Interaction and Events

Interaction

Exercise 1
For the following interactive Processing program, who or what is:

The user:

The system:

A user action:

A system feedback:

```
def setup():
    size(300,300)
    background(255,255,255)

def draw():
    ellipse(mouseX,mouseY,100,100)
```

Answer:

User	Instructor/Class
System	Our Processing program
User Action(s)	Moving the mouse about on the canvas
System Feedback	Drawing circles on the canvas

Demo 1
Below is a simple run-to-completion program.
How could we write the exact same program in interactive mode (i.e. using setup() and draw())?

```
background(0,0,0)
fill(255,255,255)
rect(20,25,20,50)
fill(255,255,255)
rect(60,25,20,50)
```

Solution

```
# CMPT 140 - Interaction and Events
# Topic: Interaction in Processing
# DEMO

def setup():
    background(0,0,0)
    fill(255,255,255)
    rect(20,25,20,50)
    fill(255,255,255)
    rect(60,25,20,50)

def draw():
    return
```

This code can be found in: `cmpt140-ch06-py/cmpt140_ch06_interaction_demo/cmpt140_ch06_interaction_demo.pyde`

It's important to note that if we placed the code inside `draw()` instead of `setup()` the visual output may appear the same, but what is actually being drawn to the screen is different! We placed it inside `setup()` because the background will only be drawn once here, which is what we want in this case since that is similar to the draw once notion of the run-to-completion code. Refer to your readings for more information on the dfiferences between the two functions.

Exercise 2
Write Processing code which draws a SINGLE circle that follows the mouse.

The circle **color** should be blue

The circle **size** should be 50x50

The **background** should be black

Solution

```
# CMPT 140 - Interaction and Events
# Topic: Interaction in Processing


def setup():
    return

def draw():
    background(0,0,0)
    fill(0,0,255)
    ellipse(mouseX,mouseY,50,50)
```

This code can be found in: `cmpt140-ch06-py/cmpt140_ch06_interaction/cmpt140_ch06_interaction.pyde`

By redrawing the background as the first action on every call to `draw()`, we are effectively clearing the canvas to start anew with a blank canvas. This blank canvas ensures that only one circle, the new one to be drawn, will be drawn to the canvas on any call to `draw()`.

Event Handling

Exercise 3

What does the following Processing code do?

```
def setup():
    size(300,300)

def draw():
    return

def mouseClicked():
    fill(255,255,0)
    ellipse(mouseX,mouseY,50,50)

def keyPressed():
    fill(0,0,0)
    ellipse(mouseX,mouseY,50,50)
```

Answer: On a mouse click, a yellow 50*50 pixel circle is drawn to the Processing canvas at the current mouse coordinates. On pressing any key on the keyboard, a black 50*50 pixel circle is drawn to the Processing canvas at the current mouse coordinates. `setup()` and `draw()` don't require any special instructions but still need to be defined for our program to run in interactive mode, so their function bodies are filled with the placeholder value `return`.

Exercise 4

Create a Processing program with a 255x255 canvas.

When the user clicks the mouse: A 50x50 circle appears at the mouse location. The circle's RGB colour depends on its **location**: green equal to its x-coodinate, blue equal to its y-coordinate and no red at all.

When the user presses a key: The canvas should be cleared, and the background should be set to a shade of gray equal to the mouse's current x-coordinate.

Solution

```
# CMPT 140 - Interaction and Events
# Topic(s): Event Handling


def setup():
    size(255,255)

def draw():
    return

def mouseClicked():
    fill(0,mouseX,mouseY)
    ellipse(mouseX,mouseY,50,50)

def keyPressed():
    background(mouseX)
```

This code can be found in: `cmpt140-ch06-py/cmpt140_ch06_events/cmpt140_ch06_events.pyde`

Exercise 5

For the following exercise, you only have access to these statements:

```
ellipse(mouseX,mouseY,50,50)   # draw ellipse at mouse coords
background(0)                  # colour background black
return                         # function is done
```

You can add any number of them anywhere to this template:

```
def setup():
    size(300, 300)
    # code here is called once at program start

def draw():
    # code here is called continuously until program end

def mouseClicked():
    # code here is called on a mouse click
```

How can the following effects be achieved?

(a) A stream of circles follows the mouse. All the circles disappear when the mouse is clicked.

Solution

```
# CMPT 140 - Interaction and Events
# Topic(s): Drawing Effects
# Part (a)


def setup():
    size(300, 300)
    background(0)

def draw():
    ellipse(mouseX,mouseY,50,50)

def mouseClicked():
    background(0)
```

This code can be found in:
`cmpt140-ch06-py/cmpt140_ch06_drawing_effects_a/cmpt140_ch06_drawing_effects_a.pyde`

(b) A single circle follows the mouse. Nothing happens when the mouse is clicked.

Solution

```
# CMPT 140 - Interaction and Events
# Topic(s): Drawing Effects
# Part (b)


def setup():
    size(300, 300)

def draw():
    background(0)
    ellipse(mouseX, mouseY, 50, 50)

def mouseClicked():
    return
```

This code can be found in:
`cmpt140-ch06-py/cmpt140_ch06_drawing_effects_b/cmpt140_ch06_drawing_effects_b.pyde`

(c) A circle appears wherever the mouse is clicked. If the mouse is clicked somewhere else, the circle **moves** to that spot.

Solution

```
# CMPT 140 - Interaction and Events
# Topic(s): Drawing Effects
# Part (c)

def setup():
    size(300, 300)
    background(0)

def draw():
    return

def mouseClicked():
    background(0)
    ellipse(mouseX, mouseY, 50, 50)
```

This code can be found in:
`cmpt140-ch06-py/cmpt140_ch06_drawing_effects_c/cmpt140_ch06_drawing_effects_c.pyde`