Vision: Visual Programming
"Integration of Reusable Software Components"

Michel F. Sanner

The Molecular Graphics Laboratory

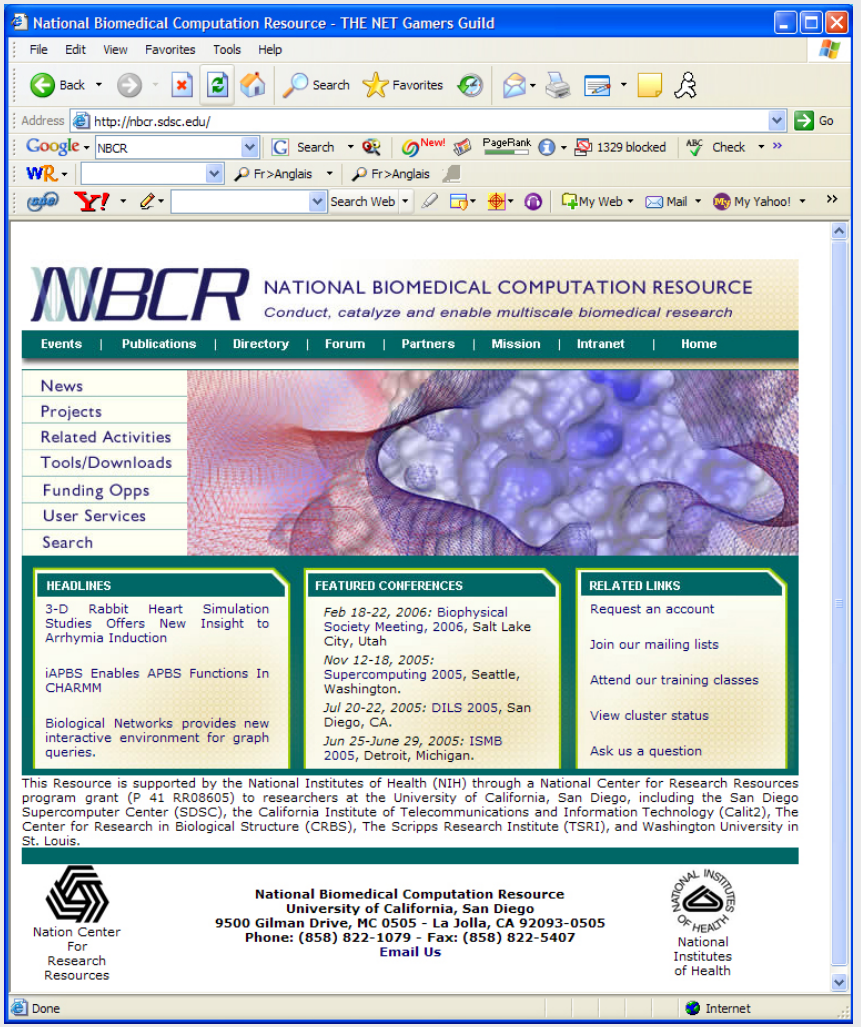
Molecular Biology Department

The Scripps Research Institute
La Jolla, California

TSRI

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).



TSRI

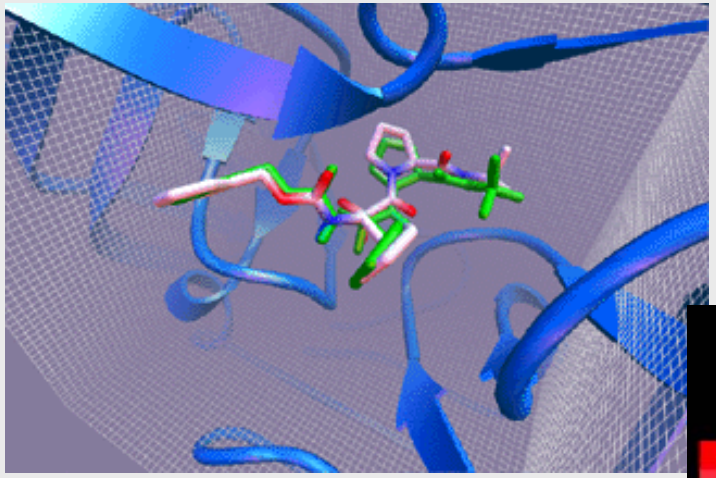
Outline

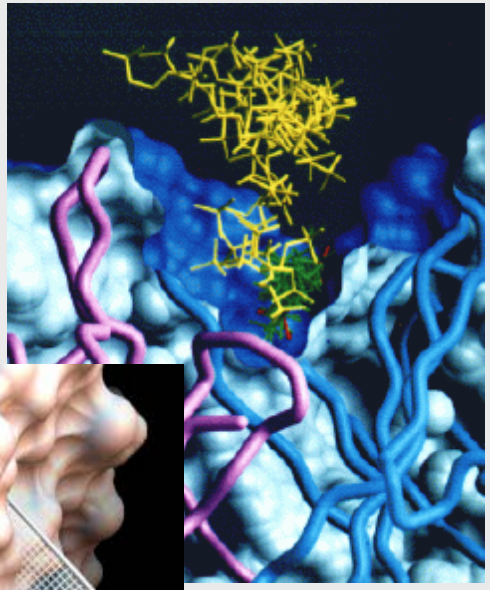
- **Background and Motivation**:
  - Scientific interests => software engineering challenge
- **Component-based software development**:
  - Overview of the strategy
  - Software components (MolKit, DejaVu,…)
- **Applications built from software components**
  - PMV: the Python Molecular Viewer, (ADT,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

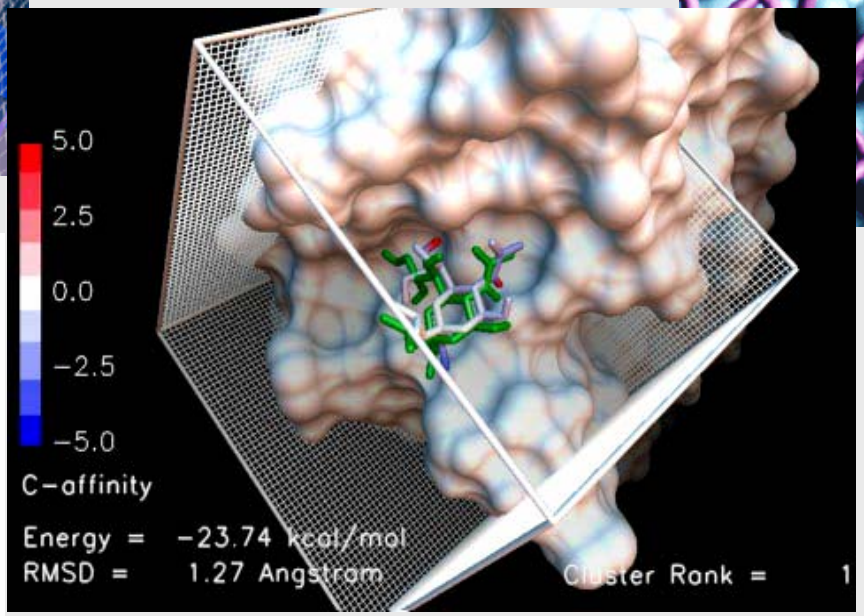
Scientific Interests

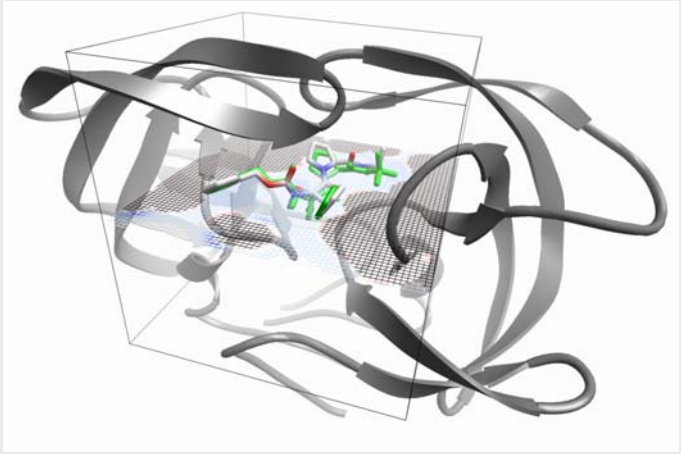
- **Modeling Molecular Interactions**
  - Design of computational models
  - Application to specific biological systems
    - HIV-I protease, Blood coagulation initiation
- **Molecular Visualization**
  - Emphasis on interactivity

AutoDock: Automated Docking of Flexible Ligands





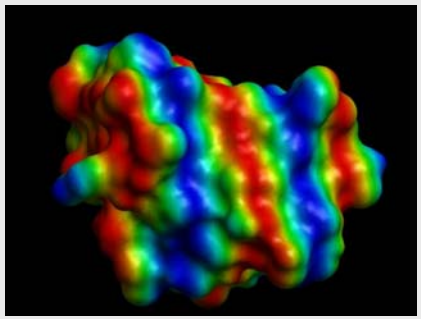


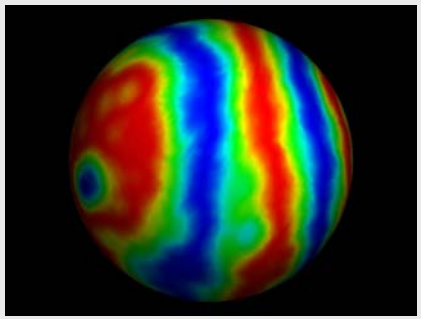


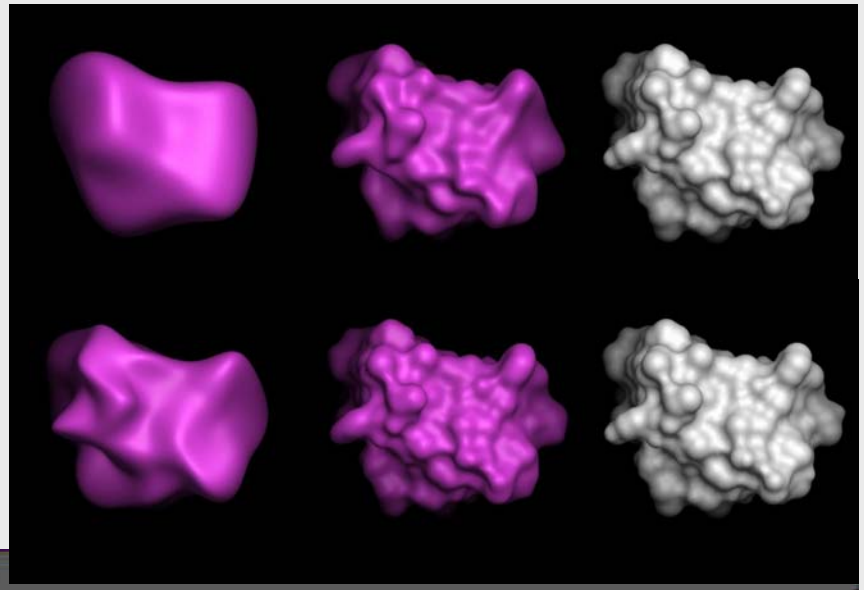
Courtesy G. M. Morris

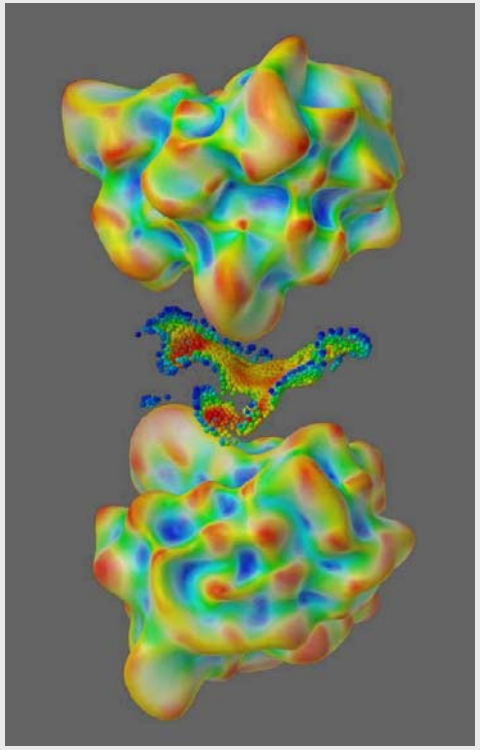
Morris *et al*. (1998) J. Comp. Chem. 19(14):1639-1662

