

# **Vision: Visual Programming**

## **“Integration of Reusable Software Components”**

Michel F. Sanner

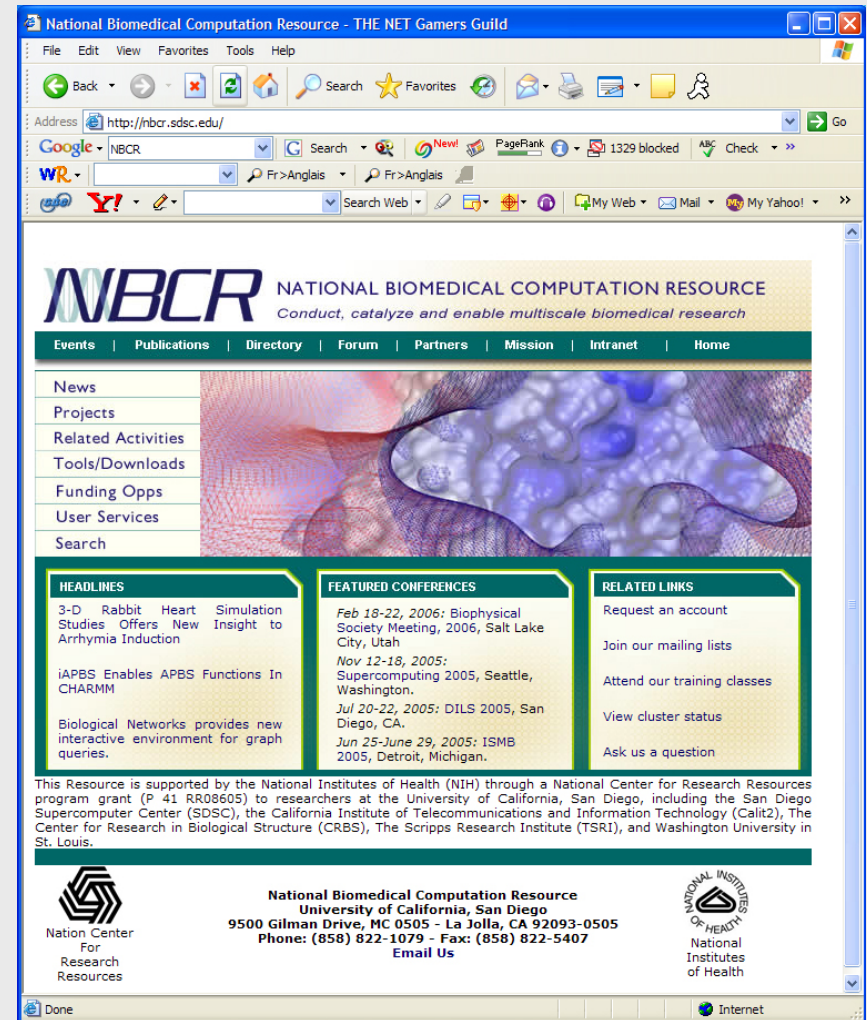
The Molecular Graphics Laboratory

Molecular Biology Department

The Scripps Research Institute  
La Jolla, California

# NBCR: co-sponsor

- **MISSION:** conduct, catalyze, and advance biomedical research by harnessing, developing and deploying forefront computational, information, and grid technologies.
  - (NIH/NCRR P 41RR08605).



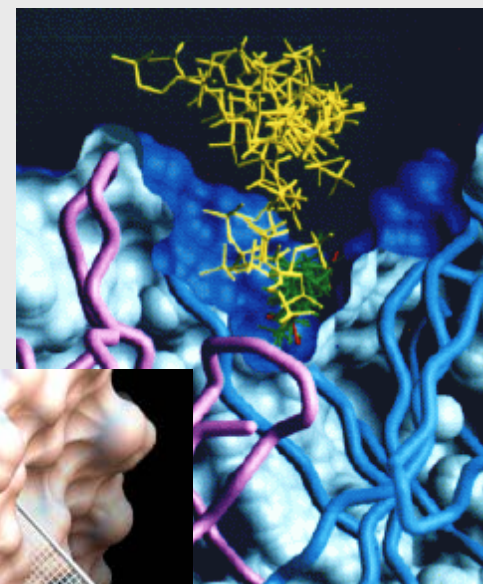
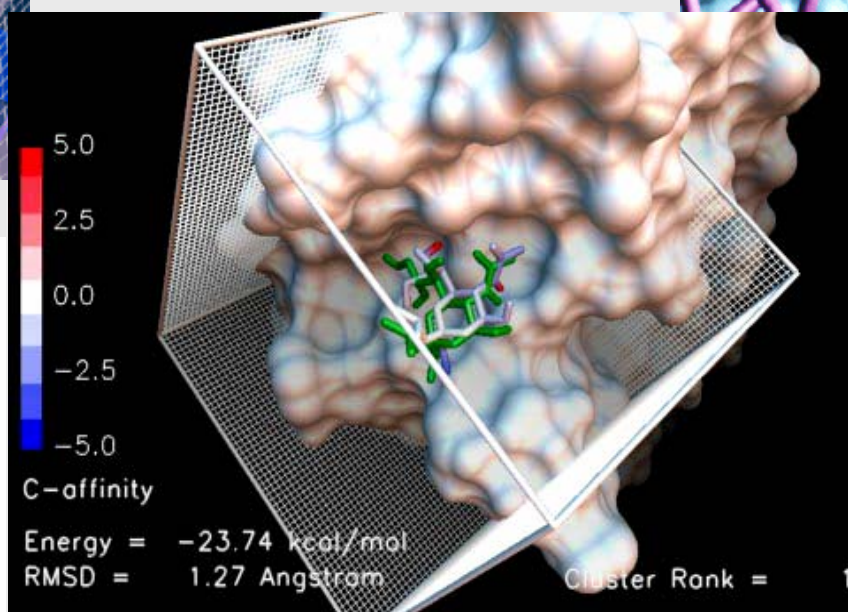
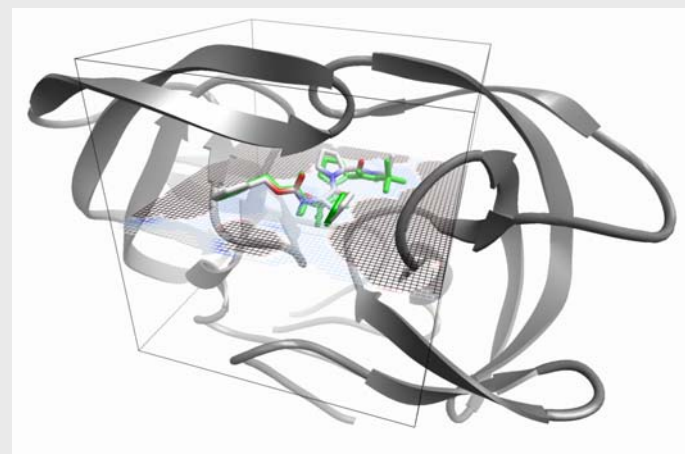
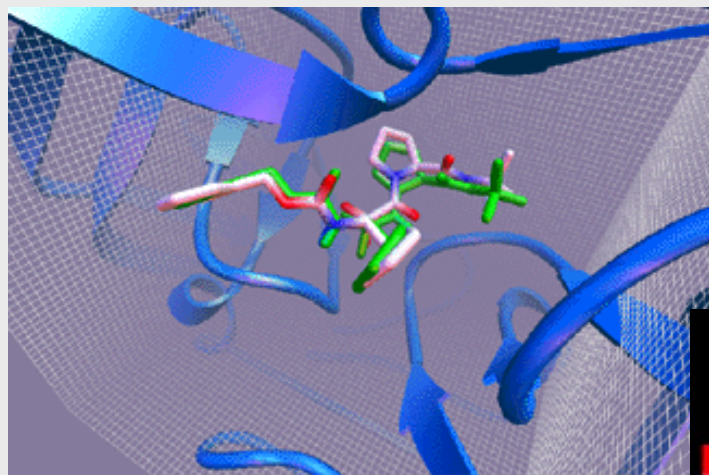
# Outline

- **Background and Motivation:**
  - Scientific interests => software engineering challenge
- **Component-based software development:**
  - Overview of the strategy
  - Software components (MolKit, DejaVu,...)
- **Applications built from software components**
  - PMV: the Python Molecular Viewer, (ADT, PyARTK)
- **Visual Programming:** Visual Software IntegratiON
  - Vision: an application and a software component
- **Demo**

# Scientific Interests

- **Modeling Molecular Interactions**
  - Design of computational models
  - Application to specific biological systems
    - HIV-I protease, Blood coagulation initiation
- **Molecular Visualization**
  - Emphasis on interactivity

# AutoDock: Automated Docking of Flexible Ligands



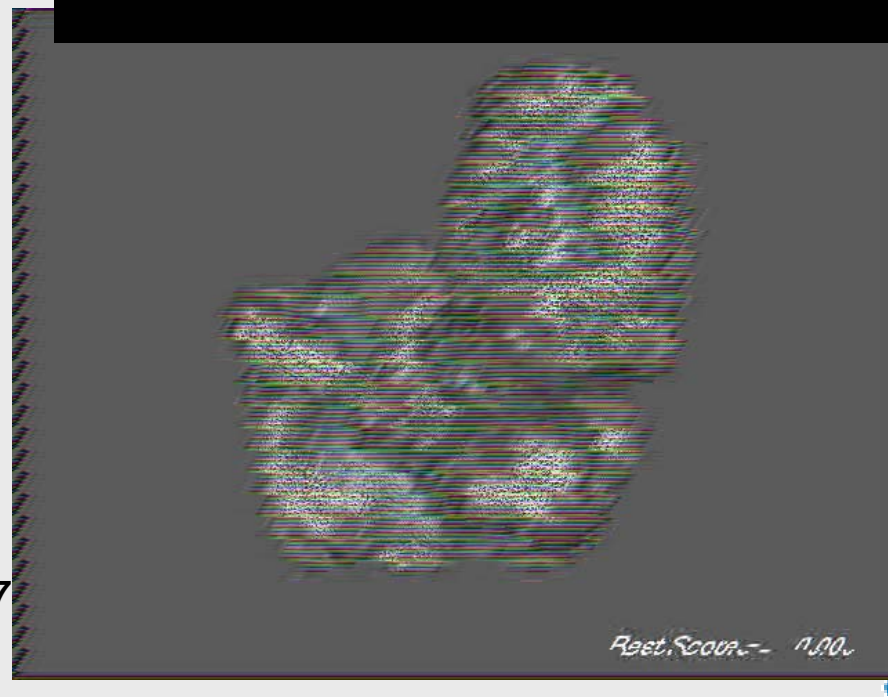
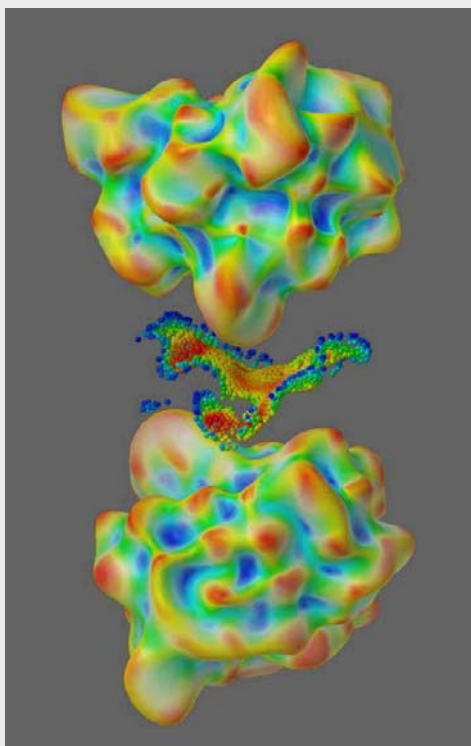
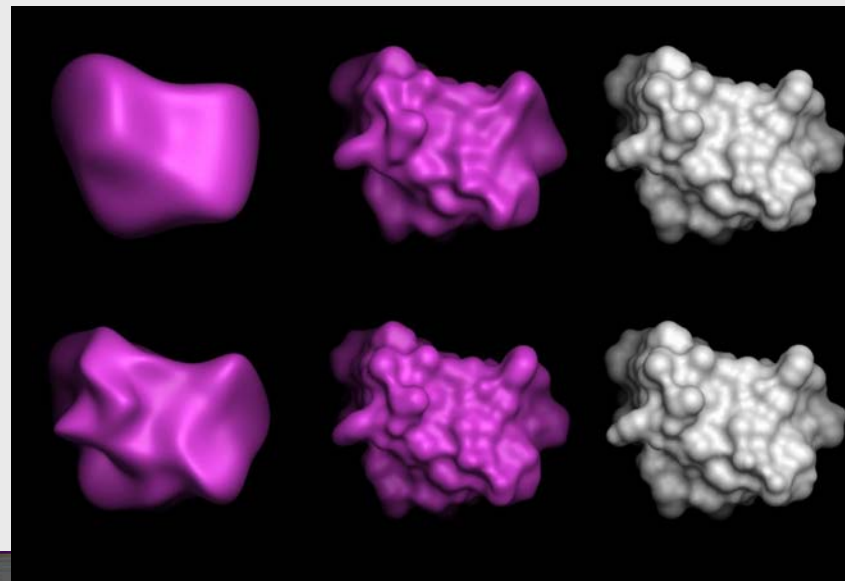
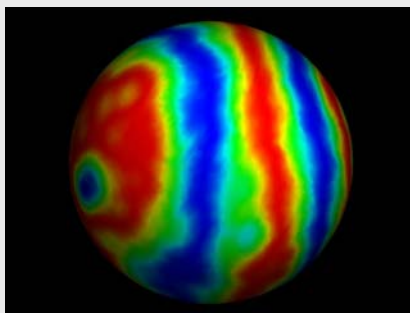
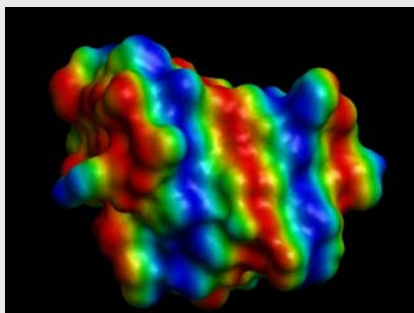
Courtesy G. M. Morris

