Biological Science Libraries
- An approach to software components for structural bioinformatics

Michel F. Sanner

The Molecular Graphics Laboratory

The Scripps Research Institute
La Jolla, California

SciPy’02
CalTech Pasadena CA. Sept. 2002
Component-based software
  - Background and motivation
  - Re-usable components

From building blocks to applications
  - PMV: a molecular visualization program
  - ADT: setting up and running AutoDock
  - PVV: interactive volume rendering
  - ViPEr: a visual programming environment

Conclusion
Protein-Ligand Complex Assemblies

AutoDock

Molecular Surfaces

MSMS

HARMONY

Protein-Protein Complex Assemblies

SurfDock

SymGen

SymSearch
Tangible Models
ARToolkit 3D tracking

1. Video stream from camera
2. Search for markers
3. Find marker 3D position and orientation
4. Identify markers
5. Virtual objects are rendered in video frame
6. Render 3D objects in video frame
7. Position and orient objects
8. Virtual objects
9. IDs of markers

SciPy’02, © M. Sanner, TSRI
Augmented reality
The challenge

- Electrostatics
- Calculations
- Molecular Surfaces
- Modeling
- Ab Initio Methods
- MM - MD

- Visualization

- Protein Engineering
- Docking Methods
- Sequence Analysis
- Etc ...

SciPy’02, © M.Sanner, TSRI
“Traditional” solution

Electrostatics Calculations
Molecular Surfaces
Modeling
Ab Initio Methods

Visualization
MM - MD

N² interfaces
low interoperability
No code Reuse

Protein Engineering
Folding
Docking Methods
Sequence Analysis
Etc ...

SciPy’02, © M.Sanner, TSRI
Python to the rescue

High level language as a scripting environment

Interactive
Dynamic
Platform independent

MM-MD
Electrostatics
Data Base
3D Viewer
Molecular Surfaces
Your Method
Delaunay
Molecules
CSG
Homology

SciPy'02, © M.Sanner, TSRI
Writing an application

- High level coding
- Code re-use
- Modularity
- Extensible
PYTHON - Features

- Interpreted, High level, Object-Oriented
- Flexible and Extensible
- Introspection, self-documenting
- Platform Independent
- Open Source
- Rapidly gaining acceptance
Why Python?

- Advanced data structures (> Tcl, Perl)
- Object-Oriented AND scripting (> Java)
- "Batteries included!"
  - Powerful data-parallel Numeric arrays (> Tcl)
- Readability and modularity (> Perl)
- High-level (> C, C++, Fortran)
- Platform independence (> C++, Java)
- Efficient enough for many things!
- Automated tools for wrapping C, C++, Fortran
Re-usable components

- **MolKit**: read/write/represent/manipulate molecules
- **PyBabel**: Atom hybridization, charges, hydrogen atoms
- **DejaVu**: General purpose 3D geometry viewer
- **ViewerFramework**: Visualization application template
- **Mslib, PyBabel, PyMead, SFF, Isocontour, ...**

(Sophie I. Coon, Michel F. Sanner and Art J. Olson, Re-usable components for structural bioinformatic, 9th Python Conference 2001)
```python
from MolKit.pdbParser import PdbParser
parser = PdbParser('1crn.pdb')
mols = parser.parse()
```
Examples

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

- From Babel v1.6 (mostly)
  - Atom types and bond order detection
  - Aromaticity detection
  - Adding Hydrogen atoms
  - Computing Gasteiger charges
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

- 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
from MolKit import Read
molecules = Read('./1crn.pdb')
coords = molecules.chains.residues.atoms.coords
radii = mol.defaultRadii()

from DejaVu import Viewer
vi = Viewer()  # create a viewer

from DejaVu.Spheres import Spheres
s=Spheres('sph', centers=coords, radii=radii,quality=10)
vi.AddObject(s)  # display atomic spheres
ViewerFramework

- Numeric
- PyOpenGL
- Tkinter
- DejaVu
- Idle
- ViewerFramework

SciPy’02, © M.Sanner, TSRI
PMV: From Building Blocks to applications
PMV Capabilities

