



## Re-usable components for structural bioinformatics

---



Sophie COON



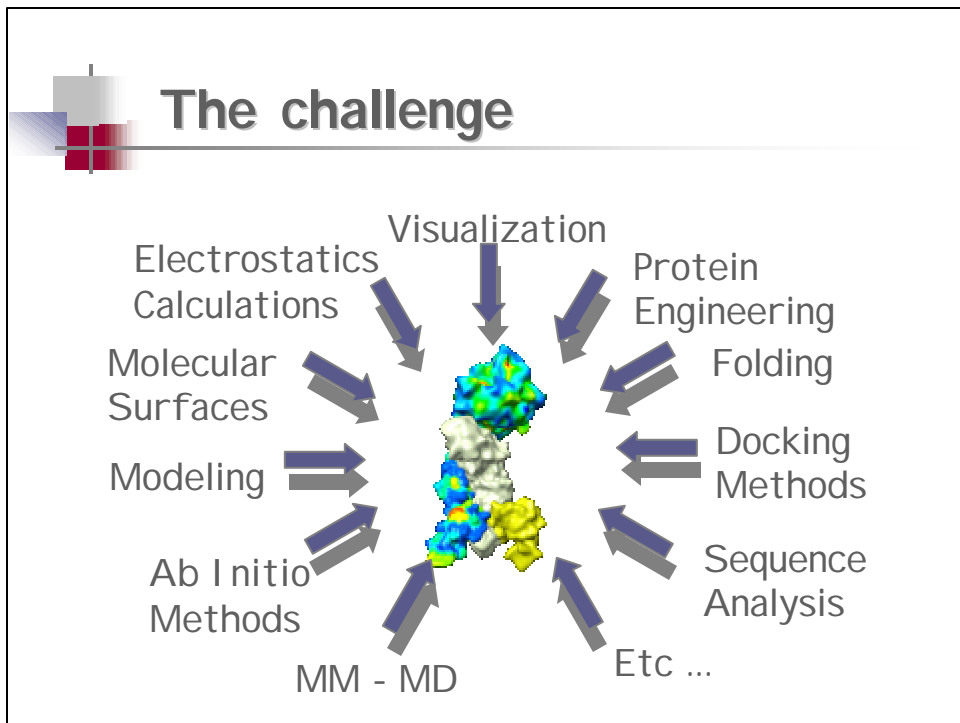
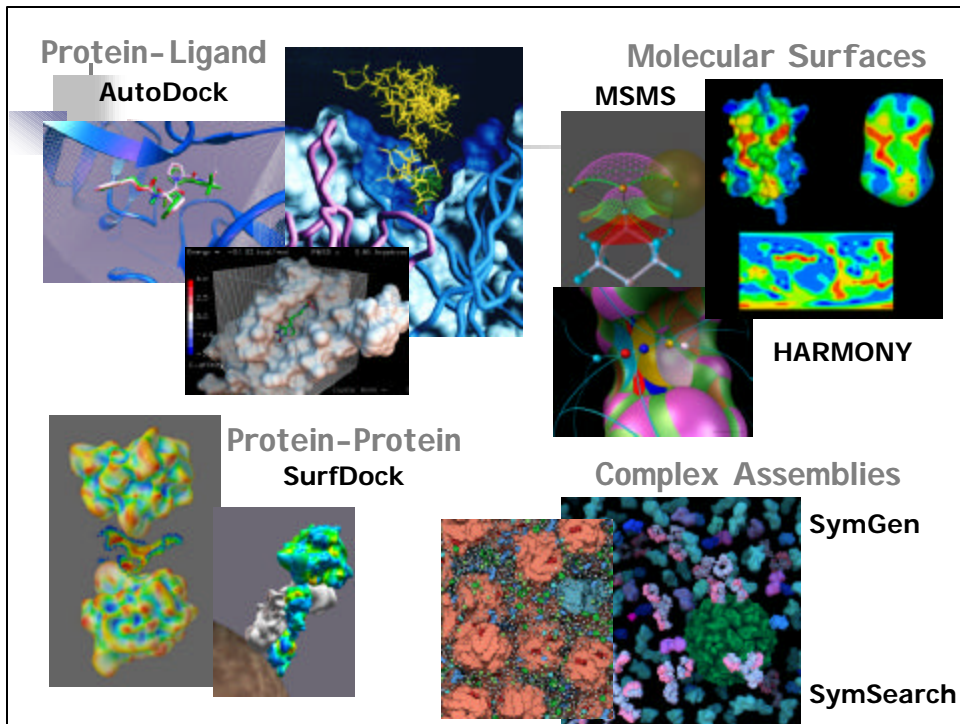
The Scripps Research Institute  
La Jolla, California

