VisBAR_edu v1.03
マニュアル

鳥取大学大学院
工学研究科機械宇宙工学専攻応用数理工学コース

物理計算工学研究グループ (星准教授グループ)

平成 25 年 4 月 22 日

目 次

図 目 次

表 目 次

第1章　「VisBAR_edu」について

1.1　概要

「VisBAR_edu」は、プログラム言語 Python を用いた、教育むけ粒子 (原子) 可視化プログラムである。100 行未満の短いコードであり、学生が自身で発展させることを想定している。スクリーンショットを図 1.1 に示す。研究用プログラム「VisBAR」(=Visualization tool with Ball, Arrow and Rod) [1] パッケージの一部であり、教育向けに機能縮小させたものである。パッケージ付属の README ファイル (内容は付録 A) も参照。

以下では、windows 環境での説明を行う。また、分子の可視化を例として取り上げるため、粒子を「原子」と呼ぶ。

本プログラムはスクリプト言語「python」および 3D 可視化用ライブラリ「Vpython」で記述される。入力ファイルとしては、xyz 形式ファイルのみに対応している。xyz 形式ファイルから読み込む情報は、原子の位置座標のみとなっており、そこからステップ数と原子数を特定する。可視化に関しては、原子は全て白い球で表示され、ステップ数を X とすると画面には「step number X」と表示される。アニメーションは実行と同時に開始され、最終ステップの原子構造が可視化されるとアニメーションは停止する。オプションとして、原子間に結合を描くことができる。

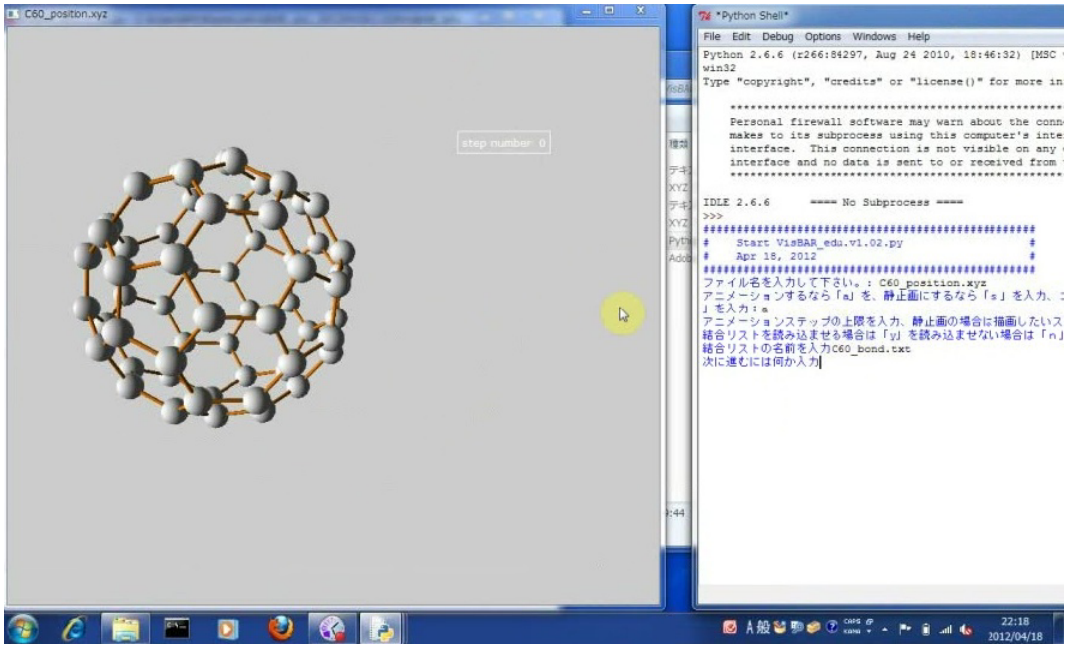

図 1.1: VisBAR_edu(v.1.02) のスクリーンショット

[1] T. Hoshi, Y. Akiyama, T. Tanaka and T. Ohno, "Ten-million-atom electronic structure calculations on the K computer with a massively parallel order-N theory", J. Phys. Soc. Jpn. 82, 023710, 4pp (2013). (http://dx.doi.org/10.7566/JPSJ.82.023710); 田中辰典, 鳥取大学修士論文 (2011); 秋山洋平, 鳥取大学卒業論文 (2012).