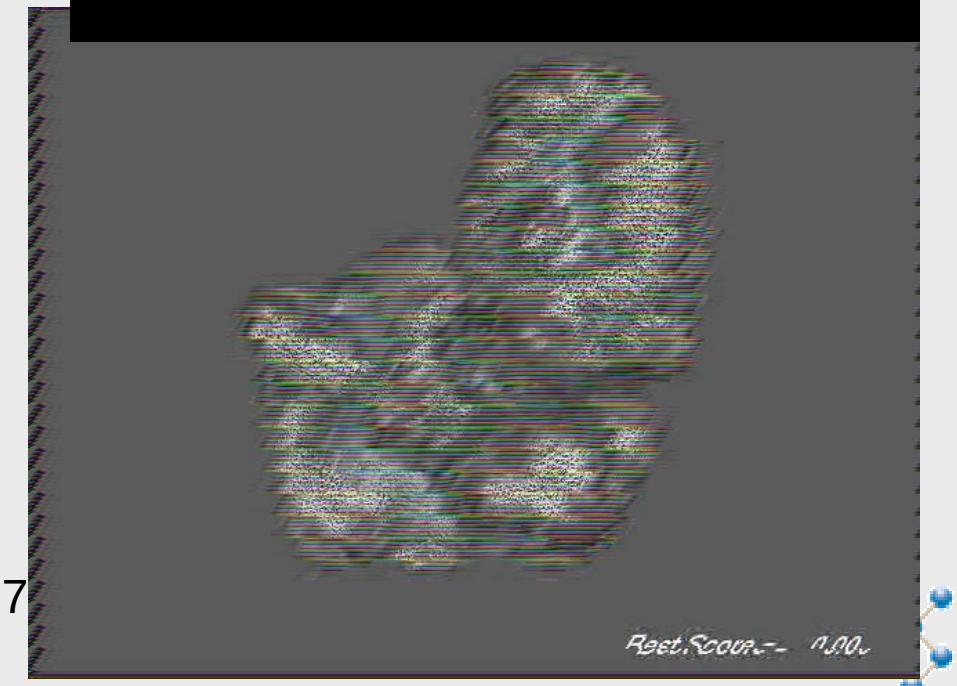
SurfDock: Protein-Protein Docking





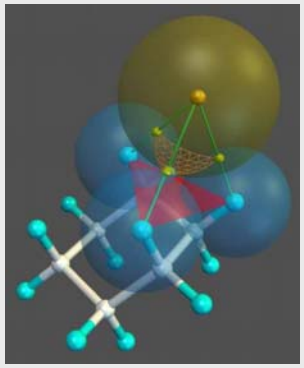
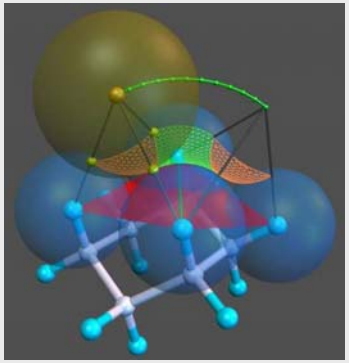
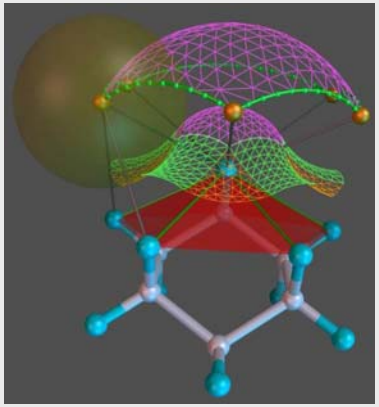
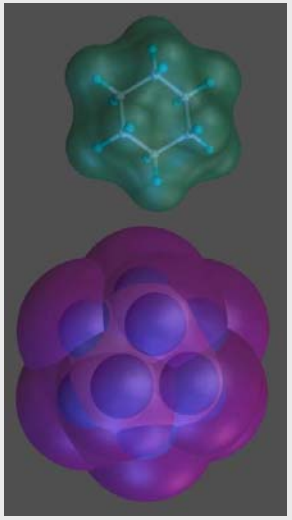
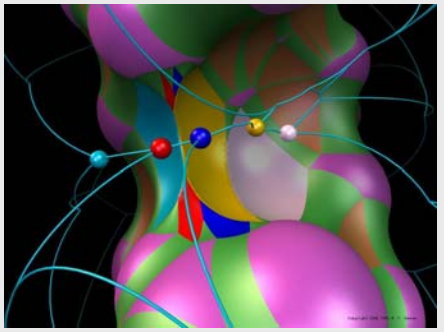
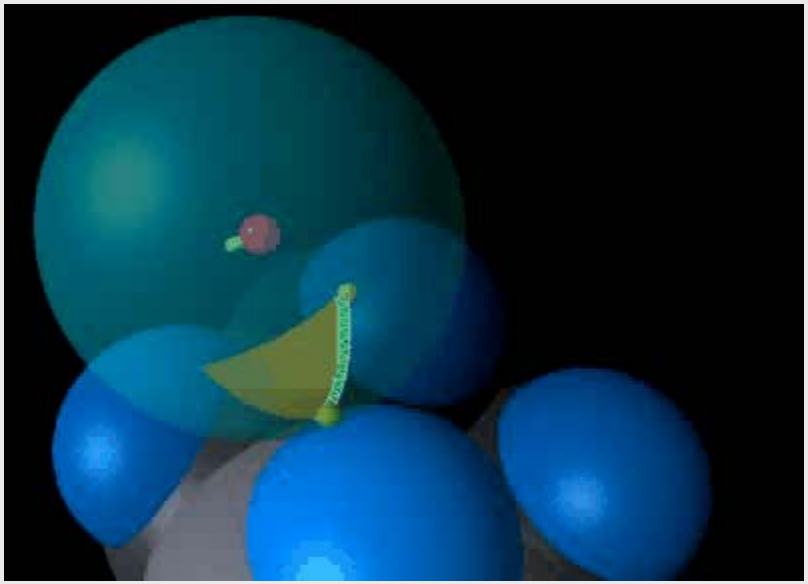
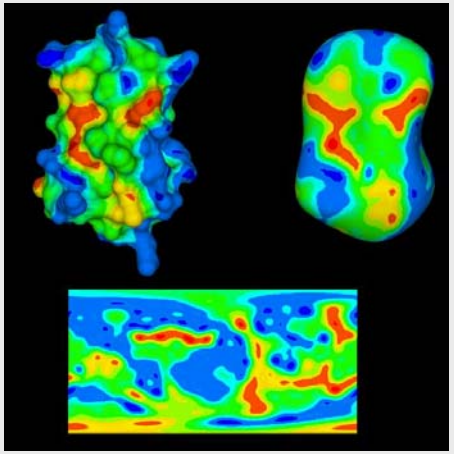






Duncan (1995) J. Mol. Graphics 13:250-257

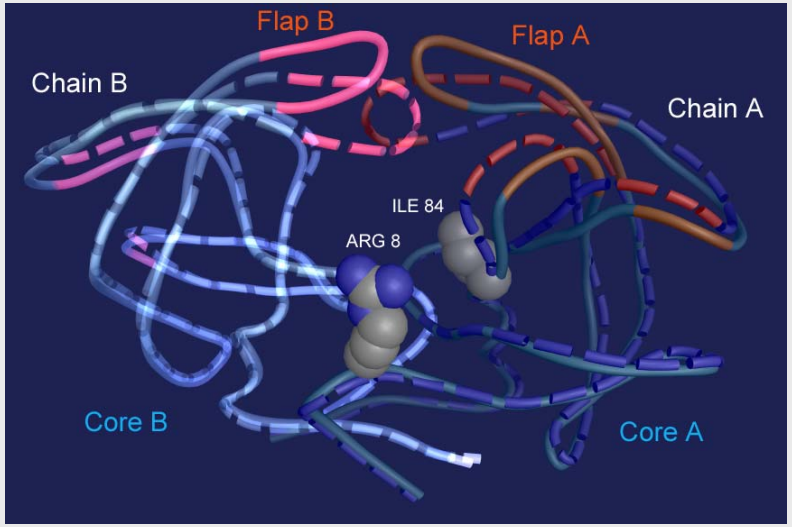
MSMS: Molecular Surfaces

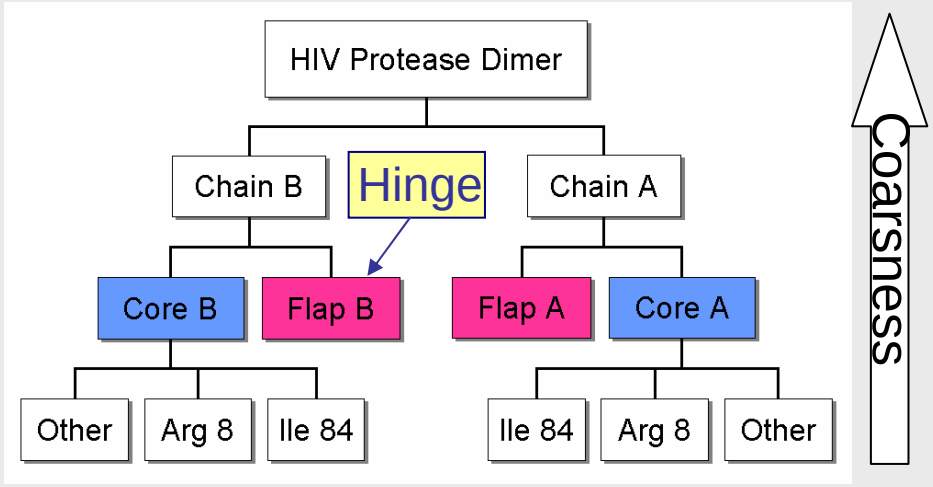








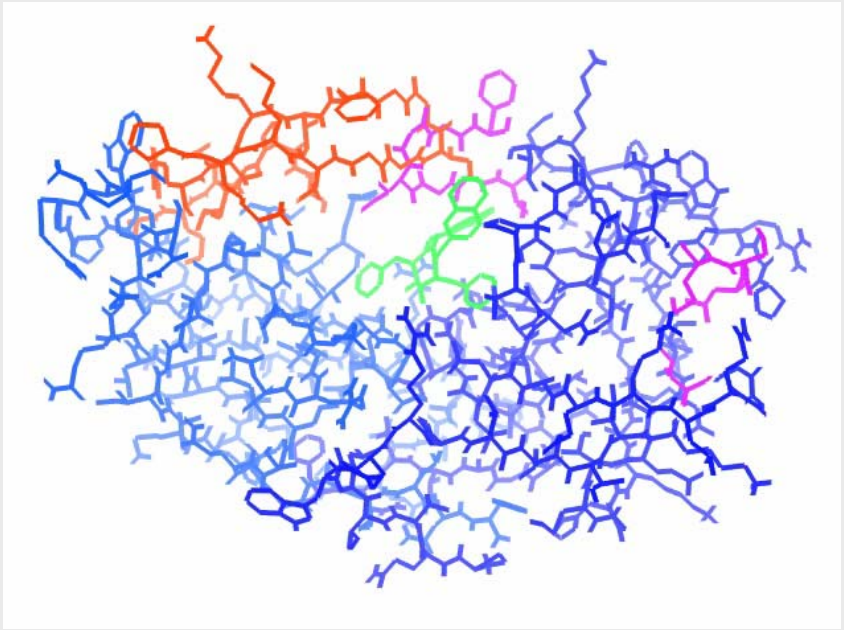
Sanner *et al.* (1995) Biopolymers, 38: 305-320

