What we'll be doing today

- Downloading a graphics module
- Looking at some basic graphics commands in the *graphics.py* module
- Practice using those commands to draw some purty pictures.

graphics.py

Get *graphics.py* from our server with the following command:

```
$ scp a_student@crashwhite.polytechnic.org:/home/rwhite/
Public/graphics.py ~  # What does this do?
```

graphics.py is a module (which is just a program) that you can import into your programs.

graphics.py

Non-standard libraries that you'll be importing into a program need to be downloaded onto your computer and placed in one of two places:

a. in the same directory as the program that imports it, or

b. in a standard system-level directory where other Python libraries are stored.

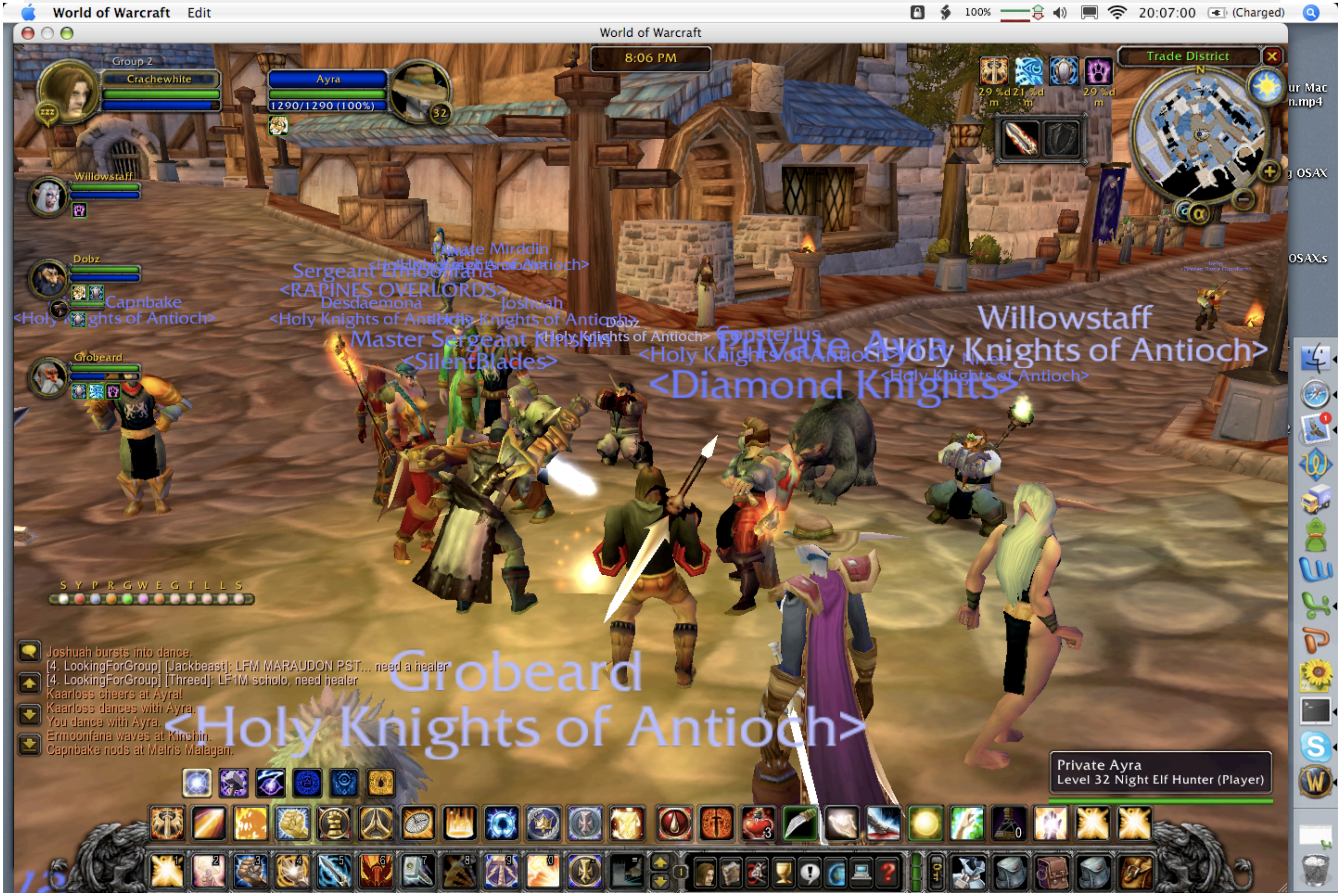
Find path for python by running Python,

```
>>> import sys
>>> sys.path
```

Look for "site-packages", then move *graphics.py* there.

```
$ mv graphics.py /Library/Frameworks/...../site-packages
```

WoW



Joshuah bursts into dance.
[4. LookingForGroup] [Jackbeast]: LFM MARAUDON PST... need a healer
[4. LookingForGroup] [Threed]: LF1M scholo, need healer
Kaarloss cheers at Ayra!
Kaarloss dances with Ayra.
You dance with Ayra.
Ermoonfana waves at Kinshin.
Capnbake nods at Melris Malagan.

Zelle’s graphics.py

Once you’ve imported graphics into your program, you can use it to create all sorts of simple figures.