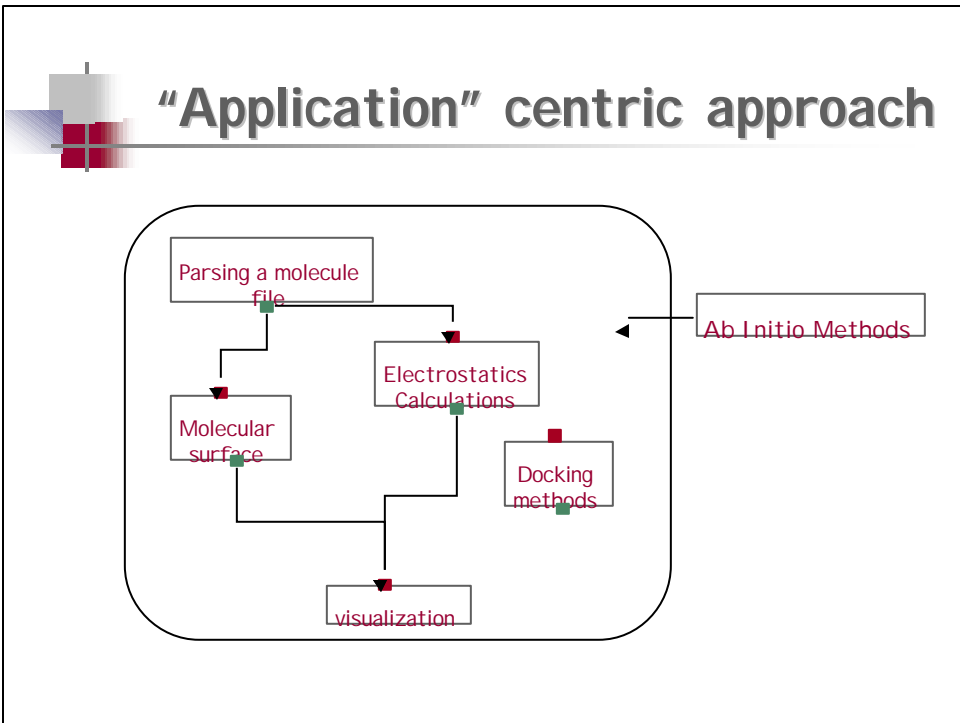
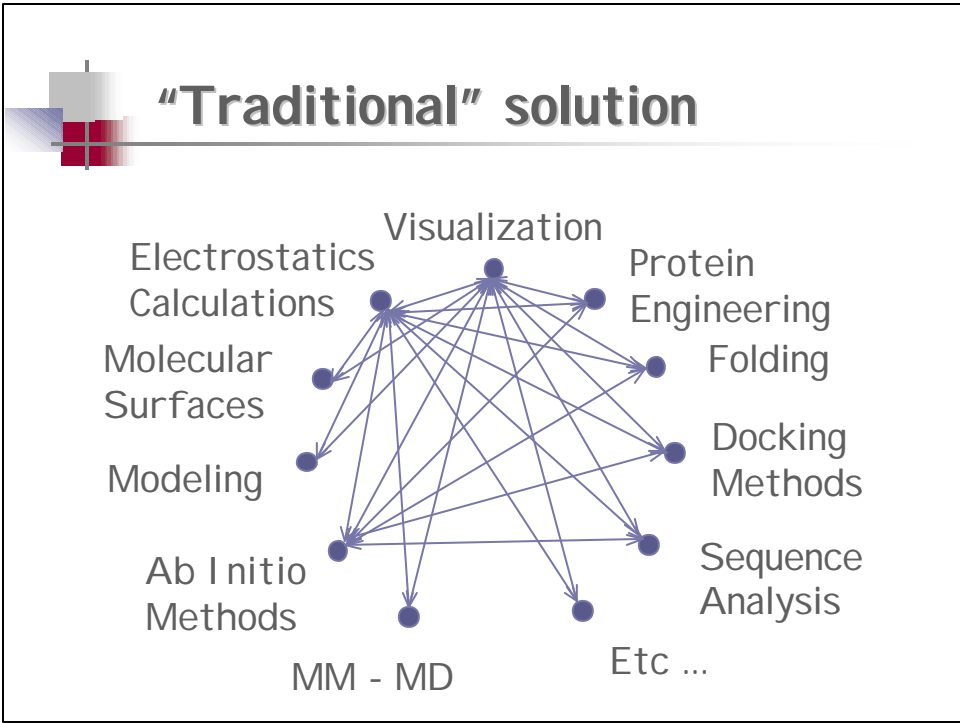


