Handling Events

A Pygame program would not be very exciting if the user could not interact with it. To respond to **events** such as mouse movements, clicks and key presses, we write code in the **event loop** to handle individual cases. When an event occurs, it is added to the **event queue** where it waits to be processed. We have already processed one event earlier: the `QUIT` event.

Each event in the queue has a `type` attribute that can be compared to a constant (see *Event Type Constants*). The code below prints a message to the terminal when the mouse is moved.

```
if event.type == pygame.MOUSEMOTION:
      print(“User moved the mouse.”)
```

When the mouse is moved, a `MOUSEMOTION` event is added to the queue. Its position is stored in the `pos` attribute. There are also attributes for the buttons that are pressed (a list, `buttons`) and the relative distance the mouse has moved (`rel`). When a button is pressed, a `MOUSEBUTTONDOWN` event is added to the queue. The button pressed is recorded as an integer, in `button` (see below).

If the event type is `KEYDOWN`, we can check the value of the key that has been pressed using the `key` attribute. The code below prints a message to the terminal when the user presses the ‘a’ key.

```
if event.type == pygame.KEYDOWN:
      if event.key == pygame.K_a:
            print(“User pressed ‘a’.”)
```

See *Keyboard Key Constants* for a full list of keys. Note that there are no capital letters. To check for these, check the `mod` (modifier) attribute for `LSHIFT` or `RSHIFT`.

Event Type Constants

Event	Description
`QUIT`	Stop the program.
`MOUSEMOTION`	Mouse movement.
`MOUSEBUTTONDOWN`	Mouse button down (1=Left, 2=Middle, 3=Right, 4=Scroll up, 5=Scroll down).
`MOUSEBUTTONUP`	Mouse button up (see above).
`KEYDOWN`	Keyboard button down (see *Keyboard Key Constants*).
`KEYUP`	Keyboard button up.
`VIDEORESIZE`	Change the window size.
`VIDEOEXPOSE`	Expose part or all of the window.
`USEREVENT`	User-created event.

There are also events for joystick movement and button presses, detailed in the documentation.

Handling Events

Keyboard Key Constants

Letters:
```
K_a ... K_z
```

Numbers:
```
K_0  ... K_9
```

Punctuation:
```
K_SPACE
K_PERIOD
K_COMMA
K_QUESTION
K_AMPERSAND
K_ASTERISK
K_AT
K_CARET
K_BACKQUOTE
K_DOLLAR
K_EQUALS
K_EURO
K_EXCLAIM
K_SLASH, K_BACKSLASH
K_COLON, K_SEMICOLON
K_QUOTE, K_QUOTEDBL
K_MINUS, K_PLUS
K_GREATER, K_LESS
```

Function keys:
```
K_F1 ... K_F15
```

Arrows:
```
K_LEFT
K_UP
K_RIGHT
K_DOWN
```

Brackets:
```
K_RIGHTBRACKET,  K_LEFTBRACKET
K_RIGHTPAREN, K_LEFTPAREN
```

Modifier keys:
```
K_LALT,   K_RALT
K_LCTRL,  K_RCTRL
K_LSUPER, K_RSUPER
K_LSHIFT, K_RSHIFT
K_RMETA,  K_LMETA
```

Control:
```
K_TAB
K_RETURN
K_ESCAPE
K_SCROLLOCK
K_SYSREQ
K_BREAK
K_DELETE
K_BACKSPACE
K_CAPSLOCK
K_CLEAR
K_NUMLOCK
```

Keypad:
```
K_KP0 ... K_KP9
K_KP_DIVIDE
K_KP_ENTER
K_KP_EQUALS
K_KP_MINUS
K_KP_MULTIPLY
K_KP_PERIOD
K_KP_PLUS
```

Edit keys:
```
K_HELP
K_HOME
K_END
K_INSERT
K_PRINT
K_PAGEUP, K_PAGEDOWN
K_FIRST,  K_LAST
```

Other:
```
K_MENU
K_MODE
K_PAUSE
K_POWER
K_UNDERSCORE
K_HASH
K_UNKNOWN
```

Handling Events

Answer the following questions.

1. Why is it useful to have an event queue, rather than processing each event the instant it occurs?

2. Think of a scenario where it is useful to have separate `KEYDOWN` and `KEYUP` events, rather than just a "`KEYPRESS`" event.

Write programs that accomplish each task, using appropriate programming conventions.

3. Beginning with a blank black screen, allow the user to change its colour by pressing one of the "rainbow colour" keys, ROYGBIV. For example, if the user presses R, change the screen to red.

4. Draw a rectangle in the centre of the screen. Have the user move the rectangle around the window using either the arrow keys or the classic WASD keys (W=up, A=left, S=down, D=right). Do not allow the rectangle to move beyond the edges of the window.

5. Create a blank window. When the user clicks a mouse button, display the coordinates of the click in the top right corner of the screen.

6. Display your name, centred on the screen. While the user holds down the R key, move the text to the right. While the user holds down the L key, move the text to the left. The text should stop if it hits the edge of the screen.

7. Modify your program so that the text moves left when the left mouse button is held down, and moves right when the right mouse button is held down.