FU チュートリアル　　平成 26 年 8 月 18 日

－ FU をベースにした実践的分子モデリングソフトプログラミング実習 －

実習テキスト

１．はじめに

本講習会では、各自が独自の計算支援プログラムを作成する際のスタートとして使える分子モデリングプログラムを作成することを目標とする。

1)Windows7 での Python の開発環境

python で GUI プログラムを開発するには、python は当然として、wxPython など多数の site-packages を install しなければならない。この際、python のバージョンと site-packages がサポートしている python のバージョンを選択する必要がある。一般的に言えば、python2.7(32bit 版）を用いると wxPython や数値計算に必須の numpy,scipy などの site-packages が使える。FU で用いている site-packages を以下に示す（「FU プログラミング説明書」より転載)。なお、この組み合わせは FU の開発を始めた 2012 年 7 月時点での選択で、その後の新しいバージョンについては検討していない。

- **Python: python**

  http://www.python.org/download/ で公開されている Python 2.7.5 Windows Installer(Windows binary -- does not include source)をダウンロードした。

- **numpy:　数値計算ツール**

  http://sourceforge.net/projects/numpy/files/NumPy/1.6.2/ で公開されている numpy-1.6.2-win32-superpack-python2.7.exe をダウンロードした。

- **scipy: 科学計算ツール**

  http://sourceforge.net/projects/scipy/files/scipy/0.11.0b1/で公開されている scipy-0.11.0b1-win32-superpack-python2.7.exe をダウンロードした。

- **wxPython:　GUI ツール**

  http://www.wxpython.org/download.php で公開されている wxPython2.8-win32-unicode-py27 をダウンロードした。

- **PyOpenGL: OpenGL rapper**

  http://www.lfd.uci.edu/~gohlke/pythonlibs/で公開されている PyOpenGL-3.0.2.win32-py2.7.exe をダウンロードした。

- **matplotlib: グラフツール**

  https://github.com/matplotlib/matplotlib/downloads　で公開されている matplotlib-1.2.0.win32-py2.7.exe をダウンロードした。

本講習会ではこれらのソフトウエアの install の実習は行わない。興味あるひとは各自で試みられたい。本講習会では、開発環境をすべて含んでいる FU 配布パッケージに組み込まれている python shell(wxPython に含まれている）を用いてプログラミング実習を行う。FU 配布パッケージでは、上記 site-packages に加えて、FU の全てのモジュールが使える環境になっている。

2）wxPython の学習に役立つ Web サイト

サイト１：入門から応用までの学習サイト、 http://www.python-izm.com/

サイト２：wxPython tutorial、http://zetcode.com/wxpython/

サイト３：wxPython 全般、

http://xoomer.virgilio.it/infinity77/wxPython/index.html

本実習では、これらのサイトを随時参照する。サイト１は python と wxPython の学習サイトである。python の初心者はここで学習するとよい。サイト２は widgets（button など GUI の部品）の使い方が例プログラムで説明されている。サイト３は wxPython のマニュアルとして使うと便利である。これら以外にも優れた学習サイトが多数あるので、自分に適したサイトを見つけるとよい。

２．実習の概要

本講習会で用意した USB から、mymodel.py ファイルと fu-programming-example フォルダーを、事前準備で作った c:￥fu-23Dec2013 フォルダーにコピーする。実習では、mymodel.py に順次コードを追加・修正してプログラムを完成させる。example フォルダーには、本実習で作成する各段階のソースプログラム（下記）を収めてある。

①mymodel.py ・・・最初のソースプログラム

②mymodel-1.py ・・・ mymodel.py に**ステップ１**でコードを追加・修正した結果

③mymodel-2.py ・・・さらに**ステップ２**の追加・修正の結果

④mymodel-3.py ・・・さらに**ステップ３**の追加・修正を行った本実習の最終プログラム

３．プログラミング実習

まず、fu-23Dec2013 フォルダーにある fumodel.exe を起動する。本実習では fumodel のメインウィンドウは不要なので、PyCrust ウインドウから、

```
>>> fum.frame.Hide()
```

と入力してメインウィンドウを隠す。

そして、change directory コマンドで mymodel.py がある directry をカレントにしておく（pwd，ls，cd などのコマンドが使える）。

