



COMP2720: Automating Tools for New Media

Introduction to Visual Python
(VPython)

A 3D graphics module

- VPython is a module suited for 3D interactive models of physical systems.
- Allows us to create, animate, and control 3D objects, and navigate around a 3D scene (spinning and zooming).
- VPython homepage: <http://www.vpython.org>
(docu at: <http://www.vpython.org/webdoc/index.html>).
- It is based on Python, Visual (a 3D graphics module for Python), and IDLE (programming environment, similar to JES, called so because:
either (poorly) **I**ntegrated **D**evelopment **E**nvironment,
or after the actor Eric Idle, one of the Monty Pythons).



A new programming environment

- VPython is NOT part of JES
 - Based on standard Python (similar to standard library modules).
- Import visual module using:

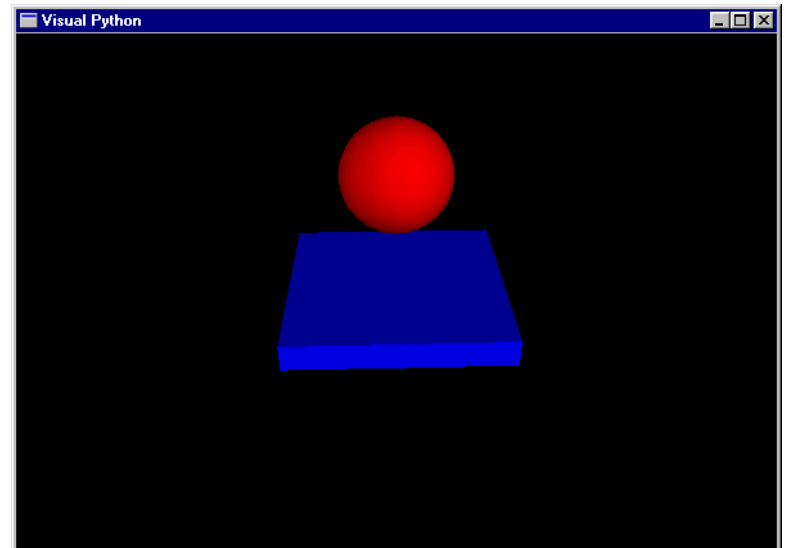
```
>>> from visual import *
```
- Simple test with:

```
>>> sphere()
```

VPython example program

- Taken from the VPython Web site

```
from visual import *
floor = box (pos=(0,0,0), length=4, height=0.5, width=4, color=color.blue)
ball = sphere (pos=(0,4,0), radius=1, color=color.red)
ball.velocity = vector(0,-1,0)
dt = 0.01
while 1:
    rate (100)
    ball.pos = ball.pos + ball.velocity*dt
    if ball.y < ball.radius:
        ball.velocity.y = -ball.velocity.y
    else:
        ball.velocity.y = ball.velocity.y - 9.8*dt
```



VPython basics

- The *display* window

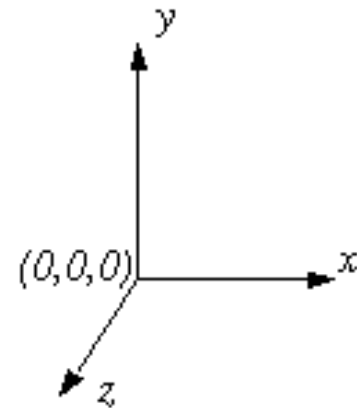
- Where the 3D scene is shown.
- $(0,0,0)$ is in the centre.
- *x-axis* runs from left to the right.
- *y-axis* runs from down to up.
- *z-axis* points towards screen.

- The *output* window

- Where text is printed (print statement).

- Mouse controls

- Click-right button (and drag): Rotate around in scene (Mac: Shift-click).
- Click-middle button (and drag): Zoom-in and out (Mac: Control-click).