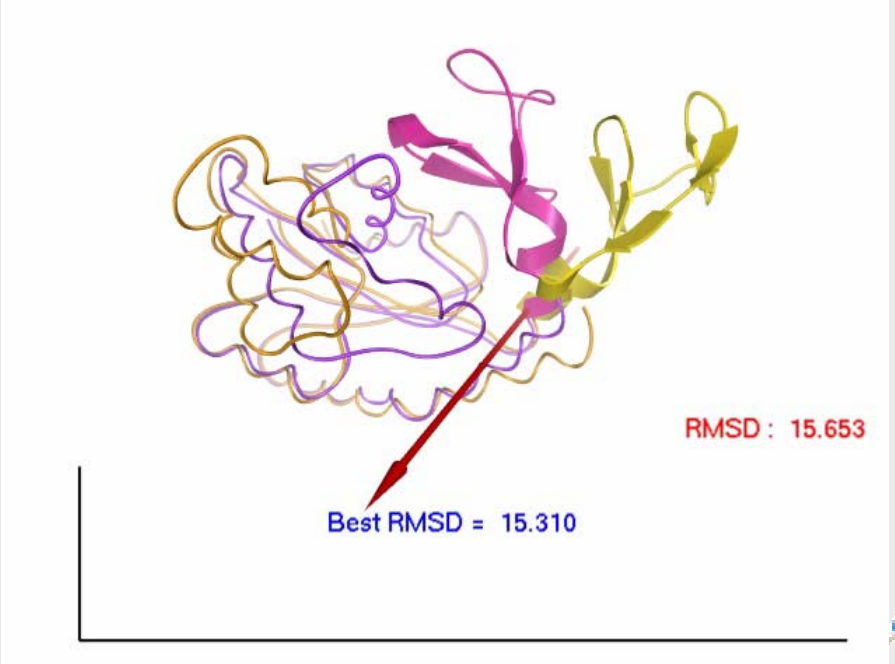
TSRI

Flexibility Tree: Protein Flexibility



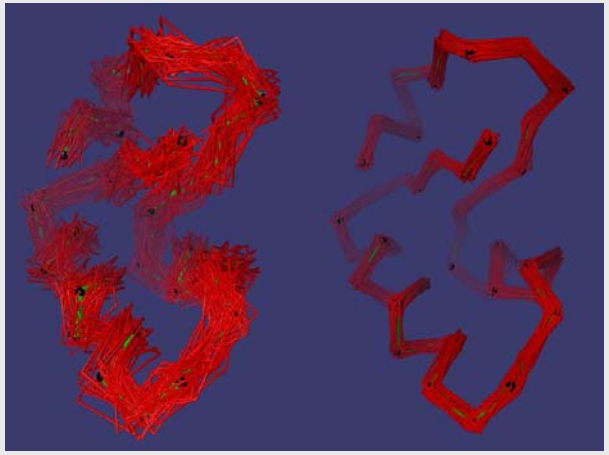


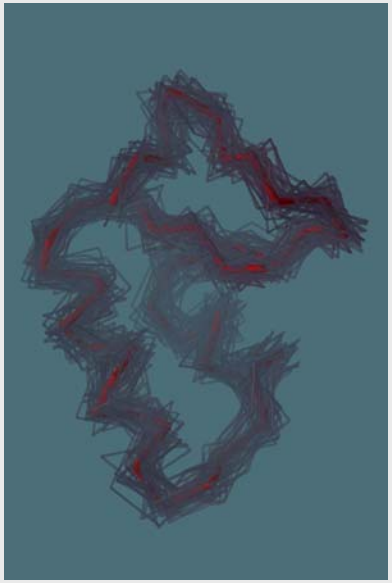


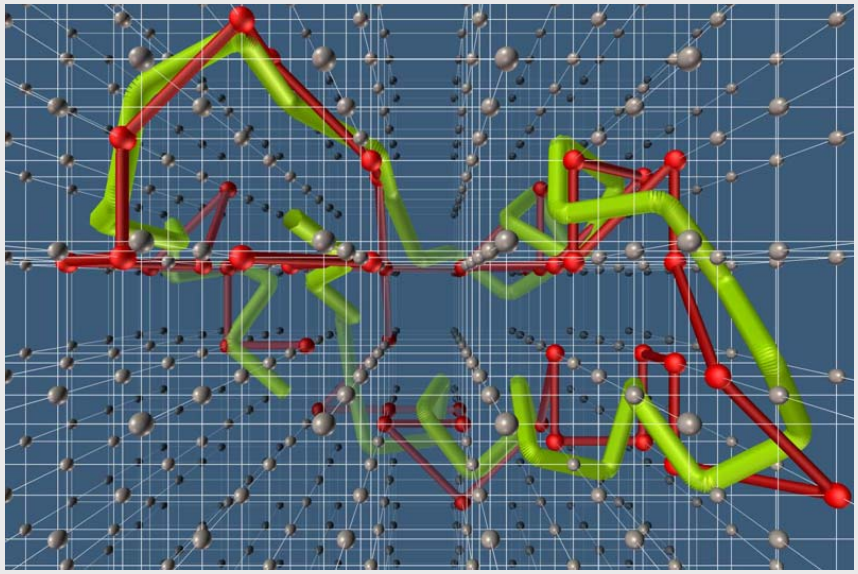


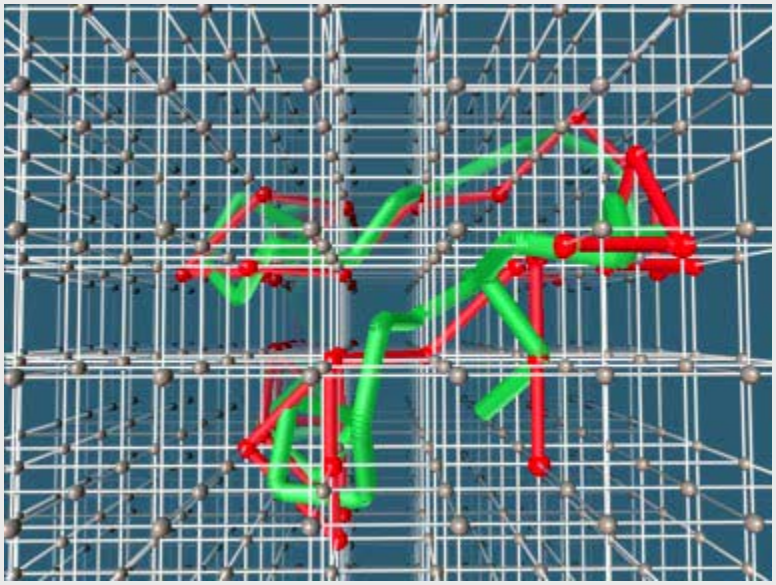
TSRI

Protein Folding



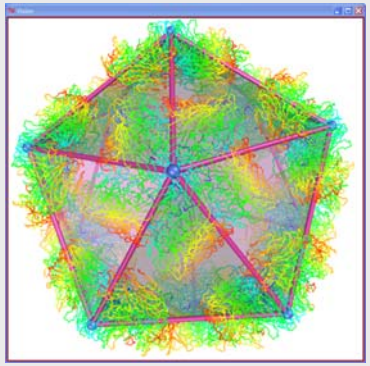
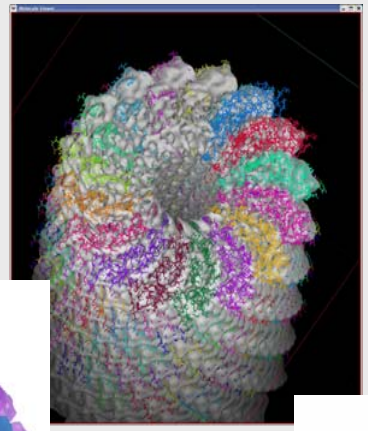
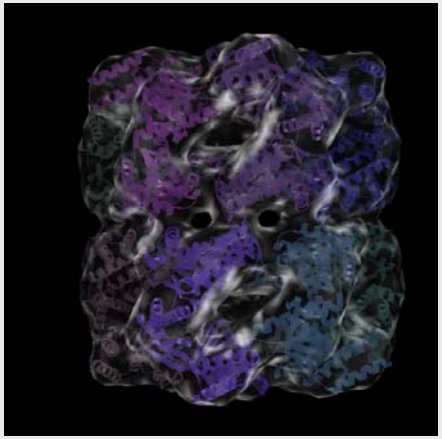
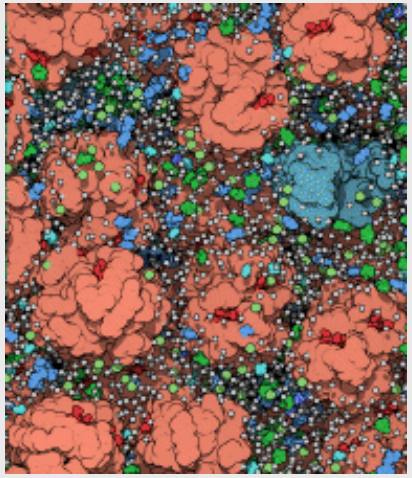
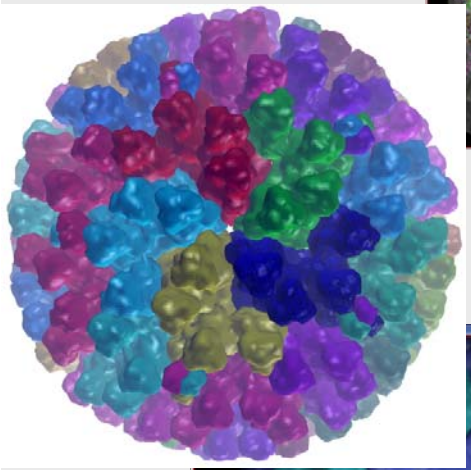
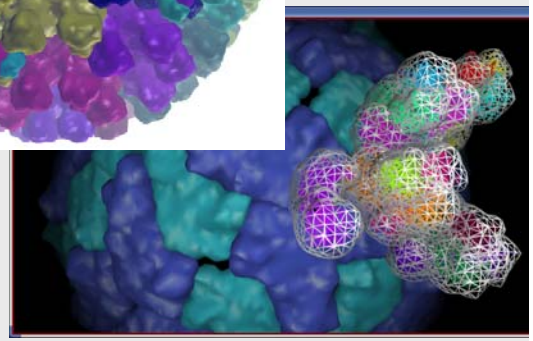
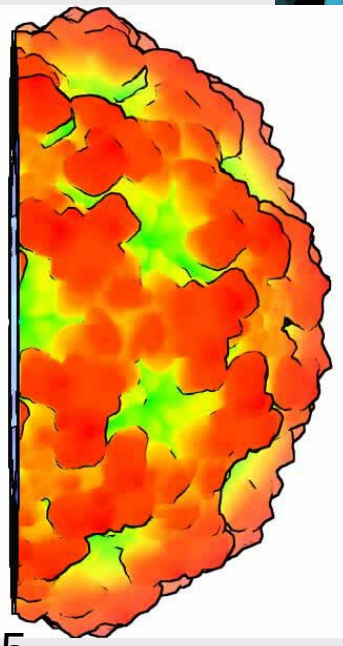
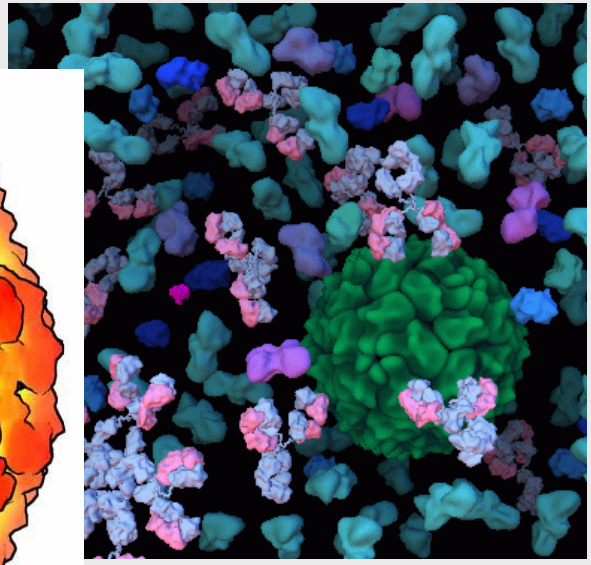






Reva, Sanner *et al.* Proteins: Struct., Funct., Genetics, 25:379-388

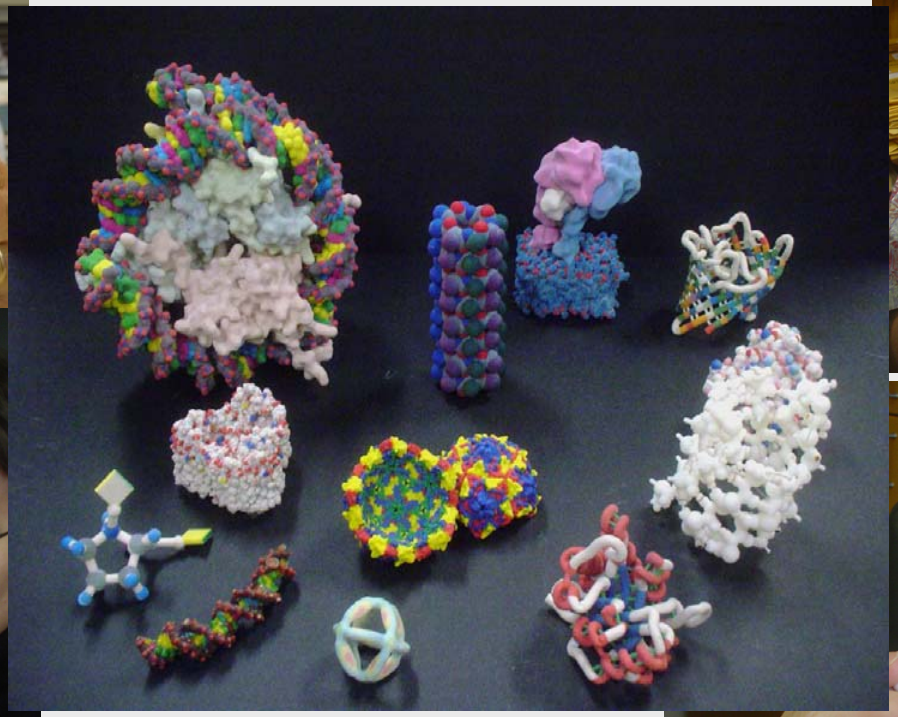
Complex Assemblies

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

TSRI

Tangible Models



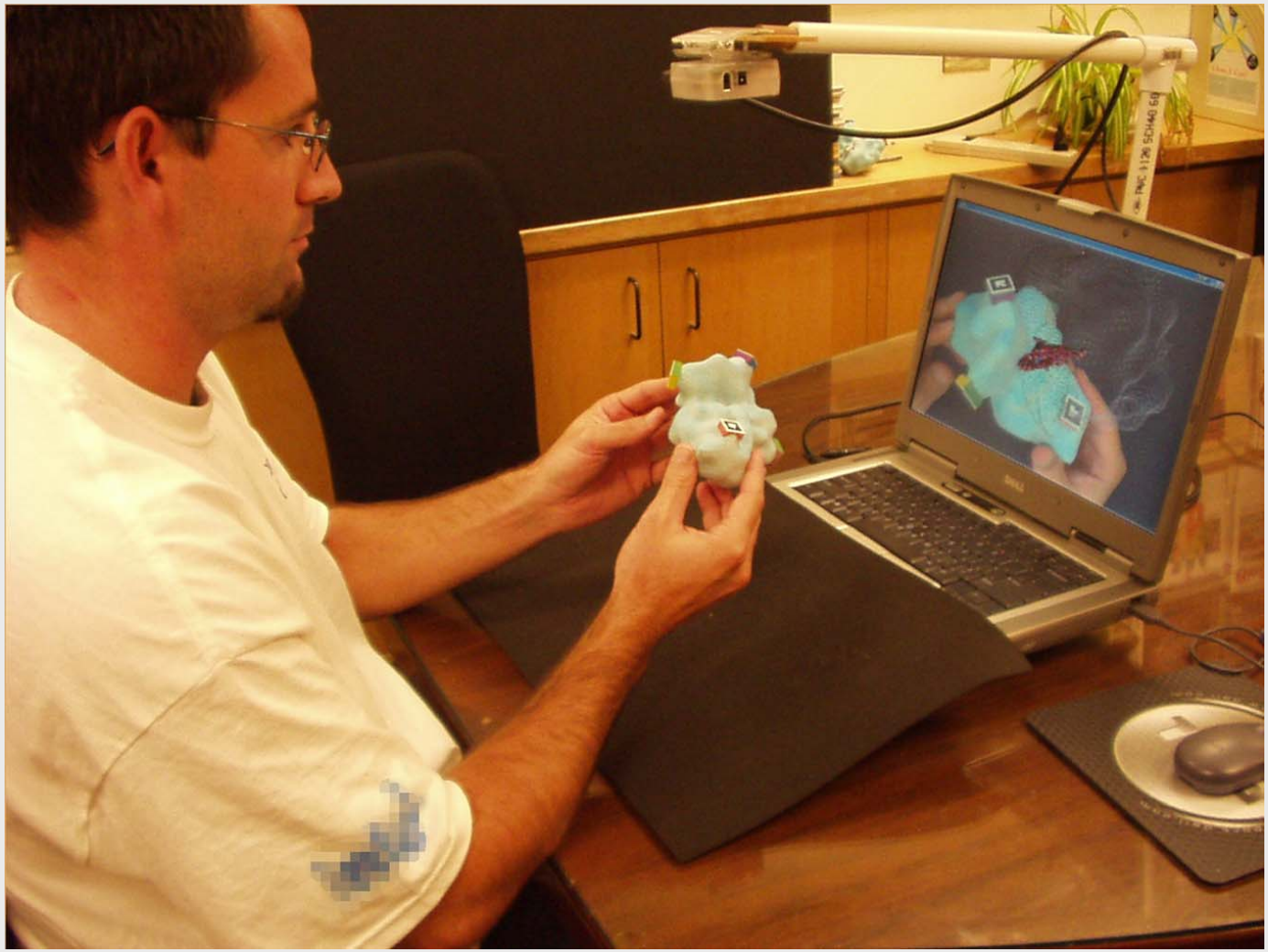
Courtesy A. J. Olson

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

Tangible models: printing

Video by A. Gillet and A. J. Olson

Augmented Reality

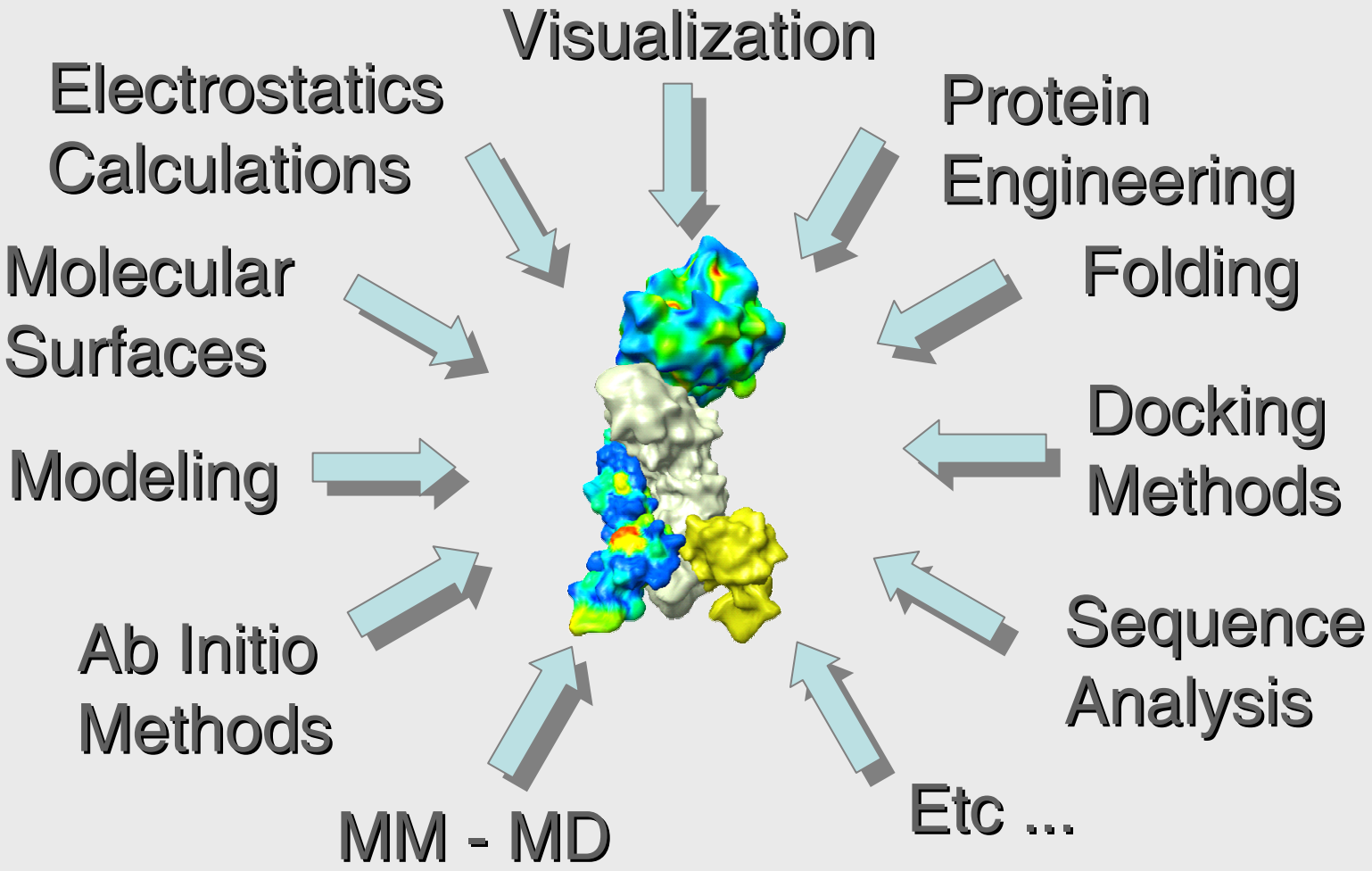


Gillet, Sanner, *et. al*. IEEE Computer Graphics and Applications. March-April 2005, Vol 25, Number 2, p13-17