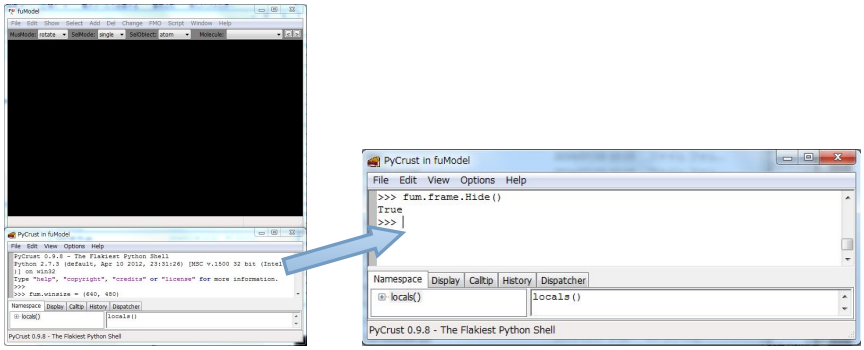

図１　fumodel を起動し（左)、PyCrust（右）で、作成したプログラムを実行する。

(注意）python は２バイト文字（日本語）をサポートしているが、本実習では、コメントを含めて、日本語は一切使わないこと。

１）mymodel.py をエディタで見る

mymodel.py をエディタで開いてソースプログラム（リスト１）を読む。

リスト１　mymodel.py

```
# -*- coding: utf-8 -*-
import wx
""" fu programming tutorial. 18Aug2014 """
class MyModel(wx.Frame):                                    ←①
    def __init__(self,parent,id,size):
        self.title='MyModel'
        wx.Frame.__init__(self,parent,id,size=size,title=self.title)
        self.bgcolor='black'                                ←②
        self.SetBackgroundColour(self.bgcolor)
# main program ------------------------------------------------------------
if __name__ == '__main__':                                  ←③
    app=wx.App(False)
    size=[400,300]
    mdl=MyModel(None,-1,size)                               ←④
    mdl.Show(True)
    app.MainLoop()
```

①　wxFrame を継承した MyModel class を定義する。__init__ method で初期値を定義する（instance が作られたときに実行される)。wx.Frame.__init__で、wx.Frame class の初期値設定 method を override する。wxFrame の説明は、学習サイト１を参照されたい。

②　‘black’ の他、’ red’ ,’ green’ などいくつかの色が定数で定義されている（wx.Colour 参照)。fuconst module に RGBA カラー（RGBColor や ElmCol)が定義してあるのであわせて参考にされたい。

③ If __name__ == ‘__main__’:行以下がメインプログラムである。

④ MyModel class の instance “mdl”を作成して、visible にする。

⑤ プログラムの実行

PyCrust ウィンドウで、

>>> execfile(‘mymodel.py’)

と入力して mymodel.py を実行する。図２のウィンドウが描かれる。

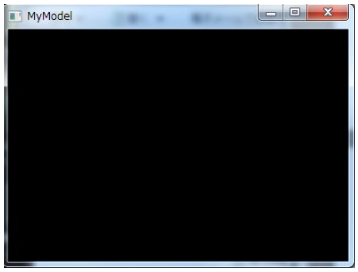

図２　mymodel.py の実行画面

Close ボタンを押すと終了し PyCrust に制御が戻る。ここで、escape キーを押すとコマンド待ち状態になる。

２）**ステップ１・・・** mymodel.py に Menu と statusbar のコードを追加し、mymodel-1.py として保存する。

リスト２　mymodel-1.py

```
# -*- coding: utf-8 -*-
import wx
import fumodel
""" fu programming tutorial. 18Aug2014 """
class MyModel(wx.Frame):
    def __init__(self,parent,id,size):
        self.title='MyModel'
        wx.Frame.__init__(self,parent,id,size=size,title=self.title)
        self.bgcolor='black'
        self.SetBackgroundColour(self.bgcolor)
        #
        menud=self.MenuItems() # method of this class
        # Create menu using fumodel.fuMenu class
        self.menubar=fumodel.fuMenu(menud) # create instance of fuMenu class
        self.SetMenuBar(self.menubar.menuitem) # method of wxFrame class
        self.menuitem=self.menubar.menuitem # attribute of fuMenu class
        # create StatusBar with two fields
        self.statusbar=self.CreateStatusBar() # method of wx.Frame class
        self.statusbar.SetFieldsCount(2) # method of wx.StatusBar class
        self.statusbar.SetStatusWidths([-8,-2]) # method of wx.StatusBar class
        # activate menu event handler
        self.Bind(wx.EVT_MENU,self.OnMenu) # method of wx.Frame class
```