1.2 インストール

本プログラムを実行するために必要なソフトウェアのインストール方法を説明する。

1. python 公式サイト[2] から「python ver2.6」のインストーラをダウンロードし、インストールする。

2. 「python」のコマンドラインを開き、「print”hello”」と入力し、hello と出力されればインストールは成功である。

3. Vpython 公式サイト[3] などから可視化用ライブラリ「Vpython」(python ver2.6 用) のインストーラをダウンロードし、インストールする。

4. 先ほどと同様コマンドラインを開いて、「from visual import *」と「ball = sphere(pos=(0.,0.,0.)」と入力し、ウィンドウが開かれボールが表示されていれば「Vpython」のインストールは成功である。

5. 本ソフトウェアパッケージ（zip ファイル）をダウンロードし、展開する。展開してできたフォルダにおいて、次章以降のの作業を行う。

[2] http://python.org/
[3] http://vpython.org/

第2章　使用例

2.1　水分子モノマー H_2O の可視化

以下では実行例として、読込ファイル「H2O_position.xyz」と結合リスト「H2O_bond.txt」を読み込み可視化を行いコマ送り描画を行う。

1. 「VisBAR_edu.v1.03.py」を右クリックして「EDIT with IDLE」を左クリックする。 さらに「Run Module」を左クリックする。
2. 画面上で原子座標のファイル名を聞いてくるので、「H2O_position.xyz」を入力しエンターキーを押す。
3. さらに描画の方法を聞いてくるのでコマ送り「c」を入力する。
4. ステップ数の上限を聞いてくるので入力。今回は 5 と入力。
5. 結合リストを読み込ませるかどうか聞いてくるので「y」を入力。
6. 読み込ませる場合は結合リストの名前「H2O_bond.txt」を入力。
7. ここまで入力すると、読み込ませたファイル名のウィンドウが開かれ、0 ステップ目である図 2.1 が表示される。
8. さらにエンターキーを入力していくことで図 2.1 から図 2.3 までの各ステップが表示され、上限ステップの図 2.3 が表示されると、インタープリタで「何か入力して終了」と表示されるので、エンターキーを押してプログラムを終了する。

可視化の際は色の指定を行っていないため、酸素原子と水素原子は区別されず、ともに白い球で表示されている。黄色で表示されているのが結合である。

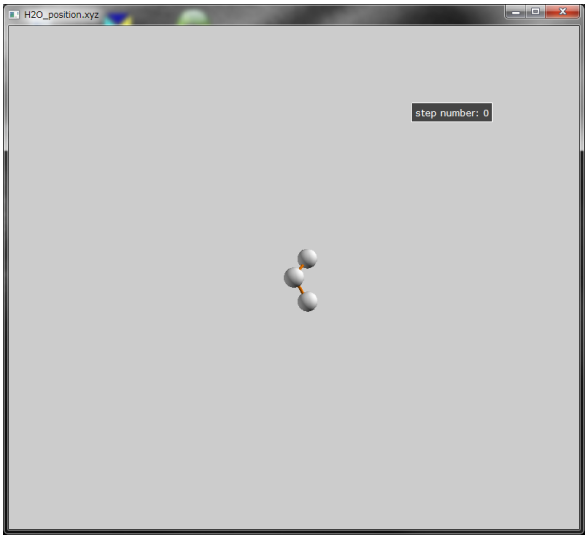

図 2.1: 水分子モノマー $(H_2O)0$ ステップ目

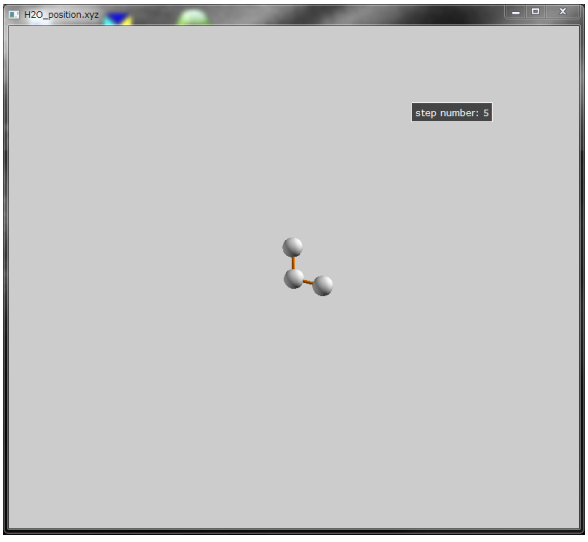

図 2.2: 水分子モノマー $(H_2O)5$ ステップ目

2.2 ファイル形式

以下では、水分子モノマー H_2O の計算（前節）を例に、本プログラムで使用するファイルについて説明する。

2.2.1 座標ファイル (xyz 形式ファイル)

