Topics for this week

Introduction to graphing and visualization software

- VPython
- MatPlotLib

VPython

- A Python graphics module for modeling and simulation
- VPython = Python + IDLE + visual
- Supports the simulation of physical systems
  - Chabay and Sherwood, the authors of “Matter and Interaction” are co-developers of VPython
- [http://www.vpython.org/VPython_Intro.pdf](http://www.vpython.org/VPython_Intro.pdf) - overview for new Python users
- [http://www.vpython.org/webdoc/visual/index.html](http://www.vpython.org/webdoc/visual/index.html)
- from visual import *
MatPlotLib

- Matplotlib is a library for making 2D plots (of arrays)
- Origins in MATLAB
- Its philosophy: *Everyone should be able to create simple plots with just a few commands*
- [http://matplotlib.sourceforge.net/users/pyplot_tutorial.html#pyplot-tutorial](http://matplotlib.sourceforge.net/users/pyplot_tutorial.html#pyplot-tutorial)

---

Clicker question

```python
def f(list, k):
    list[0] = 'change'
    list[1] = k
    k = k + 10
    return

def main():
    L1 = ['a', 'b', 'c', 'd']
    k = 10
    f(L1, 0)
    print L1
    print k
main()```

What is printed?

A. ['change', 0, 'c', 'd']
10
B. ['a', 'b', 'c', 'd', 'f']
10
C. ['change', 0, 'c', 'd']
20
D. ['a', 'b', 'c', 'd']
20
Clicker question

def what(n):
    p = 1
    for i in range(2, n+1):
        p = p*i
    return p

What is computed?

A. $n!$ for $n \geq 1$
B. $n!$ for $n \geq 2$
C. $n^p$
D. $(n+1)!$ for $n \geq 2$
E. $(n+1)!$ for $n \geq 1$

Clicker Question (participation only)

What plotting software have you used before?

A. None
B. MS Excel
C. MatLab/Mathematica/Maple
D. VPython
E. Other
About Part 1 of Project 1

- `scale_volume(data, factor)` - Scale the volume of a sound wave
- `normalize(data)` - Normalize (maximize volume of) a sound wave
- `silence(dur)` - Generate a silence
- `sin_sample(freq, amp, dur)` - Generate a sine wave
- `half_speed(data)` - Generates a new array containing the same sound, but at half of the original speed
- `combine_mean(wavs)` - Combine a given list of sound waves into one of the same length using means
- `combine_interleave(wavs)` - Combine a given list of sound waves into one by interleaving samples
- `echo(data, delay, level)` - Add an echo effect to a sound

---

```python
def main():
    data = read_wav_file('preamble.wav')
    slow = half_speed(data)
    dur = len(data)/float(SAMPLE_FREQUENCY)
    sine_data1 = sin_sample(440, .05, dur)
    sine_data2 = sin_sample(550, .05, dur)
    sine_data = combine_interleave([sine_data1, sine_data2])
    data = append(data, silence(dur))
    normalize(slow)
    data = combine_mean([sine_data, data, slow])
    data = echo(data, .3, .25)
    normalize(data)
    scale_volume(data, 1.2)
    play(data)

if __name__ == '__main__':
    main()
```
The Visual Module of VPython

VPython is the Python programming language plus a 3D graphics module called "Visual" developed by David Shew. This document describes all of the Visual capabilities. To invoke the Visual module, place the following statement at the start of the file:

```python
from visual import *
```

Introduction: for those new to Python and Visual

Basic Display Objects

- **cylinder**: Start with cylinder: much of what is said here applies to other objects as well.
- **cone**: Framing: combining several objects into one
- **pyramid**: Faces: low-level object for special purposes
- **sphere**: Additional Attributes: visible, frame, display, class, members
- **ring**: Convenient Defaults
- **box**: Rotating as Object
- **ellipsoid**: Specifying Colors
- **cube**: Deleting an Object
- **helix**: Adjusting the Animation Rate
- **convex**: Floating Division: 3/4 is 0, but 3.0/4.0 is 0.75 in Python

Vector Computations

- **vector**: including mag, mag2, norm, cross, dot, rotate, diff_angle

Plotting Graphs of Functions or Data

- **Graph Plotting**: curve, glots, etc.

http://www.vpython.org/webdoc/visual/index.html
VPython Visual Objects

- VObjects exist for program duration
- VObjects are displayed on the display window
  - One default window
  - For multiple windows, give windows names
- VObjects have attributes
  - pos, color, length/height/width/radius, etc
- Changing attributes changes display

See example programs posted

```python
# program vp2.py
from visual import *

ball = sphere(pos=(-5,0,0), radius=0.5, color=color.red)
wallR = box(pos=(6,0,0), size=(0.2,4,4), color=color.green)

dt = 0.05
ball.velocity = vector(0.2,0,0)

while (1==1):
    rate(150)
    ball.pos = ball.pos + ball.velocity*dt
    if ball.x > wallR.x:
        ball.velocity.x = -ball.velocity.x
```

vp0.py, vp1.py, vp2.py, Bounce.py, bounce 2.py, vpythonPoints.py
Other VPython Objects

- These are not displayed
- A vector object supports the usual:
  - mag, mag^2 (mag squared), norm (normalized),
  - cross, dot, rotate, etc

How we will use VPython

- Visualize/animate how data changes during the computation
- Need to decide how data is visualized
- Visualize how quantities computed change during the computation
- Summarize results produced in a visual way
  - Use VPython for 3-D
- Examples
  - Animate points in square for Pi computation (PS2)
  - Visualize a closest pair computation (PS3)
Creating points in the VPython window

def highlight_points(points):
    # all points are initially green
    # set point[0] to red and then visit all other points,
    # coloring the points blue
    
    # create the points in the VPython window
    # vpoints[i] corresponds to points[i]
    color=(0,1,0)  # green
    vpoints = []
    for i in range(len(points)):
        vpoints.append(sphere(pos=points[i],radius=.4,color=color))

    vpoints[0].color = (1,0,0) # red
    for i in range(1,len(points)):
        vpoints[i].color = (0,0,1)
    rate(10)

vpythonPoints.py

VPython Graphs

- VPython has a powerful graph plotting subsystem
- from visual.graph import *
- Lines (gcurve), marks (gdots), bars (gvbars, ghbars), bins (ghistogram)