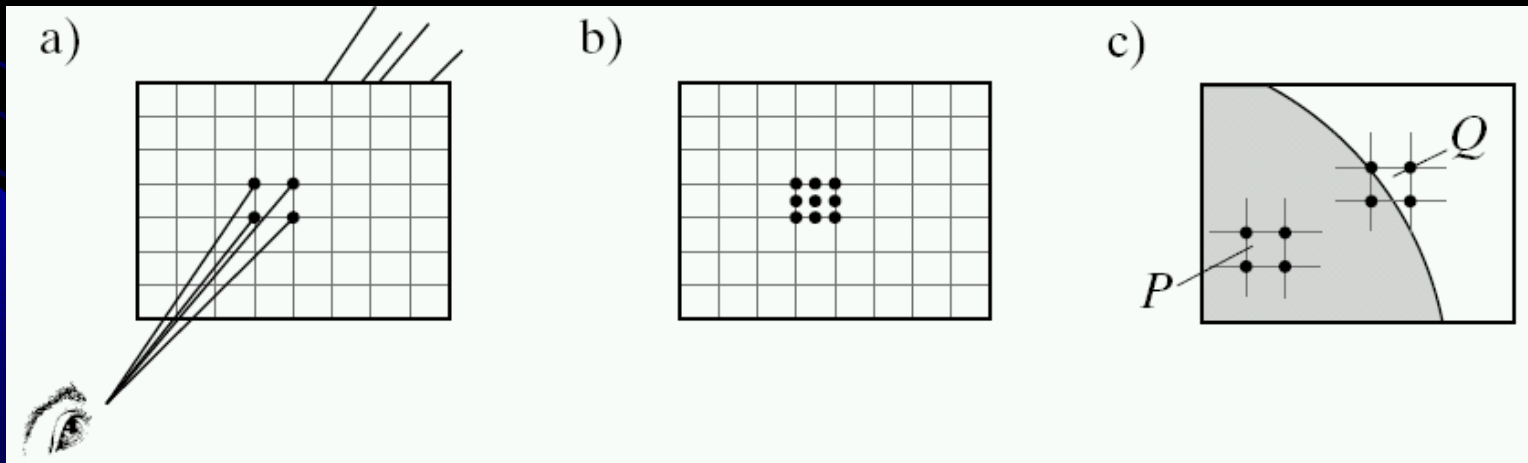
# Strom světla

- Informace o paprsku sčítají



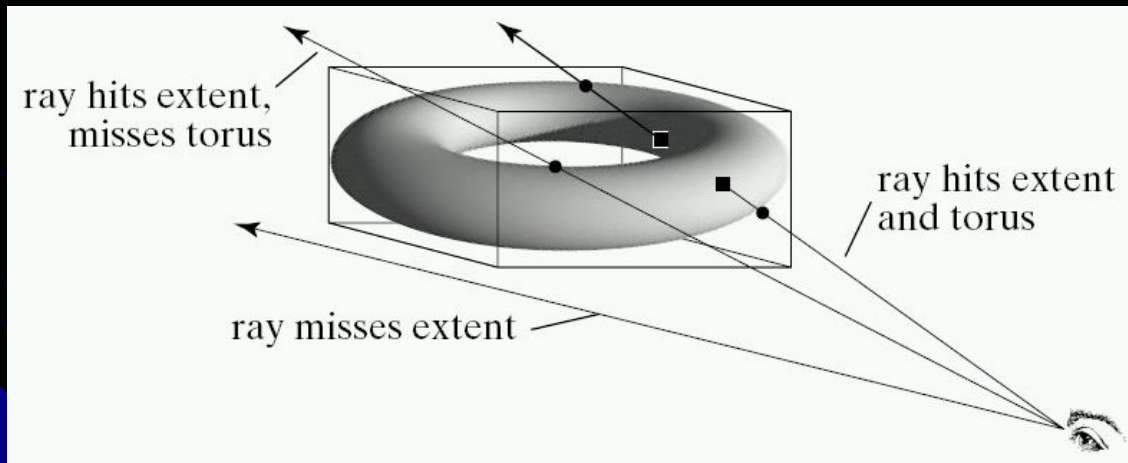
# Super-sampling

- Vyhlazení hran

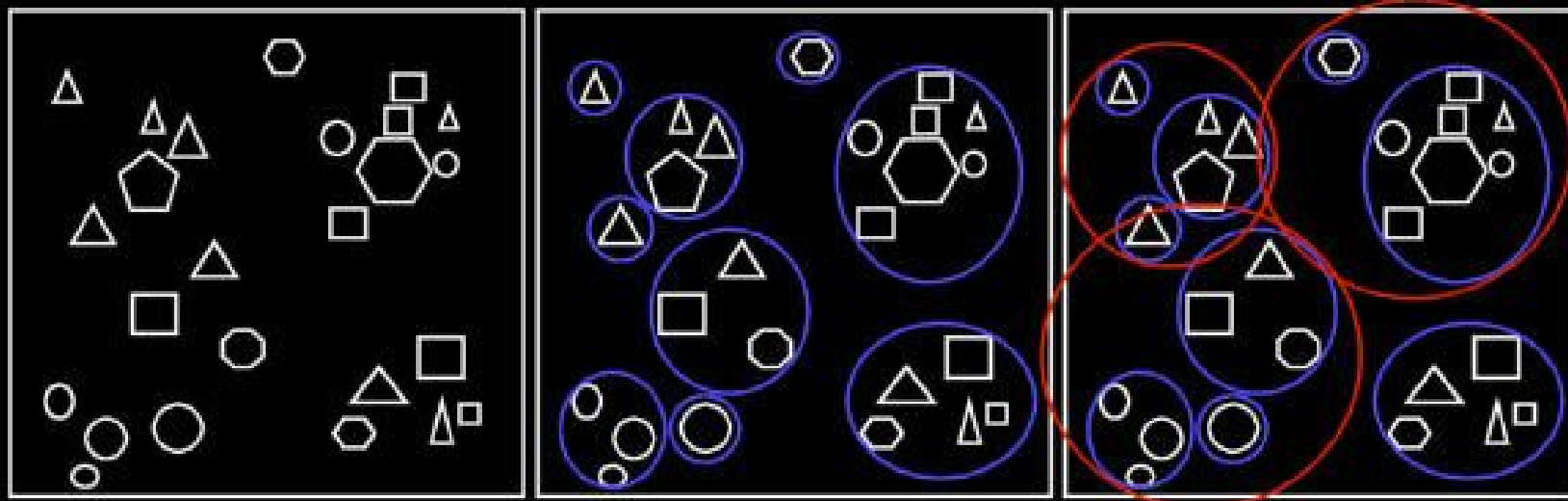




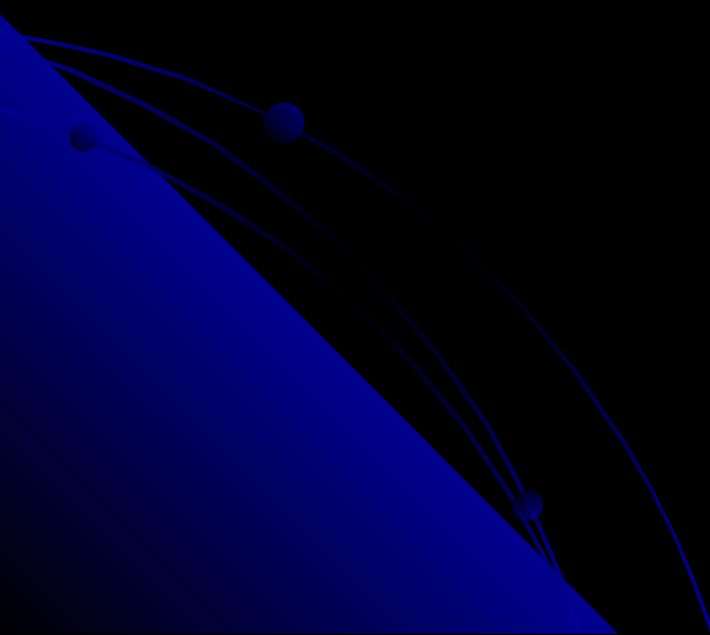
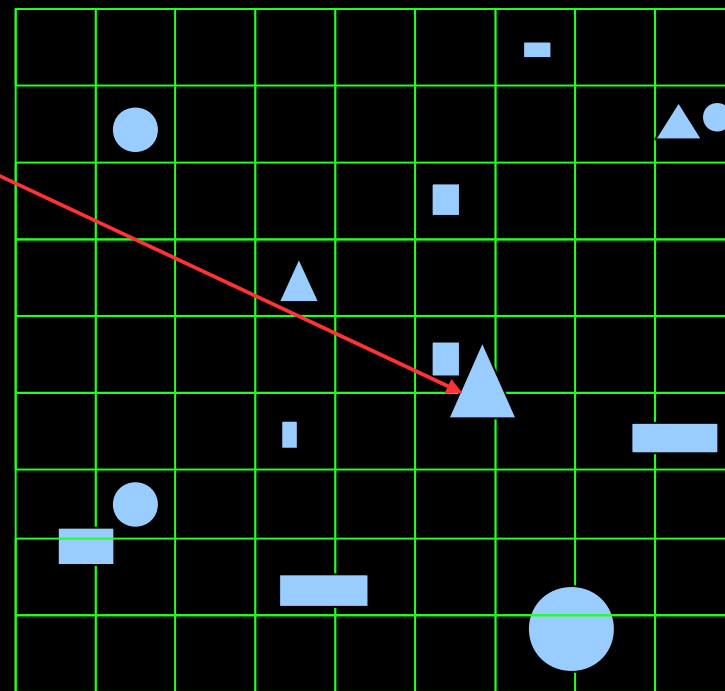
# Obal



