CPSC 231 Tutorial #19

michael-hung.ca/teaching

Reminders

TODAY

Quiz 10

THURSDAY

Quiz 10 Review

FRIDAY

Assignment 5 Paired Component

please don’t plagiarize.

student code

```python
import pygame
from pygame.locals import *

parameters_of_ttt   = "X"
on_course = [[None, None, None ],[ None, None, None ],[ None, None, None ]]
champion = None
initializing_ = 1

def drawing_board(game_home_):
    display_background = pygame.Surface (game_home_.get_size())
    display_background = display_background.convert()
    display_background.fill ((250, 250, 250))

    pygame.draw.line (display_background, (0,0,0), (100, 0), (100, 300), 2)
    pygame.draw.line (display_background, (0,0,0), (200, 0), (200, 300), 2)

    pygame.draw.line (display_background, (0,0,0), (0, 100), (300, 100), 2)
    pygame.draw.line (display_background, (0,0,0), (0, 200), (300, 200), 2)

    return display_background

def displayboard (game_home_, board):
    game_home_.blit (board, (0, 0))
    pygame.display.flip()

def boardposition (mouseX, mouseY):
    if (mouseY < 100):
        row = 0
    elif (mouseY < 200):
        row = 1
    else:
        row = 2

    if (mouseX < 100):
        col = 0
```

GitHub page from 2014

```python
# import necessary modules
import pygame
from pygame.locals import *

# declare our global variables for the game
XO   = "X"   # track whose turn it is; X goes first
grid = [ [ None, None, None ], \
         [ None, None, None ], \
         [ None, None, None ] ]


def initBoard(ttt):
    # initialize the board and return it as a variable
    # --------------------------------------------------------------------
    # ttt : a properly initialized pyGame display variable

    # set up the background surface
    background = pygame.Surface (ttt.get_size())
    background = background.convert()
    background.fill ((250, 250, 250))

    # draw the grid lines
    # vertical lines...
    pygame.draw.line (background, (0,0,0), (100, 0), (100, 300), 2)
    pygame.draw.line (background, (0,0,0), (200, 0), (200, 300), 2)

    # horizontal lines...
    pygame.draw.line (background, (0,0,0), (0, 100), (300, 100), 2)
    pygame.draw.line (background, (0,0,0), (0, 200), (300, 200), 2)

    # return the board
    return background
```

student code

```python
import pygame
from pygame.locals import *

parameters_of_ttt   = "X"
on_course = [[None, None, None ],[ None, None, None ],[ None, None, None ]]
champion = None
initializing_ = 1

def drawing_board(game_home_):
    display_background = pygame.Surface (game_home_.get_size())
    display_background = display_background.convert()
    display_background.fill ((250, 250, 250))

    pygame.draw.line (display_background, (0,0,0), (100, 0), (100, 300), 2)
    pygame.draw.line (display_background, (0,0,0), (200, 0), (200, 300), 2)

    pygame.draw.line (display_background, (0,0,0), (0, 100), (300, 100), 2)
    pygame.draw.line (display_background, (0,0,0), (0, 200), (300, 200), 2)

    return display_background

def displayboard (game_home_, board):
    game_home_.blit (board, (0, 0))
    pygame.display.flip()

def boardposition (mouseX, mouseY):
    if (mouseY < 100):
        row = 0
    elif (mouseY < 200):
        row = 1
    else:
        row = 2

    if (mouseX < 100):
        col = 0
```

GitHub page from 2014

```python
def showBoard (ttt, board):
    # redraw the game board on the display
    # ------------------------------------------------------------------
    # ttt   : the initialized pyGame display
    # board : the game board surface

    drawStatus (board)
    ttt.blit (board, (0, 0))
    pygame.display.flip()

def boardPos (mouseX, mouseY):
    # given a set of coordinates from the mouse, determine which board space
    # (row, column) the user clicked in.
    # ------------------------------------------------------------------
    # mouseX : the X coordinate the user clicked
    # mouseY : the Y coordinate the user clicked

    # determine the row the user clicked
    if (mouseY < 100):
        row = 0
    elif (mouseY < 200):
        row = 1
    else:
        row = 2

    # determine the column the user clicked
    if (mouseX < 100):
        col = 0
    elif (mouseX < 200):
        col = 1
    else:
        col = 2

    # return the tuple containg the row & column
    return (row, col)
```