Video by A. Gillet and A. J. Olson

TSRI

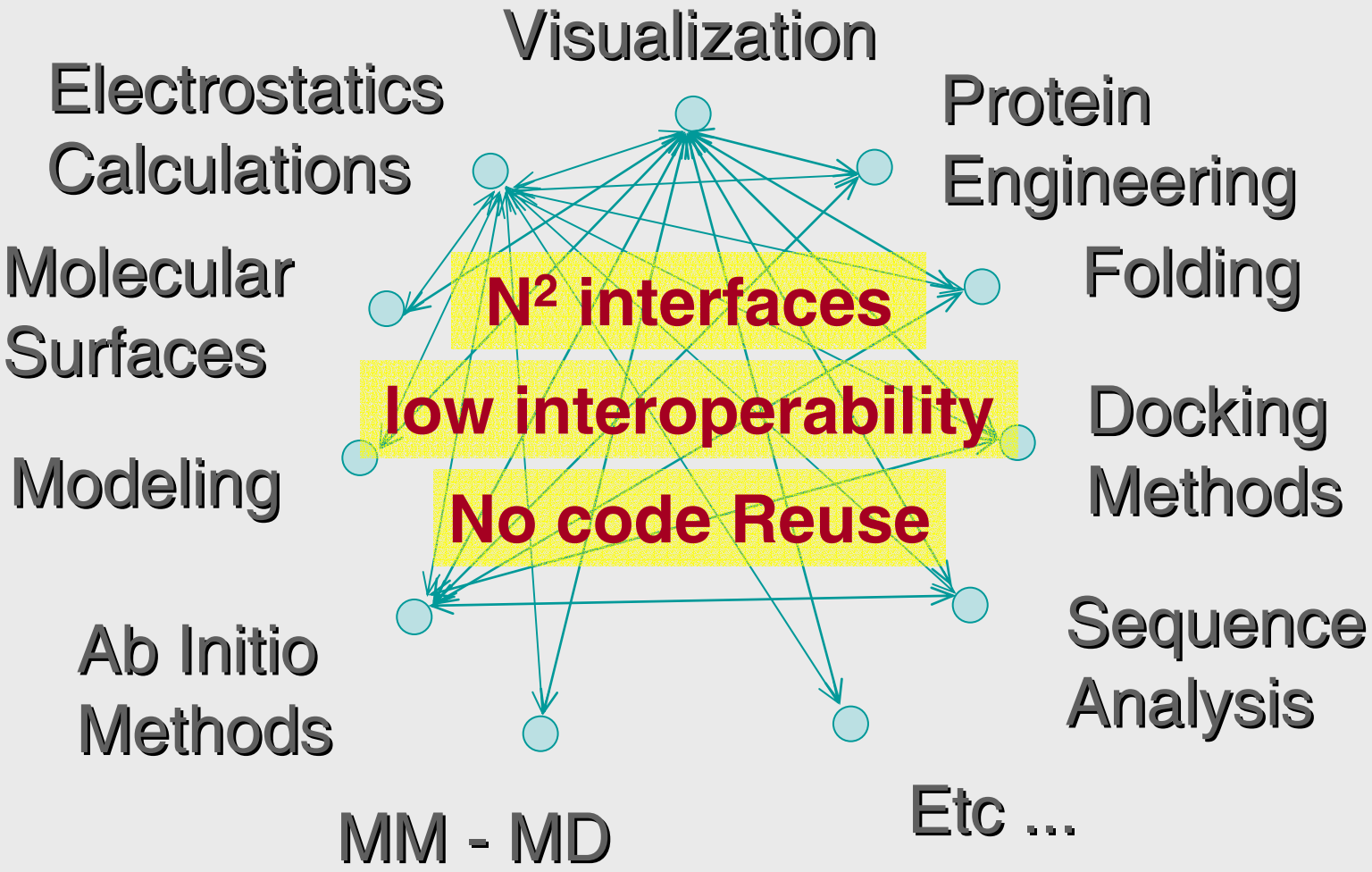
A Software Engineering Challenge



Challenges

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

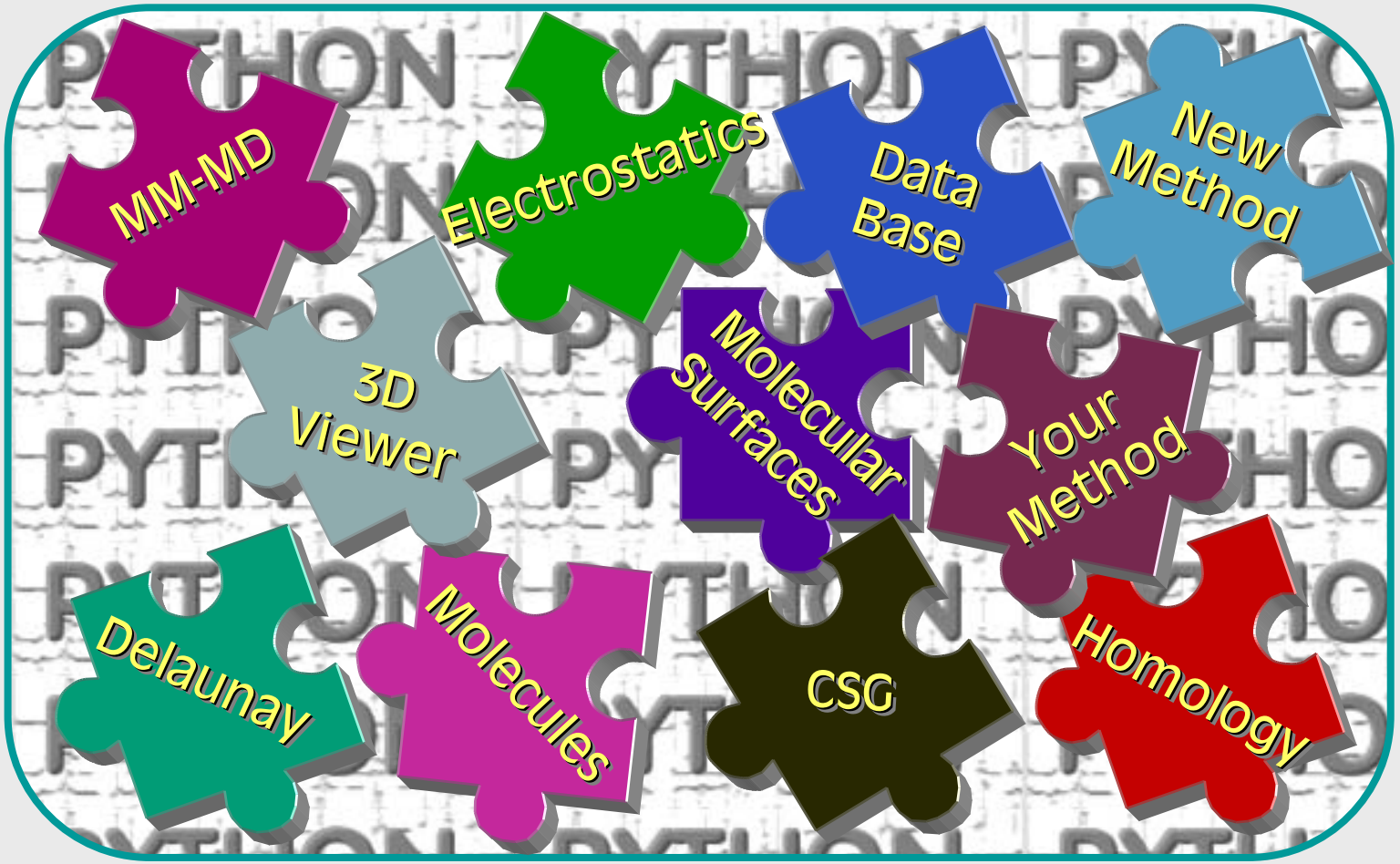
ad-hoc solution



TSRI

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

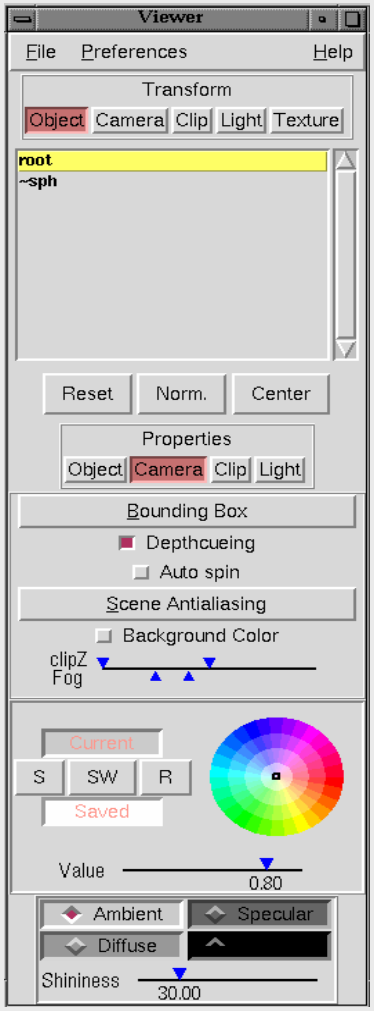
Implementation

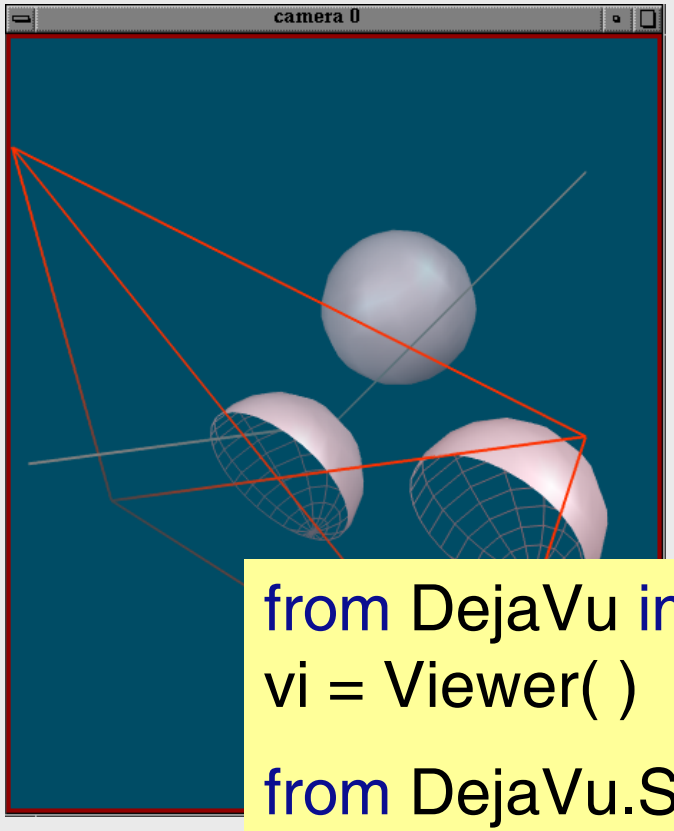
- >10 Python packages
  - > 220,000 lines of code
  - >1370 classes in Python
- >10 packages wrapping C and C++ libraries
  - > 200 classes
- 5 applications
  - Pmv, ADT, Vision, PyARTK, FlexTree
- > 2500 unit tests