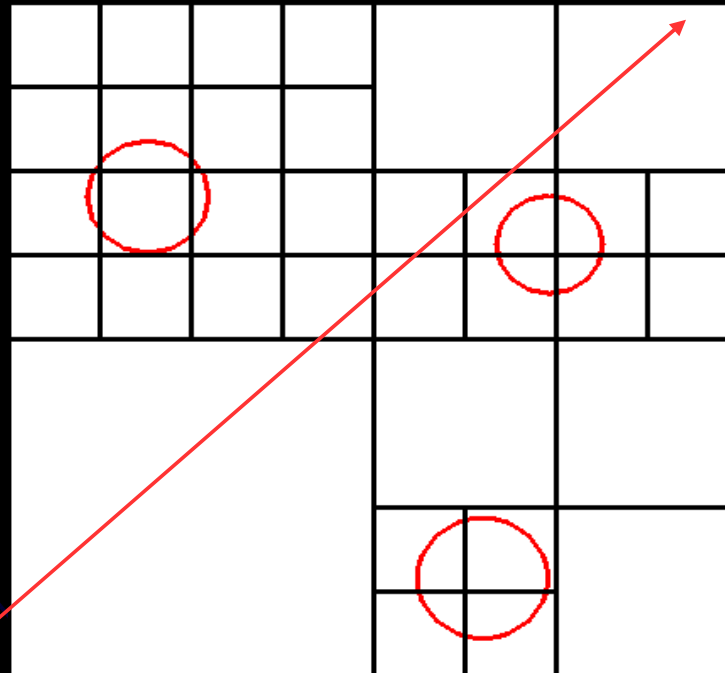
# Obal skupiny objektů



# 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.

# 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



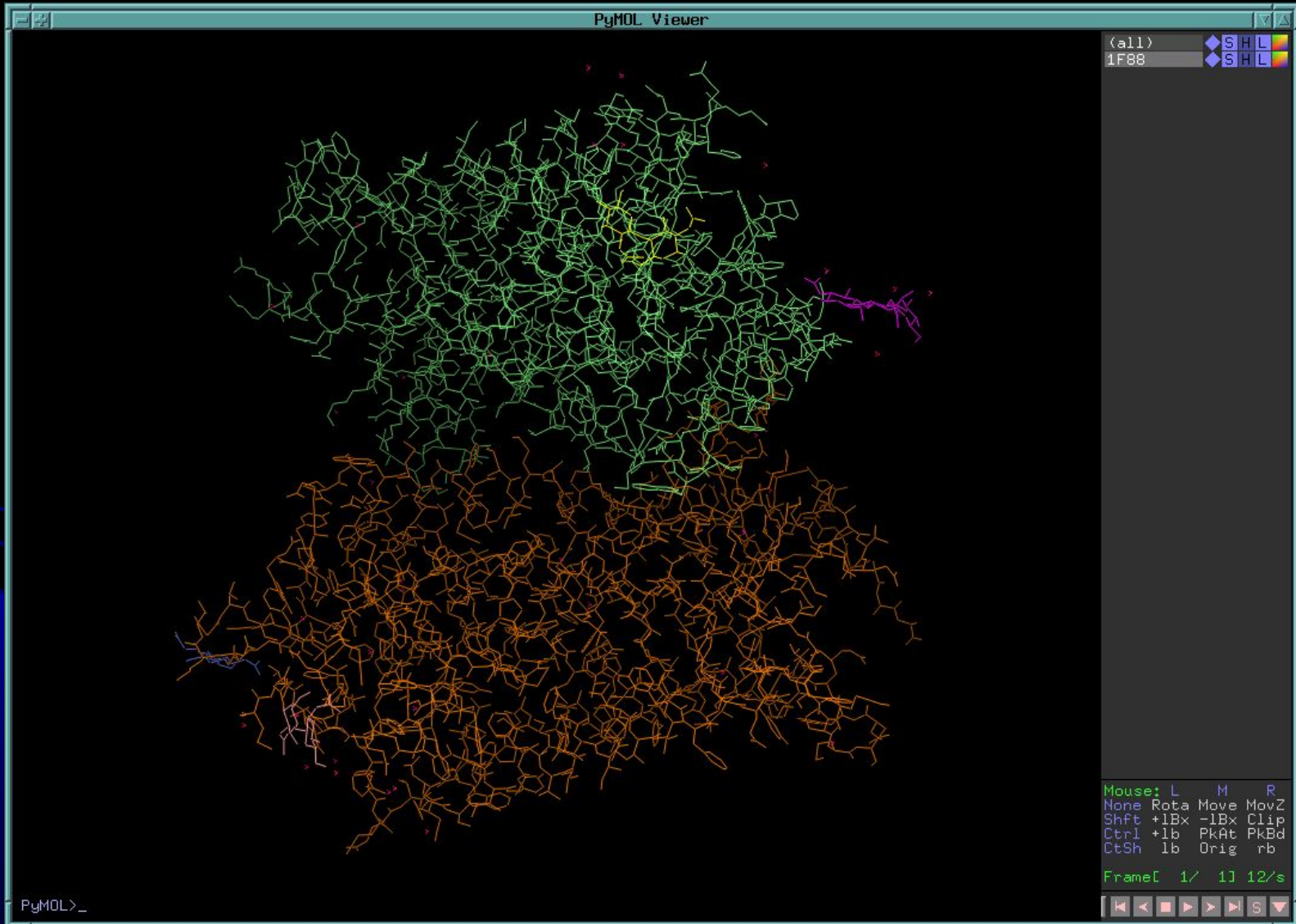
# Application Feature Summary

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

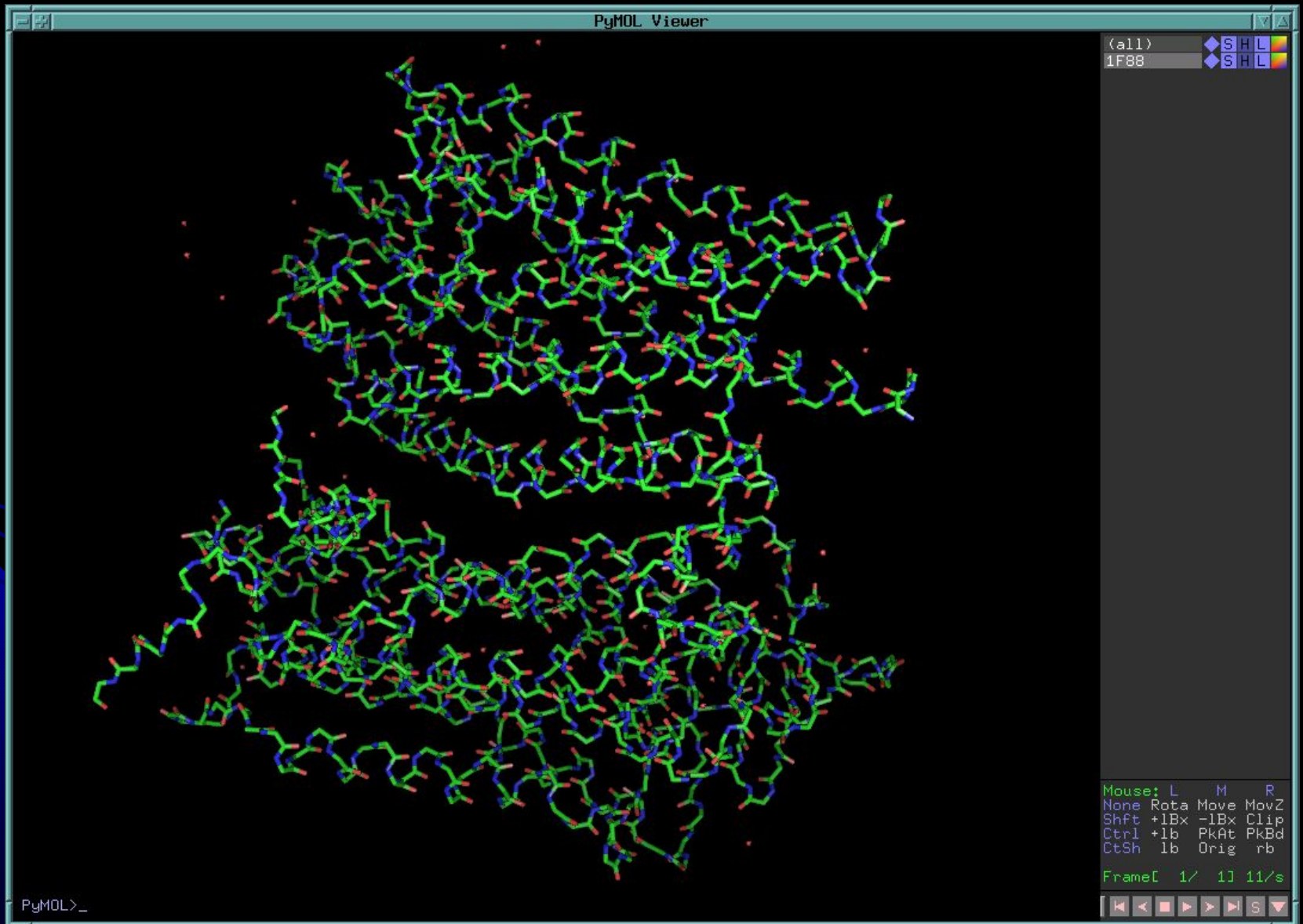
# Molecule Representations

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

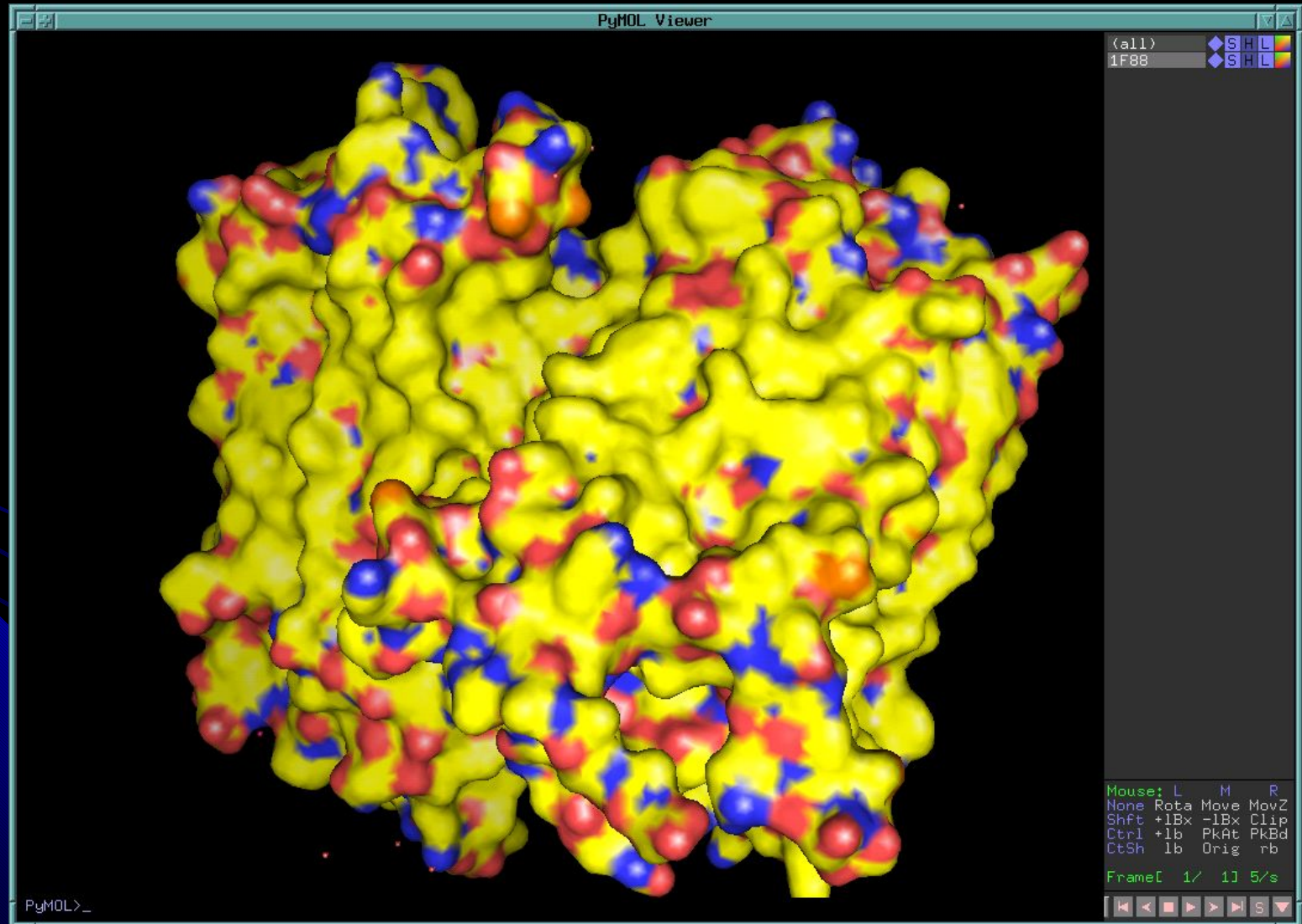
Wireframe used to show individual chains:



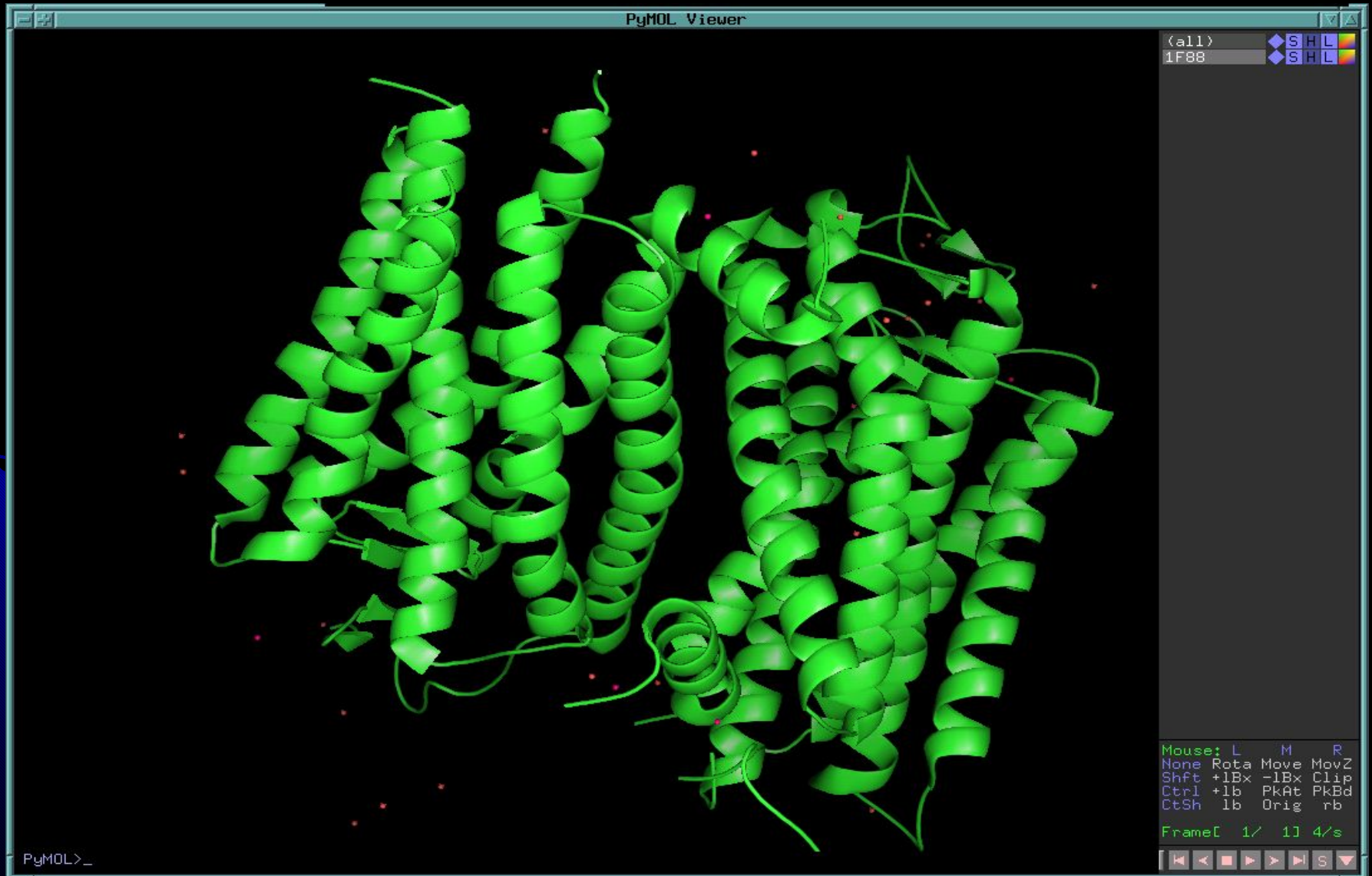
# Stick view showing atoms and bonds:



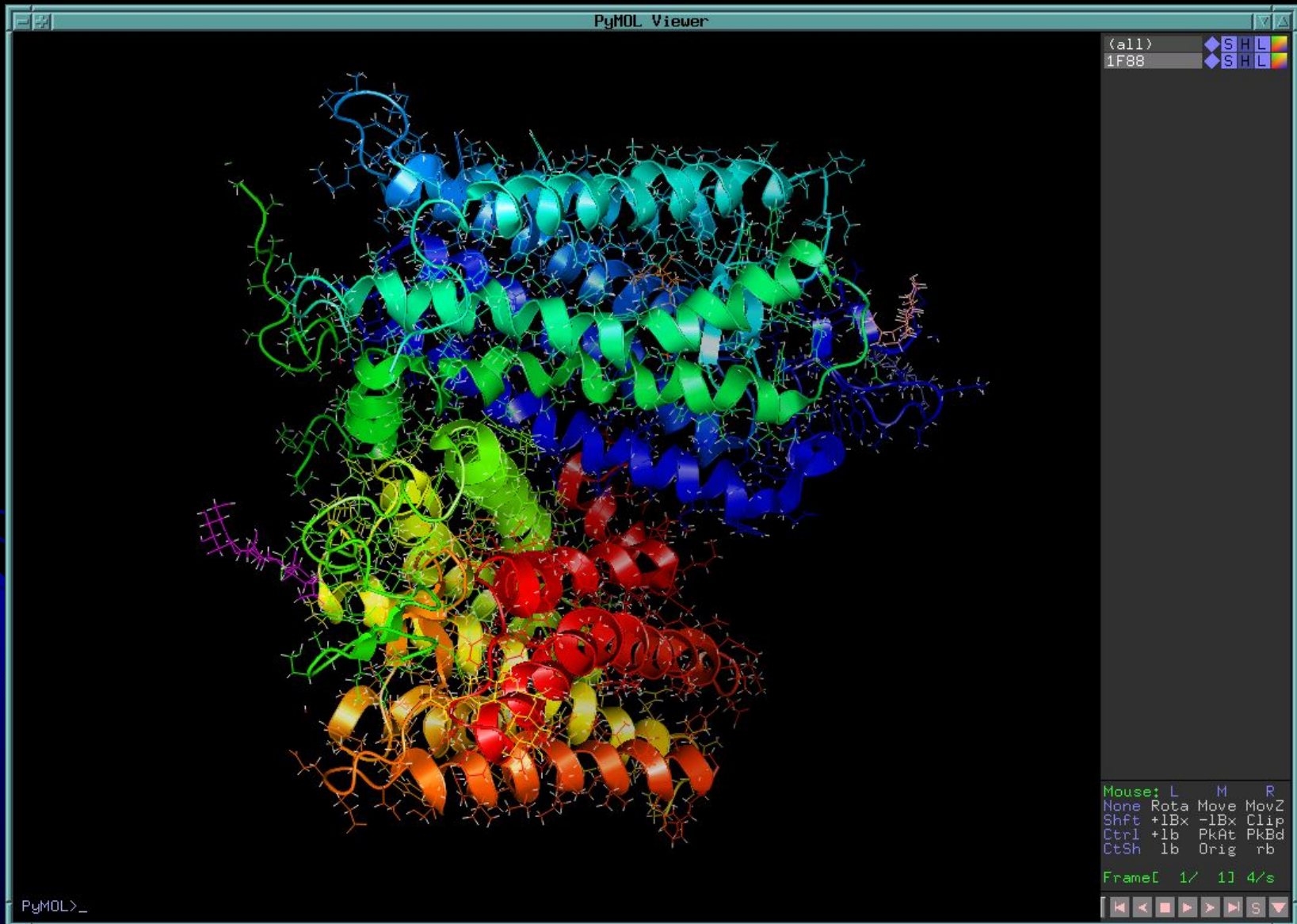
# Surface View showing surface fields:



# Ribbon view of secondary structure:

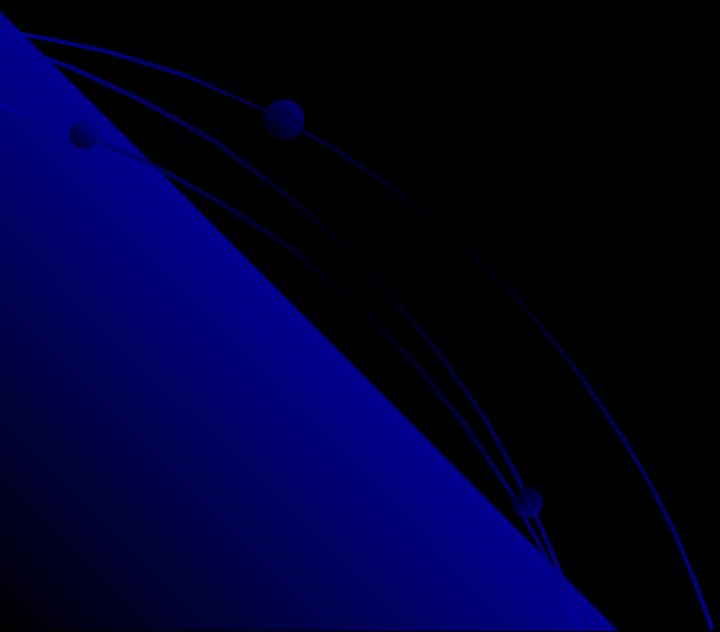


# Distinct geometrical features by color:



# 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.

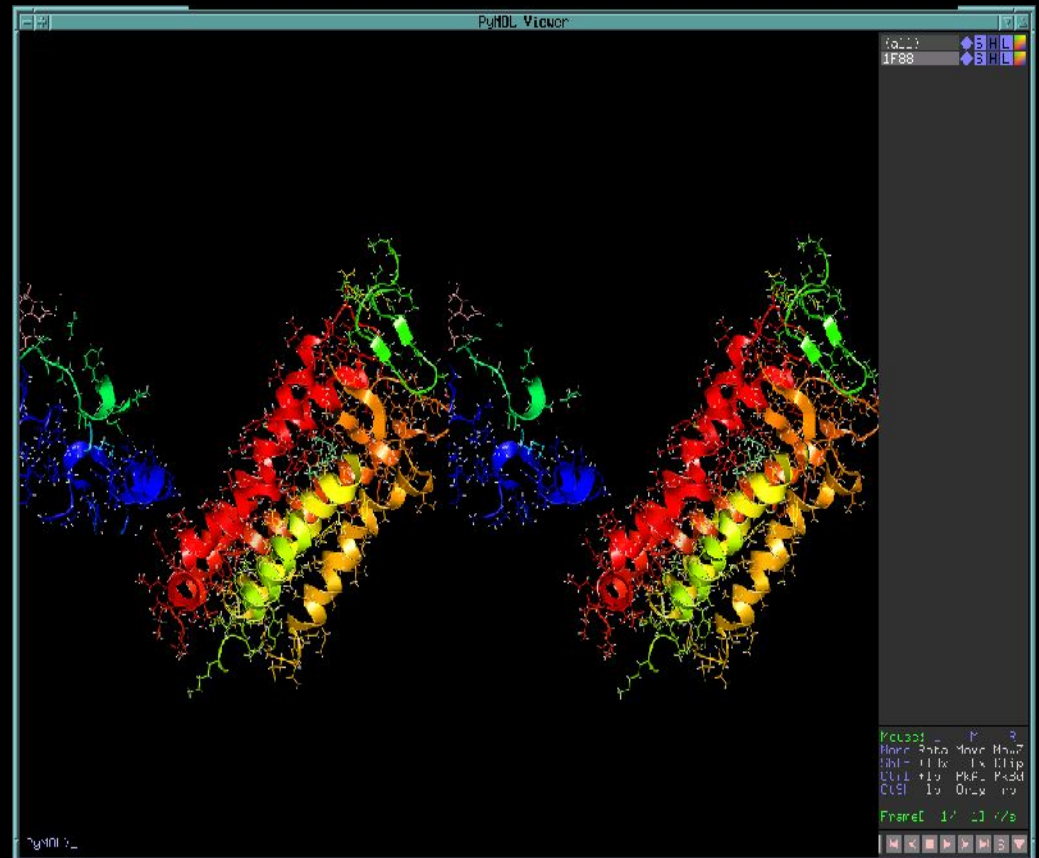




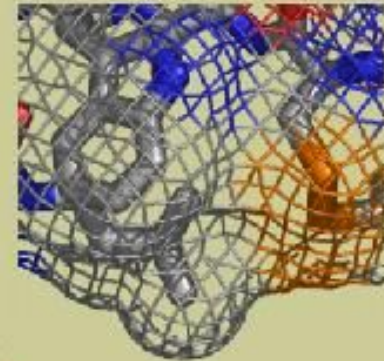
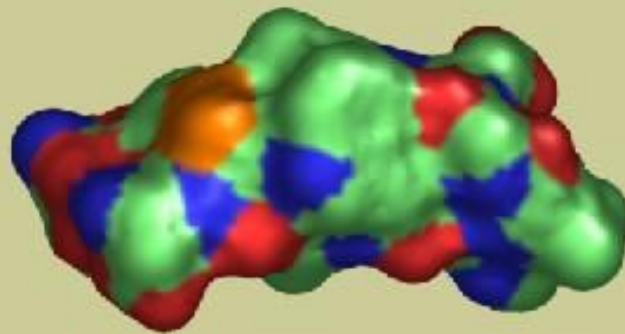
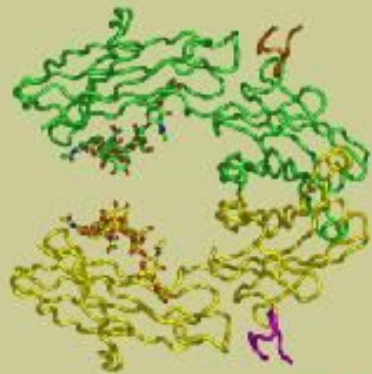
# For Protein interactions, we need a metaphor that reveals dynamics

- 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.

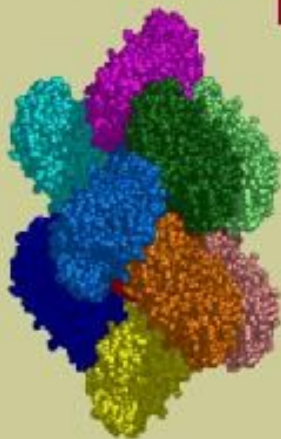


# The PYMOL Molecular Graphic System



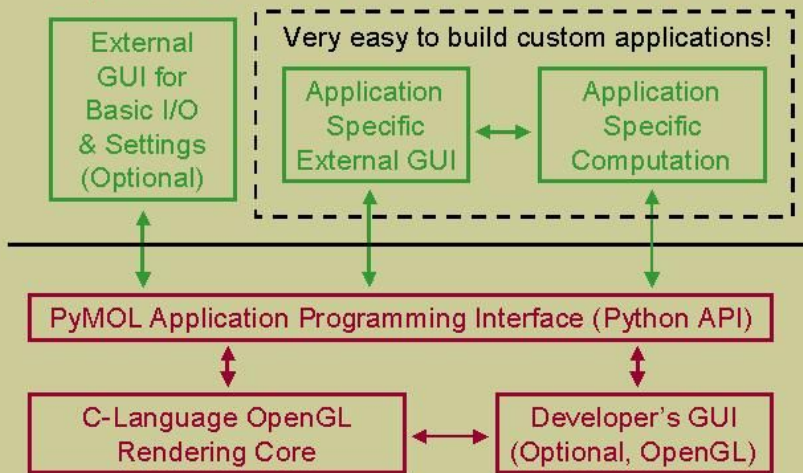
## The PyMOL Molecular Graphics System

*An extensible open-source alternative to commercial visualization software.*



# PYMOL

## PyMOL's Modular Architecture



(C) 2000 by Warren L. DeLano (www.ddanoscientific.com)

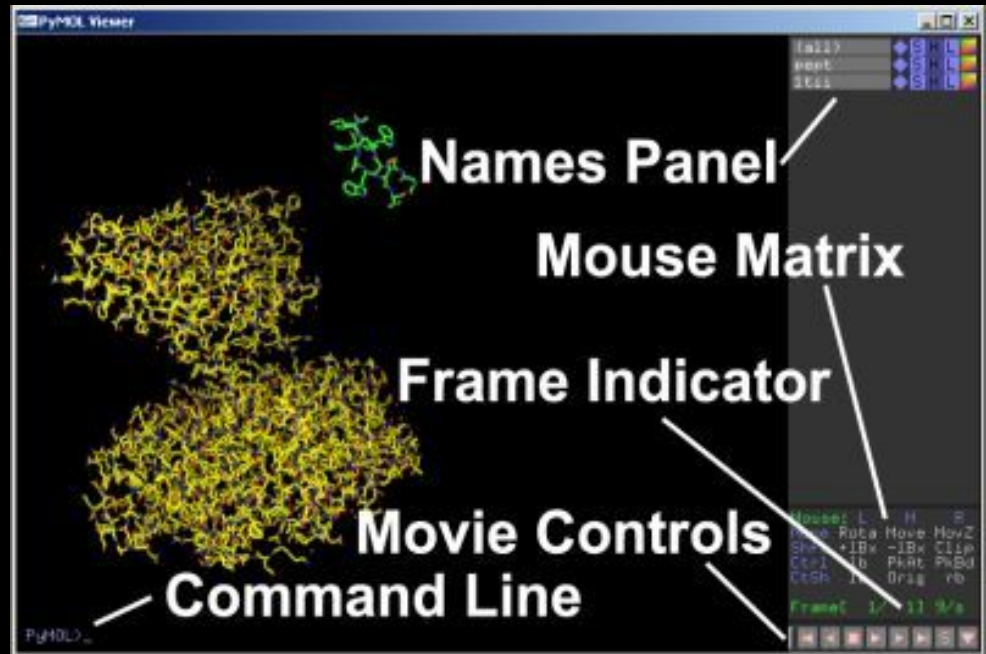
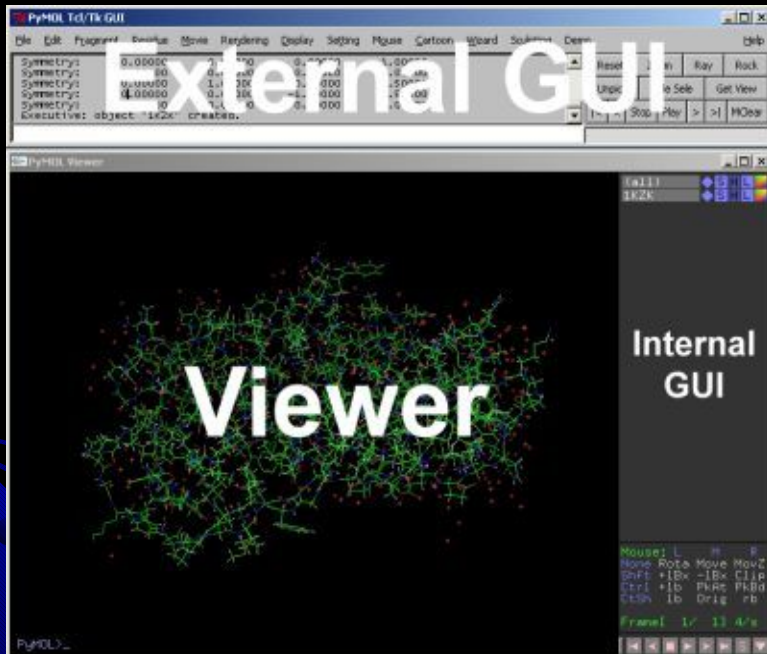
## PyMOL's Top Features

- Real-Time 3D Visualization
- Publication Quality Renderings
- Extensive Animation Capabilities
- Support for X-ray Crystallography
- Modular Architecture
- Flexible API for Custom Applications
- Open Source and Freely Available
- Written in C and Python

(C) 2000 by Warren L. DeLano (www.ddanoscientific.com)

