

Lec2 - More mechanics

- Summary of lec1
- $D > 1$. Drawing tracks
- Python functions
- Lissajou figures

Summary

- What is Computational Science ?
 - Discipline concerned with solving mathematical models of physical systems using computers
 - Small systems – “exact” solutions.
 - Large systems – simulations. Modeling. New types of phenomena not visible with few d.o.f
- Earliest origins in trying to predict motions in solar system - mechanics + gravity
- Mechanics: simplest method or *algorithm* to solve Newton’s laws on computer: *Euler method*

- Code in python ...

Python code

Create a code euler.py. Inside

```
from visual import *
# no automatic scaling of window
scene.autoscale=0
# size of window
scene.range=10.0
# standard Python 3D object
ball=sphere(radius=0.5,pos=(0,8.0,0))
ball.vel=vector(0,0,0)
dt=0.01
t=0
# loop over time (for ever!)
while(1):
    rate(100)
    t=t+dt
    ball.pos=ball.pos+ball.vel*dt
    ball.vel=ball.vel-ball.pos*dt
```

Comments

- `import` statement allows us to use the 3D graphics module called `VPython`.
- No automatic scaling of window to simulation. Fix size of window with `range` explicitly.
- `rate` is a Python function which controls the speed at which graphics is drawn to the screen. A large argument means a faster simulation in real time.
- Simple assignment and arithmetic. The statement `t=t+dt` should be read as “ add the variable `dt` to the variable `t` and put the result back in the variable `t`”.

VPython Objects

- `sphere` is a standard Python *object*. It may have a number of attributes such as position, color, size etc. You may also add attributes such as velocity (`vel`) which is defined to be a vector.
- `ball` is the name of a specific one of these `sphere` objects created in this code with the line
`ball=sphere(...)`
- Attributes such as `pos` are accessed using the dot operator eg. `ball.pos`.

Control and Indentation

`while(arg):` command tells the computer to do something until `arg` is true (1 in computer speak).

Structures like `while` typically act on a group of Python commands

Notice the colon

You can tell which ones by the level of indentation.

Eg. in the last code the lines

```
    rate(100)
    t=t+dt
    ball.pos=...
    ball.vel=...
```

are all carried out in the `while` loop.

IDLE (Python editor) automatically indents codes that follows such a command.

2D Motion

Identical equations for all components eg (x, y)
and (v_x, v_y) eg. new equations for y -motion

$$\frac{dy}{dt} = v_y$$
$$\frac{dv_y}{dt} = F_y/m$$

Same Python code will automatically integrate them!

Just need to change initial conditions to have $v_y(t = 0) \neq 0$ and add y -component to force
What about 3D ??

General forces

Nice to rewrite Python code to make it easier to keep the force definition separate from the Euler update.

```
...
from force import *
while(1):
    rate(100)
    t=t+dt
    ball.pos=ball.pos+ball.vel*dt
    ball.vel=ball.vel+
    force(ball.pos,ball.vel)/ball.mass*dt
```

Need a new code in file force.py

Force function

```
# 2d oscillator
def force(pos,vel):
    result = vector(0,0,0)
    result.x=-1.0*pos.x
    result.y=-1.0*pos.y
    result.z=0
    return result
```

Comments

- `def force()` is followed by code *defining* what the function does.
- Note indentation showing the lines of code inside the definition.
- `force` function takes *vector arguments* and returns object `result` of type `vector`
- `result.x` gives x-component of vector.
- `return` is *keyword* (like `import`, `def`, `while` etc)

Drawing the path

Nice with 2D,3D problems to visualize the trajectory. Need the `track` attribute which adds an object of type `curve` which is standard in VPython.

```
....
ball=sphere(radius=0.5,pos=(0,8.0,0),
track=curve(radius=0.1))
ball.vel=vector(5,5,0)
dt=0.01
t=0

while 1:
    rate(100)
    t=t+dt
    ball.pos=ball.pos+ball.vel*dt
    ball.vel=ball.vel+force*dt
    ball.track.append(pos=ball.pos)
```

curve()

- `curve()` is really a list of coordinate points which can be dynamically added to.
- Also knows how to plot itself to the screen
- Can be given attributes like `radius`, `color` etc
- Like other standard Vpython objects can be added as an attribute to `sphere()`

Lissajou figures

- Imagine using different harmonic springs in each direction. What is resulting motion ?
- Does motion close on finite length trajectory ? When ? What condition is needed ...
- Finite orbits termed *Lissajou figures*

Summary

- Indentation. Allows sections of code to be *under control* of python command. Eg in loops where same sequence of commands executed many times.
- Functions. Separate off code for particular force from code to integrate Newton's laws. Easier to debug. Less code to write when tackling new situation.
- 2D,3D. Drawing tracks.
- Playing with harmonic forces ...