Morris *et al.* (1998) J. Comp. Chem. 19(14):1639-1662

SciPy'05, Sept. 2005, Caltech, Pasadena, CA

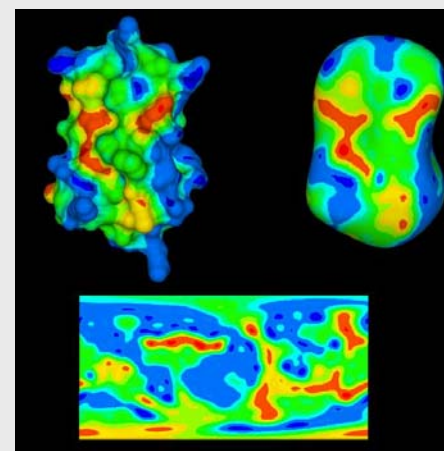
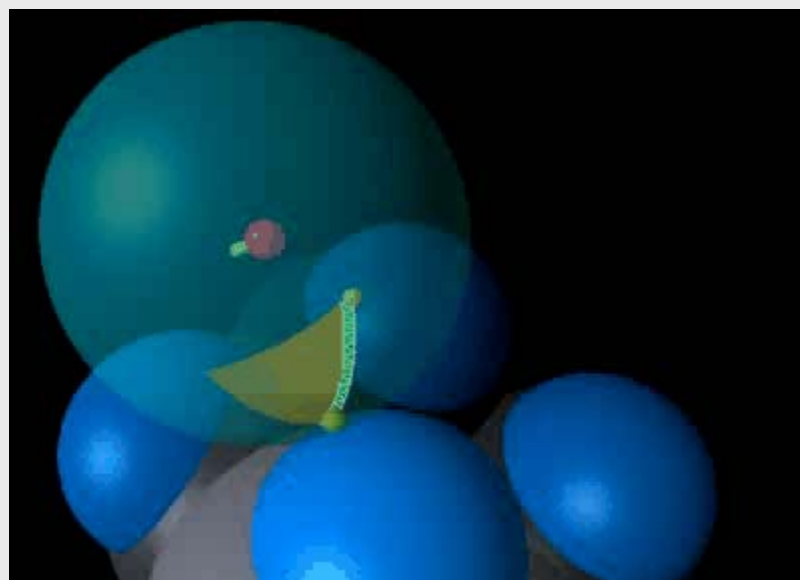
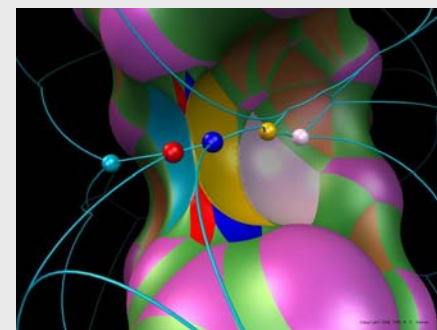
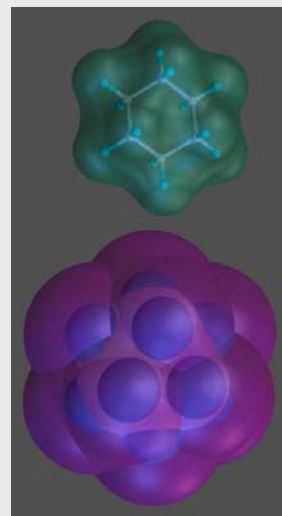
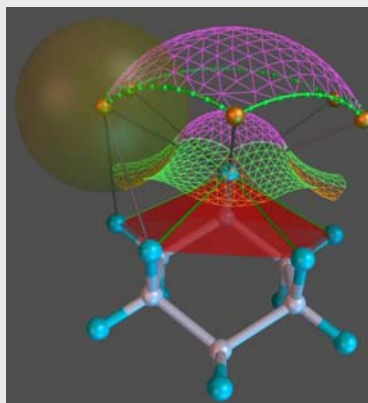
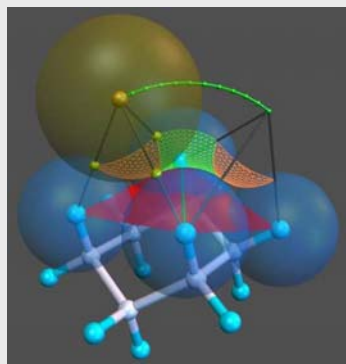
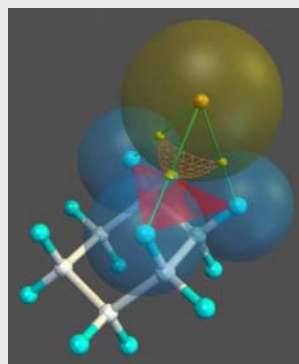


# SurfDock: Protein-Protein Docking



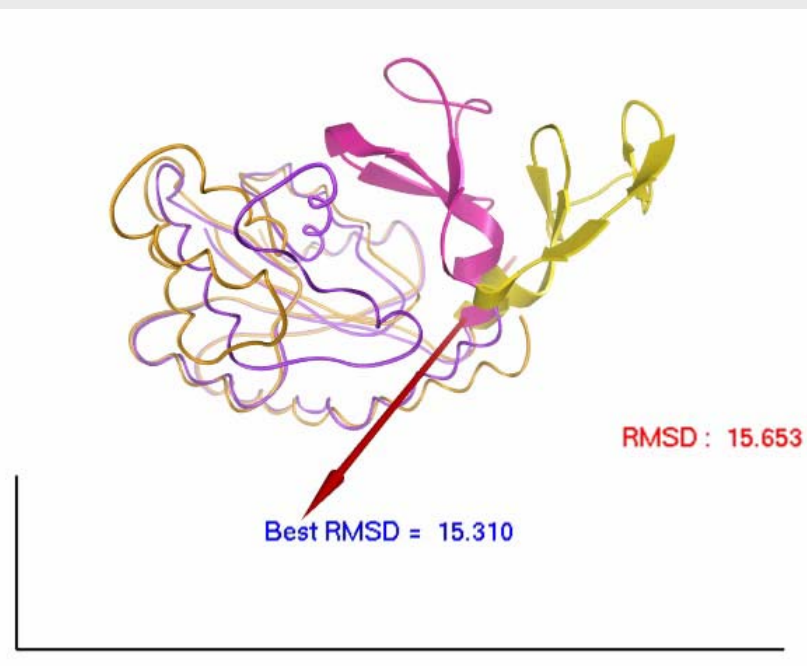
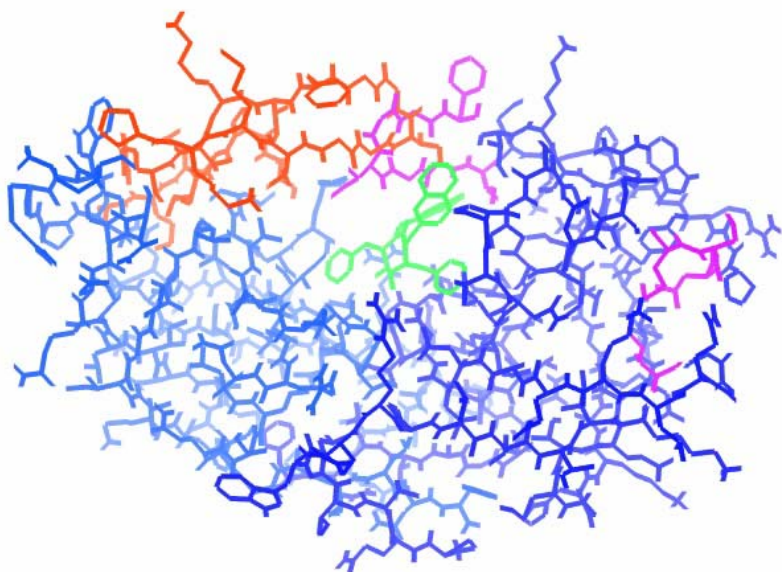
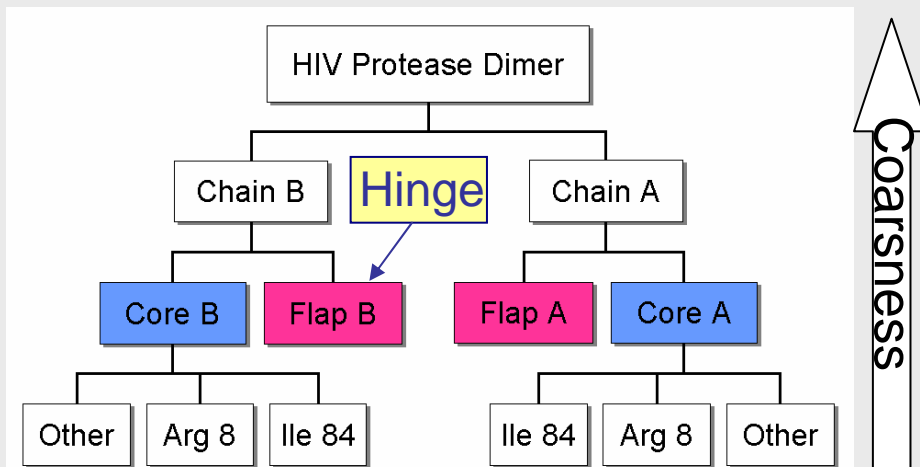
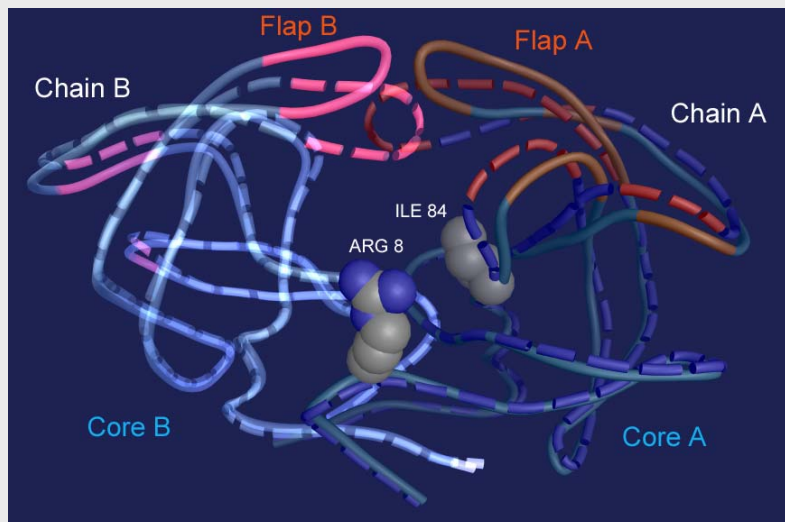
Duncan (1995) J. Mol. Graphics 13:250-257  
SciPy'05, Sept. 2005, Caltech, Pasadena, CA

# MSMS: Molecular Surfaces



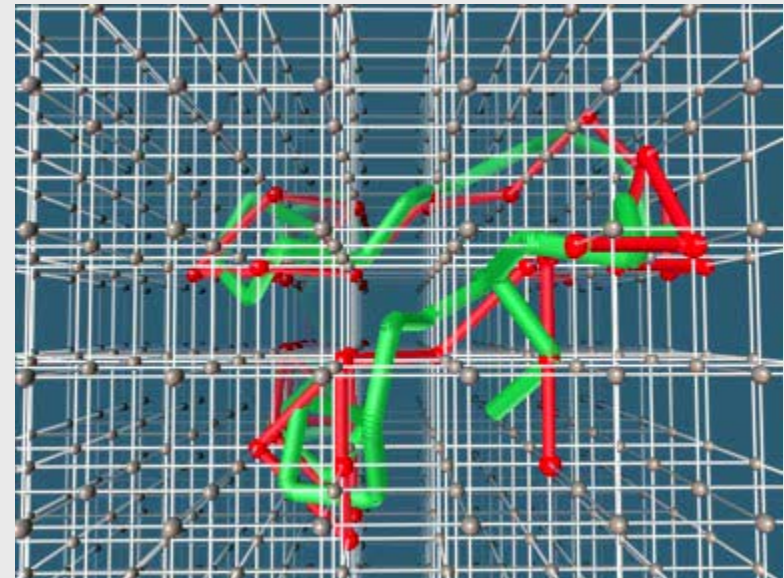
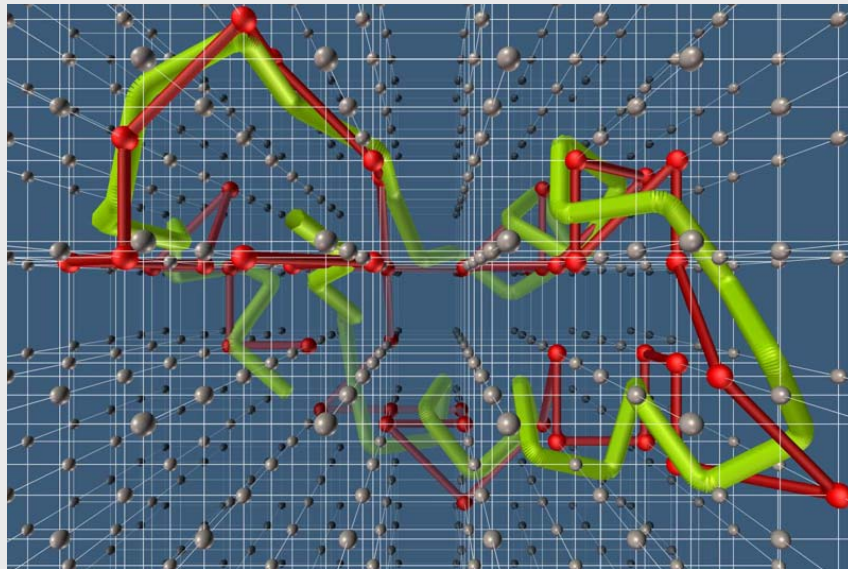
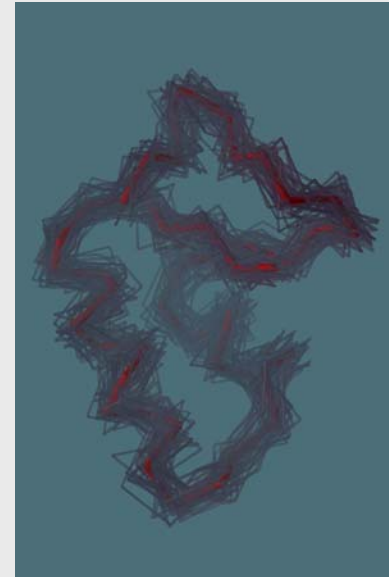
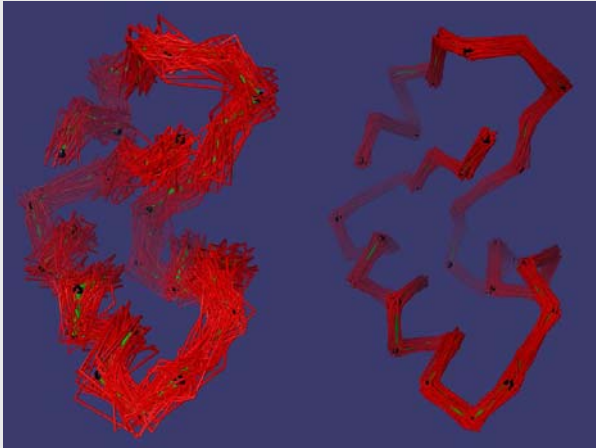
Sanner *et al.* (1995) Biopolymers, 38: 305-320  
SciPy'05, Sept. 2005, Caltech, Pasadena, CA

