Graphics in Python

Python has a library called PyGame to help with writing games, which is mainly about doing things with graphics.

# Getting Set Up

A basic PyGame program usually has the same structure regardless of the type of game being written:

1. Import PyGame.
2. Initialise PyGame.
3. Create a surface to represent the window in which the game is played.
4. Define a main loop that runs repeatedly that performs the following actions:
   1. Handle user input.
   2. Move graphics.
   3. Draw graphics.
   4. Check collisions and make decisions based on collisions.
5. Terminate.

Here’s the skeleton of a Python game:

**from** pygame.locals **import** \*  
**import** pygame  
**import** pygame.key  
  
*# Initialise PyGame*pygame.init()  
  
*# Create a window that's 640 pixels wide and 480 pixels high*s\_width = 640  
s\_height = 480  
screen = pygame.display.set\_mode((s\_width, s\_height))  
  
*# Create a clock to manage the frames per second of the game*clock = pygame.time.Clock()  
  
*# Used to terminate the main loop*user\_quit = **False***# Handle input from the user (mouse and keyboard)***def** process\_input():  
 **global** user\_quit  
 **for** event **in** pygame.event.get():  
 **if** event.type == QUIT:  
 *# Exit the game immediately if the close window icon is pressed.* user\_quit = **True  
  
 if** event.type == pygame.KEYDOWN:  
 key\_name = pygame.key.name(event.key)  
 print(key\_name)  
 **if** key\_name == **"escape"**:  
 user\_quit = **True***# Move all the graphics***def** move\_game():  
 **pass***# Draw all the graphics***def** draw\_game():  
 screen.fill((255, 255, 255))  
 *# Draw all graphics between HERE  
  
 # and HERE* pygame.display.update()

*# The main loop of the game***def** main\_loop():  
 **while not** user\_quit:  
 clock.tick(60)  
 process\_input()  
 move\_game()  
 draw\_game()  
  
  
*# Start the game*main\_loop()  
  
*# Terminate PyGame gracefully*pygame.quit()

If you run this game, all the will happen is that you get a black screen that you can terminate by clicking the “X” in the window or pressing the escape key.

## Sprites

A “sprite” is the name given to an object in a game that has a graphical representation. Sprites have the following properties:

1. An image – that is shown on the screen.
2. An x and y position – where the sprite is shown on the screen.
3. A width and height – how big the sprite is.
4. A vector - if the sprite is moving then we need to know amount and direction the sprite should move each frame.

Let’s make a football bounce around the screen. Declare the variables before the definition of process\_input.

*# Create the football variables, all begin with "fb"*fb\_img = pygame.image.load(**'football.jpg'**)  
  
*# The position of the ball*fb\_x = 100  
fb\_y = 100  
  
*# The size of the ball in pixels*fb\_width = fb\_img.get\_width()  
fb\_height = fb\_img.get\_height()  
  
*# Vector the ball is moving at (horizontal and vertical)*fb\_dx = 5  
fb\_dy = 5

Add a new function to draw the ball using the screen.blit function.

**def** draw\_ball():  
 screen.blit(fb\_img, (fb\_x, fb\_y))

Add a call to draw\_ball inside draw\_game. Run your code and you should see your image on the screen.

To make it move, add a new function move\_ball just before move\_game and call it from inside move\_game (remove the line that says pass, that’s just to tell Python there’s nothing in the function yet), so it looks like this:

**def** move\_ball():

*# Move all the graphics***def** move\_game():  
 move\_ball()

Inside move\_ball we need to tell Python that we’re going to be changing the position of the ball and its direction (if the ball bounces off the sides of the screen). To do this you use the global keyboard.

**global** fb\_x, fb\_y, fb\_dx, fb\_dy

Each frame the ball will change its position by fb\_dx and fb\_dy.

fb\_x = fb\_x + fb\_dx  
 fb\_y = fb\_y + fb\_dy

If you run this, you can see the ball move, but it sails straight off the screen! To stop this we need to do collision detection.

To do this we’ll reverse the direction of the ball if it hits the sides of the screen and reset the

*# Bounce off the left-hand side of the screen* **if** fb\_x < 0:  
 fb\_dx = -fb\_dx  
  
 *# Bounce off the right-hand side of the screen* **if** fb\_x + fb\_width > s\_width:  
 fb\_dx = -fb\_dx  
  
 *# Bounce off the top of the screen* **if** fb\_y < 0:  
 fb\_dy = -fb\_dy  
  
 *# Bounce off the bottom of the screen* **if** fb\_y + fb\_height > s\_height:  
 fb\_dy = -fb\_dy

Run your code and you now have a ball bouncing around the screen.

## Controlling a Sprite with the Keyboard

Let’s create a new sprite, Charmander, by creating variables underneath the football variables.

*# Create the Charmander variables*ch\_img = pygame.image.load(**'Charmander.png'**)  
ch\_x = 400  
ch\_y = 350  
ch\_width = ch\_img.get\_width()  
ch\_height = ch\_img.get\_height()  
ch\_dx = 0  
ch\_dy = 0

Even though Charmander will be controlled by the keyboard, we need to store a vector for him.

Add a function to draw Charmander as well (and call it from inside draw\_game)

**def** draw\_charmander():  
 screen.blit(ch\_img, (ch\_x, ch\_y))

Add another function to move Charmander:

**def** move\_charmander():  
 **global** ch\_x, ch\_y  
 ch\_x = ch\_x + ch\_dx  
 ch\_y = ch\_y + ch\_dy

Finally, let’s control it using the keyboard. When a key is pressed, PyGame sends you code a KEYDOWN event and when its released a KEYUP event is sent.

We need to make changes to process\_input to manage this. We’re going to be changing the vector (ch\_dx and ch\_dy), so add a global declaration at the top of process\_input.

**global** ch\_dx, ch\_dy

Finally, update process\_input so it looks like the listing below. The code in **bold** should already exist, so add your code beneath it.

**if event.type == pygame.KEYDOWN:  
 key\_name = pygame.key.name(event.key)  
 print(key\_name)  
 if key\_name == "escape":  
 user\_quit = True  
  
 elif** key\_name == **"left"**:  
 ch\_dx -= 2  
 **elif** key\_name == **"right"**:  
 ch\_dx += 2  
 **elif** key\_name == **"up"**:  
 ch\_dy -= 2  
 **elif** key\_name == **"down"**:  
 ch\_dy += 2  
  
**if** event.type == pygame.KEYUP:  
 key\_name = pygame.key.name(event.key)  
 print(key\_name)  
 **if** key\_name == **"left"**:  
 ch\_dx += 2  
 **elif** key\_name == **"right"**:  
 ch\_dx -= 2  
 **elif** key\_name == **"up"**:  
 ch\_dy += 2  
 **elif** key\_name == **"down"**:  
 ch\_dy -= 2

If that all works, then you should be able to control Charmander as your football bounces around the screen.