- 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 #

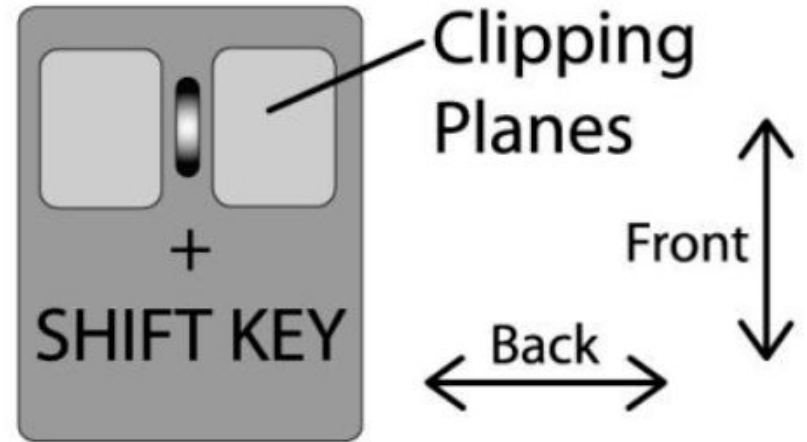
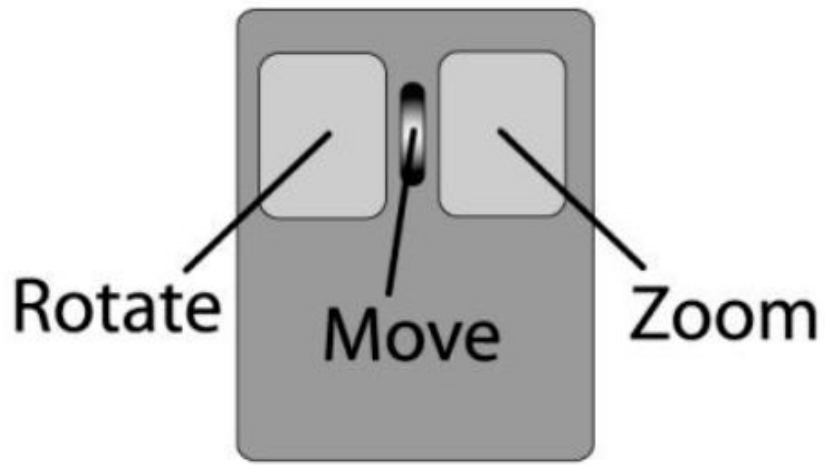
# Load your pdb file

External GUI: file-open

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

The screenshot shows the PyMOL Tk/Tk GUI. The 'File' menu is open, with the 'Open...' option highlighted. A red circle with the number '1' is placed over the 'Open...' option. The main window is titled 'PyMOL Viewer' and contains a large black area with the text 'Load files' in white. A white speech bubble with a black border contains the text 'Load data-file-name', with a red circle containing the number '2' next to it. The bottom of the window shows a command line with the text 'PyMOL->load \$c/1hng.pdb'. The Windows taskbar is visible at the bottom, showing the start button and several open applications including 'text', 'toward more accurat...', 'PyMOL Viewer', and 'PyMOL Tk/Tk GUI'. The system clock shows 3:33 PM.

# Manipulate the view by mouse



Control of clipping planes.

# Important panels

The image displays the PyMOL software interface, which is used for molecular visualization. It consists of several key panels:

- Command Console (Top Left):** Shows the execution of commands and their outputs. The visible text includes:

```
PyMOL>load $:/1hng
ObjectMolecule: e-ERROR: Unable to open file '/1hng'
PyMOL>load $:/1hng
ObjectMolecule: e-ERROR: Unable to open file '/1hng'
PyMOL>load $:/1hng.pdb
ObjectMolecule: Read crystal symmetry information.
Symmetry: found 8 symmetry operators.
Cmd_load: "/1hng.pdb" loaded as "1hng".
PyMOL>delete
Parsing-Error: missing required argument: name
```
- PyMOL Viewer (Main Window):** Displays a 3D molecular model of the protein structure 1hng, rendered in a stick representation with green and blue atoms.
- Selection List (Top Right):** A list of selected objects, currently showing '<all>' and '1hng'. Each entry has associated action buttons: 'A' (Actions), 'S' (Show/Hide), 'H' (Hide/Show), 'L' (Color), and 'C' (Color).
- Context Menu (Right):** A menu is open over the '1hng' selection, listing the following actions:
  - Actions
  - Show
  - Hide
  - Color
- Mouse Mode Panel (Bottom Right):** Provides information about the current mouse mode and a list of keyboard shortcuts for various actions like rotation, translation, and zooming.
- Taskbar (Bottom):** Shows the Windows taskbar with the Start button and several open applications, including PyMOL Viewer and PyMOL Td/...



# Manipulate your objects

A S H L C

Actions:

- zoom
- center
- origin
- orient

---

- preset
- find
- generate

---

- assign sec. struc.

---

- rename object
- duplicate object
- delete object

---

- add hydrogens
- remove hydrogens
- remove waters

---

- state
- masking
- sequence
- movement
- compute

A S H L C

Show:

- as

---

- lines
- nonbonded
- sticks
- ribbon
- cartoon

---

- labels
- cell

---

- dots
- spheres
- nb\_spheres

---

- mesh
- surface

---

- organic
- main chain
- side chain
- disulfides

A S H L C

Hide:

- everything

---

- lines
- nonbonded
- sticks
- ribbon
- cartoon

---

- labels
- cell

---

- dots
- spheres
- nb\_spheres

---

- mesh
- surface

---

- main chain
- side chain
- waters

---

- hydrogens
- unselected

A S H L C

Color:

- by element
- by chain
- by ss
- spectrum

---

- reds
- greens
- blues
- yellows
- magentas
- cyans
- oranges
- tints
- grays

# Right click your mouse: more options

The image shows a screenshot of the PyMOL software interface. At the top, a window titled "PyMOL Tcl/Tk GUI" contains a command history and a toolbar with buttons like "Reset", "Zoom", "Ray", "Rock", "Urpick", "Deselect", "Get View", and "Clear". Below this is the "PyMOL Viewer" window, which displays a 3D molecular model of a protein structure in green stick representation. A right-click context menu is open over the structure, listing various actions such as "Main Fon-Lp", "zoom (vis)", "center (vis)", "orient (vis)", "reset", "erase", "disable", "show", "hide", "color", "view", "delete all", "reinitialize", "quit", "ray", "delete all", "reinitialize", "quit", "show", "hide", "color", "view", "reset", "zoom", "center", "origin", "orient", "label", "enable", and "disable". The text "Right click: more choices" is overlaid on the left side of the viewer window. On the right side of the viewer, there is a list of objects: "<all>" and "1hng". At the bottom right, a "Mouse Mode" section lists keyboard shortcuts for various actions like "Buttons", "Keys", "Rotate", "Move", "Clip", "Menu", "Selecting Residues", and "Name". The Windows taskbar at the bottom shows the "start" button and several open applications including "PyMOL Viewer", "PyMOL Tcl/Tk GUI", "Microsoft Power...", "PyMOL Users M...", "MSN.mn - MyIE2", and "FN". The system clock shows "4:17 PM".

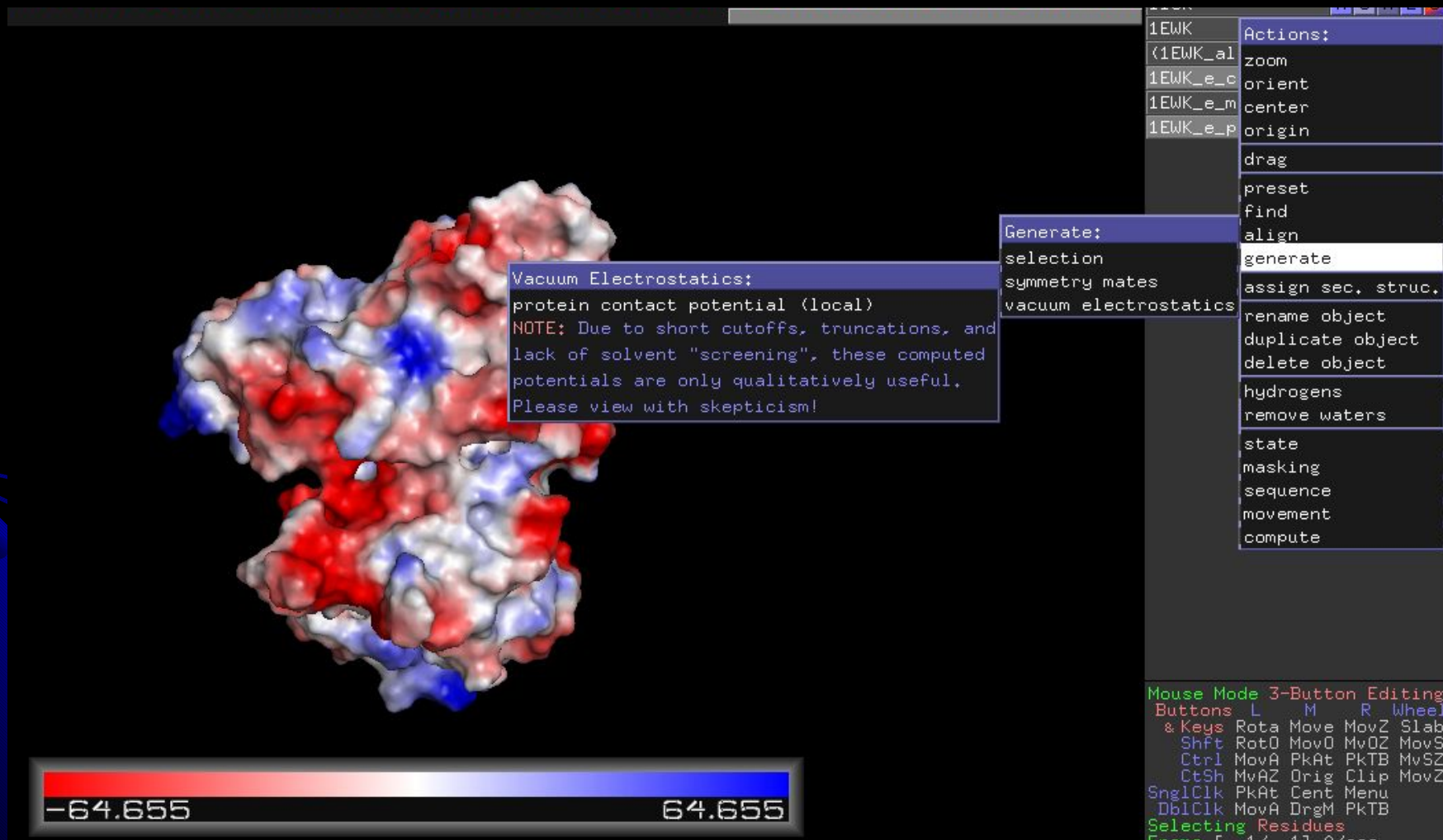
# Actions: show B-factor putty



1EWK	Actions:
(1EWK_al	zoom
1EWK_e_c	orient
1EWK_e_m	center
1EWK_e_p	origin
Preset:	drag
simple	preset
simple (no solvent)	find
ball and stick	align
b factor putty	generate
technical	assign sec. struc.
ligands	rename object
ligand sites	duplicate object
pretty	delete object
pretty (with solvent)	hydrogens
publication	remove waters
publication (with solvent)	state
default	masking
	sequence
	movement
	compute

```
Mouse Mode 3-Button Editing
Buttons L M R Wheel
& Keys Rota Move MovZ Slab
Shft Rot0 Mov0 Mv0Z MovS
Ctrl MovA PkAt PKTB MvSZ
CtSh MvAZ Drig Clip MovZ
SneIClk PkAt Cent Menu
```