xyz 形式ファイルの冒頭部分を記す。

```
           3
 H2O mdstep=            0
O     0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
H     0.584410971000000E+00  0.761619206000000E+00  0.000000000000000E+00
H     0.584410971000000E+00 -0.761619206000000E+00  0.000000000000000E+00
           3
 H2O mdstep=           10
O     0.302436079891157E-02 -0.475933184739195E-02 -0.984232411435636E-02
H     0.469278370524129E+00  0.855231752151677E+00  0.363221202155044E-02
H     0.651153798693286E+00 -0.779082442593406E+00  0.153844973808151E+00
           3
 H2O mdstep=           20
O     0.793786824770438E-02 -0.147618136821455E-01 -0.232167336225697E-01
H     0.341380631093269E+00  0.966937003449631E+00  0.239467484992550E-01
H     0.700435418943461E+00 -0.730747984535303E+00  0.347520989461861E+00
                                     .
                                     .
                                     .
```

xyz 形式ファイルは原子の情報を以下のように書くように定められている。

1. 一行目に原子の個数を書く。
2. 二行目には原子や分子の名称を書く (可視化ソフトの都合上必ず「mdstep=」という文字を入れておく必要がある)。
3. 三行目以降はひとつの行がひとつの原子の位置座標情報を表している。
4. 位置座標が書かれている行は必ず「原子種類 (元素記号)、x 座標、y 座標、z 座標」と書く。元素記号と各座標の間は半角スペースで区切る。全角スペースを使用してはいけない。

上記記述を連続的に書くことで、動画を表示させることができる。

2.2.2 結合リストファイル (txt ファイル)

任意の原子間に結合を描きたい場合は、結合リストをテキストファイル形式で与える。指定した原子間にのみ結合を描く。本例での結合リストファイルの中身を記す。

```
1 4
2  1
3  1
```

1　2
1　3

一番上に 4 と書いてある。これは結合を４本定義していることを表している。２行目からは原子のシリアル番号（座標ファイルに書かれた順番に、1,2,3... と番号付けがなされている）で結合を指定している。このとき、２行目は原子２と原子１が結合していることになる。例の結合リストファイルには４本の結合を定義している、重複しているため２本しか表示されない。

