Assignment: A1 (back to main page)

Introduction to Pygame

This first assignment is intended to get you started in the development environment and introduce you to most of the basic concepts of Pygame. Here is an outline of topics to be covered:

1. Installation and demonstration of the development environment.
    a. Python, Pygame, Notepad++
    b. Python command prompt.
2. Notepad++ (the editing/running environment).
    a. Basic editing tips.
    b. Starting and debugging programs from Notepad++.
3. Pygame basics:
    a. Importing modules and initializing Pygame.
    b. Game window and surfaces.
    c. Drawing or blitting to a surface.
    d. Rectangles, circles, lines, and polygons.
    e. Flipping (updating) the display.
    f. The event queue.
4. Python:
    a. Statements.
    b. Looping.
    c. Conditional branching and Boolean logic.
    d. Using Pygame objects: methods and parameters.
    e. Tuples and lists.
    f. Printing to the command window.
    g. Breaking code to see what kind of error messages result.

Note: You may find these Python 2.7 tutorials handy references throughout the course.

http://docs.python.org/2/tutorial/index.html

http://learnpythonthehardway.org/book/

The following link is the documentation page for Pygame. Notice the links in the green box at the top of the page. This is a quick way to check out the parameters that are needed when calling Pygame methods.

http://www.pygame.org/docs/index.html

The following tutorial from Simpson College on arcade games is excellent. Note that it uses the 3.x version of Python.

http://programarcadegames.com/

The following link lists the key differences between Python 2 and Python 3. For a beginner or intermediate programmer these differences are quite small. The main reason I use Python 2.7 is due to dependencies on supporting modules that are not available in Python 3.

http://inventwithpython.com/appendixa.html

Problem statement:

Write a Python program that draws a ball (circle) in the Pygame window at the position of the mouse cursor. Have features to control:

- The erasing of the screen (the “e” key);
- The updating of the display (the “f” key);
- The color of the ball (the left and right mouse buttons).

Algorithmic description:

- Initialize objects and variables.
- Initially fill the Pygame window with the color white.
- Until the user decides to quit, repeatedly execute the following statements:
    - Check for user input: characterize the keyboard and mouse-button events in the event queue. Then also get the cursor position.
    - If the “e” key is down, fill the Pygame window with grey color (this erases everything).
    - Choose a color for the ball based on which mouse-button is depressed:
        - If left button is down, use yellow.
        - If right button is down, use red.
        - Otherwise, if no button is depressed, use blue.
    - Draw a ball (circle) at the cursor position.
    - Update the total time by adding the differential time.
    - Print to the command window:
        - Total time
        - Differential time between frames
        - Frame rate
    - If the “f” key is not down, update the Pygame window (i.e., make the rendering actions above visible in the window).

Some advice:

Spend some time associating (reading and jumping back and forth between) the problem, the algorithm, and the Python code. Then start typing up the code from the provided image below in to a Python text file (“.py” extension). Run it and play with the features of the code. Try breaking the code and seeing what you get in terms of error messages. Try adapting the code to do something a little different.

Python code: (see images on next two pages)

This first week an example code solution is provided (as an image) in the assignment statement. Normally, code solutions will be provided as text files a day or two after the assignment is given.

```
import sys, os
import pygame

# PyGame Constants
from pygame.locals import *
from pygame.color import THECOLORS

# Initialize the pygame environment.
pygame.init()

# Create a display surface to write to.
display_surface = pygame.display.set_mode((600,400))

# Instantiate a clock to help control the framerate.
myclock = pygame.time.Clock()

# Set the intial color of the whole pygame window.
display_surface.fill(THECOLORS["white"])

# Initialize some variables.
framerate_limit = 100
time_s = 0.0
key_e = "U"
key_f = "U"
user_done = False
mouse_button_UD = "U"

while not user_done:

    dt_s = float(myclock.tick( framerate_limit) * 1e-3)
    #myclock.tick(framerate_limit)

    #=====================================================
    # Get user input
    #=====================================================

    # loop through the list of events in the event queue.
    for event in pygame.event.get():

        # This main "if" structure checks the event type of each event.
        # Depending on the event type (QUIT, KEYDOWN, KEYUP, MOUSEBUTTONDOWN, or
        # MOUSEBUTTONUP), addition checks are made to identify the characteristics of the
        # event.
        if (event.type == pygame.QUIT):
            user_done = True

        elif (event.type == pygame.KEYDOWN):
            if (event.key == K_ESCAPE):
                user_done = True
            elif (event.key==K_e):
                key_e = 'D'
            elif (event.key==K_f):
                key_f = 'D'
```

```
55
56          elif (event.type == pygame.KEYUP):
57              if (event.key==K_e):
58                  key_e = 'U'
59              elif (event.key==K_f):
60                  key_f = 'U'
61
62          elif (event.type == pygame.MOUSEBUTTONDOWN):
63              mouse_button_UD = 'D'
64
65              # The get_pressed method returns T/F values in a tuple.
66              (button1, button2, button3) = pygame.mouse.get_pressed()
67
68              if button1:
69                  mouse_button = 1
70              elif button2:
71                  mouse_button = 2
72              elif button3:
73                  mouse_button = 3
74              else:
75                  mouse_button = 0
76
77          elif (event.type == pygame.MOUSEBUTTONUP):
78              mouse_button_UD = 'U'
79
80      # Get the cursor position: x,y. Return this as a tuple.
81      mouse_xy = pygame.mouse.get_pos()
82
83      #=====================================================
84      # End of user input collection
85      #=====================================================
86
87      # Erase the screen if the "d" key is pressed. Do this by filling the entire
88      # screen with grey color.
89      if (key_e == 'D'):
90          display_surface.fill(THECOLORS["grey"])
91
92      # Determine the color for the circle based on if a mouse button is up (U) or down (D).
93      if ((mouse_button_UD == 'D') and (mouse_button == 1)):
94          circle_color = THECOLORS["yellow"]
95      elif ((mouse_button_UD == 'D') and (mouse_button == 3)):
96          circle_color = THECOLORS["red"]
97      else:
98          circle_color = THECOLORS["blue"]
99
100     # Draw the circle
101     pygame.draw.circle(display_surface, circle_color, mouse_xy, 10, 0)
102
103     # Add the incremental time to our time variable.
104     time_s += dt_s
105
106     # Print to the command window.
107     print time_s, dt_s, myclock.get_fps()
108
109     # If the "f" key is up (not Down), update the entire display window.
110     if (key_f != 'D'):
111         pygame.display.flip()
```