リスト２　続き

```
    def Message(self,mess,loc,color):
        # write message. "color" is a dummy argument here.
        self.statusbar.SetStatusText(mess,loc) # method of wx.StatusBar

    def MenuItems(self):
        # menuitemdata: (top menu item, (submenu item, comment to be displayed
        #    in the first field of statusbar, checkable),..,))
        mfil= ("File", (
                  ("Open","Open...",False),
                  ("","",False),
                  ("Quit","Quit...",False),
               ))
        mdrw= ("Draw", (
                  ("Line model","Draw molecule in line model",True), # checkable
submenu
                ))

        menud=[mfil,mdrw]
        return menud

    def OnMenu(self,event):
        # menu event handler
        menuid=event.GetId()
        item=self.menuitem.GetLabel(menuid)
        bChecked=self.menuitem.IsChecked(menuid) # method of wx.Menu
        # File
        if item == "Open":
            print 'Menu:Open'
        if item == "Quit":
            print "Menu:Quit, quit the program"
            self.Destroy()
        if item == "Line model":
            mess='Menu:Draw, Checked'
            if not bChecked: mess='Menu:Draw, Unchecked'
            self.Message(mess,0,"black")

# main program ------------------------------------------------------------
if __name__ == '__main__':
    app=wx.App(False)
    size=[400,300]
    mdl=Mymodel(None,-1,size)
    mdl.Show(True)
    app.MainLoop()
```

①エディタで、mymodel.pyにリスト２の黒字で示すコードの追加・修正を行って、’mymodel-1.py’と名づけて保存する。

② MenuBarの作成については、サイト１を参照のこと。本実習では、簡単にメニューを作成できるfuMenu class（fumodel moduleにある）を用いている。これを用いると

MenuItems methodでメニュー項目を定義して、OnMenu methodでメニュー項目に対応する処理を記述するだけでよい。メニューの項目名で同じものがあるとエラーになるので注意すること。

③ wxFrameのBind method（*self*.Bind(wx.EVT_MENU,*self*.OnMenu)）で、Menu eventが発生したときの処理をOnMenu methodで行うことを指定している。

④ StatusBarは、wxFrameのmethod（CreateStatusBar）で作成する。ここでは、2つのfieldsを定義している。本実習で描画に用いるfuView class(fuview moduleにある)では、2番目のfieldに'Drawing...'というメッセージを書き出すので、1 filedのStatusBarだとエラーになるので注意すること。

⑤ プログラムの実行

PyCrustウィンドウで、

>>> execfile('mymodel-1.py')

と入力して実行すると、次のウィンドウが表示される。

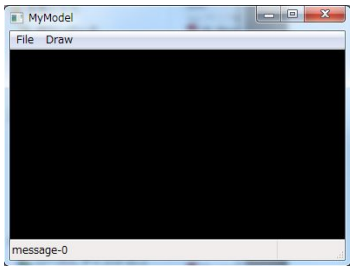

図３　mymodel-1.py の実行画面

Menuが動作することを確認後、MyModelウィンドウをcloseして、PyCrustに戻る。

３）**ステップ２・・・mymodel-1.py**に分子構造データ(PDBデータ）を読み込むコード、Frameにglcanvasを配置してOpenGLで描画するコードを追加して、mymodel-2.pyで保存する。

リスト３ mymodel-2.py

```
# -*- coding: utf-8 -*-
import wx
import fumodel
import os
import fumole
import fuview
import fuctrl
""" fu programming tutorial. 18Aug2014 """
class MyModel(wx.Frame):
    def __init__(self,parent,id,size):
        self.title='MyModel'
        wx.Frame.__init__(self,parent,id,size=size,title=self.title)
        self.bgcolor='black'
        self.SetBackgroundColour(self.bgcolor)
```

