Pharaoh Tutorial 2:

Materials:

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

We will modify:

* Pharaoh class
* PlatformTile class
* Game class

Instructions

First, add two if statements to the update method in Pharaoh:

def update(self):

super(Pharaoh, self).update()

**if not self.overlapping\_sprites:**

**self.jumping = True**

**if self.jumping:**

**self.dy += self.gravity**

….

Here we start looking at collision. In the first statement, we check to see if Pharaoh is colliding with other sprites. If not, then he isn’t touching the ground or another platform, so we set jumping to true.

The next if statement adds gravity to dy if Pharaoh is jumping—this will make Pharaoh accelerate downward as long as he is in the air and not on a platform.

Initially, gravity is set to zero, so add a line of code to Game’s \_\_init\_\_ method to give it a higher value:

class Game(object):

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

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()

**pharaoh.gravity = .05;**

games.screen.mainloop()

Now, if you run the program, Pharaoh will fall straight down. But you’ll notice he’ll fall straight through the floor! This is the next part we need to add.

We will modify PlatformTile to have an update method that detects a collision with Pharaoh. Before this, though, we need a few methods to determine if a sprite is standing on TOP of a PlatformTile, or coming up from BELOW or from the LEFT or RIGHT.

Add these methods to PlatformTile:

def spriteOnTop(self, sprite):

return sprite.bottom - self.top <= 7

def spriteOnBottom(self, sprite):

return self.bottom - sprite.top <= 7

def spriteOnRight(self, sprite):

return self.right - sprite.left <= 7

def spriteOnLeft(self, sprite):

return sprite.right - self.left <= 7

**(sprite.bottom returns the Y coordinate of the lowest point of sprite, and self.top returns the Y coordinate of the highest point of self)**

Collision is not always as precise as we would like; two sprites may be overlapping by several pixels when we test for collision, or they may only be overlapping by one or two. In order to tell if one sprite is ‘on top’ of another, it’s not enough to see if the very bottom position of the first sprite has the same Y coordinate as the very top of the second sprite. We have to see if the two coordinates are within a range of each other. By subtracting the two values, you get the number of pixels separating the two points. In testing, I have found 7 to be a reasonable number of pixels to separate the bottom of one sprite and the top of the other and be sure that they will test positive for collision. The same principle works for seeing if a sprite is directly to the left or right of another. (For a more in-depth discussion of why this is, see the end of this tutorial).

Now we are ready to build the update method. Here is the entire method, with explanations of each part afterward:

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

This is a little confusing to look at all at one time, so here is a piece-by-piece look:

super(PlatformTile, self).update()

As before, this is a call to the update method of the class we built PlatformTile on top of (games.Sprite). We always want this to run before we do anything ourselves in this method.

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

Here we have a for loop that loops through all the sprites that are overlapping with this PlatformTile. Inside the loop, first we check to see if the overlapping sprite is an Actor. (Remember, Pharaoh is built on top of Actor). If the sprite is an Actor, first we want to see if the sprite is on top of the PlatformTile. We also want to see if it is currently falling (sprite.dy >= 0). If the sprite is moving toward the bottom of the screen (falling), its dy value will be positive. If the sprite is on top of the PlatformTile and is falling, then we set sprite.jumping to false (the sprite is no longer ‘in the air’), and we set the bottom of the sprite equal to the top of the PlatformTile. This keeps the game images looking uniform—whenever a sprite lands on the PlatformTile, it will stay on top and not ‘fall into’ the PlatformTile.

Here is the next section:

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

This ‘elif self.isHard…’ statement is the ‘else’ clause for the “if self.spriteOnTop…” in the previous section.

Here is where PlatformTile.isHard comes into play—this section of code will only run if isHard is set to True. Remember, a PlatformTile with isHard = True is to be treated as a wall, stopping sprites from walking through it. The first thing we do is make sure the sprite is not standing on top of the PlatformTile. (If this PlatformTile is acting like a wall, we are more interested in the sprite coming from the sides or below). If the sprite is on the left, then we set sprite’s rightmost position to PlatformTile’s leftmost position, thus keeping the sprite from moving ‘through’ the wall. We essentially do the same thing if the sprite comes from the right—just set the sprite’s leftmost position to the PlatformTile’s rightmost. Last, we take care of the bottom. If the sprite comes from below, there are two things we need to address; first, we set the top of the sprite to the bottom of the PlatformTile (the exact opposite of what we did if the sprite landed on top of the PlatformTile), and then we reverse sprite.dy, the sprite’s speed going up or down. This way, if the sprite jumps up into the bottom of the PlatformTile, it will be ‘reflected’ back down at the speed it hit the platform.