## SCHEDULE

---

- o I - Code development strategies
- o II - Re-usable components
  - ⌞ MolKit, DejaVu, ViewerFramework
- o III - From building blocks to applications
  - ⌞ PMV
- o Conclusion

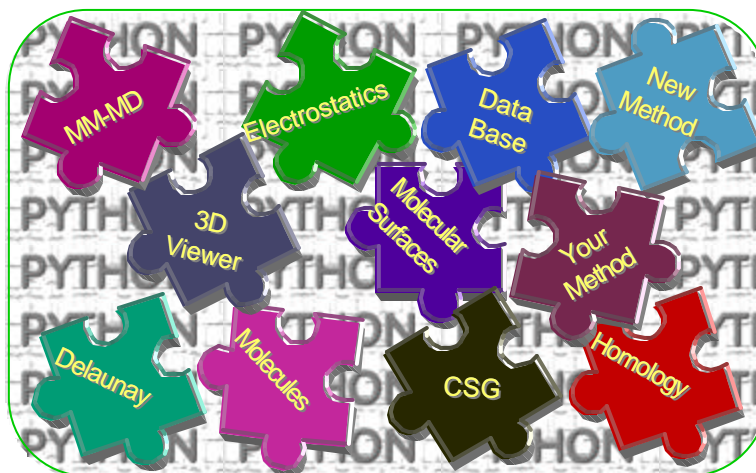




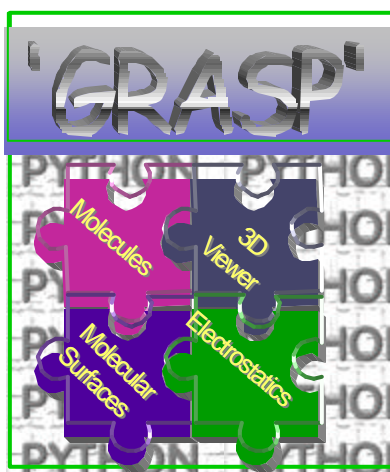


## Our solution

High level language as a scripting environment



## Writing an application

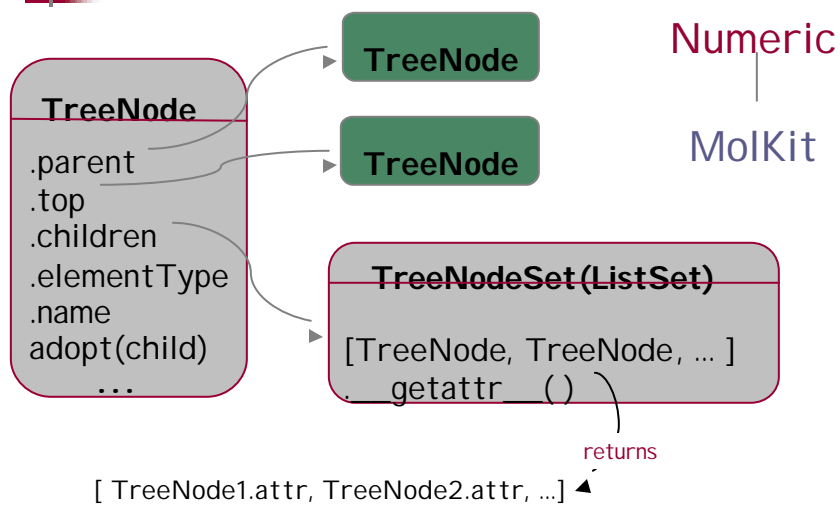


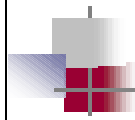
- High level coding
- Code re-use
- modularity
- Extensible