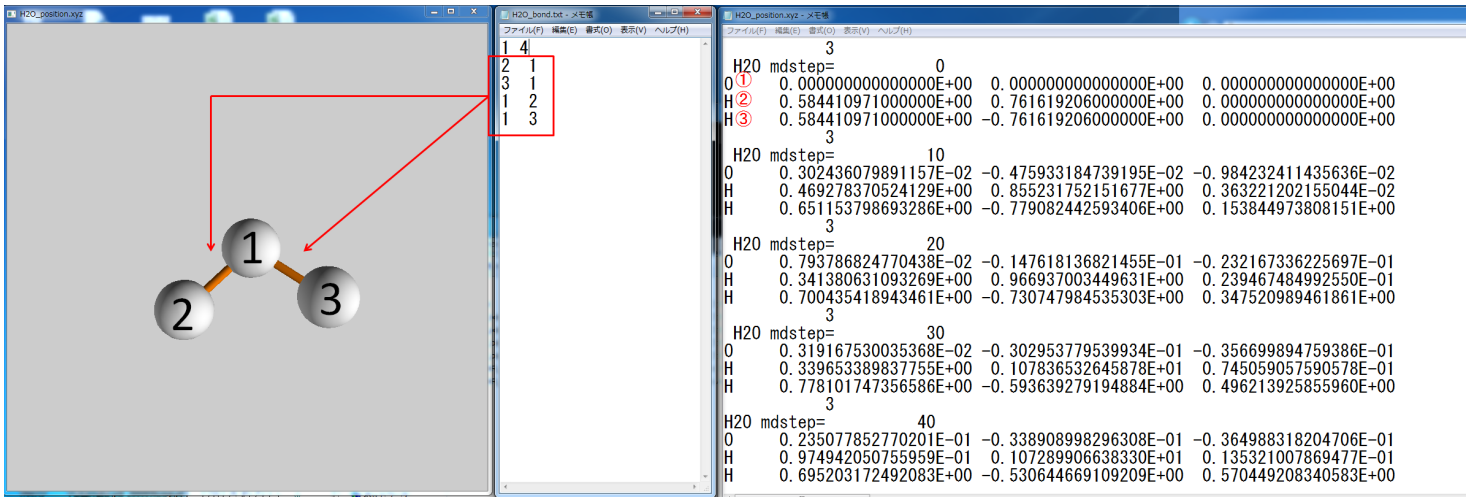

図 2.3: 水分子の結合の様子

2.3 C_{60} フラーレンの可視化

以下では実行例として、読込ファイル「C60_position.xyz」と結合リスト「C60_bond.txt」を読み込み可視化を行いコマ送り描画を行う。作業手順は前節と同様である。炭素原子が白い球で表示されている。黄色で表示されているのが結合である。

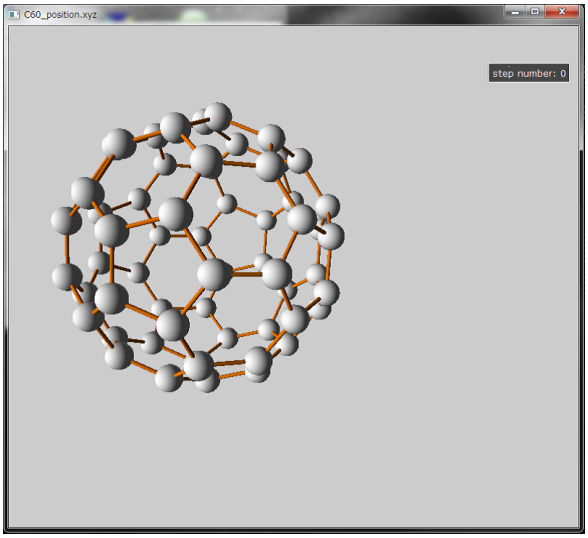

図 2.4: C_{60} フラーレン 0 ステップ目

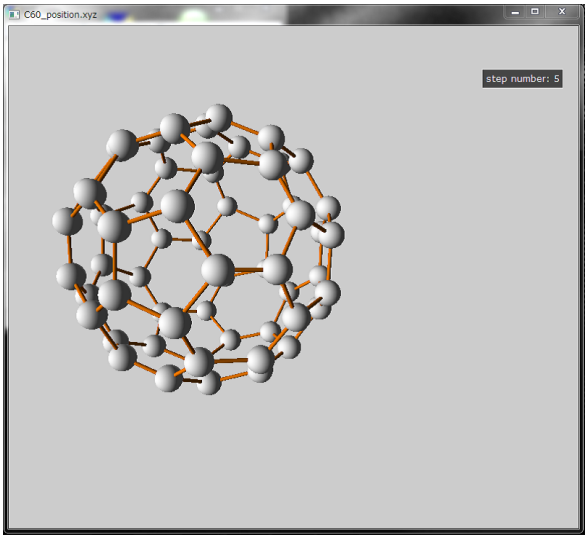

図 2.5: C_{60} フラーレン 5 ステップ目

2.4　単振動の可視化（初等教育むけ）

もっとも簡単な系として、単振動の可視化を説明する。

まず、可視化例の初期ステップと最終ステップを図 2.7 と図 2.8 にそれぞれ載せる。図の右側の球体が質点を表し、左側の球体はばねと壁の接点の位置を表し、球体間の結合の棒はばねを表現している。

