This week's classes

Homework #11, due 11/25

Notice that the value of (dimension + eyes) is conserved!

Alien playground

What data does a computer AI player need?

Connect 4 Player
Twitter's power?

a public API

Object-oriented philosophy...

Consider the tweet. It’s short—140 characters and done—but hardly simple. If you open one up and look inside, you’ll see a remarkable clockwork, with 31 publicly documented data fields. Why do these tweets, typically born of a stray impulse, need to carry all this data with them?

While a tweet thrives in its timeline, among the other tweets, it’s also designed to stand on its own, forever. Any tweet might show up embedded inside a million different websites. It may be called up and re-displayed years after posting. For all their supposed ephemerality, tweets have real staying power.

Once born, they’re alone and must find their own way to the world, like a just-hatched sea turtle crawling to the surf. Luckily they have all of the information they need in order to make it: A tweet knows the identity of its creator, whether bot or human, as well as the location from which it originated, the date and time it went out, and dozens of other little things—so that wherever it finds itself, the tweet can be reconstituted. Millennia from now an intelligence coming across a single tweet could, like an archaeologist pondering a chunk of ancient skull, deduce an entire culture.
Tuples are similar to lists, but they're parenthesized:

\[ T = (4, 2) \quad V = (1, 0, 0) \]

example of a two-element tuple named T and a three-element tuple named V
Tuples!

Lists that use parentheses are called **tuples**:

```python
>>> T = (4, 2)
>>> T
(4, 2)
>>> T[0]
4
>>> T[0] = 42
Error!
>>> T = ('a', 2, 'z')
```

Tuples are **immutable** lists: you can't change their elements...

...but you can always redefine the whole variable, if you want!

+ Tuples are more memory + time efficient
+ Tuples can be dictionary keys; lists can't
- **But you can't change tuples' elements**...
Tuple problems...

A bug from yesterday's Board class:

```python
W = 4
s = " ",
for col in range(W):
s += str(col), " 
```

yields a surprising result for `s`:
Tuple problems...

A bug from yesterday's Board class:

\[
W = 4 \\
S = \"\"\, \\
\text{for } \text{col} \text{ in range}(W): \\
s += \text{str}(\text{col}), \"\"\ \\
\]

yields a surprising result for \( s \):

\((\ ' ', \ '0', \ ' ', \ '1', \ ' ', \ '2', \ ' ', \ '3', \ ' ')\)
Python details used in VPython...

Functions can have *default input values* and can take *named inputs*

```python
def f(x=9, y=33):
    return x + y
```

example of default input values for x and y
Python details used in VPython...

Functions can have *default input values* and can take *named inputs*

```python
def f(x=9, y=33):
    return x + y
```

*example of default input values for x and y*

```python
f()
```

*example of a named input*

```python
f(y=1)
```

```python
f(1)
```
def f(x=2, y=0):
    return x*(1+4*y)

What will these function calls to $f$ return?

<table>
<thead>
<tr>
<th>$f(3, 2)$</th>
<th>$f(3)$</th>
<th>$f()$</th>
<th>$f(y=3, x=1)$</th>
</tr>
</thead>
</table>

What is a call to $f$ that returns 42?  
What is the shortest call to $f$ returning 42?

Extra!
feedback feedback
What is something you'd keep about CS5 ...?
CS5! Picobot.

What is something you'd change about / get rid of / add to CS5 ...?
PICOBOT

What is something you'd change about / get rid of / add to CS5 ...?
I didn't like Picobot, and I would not be sad to see it go
What is something you'd keep about CS5 ...?

CS5! Picobot.

What is something you'd change about / get rid of / add to CS5 ...?

Picobot

I didn't like Picobot, and I would not be sad to see it go.

Something you'd keep about CS5 ...?

Recursion + all the recursion practice

/ get rid of / add to CS5 ...?

Recursion! 😞
feedback

Something you'd keep about CS5 ...?
I really enjoyed logicism.

Something you'd keep about CS5 ...?
I really like all the material we're studying, especially logicism.