## II - Re-usable components

- o MolKit
- o DeJaVu
- o ViewerFramework

### MolKit



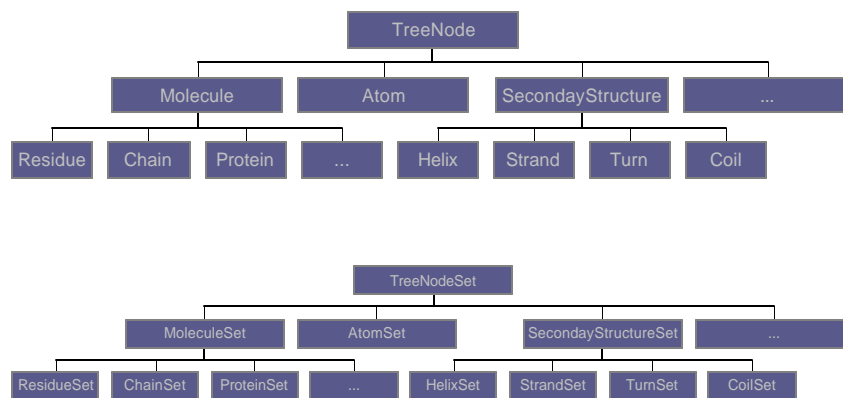


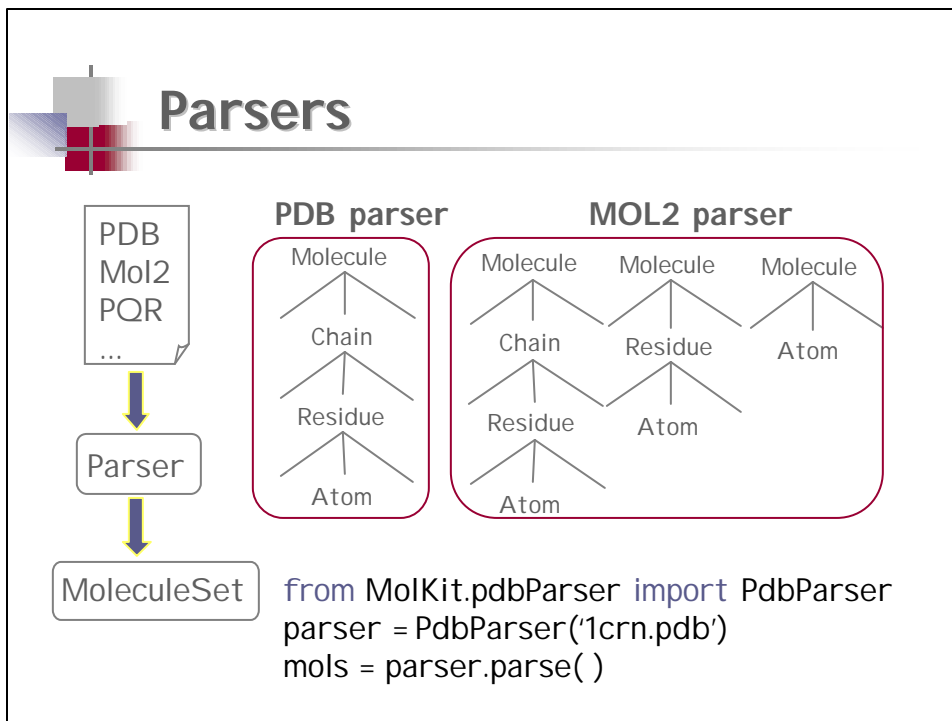
## TreeNode, TreeNodeSet

- o **TreeNodeSet:**
  - ≲ Boolean operation
  - ≲ uniq()
  - ≲ split()
  - ≲ sort()
  - ≲ NodesFromName()
  - ≲ findChildrenOfType()
  - ≲ findParentOfType()
  - ≲ ...
- o **TreeNode:**
  - ≲ adopt() / remove()
  - ≲ full\_name()
  - ≲ NodeFromName()
  - ≲ split() / merge()
  - ≲ getParentOfType()
  - ≲ findType()
  - ≲ compare()
  - ≲ assignUniqIndex()
  - ≲ isAbove() / isBelow()
  - ≲ ...