# Flexibility Tree: Protein Flexibility





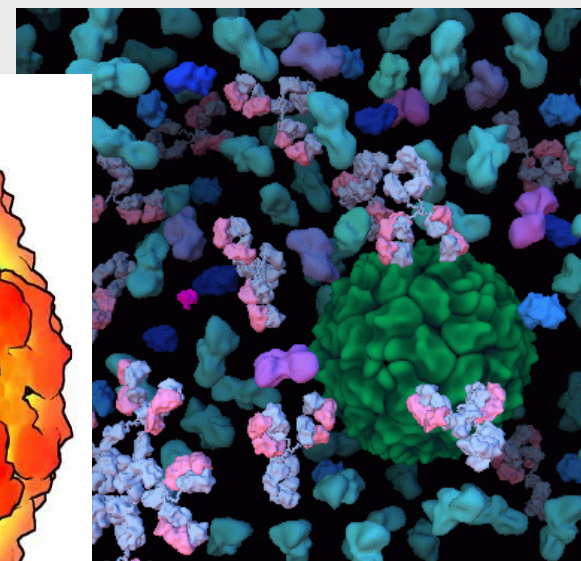
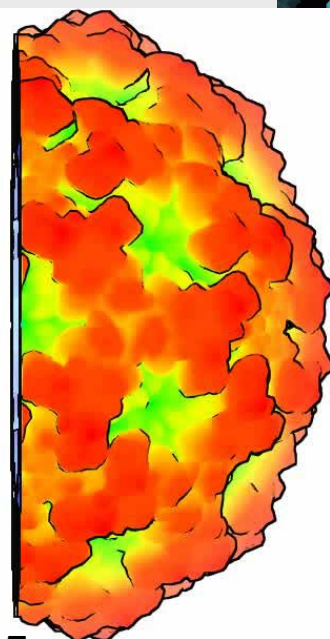
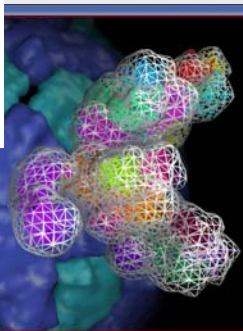
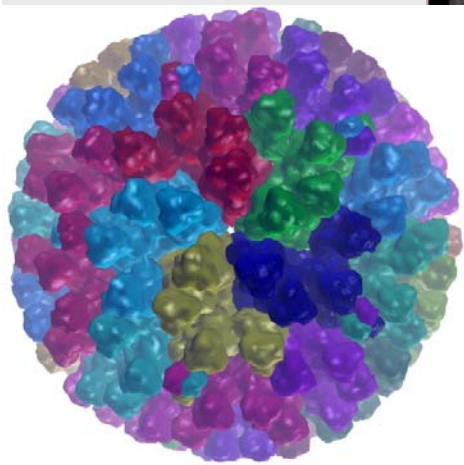
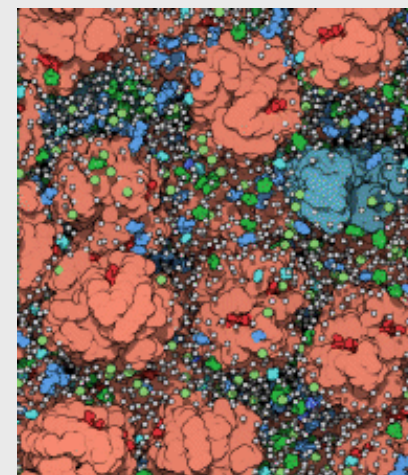
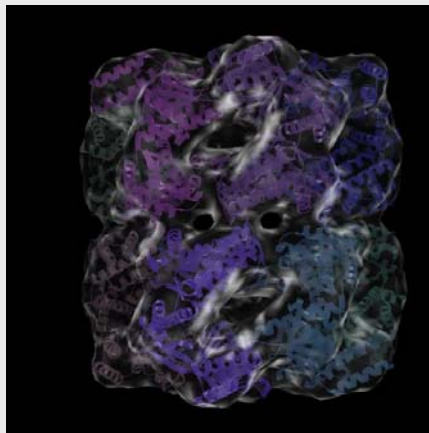
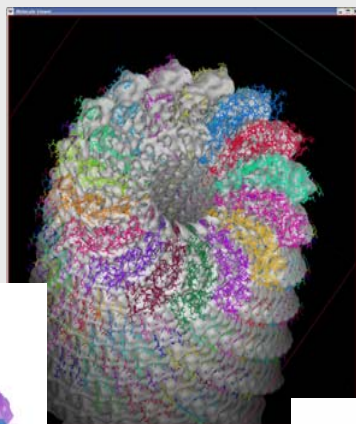
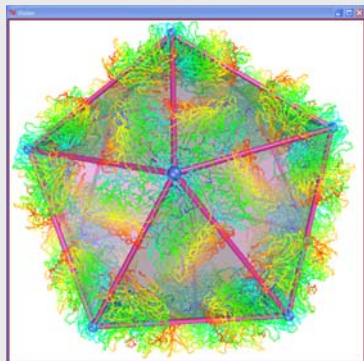
# Protein Folding



Reva, Sanner *et al.* Proteins: Struct., Funct., Genetics, 25:379-388

SciPy'05, Sept. 2005, Caltech, Pasadena, CA

# Complex Assemblies

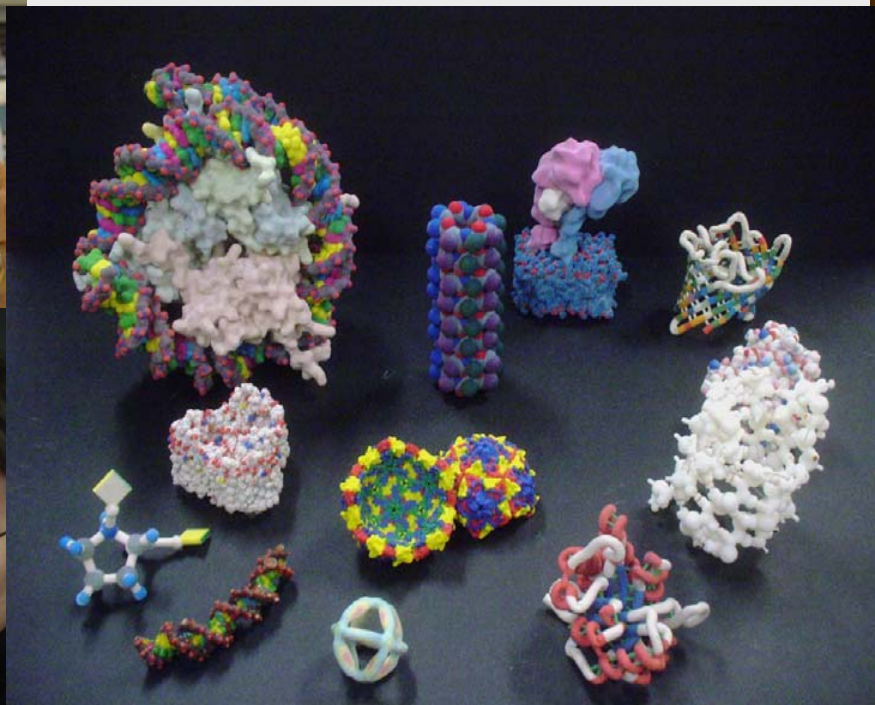
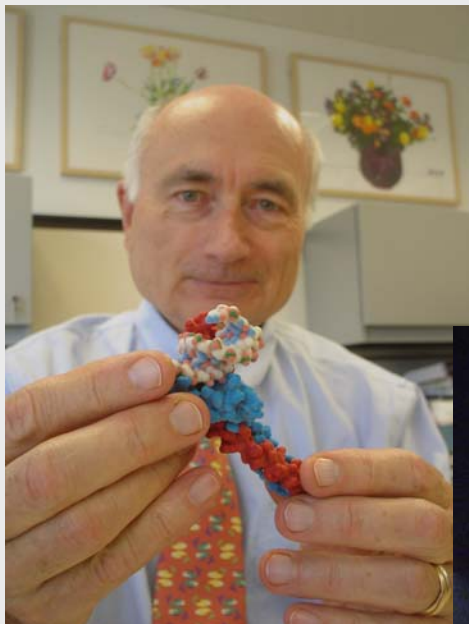


Sanner. Structure, Vol 13, 483-491, March 2005

SciPy'05, Sept. 2005, Caltech, Pasadena, CA



# Tangible Models



Courtesy A. J. Olson

Gillet, Sanner *et al.* Structure, Vol 13, 483-491, March 2005

SciPy'05, Sept. 2005, Caltech, Pasadena, CA

TSRI 

# Tangible models: printing

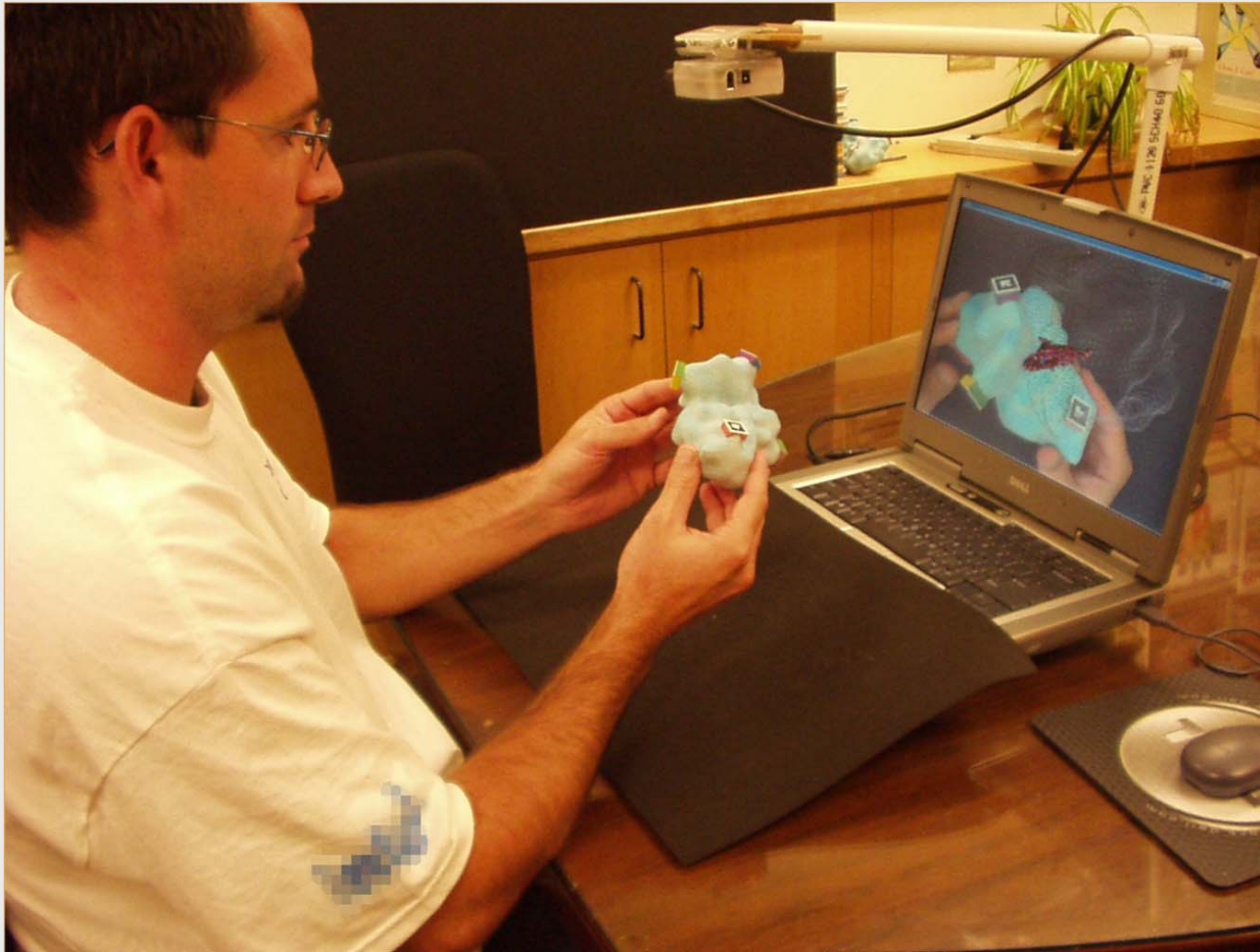


Video by A. Gillet and A. J. Olson

SciPy'05, Sept. 2005, Caltech, Pasadena, CA




# Augmented Reality



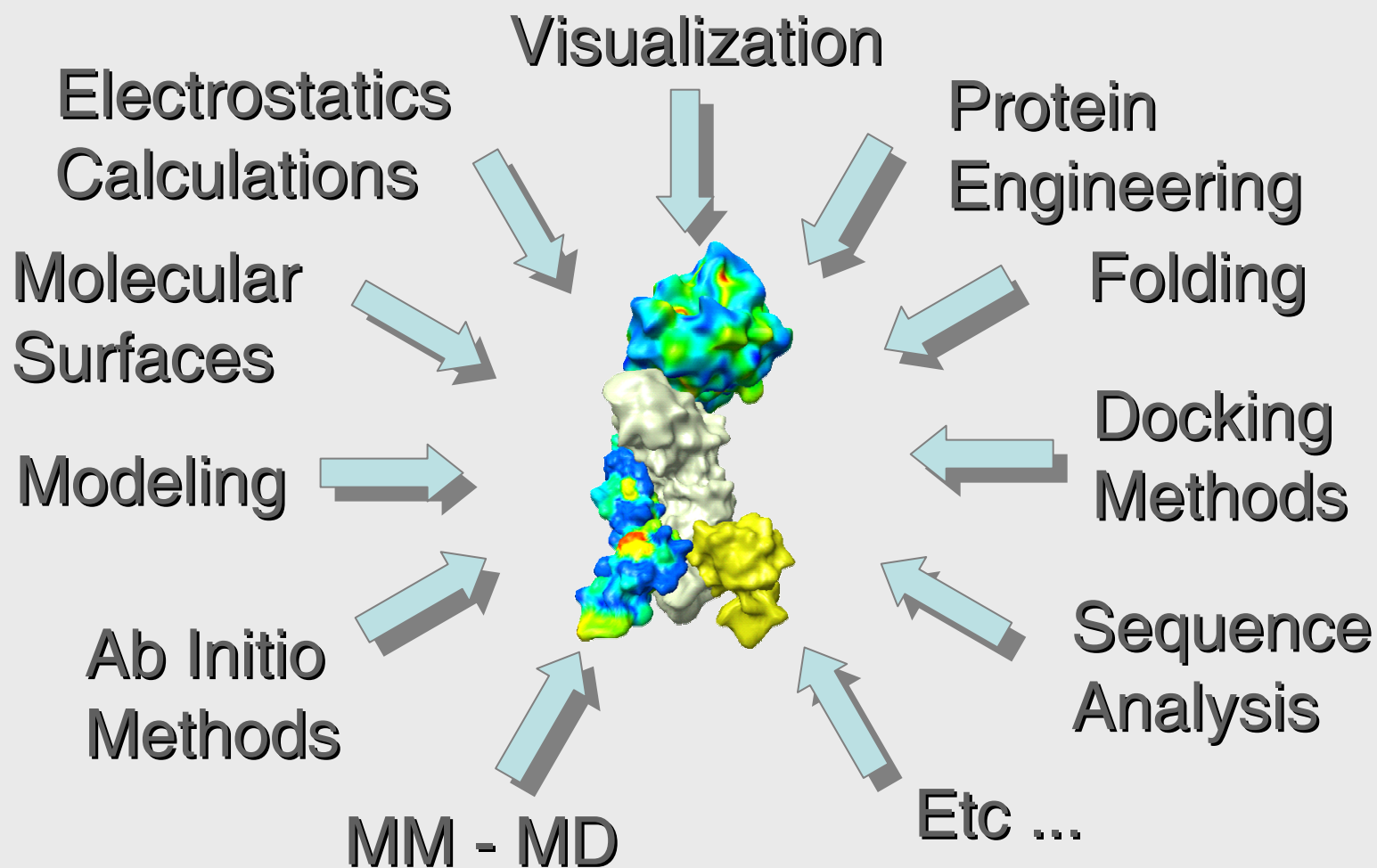
Gillet, Sanner, *et. al.* IEEE Computer Graphics and Applications.  
March-April 2005, Vol 25, Number 2, p13-17

