Python: Input Events

Computer Science ICS20

Event Types

Handling input is more complicated than using the simple Python input() function, but it is much more versatile. The standard Python input() function has at least 2 limitations:

- the input() function halts program execution and waits for input, making it unusable for many types of programs such as games, that require constant screen updates.
- the input() function requires the user to type the ENTER key after specifying their input, again making it difficult to make certain programs that need quick single key inputs.

For these reasons pygame has an event queue (like a list) to store all events as they happen. An event can be any action that the user initiates, such as typing a key or moving a mouse. Events are generated while the actual program is running. You can think of the pygame event system as a program that runs in parallel with your program and informs you of various inputs that occur from different sources.

All pygame Events have a type identifier to tell you what type of input has occurred. Here is a list of Event types that we will be using:

Event Type	**Purpose**	***Event Attributes***	
QUIT	User has clicked the close button	*none*	
ACTIVEEVENT	Pygame has been activated or hidden	gain, state	
KEYDOWN	Key has been pressed	unicode, key, mod	for interacting with the keyboard
KEYUP	Key has been released	key, mod	
MOUSEMOTION	Mouse has been moved	pos, rel, buttons	for interacting with the mouse
MOUSEBUTTONDOWN	Mouse button was pressed	pos, button	
MOUSEBUTTONUP	Mouse button was released	pos, button	
JOYAXISMOTION	Joystick or pad was moved	joy, axis, value	for interacting with joysticks, gamepads and trackballs
JOYBALLMOTION	Joy ball was moved	joy, ball, rel	
JOYHATMOTION	Joystick hat was moved	joy, hat, value	
JOYBUTTONDOWN	Joystick or pad button was pressed	joy, button	
JOYBUTTONUP	Joystick or pad button was released	joy, button	
VIDEORESIZE	Pygame window was resized	size, w, h	
VIDEOEXPOSE	Part or all of the Pygame window was exposed	*none*	

Notice that each event type could have attributes associated with it. For example, the KEYDOWN event notifies you that a key has been typed and the “key” attribute will allow you to tell which key was pressed.

Handling Events

We already have template code that uses the QUIT event to notify us to close the window and end the program. Let’s now look at this part of the code more closely:

```
import pygame
pygame.init()     # initializes the graphics module
window = pygame.display.set_mode((800,600))     # define window size
pygame.display.set_caption('PyGame Events')     # title of program that
                                                # appears on window frame
clock = pygame.time.Clock()     # used to track time within the game (FPS)
quit = False
while not quit:                             # main program loop
    for event in pygame.event.get():        # check if there were any events
        if event.type == pygame.QUIT:       # check if user clicked the upper
            quit = True                     # right quit button

      # your code that draws to the window goes here


    pygame.display.update()         # refresh your display
    clock.tick(60)                  # wait a certain amount of time that
                                    # ensures a frame rate of 60 fps
pygame.quit()                       # shutdown module
```

Notice that for every frame we process all of the events that may have occurred since the last time we checked. The part of the code that does this is the following for loop:

```
for event in pygame.event.get():          # check if there were any events
```

The for loop retrieves and iterates over all the events that have occurred. It is important to retrieve all of the events using the `pygame.event.get()` function during each frame so that the events don't accumulate in the event queue. (If too many events accumulate, pygame will assume that your program has become non-responsive)

Keyboard Events

For each event, we check what type of event it may be. We don't have to check for all events, only the ones we are interested in. (If other events happen, they will simply be discarded) In our template example, we were only interested in a single event, the QUIT event. Now we will extend it to check for keyboard input by placing another if statement to check for keyboard events:

```
for event in pygame.event.get():                # check if there were any events
        if event.type == pygame.QUIT:           # check if user clicked the upper
            quit = True                         # right quit button
        elif event.type == pygame.KEYDOWN:      # check if user typed a key
            if event.key == pygame.K_a          # "a" key was pressed
                print("The A key was pressed.")
```

After we determine that a KEYBOARD event occurred, we use another if statement to check which key was actually pressed by using the **key** attribute of the KEYBOARD event. We compare this to whatever key we are interested in, in the case the "a" key. The "a" key has a pygame variable associated with it called K_a. In fact all keyboard keys have a similar name associated with them. You can see the whole list of them here.

Mouse Events

Mouse events are handled in the same way as all other events. Here is the event handling loop with mouse movement events:

```
for event in pygame.event.get():                # check if there were any events
        if event.type == pygame.QUIT:           # check if user clicked the upper
            quit = True                         # right quit button
        elif event.type == pygame.KEYDOWN:      # check if user typed a key
            if event.key == pygame.K_a          # "a" key was pressed
                print("The A key was pressed.")
        elif event.type == pygame.MOUSEMOTION: # check if event was a mouse motion
            print("Mouse (x, y) = ", event.pos)
```

The above handler for mouse movement will print the (x, y) coordinates of the mouse as it moves around the screen. But you can also check which mouse button was pressed by looking at the **button** attribute of the MOUSEBUTTONDOWN event:

> **button attribute** - It will return the number of the button that was pressed. A value of 1 indicates that the left mouse button was pressed, 2 indicates that the middle mouse button was pressed and 3 indicates that the right button was pressed.

The following code will print the coordinates of the mouse only if the left mouse button is clicked somewhere on the screen:

```
for event in pygame.event.get():                # check if there were any events
        if event.type == pygame.QUIT:           # check if user clicked the upper
            quit = True                         # right quit button
        elif event.type == pygame.KEYDOWN:      # check if user typed a key
            if event.key == pygame.K_a          # "a" key was pressed
                print("The A key was pressed.")
        elif event.type == pygame.MOUSEBUTTONDOWN:    # mouse button pressed
            if event.button == 1:                     # left mouse button pressed
                print("Mouse (x, y) = ", event.pos)
```

Exercises

1. Write a Python program that draws a blue rectangle somewhere on the screen. Then add some event handler code that moves the rectangle left by one pixel every time the a key is pressed or right by one pixel if the d key is pressed.

2. Write a Python program just like in question 1 but this time, instead of using the a and d keys, use the left arrow and right arrow keys. You can find the name of these keys (as well as all others) here.

   Then make the rectangle move up and down as well using the up arrow and down arrow keys.

3. Write a Python program that draws a red circle of radius 20 pixels wherever the current mouse pointer is.

4. Write a Python program that draws a line between any two points on the screen that are chosen by clicking on the left mouse button. ie. The left mouse is clicked twice in two different locations on the screen and a line is then drawn between those two points.

5. Use Python to write a simple painting program as follows:

   a) Create a palette of the following 8 colours in a list: [red, green, blue, yellow, cyan, magenta, white, black]

   b) On the left side of your window draw 8 squares vertically each of size 20x20 pixels. Each square will be coloured from the colours in the palette list, with red being on the top.

   c) The current colour can be selected by left clicking on one of the 8 coloured squares.

   d) If the left mouse button is pressed anywhere else on the screen, colour the pixel at the mouse location in the current colour. If the mouse is moving while the left mouse button is pressed, then the pixels under the mouse pointer should continually be coloured with the current colour.

   e) If the user right clicks on any of the 8 squares on the left side of the window, the screen is erased with that colour.