## TreeNode and TreeNodeSet specialization





## Examples

```

>>> from MolKit import Read
>>> molecules = Read('./1crn.pdb')
>>> mol = molecules[0]           # Read returns a ProteinSet

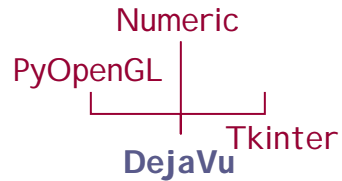
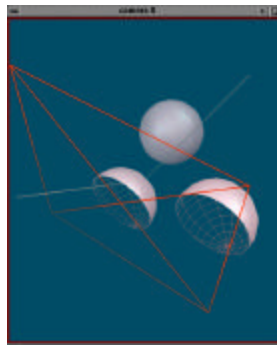
>>> print mol.chains.residues.name
>>> print mol.chains.residues.atoms[20:85].full_name()

>>> from MolKit.molecule import Atom
>>> allAtoms = mol.findType(Atom)
>>> set1 = allAtoms.get(lambda x: x.temperatureFactor > 20)

>>> allResidues = allAtoms.parent.uniq()
>>> import Numeric
>>> for r in allResidues:
...     coords = r.atoms.coords
...     r.geomCenter = Numeric.sum(coords) / len(coords)

```

# DejaVu



```
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)
```

# Features

- o OpenGL Lighting and Material model
- o Object hierarchy with transformation and rendering properties inheritance
- o Arbitrary clipping planes
- o Material editor
- o DepthCueing (fog), global anti-aliasing
- o glScissors/magic lens
- o Multi-level picking
- o Extensible set of geometries



## Geometries

### Geom

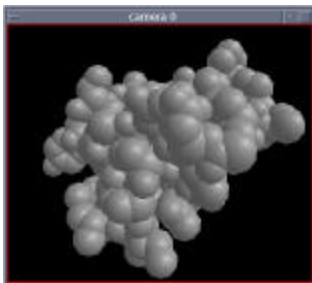
- o PolyLine
- o Points
- o Spheres
- o Labels
- o Arc3D...

### IndexedGeoms

- o IndexedPolyLines
- o IndexedPolygons
- o Triangle\_Strip
- o Quad\_Strip
- o Cylinders ....

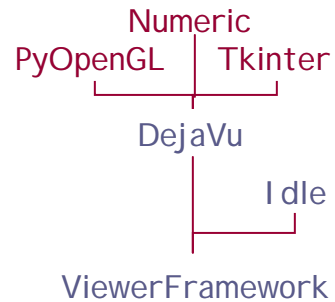
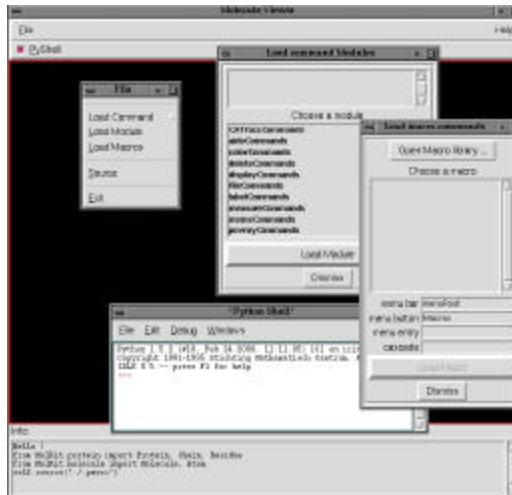