Video by A. Gillet  
and A. J. Olson

SciPy'05, Sept. 2005, Caltech, Pasadena, CA

TSRI 

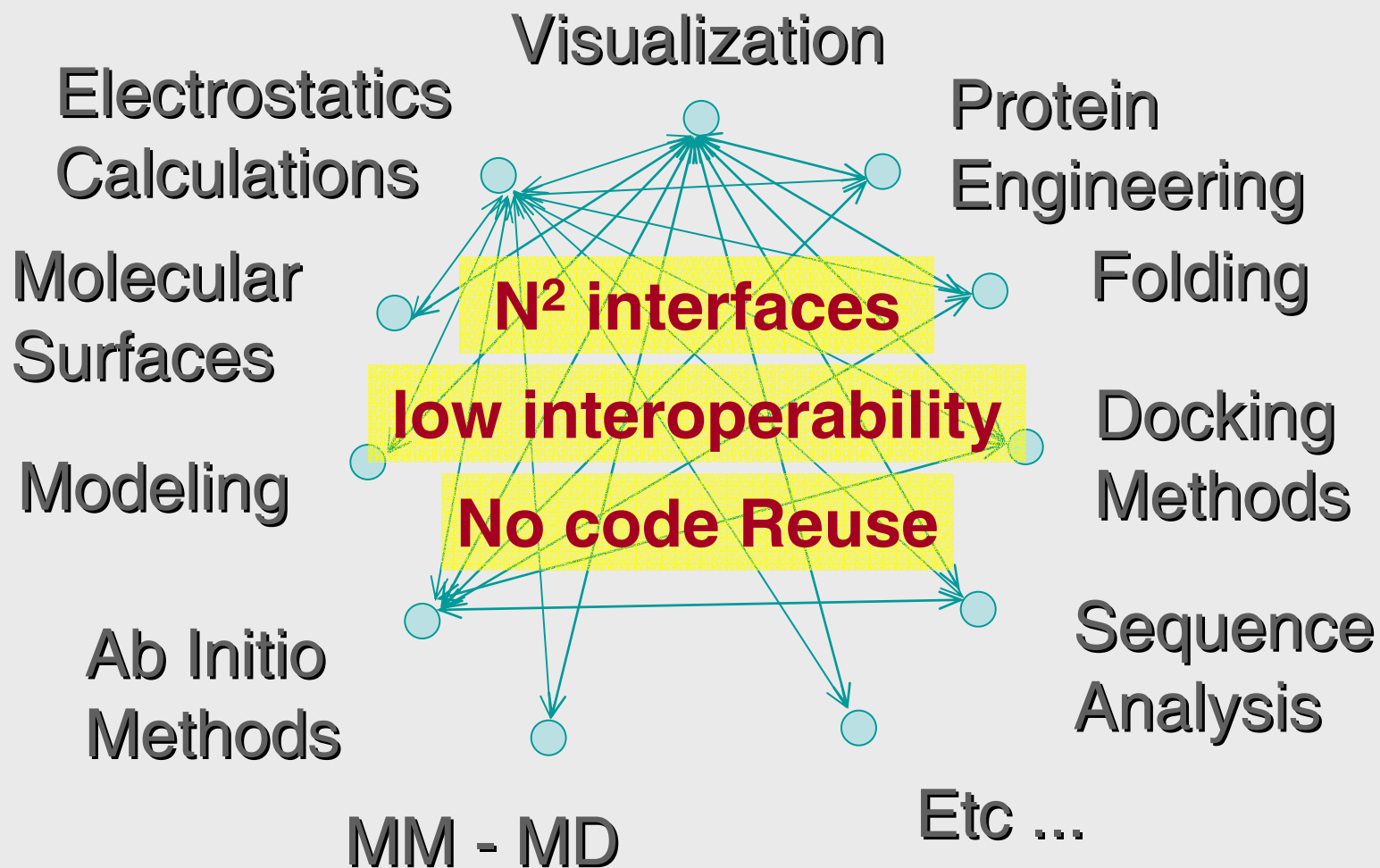
# A Software Engineering Challenge



# Challenges

- **Software components**
  - Interoperable
  - Reusable
  - exchangeable
- **Software applications**
  - Versatile
  - Adaptive
  - Customizable / user-programmable
  - Platform independent

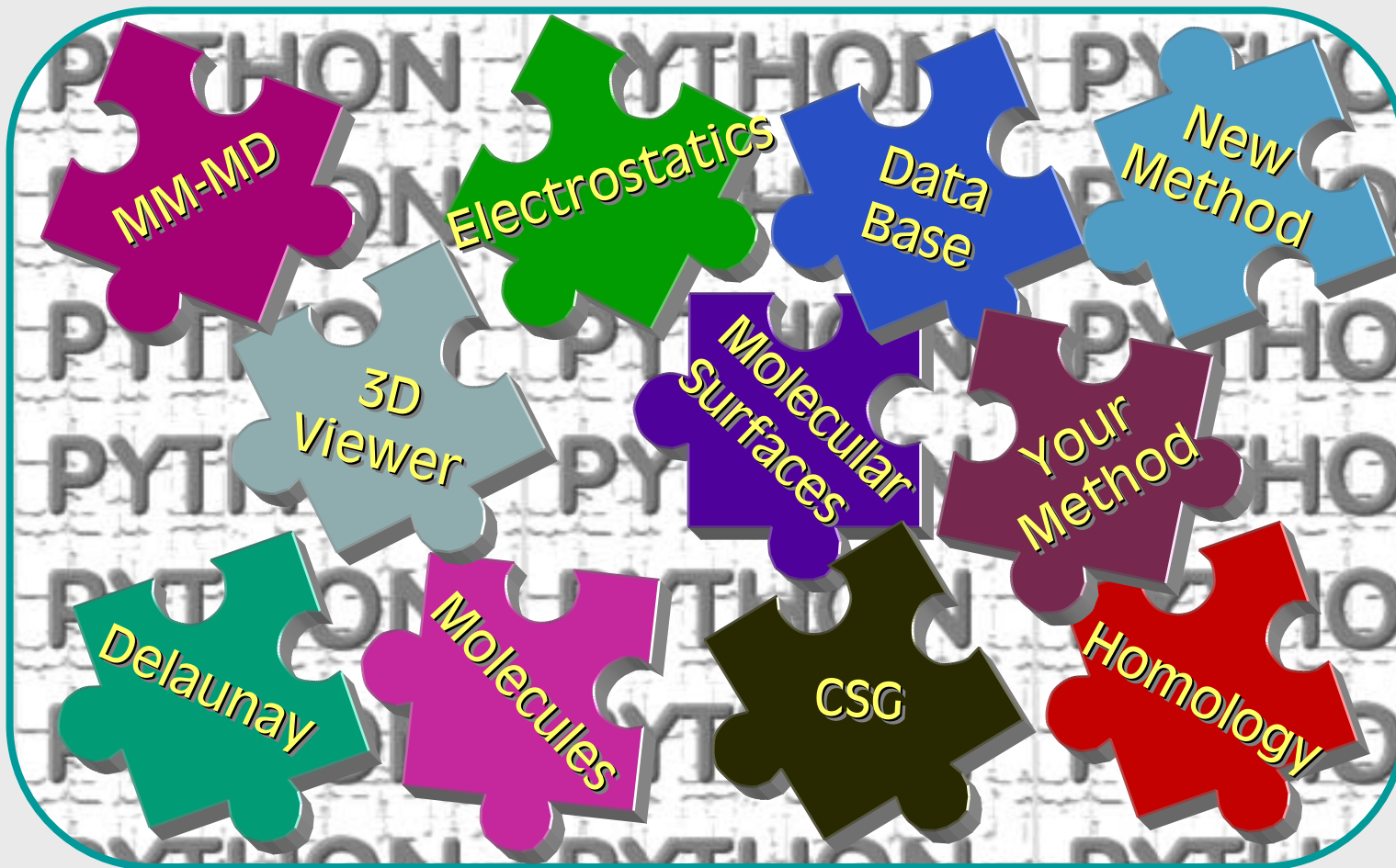
# ad-hoc solution





# Python-centric approach

The Python interpreter is the integration framework



# Take home message(s)

- Write components NOT applications
- Control dependencies between packages
- Pass the simple data types between components
- Native extensions:
  - Avoid C++ templates
  - No global variables in your libraries
  - Do not rely on the program termination to release memory
  - Do not exit if something goes wrong
  - Do not assume A() has been called when B() is called

# Implementation

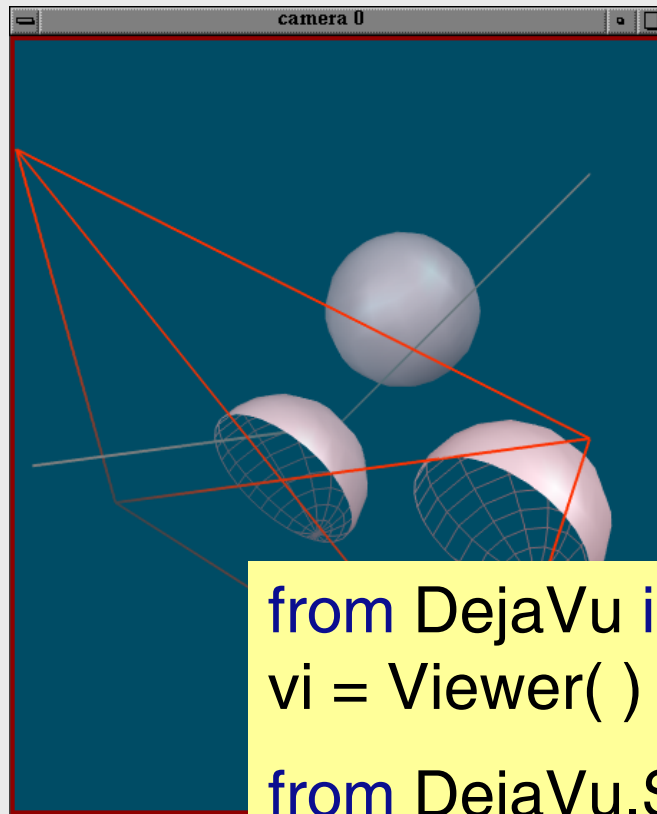
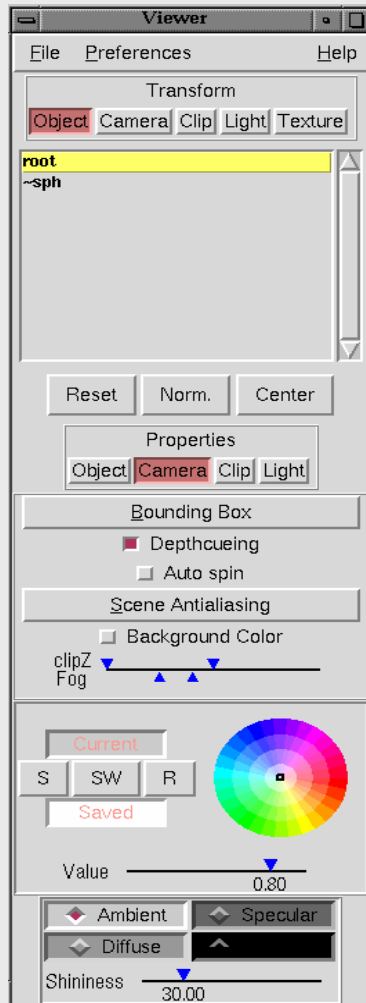
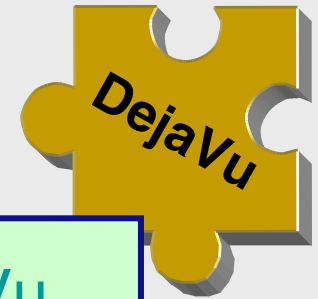
- >10 Python packages
  - > 220,000 lines of code
  - >1370 classes in Python
- >10 packages wrapping C and C++ libraries
  - > 200 classes
- 5 applications
  - Pmv, ADT, Vision, PyARTK, FlexTree
- > 2500 unit tests

# Re-usable components

- MolKit:
  - read/write/represent/manipulate and query molecules
- DejaVu:
  - General purpose 3D-geometry viewer
- ViewerFramework:
  - Visualization application template
- Vision (formerly ViPEr):
  - Visual Software IntegratiON
- Volume, PyQslim, Mslib, PyBabel, PyMead, SFF, UT-Isocontour, ...



# DejaVu



DejaVu

Tkinter

Numeric

openglTk

```
from DejaVu import Viewer  
vi = Viewer( )
```

```
from DejaVu.Spheres import Spheres  
centers = [[0,0,0],[3,0,0],[0,3,0]]  
s = Spheres('sph', centers = centers)  
s.Set(quality=10)  
vi.AddObject(s)
```