Something you'd change about / get rid of /
Circuits :

Something you'd change about / get rid of /
No more logicism!

Something you'd change about / get rid of / add
More projects, less logicism.
Something you'd *change about / get rid of / add to* CS5 ...?

- More music

I particularly liked Logisim, hummm.

Something you'd *keep about* CS5 ...

get rid of sound editing (Star Wars sounds)
feedback?

Something you'd keep about CS5 ...?
I don't find the hardware stuff not that interesting.

Something you'd change about / get rid of / add to CS5 ...?
I don't really like the readings, but don't get rid of it.

Hmmm, nothing.

Something you'd change about / get...

More candy.

Other thoughts optional, but 142% welcome:

Remember to carry your towel.
Something you'd **keep** about CS5 ...?
I don't find the hardware stuff not that interesting.

Something you'd **change about / get rid of / add to CS5 ...**?
Something you'd change about / get rid of / add to CS5 ...?
The project portion of CS at the end of the semester

Other thoughts optional, but 142% welcome:

More candy.

Other thoughts optional, but 142% welcome:

Remember to carry you
<table>
<thead>
<tr>
<th>Question</th>
<th>Mean</th>
<th>St Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>On average how much time per week do you spend on CS5 outside class + lab?</td>
<td>4.51hrs.</td>
<td>1.9 st dev.</td>
</tr>
<tr>
<td>How does CS5’s workload compare to other classes you're taking this term?</td>
<td>2.90</td>
<td>0.90 st dev.</td>
</tr>
<tr>
<td>How would you judge the pace of CS5?</td>
<td>3.03</td>
<td>0.40 st dev.</td>
</tr>
</tbody>
</table>
### Feedback

<table>
<thead>
<tr>
<th>Question</th>
<th>1.5</th>
<th>2.90</th>
<th>3.03</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>much lighter</td>
<td>about the same</td>
<td>much heavier</td>
</tr>
<tr>
<td>2.90</td>
<td>much too slow</td>
<td>about right</td>
<td>much too fast</td>
</tr>
<tr>
<td>3.03</td>
<td>much too slow</td>
<td>about right</td>
<td>much too fast</td>
</tr>
</tbody>
</table>

- **On average how much time per week do you spend on CS5 outside class + lab?**
- **How does CS5's workload compare to other classes you're taking this term?**
- **How would you judge the pace of CS5?**
## Feedback

<table>
<thead>
<tr>
<th>Question</th>
<th>1st Year</th>
<th>2nd Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>On average how much time per week do you spend on CS5 outside class + lab?</td>
<td>4.51 hrs.</td>
<td>2.90</td>
</tr>
<tr>
<td>1.9 st dev.</td>
<td>0.90 st dev.</td>
<td>0.40 st dev.</td>
</tr>
</tbody>
</table>

How does CS5's workload compare to other classes you're taking this term?

How would you judge the pace of CS5?

More pop tarts
feedback feedback

end with libraries
VPython

3D Programming for Ordinary Mortals

built by and for physicists to simplify 3d simulations

lots of available classes, objects and methods in its API

Easily installable for windows... and mostly easy on Macs.
Installing VPython

Windows: www.vpython.org/contents/download_windows.html

Mac: http://www.vpython.org/contents/download_mac.html

I've tried both of these and they worked so far...

Tutorials and documentation...
API

... stands for *Application Programming Interface*

a description of how to use a software library

A demo of VPython's API:

```python
from visual import *
c = cylinder()
```

What's `cylinder`?
What's `visual`?
What's `c`?

VPython example API calls: must be from a *file*
from visual import *
c = cylinder()
print "c.pos is", c.pos
print "c.color is", c.color
# set the color to color.blue or a tuple
# set the pos... hard to tell what's happening...
scene.autoscale = False
b = box(pos=(4,0,0))
a = sphere(pos=(0,0,4))

while True:
    rate(100)  # limits the loop rate in hz
dt = 0.01   # the loop time
    a.pos += dt*vector(-5,0,0)
from visual import *

floor = box( length=4, width=4, height=0.5, color=color.blue )
bball = sphere( pos=(0,8,0), radius=1, color=color.red )

vel = vector(0,-1,0)
dt = 0.01

while True:
    rate(100)
    ball.pos += vel*dt

    if ball.pos.y < ball.radius:
        vel.y *= -1.0
    else:
        vel.y += -9.8*dt

What physics is this if/else doing?

with the Higgs boson!

