pygame!

but first: inheritance

```
class Point3D(Point): # Point3D extends Point
    z = 0             # add a z field
```

- python also supports *multiple inheritance*:

```
class Name(Superclass, Superclass, ...):
    ...
```

superclass methods

```
class Point3D(Point):
    z = 0

    def __init__(self, x, y, z):
        Point.__init__(self, x, y)
        self.z = z

    def translate(self, dx, dy, dz):
        Point.translate(self, dx, dy)
        self.z += dz
```

- use superclass name instead of `super`
- must explicitly pass `self`

- a set of python modules to make writing games easier
- deals with media (pictures, sound) and user input (keyboard, mouse, joystick) nicely

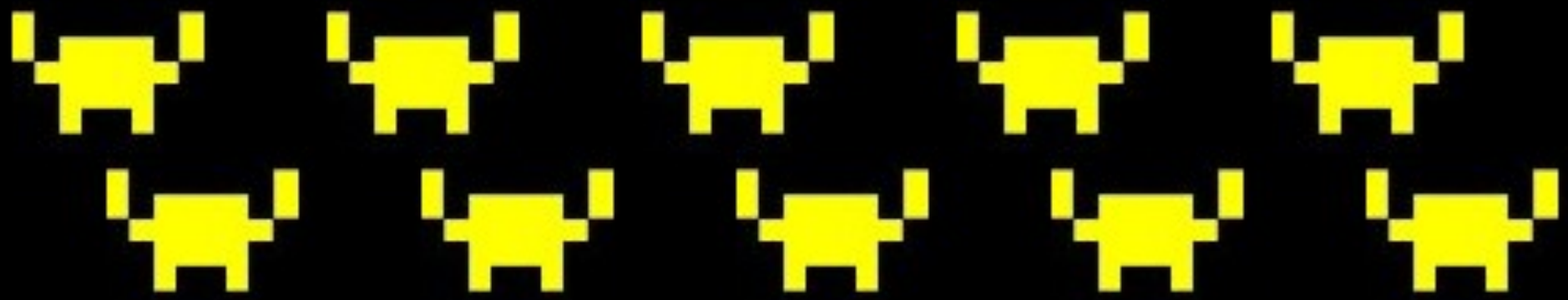
Simple Breakout

Deaths: 3

Points: 22

PyInvaders

Score 500 Highscore 0 Life 2 Bullets 5



installing pygame

- go to http://pygame.org/download.shtml
- download and run the latest installer
  - windows: `pygame-1.8.1.win32-py2.6.msi`
  - mac: `pygame-1.8.1release-py2.5-macosx10.5.zip`

pygame resources

- official docs: http://pygame.org/docs/
- tutorials: http://www.linuxjournal.com/article/7694
- experiment!

our goal: whack-a-mole

- clicking the mole plays a sound and makes the mole move
- number of hits is displayed at top of screen
- enhancements:
  - hit the mole with a shovel cursor
  - make the mole move every second

initializing a game

```
import sys
import pygame
from pygame import *
from pygame.locals import *
from pygame.sprite import *

pygame.init()
```

creating a window

```
display.set_caption("Whack-a-Mole")

screen = display.set_mode((640, 480))
# or
screen = display.set_mode((640, 480), FULLSCREEN)
```