2-85		10-91	
106-156		109-156	
91-93		93-95	
96-104		99-108	

```
        y_int = 0
    elif num == 5:
        x_int = 0
        y_int = 0
    elif num == 6:
        x_int = 3
        y_int = 0
    elif num == 7:
        x_int = -3
        y_int = -3
    elif num == 8:
        x_int = 0
        y_int = -3
    elif num == 9:
        x_int = 3
        y_int = -3

    print(x_int, y_int)

    if z == 1:
        stddraw.setPenColor(BLUE)


        stddraw.line(x_int+1, y_int+1, x_int-1, y_int-1)
        stddraw.line(x_int+1, y_int-1, x_int-1, y_int+1)
    elif z == 2:
        stddraw.setPenColor(RED)
        stddraw.setPenRadius(0.02)


        stddraw.circle(x_int, y_int, 1.2)
        stddraw.show(1000)

num_turn = 0

while not num_turn == 9:
    if stddraw.mousePressed():
        x = stddraw.mouseX()
        y = stddraw.mouseY()
        print(x,y)


        num_turn += 1
```

```
        x_int = -3
        y_int = 0
    elif num == 5:
        x_int = 0
        y_int = 0
    elif num == 6:
        x_int = 3
        y_int = 0
    elif num == 7:
        x_int = -3
        y_int = -3
    elif num == 8:
        x_int = 0
        y_int = -3
    elif num == 9:
        x_int = 3
        y_int = -3
    if z == 1:


        stddraw.line(x_int+1,y_int+1,x_int-1,y_int-1)
        stddraw.line(x_int+1,y_int-1,x_int-1,y_int+1)
        stddraw.show(1000)
    elif z == 2:
        stddraw.setPenRadius(0.03)
        stddraw.circle(x_int,y_int,1.2)


        stddraw.setPenRadius(0.11)
        stddraw.show(1000)

num_turn=0

while not num_turn == 9:
    #stddraw.show(0.0)
    if stddraw.mousePressed():
        x = stddraw.mouseX()
        y = stddraw.mouseY()


        num_turn += 1
        if num_turn%2 == 0:
            z=2
```

Last time...

- Minesweeper
  - Graphics
  - Game State
  - Game Logic
- Live implementation
  - Classes/Objects

Classes and Objects

Analogous to **data types** and **instances**.

In other words, we can define our own data types and create multiple unique instances of them.

Defining a Class

```
class Human():
    population = 0
    def __init__(self, name):
        self.name = name
        Human.population += 1

    def talk(self):
        print(“Hi, my name is” + self.name)
```

Defining a Class

```
class Human():
    population = 0
    def __init__(self, name):
        self.name = name
        Human.population += 1

    def talk(self):
        print(“Hi, my name is” + self.name)
```

class Human()

The class keyword designates all lines in this block to be part of the definition of the class.

You define **class/static** variables as well as functions directly under this.

Defining a Class

```
class Human():
    population = 0
    def __init__(self, name):
        self.name = name
        Human.population += 1

    def talk(self):
        print(“Hi, my name is” + self.name)
```

def __init__(self, name):

The __init__() function is special:

1. Run every time you make an object of this class. It **initializes** the object.
2. Its name is replaced with the name of the class. In this case, we would call Human(name) to make a new Human type object

You usually declare **instance** variables under here.

self

Used to call and declare instance variables. It provides a way for an object to find and reference itself, hence, **self**.

Therefore, **self** must be the first parameter to *any* class function.

Class/Static Variables vs Instance Variables

Class/Static variables have **one copy** that is shared among *all* objects of the same class. Every object of this class knows the value of the variable at any time.

Each object has **their own copies** of instance variables. An object does not know the value of another object's instance variable unless explicitly called.

Defining a Class

```
class Human():
    population = 0 # This is a static variable
    def __init__(self, name):
        self.name = name
        Human.population += 1 # Call using the class name

    def talk(self):
        print("Hi, my name is" + self.name)
```

Defining a Class

```
class Human():
    population = 0
    def __init__(self, name):
        self.name = name # Declaring an instance variable
        Human.population += 1

    def talk(self):
        print("Hi, my name is" + self.name)
```

Defining a Class

```
class Human():
    population = 0
    def __init__(self, name):
        self.name = name
        Human.population += 1

    def talk(self):
        print(“Hi, my name is” + self.name)
```