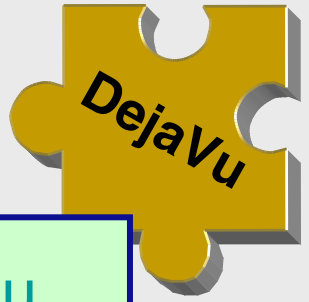
Re-usable components

- 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

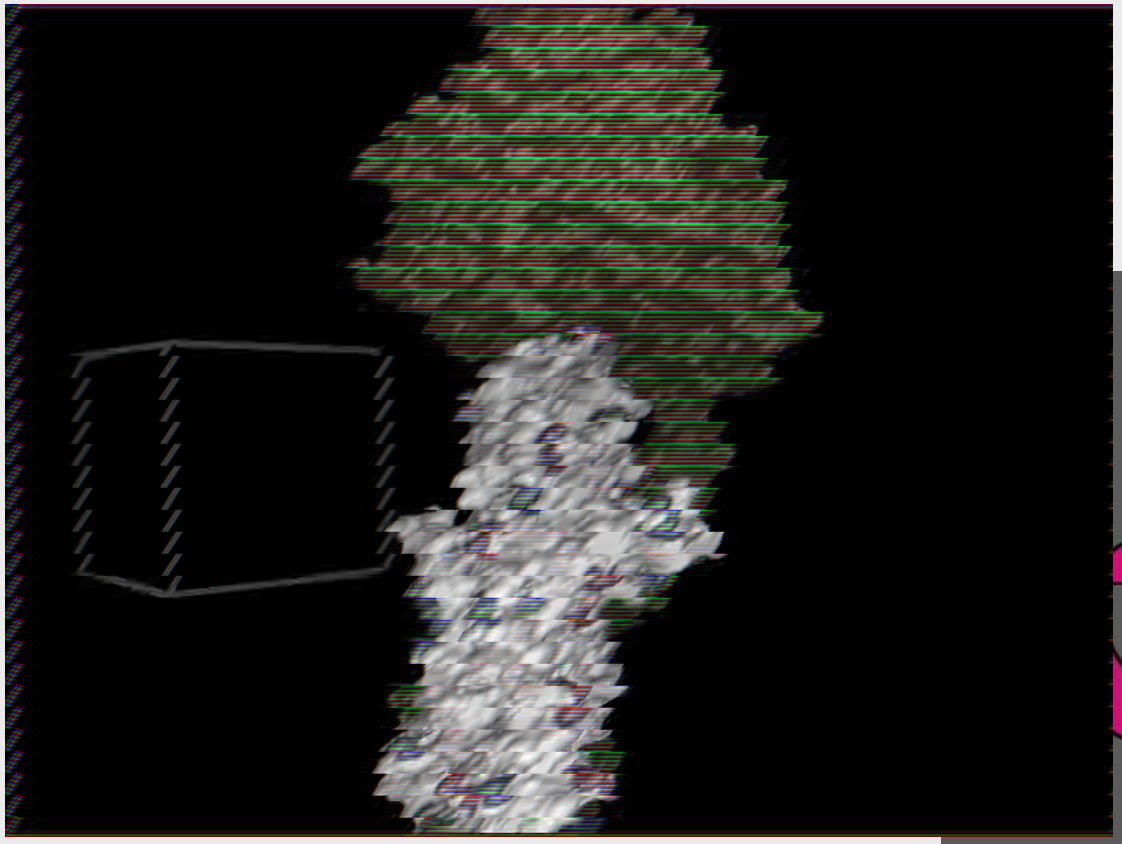
opengltk

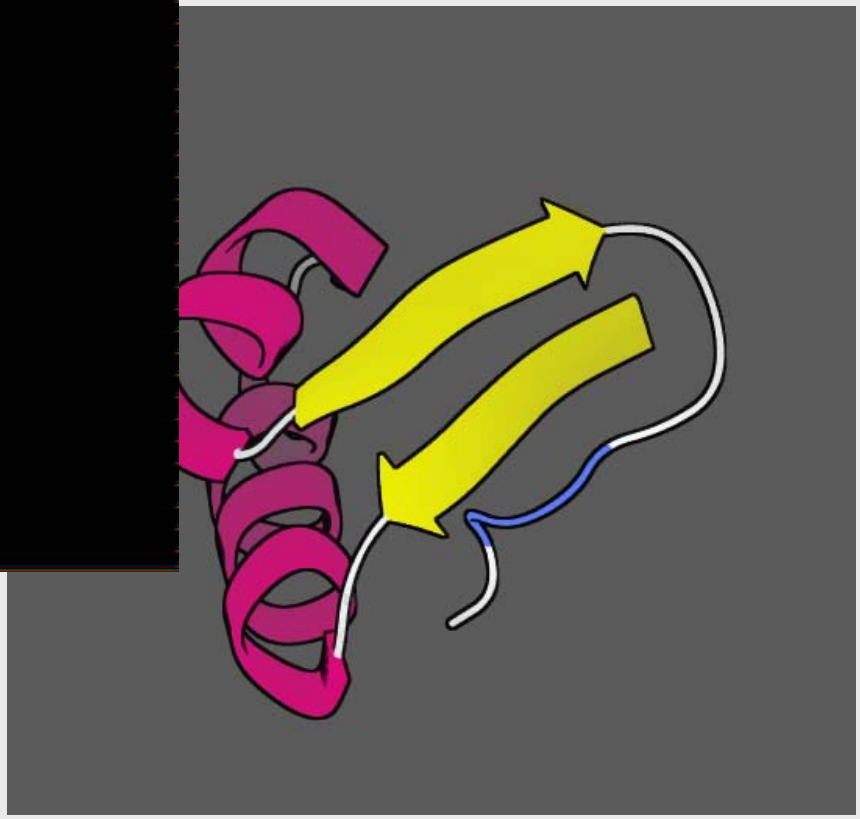
```
from DejaVu import Viewer
vi = Viewer( )

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
- glSscissors/magic lens
- Object hierarchy, Instances
- Multi-level picking
- Extensible set of geometries
- Direct volume rendering, …

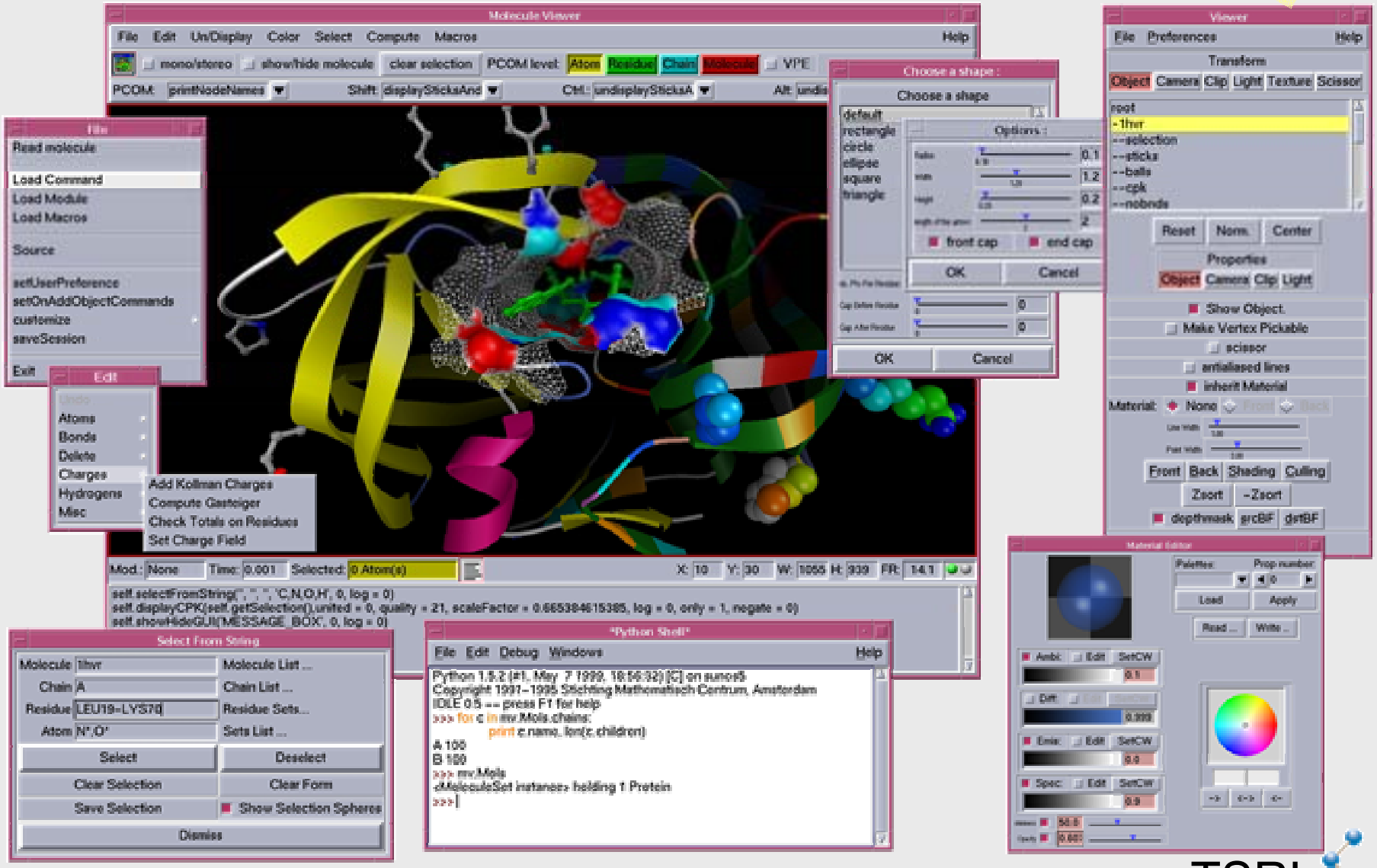
New DejaVu Features





PMV: Python Molecular Viewer

From Building Blocks to applications

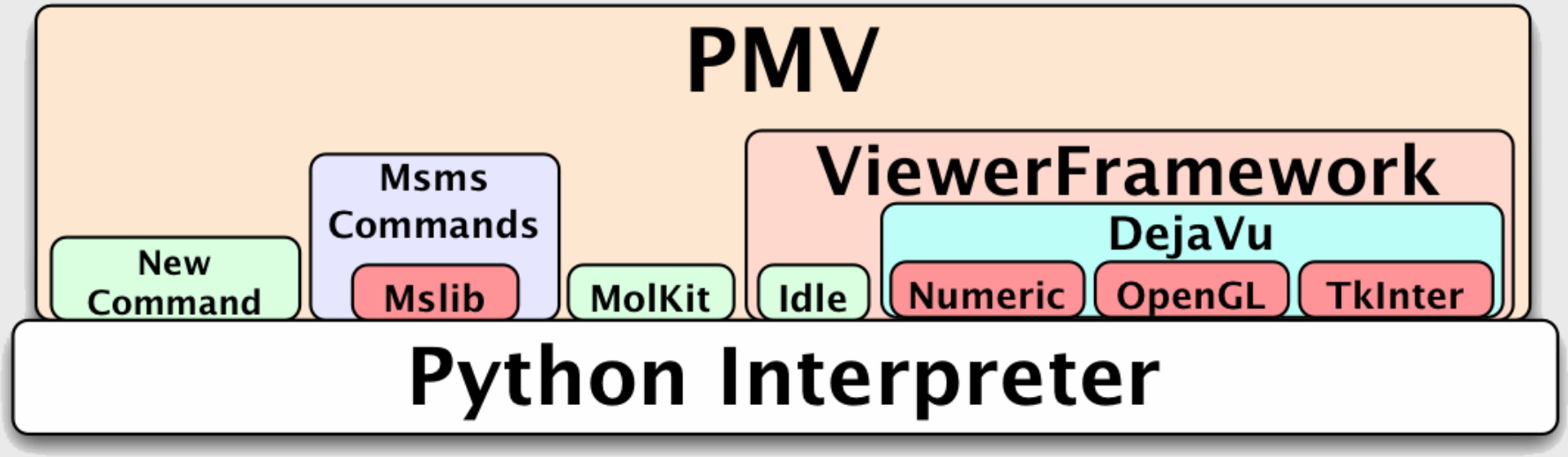


SciPy'05, Sept. 2005, Caltech, Pasadena, CA

TSRI

PMV: Architecture

Generic Molecule Viewer built from reusable components



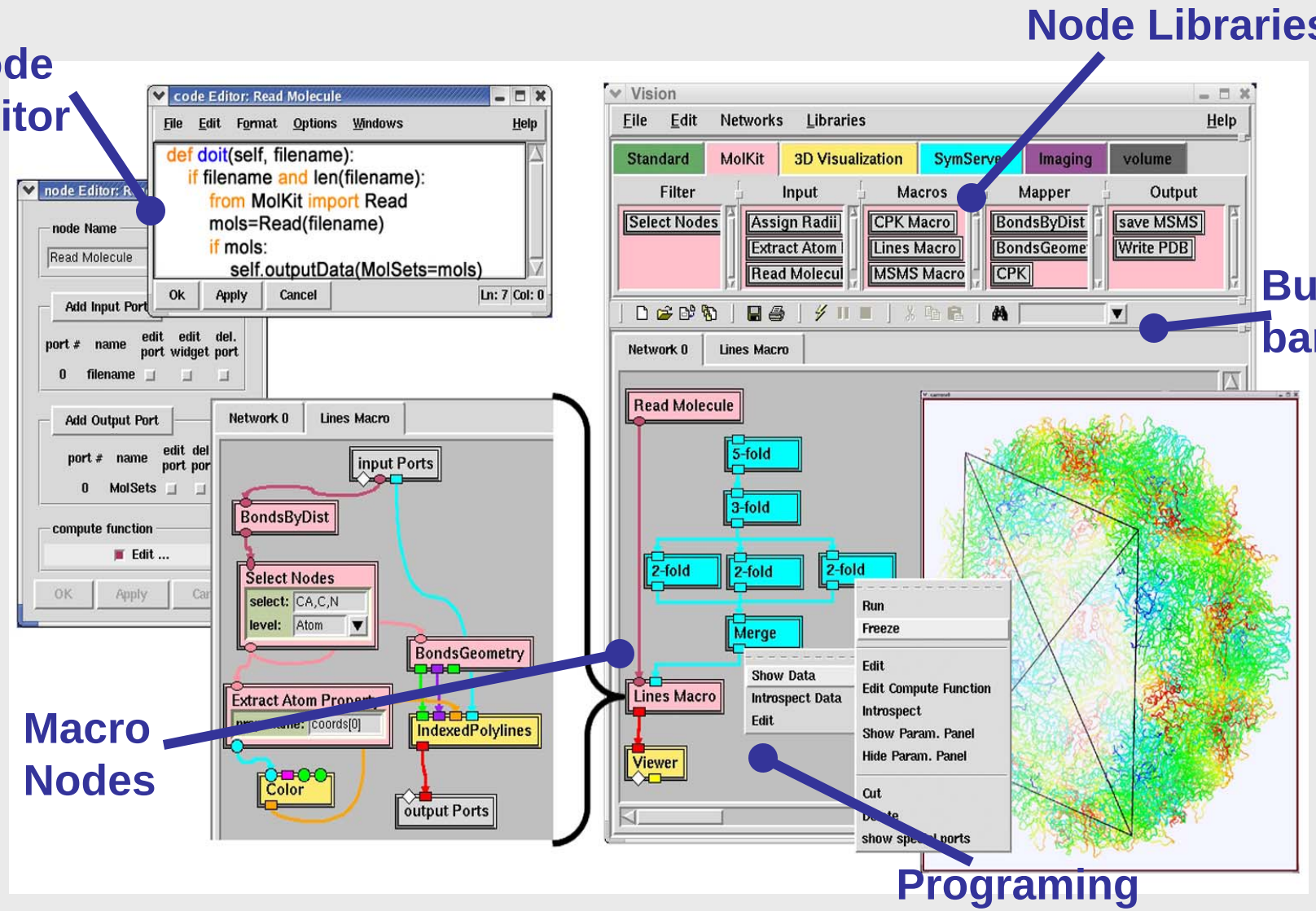
Software engineering

Interoperable, reusable, software component

Versatile, adaptive, user-programmable

Platform independent

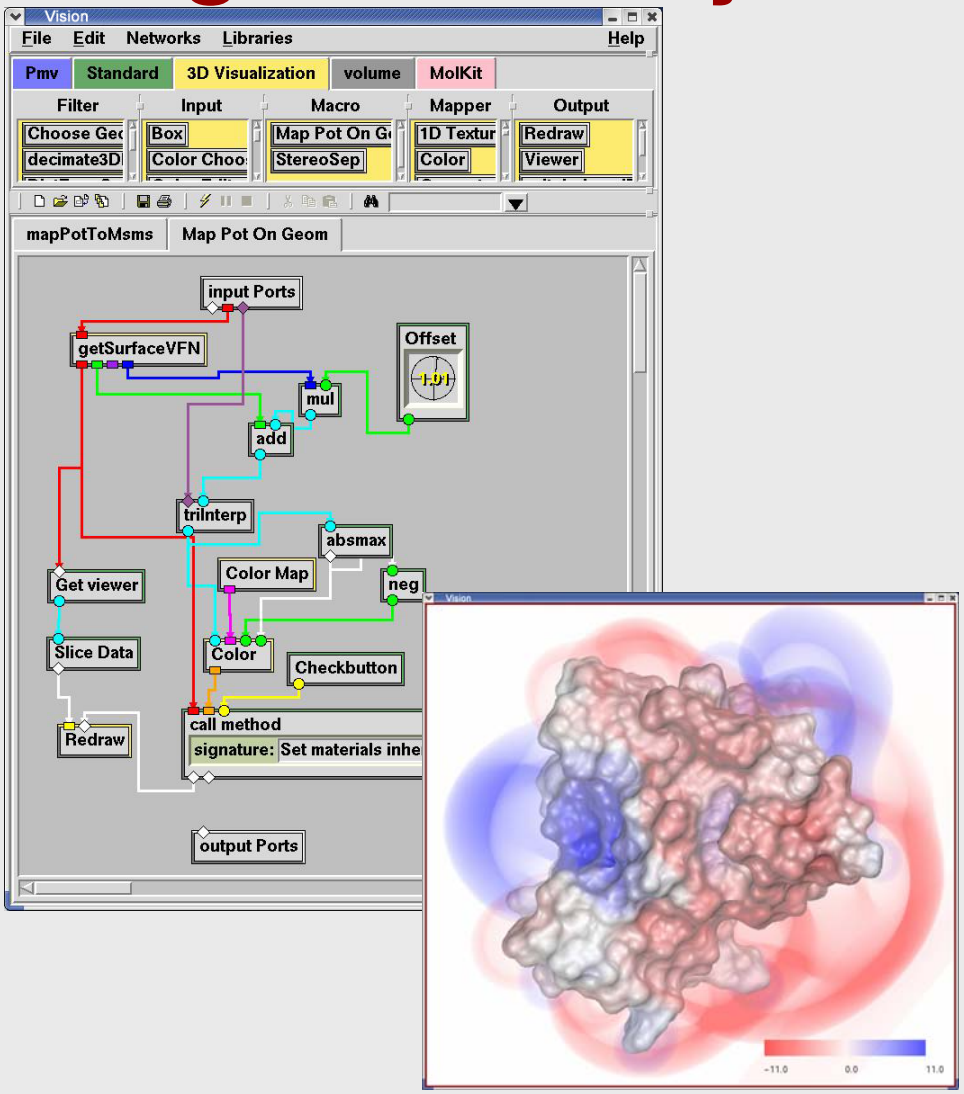