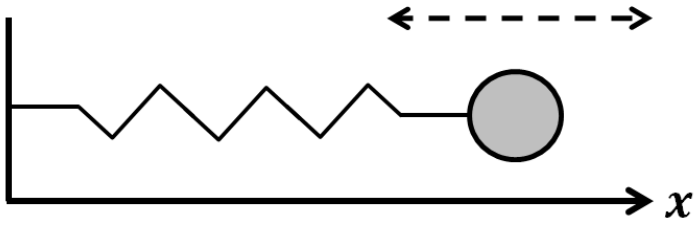

図 2.6: 単振動

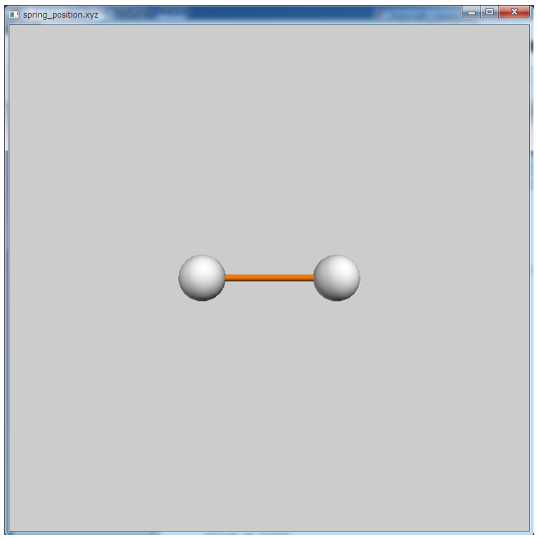

図 2.7: 単振動の初期ステップ

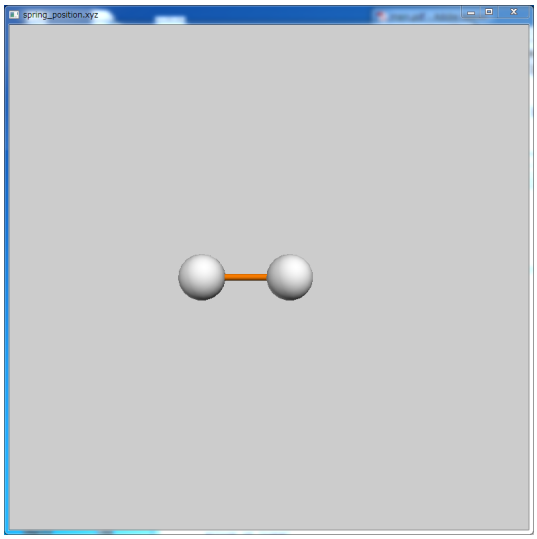

図 2.8: 単振動の最終ステップ

以下には、単振動の可視化に利用した座標ファイルと結合リストファイルを載せる。2.2.1 および 2.2.2 の説明に従い、入力ファイルを作成している。

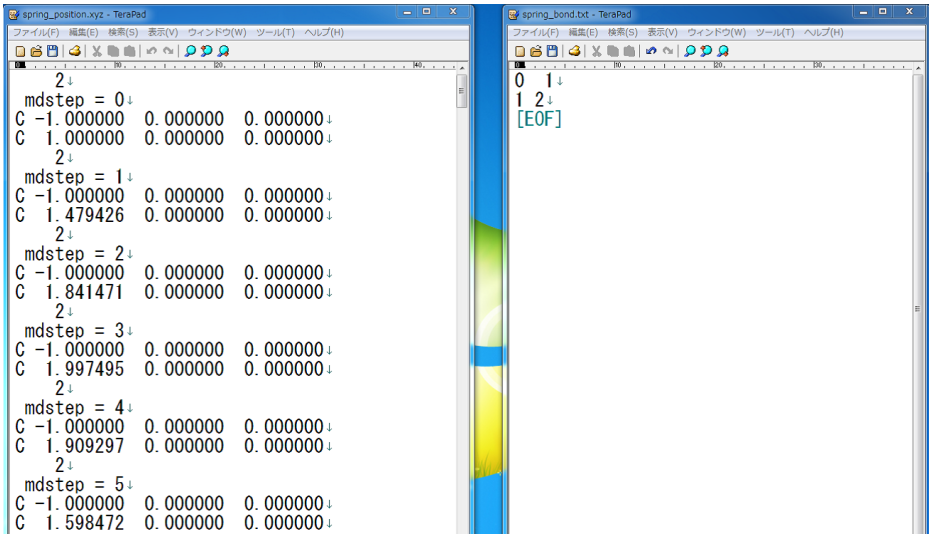

図 2.9: 単振動の座標ファイル (左) および結合リストファイル (右)

座標ファイルについては一つ目の原子は壁とばねの接点を表しているため (-1,0,0) の位置に固定している。そして 2 つめの原子が単振動する質点の位置座標を表す。また結合リストファイルについては、常に二つの原子間に結合を引くように定義している。

