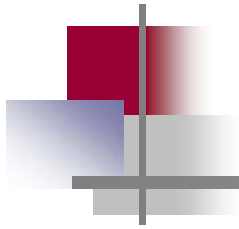


Pmv's Architecture and How to write new commands

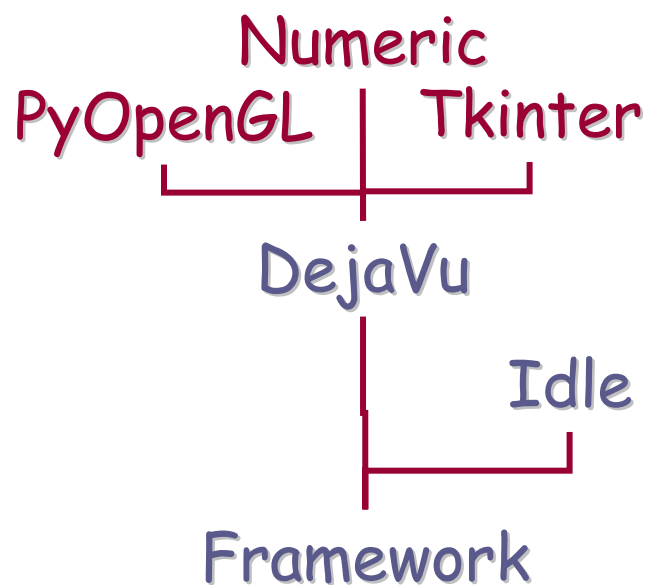
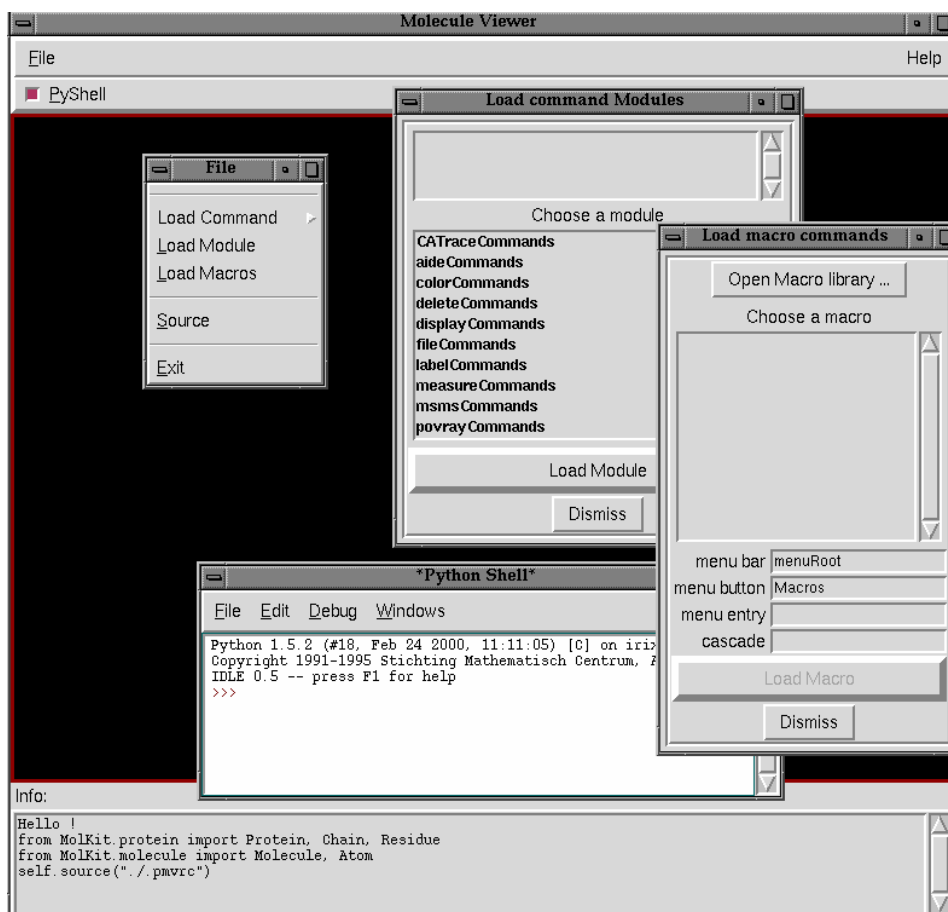