so far

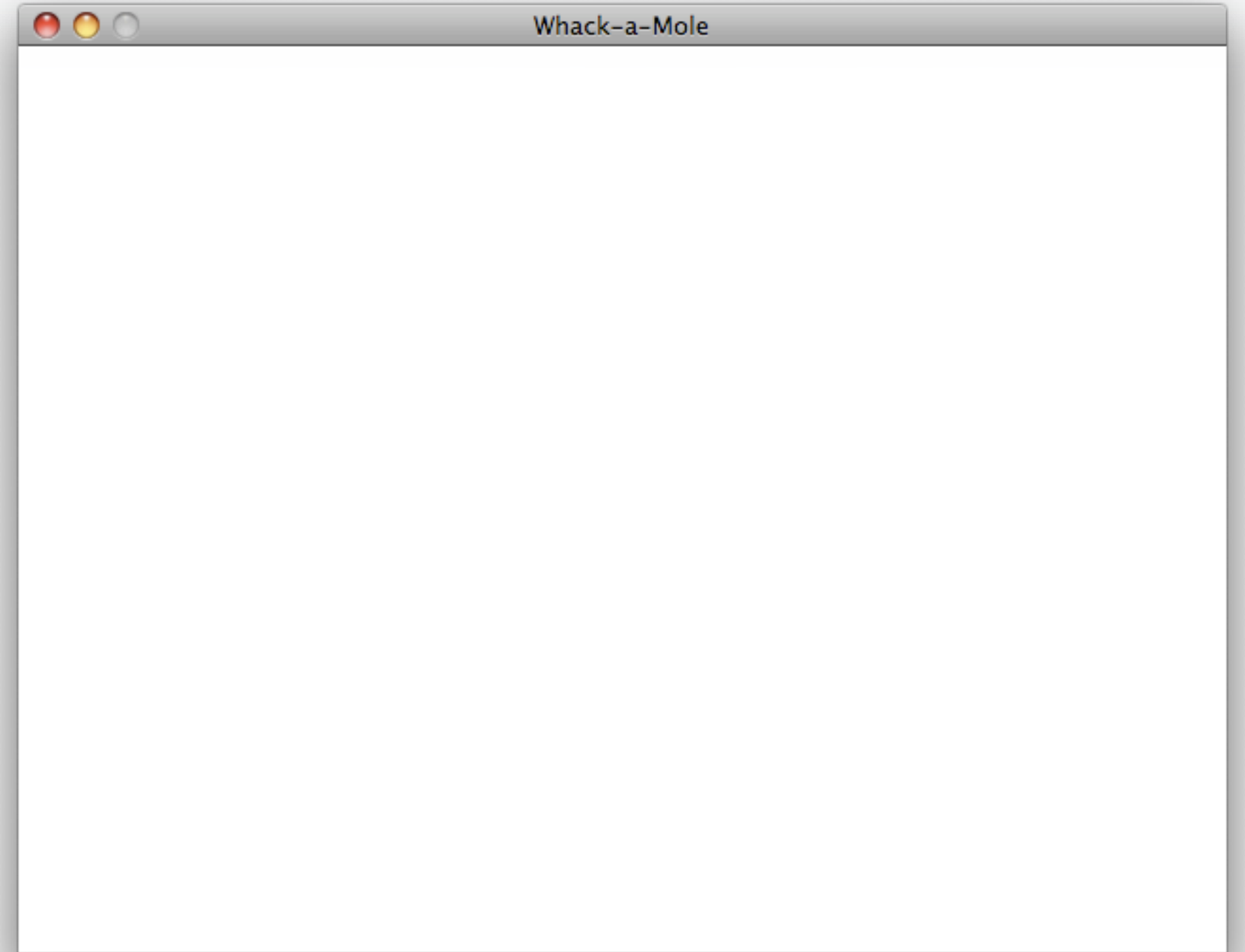
```
import pygame
from pygame import *
from pygame.locals import *
from pygame.sprite import *

pygame.init()

# set window title
display.set_caption("Whack-a-Mole")

screen = display.set_mode((640, 480))
```

so far



sprites

- a character, enemy, or other object in the game
- pygame sprites are objects extending the `Sprite` class

our first sprite

```
class Mole(Sprite):
    def __init__(self):
        Sprite.__init__(self)
        self.image = image.load("mole.gif")
        self.rect = self.image.get_rect()
```

- special fields in every sprite:
  - `self.image`: the image/shape to draw
  - `self.rect`: position and size of the sprite

sprite groups

```
my_mole = Mole()
all_sprites = Group(my_mole)
```

- to draw sprites on the screen, they must be put into a `Group`
- `Group` methods:
  - `draw(surface)` draws all sprites onto `surface`
  - `update()` updates every sprite's appearance

surfaces

- in pygame, every 2d object is a `Surface` object
  - the `screen` we got from `display.set_mode`
  - every sprite
  - every image

