Physics 232: VPython Reference Sheet

1. Creating a program file

(a) Start the IDLE editor for Python by double-clicking on the snake icon. You will get an empty editing window.

(b) Enter the following two lines of code at the beginning of your program:

```
from visual import *
from __future__ import division
```

(Note: this should be typed "underscore underscore future underscore underscore". There are no spaces between the underscores and "future". This is an important line of code which tells Python to consider a fraction like 1/2 to be a floating point number 0.5, instead of taking the integer part of the result, which would be zero in this case.)

(c) From the file menu choose Save As. Save on your hard drive, making sure to add ".py" to the end of your filename.

2. Creating objects in VPython

(a) Sphere: The basic attributes are `pos`, `radius`, and `color`. The `pos` attribute specifies the location of the center of the sphere.

```
sphere(pos=vector(3, 5, -11), radius=0.15, color=color.magenta)
```

(b) Arrow: The basic attributes are `pos, axis`, and `color`. The `pos` attribute specifies the location of the tail of the arrow. The `axis` attribute specifies a vector that points from the tail to the tip of the arrow.

```
arrow(pos=vector(2, 4, 5), axis=vector(5, 2, 3), color=color.red)
```

3. Viewing objects in the VPython 3-D graphics scene

(a) To zoom in and out: hold down both mouse buttons and move the mouse up or down
(b) To revolve around the scene: hold down the right mouse button alone and move the mouse

4. Scalar constants or variables in VPython

At the beginning of a program, you can create named constants. Be sure to use comments indicating what the variable is and what its units are. For example:

```
g = 9.8            ## (m/s2)
oofpez = 9e9       ## One Over Four Pi Epsilon Zero (N m2/C2)
qproton = 1.6e-19  ## charge on a proton (C)
s = 1e-8           ## a constant distance (m)
```

In the rest of the program you can use these names in equations.

5. Vector variables in VPython

Creating a vector:

```
velocity = vector(0, -1.8e4, 0)   ## (m/s)
```

Components of a vector may be referred to by adding ".`x`", ".`y`", or ".`z`" to the name of a vector:

`velocity.x` is the x-component of the vector "`velocity`" defined above
`baseball.pos.z` is the z-component of the position of a sphere named "`baseball`"

6. Common mathematical expressions

To square a variable or number, type `**2`. a^2 would be `a**2` and r_x^2 would be `r.x**2`

To take a square root of a number or an expression, use `sqrt( )`. $\sqrt{3}$ is written `sqrt(3)`

π is a constant named "`pi`", which is already defined by VPython.

Sines and cosines are `sin()` and `cos()`, for example `sin(pi/2)`. Note that trigonometric functions use radians, not degrees!

To add a quantity to a variable, type:

```
                    myvariable = myvariable + 3
```
which means get the current value of `myvariable`, add 3 to it, and replace the previous value of `myvariable` with the result.

7. Simple loops in VPython

(a) `while` Loop:

```
deltat = 0.5
t = 0
while t < 10:    ## statements to be done inside loop are indented
    t = t + deltat
    print t
print 'End of loop'
```

In this example, the variable `t` is given the initial value of zero before the loop begins. The `while` statement instructs VPython to execute the indented statements over and over, until the value of `t` becomes equal to or greater than 10. At that point, the indented lines will no longer be executed.

(b) `for` Loop:

```
xlist = [0, 3, 9, 11]
for x in xlist:
    print x**2
print 'End of loop'
```

In this example, the list `xlist` is defined before the loop. The `for` statement instructs VPython to execute the indented statements with `x` set to each of the values in the list.

8. "if" statements

An `if` statement can be used to control the flow of a program. The indented statements following it are executed if the condition is true.

```
if x < 0:
    print 'Negative'
elif x == 0:
    print 'Zero'
elif x == 1:
    print 'Single'
else:
    print 'More'
```

There can be zero or more `elif` parts and the `else` part is optional. The keyword `elif` is short for "else if." Note that "==" is used to test if two expressions are equal.