

# Scripting with VIDA

Joe Corkery



OpenEye Scientific Software

CUP IX, Santa Fe, NM – March 2008

# Overview

- **Scriptable via embedded Python interpreter**
  - VIDA 3.0: Python 2.3
  - VIDA 3.1: Python 2.5
- **Mechanisms of control**
  - Interactive via Scripting Window
  - Scripts can be run by explicitly opening them
  - Scripts can be auto-loaded and run at startup



# Script Files

- File extensions
  - \*.py, \*.pyv, \*.vpy
- Auto-loading from user directory
  - "startup.py"
  - Contents of "scripts" sub-directory
  - User directory is platform specific
  - *Help > Open User Directory*

# Journal File

- **Journal file is recorded in user directory**
  - Contains all Python commands issued during a VIDA session
  - Entire session can be replayed by opening file
  - Useful aid to help explore VIDA scripting capabilities
- **“journal.py”**
  - written on successful exit
- **“journal\_timestamp.py”**
  - written on unsuccessful exit (crash)
  - very useful to include when submitting support requests

# User Interface

- All UI elements are bound to scripting functions
  - Menus, menu items, and toolbar items
- Can add/delete UI elements using scripting commands
  - Exceptions: View, Window, Recents, and Undo/Redo History Menus
- Can override many built-in operations



# Adding a new menu item

- **MenuAddButton(MenuName, ItemName, Command, Last)**
  - **Command:** Python code to be called when button is pushed
  - **Last:** Whether button should always appear last in the menu
- **MenuAddButton(MenuName, ItemName, Command, Shortcut, Last)**
  - **Shortcut:** Keyboard shortcut associated with button
- **MenuAddToggleButton(MenuName, ItemName, Command, Update, Last)**
  - **Update:** Python code to determine the button state



# Example: New Menu Item

```
MenuAddButton("Open Special",  
    "From Corporate DB...",  
    "OpenFromCorporateDB(PromptString())",  
    True)
```

```
MenuAddToggleButton("Style",  
    "Show Reference",  
    "ToggleReference()",  
    "IsReferenceShown()",  
    True)
```



# Adding a new menu

- **MenuAddSubmenu(Parent, MenuName, Last)**
  - **Parent:** Name of parent menu
  - **MenuName:** Name of new menu to be created
  - **Last:** Whether submenu should always appear at end of parent menu
- **MenuAddSubmenu(Parent, MenuName, DisplayName, Last)**
  - **DisplayName:** Name to be shown instead of using the identifier name MenuName
- **MenuAddSubmenu(Parent, MenuName, DisplayName, Update, Last)**
  - **Update:** Python code to populate the menu called right before menu is shown



# Example: New Menu

```
MenuAddSubmenu ("MenuBar", "Examples",  
False)
```

```
MenuAddSubmenu ("Examples",  
"Examples_OpenEye", "OpenEye", False)
```

```
MenuAddSubmenu ("Examples",  
"Examples_Local", "Local",  
"CreateLocalExamplesMenu()", False)
```



```

#####
# VIDA Cookbook - pdbopen.py                               Last Updated: March 12, 2008
#
# Copyright (C) 2005,2006,2007,2008 by OpenEye Scientific Software
#
# Description: This script demonstrates how to use the Python httplib to
#              load molecules directly from the PDB using their PDB code
#
import httplib

def LoadPDB(id):
    if (id == ""):
        return;

    WaitBegin()

    conn = httplib.HTTPConnection("www.rcsb.org")
    conn.request("GET", "/pdb/cgi/export.cgi/" + id + ".pdb?format=PDB&pdbId=" + id + "&compression=gz")

    r1 = conn.getresponse()
    if r1.status == 200:
        data = r1.read()

        if len(data) > 0:
            conn.close()
            f = open(id + ".pdb.gz", "wb")
            f.write(data)
            f.close()
            Open(id+ ".pdb.gz")
        else:
            raise IOError, "PDB id not found"

    WaitEnd()

#####
# Add menu item

MenuAddButton("Open Special", "From PDB...", 'LoadPDB(PromptString("Enter PDB code"))')