Surface methods

Method Name	Description
`fill((red, green, blue))`	paints surface in given color (rgb 0-255)
`get_width()`, `get_height()`	returns the dimensions of the surface
`get_rect()`	returns a `Rect` object representing the x/y/w/h bounding this surface
`blit(src, dest)`	draws this surface onto another surface

drawing and updating

```
my_mole = Mole()
all_sprites = Group(my_mole)
all_sprites.draw(screen)
display.update()    # redraw to see the sprites
```

- each `Surface` and `Group` object has an `update` method that redraws it to the screen
- once sprites are drawn, call `display.update()` to see the changes

so far

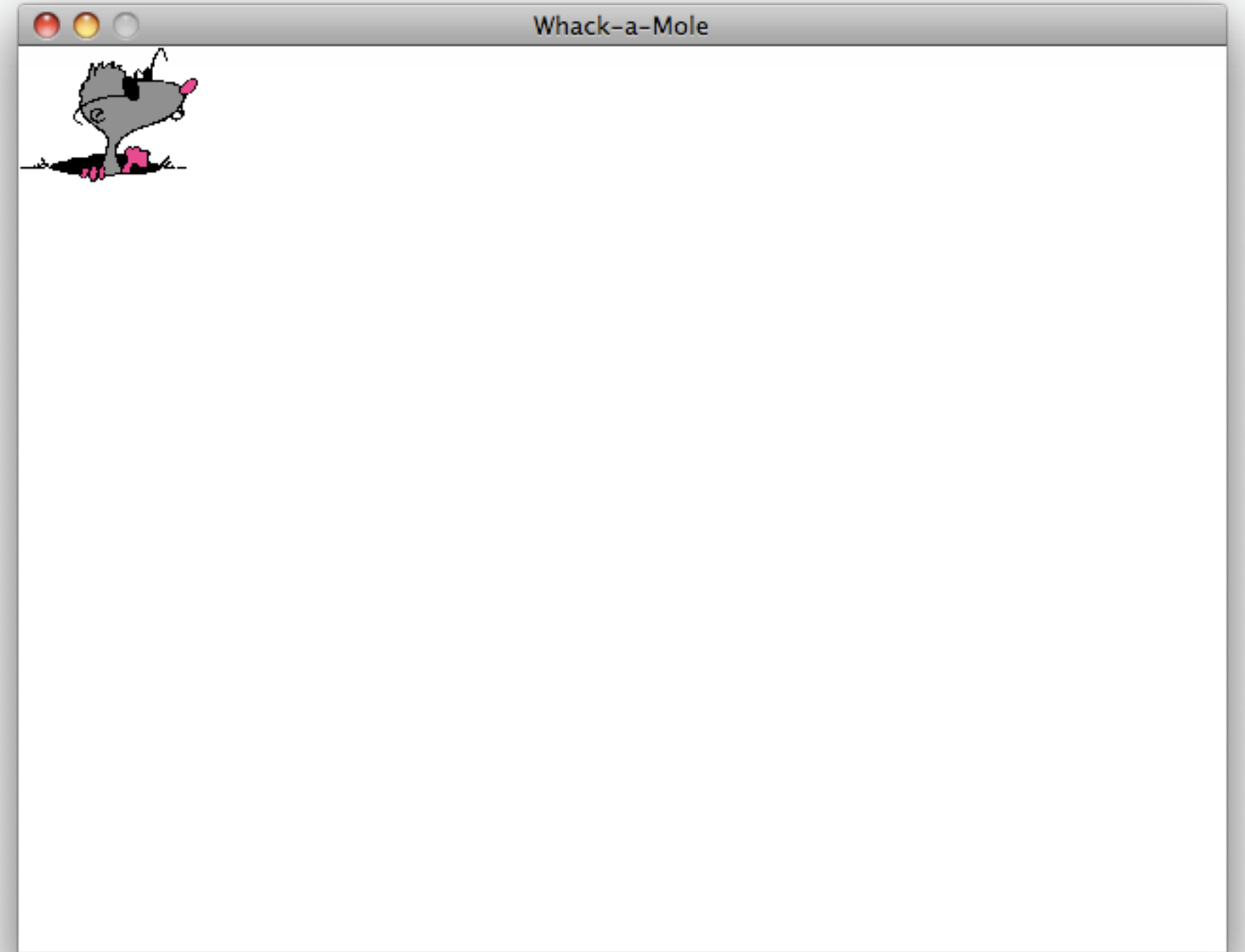
```
import pygame
from pygame import *
from pygame.locals import *
from pygame.sprite import *

class Mole(Sprite):
    def __init__(self):
        Sprite.__init__(self)
        self.image = image.load("mole.gif")
        self.rect = self.image.get_rect()

# main
pygame.init()
display.set_caption("Whack-a-Mole")
screen = display.set_mode((640, 480))

my_mole = Mole()                    # initialize sprites
all_sprites = Group(my_mole)
screen.fill((255, 255, 255))        # white background
all_sprites.draw(screen)
display.update()
```