# Actions: generating electrostatics map



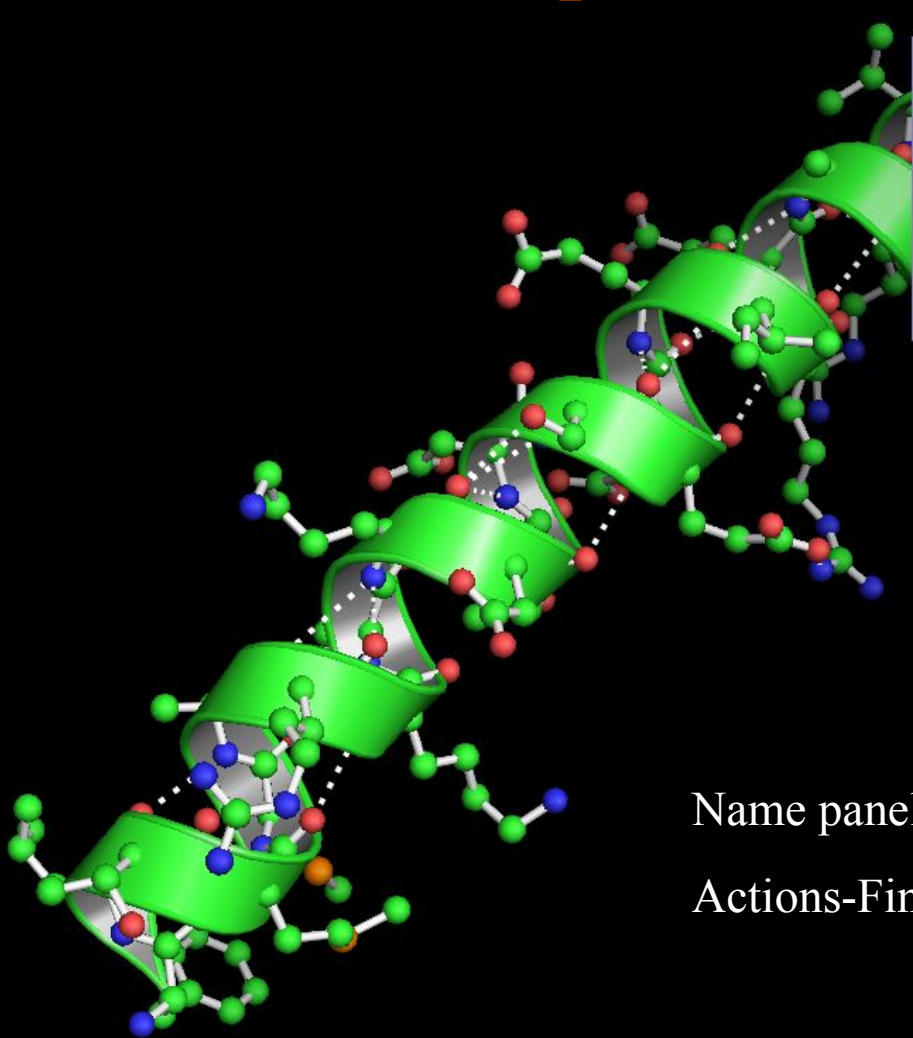
Vacuum Electrostatics:  
protein contact potential (local)  
NOTE: Due to short cutoffs, truncations, and lack of solvent "screening", these computed potentials are only qualitatively useful. Please view with skepticism!

1EWK Actions:  
<1EWK\_al zoom  
1EWK\_e\_c orient  
1EWK\_e\_m center  
1EWK\_e\_p origin  
drag  
preset  
find  
align  
Generate:  
selection generate  
symmetry mates assign sec. struc.  
vacuum electrostatics rename object  
duplicate object  
delete object  
hydrogens  
remove waters  
state  
masking  
sequence  
movement  
compute

Mouse Mode 3-Button Editing  
Buttons L M R Wheel  
& Keys Rota Move MovZ Slab  
Shft Rot0 Mov0 MvOZ MovS  
Ctrl MovA PkAt PkTB MvSZ  
CtSh MvAZ Drig Clip MovZ  
SnglClk PkAt Cent Menu  
DblClk MovA DrgM PKTB  
Selecting Residues  
From 1 1 1 0/100

-64.655 64.655

# Actions: Find polar contacts



Polar Contacts:	Find:
within selection	polar contacts
involving side chains	
involving solvent	
excluding solvent	
excluding main chain	
excluding intra-main chain	
just intra-side chain	
to other atoms in object	
to others excluding solvent	
to any atoms	
to any excluding solvent	

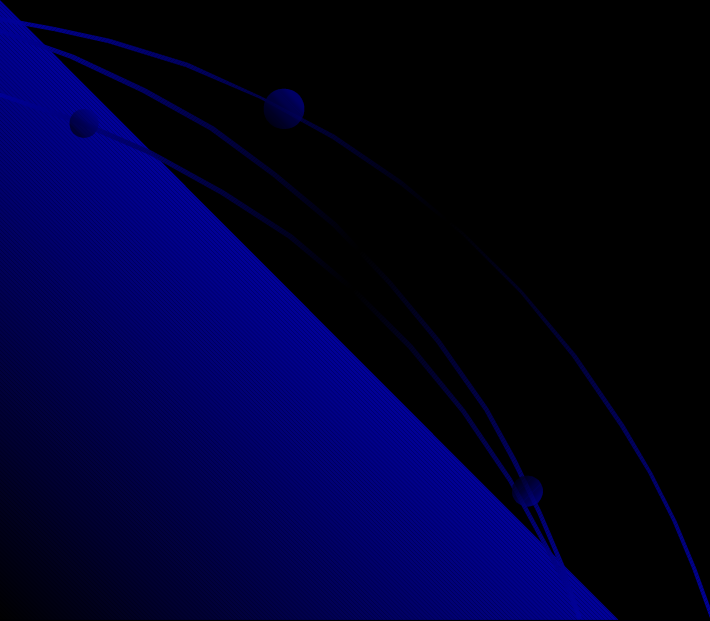
(sel01)	Actions:
sel01_p	delete selection
(sel02)	rename selection
	zoom
	center
	origin
	orient
	preset
	find
	remove atoms
	around
	expand
	extend
	invert
	complete
	duplicate selection
	create object
	masking
	movement
	compute

