Pharaoh Tutorial 1:

Materials:

* Python 3.3.3
* Pygame
* LiveWires
* Background.png
* Platform.png
* Pharaoh1.bmp
* Pharaoh1\_Back.bmp
* Pharaoh2.bmp
* Pharaoh2\_Back.bmp

We will construct:

* Actor class
* Pharaoh class

We will modify:

* Game class

This is the first part of the Pharaoh tutorial. We will focus on the structure of the Pharaoh himself. Part 2 will focus on the interaction between Pharaoh and PlatformTiles.

As always, take a look at the images before starting.

This builds off of the Platform Tutorial.

Instructions:

The Actor Class:

Start by setting up the Actor class as follows:

class Actor(games.Sprite):

def \_\_init\_\_(self, x, y, image):

super(Actor, self).\_\_init\_\_(image = image, x = x, y = y)

self.jumping = False

self.gravity = 0

self.lives = 1

This class will be the basis for Pharaoh as well as other classes in future tutorials.

We are building Actor on top of games.Sprite, and adding the variables self.jumping, self.gravity, and self.lives .These are three variables that will be used in Pharaoh later.

Next, add the update method:

def update(self):

super(Actor, self).update();

if self.lives == 0:

self.deactivate()

This method runs each time the game generates the next frame. The if statement checks to see if the Actor has any remaining lives; if it does not, then it removes it from the game.

Next up is the Pharaoh class:

Start by setting up Pharaoh and building it on top of Actor:

class Pharaoh(Actor):

IMAGE1\_R = games.load\_image("images/pharaoh1.bmp") #Face Right

IMAGE1\_L = games.load\_image("images/pharaoh1\_Back.bmp") #Face Left

IMAGE2\_R = games.load\_image("images/pharaoh2.bmp") #Walk Right

IMAGE2\_L = games.load\_image("images/pharaoh2\_Back.bmp") #Walk Left

WALK\_SPEED = 20 #The rate at which the pharaoh appears to walk (in Frames)

def \_\_init\_\_(self, x, y, game):

super(Pharaoh, self).\_\_init\_\_(

image = Pharaoh.IMAGE1\_R,

x = x, y = y)

**(Anything appearing immediately after a ‘#’ is a comment)**

Here, we load one image each for the Pharaoh standing facing left and right, and stepping forward facing left and right. We will use these to make Pharaoh look like he is actually walking instead of just sliding. WALK\_SPEED is also used in making him walk. We initially set him to facing forward, standing.

Now, continue building the \_\_init\_\_ method:

self.game = game

self.slide1 = Pharaoh.IMAGE1\_R

self.slide2 = Pharaoh.IMAGE2\_R

self.activeImage = 1

self.walkTimer = 0

self.score = 0

self.lives = 2

self.scoreTxt = games.Text(value = "Coins: "+str(self.score),

size = 35, color = color.black,

top = 5, left = 10)

self.livesTxt = games.Text(value = "Lives: "+str(self.lives),

size = 35, color = color.black,

top = 5, right = games.screen.width - 10)

This finishes the \_\_init\_\_ method. self.game is the Game object from the last tutorial. We give Pharaoh a copy of this because in future tutorials Pharaoh will need access to Game’s information.

self.slide1 and self.slide2 will be used like a flip-book to make it look like Pharaoh is walking; slide1 will always be Pharaoh standing and slide2 will always be Pharaoh stepping. We can set them to be facing right or left depending on which way Pharaoh is walking.

activeImage will be used to determine which slide (slide1 or slide2) should be drawn to the screen. walkTimer will later be used to tell the program when to switch from slide1 to slide2 or vice-versa.

self.score is the player’s score in the game. self.lives is the same variable ‘lives’ defined in Actor; whatever you set it to will be the number of lives a player has when they play the game.

Lastly, self.scoreTxt and self.livesTxt make up the Heads Up Display (HUD), which tells the player how many lives they have remaining and what their score is.

Next, define the activate and deactivate methods:

def activate(self):

games.screen.add(self)

games.screen.add(self.scoreTxt)

games.screen.add(self.livesTxt)

def deactivate(self):

self.end\_game()

self.destroy()

Like before, these are used to add and remove the Pharaoh from the game. This time, though, the activate method also adds the HUD to the game, and the deactivate method calls end\_game to show the game over screen. Here is the end\_game method:

def end\_game(self):

end\_message = games.Message(value = "Game Over",

size = 90,

color = color.red,

x = games.screen.width/2,

y = games.screen.height/2,

lifetime = 2 \* games.screen.fps,

after\_death = games.screen.quit)

games.screen.add(end\_message)

This looks like a lot, but it’s actually just two commands. First, you construct a message that tells the player “Game Over”, then you show it to him. “lifetime” tells the message how long to stay on the screen, and “after\_death” tells the game what to do after the message disappears. In this case, we want to end the game.

Now we want to add two methods to refresh the HUD, one for score and one for lives:

def refreshScore(self):

self.scoreTxt.value = "Coins: "+str(self.score)

def refreshLives(self):

self.livesTxt.value = "Lives: "+str(self.lives)

self.livesTxt.right = games.screen.width – 10

All these do is refresh the scoreTxt and livesTxt displays—you want to call them whenever you change these values. refreshLives() also re-positions the livesTxt display; depending on how high a number of lives the player has, it might need to be re-positioned so the whole number can fit on the screen.