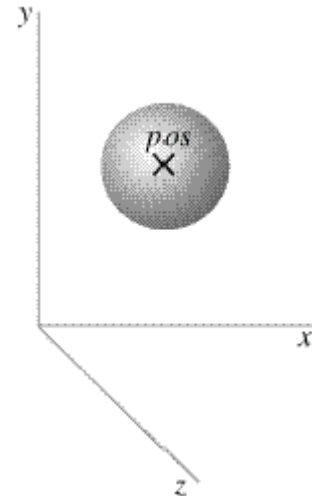


3D Vectors

- VPython uses vectors (for positions, directions, etc.)
- Define vectors: $a = \text{vector}(1,2,3)$, $b = \text{vector}(5,6,8)$
- Vector operations:
 - Addition: $c = a + b$ (c will be a vector with components (6,8,11))
 - Multiplication: $d = 3 * a$ (d will be a vector with components (3,6,9))
 - Magnitude: $e = \text{mag}(a)$ (e will be a scalar of value $\sqrt{1*1+2*2+3*3}$)
 - Vector products, etc.
- Access vector components with $v.x$, $v.y$, and $v.z$
 - In our examples: $a.z = 3$, $c.y = 8$, $d.z = 9$
 - Possible to set vector components: $a.z = 42$

VPython objects

- Various objects are available
cylinder, arrow, cone, pyramid, sphere, ring, box, ellipsoid,
curve, (helix), and convex
- All have a set of common attributes
 - pos – Object position (note that for some objects pos is in their centre, while for others it is not).
 - color – Object color (*red, green, blue* values).
 - visible – 1=visible, 0=not visible.
- Plus more object specific attributes...
- Example: `ball = sphere(pos=(1,2,1))`



Setting colors

- Objects have color attributes:
 - `object.color = (red, green, blue)`
 - `object.red = red`
 - `object.green = green`
 - `object.blue = blue`
- Color values are between 0.0 and 1.0
 - Not 0 and 255 as in JES!
 - 0.0 in VPython is the same as 0 in JES
 - 1.0 in VPython is the same as 255 in JES
 - For example:
`ball = sphere(color=(1.0,0.5, 0.0)) # Orange ball`
 - Run demo program `colorsliders.py`

arrow and box objects

- arrow attributes:

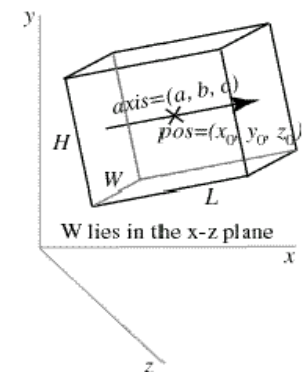
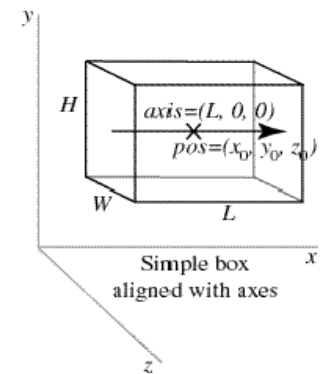
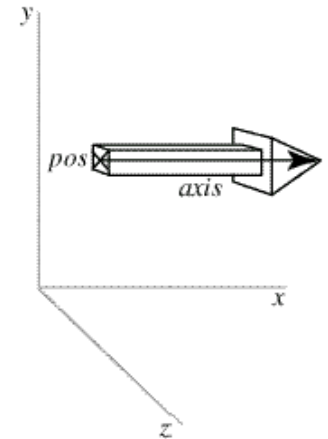
- pos (a vector) is at the centre at the end of the shaft.
- axis (a vector), points from pos to arrow head.
- shaftwidth, headwidth, headlength (three scalars that define the shape of the arrow).
- up (a vector), direction of arrow head.

- Example:

`whitePointer = arrow(pos=(1,2,0), color=(1,1,1), up=(0,1,0))`

- box attributes:

- pos (a vector) is at the centre of the box (!)
- axis (a vector), orientation (and length) of a box.
- height, length, width (three scalars) that define the shape of a box.