```



# Additional Menu Options

- `MenuAddSeparator(MenuName, SeparatorName, Last)`
- `MenuButtonActionSet(MenuName, ItemName, Command)`
- `MenuEnableItem(MenuName, ItemName, Enabled)`
- `MenuExists(MenuName)`
- `MenuRemoveItem(MenuName, ItemName)`
- `MenuRemoveAll(MenuName)`



# Adding to the popup menu

- **PopupAddSetupFunc(Function)**
  - **Function:** Python function which populates the menu and is called right immediately before the menu is shown
- **PopupAddSubmenu(MenuName)**
  - **MenuName:** Name of submenu to be created
- **PopupAddButton(ItemName, Command)**
  - **ItemName:** Name of menu item to be created
  - **Command:** Python code to be called when the menu item is clicked

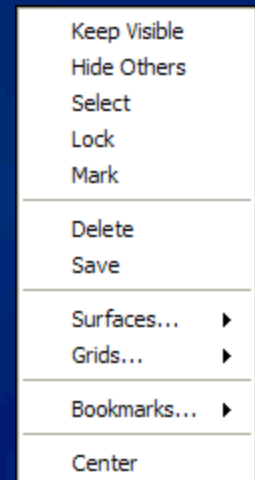


# Example: Popup Menu

```
menu = PopupAddSubmenu("View")
MenuAddButton(menu, "Center",
    "CenterOnClicked()")

PopupAddButton("Center",
    "CenterOnClicked()")

PopupAddSetupFunc(
    "BuildPopupViewMenu()")
```