リスト３　続き

```
        # ctrlflag is needed to keep internally generated control flags in FU
        self.ctrlflag=fuctrl.CtrlFlag()
        # initialize mole instance
        self.mole=None
        # create OpenGL canvas
        self.canvas=self.CreateCanvas()
        # create view instance
        self.view=fuview.fuView(self,self.canvas)
        # menu data

          (変更しない部分を省略)

        # activate menu event handler
        self.Bind(wx.EVT_MENU,self.OnMenu) # method of wx.Frame class
        self.canvas.Bind(wx.EVT_SET_FOCUS,self.OnFocus)

    def CreateCanvas(self):
        # OpenGL drawing canvas
        size=self.GetClientSize()
        w=size.width; h=size.height
        attribList = (wx.glcanvas.WX_GL_RGBA,  # RGBA
                      wx.glcanvas.WX_GL_DOUBLEBUFFER,  # Double Buffered
                      wx.glcanvas.WX_GL_DEPTH_SIZE,32)  # 32 bit
        canvas=wx.glcanvas.GLCanvas(self,-1,pos=(-1,-1),size=(w,h),attribLi
st=attribList)
        canvas.SetBackgroundColour(self.bgcolor)
        return canvas

    def Message(self,mess,loc,color):
        # write message. "color" is a dummy argument here.
        self.statusbar.SetStatusText(mess,loc) # method of wx.StatusBar

    def ConsoleMessage(self,mess):
        # this method is in fumodel.py and is dummy here.
        pass

    def DrawMol(self,on):
        if on:
            self.view.CenterMolecular() # set center of draw canvas
            self.view.FitMolecular() # set scale to fit the canvas
            # 'updated' flag should be turn on (True) after any modification
(i.e. color change)..
            self.view.updated=True
            self.view.OnPaint()

    def OnFocus(self,event):
        self.canvas.SetCurrent()
        self.canvas.SetCursor(wx.StockCursor(wx.CURSOR_ARROW))

    def MenuItems(self):

          (変更しない部分を省略)
```

リスト３　続き

```
    def OnMenu(self,event):
        # menu event handler
        menuid=event.GetId()
        item=self.menuitem.GetLabel(menuid)
        bChecked=self.menuitem.IsChecked(menuid) # method of wx.Menu
        # File menu
        if item == "Open": # open file
            filename=self.OpenFile()
            if len(filename) > 0:
                # read PDB file using staticmethod of fuMole class in fumole
module
                pdbmol=fumole.fuMole.ReadPDBMol(filename)
                self.mole=fumole.fuMole(self)
                self.mole.SetPDBAtoms(pdbmol) # set atom data
                # set window title
                self.SetTitle(self.title+': '+filename)
                # message on statusbar
                self.Message('ReadPDBFile: '+filename,0,'black')

       if item == "Quit":
            print "Menu:Quit, quit the program"
            self.Destroy()
       # Draw menu
       if item == "Line model":
           # make draw bond data
           drwbnd=self.mole.MakeDrawBondData([])
           # set draw  bond data
           self.view.SetDrawBondData(True,drwbnd) # True: draw bond
           self.DrawMol(True)

    def OpenFile(self):
        filename=''
        wcard='pdb file(*.pdb;*.ent)|*.pdb;*.ent'
        dlg=wx.FileDialog(self,"Open file...",
                          os.getcwd(),style=wx.OPEN,wildcard=wcard)
        if dlg.ShowModal() == wx.ID_OK:
            filename=dlg.GetPath()
            if not os.path.exists(filename):
                wx.MessageBox(filename+" file not found.","ERROR (OpenFile)!",
                              wx.OK|wx.ICON_EXCLAMATION)
                return ''
        dlg.Destroy()
        return filename

# main program ------------------------------------------------------------
if __name__ == '__main__':
    app=wx.App(False)
    size=[400,300]
    mdl=MyModel(None,-1,size)
    mdl.Show(True)
    app.MainLoop()
```

①エディタで、mymodel-1.pyにリスト3の黒字で示すコードの追加・修正を行って、’mymodel-2.py’と名づけて保存する。
② fuCtrl classは、FUでプログラム間で制御のためのflagの受け渡しに用いる。本実習で使う、FUのclassや関数の中で暗に使われているので、これを定義しておかないとエラーがおきる。
③ OpenGLで描画するために、Frameにwx.glcanvas.GLCanvas classを貼り付ける（CreateCanvas method)。wx.glcanvas.GLCanvasについては、サイト３を参照せよ。本実習では、fuView class（fuview module)のmethodsを用いてglcanvasへの描画を行う。
④ file名を入力するOpenFile methodでは、wxFileDialog classを使っている。これについてはサイト１を参照せよ（wxMessageBoxについても同様)。
⑤ PDBデータの読み込みは、fumole moduleのReadPDBMol method（@staticmethod)で行っている（Fuのソースプログラムを参照、fileの読み込みについてはサイト１を参照されたい)。fumole moduleには、デカルト座標の読み込みmethod（ReadXYZAtomやReadXYZMolなど）があるので参考にされたい。
⑥ 構造データは、fumole moduleのfuMole classのattribute self.moleに格納している（self.moleはAtomデータを定義したAtom class instanceを格納したリストである)。fuMole classのSetPDBAtoms methodがPDBデータからAtom classのinstanceを作っている。これらについては、fumole moduleのfuMole classとAtom classのソースプログラム（または、本稿の４）を参照されたい。
⑧ プログラムの実行
PyCrustウィンドウで、
>>> execfile(‘mymodel-2.py’)
と入力して実行する。”File”-”Open”メニューで’sample’フォルダーにある’1crnhadd.pdb’ファイルを読み込み、”Draw”-”Line model”メニューを実行すると、図６のウィンドウが表示される。

