Computational Physics

3D Plotting & Animation

Prof. Paul Eugenio
Department of Physics
Florida State University
Feb 05, 2019

http://comphy.fsu.edu/~eugenio/comphy/

Announcements

- Due Today:
  - Turn-In Questions
    - Chapter 4: Accuracy & Speed
- Next Week:
  - No reading assignment or Turn-In questions

VPYTHON 7

NOTE: The textbook uses VPython 6, which is no longer supported

Many programs written in Classic VPython 6 will run in VPython 7 but will require minor changes:

- Module name changed to "**vpython**" from "**visual**"
- Object position is now a vpython is a **vector(x,y,z)**, and not a list **[x,y,z]**
- VPython7 uses "**canvas**" to replace "**display**"
- VPython 7 uses a web browser to display graphics whereas VPython 6 opened separate window

VPython7

```
import vpython as vp
ball = vp.sphere(pos=vp.vector(-5,0,0))

ball.pos.x += 5
```

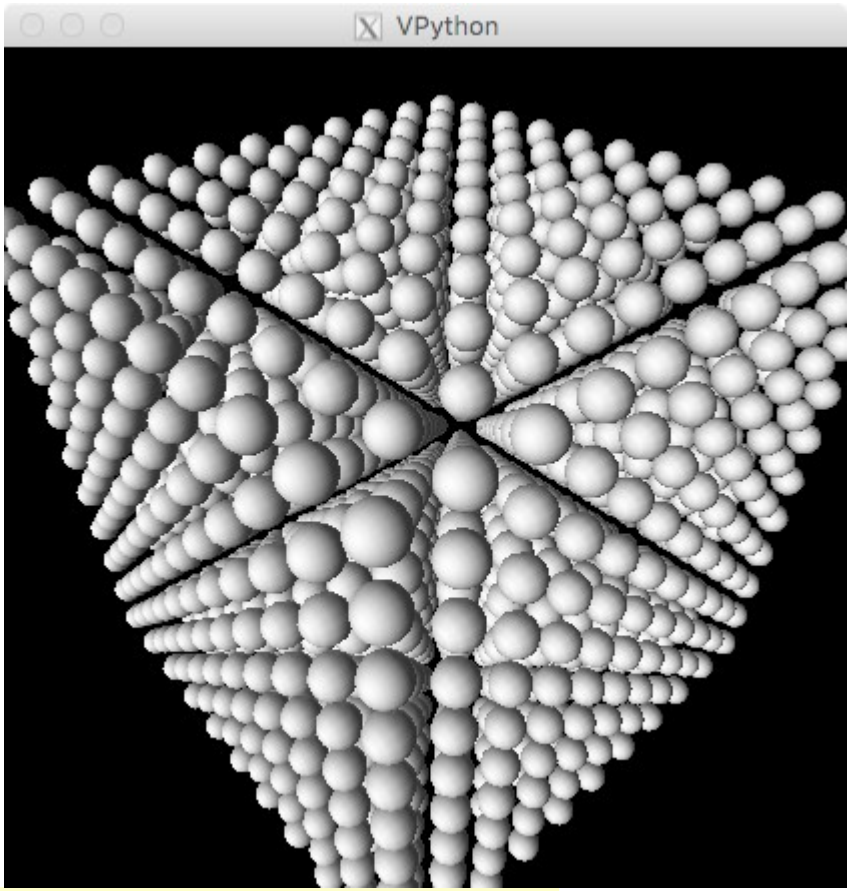
VPython6

```
import visual as vp
ball = vp.sphere(pos=[-5,0,0])

ball.pos[0] += 5
```

