Introduction to 3D Computer Modeling

OBJECTIVES

In this course you will construct computer models to:

- Visualize motion in 3D using a programming environment called VPython, which is the widely-used Python programming language (python.org) plus a 3D graphics module, visual
- Visualize vector quantities like position, momentum, and force in 3D
- Do calculations based on fundamental principles to predict the motion of interacting objects
- Animate the predicted motions in 3D

In this lab you will learn:

- How to use VIDLE, the interactive editor for VPython
- How to structure a simple computer program
- How to create 3D objects such as spheres and arrows
- How to use vectors in VPython

OVERVIEW OF A COMPUTER PROGRAM

- A computer program consists of a sequence of instructions.
- The computer carries out the instructions one by one, in the order in which they appear, and stops when it reaches the end.
- Each instruction must be entered exactly correctly (as if it were an instruction to your calculator).
- If the computer encounters an error in an instruction (such as a typing error), it will stop running and print a red error message.

1 Using the VIDLE program editor to create a program

Open the VIDLE program editor by clicking on its icon in the Dock.

Starting a program: Setup statements

Enter the following statement in the editor window (*not* the shell window:

```
from visual import *
```

Every VPython program begins with this setup statement. It tells the program to use the 3D module (called "visual"). The asterisk means, "Add to Python all of the features available in the visual module".

Before we write any more, let's save the program:

- **In the editor, from the "File" menu, select "Save". Browse to a location where you can save the file and give it the name "intro.py". YOU MUST TYPE the ".py" file extension. Without the ".py" file extension the editor won't colorize your program statements in a helpful way.**

Both Python and VPython are undergoing continuous improvement. For example, before Python version 3.0, 1/2 was truncated to zero, but beginning with Python 3.0, 1/2 means 0.5. Also, starting with Python 3.0, print statements must be in the form print('hello') rather than the older form, print 'hello'. If you are using a version of Python earlier than 3.0, you should place the following statement as the first statement in your program, before the import of visual:

```
from __future__ import division, print_function
```

This statement (from *space* **underscore underscore** future **underscore underscore** *space* division, print_function) tells the Python language to treat 1/2 as 0.5, and makes the new form of print statements work on older versions of Python. You don't need this statement if you are using Python 3.0 or later, but it doesn't hurt, because it is simply ignored by later versions of Python.

2 3D Objects

Watch **VPython Instructional Videos: 1. 3D Objects** (http://www.youtube.com/VPythonVideos) demonstrating how to easily create 3D objects in VPython.

- **Complete the challenge task mentioned at the end of the video. Feel free to use any of the information in the video to complete the challenge.**

Checkpoint: ASK THE TA TO LOOK OVER YOUR WORK.

The 3D graphics scene

By default the origin $\langle 0, 0, 0 \rangle$ is at the center of the scene, and the "camera" (that is, your point of view) is looking directly at the origin.

- **Hold down both buttons and move the mouse up and down to make the camera move closer or farther away from the center of the scene. (On a Macintosh, hold down the Options key and the mouse button while moving the mouse.)**
- **Hold down the right mouse button alone and move the mouse to make the camera "revolve" around the scene, while always looking at the center. (On a Macintosh, hold down the Apple Command key and the mouse button while moving the mouse.)**

When you first run the program, the coordinate system has the positive x direction to the right, the positive y direction pointing up, and the positive z direction coming out of the screen toward you. You can then rotate the camera view to make these axes point in other directions.

Autoscaling and units

VPython automatically "zooms" the camera in or out so that all objects appear in the window. Because of this "autoscaling", the numbers for the "pos" and "radius" could be in any consistent set of units, like meters, centimeters, inches, etc. For example, we could have a sphere with a radius of 0.20 m and a position vector of $\langle 2, 4, 0 \rangle$ m. In this course we will always use SI units in our programs ("Système Internationale", the system of units based on meters, kilograms, and seconds).

The Python Shell window is important – Error messages appear here

IMPORTANT: Arrange the windows on your screen so the Shell window is always visible. DO NOT CLOSE THE SHELL WINDOW.

KILL the program by closing only the graphic display window.

Alternatively, simply rerunning your program will kill the graphics window and create a new one.

Scaling Arrows and Comment lines (lines ignored by the computer)

Comment lines start with a # (pound sign).
A comment line can be a note to yourself, such as:

```
# objects created in the following lines
```

Or a comment can be used to remove a line of code temporarily, without erasing it.

You can also put a comment at the end of a line: `sphere() # it's round`.

Comment out all but one arrow in your program. For the remaining arrow:

- **Change something in the arrow's code such that the arrow is half as long and points in the opposite direction, with its tail remaining on the same sphere.**

3 Debugging Syntax Errors

Watch **VPython Instructional Videos: A. Debugging Syntax Errors**
http://www.youtube.com/VPythonVideos
which discusses common syntax errors produced by novice users of VPython.

4 Variable Assignment

Watch **VPython Instructional Videos: 2. Variable Assignment**
http://www.youtube.com/VPythonVideos
which demonstrates how to create variables to store information and reference it later.

- **Complete the challenge task mentioned at the end of the video. Assign each 3D object a variable name. Feel free to use any of the information in the video to complete the challenge.**
- **Move one sphere twice as far from the y-axis. What happened to the three arrows? (Move it back to its original position when you're done experimenting.)**

Print Command

- **Start a new line at the end of your program and type:**

  `print(`*`variable.attribute`*`)`
  Replace *variable.attribute* with the name of one of your 3D objects and one of the valid attributes

associated with that object. For example, if you want to print the position attribute of a sphere named ball, it would look like this:
`print(ball.pos)`

- **Run the program.**
- **Look at the Shell window. The printed value should be the same as the value of the attribute you printed.**

Checkpoint: ASK THE TA TO LOOK OVER YOUR WORK. Make sure you are using variable references when defining the attributes of your arrows.

Another example of change in Python is that before Python 3.0, you could say `print ball.pos`, but starting with Python 3.0 one must say `print(ball.pos)`. If you are using an earlier version of Python, it is a good idea to use parentheses anyway, because it doesn't hurt, and it works with later versions of Python.

5 Turn in your program

Make sure everyone in your group agrees that the program is correct. Check with a neighboring group.

Upload your final program to the eLC assignment dropbox. Make sure the names of *everyone* in the group appear in the comment header at the beginning of the file. (Only one of you should upload the program, but everyone should keep a copy.)

6 Using VPython outside of class

You can download VPython from http://vpython.org/ and install it on your own computer.

7 Reference manual and programming help

There is an on-line reference manual for VPython. In the text editor (VIDLE), on the Help menu, choose "Visual" for information about 3D objects, or choose "Python Docs" to obtain detailed information on the Python programming language upon which VPython is based. We will use only a small subset of Python's extensive capabilities.