Simple version: (no Arguments)
Advanced version

Michel Sanner

ViewerFramework

The ViewerFramework package is a boilerplate Visualization application

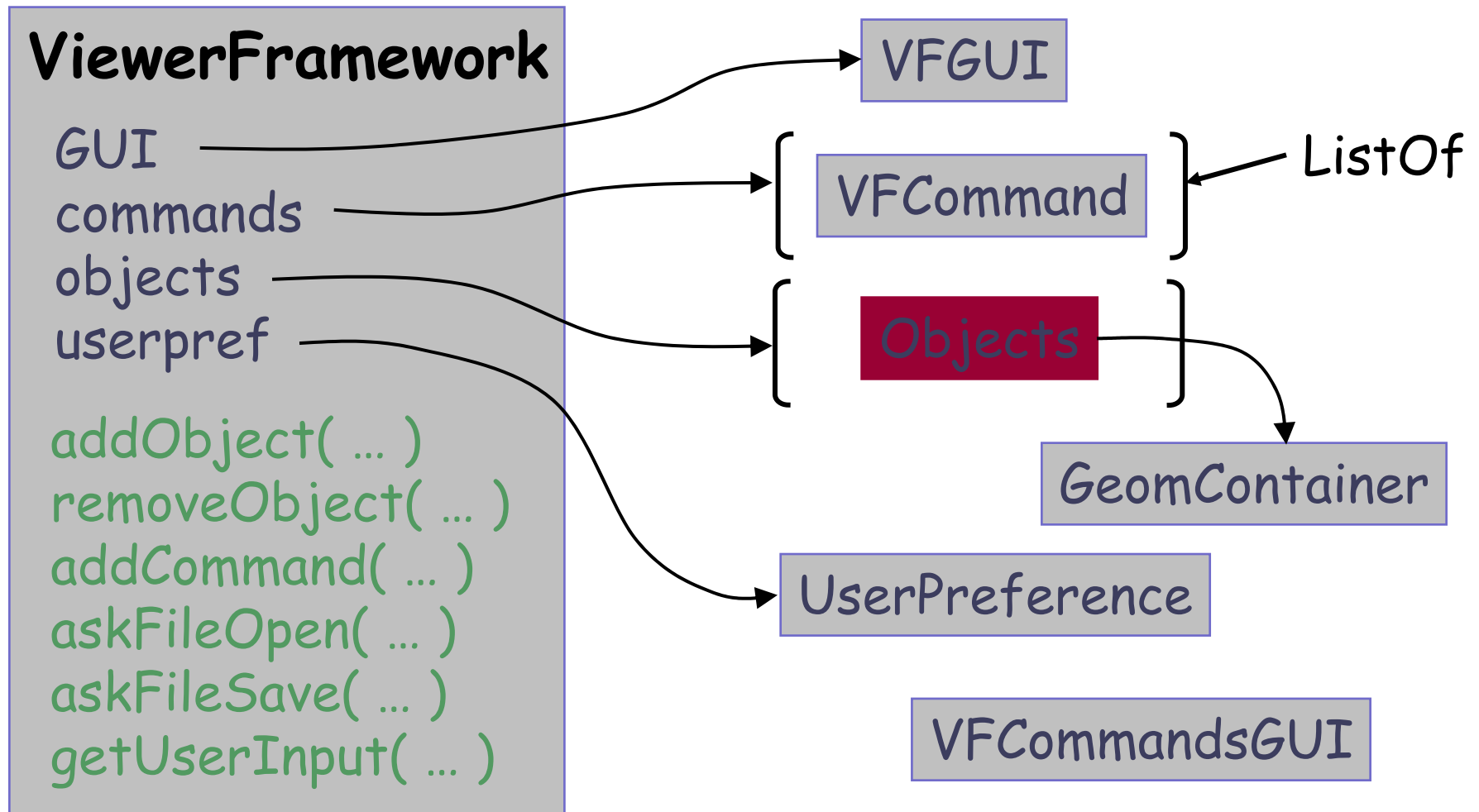




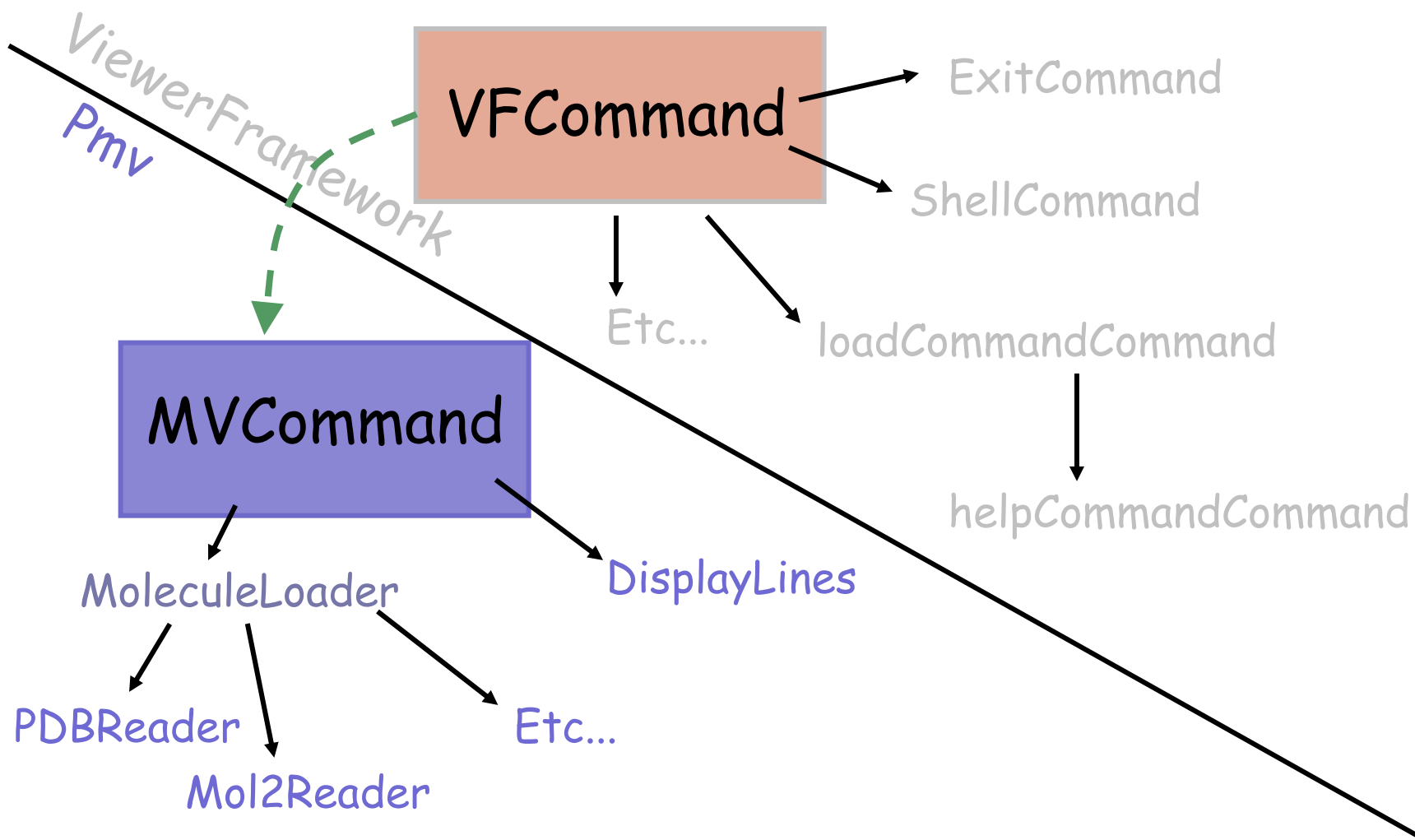
ViewerFramework - Design

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

ViewerFramework



Specializing ViewerFramework -2





Extending PMV

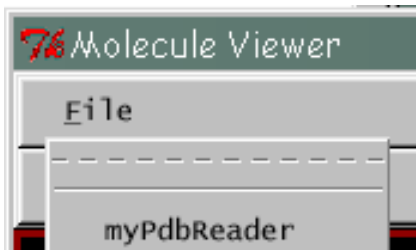
VFCommand

InputForm

Writing a Command for PMV.

VFCommand - Overview

When a command is called from the GUI
The `guiCallback` method is invoked



```
class MyCommand(VFCommand):
```

```
def guiCallback(self, *args, **kw):  
    apply( self.doitWrapper, args, kw)
```

```
def doitWrapper(self, *args, **kw):  
    self.beforeDoit()  
    self.doit()  
    self.afterDoit()
```

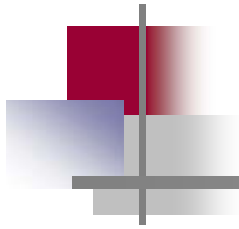
```
def doit(self, *args, **kw):  
    # do the work here
```

When a command is called from the Python
Shell the `__call__` method is invoked



```
def __call__(self, *args, **kw):*  
    apply( self.doitWrapper, args, kw)
```

