Teach Pygame 2: Keyboard and Mouse

Event Handling

The nice thing about pygame is that you don't have to do any real event handling in order to respond to mouse and keyboard events. The pygame system captures any events that happen throughout the life of the game and stores them in a queue. It also takes certain actions in response to these events.

Most relevant for us, it keeps track of which keys and buttons are currently being held down and also keeps track of where the mouse was last positioned. This means that for most purposes, all you have to do to create a game with keyboard or mouse control is to build a game loop which handles the animations, repeatedly asks the pygame system where the mouse is and which keys and buttons are currently in the down position and responds appropriately.

A Simple Game Loop

A simple game loop can be built as an extension of the animation loop from the previous handout:

1. Initialize game variables (object positions, speeds, etc.)
2. Check for and respond to the QUIT event using the code from the last handout.
3. Check keyboard and mouse and respond appropriately (e.g. if down key is currently pressed, change the y location of the player object)
4. Update position variables for objects that are moving automatically.
5. Clear and redraw the screen and all game objects.
6. Pause for 1/60 of a second or so (e.g. `time.sleep(1/60)`).
7. Go back to step 2.

An example of a game loop like this in action can be found in the file *keyboard_and_mouse_control.py*. There are no automatic animations in this, but the up and down keys, mouse movements, and mouse button presses all cause changes to the state of the game objects.

A Limitation

One big issue with the game loop pattern shown above is its timing. You are pausing for 1/60 of a second, but the actual frame rate will be lower than that and will depend on the speed of the computer on which the game is running. The fix for this is to keep track of the actual time elapsed at the top of each loop and then do some calculations to decide how many frames you need to process each time through the loop.

Checking the Quit Event

Use the same code here as shown on the last handout.

```
if pygame.event.peek(pygame.QUIT):
    pygame.display.quit()
    quit()
```

Note that it is very important that the check for the QUIT event at the top of your game loop. Due to the idiosyncrasies of pygame's implementation, if you place it after you check the keyboard and mouse state, it will not work.

Mouse Position

Get the current mouse location like this:

```
mx, my = pygame.mouse.get_pos()
```

Mouse Buttons

Check if each of the three buttons are currently depressed like this:

```
b1, b2, b3 = pygame.mouse.get_pressed()
```

If Button 1 is held down, `b1` will be True, etc.

Keys

You can retrieve a data structure containing the current state of every key like this:

```
keys = pygame.key.get_pressed()
```

Then you can index into the keys data structure using predefined pygame constants. For example, to check if the left arrow key is currently depressed, you check whether `keys[pygame.LEFT]` is `True` or `False`.

A full list of all the pygame key constants is here: http://www.pygame.org/docs/ref/key.html

More Limitations

If your game loop is operating quickly the above may be all you need to generate a cool game. But there is always a possibility that you will miss an event (for example, the user presses and releases a key so fast that it happens between calls to `pygame.key.get_pressed`).

Another limitation is that every key and button press generates two events – one when the button is pressed down and another when it is released. The above methods don't let you do that.

Finally, there are other events (such as scroll wheel events) that cannot be handled in this way.

If any of these situations apply to you, you will have to retrieve the entire list of recent events from pygame and sort through them looking for and responding to particular events (Key Down, Scroll Up, etc.).

For more information on how to do this, see the pygame documentation. In particular, you should read:

http://www.pygame.org/docs/ref/event.html
http://www.pygame.org/docs/ref/key.html
http://www.pygame.org/docs/ref/mouse.html