## DejaVu and MolKit



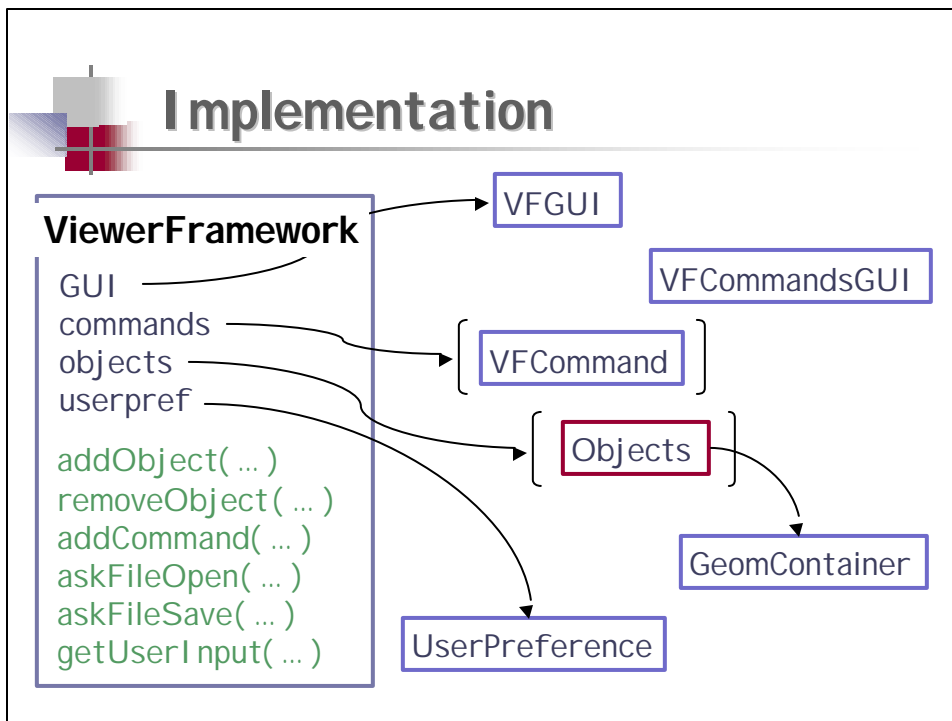
```
>>> from MolKit import Read
>>> molecules = Read('./1crn.pdb')
>>> mol = molecules[0] # Read returns a ProteinSet
>>> coords = mol.chains.residues.atoms.coords
>>> radii = mol.defaultRadii()
>>> from DejaVu.Spheres import Spheres
>>> from DejaVu import Viewer
>>> vi = Viewer()
>>> sph = Spheres('sph', centers = coords, radii = radii, quality=10)
>>> vi.AddObject(sph)
```

## ViewerFramework



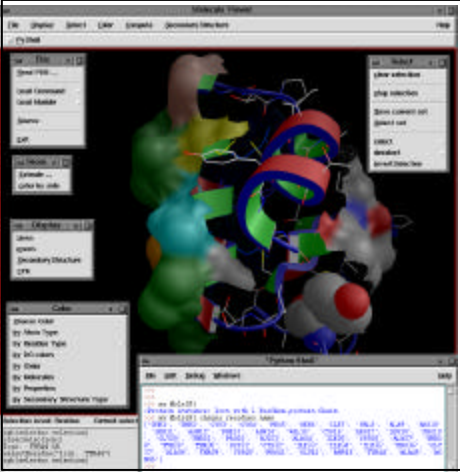
## Design features

- Dynamic loading of commands
- Python shell for scripting
- Dual interaction mode (GUI /Shell)
- Support for command:
  - ⚡ development, logging, GUI , dependencies
- Lightweight commands: Macros
- Dynamic commands (introspection)
- Extensible set of commands



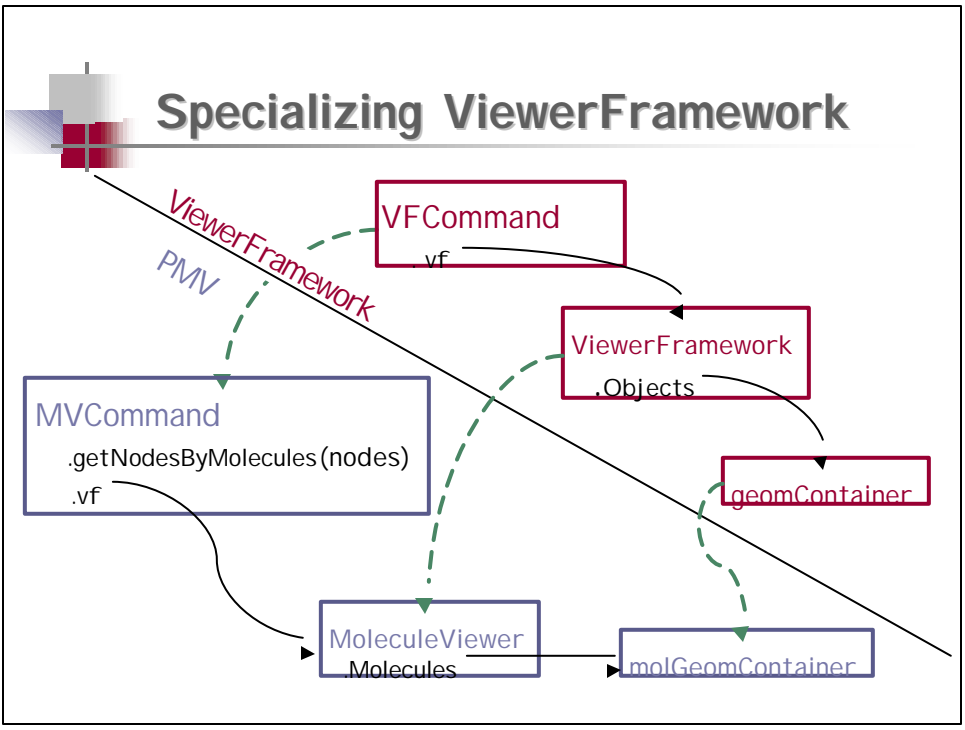
- ## III - From Building Blocks to applications
- o PMV: Python molecular viewer