1. In Python interactive, `import graphics`
2. Enter `dir(graphics)` to get a list of methods.
3. Enter `help(graphics)` for some documentation
4. Enter `print graphics.__doc__` and see what comes up

graphics.py commands

Some useful commands (a full list is in your book):

```
graphics.GraphWin(title, width, height) # creates window
close()                                             # closes window
plot(x, y, color)                                   # draws pixel at x, y
```

Drawable objects include `Point`, `Line`, `Circle`, `Oval`, `Rectangle`, `Polygon`, and `Text`.

```
Point(x,y)
Line(point1, point2)        # draws a line between given points
Circle(centerPoint, radius)
Rectangle(point1, point2)
Oval(point1, point2)
Text(anchorPoint, string)
setFill(color)          # set the fill color of the object
setOutline(color)       # set the color of the outline
setWidth(pixels)        # set the width of the outline
draw(aGraphWin)         # Draws the object in the window
move(dx,dy)             # Moves the object the specified units
clone()                 # Return an (as yet undrawn) duplicate
```

Getting started

```
# graphics_demo.py
from graphics import * # get the graphics.py module
win = GraphWin("My Example",640,480)
win.setBackground("gray")
center = Point(100,100)
circ = Circle(center, 40)
circ.setFill("red")
circ.draw(win)  # You have to "draw" the circle for it to finally appear
label = Text(center, "Hi!")
label.draw(win) # Have to draw the text for it to appear
rect = Rectangle(Point(30,30), Point(70,70))
rect.draw(win)
for y in range(100):
    rect.move(2, 5)
line = Line(Point(639,0),Point(0,479))
line.draw(win)
for i in range(100):
    line.move(-1,0)
for i in range (320):
    line = Line(Point(i*2,0),Point(0,479))
    line.setFill('blue')
    line.draw(win)
    line.undraw()
input=raw_input("Press [Enter] to quit.")
win.close # closes the window
```

Try this

Write a program to...

- Draw a picture of a house
- Draw a picture of a smiley face with your name underneath.
- Draw a volleyball flying as a projectile from one side of the screen to the other. (Specify initial x-y coordinates, initial x-y velocities, factor in the x-y acceleration, and let physics be your guide.) Print the x-y components of the velocity at the top of the screen as the volleyball moves.

More graphics features

Some of you have already seen some of the text features of our the graphics package:

```
>>> from graphics import *
>>> w = GraphWin("My window",500, 300)
>>> t = Text(Point(100,20),"Hello")   # Center-justified
>>> t.draw(w)                         # Draw text
>>> t.setFace("helvetica")
>>> t.setText("Goodbye")
>>> t.setSize(36)
>>> t.setColor("blue")
```

Getting text entry

```
from graphics import *
win = GraphWin("Accepting Text", 600, 400)
win.setCoords(0.0, 0.0, 6.0, 4.0)                          # So convenient!
Text(Point(1,3),"Enter something here:").draw(win)         # Just text
input = Entry(Point(2,3),5)                        # Create the input field
input.draw(win)                                    # Draw the input field
Text(Point(1,1),"Now click this button:").draw(win)
rect = Rectangle(Point(3,1.5),Point(4,1))            # Create the button
rect.draw(win)                                     # Draw the button
mssg = Text(Point(4,3),"")                         # Blank message so that
mssg.draw(win)                                     # we can undraw it later
while True:
    click = win.getMouse()
    if click.getX()>3 and click.getX()<4 and click.getY()<1.5 and 
click.getY()>1:
        mssg.undraw()
        mssg = Text(Point(4,3),"Thanks for clicking the button!")
        mssg.draw(win)
        response = input.getText()
        print "They entered the text",response
        break
    else:
        mssg.undraw()
        mssg = Text(Point(4,3),"You didn't click the button!")
        mssg.draw(win)
pause = raw_input("Click to finish")
```

Interactive graphics

What we really want, in the end, is for the user to be able to interact with graphics. *Events* like typing a key, moving the mouse, or clicking on the screen, can be used to drive a program.

Although graphics.py is a relatively simple module, it does facilitate a user entering text into the graphics window, and capturing mouse clicks...

Getting mouse clicks

`getMouse()` waits for the user to click a mouse in the window, & returns the Point where it was clicked.

```
>>> w = GraphWin()                  # Open up a window
>>> p1 = w.getMouse()               # Capture click event
>>> p1.getX(); p1.getY()            # Show coordinates
>>> p2 = w.getMouse()               # Get another click
>>> rect = Rectangle(p1,p2)         # Draw a rectangle...
>>> rect.draw(w)                    # ... using those clicks
```

Try this

Write a program that displays a "cards" face down on a graphics screen. When the user clicks on the card, it is "flipped over" to reveal a message on the underside. Clicking on the card again will flip it back over to hide the message.

Basic syntax

```
>>> from graphics import *      # graphics is a "module"
>>> w = GraphWin()               # GraphWin is a "class"
       # "GraphWin() is a "constructor" that creates "w"
         # The object "w" is an "instance" of that class
>>> p = Point(30,20)     # p.x is an "instance variable"
>>> p.getX()            # "getX" is a "method" of Point
>>> p.draw(w)
>>> p.undraw()

# Note that if you want to copy an object, you have to
# /clone/ it. Otherwise you just have two variables
# that point to the same object.

>>> q = p.clone()
```