so far



event-driven programming

- **event**: a user interaction with the game (a mouse click, key press, etc)
- an **event-driven** program waits for and responds to events
- pygame programs must be event-driven

event loop

```
while True:
    ev = event.wait()          # wait for an event
    if ev.type == QUIT:
        pygame.quit()          # exit the game
        break
elif ev.type == SOME_OTHER_EVENT_TYPE:
    ...
...
# code to update the screen
```

mouse events

- MOUSEBUTTONDOWN, MOUSEBUTTONUP, MOUSEMOTION
- mouse.get_pos() returns the cursor's current (x,y) position

```
...
elif ev.type == MOUSEBUTTONDOWN:
    # user pressed a mouse button
    x, y = mouse.get_pos()
...
```

keyboard events

- KEYDOWN, KEYUP
- ev.key will be K_LEFT, K_RIGHT, K_UP, K_DOWN, K_a - K_z, K_0 - K_9, K_F1 - K_F12, K_SPACE, K_ESCAPE, K_LSHIFT, K_RSHIFT, K_LALT, K_RALT, K_LCTRL, K_RCTRL, …

```
...
elif ev.type == KEYDOWN and ev.key == K_ESCAPE:
    pygame.quit()
...
```

collision detection

- noticing whether one sprite has touched another, and responding accordingly
- a major part of game programming
- in pygame, done by examining sprites, rectangles, and points, and asking whether they intersect

Rect objects

- a 2d rectangle associated with each sprite
- fields: `top, left, bottom, right, center, centerx, centery, topleft, topright, bottomleft, bottomright, width, height, size, ...`

Method Name	Description
`collidepoint(p)`	returns `True` if this `Rect` contains the point
`colliderect(rect)`	returns `True` if this `Rect` contains the rect
`contains(rect)`	returns `True` if this `Rect` contains the other
`move(x, y)`	moves this `Rect` to a new position
`inflate(dx, dy)`	grow/shrink this `Rect` in size
`union(rect)`	joins two `Rect`s

collision example

```
...
elif ev.type == MOUSEBUTTONDOWN:
    if sprite.rect.collidepoint(mouse.get_pos()):
        # then the mouse cursor touches the sprite
...
```

- exercise: make the mouse flee when the user clicks it

solution

```
class Mole(Sprite):
    def __init__(self):
        Sprite.__init__(self)
        self.image = image.load("mole.gif")
        self.rect = self.image.get_rect()

    def flee(self):
        self.rect.left = randint(0, 600)   # random location
        self.rect.top = randint(0, 400)

...

while True:
    ev = event.wait()
    if ev.type == QUIT:
        pygame.quit()
        break
    elif ev.type == MOUSEBUTTONDOWN:
        if my_mole.rect.collidepoint(mouse.get_pos()):
            my_mole.flee()
    ...
```