Name panel: your selected residues

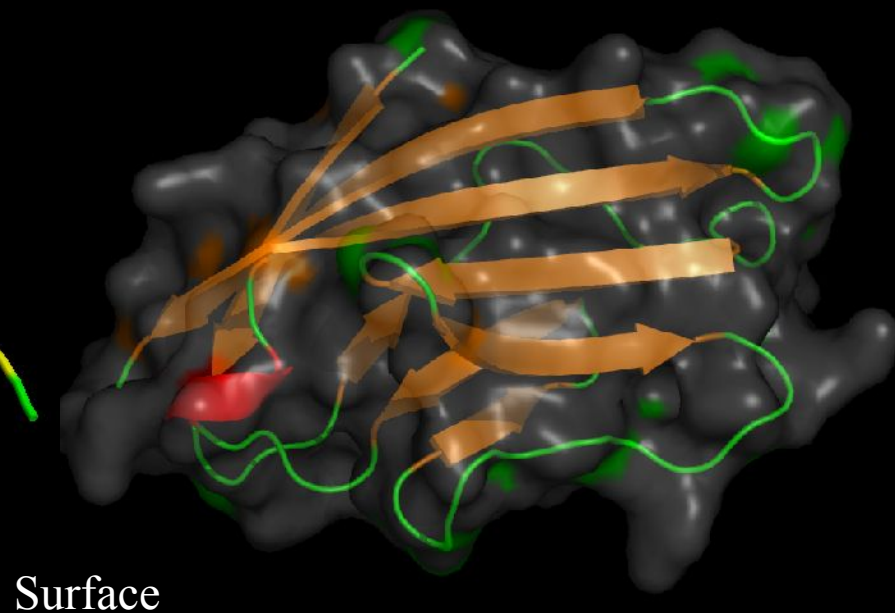
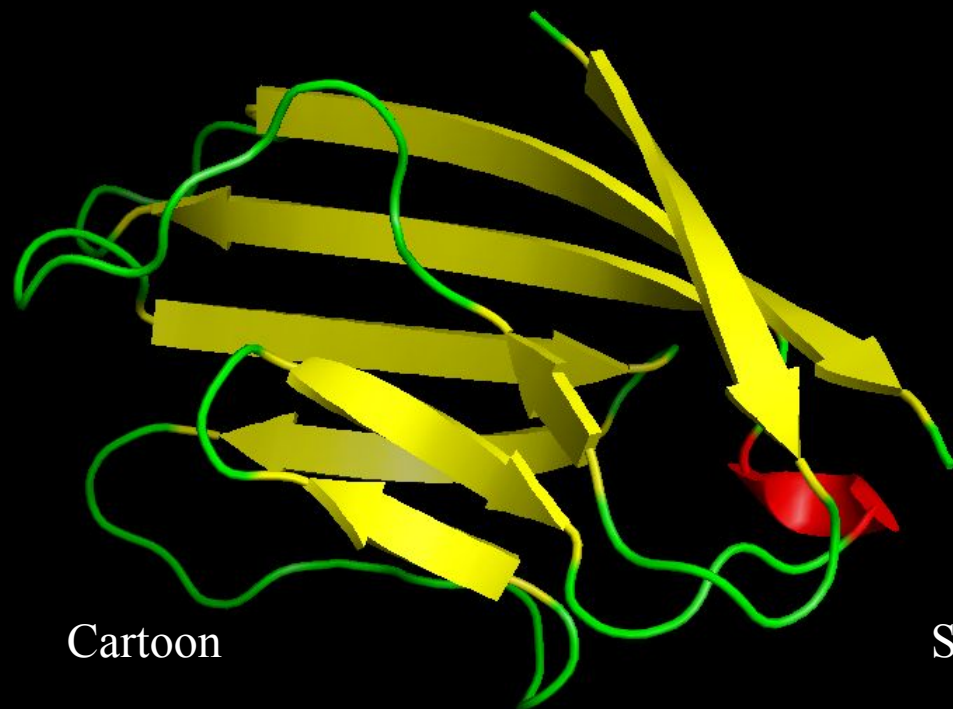
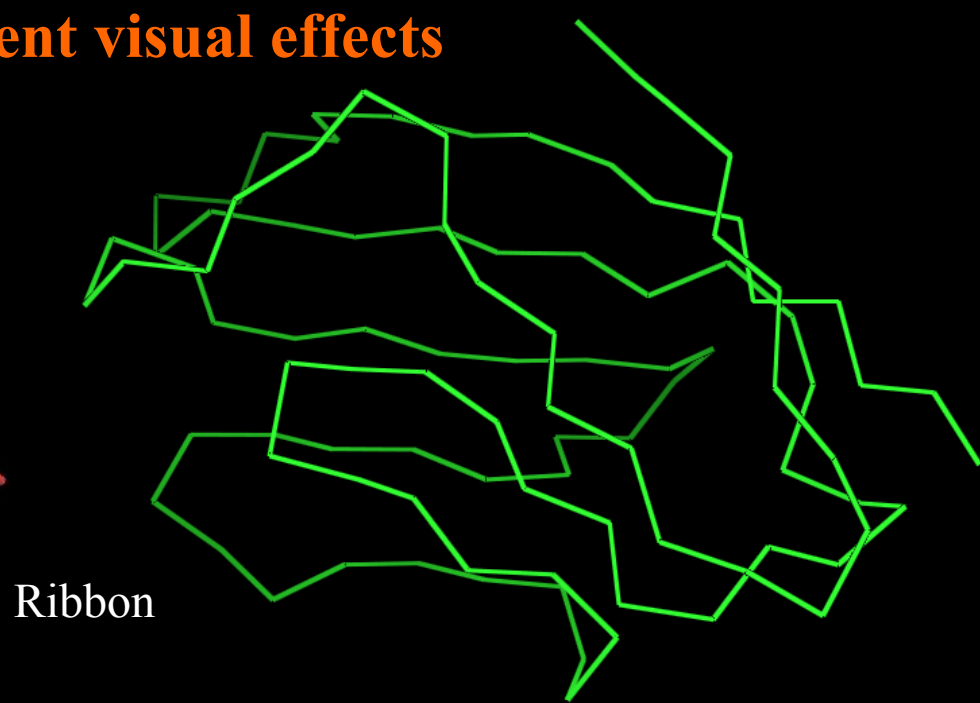
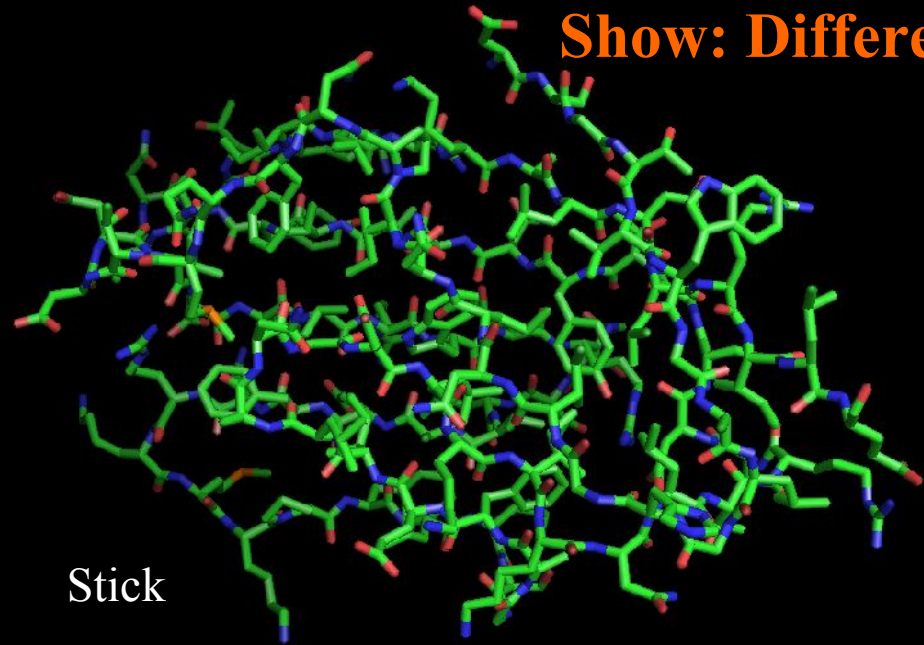
Actions-Find-Polar contacts-...

```
Mouse Mode 3-Button Viewing
Buttons L M R Wheel
& Keys Rota Move MovZ Slab
ShFt +Box -Box Clip MovS
Ctrl +/- PkAt Pk1 MvSZ
CtSh Sele Orig Menu MovZ
SnglClk +/- Cent Menu
DBlClk Menu - PkAt
Selecting Residues
Frame [ 1 / 1 ] 2/sec
```

