Pharaoh Tutorial 2:

Materials:

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

We will build:

* Enemy class

We will modify:

* PlatformTile class
* Game class

Instructions:

We will start by constructing the Enemy class on top of the Actor class:

class Enemy(Actor):

IMAGE\_1 = games.load\_image("images/enemy.png")#Face Left

IMAGE\_2 = games.load\_image("images/enemy\_back.png")#Face Right

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

super(Enemy, self).\_\_init\_\_(image = Enemy.IMAGE\_1, x = x, y = y)

It starts very similarly to the Pharaoh class. We load two images (one of the enemy facing right and one facing left). I made the Enemy images a snake—since snakes slither, we don’t need to animate it ‘walking’ like we did with the Pharaoh. This will simplify the code. As before, we use the ‘super’ call in the \_\_init\_\_method to run Actor’s \_\_init\_\_ method before running our code.

We will finish building the init method as follows:

self.mvSpeed = 0

self.moving = False

We give Enemy two parameters, mvSpeed and moving. These will be used to determine how the Enemy will move/slither around the level. Also, remember, Enemy inherits ‘jumping’, ‘gravity’, and ‘lives’ from Actor.

Next, define the activate and deactivate methods. We will also want Enemy to have a spriteOnTop method, just like PlatformTile does.

def activate(self):

games.screen.add(self)

def deactivate(self):

self.destroy()

def spriteOnTop(self, sprite):

return sprite.bottom - self.top <= 7

The last part we need is the update method:

def update(self):

super(Enemy, self).update()

As always, we start with the ‘super’ call to run Actor’s update method first.

Next, set ‘dx’ to Enemy’s ‘mvSpeed’ variable:

self.dx = self.mvSpeed

Other objects might modify Enemy’s mvSpeed, so every frame we will need to set Enemy’s X velocity to mvSpeed to keep it up-to-date.

Now, we will use Enemy’s ‘moving’ variable:

if self.x - games.screen.width <= 50:

self.moving = True

if self.moving:

if self.mvSpeed == 0:

self.mvSpeed = -1

This says, if Enemy is less than or equal to 50 pixels to the right of the visible screen, set moving to True. It may be a little unclear why we are doing this right now, but when we implement side-scrolling into the game, it will make more sense. Then, if moving is True, set mvSpeed to -1 (moving left at 1 pixel per frame). This way the Enemy won’t just be standing still; the Pharaoh will actually have to dodge it.

This next part should look very similar from the Pharaoh tutorials:

if not self.overlapping\_sprites:

self.jumping = True

if self.jumping:

self.dy += self.gravity

Like the Pharaoh, if Enemy is in mid-air or walks off a platform, then it will no longer be colliding with a sprite and we will want to apply Gravity so it falls to the ground.

The next step is to handle what happens if the Enemy falls off the bottom of the screen:

if self.y >= games.screen.height:

self.deactivate()

This way, if the Enemy falls out of the bottom of the screen, it will be removed from the game.

The last thing we need to do in Enemy is handle collision between Enemy and Pharaoh:

for sprite in self.overlapping\_sprites:

if isinstance(sprite, Pharaoh):

if not self.spriteOnTop(sprite):

sprite.lives -= 1

sprite.refreshLives()

We loop through the overlapping sprites looking to see if any of them are Pharaoh. If they are, we want to see if Pharaoh is ‘standing’ on top of the Enemy. If not, that means the Enemy is hitting Pharaoh from the side (or from above), so we want Pharaoh to lose a life. Then we need to refresh the HUD to show that Pharaoh has one less life.

Last, we need to handle what happens if the Pharaoh is ‘standing’ on top of the Enemy:

else:

self.lives -= 1

sprite.dy = -1

If this is happening, it means that Pharaoh has stomped on the Enemy and should kill it. We reduce the Enemy’s lives—if you remember from when we built the Actor class, lives is initially set to 1. Also, Actor’s update method checks to see if it’s lives have reached zero, and if so, deactivates the Actor. This all applies to Enemy since it is built on top of Actor.

This finishes the Enemy class—but we still need to handle the collision between Enemy and PlatformTile. Here is the code that we already have in PlatformTile to handle collisions with Pharaoh:

def update(self):

super(PlatformTile, self).update()

for sprite in self.overlapping\_sprites:

if isinstance(sprite, Actor):

if self.spriteOnTop(sprite) and sprite.dy >= 0:

sprite.bottom=self.top

sprite.jumping = False

elif self.isHard == True:

if not self.spriteOnTop(sprite):

if self.spriteOnLeft(sprite):

sprite.right = self.left

elif self.spriteOnRight(sprite):

sprite.left = self.right

elif self.spriteOnBottom(sprite):

sprite.top = self.bottom

sprite.dy = -sprite.dy

Remember—we already have this code written. Notice that when we loop through the overlapping sprites, we are looking for sprites that are Actors. Pharaoh and Enemy are both Actors, so this code will apply to both of them. We only need to add one small piece of code at the end to handle Enemies slightly differently than Pharaoh:

…

elif self.isHard == True:

if not self.spriteOnTop(sprite):

if self.spriteOnLeft(sprite):

sprite.right = self.left

elif self.spriteOnRight(sprite):

sprite.left = self.right

elif self.spriteOnBottom(sprite):

sprite.top = self.bottom

sprite.dy = -sprite.dy

**if isinstance(sprite, Enemy):**

**sprite.mvSpeed = sprite.mvSpeed \* -1**

**if self.left == sprite.right:**

**sprite.x -= 1**

**sprite.image = Enemy.IMAGE\_1**

**if self.right == sprite.left:**

**sprite.x += 1**

**sprite.image = Enemy.IMAGE\_2**

Notice that we added this to the part of the code that handles collision with ‘hard’ PlatformTiles—so this will run if the Enemy hits a wall. This code checks to see if the sprite is an Enemy, and if it is, it reverses the mvSpeed of the Enemy so that it will start moving in the opposite direction. Then, we want to see if the Enemy is on the right or left side of the PlatformTile. If it’s on the left, we set the image of sprite so it is facing right, and move it slightly to the right so it won’t be colliding with PlatformTile again in the next frame. We do the opposite if the sprite is to the right of the PlatformTile.

This finishes up the code to handle the Enemy class; now let’s test it. For a simple test, modify Game’s \_\_init\_\_ method to look like this:

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

games.screen.background = Game.BACKGROUND

platform = Platform(x = 42,

y = games.screen.height,

direction = "HORIZONTAL",

hrd = True,

num = 10)

**plat2 = Platform(x = 42,**

**y = games.screen.height - 60,**

**direction = "VERTICAL",**

**hrd = True,**

**num = 1)**

**plat3 = Platform(x = games.screen.width - 42,**

**y = games.screen.height - 60,**

**direction = "VERTICAL",**

**hrd = True,**

**num = 1)**

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

y = games.screen.height/2,

game = self)

**enemy = Enemy(x = games.screen.width - 10,**

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

platform.activate()

plat2.activate()

plat3.activate()

pharaoh.activate()

**enemy.activate()**

**enemy.gravity = .05;**

pharaoh.gravity = .05;

games.screen.mainloop()

This adds two small hard Platforms that will act as walls for the Enemy to run into. It also adds an Enemy and gives it the same gravity that Pharaoh has.

Here is an example of what you may see if you run the code and play around in the game a little: 