第3章　プログラム

3.1　基本

本節ではプログラム言語「Python」と可視化ライブラリ「Vpython」の説明を載せる

3.1.1　Python

Python は汎用の高水準言語である。特徴としては文字処理に優れ、クロスプラットフォームであり、可読性が高いということがあげられる。これらの性質は特に統合シミュレーション環境構築において重要な要素であるといえる (ここでの可読性とは、特にコードの読みやすさと整合性の事を指す)。また「Python」はスクリプト言語であるため、コンパイルの必要が無く、個々のモジュールを容易にテストでき、利便性が高い。さらに、ガーベージコレクション (プログラムが動的に確保したメモリ領域のうち、不要になった領域を自動的に解放する機能) を持っているため、連続使用においてPC 自体が重くなることを防ぐ。利便性の高い大規模な標準ライブラリを備えている。例として、以下にサンプルコードを記す。

```
f = open("sample.txt","r")
line = f.readline()
while line:
    line = f.readline()
    if line.find("check") >= 0:
        print line
        break
f.cloce()
```

上記のサンプルコードは、ファイル「sample.txt」を一行ずつ読み込み、文字列「check」を検索し、表示するというプログラムである。Python にはファイルを読み込むためのメソッドがいくつかあり、使用目的に応じて使い分けることができる。このコードを見るとわかるように、「python」は、条件文や関数を記述する際は必ずインデントを用いて区別する。さらに特に変数について型を指定する必要が無いため (値を入れた際に、入れた値の型として勝手に変数を定義してくれる)、大量の変数を使用する場合も見づらくなることは少ない。

3.1.2　vpython

３次元的に可視化するにあたって、「Vpython」という Python の 3D 可視化ライブラリを用いた。Vpython の特徴としては簡潔なコードであり、アニメーションが可能である。例として、以下にサンプルコードを記す。
-サンプルコード-

```
from visual import *
ball = sphere( pos = (0.,0.,0.) , radius = 0.5 , color = (0., 0., 1.))
```

上記のコードは、球を表示するというプログラムである。サンプルコードを実行すると、図 3.1 のようになる。1 行目のコードは Vpython ライブラリからの読み込みを行っている。上記コードは 2 行目でボールの可視化を行っている。球の位置を (0,0,0) として、半径を 0.5、色を RGB で上限を 1 として指定している。このような簡潔なコードで球体の条件を指定して可視化することができる。

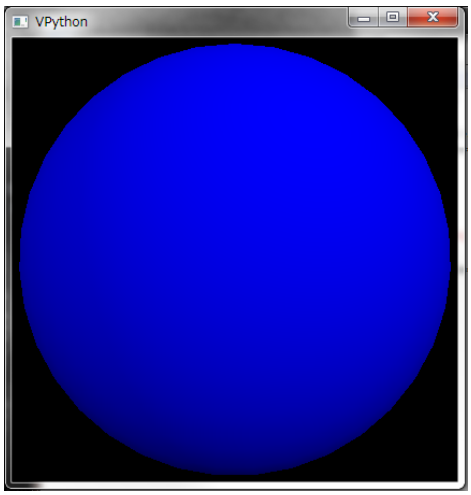

図 3.1: サンプルコードの実行例

3.2 ソースコード

3.2.1 解説

ソースコードについて簡単に解説する。

`import visual as Vp`
は Vpython を Vp という名前で読み込んでいる。「Vp.〜」と書いてある関数は Vpython の関数を読んでいることを意味する。
raw_input() 関数で各種設定をユーザーに入力させる。たとえば
`filename = raw_input("Input:file_name=")`
は「Input:filename=」と表示させて、ユーザーが入力した文字列を filename という変数で定義する。
h=open(filename,"r") から h.close() までは位置座標（xyz 形式）ファイルを一行ずつ読み込んでいる。

```
scene = Vp.display(title="%s"%filename , width=800 , height=800 ,
background=(0.8,0.8,0.8))
```
により可視化 window について window サイズを 800x800 として背景の色を (0.8,0.8,0.8) と白に近い灰色に設定している。

for i in range(len(allion)):は粒子のステップ数だけループを回している。また for j in range(len(allion[0])):は粒子数でループを回している。
ionball=Vp.sphere(frame=viewobject,pos=(allion[i][j][1],allion[i][j][2],allion[i][j][3]), radius=0.35) により球体を描いている。
if bond_select == "y":より下の処理は結合を描画するときだけ処理するように条件文を付けている。
rod = Vp.cylinder (frame=viewobject,pos=ballposition_ini, axis=ballposition_bin, radius=0.05, color=(1,0.5,0)) により円柱を描いている。