# Toolbars

- Five named toolbars available

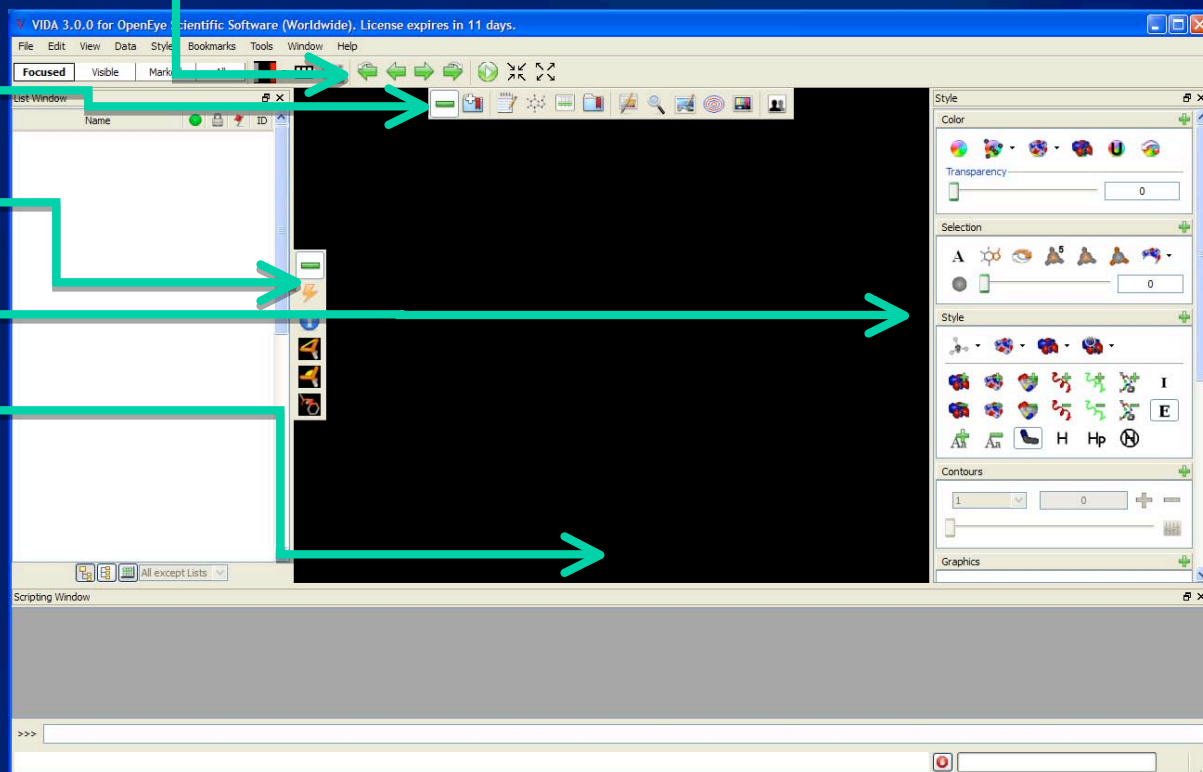
- Application

- Style

- Mouse

- 3D Right

- 3D Bottom



# Adding to a toolbar

- **ToolBarAdd(Icon/Text, Command, ToolTip, ToolBar, Name)**
  - **Icon:** Icon in XPM format stored in a list object
  - **Text:** Text to be displayed in the button
  - **Command:** Python code to be run when button is clicked
  - **ToolTip:** Text to be displayed when mouse hovers over button
  - **ToolBar:** Name of the toolbar to add the button to
  - **Name:** Name of the button to be added
- **ToolBarAddToggle(OnIcon/OnText, OffIcon/Offtext, OnCommand, OffCommand, ToolTip, Update, ToolBar, Name)**
  - **Update:** Python code to update the state of the button



# Overriding Built-in Commands

- Python allows reassignment of functions

```
def Open(Filename, _open = Open):  
    ids = _open(Filename)  
    for id in ids:  
        my_function(id)
```



# oechemlite

- Read-only implementation of pyoechem
  - Accessible within VIDA
- Access copies of molecules currently loaded
- Cannot directly modify molecules in VIDA



# oechemlite

- **ScriptableObjectGet( Id )**
  - **Id:** VIDA ID corresponding to the molecule desired
- **\*.GetKey()**
  - Returns a unique key corresponding to the calling object
  - This function has been added to the molecule, atom, and bond classes



# Example: oechemlite

```
PushIgnoreHint(1)
for id in ListGetObjects(1):
    mol = ScriptableObjectGet(id)
    if mol:
        color = OEColor(255, 128, 128)
        for atom in mol.GetAtoms():
            if atom.IsCarbon():
                ColorSet(atom.GetKey(), color)
PopIgnoreHint()
```



# VIDA 3.1: Coming soon

- OpenEye Toolkits

- All OpenEye toolkits will be directly accessible
  - This will require a separate license for the toolkits
- Modify molecules and check them back in

- PythonQt

- Exposes many Qt widget classes to Python
- Create new widgets within VIDA



# Example: OpenEye Toolkits

```
def DoShapeAlign(id1, id2):  
    refmol = oechem.OEMol()  
    fitmol = oechem.OEMol()  
  
    MoleculeCheckOut(refmol, id1)  
    MoleculeCheckOut(fitmol, id2)  
  
    best = oeshape.OEBestOverlay()  
    best.SetColorForceField(  
        oeshape.OEColorFFType_ImplicitMillsDean)  
    best.SetColorOptimize(True)  
    if not best.SetRefMol(ref):  
        raise Exception("Unable to set refmol as reference")
```



# Example: OpenEye Toolkits

```
scoreiter = oeshape.OEBestOverlayScoreIter()
oeshape.OESortOverlayScores(scoreiter,
                             best.Overlay(fitmol,
                                           oeshape.OEHighestTanimotoCombo))
score = scoreiter.Target()

newmol = oechem.OEMol(fitmol.GetConf(
                      ochem.OEHasConfIdx(score.fitconfidx)))
score.Transform(newmol)
oechem.OESetSDDData(newmol, "Tanimoto",
                    "-.3f" % score.tanimoto)

id = MoleculeAdd(newmol)
Visible(id, True)
```



# Example: PyQt

```
widget      = QWidget()
ref         = QPushButton(widget)
ref.text    = "Reference"
fit        = QPushButton(widget)
fit.text    = "Fit"
align      = QPushButton(widget)
align.text  = "Align"

... do some layout stuff here ...

ref.connect("clicked()", GetRef)
fit.connect("clicked()", GetFit)
align.connect("clicked()", Align)
```