Vision: Visual Programming



TSRI

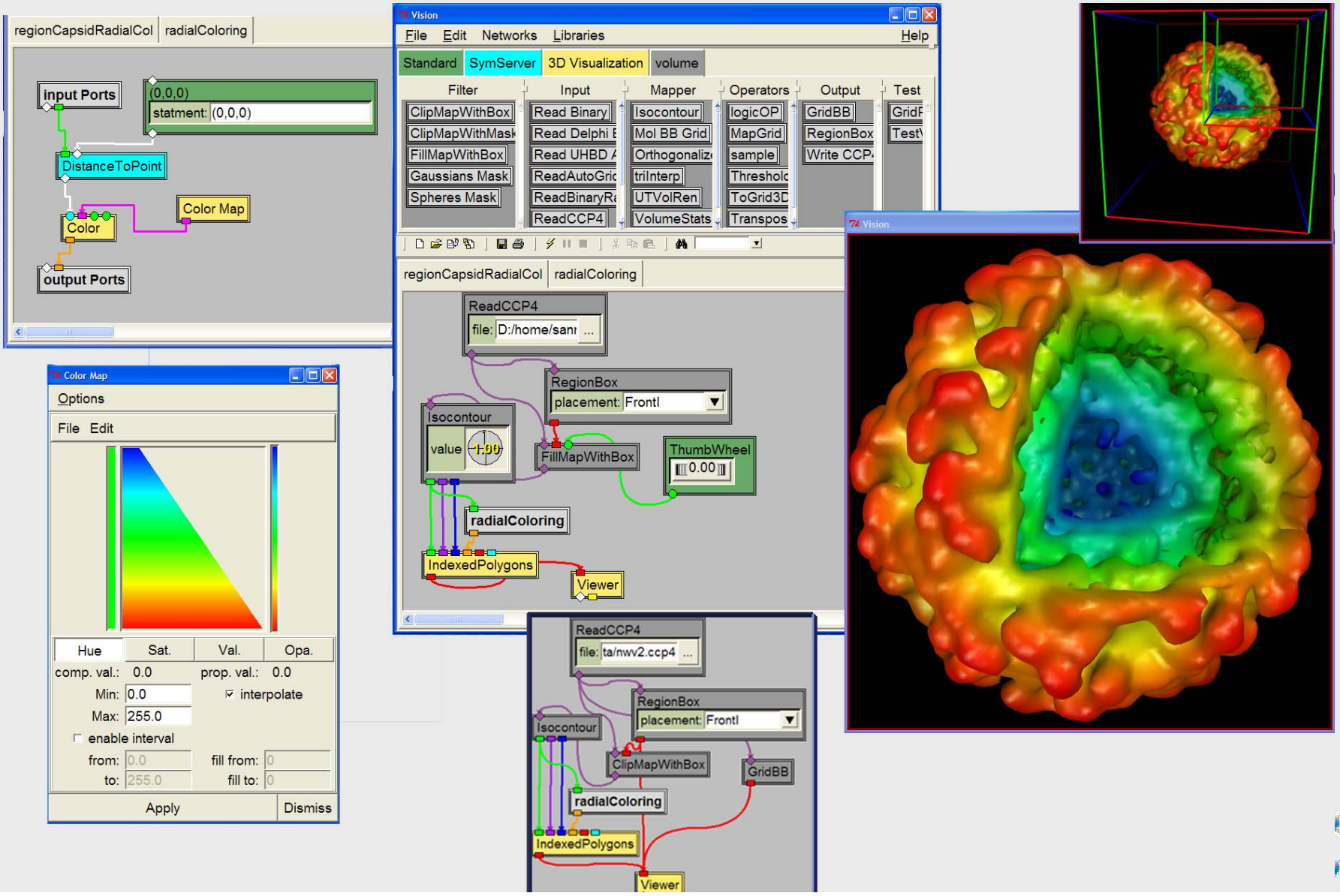
Vision: programming made easy!

- **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

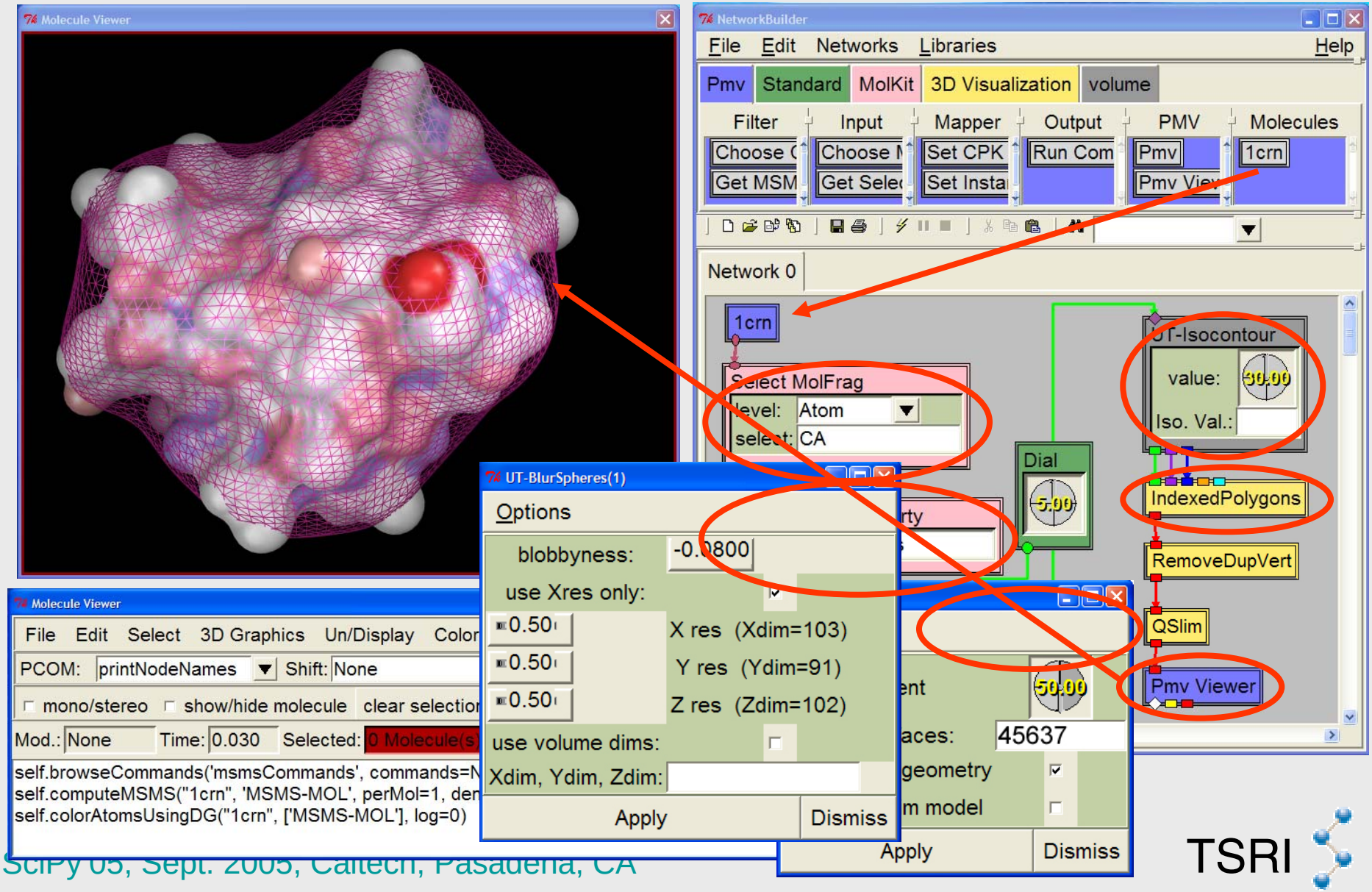


TSRI

Vision: a standalone application

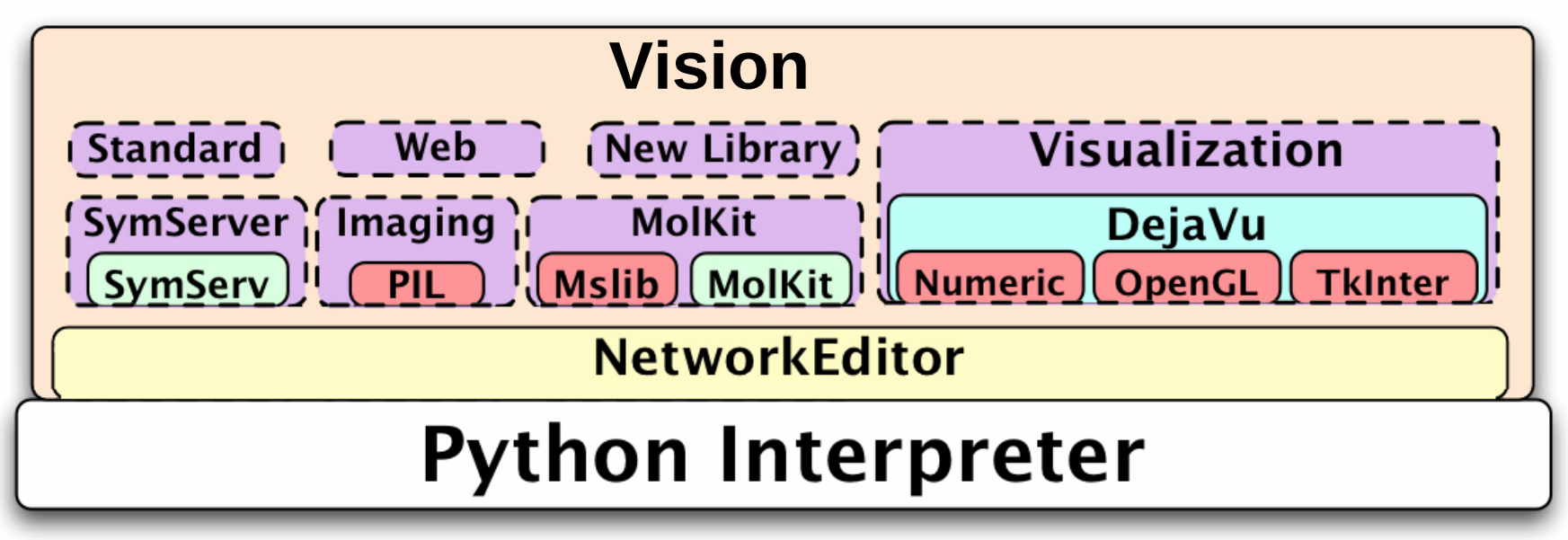


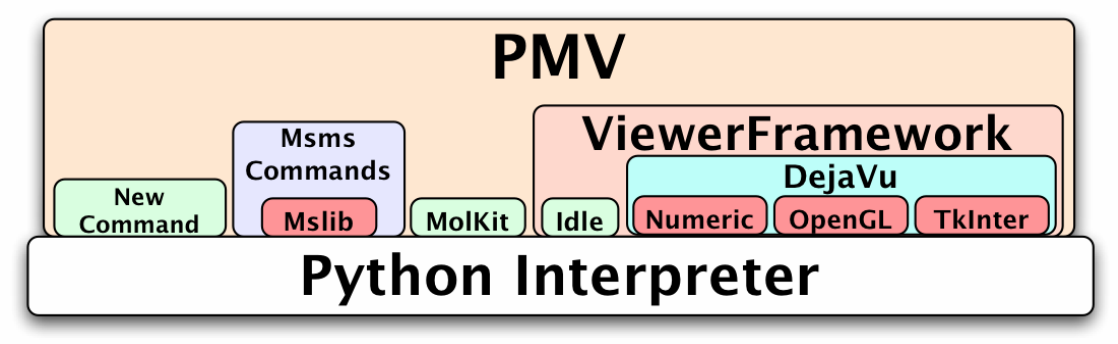
Vision for Integrating computational methods



TSRI

Vision Architecture



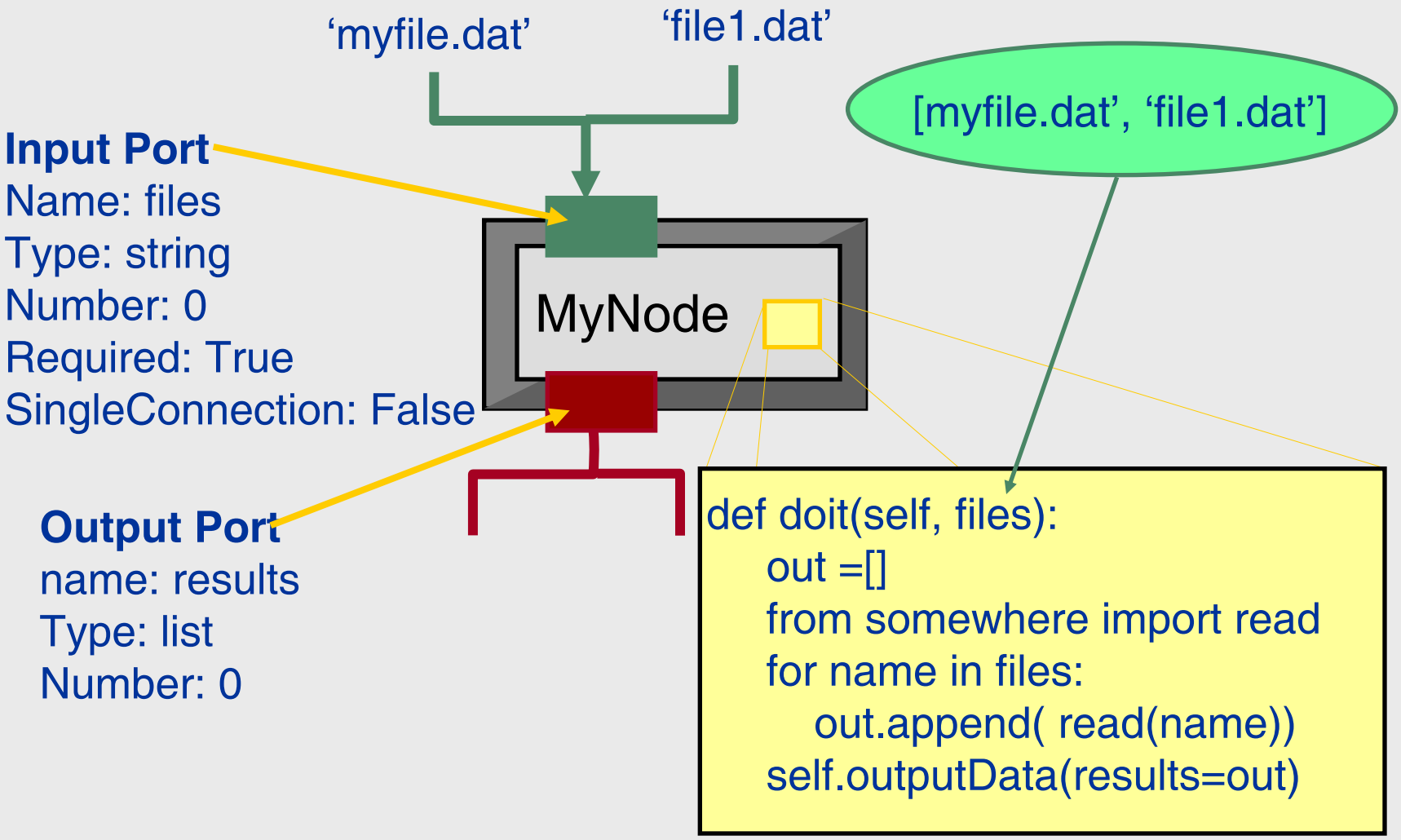


TSRI

Vision Innovations

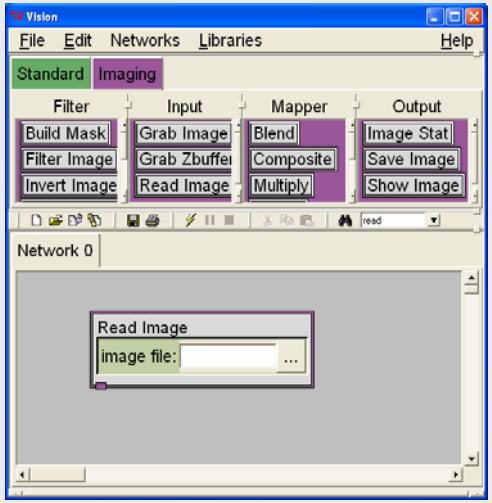
- Based on reusable components
- Dynamic: on-the-fly node editing
- Scriptable, flexible
- Platform independent
- No constraining data types or data model
- Optional data duplication
- Nodes are lightweight wrappers of computations
- Automatic detection of Vision interface in other Python packages
-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

Anatomy of a Network Node



TSRI

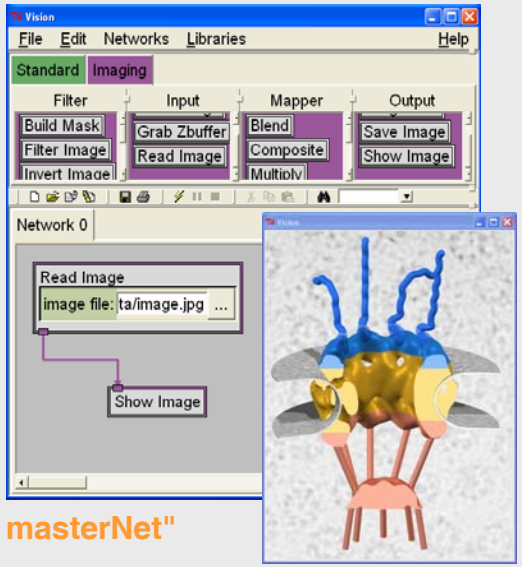
Anatomy of a Network Node