3.2.2 書き換え可能なパラメーター

本節では簡単に書き換え可能なパラメータの説明と書き換え方を載せる。

粒子の大きさと色

粒子を球体で表示するが、粒子の半径を変える場合は 46 行目の
ionball=Vp.sphere(frame=viewobject,pos=(allion[i][j][1],allion[i][j][2],allion[i][j][3]), radius=0.35)
について radius=0.35 の値を変更すればよい。例えば今の 2 倍の大きさにしたければ
ionball=Vp.sphere(frame=viewobject,pos=(allion[i][j][1],allion[i][j][2],allion[i][j][3]), radius= 0.7)
のように書き換える。

色を変更する場合は、同じ 46 行目について color=(0.,0.5,1.) のように設定する。もし青色で球体を表示させたいなら。
ionball=Vp.sphere(frame=viewobject,pos=(allion[i][j][1],allion[i][j][2],allion[i][j][3]), radius= 0.7,color=(0.,0.,1.))
のように書き換える。
色は RGB カラーコードで（red,green,blue）の色の要素を上限を 1 として指定している。（RGB の詳しい説明しないのでは Web などを参照。）

結合の太さと色

粒子間を円柱で結ぶことができるが、その太さを変更する場合は 59 行目と 74 行目の二カ所を書き換える必要がある。ただしこの 2 つの行は同じ記述である。
rod = Vp.cylinder (frame=viewobject,pos=ballposition_ini, axis=ballposition_bin, radius=0.05, color=(1,0.5,0))
先ほどの球体と同様に、radius=0.05 の値を変更する。

色についても同様に 59 行目と 74 行目の
rod = Vp.cylinder (frame=viewobject,pos=ballposition_ini, axis=ballposition_bin, radius=0.05, color=(1,0.5,0))
に対して、color=(0.5,1.,0.) などと数値を変更する。

可視化 window の大きさ

可視化 window の大きさを変えるには 39 行目の

```
scene = Vp.display(title="%s"%filename , width=800 , height=800 ,
 background=(0.8,0.8,0.8))
```

「width=800,height=800」の値を変更する。window の大きさを半分にしたい場合は

```
scene = Vp.display(title="%s"%filename , width=400 , height=400 ,
 background=(0.8,0.8,0.8))
```

のように設定する。この時 width と height の値は同じである必要はないが、同じ値に設定することを推奨する。

可視化 window の背景の色

画面の背景の色の設定を変更する。先ほどと同じ 39 行目の

```
scene = Vp.display(title="%s"%filename , width=400 , height=400 ,
 background=(0.8,0.8,0.8))
```

について background=(0.8,0.8,0.8) を RGB で設定する。デフォルトでは白っぽい灰色となっている。
白色にしたい場合は background=(1.,1.,1.)、
黒色にしたい場合は background=(0.,0.,0.) などと設定する。

ステップ数の書いてあるラベルの位置と色

このラベルは 44 行目により設定されており

```
Vp.label(frame=viewobject,x=10,y=10, text='step number: %1i' % (i))
```

x=10,y=10 の位置にステップ数を表示させるという意味である。視点の中心に近づけたい場合は x=10,y=10 の値を小さくする。x,y は同じ値である必要はない。
また color,linecolor により文字と枠の色を RGB カラーコードで設定する。
例えば、文字の色を赤として枠の色を黒にする場合は

```
Vp.label(frame=viewobject,x=10,y=10,color=(1.,0.,0.),linecolor=(0.,0.,0.),
 text='step number: %1i' % (i))
```

と設定する。

付 録A　付属「README」ファイルの内容

A.1　日本語版ファイル (00README_FIRST_JP.txt)

VisBAR_edu は Python による、教育むけ可視化ツールです。VisBAR_edu は「VisBAR」パッケージ [1] の一部です。もし VisBAR_edu を出版で使う場合は、論文 [1] を引用してください。

詳細は、付属日本語版マニュアル (VisBAR_edu_manual_jp.pdf) に記されています。英語版マニュアルは準備中です。

[1] Takeo Hoshi, Yohei Akiyama, Tatsunori Tanaka and Takahisa Ohno, "Ten-million-atom electronic structure calculations on the K computer with a massively parallel order-N theory", J. Phys. Soc. Jpn. 82, 023710, 4pp (2013). http://dx.doi.org/10.7566/JPSJ.82.023710

A.2　英語版ファイル (00README_FIRST_EN.txt)

VisBAR_edu is a python-based visualization tool for education. VisBAR_edu is a part of the package of 'VisBAR' [1]. If you use VisBAR_edu in a publication, please cite the paper [1].

Details are explained in the attached japanase manual (VisBAR_edu_manual_jp.pdf). An english manual is in preparation.