Now, modify the Game class to include a Pharaoh and activate it. Also, modify the Platform object in Game so that instead of being in the middle of the screen, it makes a floor across the bottom:

class Game(object):

BACKGROUND = games.load\_image("images/background.png",transparent = False)

def \_\_init\_\_(self):

games.screen.background = Game.BACKGROUND;

platform = Platform(**x = 42**,

**y = games.screen.height**,

direction = "HORIZONTAL",

hrd = 1,

**num = 10**)

**pharaoh = Pharaoh(x = games.screen.width/2,**

**y = games.screen.height/2,**

**game = self)**

platform.activate()

**pharaoh.activate()**

games.screen.mainloop()

At this point, if you run the program, you should get:



Now, we need to make him walk when you press the left or right arrow keys.

This will be done in the update method, but before we write that, we need to write three functions, faceLeft, faceRight, and refreshGraphic.

def faceLeft(self):

self.slide1 = Pharaoh.IMAGE1\_L

self.slide2 = Pharaoh.IMAGE2\_L

self.refreshGraphic()

def faceRight(self):

self.slide1 = Pharaoh.IMAGE1\_R

self.slide2 = Pharaoh.IMAGE2\_R

self.refreshGraphic()

def refreshGraphic(self):

if(self.activeImage == 1):

self.image = self.slide1

else:

self.image = self.slide2

faceLeft sets the slide1 and slide2 variables from earlier to the images of the Pharaoh facing left standing (slide1) and walking (slide2). faceRight does the same thing, but with the images of the Pharaoh facing right. refreshGraphic looks at the activeImage variable; if it equals 1, that means slide1 should be drawn to the screen, and if it equals 2, then slide2 should be drawn to the screen.

Last is the update method:

def update(self):

super(Pharaoh, self).update()

As in other methods, the first thing we do is use this ‘super’ call. Since we built Pharaoh on top of Actor, we want to run Actor’s update method before we add new code. That’s what this line is for.

Now we will write the code for Pharaoh to walk around the screen:

#Keep the pharaoh in the bounds of the screen

if(self.left < 0):

self.left = 0

if(self.right > games.screen.width):

self.right = games.screen.width

If the Pharaoh is walking around, we want to make sure he doesn’t leave the screen. This code will keep him from walking off the right or left edge of the game screen.

Next, get the input from the left arrow key:

if games.keyboard.is\_pressed(games.K\_LEFT):

#Face left, start moving left, and increment the walkTimer

self.faceLeft()

self.dx = -0.9

self.walkTimer = self.walkTimer +1

If the left arrow key is pressed, make Pharaoh face left using the faceLeft method from earlier.

Then, set dx to -0.9. This makes him move left at a rate of 0.9 pixels per frame. There are 50 frames per second, so his speed to the left is 45 pixels per second. This is also where walkTimer comes into play; remember, if you are holding down the left arrow key, this part of the code will run every frame. By incrementing the walkTimer, we can tell how long the player has held down the left key. We can use this to figure out when to ‘flip’ the image of the Pharaoh from ‘standing’ to ‘stepping’ in order to make it look like he’s walking.

Now, if the left key is NOT pressed, let’s handle input from the right arrow key:

elif games.keyboard.is\_pressed(games.K\_RIGHT):

#Face right, start moving right, and increment the walkTimer

self.faceRight()

self.dx = 0.9

self.walkTimer = self.walkTimer +1

This is very similar to the code for the left arrow key, but now we make the Pharaoh face right and give him a positive velocity in the X direction, making him move to the right of the screen.

Now we need to handle what happens if neither arrow key is being pressed:

else:

#Not walking anymore; set walkTimer to 0 and

#make the pharaoh 'stand' still by

#changing the active Image to slide 1

self.walkTimer = 0

self.activeImage = 1

self.image = self.slide1

self.dx = 0

If neither arrow key is pressed, then the Pharaoh stands still. We reset the walkTimer to zero since he is not walking, and we set reset the image of the pharaoh to him standing, facing the last direction he was facing. Last, we reset dx to zero; this will make him stand still.

The last thing we have to do is write the code to actually animate the Pharaoh walking:

#Animate the pharaoh's walking

if(self.walkTimer >= Pharaoh.WALK\_SPEED):

self.walkTimer = 0

if(self.activeImage == 1):

self.image = self.slide2

self.activeImage = 2

else:

self.image = self.slide1

self.activeImage = 1

For reference, WALK\_SPEED from the beginning of this tutorial is set to 20. This piece of code will run every time the game generates a new frame. If walkTimer equals WALK\_SPEED, then an arrow key has been held down for 20 frames, or just less than half a second. This is the time to change the Pharaoh’s image so he looks like he’s walking.

activeImage tells us the current image of the Pharaoh; if it is 1, then he is ‘standing’, and if it is 2, he is ‘stepping’. Whichever it is, we want to switch the image to the other one. Then we update the value of activeImage to accurately show which slide is being used.

Notice also that we reset the walkTimer to zero; this will start the count over again. When it reaches 20, the slides will ‘flip’ again, and so on. Like a flip book, it gives the appearance that the Pharaoh is walking.

Now, if you run the program you will be able to make the Pharaoh walk from left to right on the screen!