WE WILL USE VPYTHON 7 BUT ON THE CLASSROOM MACs

Picturing an atomic lattice

Source: lattice.py

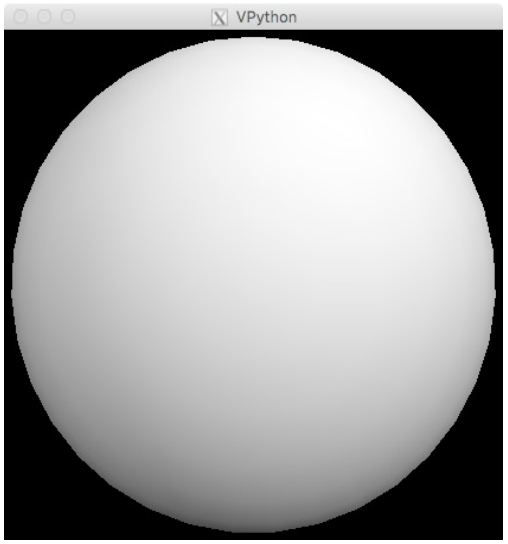



```
import vpython as vp

L = 5           # lattice size equals 2*L+1
radius = 0.3

# generate cubic lattice
for i in range(-L, L+1):
    for j in range(-L, L+1):
        for k in range(-L, L+1):
            vp.sphere(pos=vp.vector(i,j,k),radius=radius)
```

3D Graphics

The **vpython** package provides simple 3D images and animations designed with physicists in mind.

```
>>> import vpython as vp
>>> vp.sphere()
```



For a detailed explanation of the `vpython` package take a look at the online documentation at www.vpython.org. The `vpython` package provides a plethora of 3D variable types, i.e. objects, such as: arrow, box, cone, sphere, ring, …

Each 3D object has its own collection of variables for storing and changing the properties of elements after they are created.

vpython.org documentation sphere.html

Home

If you're new to Python and VPython: Introduction

A VPython tutorial

Introductory Videos

Pictures of 3D objects

Choose a 3D object

Work with 3D objects

Canvases/Events

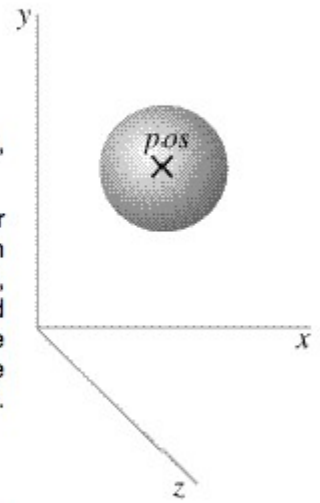
What's new

Classic VPython web site
VPython license
Python web site

sphere

Here is an example of how to make a sphere:

```
ball = sphere(pos=vector(1,2,1),
              radius=0.5)
```



This produces a sphere centered at location (1,2,1) with radius = 0.5, with the current foreground color.

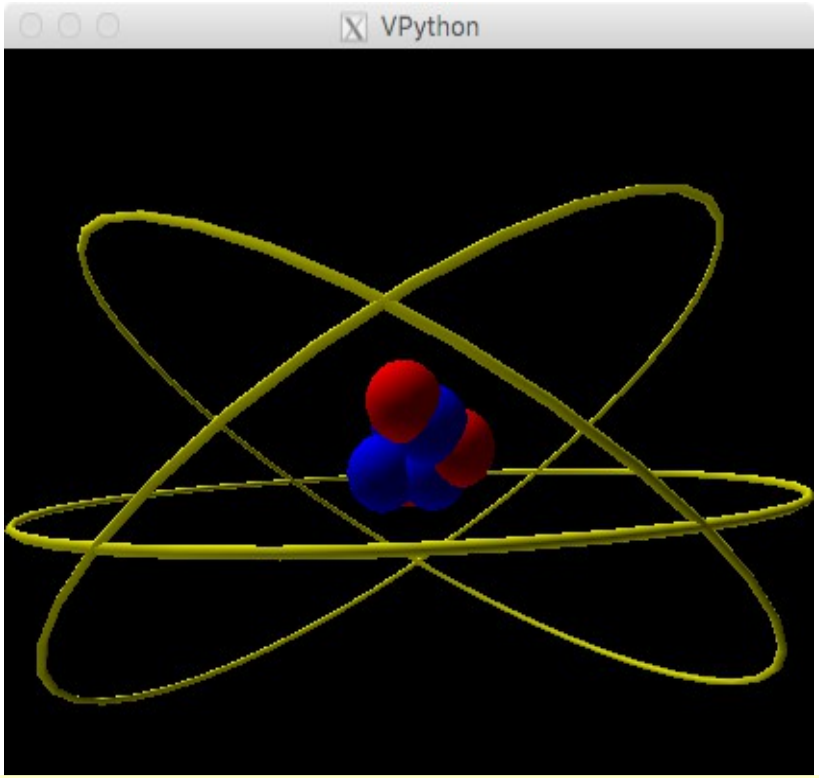
The sphere object has the following attributes and default values, similar to cylinder: pos vector(0,0,0), axis vector(1,0,0), color vector(1,1,1) which is color.white, red (1), green (1), blue (1), opacity (1), shininess (0.6), emissive (False), texture, and up vector(0,1,0). As with cylinders, up and axis attributes affect the orientation of the sphere but have only a subtle effect on appearance unless a non-smooth texture is specified or the cross section is oval. The magnitude of the axis attribute is irrelevant. Additional sphere attributes:

radius The radius of the sphere, default = 1

size Default is vector(2,2,2). Instead of specifying the radius, you can set size=vector(length,height,width), which means that the cross section of the sphere can be elliptical, making it like the ellipsoid object. Unlike other objects, changing size doesn't change axis, changing axis doesn't change size.

canvas By default, an object such as a sphere will be displayed in the most recently created 3D canvas, which will be the default canvas named "scene" unless you create a canvas yourself (see the related discussion at the start of the canvas documentation).

Picturing a Lithium atom

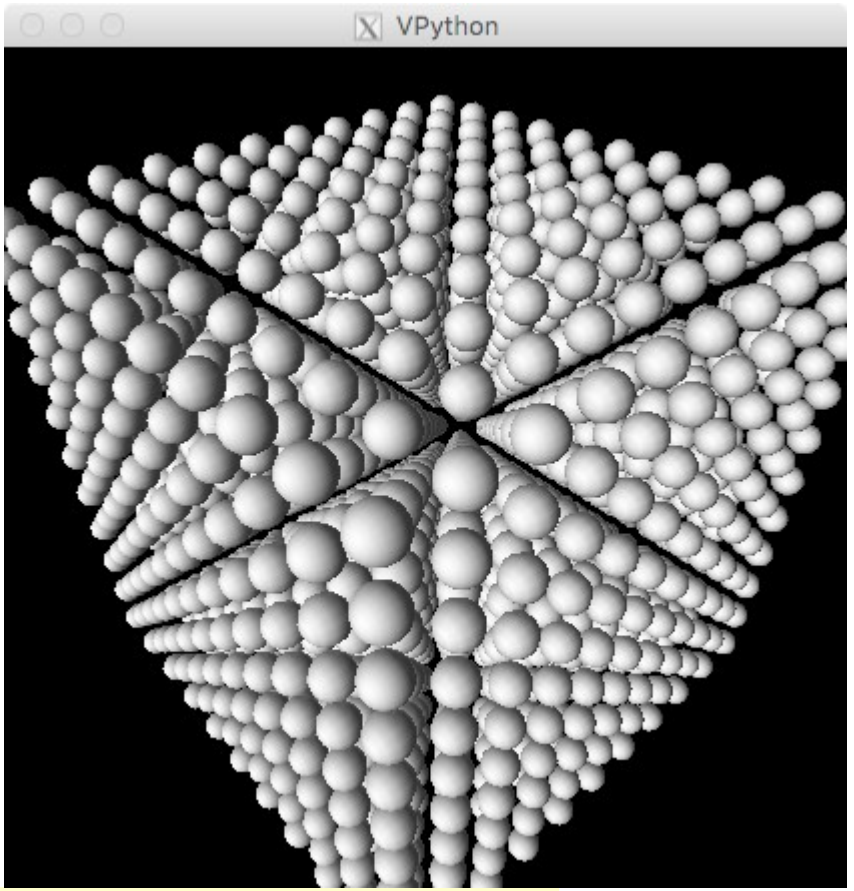


Source: atom.py

```
from vpython import sphere, color, ring, vector

# constructing the Lithium nucleus
sphere(pos=vector(0.1,0.1,0.1), radius=0.1, color=color.blue)
sphere(pos=vector(0,0,0.1), radius=0.1,color=color.blue)
sphere(pos=vector(0,0.1,0), radius=0.1,color=color.red)
sphere(pos=vector([0.1,0,0), radius=0.1,color=color.blue)
sphere(pos=vector(0.2,0,0), radius=0.1,color=color.red)
sphere(pos=vector(0.2,0,0.2), radius=0.1,color=color.red)
sphere(pos=vector(0.2,0,0.1), radius=0.1,color=color.blue)

# constructing the electron shell
ring(axis=vector(1,1,0), color=color.yellow, thickness=0.01, pos=vector(0,0,0))
ring(axis=vector(0,1,1], color=color.yellow, thickness=0.01, pos=vector(0,0,0))
ring(axis=vector(1,1,1), color=color.yellow, thickness=0.01, pos=vector(0,0,0))
```