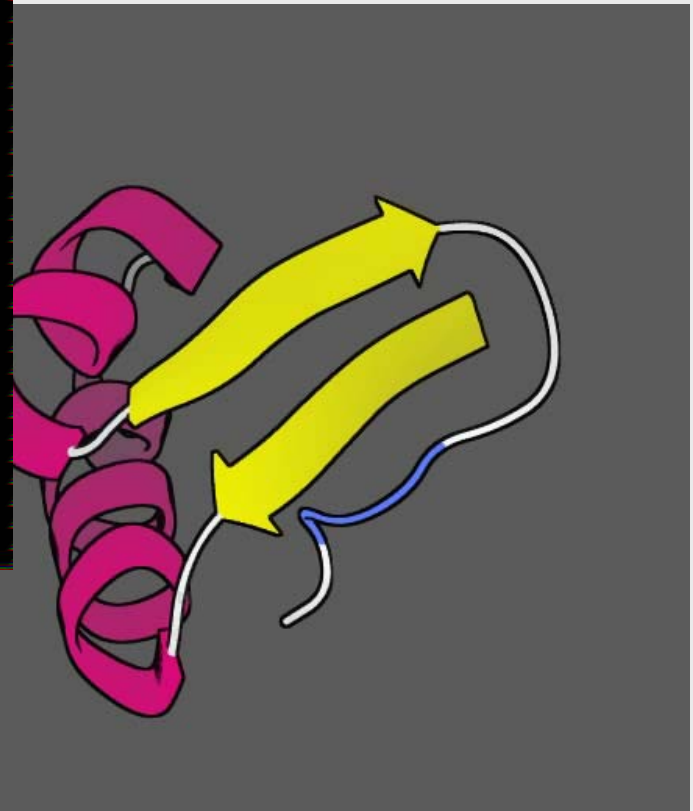
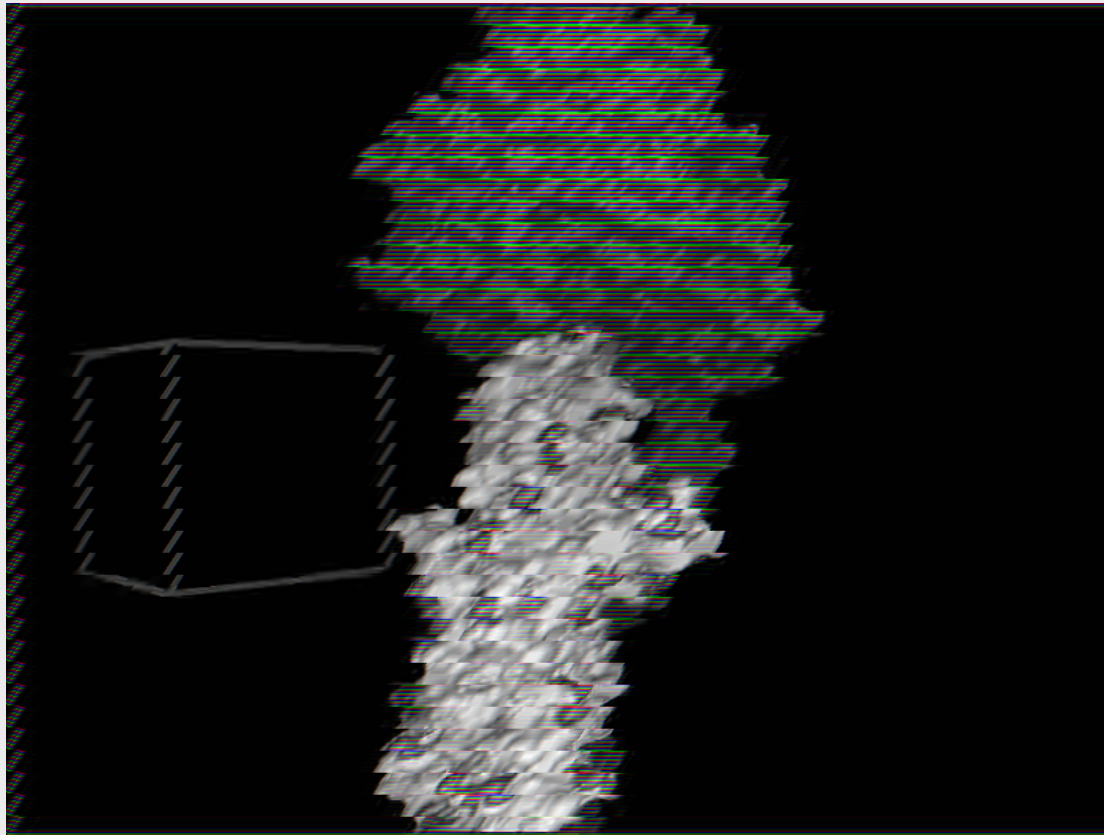


# DejaVu Features



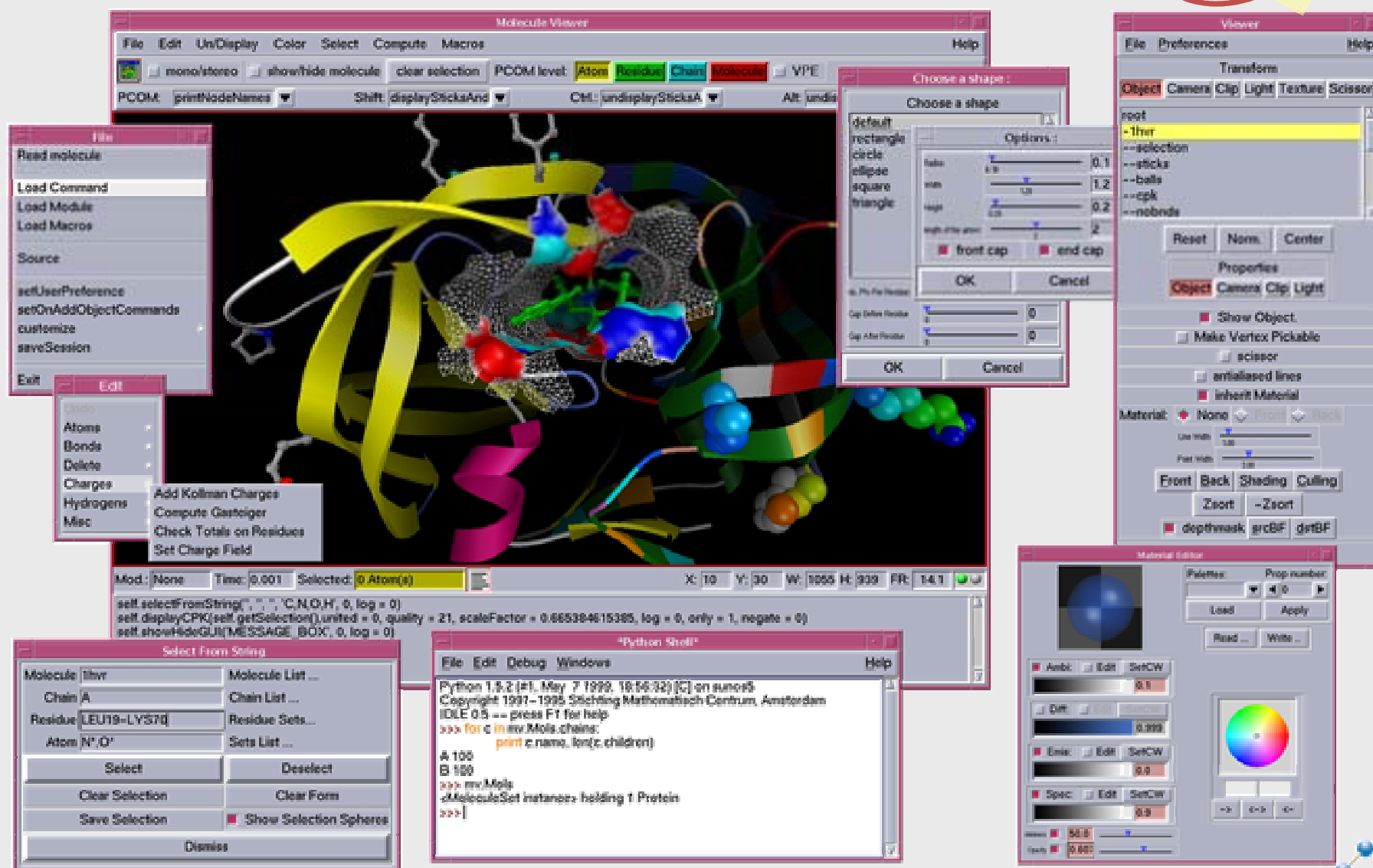
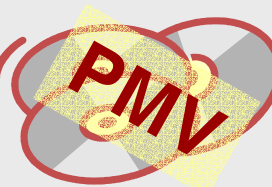
- OpenGL Lighting and Material model
- Material editor, transparency
- Arbitrary clipping planes
- Multiple light sources
- DepthCueing (fog), global anti-aliasing
- glScissors/magic lens
- Object hierarchy, Instances
- Multi-level picking
- Extensible set of geometries
- Direct volume rendering, ...

# New DejaVu Features



# PMV: Python Molecular Viewer

## From Building Blocks to applications



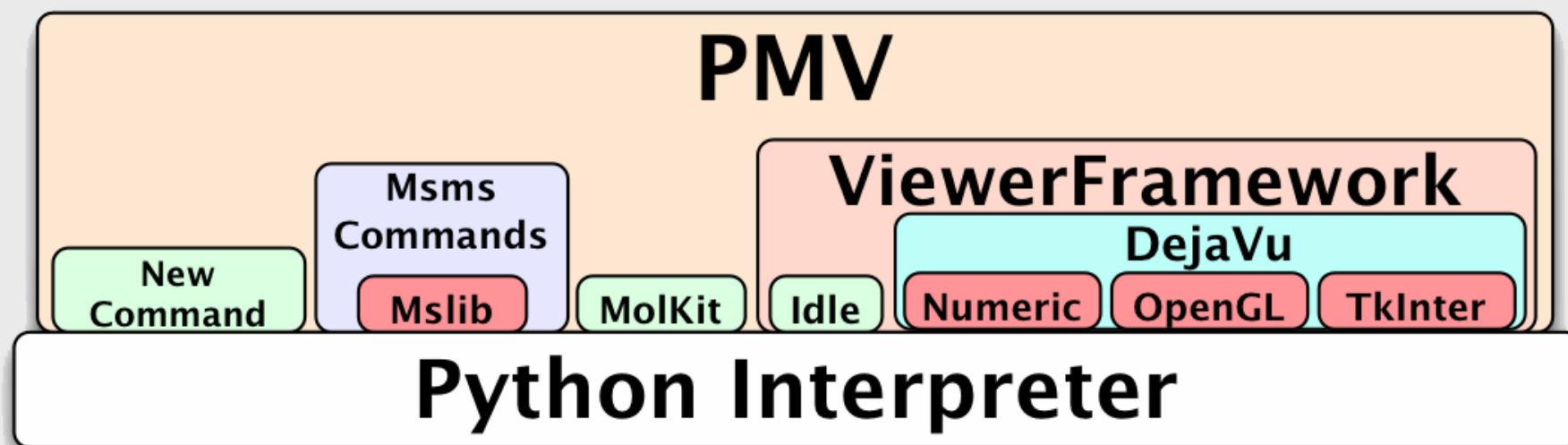
SciPy'05, Sept. 2005, Caltech, Pasadena, CA

TSRI



# PMV: Architecture

Generic Molecule Viewer built from reusable components



## Software engineering

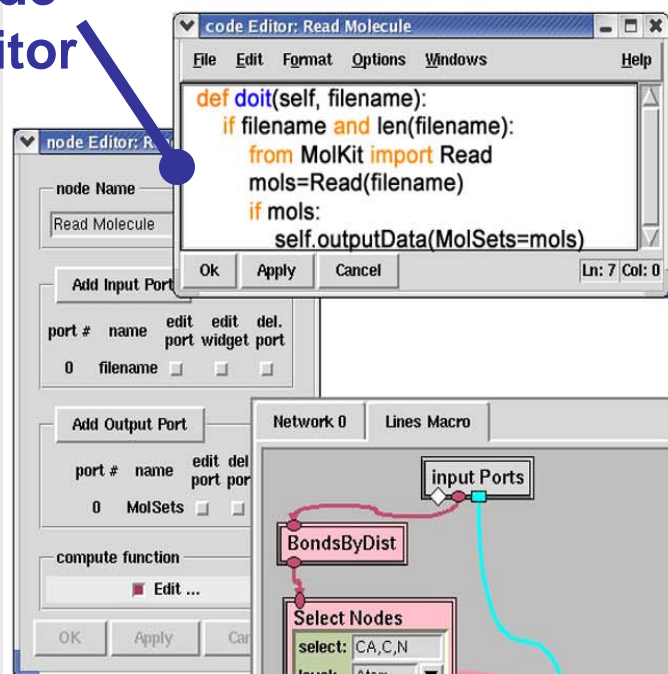
Interoperable, reusable, software component