- **Display/compute:**
  - Lines, CPK, Stick and Balls, Surfaces, Splines, Ribbons, H-bonds, bond-order, Gasteiger and Kollmann charges

- **Editing:**
  - deleting atoms, adding hydrogens

- **Electrostatic potential**
  - MEAD*, APBS*

- **Molecular Dynamics:**
  - Amber*
PMV: Architecture

PMV

Python Interpreter

Msms Commands
Mslib
MolKit
Idle

ViewerFramework
DejaVu

Numeric
PyOpenGL
Tkinter
AutoDock fundamentals

Automated docking of a flexible ligand to macromolecules using affinity grids
AutoDock ToolKit (ADT)

- **AutoTors**: ligand preparation
- **AutoGpf**: grid definition
- **AutoDpf**: docking parameters definition
- **AutoStart**: job launching and monitoring
- **AutoAnalyze**: docking results analysis
ADT: extending PMV
ADT: Architecture

ADT
AutoTors, AutoGpf
AutoDpf, AutoLaunch
AutoAnalyze

PMV
ViewerFramework
DejaVu

Python Interpreter
Msms
Commands
MolKit
Idle
Numeric
PyOpenGL
Tkinter
Mslib

SciPy’02, © M.Sanner, TSRI
Interactive Volume Rendering

VolumePro™ 500 Key Features

- Renders a $256^3$ volume up to 30 frames per second
- 256MB of volume memory
- Real-time parameter changes
- Full phong based lighting model on per sample basis
- (8, 8, 8, 12) bit voxels
- One oblique cut plane with thickness
- Six box cropping planes
- 4k color lookup table with high precision alpha blending
- Support for anisotropic and gantry tilted volume data
VolPro and DejaVu

- **VLI**
  - C++ library driving the VolPro card
  - Wrapped for Python using SWIG

- **VLIGeom**
  - Subclasses DejaVu.Geom
  - Ties the Geometry transformation to the VLI Camera position
  - Display the result of a volume rendering operation in the DejaVu OpenGL context
VV component

Set of ViewerFramework commands exposing the VLI API
Large Dataset: Mesh

- Mesh Toolkit
  - PyMesh: wrapped for Python using SWIG
- Zoom command
Software Volume Rendering

- UT Austin volserv.idl
- OmniORB
  - C++ ORB with Python mappings

UT Austin VolRen
PVV: Architecture

PVV

PyOpenGL

DejaVu

ViewerFramework

Numeric

PyOpenGL

Tkinter

VLI

MolKit

Idle

VolView

PMV

Python Interpreter

SciPy’02, © M.Sanner, TSRI
ViPEr: a visual programming

- Basics, Architecture, Libraries
- Nodes, Execution Flow
- Demos

(Michel F. Sanner, Daniel Stoffler and Arthur J. Olson, ViPEr, a Visual Programming Environment for Python, 10th Python Conference 2002)
Enabling scientists (non-programmers) to build computational networks
ViPER: visual programming
Architecture Overview

NetworEditor

ViPER

VisualProgrammingEnvironment

Node Library

LibraryNodeIcon

MacroNetwork

MacroOutputNode

MacroInputNode

Network

NetworkBuilder

TypeManager

NetworkConnection

NetworkNode

InputPort

OutputPort

SciPy'02, © M. Sanner, TSRI
Why Python worked!

- Promotes better coding:
  - readability
  - high level
  - modular
- Software “Lego”
What could be better

- Handling of platform dependent and platform independent code
- Version management
- Distribution of sets of compatible modules
Conclusion

- **Software components used in:**
  - PMV: flexible and extensible molecular visualization environment
  - ADT: AutoDock front/back end
  - ViPER: General Purpose Visual Programming Environment

- **Python promotes modularity**
- **Code reuse**

SciPy’02, © M.Sanner, TSRI
Acknowledgments

- Molecular Graphics Lab. at TSRI
- NBCR, NIH, NSF
- Swiss National Science Foundation
- Daniel Stoffler (SymServer, AUM)
- Sophie Coon, Ruth Huey

Available at:
http://www.scripps.edu/~sanner/python