There is one more thing we want to add: we want Pharaoh to be able to jump. Right now, he will fall if he walks off the edge of a platform or is already in mid-air, but a platformer isn’t a platformer without a jump button. We’ll need to add one if statement to Pharaoh’s update method:

def update(self):

super(Pharaoh, self).update()

if not self.overlapping\_sprites:

self.jumping = True

if self.jumping:

self.dy += self.gravity

if(self.left < 0):

self.left = 0

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

self.right = games.screen.width

… … … …

elif not self.jumping:

self.walkTimer = 0

self.activeImage = 1

self.image = self.slide1

self.dx = 0

**if games.keyboard.is\_pressed(games.K\_SPACE):**

**if not self.jumping:**

**self.dy = -4**

**self.jumping = True**

**self.image = self.slide2**

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

…

This works very much like the arrow key input from the last tutorial. First, we check to see if the space bar is pressed. Then, if Pharaoh.jumping is not already true, we set dy to -4. This will give Pharaoh an upward velocity. Then we set jumping to True since he now is in the air, and we set the image that gets drawn to the screen to slide2 (Pharaoh stepping). This line isn’t 100% necessary, but it is a better visual of jumping than is Pharaoh just standing still as he moves up into the air.

That completes the changes you have to make to give Pharaoh the ability to jump and to handle collisions between Pharaoh and PlatformTiles. To test it, just add a few more platforms to the Game, like so:

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 = True,

num = 10)

**platform2 = Platform(x = games.screen.width - 100,**

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

**direction = "VERTICAL",**

**hrd = True,**

**num = 4)**

**platform3 = Platform(x = games.screen.width/2 - 193,**

**y = games.screen.height/2 + 178,**

**direction = "HORIZONTAL",**

**hrd = False,**

**num = 2)**

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

y = games.screen.height/2,

game = self)

platform.activate()

platform2.activate()

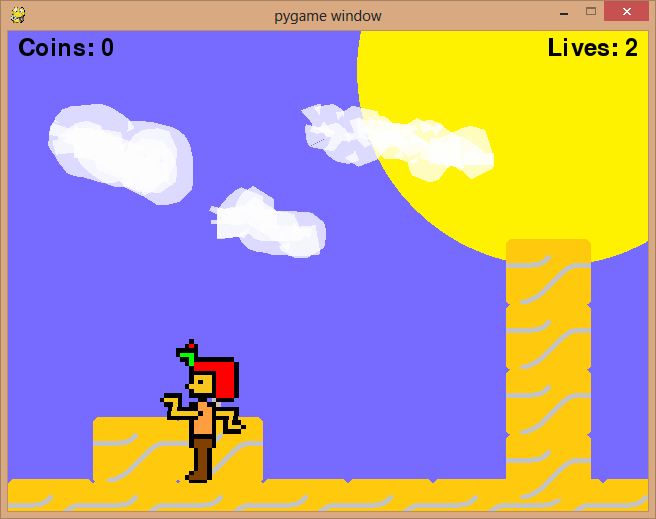
platform3.activate()

pharaoh.activate()

pharaoh.gravity = .05;

games.screen.mainloop()

Here are some screenshots of what you should see and be able to do:



EXPLANATION OF SPRITE COLLISION

Sometimes, when we check to see if two sprites are colliding, we may not actually ‘see’ them collide until they are overlapping each other by many pixels. To our eyes, we can’t tell much of a difference between 2 pixels and 4 or 6 pixels, but that can matter to the computer. Let’s say Pharaoh was standing on a platform, and the space bar is pressed, making him jump. This sets his dy value (speed going up or down) to -4. This means with every new frame that gets generated, 4 is subtracted from Pharaoh’s Y coordinate. If he is standing directly on top of a PlatformTile and is overlapping it by 1 pixel, there is no problem because in the next frame Pharaoh will be drawn a total of 3 pixels higher than the PlatformTile. But if Pharaoh is overlapping the tile by 5 pixels, in the next frame he will still be overlapping the tile by 1 pixel. This will still register as a collision with the PlatformTile. In this case, we would not want the two sprites to still be colliding after Pharaoh has jumped.

This is an even bigger problem when Pharaoh is falling onto a PlatformTile. Gravity may be set to .05, but that .05 gets added to Pharaoh’s downward velocity every frame. At 50 frames-per-second, after two seconds of falling he is going 5 pixels per frame. Let’s say he was one pixel higher than a PlatformTile while falling at this rate. In the next frame, he would be overlapping the PlatformTile by 4 pixels. Our eyes couldn’t tell the difference very well, but if we decided Pharaoh was only ‘on top’ of a PlatformTile when Pharaoh.bottom = PlatformTile.top, then we wouldn’t be able to recognize the Pharaoh had landed on the PlatformTile, and he would fall straight through. This is why, to see if Pharaoh is on top of a PlatformTile, we check to see if the bottom of Pharaoh is within 7 pixels of the top of PlatformTile. If you change gravity to a different value, particularly a higher value, you may need to re-adjust this range to a larger number.