Versatile, adaptive, user-programmable

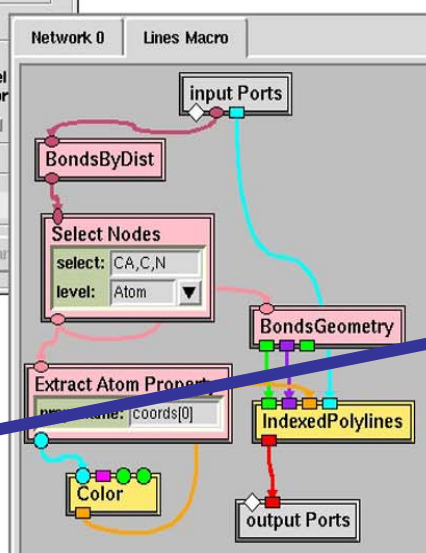
Platform independent

# Vision: Visual Programming

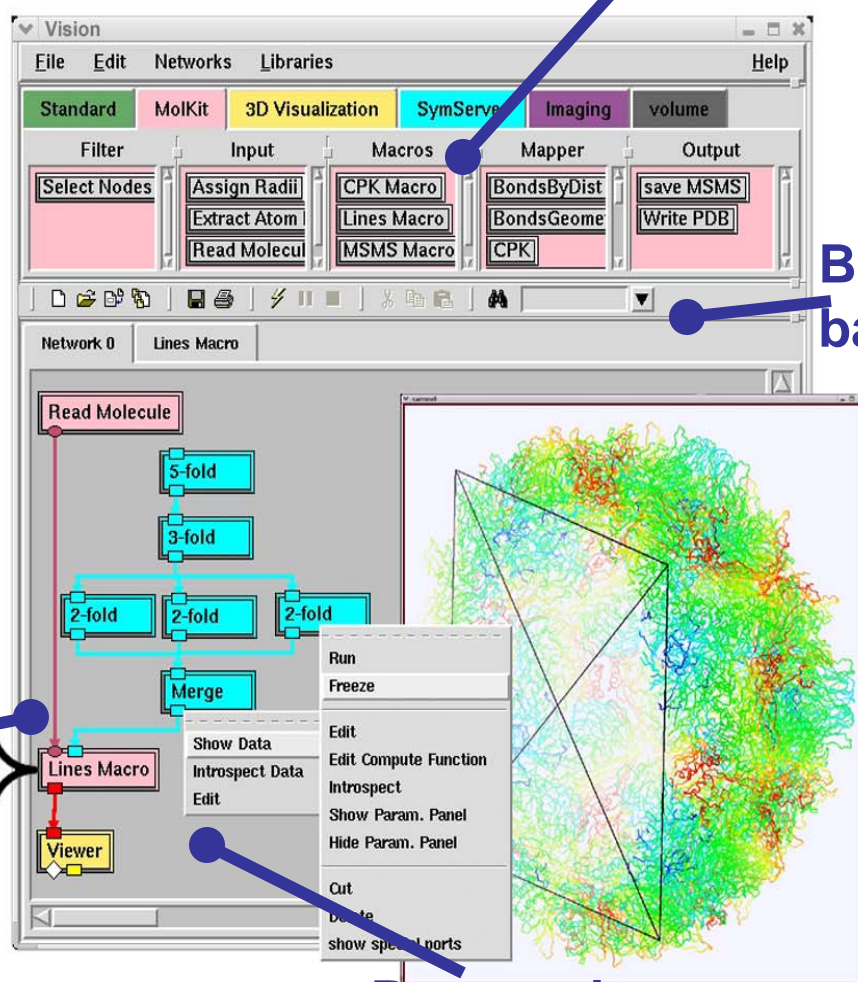
Node  
Editor



Macro  
Nodes



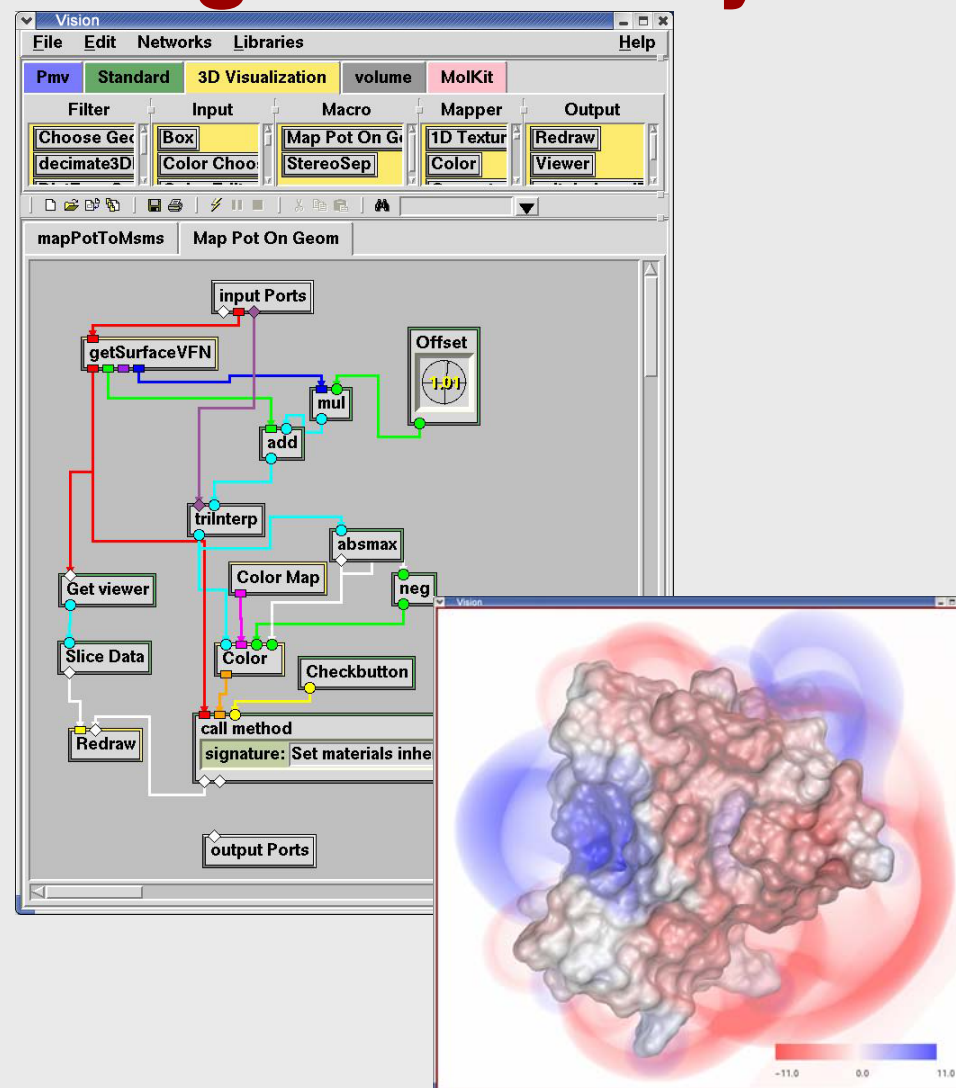
Node Libraries



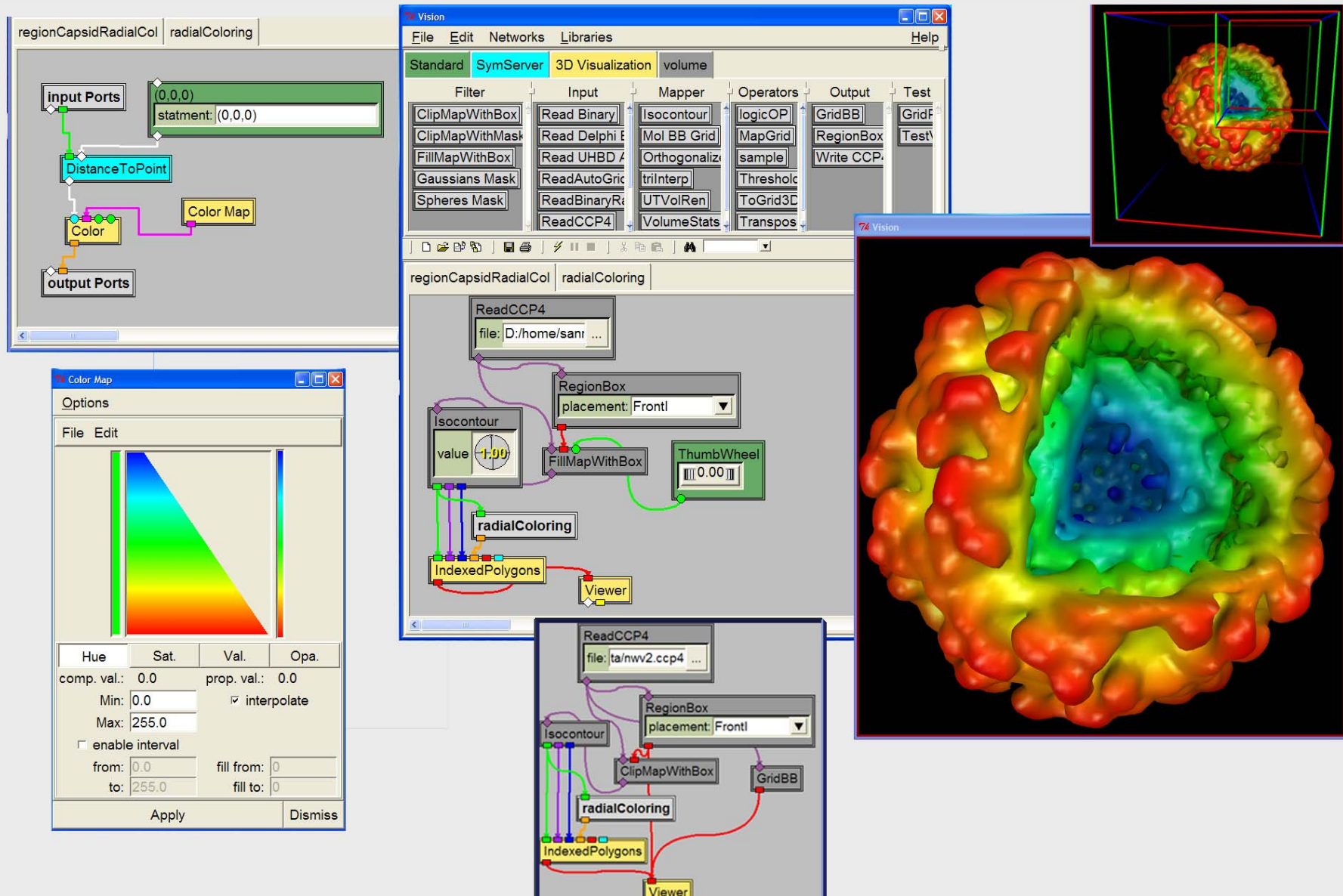
Programing  
Canvas

# Vision: programming made easy!

- **Abstracts** programming syntax and data structures
- Scientists (**non-programmers**) can build computational networks
- Rapid prototyping
- **Encapsulation** of basic tasks into shareable computational nodes
- Capture protocols



# Vision: a standalone application





# Vision for Integrating computational methods

The image displays a software interface for integrating computational methods, featuring several windows and a central network diagram.

**Molecule Viewer (Top Left):** Shows a 3D molecular model with a pink mesh surface and a red sphere. An orange arrow points from the '1crn' node in the NetworkBuilder to this model.

**NetworkBuilder (Top Right):** A window for building a network. It includes tabs for File, Edit, Networks, Libraries, and Help. The '3D Visualization' tab is active. The network diagram shows a '1crn' node connected to a 'Select MolFrag' node, which is connected to a 'UT-Isocontour' node. The 'UT-Isocontour' node is connected to an 'IndexedPolygons' node, which is connected to a 'RemoveDupVert' node, which is connected to a 'QSLim' node, which is connected to a 'Pmv Viewer' node. The 'Pmv Viewer' node is connected to the 'Molecule Viewer' window.

**UT-BlurSpheres(1) (Bottom Center):** A configuration window for the 'UT-BlurSpheres(1)' node. It includes options for 'blobbyness' (set to -0.0800), 'use Xres only' (checked), and 'X res (Xdim=103)', 'Y res (Ydim=91)', and 'Z res (Zdim=102)'. It also has a 'use volume dims' checkbox and a 'Dismiss' button.

**Molecule Viewer (Bottom Left):** A window for viewing the molecule. It includes a menu bar (File, Edit, Select, 3D Graphics, Un/Display, Color) and a status bar showing '0 Molecule(s)'. The 'PCOM' field is set to 'printNodeNames'. The 'Mod.' field is set to 'None'. The 'Time' field is set to '0.030'. The 'Selected' field is set to '0 Molecule(s)'. The 'self.browseCommands' and 'self.computeMSMS' methods are visible in the console.

**UT-Isocontour (Bottom Right):** A configuration window for the 'UT-Isocontour' node. It includes a 'value' field (set to 50.00) and an 'Iso. Val.' field. It also has a 'Dismiss' button.

**IndexedPolygons (Bottom Right):** A configuration window for the 'IndexedPolygons' node. It includes a 'Dismiss' button.

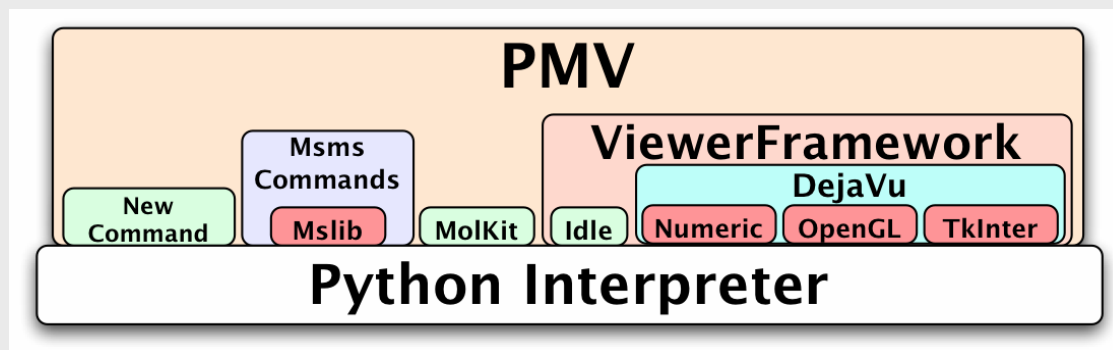
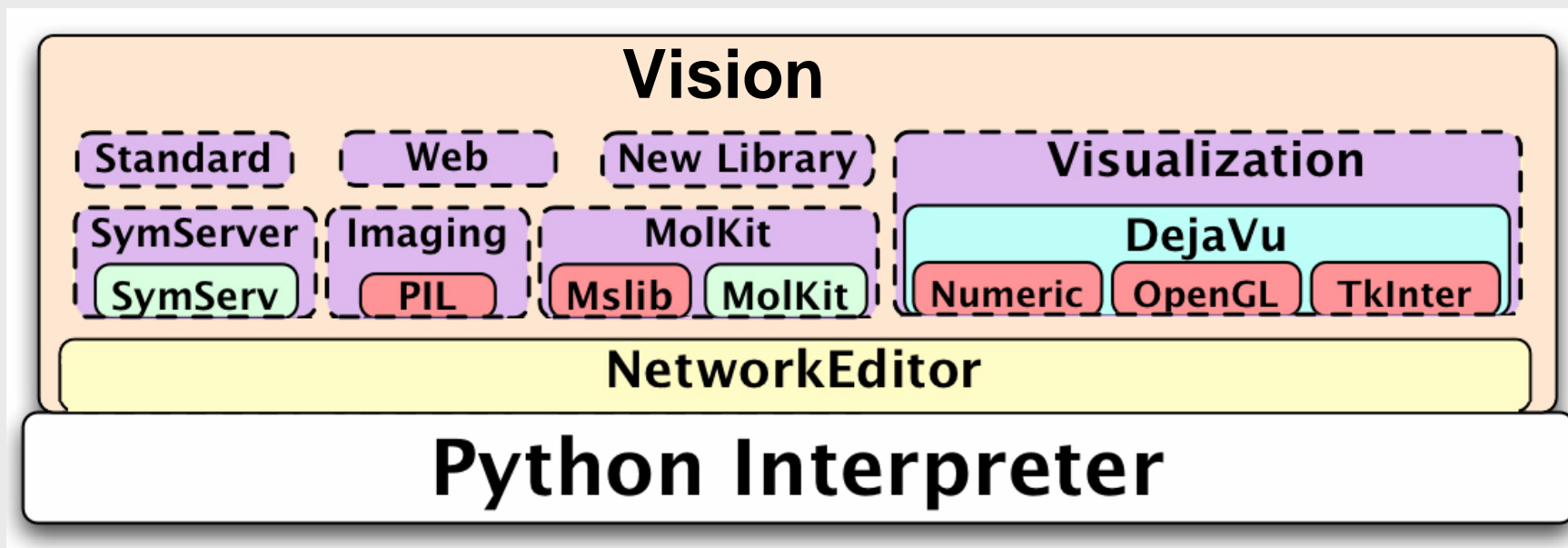
**RemoveDupVert (Bottom Right):** A configuration window for the 'RemoveDupVert' node. It includes a 'Dismiss' button.

**QSLim (Bottom Right):** A configuration window for the 'QSLim' node. It includes a 'Dismiss' button.

**Pmv Viewer (Bottom Right):** A configuration window for the 'Pmv Viewer' node. It includes a 'Dismiss' button.

**TSRI (Bottom Right):** The logo for the Technical Support and Research Institute (TSRI), featuring a stylized molecular structure.