Picturing an atomic lattice

Source: lattice.py

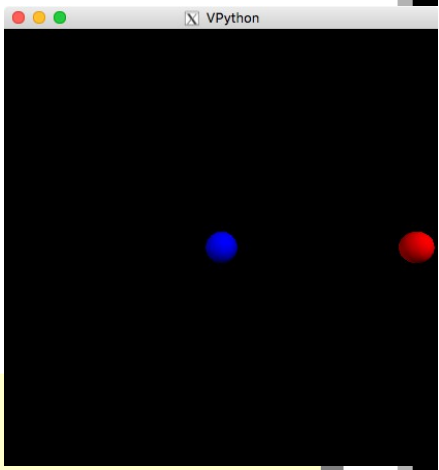



```
import vpython as vp

L = 5           # lattice size equals 2*L+1
radius = 0.3

# generate cubic lattice
for i in range(-L, L+1):
    for j in range(-L, L+1):
        for k in range(-L, L+1):
            vp.sphere(pos=vp.vector(i,j,k),radius=radius)
```

Animation

You can change the properties of a 3D object such as its size, color, orientation, or position.

If you change the position fast enough the object appears to be moving, i.e. you have an animation



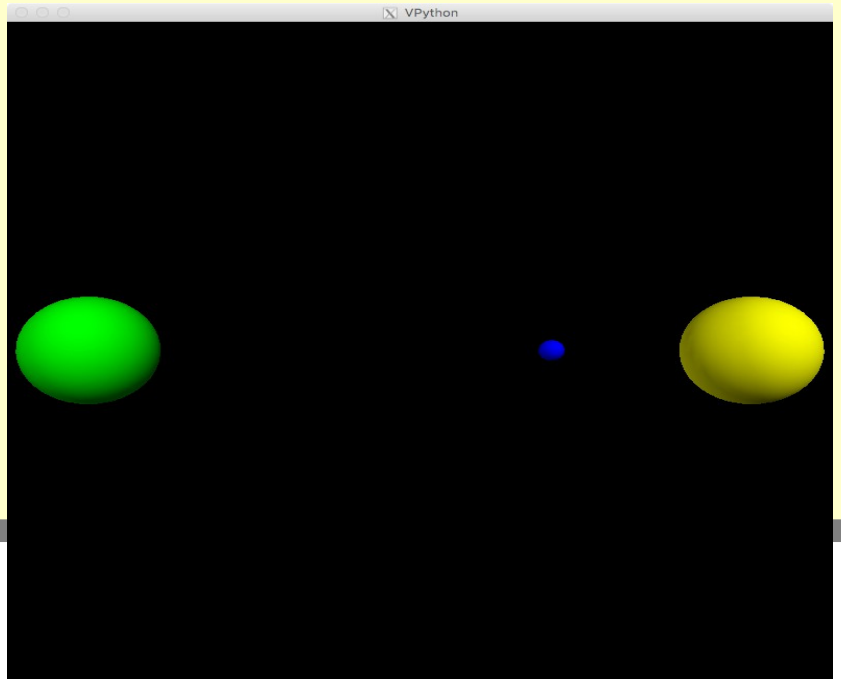
Source: animi.py

```
from __future__ import division, print_function
import vpython as vp
from time import sleep


# create two spheres
a=vp.sphere(pos=vp.vector(-5,0,0), radius=0.5, color=vp.color.blue)
b=vp.sphere(pos=vp.vector(6.1,0,0), radius=0.5, color=vp.color.red)

# have one sphere move increasingly faster
# towards the other sphere and stop at the sphere
for i in range(1,11):
    sleep(1/i)
    a.pos.x += 1         # change only the x position
```