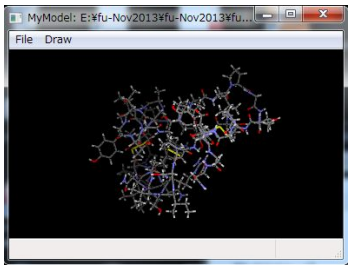

図６ mymodel-2.pyの実行画面。

⑨ ウインドウのサイズ変更の処理を行っていないので、サイズを変更しても書き換えが行われない。

４）**ステップ３・・・**マウス event、ウインドウサイズの変更 event の取得と処理コードを追加する

リスト４　mymodel-3.py

```
# -*- coding: utf-8 -*-
import wx
import fumodel
import os
import fumole
import fuview
import fuctrl

""" fu programming tutorial. 18Aug2014 """
class MyModel(wx.Frame):
    def __init__(self,parent,id,size):
        self.title='MyModel'
        wx.Frame.__init__(self,parent,id,size=size,title=self.title)
        self.bgcolor='black'
        self.SetBackgroundColour(self.bgcolor)
        # ctrlflag is needed to keep internally generated control flags in FU
        self.ctrlflag=fuctrl.CtrlFlag()
        # initialize mole instance
        self.mole=None
        # initialize mouse status
        self.mouseleftdown=False
        self.mousepos=[0,0]
        # create OpenGL canvas
        self.canvas=self.CreateCanvas()
        # create view instance
        self.view=fuview.fuView(self,self.canvas)
        self.view.fog=False # fog flag off (default: True)
        # menu data
        menud=self.MenuItems() # method of this class
        # Create menu using fumodel.fuMenu class

          (変更しない部分を省略)

        # create StatusBar with two fields

          (変更しない部分を省略)
```

リスト４　続き

```
        # activate menu event handler
        self.Bind(wx.EVT_MENU,self.OnMenu) # method of wx.Frame class
        # window event handler
        self.canvas.Bind(wx.EVT_SET_FOCUS,self.OnFocus)
        self.Bind(wx.EVT_CLOSE,self.OnClose)
        self.canvas.Bind(wx.EVT_SIZE,self.OnResize)
        # mouse event handler
        self.canvas.Bind(wx.EVT_LEFT_DOWN,self.OnMouseLeftDown)
        self.canvas.Bind(wx.EVT_LEFT_UP,self.OnMouseLeftUp)
        self.canvas.Bind(wx.EVT_MOTION,self.OnMouseMove)
        self.canvas.Bind(wx.EVT_MOUSEWHEEL,self.OnMouseWheel)

    def CreateCanvas(self):
        # OpenGL drawing canvas

          (変更しない部分を省略)

    def Message(self,mess,loc,color):
        # write message. "color" is a dummy argument here.
        self.statusbar.SetStatusText(mess,loc) # method of wx.StatusBar

    def ConsoleMessage(self,mess):
        # this method is in fumodel.py and is dummy here.
        pass

    def DrawMol(self,on):
        # draw molecular model

          (変更しない部分を省略)

     def SetDraw(self,model):
        # model: LINE = 0, (STICK = 1), BALL_STICK = 2, CPK = 3
        # set model to Atom class attribute
        for atom in self.mole.mol: atom.model=model
        # clear draw data
        self.view.SetDrawAtomData(False,[])
        self.view.SetDrawBondData(False,[])
        if model == 0: # "Line model"
            # make draw bond data
            drwbnd=self.mole.MakeDrawBondData([])
            # set draw  bond data
            self.view.SetDrawBondData(True,drwbnd) # True: draw bond
        elif model == 2: # "Ball-and-stick"
            # make draw atom and bond data
            drwatm=self.mole.MakeDrawAtomData([])
            drwbnd=self.mole.MakeDrawBondData([])
            # set draw atom and bond data
            self.view.SetDrawAtomData(True,drwatm) # True: draw atom
            self.view.SetDrawBondData(True,drwbnd) # True: draw bond
```