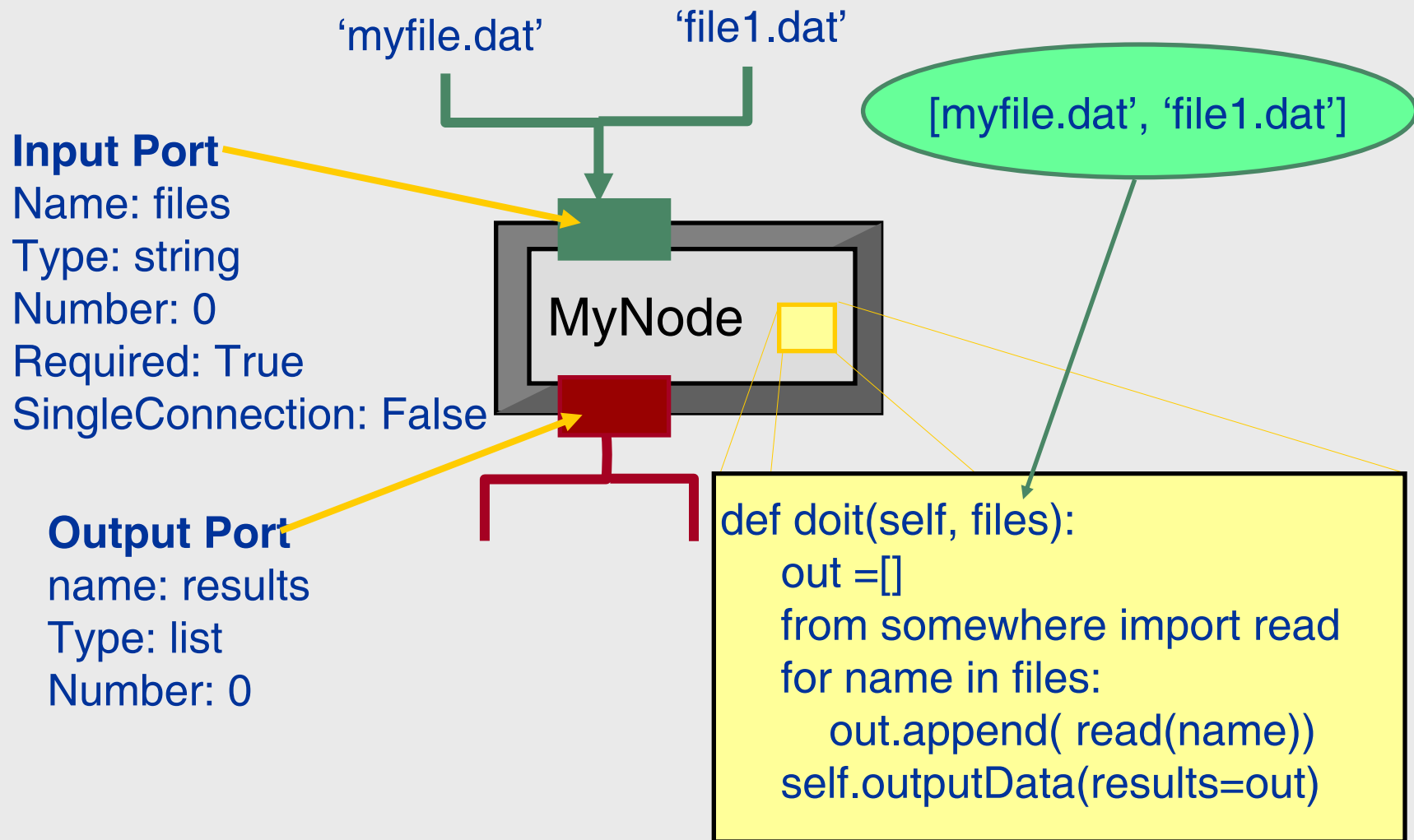
# Vision Architecture



# Vision Innovations

- Based on reusable components
- Dynamic: on-the-fly node editing
- Scriptable, flexible
- Platform independent
- No constraining data types or data model
- Optional data duplication
- Nodes are lightweight wrappers of computations
- Automatic detection of Vision interface in other Python packages
- Both: a Python package AND a program !
  - Can be added to any program
  - API for exposing 3<sup>rd</sup> party application's objects

# Anatomy of a Network Node



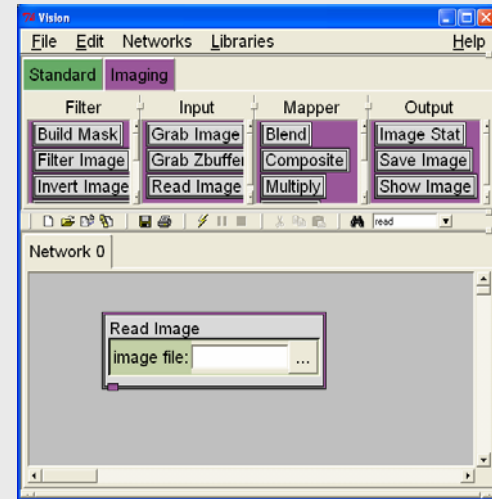


# Anatomy of a Network Node

```
class ReadImage(NetworkNode):
    """This node reads an image file.
    Input: filename (string)
    Output: Image"""
    def __init__(self, name='Read Image'):
        NetworkNode.__init__(self, name=name)
        self.readOnly = 1
        self.inputPortsDescr.append( {'name':'filename', 'datatype':'string'} )
        self.outputPortsDescr.append( {'name':'image', 'datatype':'image'} )
        self.widgetDescr['filename'] = {'class':NEEntryWithFileBrowser,
                                         'master':'node', 'filetypes': [('all', '*')],
                                         'title':'read image', 'width':10 }
        code = """def doit(self, filename):
                    import Image
                    im = Image.open(filename)
                    if im: self.outputData(image=im)\n"""
```

```
self.setFunction(code)
```

SciPy'05, Sept. 2005, Caltech, Pasadena, CA



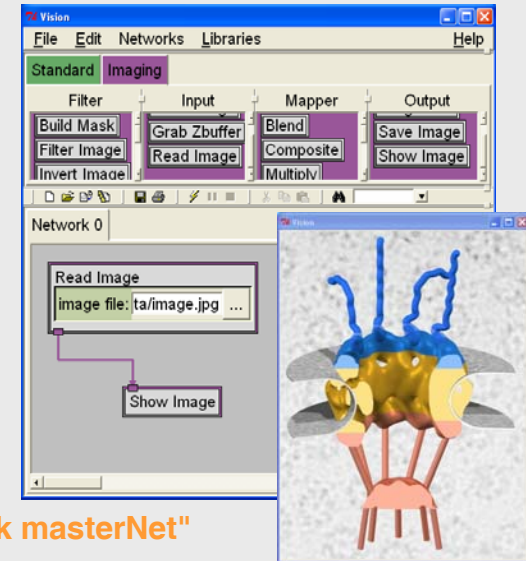
# Saving Networks

```
from traceback import print_exc
#### Network: Demo Network ####
#### File written by Vision ####
## loading libraries ##
from Vision.PILNodes import imagelib
masterNet.getEditor().addLibraryInstance(imagelib,
                                         "Vision.PILNodes", "imagelib")

try:
    ## saving node Read Image ##
    from Vision.PILNodes import ReadImage
    node0 = ReadImage(constrkw = {}, name='Read Image', library=imagelib)
    masterNet.addNode(node0,100,20)
    node0.inputPorts[0].widget.set("Data/image.jpg",0)
except:
    print "WARNING: failed to restore ReadImage named Read Image in network masterNet"
    print_exc()
    node0=None

try:
    ## saving node Show Image ##
    from Vision.PILNodes import ShowImage
    node1 = ShowImage(constrkw = {}, name='Show Image', library=imagelib)
    masterNet.addNode(node1,164,189)
except:
    print "WARNING: failed to restore ShowImage named Show Image in network masterNet"
    print_exc()
    node1=None

## saving connections for network Demo Network ##
if node0 is not None and node1 is not None:
    masterNet.connectNodes(
        node0, node1, "image", "image", blocking=True)
```



# Node Library / Data Types

```
newlib = NodeLibrary('mylibrary', '#AAEECC')  
newlib.addNode(ReadImage, 'Read Image', 'input')
```

```
class ImageType(AnyType):  
    def __init__(self):  
        self.name = 'image'  
        self.color = '#995699'  
        self.shape = 'rect1'  
    def validate(self, data):  
        import Image  
        return isinstance(data, Image.Image)  
    def cast(self, data):  
        return False, None # or True, cast(data)
```

```
newlib.typesTable.append( ImageType() )
```

# Vision

- Refactoring:
  - New implementation of:
    - Widget management, network saving/restoring, data validation and casting mechanism, etc.
  - Delocalization of node libraries
    - Node libraries in software components

```
DejaVu/  
...  
VisionInterface/  
DejaVuNodes.py
```

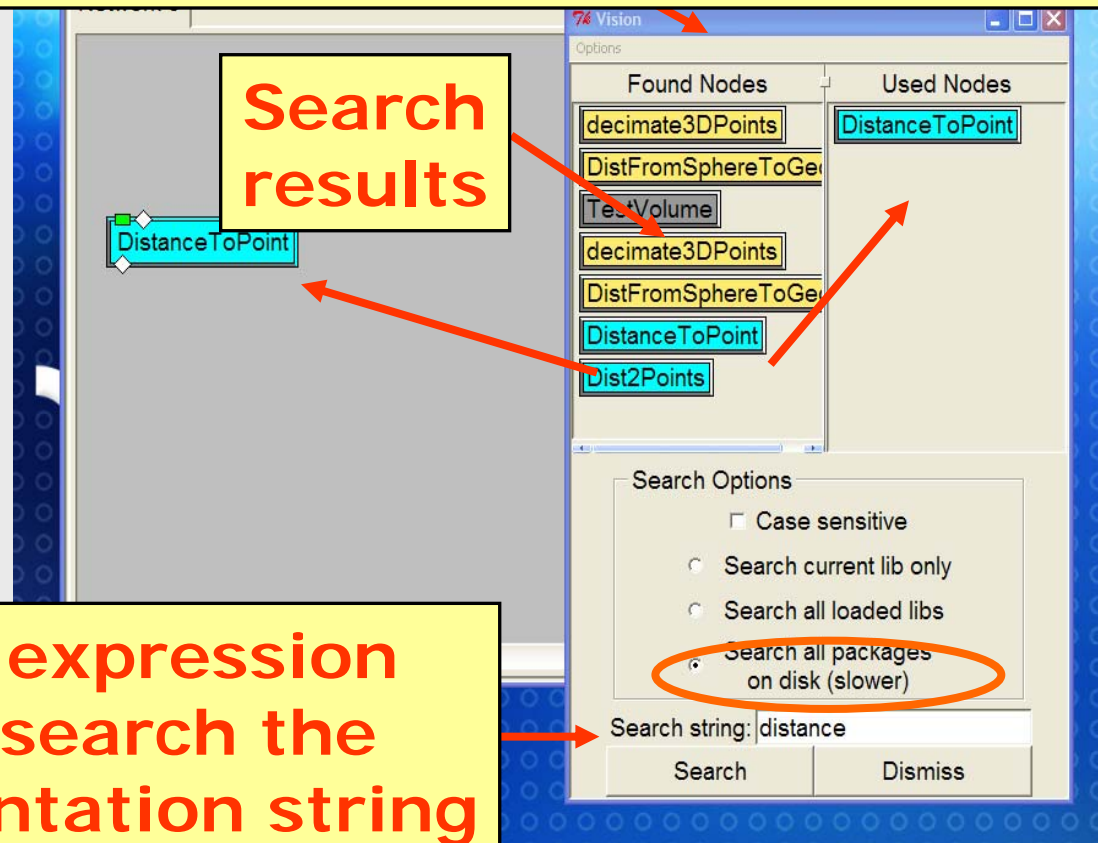


# Vision

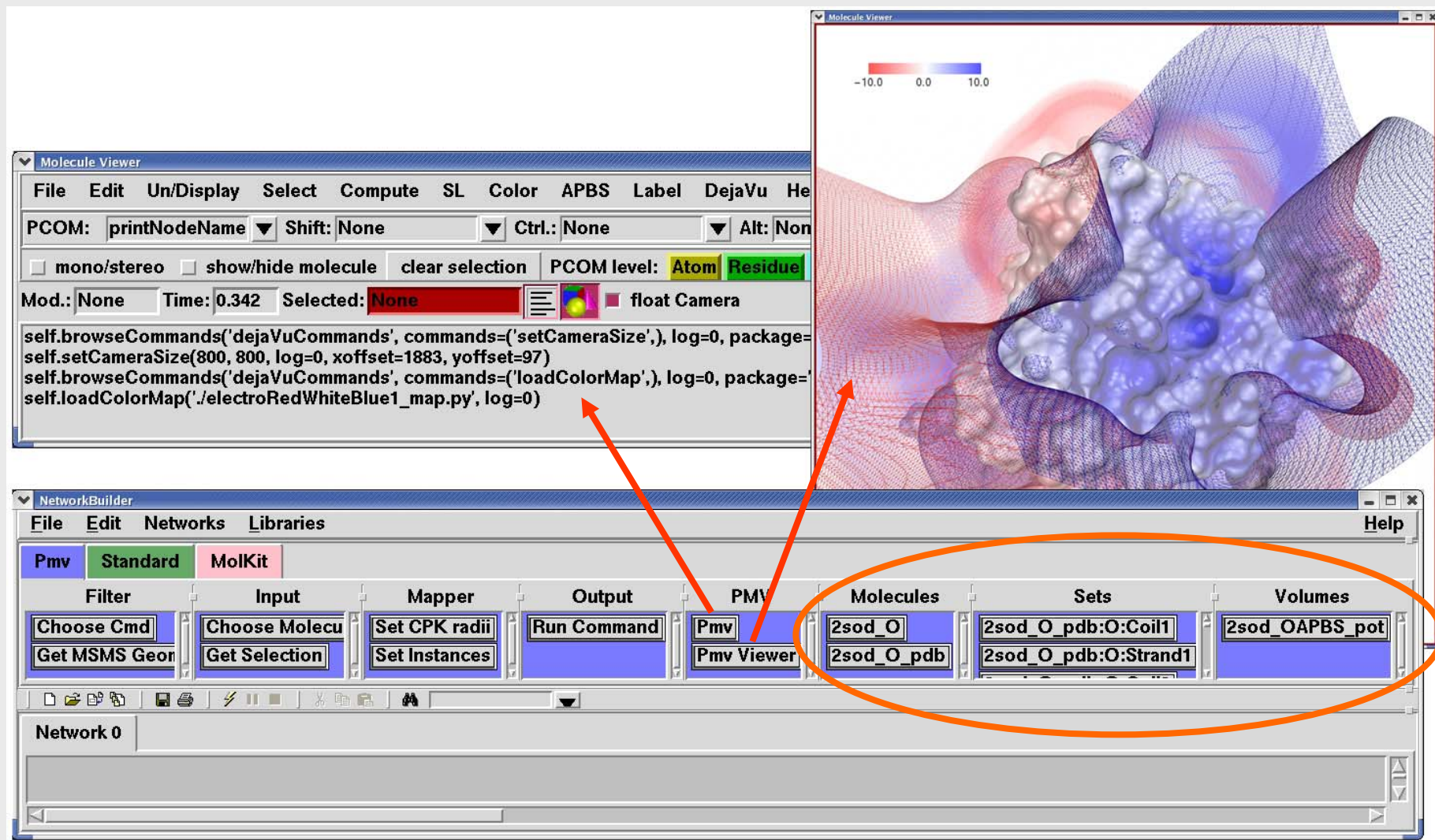
- Added Features:
  - Node discovery mechanism
  - API for integration with other applications
  - Application packaging support (UserPanels)
  - New library of nodes for volumetric data
  - User definable node libraries