Writing a command for Pmv



Simple version: (no Arguments)
Advanced version

Sophie Coon



Let's Write a Command

- o Write drawMolBoundingBox command to display a bounding box enclosing a molecule.
 - = 1 - Simple Version (No GUI)
 - = 2 - More advanced Version (with GUI)



drawMolBoundingBox overview

```
class DrawMolBoundingBox(MVCommand):
```

```
    def onAddObjectToViewer(self, obj):
```

```
        """Add a box geometry to each molecule"""
```

```
    def guiCallback(self)
```

```
        """ Call 'doit' with the current selection """
```

```
    def __call__(self, nodes):
```

```
        """ Call 'doit' with the specified nodes """
```

```
    def doit(self, nodes):
```

```
        """ Identify molecules specified in 'nodes' argument,  
        compute the size of each molecule's bounding-box,  
        and update the box geometries """
```



onAddObjectToViewer...

```
def onAddObjectToViewer(self, obj):  
    if not self.vf.hasGui: return  
    # get a handle to the molecule's geometry container  
    geomC = obj.geomContainer  
    # create a box geometry for the molecule 'obj'  
    mat= [(0,0,1),(0,1,0),(0,0,1),(0,1,0),(1,0,0),(1,0,0)]  
    g = Box('molboundingbox', materials = mat, visible = 0)  
    # add the geometry to the geometry container  
    geomC.geoms["molboundingbox"] = g  
    # add the box geometry to the viewer.  
    geomC.VIEWER.AddObject(g,parent =  
        geomC.masterGeom,unique = 0)
```



guiCallback & call...

```
def guiCallback(self):  
    self.doit(self.vf.getSelection())
```

```
def __call__(self, nodes = None):  
    if not nodes:  
        self.guiCallback()  
    else:  
        self.doit(nodes)
```

NOTE: The .vf attribute of a command points to the ViewerFramework application



doit... (1)

```
def doit(self, nodes):
    molecules, nodeSets = self.getNodesByMolecule(nodes)
    if len(molecules) == 0 : return
    self.log(nodes)
    for mol in molecules:
        # get a handle on the molecule's geometry container.
        g = mol.geomContainer.geoms["molboundingbox"]
        # get the atoms and their coordinates
        atms = mol.findType(Atom)
        coords = Numeric.array( atms.coords , 'f')
```



doit... (2)

```
# compute the size of the box geometry container
mincorner = Numeric.minimum.reduce(coords)
maxcorner = Numeric.maximum.reduce(coords)

# Set the size of the box geometry
g.Set(cornerPoints = (mincorner, maxcorner))
# Show the box.
g.Set(visible = 1)

self.vf.GUI.VIEWER.Redraw()
```



Let's be more ambitious !

```
def guiCallback(self)
```

```
    """Create an InputForm to let the user  
    specify:
```

- one or several molecules
- to display or undisplay the box ."""

More info on input forms are:

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



guiCallback...

```
def guiCallback(self):
    idf = InputFormDescr(title = 'Box features')
    molecules = self.vf.getSelection().top.uniq()
    # Build a MoleculeListChooser.
    idf.append({ 'widgetType' : MoleculeListChooser,
                'name' : 'chosenmolecules',
                'wcfg' : { 'title' : 'choose molecule',
                          'molecules' : molecules,
                          'mode' : 'multiple',
                          'lbwcfg' : { 'exportselection':0,
                                        'width' : 20,
                                        'height' : 5}},
                'gridcfg' : { 'sticky' : 'w' }})
```



guiCallback...(2)

```
idf.append({ 'widgetType' : Tkinter.Checkbutton,  
            'name': 'display', 'text': 'displayBox',  
            'defaultValue': 1, 'type': int,})
```

```
# build the form and get the specified parameters  
val = self.vf.getUserInput(idf)
```

```
# process the parameters and call 'doit' with the  
# proper arguments.
```

```
if len(val[ 'chosenmolecules' ]):  
    self.doit(val[ 'chosenmolecules' ], val[ 'displaybox' ])
```



doit... (1)

```
def doit(self, nodes, display=1):
    mols, atmSets = self.getNodesByMolecule(nodes, Atom)
    if len(molecules) == 0 : return
    self.log(nodes)
    for mol,atms in map(None, mols, atmSets):
        # get a handle on the molecule's geometry container.
        g = mol.geomContainer.geoms["molboundingbox"]

    if not display:
        # undisplay the box geometry
        g.Set(visible=0)
```



doit... (2)

else:

```
# get the atomic coordinates
```

```
coords = atms.coords
```

```
mincorner = Numeric.minimum.reduce(coords)
```

```
maxcorner = Numeric.maximum.reduce(coords)
```

```
# Set the size of the box geometry
```

```
g.Set(cornerPoints = (mincorner, maxcorner))
```

```
# Show the box
```

```
g.Set(visible = 1)
```

```
self.vf.GUI.VIEWER.Redraw()
```



Let's Run this command

```
from Pmv.moleculeBB import MoleculeBB  
from Framework.VFCommand import CommandGUI
```

```
MBBgui = CommandGUI()  
MBBgui.addMenuCommand( 'menuRoot',  
                        'myFunctions', 'molBB')
```

```
mv.addCommand( MoleculeBB(), 'molBB',  
               MoleculeBBgui )
```