Look over this VPython program to determine:
(0) How many tuples appear in this code? ________
(1) How many classes are used here? ________
(2) How many objects are used here? ________
(3) How do collisions work?
(4) How does physics/gravity work?

What physics is this if/else doing?

with the Higgs boson!
vectors

vel = vector(0, -1, 0)

vel += 0.01*vector(0, -9.8, 0)

pos = pos + 0.01*vel
Orbiting

```
from visual import *

e = sphere(pos=(0,0,10),color=color.blue)  # earth
s = sphere(color=color.yellow,radius=2)    # sun

e.vel = vector(5,0,0)       # initial velocity

RATE = 50
dt = 1.0/RATE
k = 70.0                    # G!

while True:
    rate(RATE)
    diff = s.pos - e.pos      # vector difference
    force = k*diff/(mag(diff)**2)  # mag
    e.vel += dt*force         # acceleration d.e.
    e.pos += dt*e.vel         # velocity d.e.
```
from visual import *

class Alien:
    """ This class represents a three-eyed alien object... """

    # The constructor, named __init__ (as always in Python)
def __init__(self, init_framepos):
        """ The constructor creates a frame (container)
        at initial location init_framepos """

        # a frame is VPython's collection of shapes
        # within a single coordinate system
        self.f = frame(pos=init_framepos)

        # all of these parts are within the frame self.f
        self.body = sphere(pos=vector(0, 0, 0),
                           radius=1,
                           color=color.green,
                           frame=self.f)

        self.left_eye = sphere(pos=self.body.pos + vector(.35, .5, .6),
                                radius=0.20,
                                color=color.white,
                                frame=self.f)

        self.right_eye = sphere(pos=self.body.pos + vector(-.35, .5, .6),
                                 radius=0.20,
                                 color=color.white,
                                 frame=self.f)
if scene.kb.keys:                    # is there a keyevent?
s    s = scene.kb.getkey()       # get keypress
if s == "p":
    print alien

# things the alien(s) can do!
if s == 'J':                    # JUMP!
    alien.f.pos = vector(0,HEIGHT,0)
    alien.vel = vector(0,0,0)       # stop the alien!
    run_gravity = not run_gravity  # fun!
    print "run_gravity is", run_gravity

# move the alien around
if s == "i":
    alien.f.pos += vector(0,0,1)
if s == "k":
    alien.f.pos += vector(0,0,-1)
if s == "j":
    alien.f.pos += vector(-1,0,0)
if s == "l":
    alien.f.pos += vector(1,0,0)

Note that the frame is being moved here ~ this moves all of the parts!
while True:
    rate(RATE)
    # Here begins PHYSICS!
    if run_gravity == True:
        alien.update(dt)
        alien.check_beach(beach)
    # Here ends physics...

def update(self, dt):
    """ this defines the physics... """
    gravity = -9.8*10
    self.vel += dt*vector(0,gravity,0)
    self.f.pos += dt*self.vel

def check_beach(self, beach):
    """ checks for a bounce! """
    bottom_of_self = self.f.pos.y - self.body.radius
    if bottom_of_self < beach.pos.y:
        self.f.pos.y = beach.pos.y + self.body.radius
        self.vel.y *= -1.0
Looking back (before looking forward...)
Lab goals

(0) Try out VPython!

(1) Implement air resistance...

(2) Add at least 1 more dimension

(3) Add a target and initial velocity

(4) *Improve your character!*

(5) Add scoring or enemies or a moving target, hoops, traps, holes, etc. ~ *your own game*...
Next time...

What data does a computer AI player need?

An AI for Connect Four
Phunky Physics!

-- falls through
-- loses energy
-- perfect collisions – still imperfect – why?

-- air resistance