リスト４　続き

```
        elif model == 3: # "CPK model"
            # make draw atom data
            drwatm=self.mole.MakeDrawAtomData([])
            # set draw atom data
            self.view.SetDrawAtomData(True,drwatm) # True: draw atom

    def OnMouseLeftDown(self,event):
        self.mouseleftdown=True

    def OnMouseLeftUp(self,event):
        self.mouseleftdown=False

    def OnMouseMove(self,event):
        if not self.mouseleftdown: return
        pos=event.GetPosition()
        dif=pos-self.mousepos
        self.mousepos=pos
        self.view.MouseRotate(dif)
        self.view.OnPaint()

    def OnMouseWheel(self,event):
        rot=event.GetWheelRotation()
        self.view.Zoom(rot)
        self.view.OnPaint()

    def OnResize(self,event):
        self.view.CenterMolecular() # set center of draw canvas
        self.view.FitMolecular() # set scale to fit the canvas
        self.view.OnPaint()

    def OnClose(self,event):
        self.canvas.Destroy()
        self.Destroy()

    def OnFocus(self,event):
        self.canvas.SetCurrent()
        self.canvas.SetCursor(wx.StockCursor(wx.CURSOR_ARROW))
```

リスト４　続き

```
    def MenuItems(self):
        # menuitemdata: (top menu item, (submenu item, comment to be displayed
        #    in   the first field of statusbar, checkable),..,))
        mfil= ("File", (
                  ("Open","Open...",False),
                  ("","",False),
                  ("Quit","Quit...",False),
               ))
        mdrw= ("Draw", (
                  ("Line model","Draw molecule in line model",False), # line
mode
                  ("Ball-and-stick model","Draw molecule in ball-and-stick
model",False), # ball-and-stick model
                  ("CPK model","Draw molecule in CPK model",False), # CPK model
                ))
        menud=[mfil,mdrw]
        return menud

    def OnMenu(self,event):
        # menu event handler

          (変更しない部分を省略)

        # draw menu
        if item == "Line model":
            self.SetDraw(0)
            self.DrawMol(True)

        if item == "Ball-and-stick model":
            self.SetDraw(2)
            self.DrawMol(True)

        if item == "CPK model":
            self.SetDraw(3)
            self.DrawMol(True)

    def OpenFile(self):

          (変更しない部分を省略)


# main program ------------------------------------------------------------
if __name__ == '__main__':
    app=wx.App(False)
    size=[400,300]
    mdl=MyModel(None,-1,size)
    mdl.Show(True)
    app.MainLoop()
```

①エディタで、mymodel-2.pyにリスト４の黒字で示すコードの追加・修正を行って、’mymodel-3.py’と名づけて保存する。

② Window eventの処理

・*self*.Bind(wx.EVT_CLOSE,*self*.OnClose)・・・MyModel ウインドウがCloseされたとき、OnClose methodを実行する。

・*self*.canvas.Bind(wx.EVT_SIZE,*self*.OnResize)・・・canvasのサイズが変更されたとき、OnResize methodを実行する。

・ *self*.canvas.Bind(wx.EVT_SET_FOCUS,*self*.OnFocus)・・・canvasがFocusされたとき、OnFocus methodを実行する。

③ mouse eventの処理

・*self*.canvas.Bind(wx.EVT_LEFT_DOWN,*self*.OnMouseLeftDown)・・・マウスの左ボタンが押されたとき、OnMouseLeftDown methodを実行する。

・*self*.canvas.Bind(wx.EVT_LEFT_UP,*self*.OnMouseLeftUp)・・・マウスの左ボタンを離したとき、OnMOuseLeftUp methodを実行する。

・*self*.canvas.Bind(wx.EVT_MOTION,*self*.OnMouseMove)・・・マウスが移動したとき、OnMouseMove methodを実行する。

・*self*.canvas.Bind(wx.EVT_MOUSEWHEEL,*self*.OnMouseWheel)・・・マウスのwheelが回転したとき、OnMouseWheel methodを実行する。

④ プログラムの実行

PyCrustウィンドウで、

```
>>> execfile(‘mymodel-3.py’)
```

と入力して実行し、’File’-’Open’メニューで’sample’フォルダーにある’1crnhadd.pdb’を読み込み、’Draw’-’Ball-and-stick model’メニューを実行すると、図７のウィンドウが表示される。

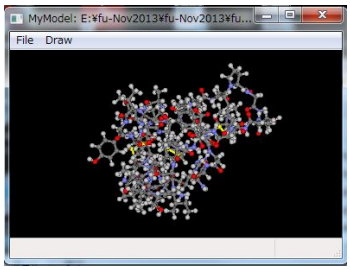

図７ mymodel-3.pyの実行画面。

⑤ mouseの左ボタンを押しながらドラッグすると分子が回転し、wheelをまわすと拡大・縮小する。ウインドウのサイズを変更すると、サイズにあわせて分子模型を再描画する。