# Node Discovery

Mechanism for searching future Web-based repositories of nodes and networks

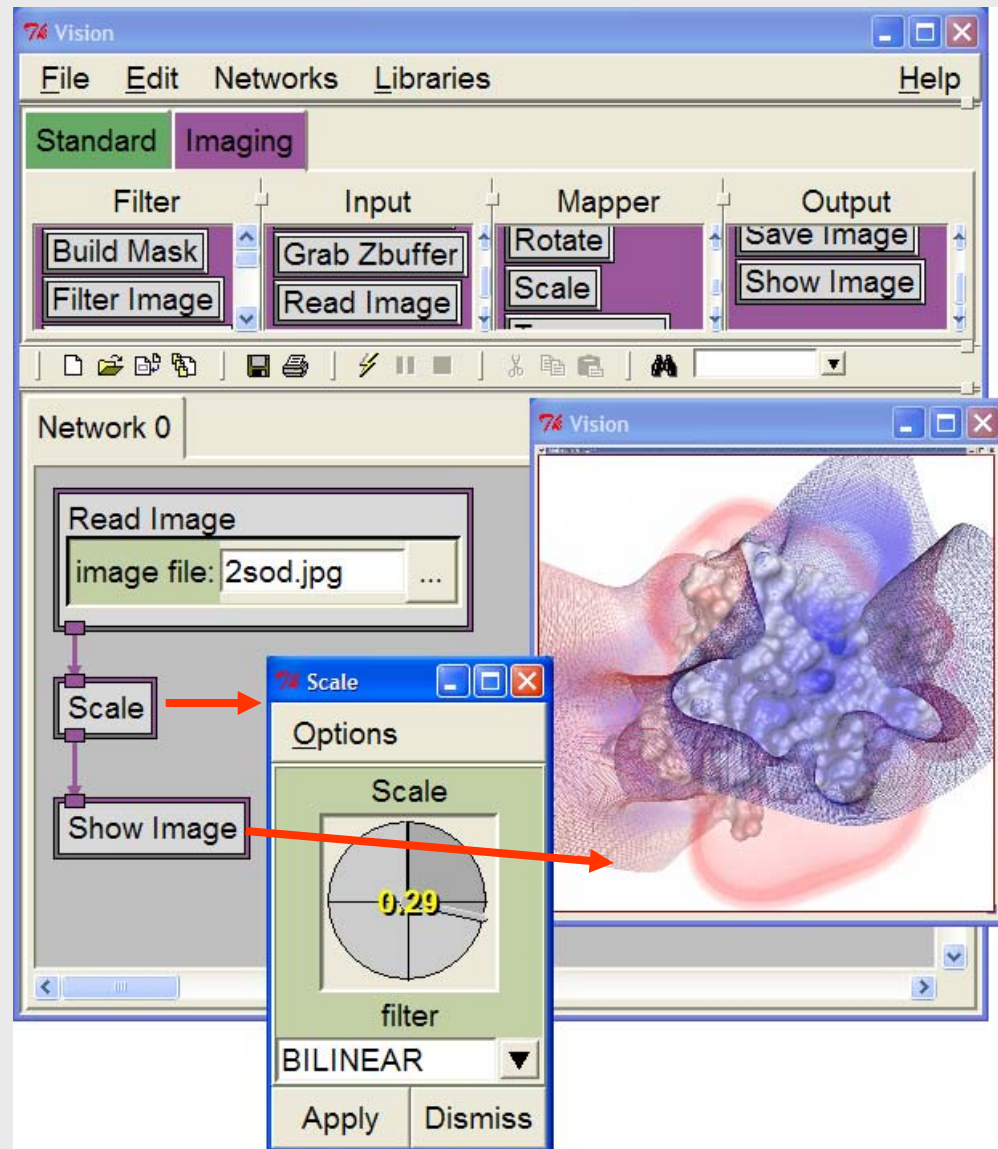


# Integration API



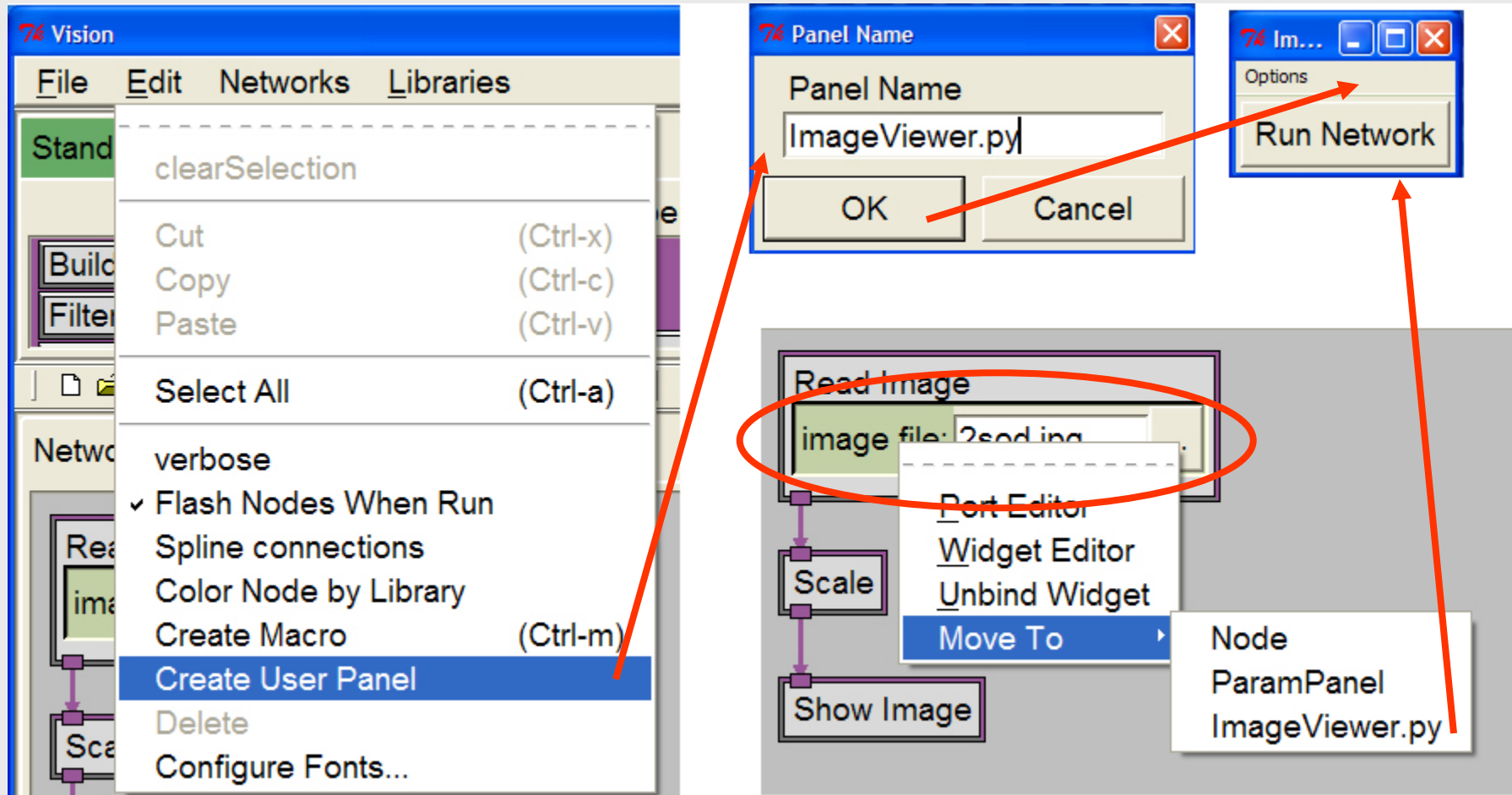
# End-user applications

**"image viewer":**  
An application  
Implemented as  
a Vision network

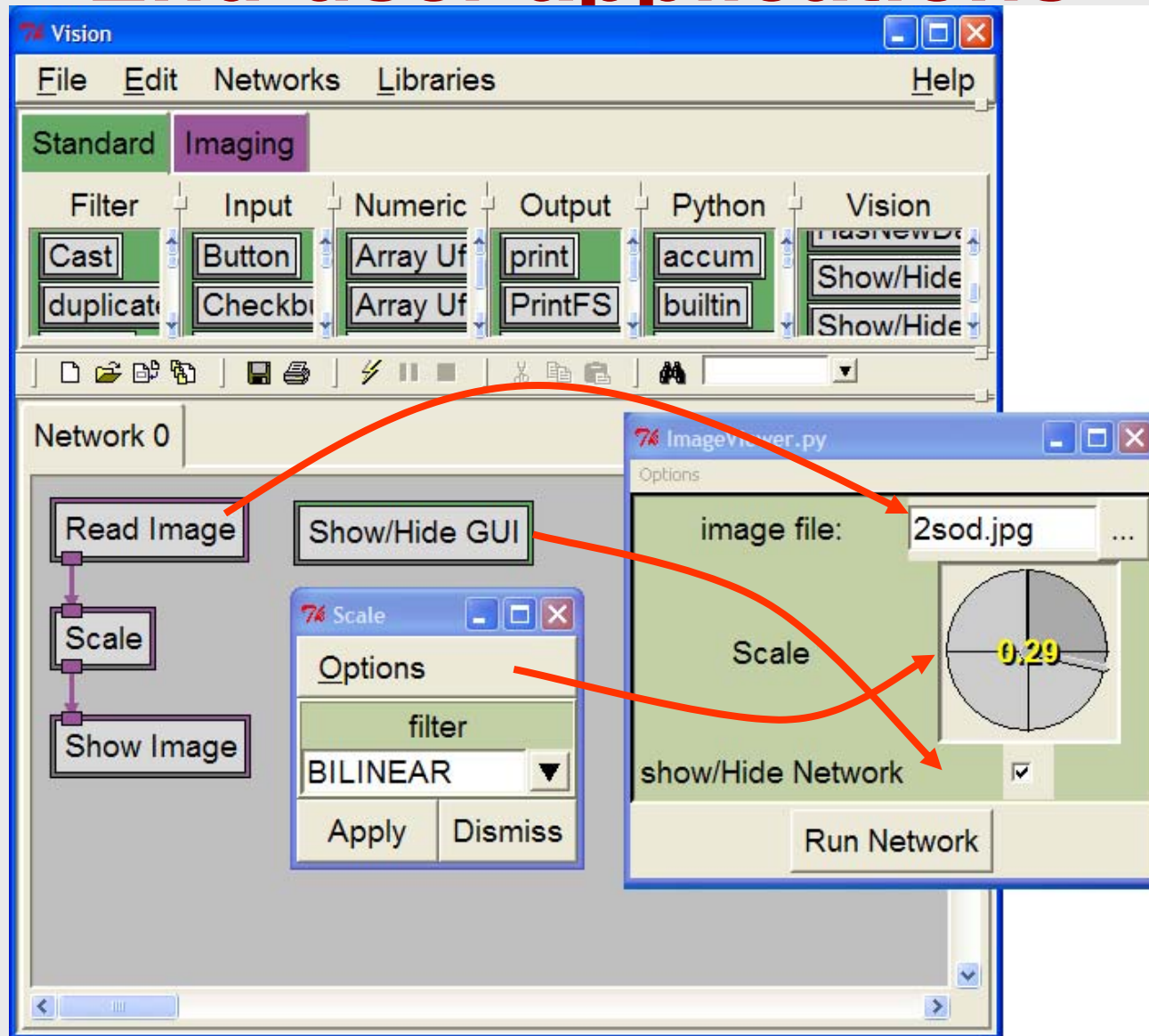




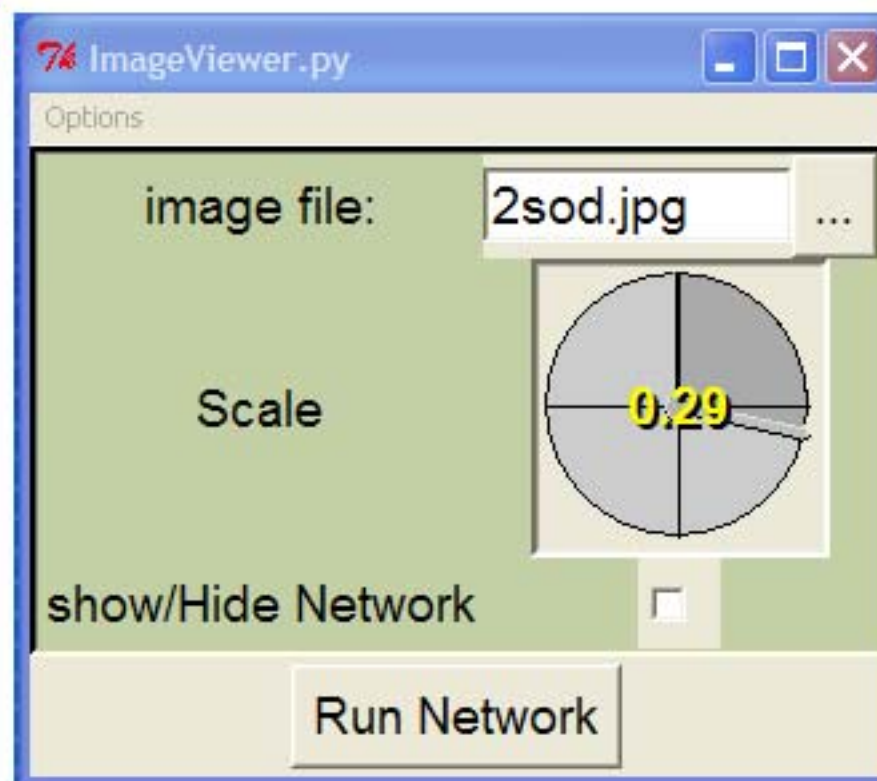
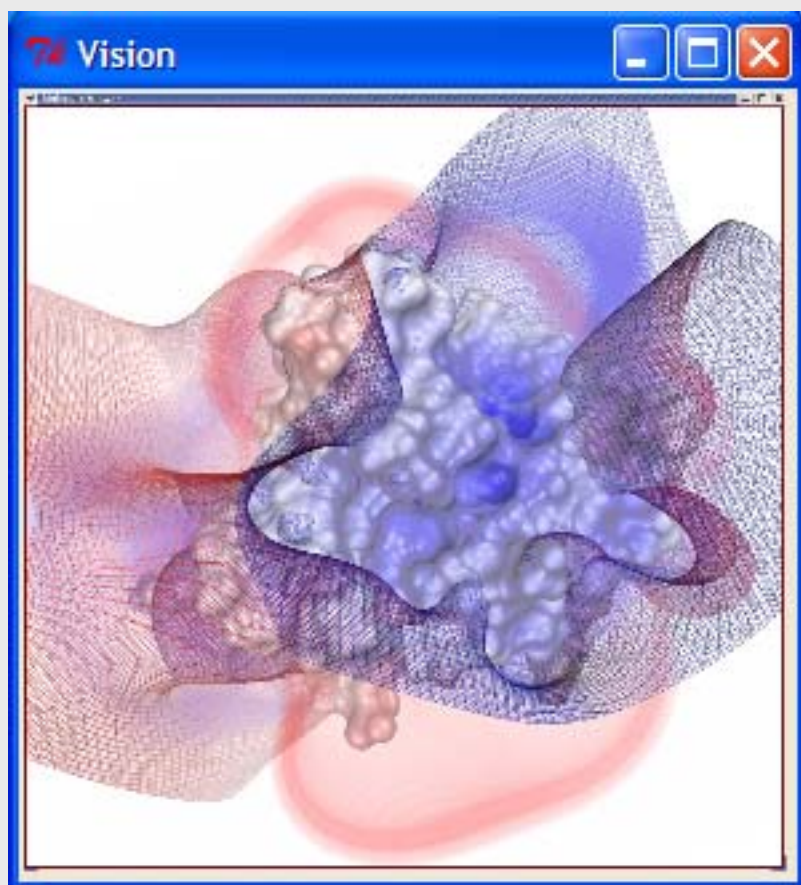
# End-user applications



# End-user applications



# End-user applications



# Acknowledgments

- Sophie Coon (PMV)
- Anna Omelchenko (PVV)
- Daniel Stoffler (Vision)
- Alex Gillet (PyARTK)
- Sowjanya Karnati (Unit tests)

NSF (NPACI, CA ACI9619020 )

NIH (NBCR: RR08605, BISTI: GM65609)



<http://www.scripps.edu/~sanner/software>