```
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)
```

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)
```



TSRI

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

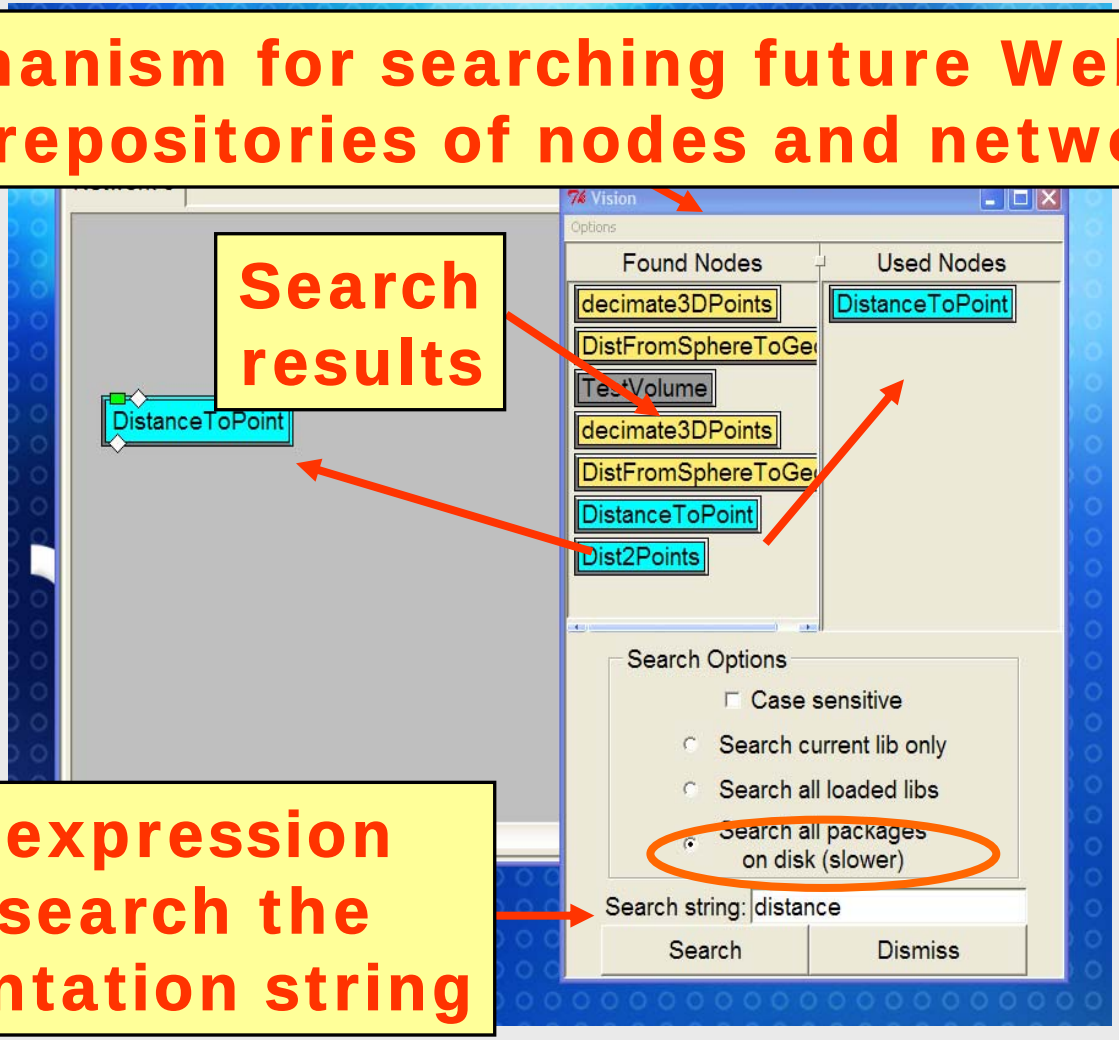
```
DejaVu/
        ...
        VisionInterface/
                DejaVuNodes.py
```