これで本講習会の目標としたプログラムが完成した。

⑥ ここで、PyCrust から attribute を変更したり、method を実行してみよう。
PrCrust から、

```
>>> len(mdl.mole.mol)
```

と入力すると、原子数がプリントされる。

```
>>> mdl.SetDraw(3)
>>> mdl.DrawMol(True)
```

と入力して分子模型を変更して再描画する。

```
>>> mdl.view.fog=True
>>> mdl.DrawMol(True)
```

で、fog を有効にして再描画する。

```
>>> for atom in mdl.mole.mol:
>>>     if atom.elm == 'C': atom.color=[0.0,1.0,0.0,1.0]
>>>
```

と入力する。これで、炭素原子のカラーデータ (RGBA)を緑色に変更した。
続いて、

```
>>> mdl.SetDraw(2)
```

を実行し、分子模型を'ball-and-stick'に変更し、

```
>>> mdl.DrawMol(True)
```

を実行すると、変更した色とモデルで分子が表示される。
遊びで、もうひとつ。

```
>>> mdl.view.stereo=0
>>> mdl.DrawMol(True)
```

と入力するとステレオ (cross eye)表示になる（もとに戻すには mdl.view.stereo=2 とする）

```
>>> mdl.OnClose(1)
```

でプログラムを終了する。

これらの例で分かるように、実行時に **attributes** を変更したり、**method** を実行したり、内部のデータを加工したりが自由自在にできる。

以下では、FU をベースとしてプログラミングを行う際に役立つ分子データ（原子データ）ルーチンと描画ルーチンについて説明する（「FU プログラミング説明書」からの抜粋)。

４．fuMole class と Atom class: 分子データの保持と操作

FU の分子データは、fumole module の fuMole class である。この attributes をリスト５に示す。

リスト５　fuMole classのattributes

```
class fuMole():
    def __init__(self,parent):
        self.parent=parent
        try: self.frame=parent.frame
        except: pass
        self.molname='' # name of molecule, made from input file name
        self.inpfile='' # input file name
        self.outfile='' # save file name
        self.inpform='' # pdb,xyz,fmoinp,gmsinp,zmt
        self.mergedfile=''
        self.mergedmolname=''
        self.remark='' # comment
        self.mol=[] # list of atom instance
        self.bdadic={} # for fragmentation
        self.nter=0
        #
        self.labelcolor=[0.0,1.0,0.0]
        # parameters for view
        self.eyepos = [0.0,0.0,300.0]
        self.center = [0.0,0.0,0.0]
        self.upward = [0.0,1.0,0.0]
        self.ratio=1.0
```

① self.mol(list)に、Atom class の instance が格納される。

② Atom class の attibutes をリスト６に示す。

リスト６　Atom classのattibutes

```
class Atom():
    def __init__(self):
        # pdb data. See PDB manual for each items
        self.seqnmb=-1 # seq number of atoms 0,1,..,natom-1
        self.cc=[] # cartesian coordinate [x,y,z] in Angstrom
        self.conect=[] # connect data
        self.atmnam='' # atom name
        self.atmnmb=-1 # atom number
        self.resnam='' # residue name
        self.resnmb=-1 # residue number
        self.chainnam='' # chain name
        self.altloc=' '
        self.elm='' # element name
        self.focc=0 # occupancy
        self.bfc=0 # thermal factor
        self.charge=0 # atom charge
        # additional to pdb data
        self.bndmulti=[]                                                    #
{'single':1,'double':2,'triple':3,'aromatic':4,'HB':5,'CH/pi':6,'vdw':7}
        self.extraconect=[] # connect data for extrabond
        self.extrabnd=[]            #            extra            bonds,
1:H-bond,2:vdW,3:salt-bridge,4:CH/pi,...
```

リスト６　続き

```
        # draw parameters
         self.color=fuconst.ElmCol['ZZ'] # atom color. defaul:unknown elm
         self.show=True # show flag
         self.select=False # select flag
         self.model=0 # draw model, 0:line model
         self.atmrad=1.0 # default value defined in fuconst is set
         self.vdwrad=1.0 # default value defined in fuconst is set
         self.atmradsc=1.0 # scale factor of atom radius for ball and stick model
         self.vdwradsc=1.0 # scale factor of van der Waals radius
         self.thick=1  # bond thicknes. default balue in fuconst is set
         self.thicksc=1.0 #1.0 # scale factor of bond thickness
         # group data
         self.group=0 # group number
         self.grpnam='trg' # group name
         self.grpchg=0 # gropu charg
         self.envflag=False # environment (special group) flag
         self.parnam='' # name of parent molecule

           (一部省略)
         #
         self.SetDefaultAtmRad()
         self.SetDefaultBondThick()
         self.SetDefaultVdwRad()
```

