Animations and Games in Calico

Doug Blank
Fall 2011
Introduction to Computing
Bryn Mawr College
Objects

- A new way of writing programs: *Objects*
- Object-Oriented Programming (OOP)
- Create instances of a *Class* using a *constructor*
  - *Instances* are nouns
  - Functions (verbs) of nouns, called *Methods*

```python
win = Window()
rec = Rectangle(((10, 10), (50, 50)))
rec.draw(win)
```
Window Modes

- Four modes of a Window:
  - "auto" - will update as quickly as possible, without overwhelming the system
  - "manual" - user controls the updates using win.step(), so that they can happen as slow or fast as desired
  - "physics" - objects have physical properties; must call win.run()
  - "bitmap" - a mode that will draw, but doesn't create objects that can be moved
Running a Window

- `win.run()` can be used when `win.mode` is set to “physics”
- Can also be used when you want to do the same thing in a window over and over

```python
win = Window()
win.mode = "manual"
def main():
    ...
    win.run(main)
```
- Takes a function of no arguments
Running a Window, cont.

```python
win = Window()
...

def main():
    win.mode = "manual"
    for s in range(360):
        sun.rotate(1)
        earth.rotate(5)
        win.step(.05)
        pen.appendPath(Point(moon.gx, moon.gy))

win.run(main)
```
Running a Window, cont.

```python
win = Window()
...

def main():
    win.mode = "manual"
    while True:
        sun.rotate(1)
        earth.rotate(5)
        win.step(.05)
        pen.appendPathPoint(moon.gx, moon.gy)

win.run(main)
```
Polling vs. Events

• How can you do something when:
  • The mouse is pressed?
  • The mouse is released?
  • A key is pressed?
  • A key is released?
  • A gamepad state is changed?

• Two ways:
  • Poll – “are you pressed?; are you pressed?...”
  • Event - “tell me when something happens”
Calico Allows Either

- Polling
  
  \[ \text{mxy} = \text{getMouseNow()} \]

- Event
  
  \[
  \text{def mouseDown(obj, event):}
  \]
  
  \[
  \quad \text{global dragging}
  \]
  
  \[
  \quad \text{if Point(event.x, event.y).distance(Point(ball.x, ball.y)) < radius:}
  \]
  
  \[
  \quad \quad \text{dragging = True}
  \]
  
  \[
  \quad \text{mouseMove(obj, event)}
  \]
  
  \[
  \text{win.onMouseDown(mouseDown)}
  \]