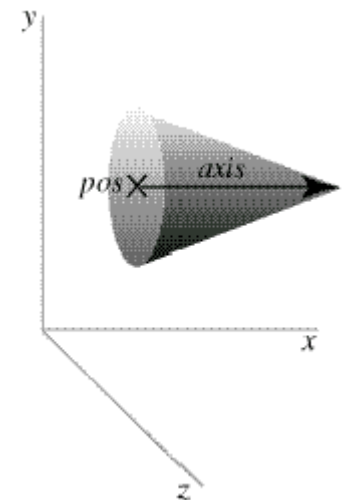
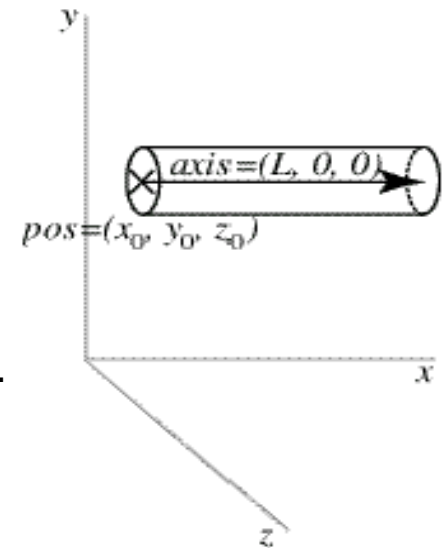
cylinder and cone objects

- cylinder attributes:

- pos (a vector) is at centre of one end of the cylinder.
- axis (a vector), points from pos to the other end of the cylinder (length of cylinder).
- radius and length (two scalars), length can override axis.

- cone attributes:

- pos (a vector) is at centre of the base of the cone.
- axis (a vector), points from pos to the point of the cone.
- radius (a scalar), radius of the wide end of the cone.



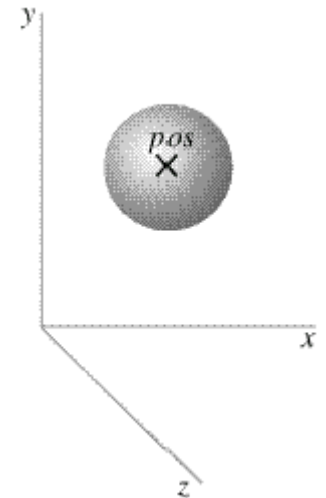
pyramid and sphere objects

- pyramid attributes:

- pos (a vector) is centre of the base of the pyramid.
- axis (a vector), points from pos to the point of the pyramid.
- height and width (two scalars), or size (a vector: *length, height, width*), size of a pyramid. Note: length is from base to tip, size of the base rectangle is height x width.

- sphere attributes:

- pos (a vector) is at centre of sphere (!)
- radius (a scalar), radius of the sphere.



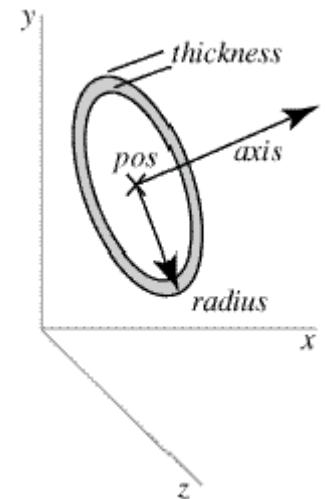
ellipsoid and ring objects

- ellipsoid attributes:

- A long ellipsoid looks like a cigar, a short one like a pill.
- Has same attributes as `box` object (can be thought as fitting inside a box).