Animation: exchange particle

Source: exchangeBoson.py

```
from __future__ import division, print_function
import vpython as vp



fSize = 0.5
bSize=0.1

vp.canvas(width=800, height=800)

# create the spheres
fermion1 = vp.sphere(pos=vp.vector(-2.5,0,0), radius=fSize ,vp.color=color.green)
fermion2 = vp.sphere(pos=vp.vector(2.5,0,0), radius=fSize ,vp.color=color.yellow)
boson = vp.sphere(pos=vp.vector(0,0,0), radius=bSize, vp.color=color.blue)

# initialize time and position
t,dt = 0.0,0.1
x,y,z,v = 0,0,0,5



while True:
    vp.rate(10)
    t += dt
    if x <-2 or x > 2:
        v = -v
    x = x + v*dt
    boson.pos = vp.vector(x, y, z)
```

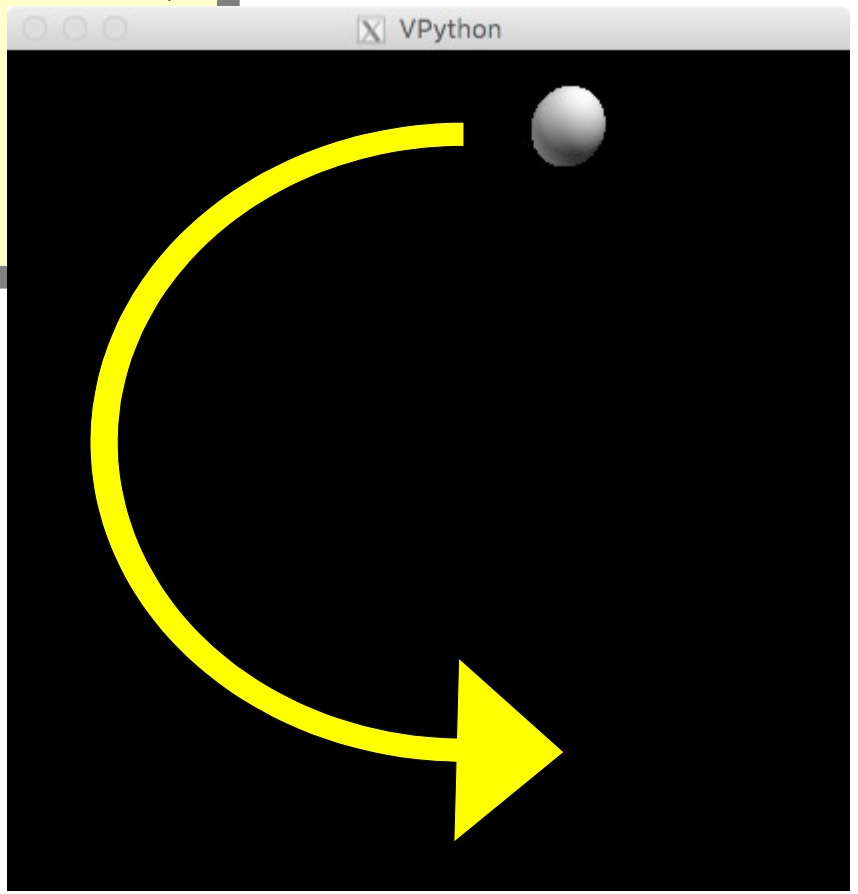


Revolving sphere

See textbook page 118

```
from __future__ import division, print_function

import vpython as vp
import numpy as np


s = vp.sphere(pos=vp.vector(1,0,0),radius=0.1)
for theta in np.arange(0,50*pi,0.1):
    rate(30)
    x = np.cos(theta)
    y = np.sin(theta)
    s.pos = vp.vector(x,y,0)
```

Circular motion



Let's See Some Examples

- http://hadron.physics.fsu.edu:/~eugenio/comphy/examples/
  - `vpython_examples.tgz`
  - `vpython_examples/`
    - `atom.py`
    - `lattice.py`
    - `animi.py`
    - `exchangeBoson.py`
    - `revolve.py`

Setting up VPython On A Classroom MAC

In a Mac Terminal Execute:

```
mkdir ~/python
cd ~/python/

/Applications/anaconda3/bin/python  -m  venv   myenv
source  ~/python/myenv/bin/activate
pip install --upgrade pip
pip install vpython
```

FOR FUTURE USE EDIT YOUR "~/.profile" file and add line:

```
alias myenv="source  ~/python/myenv/bin/activate"
export PATH="./:${PATH}"
```

Note: This setup is specific to the current Mac in use.

Now let's get back to programming