③ *self*.cc, self.conect, *self*.atmnam, *self*.atmnmb, *self*.resnam, *self*.resnmb, *self*.chainnam, *self*.altloc, *self*.elm, *self*.focc, *self*.bfc, *self*.chargeは、PDBデータの各項目データである。その他、原子の属性データが定義されている。これらのうち原子種（元素）に依存するdefault値は、fuconst moduleで定義されている。

５．fuView class：分子模型描画ルーチンの説明

fuview module の fuView class が分子模型の描画を制御するルーチンである。この attributes をリスト７に示す。

リスト７　fuView classのattributes

```
class fuView():
    def __init__(self,parent,canvas):
        self.parent=parent
        self.ctrlflag=parent.ctrlflag
        self.canvas=canvas
        # initialize once
        self.gl_initialized = False
        # control flags
        self.ready=self.ctrlflag.ready
        self.ready=True
        self.updated=True
        # the draw object data
        self.atomdata=[]
        self.bonddata=[]
        self.chaintubedata=[]
        self.drawtube=[]
        self.arrowdata=[]
        self.extrabonddata=[]
        # initial setting
        self.eyepos = [0.0, 0.0, 300.0]
        self.center = [0.0, 0.0, 0.0]
        self.upward = [0.0, 1.0, 0.0]
        self.ratio = fuView.DEFAULT_RATIO  # angstrom per pixel
        self.DisplayList = None
        self.fog = True
        self.fogscale=5.0
        # default parameters
        self.bgcolor = fuView.DEFAULT_BGCOLOR
        self.rad_stick = fuView.DEFAULT_RAD_STICK
        self.rad_ball = fuView.DEFAULT_RAD_BALL
        self.rad_cpk_scale = fuView.DEFAULT_RAD_CPK_SCALE
        self.rad_peptide = fuView.DEFAULT_RAD_PEPTIDE
        self.stereo = fuView.STEREO_OFF
        self.tubecolor=[] # []: use default
        self.arrowhead=0.25 # length ratio of arrow head
        # flags for draw object
        self.atom=False
        self.bond=False
        self.chaintube=False
        self.selectcircle=False
        self.selectrectangle=False
        self.labelelm=False
        self.labelatm=False
        self.labelres=False
        self.labelfrg=False
        self.labelchain=False
        self.bdapoint=False
        self.formalchg=False
        self.distance=False
        self.extrabond=False
        self.sphere=False
        self.arrow=False
```

① *self*.atom,*self*.bond, *self*.chaintube, *self*.selectcircle, *self*.selectrectangle, *self*.labelelm, *self*.labelatm, *self*.labelres, *self*.labelfrg, *self*.labelchain, *self*.bdapoint, *self*.formalchg, *self*.distance, *self*.extrabond, *self*.sphere, *self*.arrowが、それぞれのオブジェクトを描画する(True)・しない(False)のflagである。

② 描画ルーチンは以下のものがある（fuview moduleにあるfuView classのmethods）。

・DrawAtom・・・原子（円または球）

・DrawBonds・・・結合（線または棒）

・DrawChainTube・・・ペプチド鎖（spline fitによる曲線のチューブ表示）

・DrawArrow・・・矢印（FUのscript ‘draw_pmi_vector.py’を参照）

・DrawSphere・・・球

・DrawLabel,DrawText2,DrawText3・・・文字列

③ それぞれ、事前に対応するmethod（原子を描く場合は、SetDrawAtomDataまたはSetDrawAtomList。前者は描画データをオブジェクトで、後者はListで渡す場合に用いる）を用いて描画データをセットしてから、OnDraw methodで描画する。

６．おわりに

本実習で作成したGUIプログラムは、単に分子モデルを描画するだけなので、とても分子モデリングソフトや分子計算支援プログラムと呼べる代物ではない。これをベースとして、分子構造を編集するための様々なmethodsを作り込まなければならない。その際、FUのソースコードが参考になると思う。また、特定の分子計算プログラムの入力データを作成したり、実行したりするコードを作成する際は、FUのscript ‘gamess-assist.py’や’tinker-optimize.py’が参考になるだろう。

FUが皆様のGUIソフトの開発に少しでも役立つことを願っている。