- ring attributes:

- `pos` (a vector) is at centre of ring (!)
- `axis` (a vector), direction of the ring.
- `radius` (a scalar), outer radius of the ring.
- `thickness` (a scalar), thickness of the ring.



curve object

- Displays straight lines between points (appearance of a smooth curve if points are close together).
- curve attributes:
 - pos (a list of 3D vectors).
 - x, y, z (lists of the components of the points).
 - radius (a scalar), radius of the cross section of the curve.
- Use `arange()` to create arrays of points (similar to `range()`)
- Example:

```
c = curve(x = arange(0, 20, 0.1), color = (1,0,1)) # Magenta
```

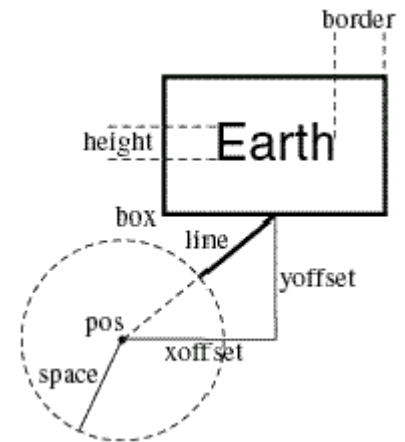
```
c.y = sin(2.0*c.x)
```

```
c.z = cos(2.0*c.x)
```

```
c.radius = 1.5
```

label object

- label attributes:
 - Display text in a box (always the same size, independent of zoom).
 - pos (a vector), the position being labeled.
 - xoffset, yoffset (two scalars), components of line, in pixels (!)
 - text (a string), the text to be displayed.
 - height (a scalar), size of text.
 - box (boolean), 1=draw box, 0=don't draw.
 - opacity (a scalar), 0.0=transparent, 1.0 opaque.
 - line (boolean), 1=draw line, 0=don't draw.
 - space (a scalar), radius of a sphere around pos, into which the line is NOT drawn.





convex object

- The `convex` object takes a list of points (like the `curve` object) and creates an object that is convex everywhere.
- Any points that would make an object bulge inwards (concave) are discarded.
- See `convex.py` demo.

Rotating objects

- Most objects (arrow, box, cone, cylinder, ellipsoid, pyramid, ring, and sphere; but not curve and convex) can be rotated:
`object.rotate(angle=pi/4.0, axis=axis, origin=pos)`
- rotate method applies a transformation to the object, with
 - rotation in angle radians ($0.0 \dots 2.0 \cdot \pi$),
 - counter-clockwise around the line defined by origin and origin+axis,
 - by default, rotations are around the object's own pos and axis.
- **Example:**

```
b= box()
for i in range(300):
    rate(50)
    b.rotate(angle=pi/100.0)
```

Composite objects with frame

- Often one wants to group objects together.
 - So that they can be moved or rotated as if they were one object.
- To do so, create a frame object.

```
f = frame()
```

```
cylinder(frame=f, pos=(0,0,0), radius=0.1, length=1, color=color.cyan)
```

```
sphere(frame=f, pos=(1,0,0), radius=0.2, color=color.red)
```

```
f.axis = (0,1,0)
```

```
f.pos = (-1,0,0)
```

```
for i in range(300):
```

```
    rate(50)
```

```
    f.rotate(angle=pi/50.0, axis=(1,1,0))
```

Limiting animation rate

- How to make sure an animation runs at the same speed on different computers (with different CPU speeds)?
 - Limit the *animation rate* (number of times per second a scene is re-calculated and re-drawn on screen).
- Use: `rate(frequency)`
 - Halts computation until $1/\text{frequency}$ seconds after previous call to `rate()`
 - No halt is performed if this time has already elapsed.
 - Important: Place *inside* the loop that computes animation.

More VPython demos

- More example programs on COMP2720 Web site
- Go to **Further Links**, then **course related material**.
- Copy into your local directory
- Start a demo using Python (might not work on all machines, depending upon if VPython is installed):
`python bounce2.py`
- Vpython is currently installed on all ANU InfoCommons lab PCs and Macs, and of course in CS labs.
- Read programs and try to understand them!

What to do now, and next week

- Go to VPython Web site (www.vpython.org)
 - Read through *Overview* and *Tutorial Introduction*.
- Work on labs (the lab 6 in the week 11 is on VPython).
- Work on assignment 2 (due Saturday **18 October 6 pm**)
- Next: Guest lecture by *Hugh Fisher* (more on 3D Python)
- New lecture on Movies next Friday (Sept. 19)
- Next week: Start the last module **Computer Science topics**