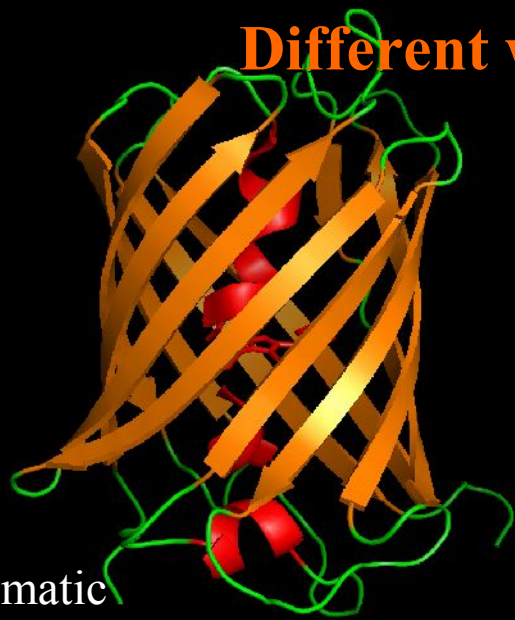
## Actions: Others

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

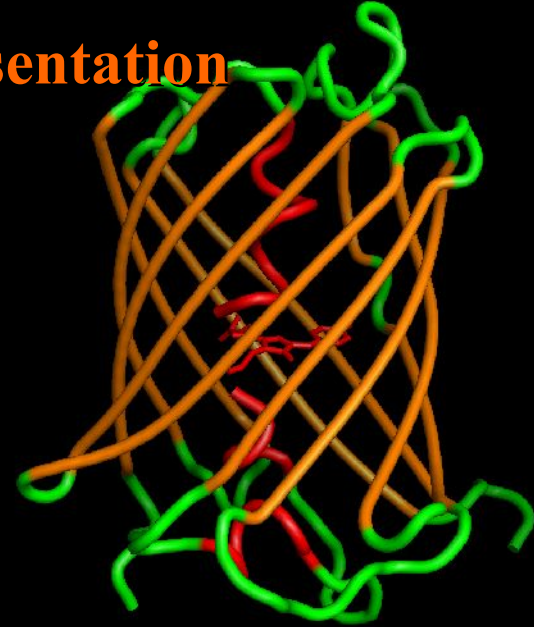
# Show: Different visual effects



# Different way of cartoon presentation



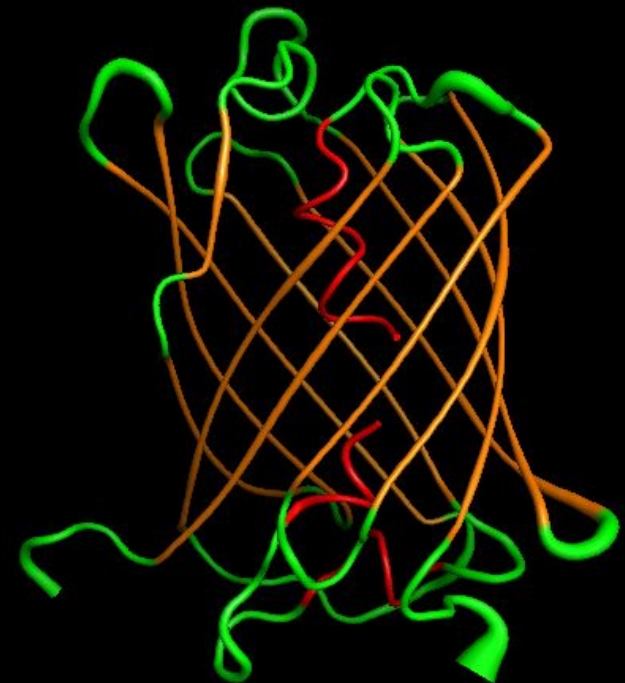
automatic



tube



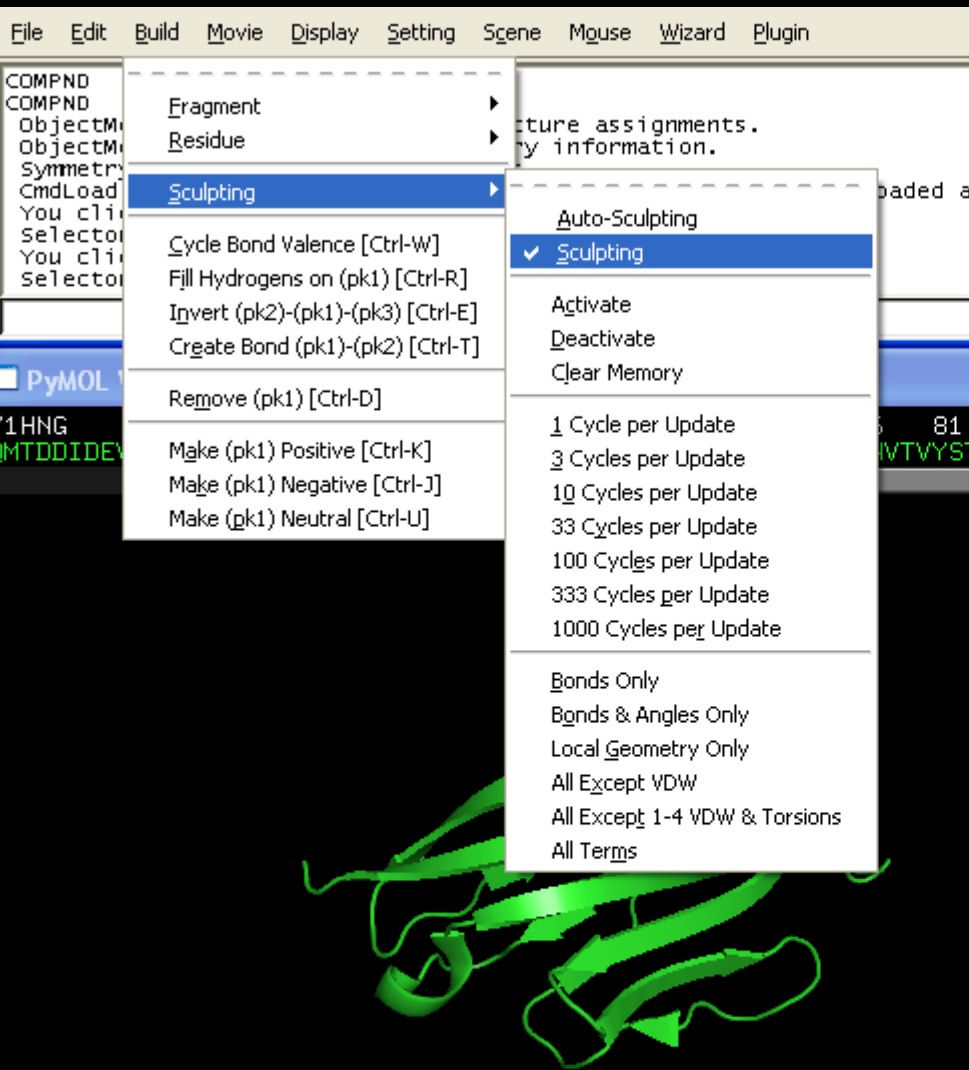
loop



putty



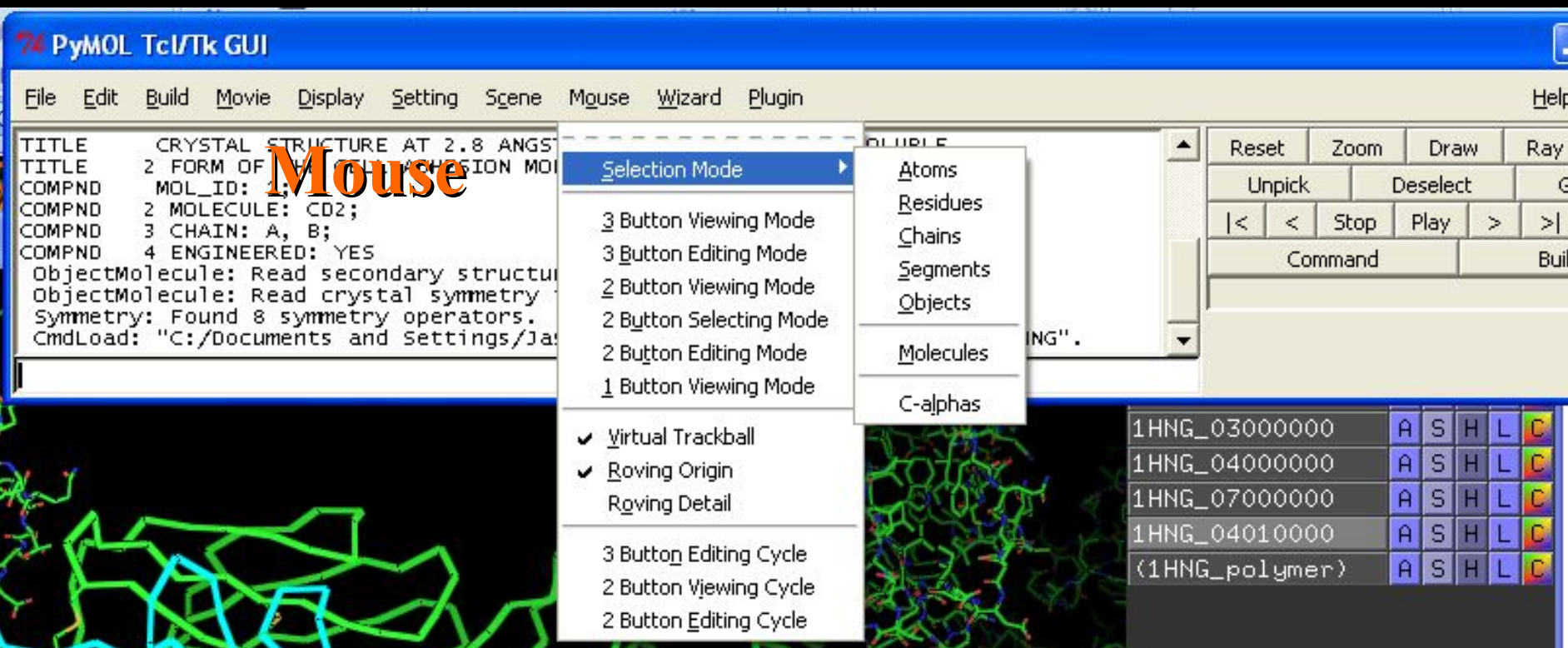
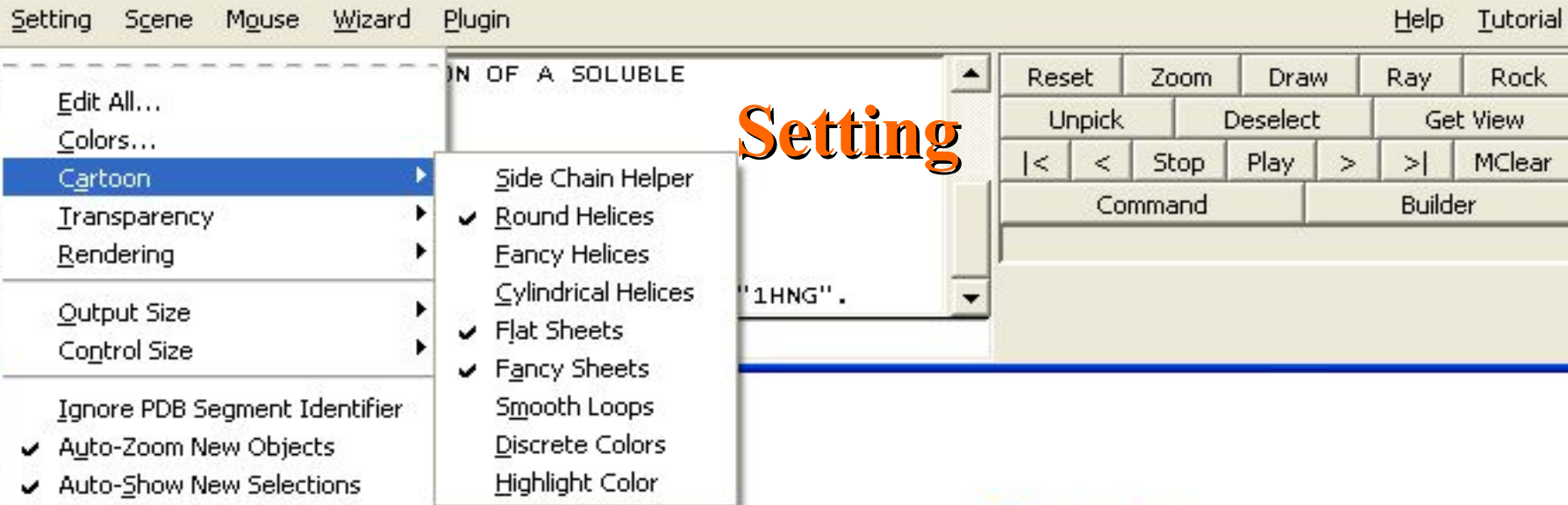
# Build: sculpting



Ctrl+ left click



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



74 PyMOL Tcl/Tk GUI

File Edit Build Movie Display Setting Scene Mouse Wizard Plugin Help

```

Selector: selection "sel13" defined with 0 at
You clicked /3CLN/3CLN//GLU 31/CG
Selector: selection "sel13" defined with 9 at
SceneClick: no atom found nearby.
SceneClick: no atom found nearby.
You clicked /3CLN/3CLN//ASP 20/OB2
Selector: selection "sel14" defined with 1 at
You clicked /3CLN/3CLN//CA 1/CA
SceneClick: no atom found nearby.

```

PyMOL Viewer

```

/3CLN/3CLN//1 6 11 16
CA CA CA CA TEEQIAEFKEAFSLFD
VMRSLGQNPTAEELQDMINEVDADGNGTIDFPEFLTMMARKMKDTSSEEEIREFRFRVFDKDGNGYISAELRHVMTN

```

Please click on the first atom...

# Wizard: Measuring distances

External GUI:  
wizard-measurement

Then click the target atoms

(all)	A	S	H	L	C
3CLN	A	S	H	L	C
(sel01)	A	S	H	L	C
(sel02)	A	S	H	L	C
(sel03)	A	S	H	L	C
(sel04)	A	S	H	L	C
(sel05)	A	S	H	L	C
(sel06)	A	S	H	L	C
(sel07)	A	S	H	L	C
(sel08)	A	S	H	L	C
(sel09)	A	S	H	L	C
(sel10)	A	S	H	L	C
(sel11)	A	S	H	L	C
(sel12)	A	S	H	L	C
(sel13)	A	S	H	L	C
measure01	A	S	H	L	C

Measurement

Distances

Create New Object

Delete Last Object

Delete All Measurements

Done

Mouse Mode 3-Button Viewing  
Buttons L M R Wheel  
& Keys Rota Move MovZ Slab  
Shft +Box -Box Clip MovS  
Ctrl +/- PkAt Pk1 MvSZ  
CtSh Sele Drig Menu MovZ  
SnglClk +/- Cent Menu  
DblClk Menu - PKAt

Selecting Atoms

Frame [ 1 / 1 ] 1/sec

```
TITLE 2 FORM OF THE CELL ADHESION MOLECULE
COMPND MOL_ID: 1;
COMPND 2 MOLECULE: CD2;
COMPND 3 CHAIN: A, B;
COMPND 4 ENGINEERED: YES
ObjectMolecule: Read secondary structure assi
ObjectMolecule: Read crystal symmetry informa
Symmetry: Found 8 symmetry operators.
CmdLoad: "C:/Documents and Settings/Jason/Des
PyMOL>center
```

loaded as "1HNG".

- Appearance
- Measurement**
- Mutagenesis
- Pair Fitting

---

- Density
- Filter
- Sculpting

---

- Label
- Charge

---

- Demo ▶

Reset Zoom Draw Ray Rock

Unpick Deselect Get View

|< < Stop Play > >| MClear

Command Builder

Pick a residue...

# Wizard: Making mutations



Substitution

- No Mutation
- > ALA
- > ARG
- > ASN
- > ASP
- > CYS
- > GLN
- > GLU**
- > GLY
- > HIS
- > ILE
- > LEU
- > LYS
- > MET
- > PHE
- > PRO
- > PRO
- > SER
- > THR
- > TRP
- > TYR
- > VAL

Mutagenesis

- Show Lines
- Backbone-Depen
- Apply
- Clear
- Done

Mouse Mode 3-B

- Buttons L
- & Keys Rota M
- Shft +Box -]
- Ctrl +/- P]
- CtSh Sele O]
- SnglClk +/- C]
- DblClk Menu

Selecting Resi

Frame [ 1/ 1

16

17

18

PyMOL>\_



# 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/3c1n.pdb
mset 1 x60
util.mrock 1,60,180
```

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

```
load $c/3c1n.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