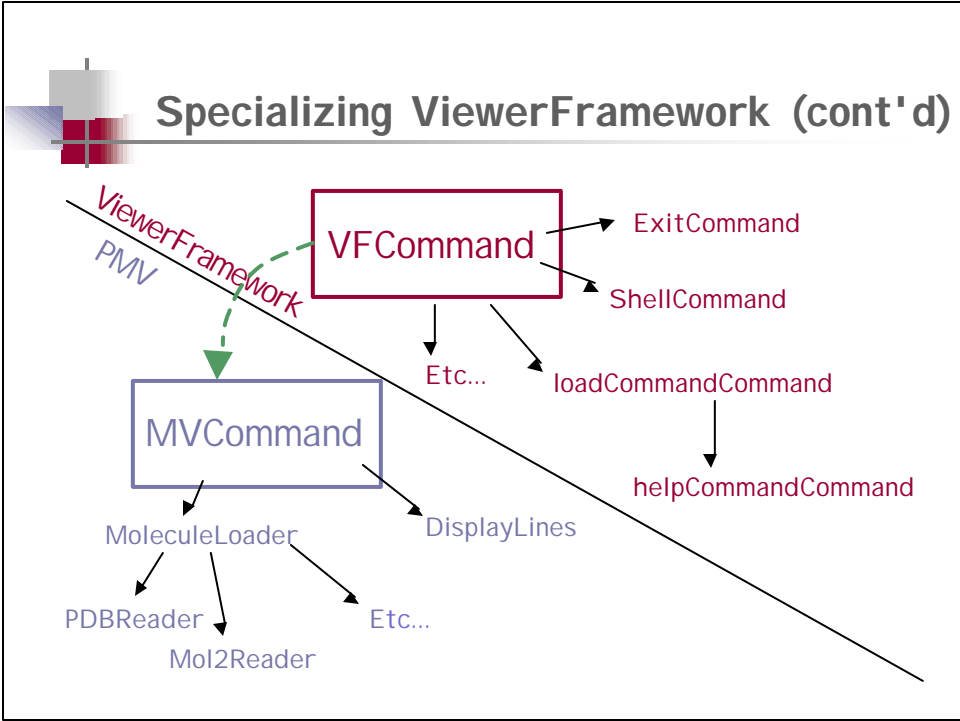
# PMV



```

    Numeric
    PyOpenGL | Tkinter
    -----|-----
    DeJaVu
    |
    Idle
    |
    ViewerFramework
    |
    Pynche | MolKit
    -----|-----
    Pmv
  
```





## PMV

```

      Numeric
      PyOpenGL | Tkinter
      DeJaVu
      | Idle
      ViewerFramework
      Pynche | MolKit
      -----
      Pmv
    
```

**DEMO**



## Conclusion

---

- Validity of the approach
- Availability
- Future directions



## Validity of the approach

---

- Set of components
  - ⌞ extensible
  - ⌞ inter-operable
  - ⌞ **re-usable**
  - ⌞ short development cycle
- User base expanding beyond our lab.
- Components re-use outside the field of structural biology



It Works!



## Availability

---

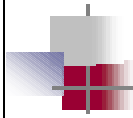
- Modularity enables fine grain licensing schemes (a la carte)
- Core modules are freely available
- Online Download site:  
<http://www.scripps.edu/~sanner/Python>



## Future directions

---

- Add support for editing molecular structures (i.e. mutations, deletion, addition)
- Interface with MMTK, Tinker, ...
- Enhance documentation and tutorials
- Setup a CVS server for programmers wanting to help !
- Too many to list ...



## Acknowledgments

- Michel Sanner
- Christian Carrillo, Kevin Chan
- Ruth Huey, Fariba Fana
- Vincenzo Tchinke, Greg Paris
- MGL at TSRI
- Pat Walters, Matt Stahl
- Don Bashford
- Guido van Rossum & Python community