[1] Takeo Hoshi, Yohei Akiyama, Tatsunori Tanaka and Takahisa Ohno, 'Ten-million-atom electronic structure calculations on the K computer with a massively parallel order-N theory', J. Phys. Soc. Jpn. 82, 023710, 4pp (2013). http://dx.doi.org/10.7566/JPSJ.82.023710

付 録B　ソースコード

ソースコード B.1: VisBAR_edu_v1.03

```
# −∗− coding: cp932 −∗−
import visual as Vp
import re
print "################################################"
print "#␣␣␣␣Start␣VisBAR_edu.v1.03.py␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣#"
print "#␣␣␣␣Apr␣18,␣2012␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣#"
print "################################################"
filename = raw_input("Input:file_name=")
move_type = raw_input("Input:[animation:a,␣picture:s,␣frame␣by␣fram:c]=")
move_limit = raw_input("input:max_step=")
bond_select = raw_input("[Input␣bond_list_file:y,␣no␣bond␣list:n]=")
if bond_select == "y":
    bondlist = raw_input("Input:bond_list_name=")
    bond = open(bondlist,"r")
h=open(filename,"r")
ions=[]
allion=[]
line = h.readline()
while line:
    line=h.readline()
    if line.find("mdstep")>=0:
        while 1:
            ion = h.readline()
            ion =ion.strip()
            ion = re.sub("␣␣*","␣",ion)
            ion = re.split("␣",ion)
            if len(ion)<4 or [ion]==['']:
                break
                line =[]
            else:
                for x in range(3):
                    ion[x+1]=float(ion[x+1])
            ions += [ion]
        allion += [ions]
        ions=[]
    if [line]==['']:
        break
h.close()
scene = Vp.display(title="%s"%filename , width=800 , height=800 , background=(0.8,0.8,0.8))
viewobject=Vp.frame()
for i in range(len(allion)):
    if move_type == "s":
        i =int(move_limit)
    Vp.label(frame=viewobject,x=10,y=10, text='step␣number:␣%1i' % (i))
    for j in range(len(allion[0])):
        ionball=Vp.sphere(frame=viewobject,pos=(allion[i][j][1],allion[i][j][2],allion[i][j][3]), radius=0.35)

    if bond_select == "y":
        line = bond.readline()
        if line =="":
            for items in sub_items:
                ballposition_ini = [float(allion[i][int(items[0])−1][1]),
                                    float(allion[i][int(items[0])−1][2]),
                                    float(allion[i][int(items[0])−1][3])]
                ballposition_ter = [float(allion[i][int(items[1])−1][1]),
                                    float(allion[i][int(items[1])−1][2]),
                                    float(allion[i][int(items[1])−1][3])]
                ballposition_bin = [ballposition_ter[k]−ballposition_ini[k] for k in range(len(ballposition_ter))]
```

```
59                  rod = Vp.cylinder (frame=viewobject,pos=ballposition_ini, axis=ballposition_bin, radius=0.05, color
                        =(1,0.5,0))
60          else:
61              sub_items = []
62              items = line.split()
63              for line in range(int(items[1])):
64                  line = bond.readline()
65                  items = line.split()
66                  sub_items +=[items]
67                  ballposition_ini = [float(allion[i][int(items[0])-1][1]),
68                                      float(allion[i][int(items[0])-1][2]),
69                                      float(allion[i][int(items[0])-1][3])]
70                  ballposition_ter = [float(allion[i][int(items[1])-1][1]),
71                                      float(allion[i][int(items[1])-1][2]),
72                                      float(allion[i][int(items[1])-1][3])]
73                  ballposition_bin = [ballposition_ter[k]-ballposition_ini[k] for k in range(len(ballposition_ter))]
74                  rod = Vp.cylinder (frame=viewobject,pos=ballposition_ini, axis=ballposition_bin, radius=0.05, color
                        =(1,0.5,0))
75      if j==len(allion[0])-1:
76          if i==len(allion)-1 or i==int(move_limit):
77              break
78          Vp.rate(5)
79          if i == 0 or move_type == "c":
80              raw_input("[push␣any␣key]")
81          for obj in viewobject.objects:
82              obj.visible=0
83      if move_type == "s" or i == int(move_limit):
84          break
85  raw_input("end␣program␣[push␣any␣key]")
86  for obj in viewobject.objects:
87      obj.visible=0
88      scene.visible = 0
89  bond.close()
```