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

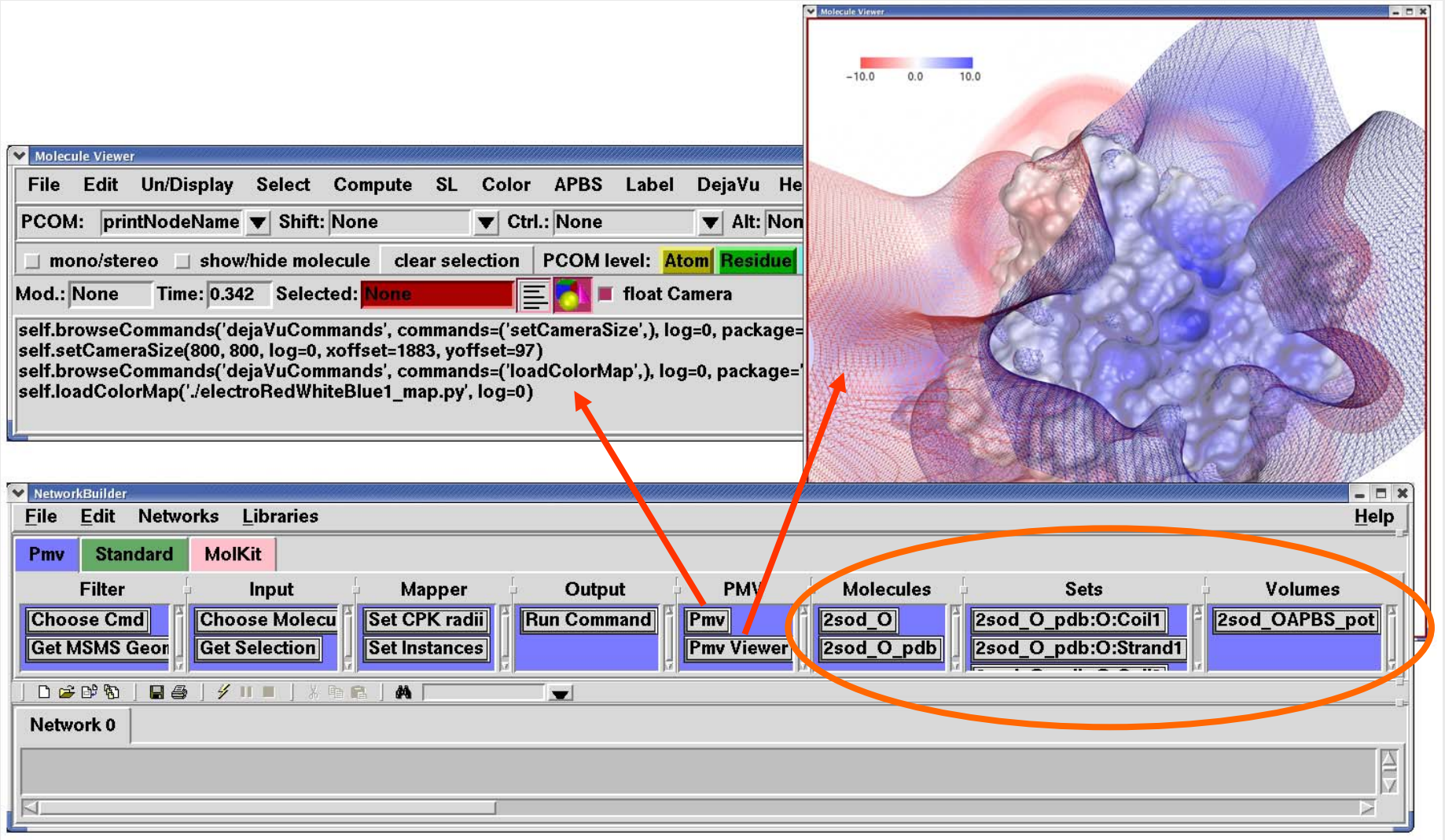
Node Discovery

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



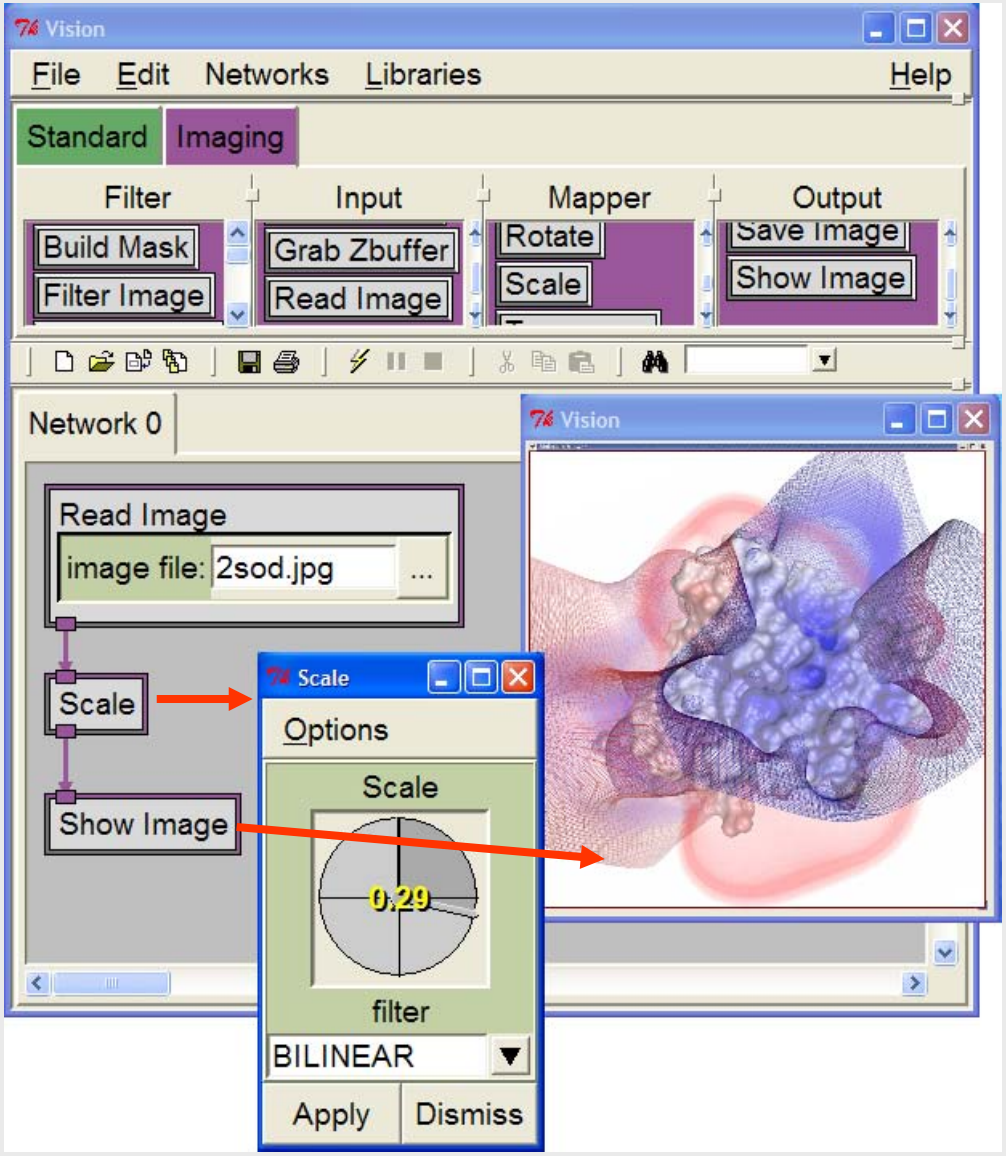
Regular expression Used to search the documentation string

Integration API

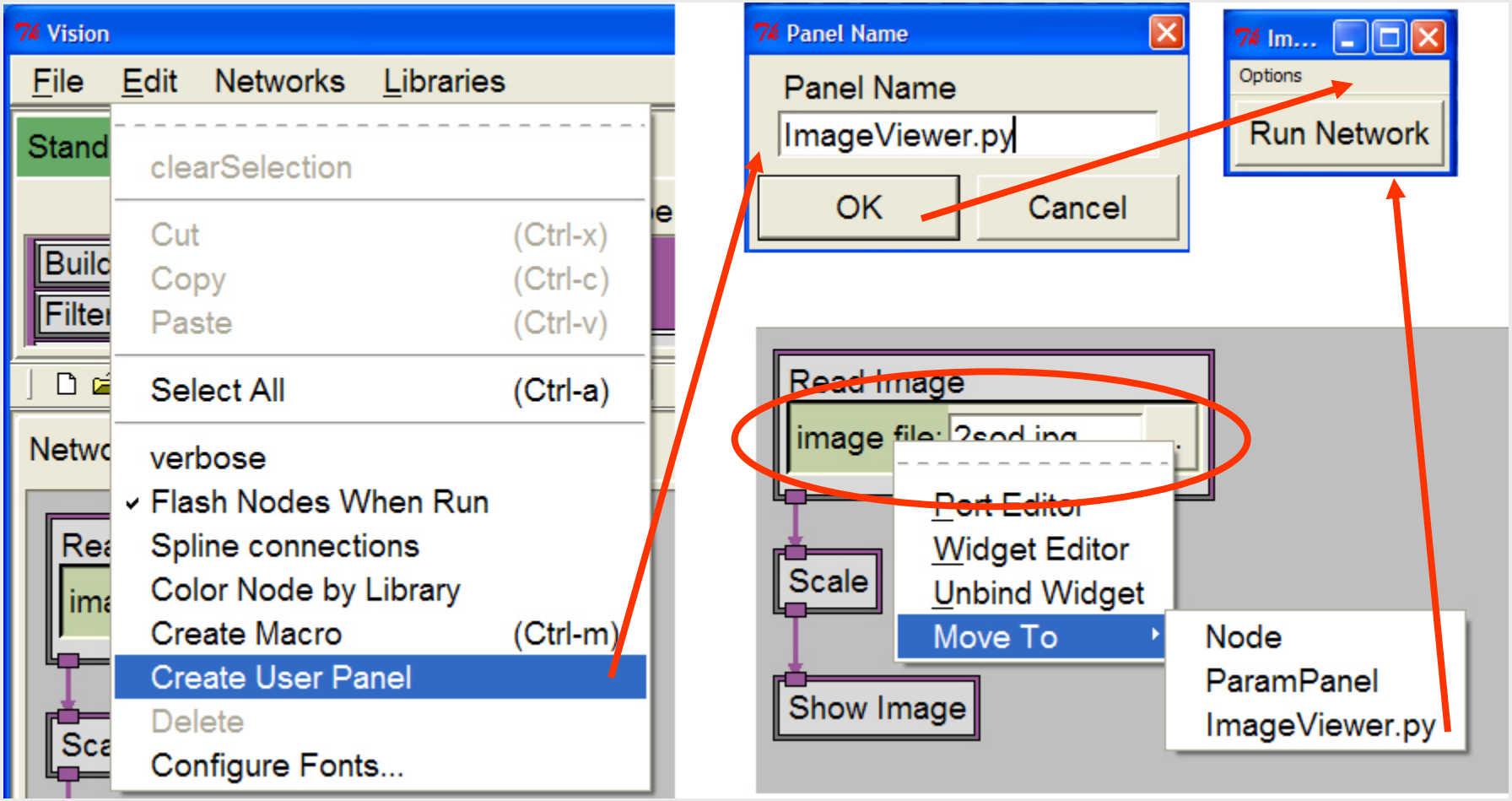


End-user applications

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

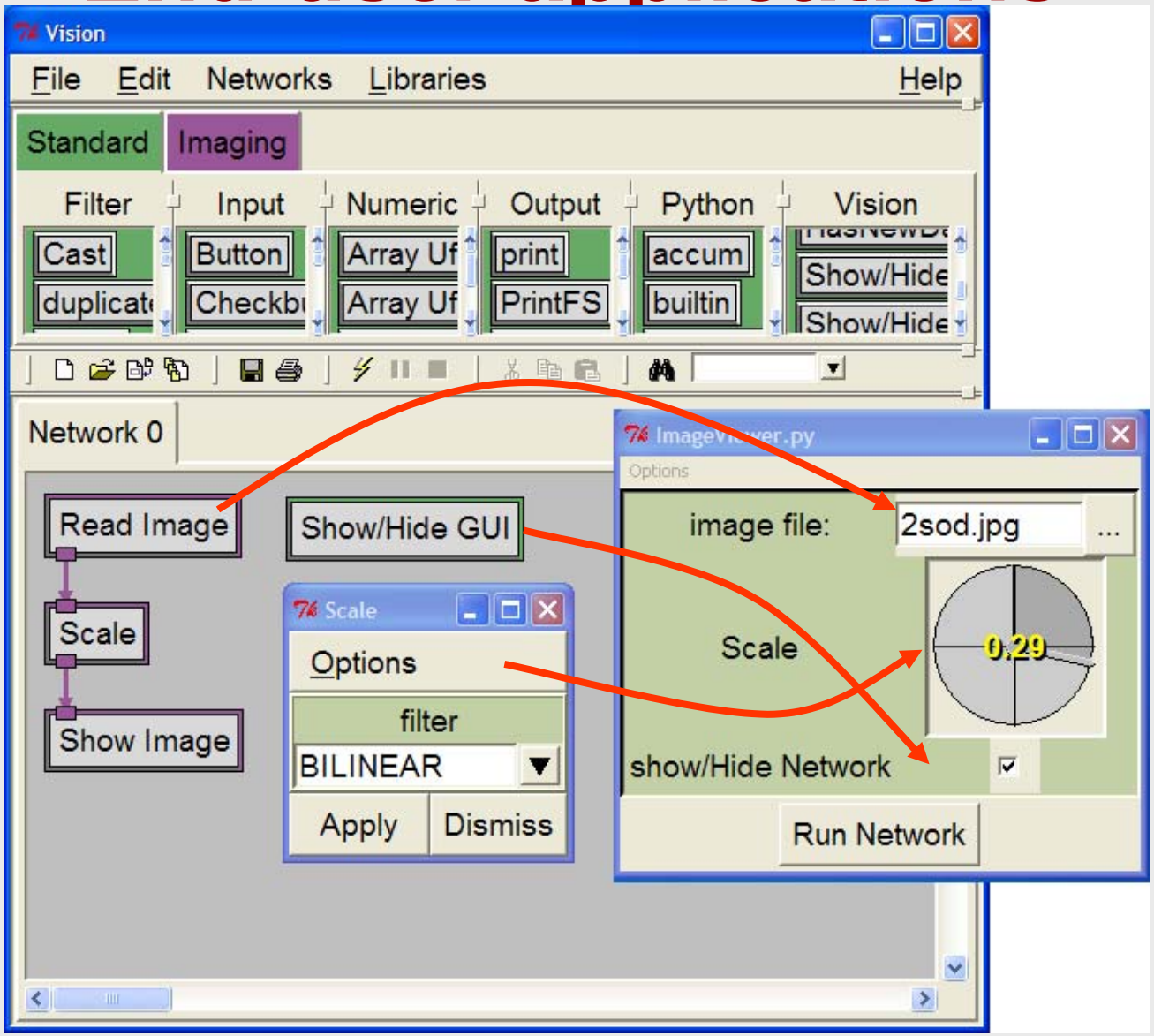


End-user applications



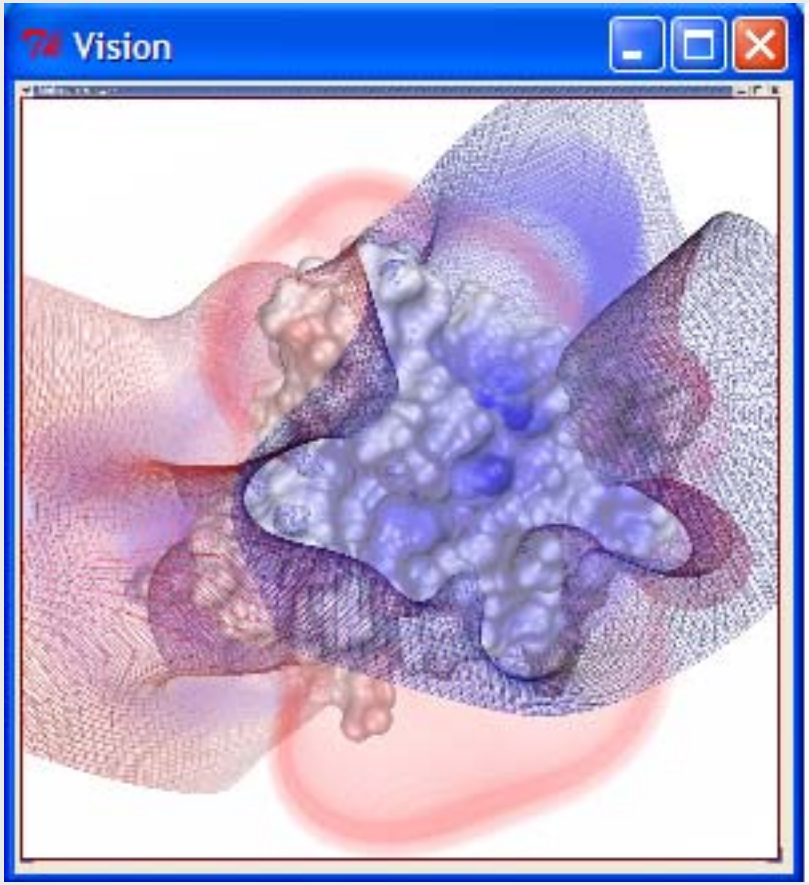
TSRI

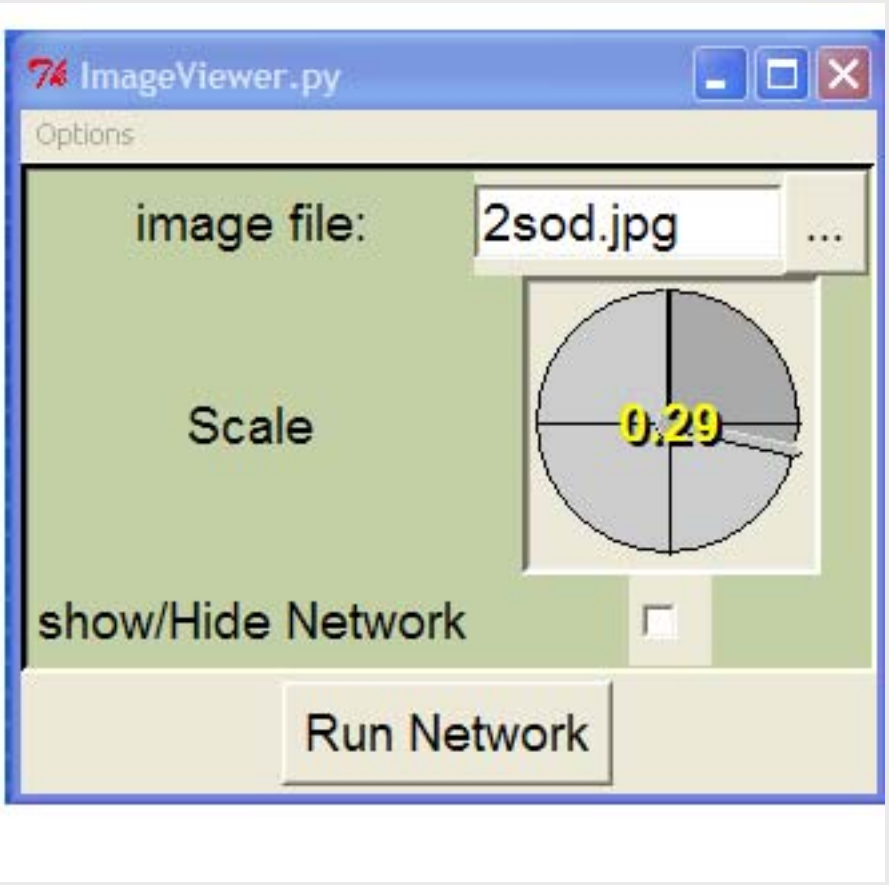
End-user applications



TSRI

End-user applications





TSRI

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