Platform Tutorial:

Materials:

* Python 3.3.3
* Pygame
* LiveWires
* Background.png
* Platform.png

We will construct:

* Game class
* PlatformTile class
* Platform class

Before starting, take a look at the image files so you know what to expect.

Instructions:

Import livewires and initialize the game screen:

from livewires import games, color

games.init(screen\_width = 640, screen.height = 480, fps = 50)

Assemble the Game class:

class Game(object):

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

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

games.screen.background = Game.BACKGROUND

games.screen.mainloop()

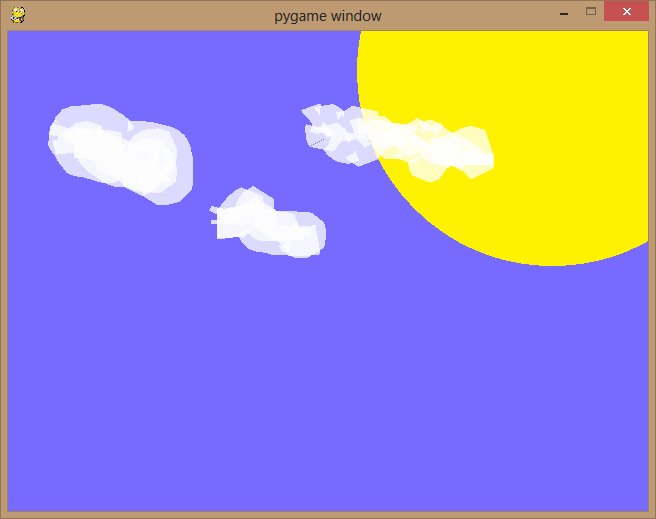
Add a main method to the program and call it at the end:

def main():

gm = Game()

#Start main

main()

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

We will build the game off of this class.

Next, build the PlatformTile class:

class PlatformTile(games.Sprite):

IMAGE = games.load\_image(“images/platform.png”)

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

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

image = PlatformTile.IMAGE,

x = x, y = y)

self.isHard = hrd

X and Y are coordinates, and IMAGE is the image we assign to the sprite.

Notice we are building this off of games.Sprite; this way we can use PlatformTile as a sprite itself.

We are defining isHard for use later. There are some platforms we will want to act like walls, and stop the player from walking through. However, there are others that we may want the player to be able to walk through from the side, but stand on from the top. If isHard = True, the platform will be a wall, but for isHard = False, the player will be able to walk through the platform from the side or jump up onto it from directly below. We will explore this in a later tutorial.

We also need to define activate and deactivate methods, that we will use to add or remove the PlatformTile from the game:

def activate(self):

games.screen.add(self)

def deactivate(self):

self.destroy()

This finishes the PlatformTile class.

But it’s very tedious to place individual tiles to build the game, since each tile is very small. Also, most tiles will be placed right beside other tiles to make the ground or floating platforms, or even walls. To make it easier to build a level, we will build the Platform class, which will draw a series of PlatformTile sprites in a straight line, either horizontally or vertically.

To start the Platform class:

class Platform(object):

def \_\_init\_\_(self, x, y, hrd, direction, num):

self.tiles = []

Notice we are NOT building Platform on top of games.Sprite or PlatformTile: this is because Platform only needs to act as a ‘container’ for several PlatformTiles.

X and Y are the coordinates of the first PlatformTile, and hrd will determine the isHard value for each PlatformTile in the Platform. Direction can have the value “VERTICAL” or “HORIZONTAL”; this determines if this is a vertical or horizontal platform. Lastly, num is the number of PlatformTiles that will be in the line.

Next, we need to add a for loop to the \_\_init\_\_ method to actually assemble the PlatformTiles:

for i in range(0,num):

xval = x

yval = y

if direction == “HORIZONTAL”:

xval = x + 85 \* i

else:

yval = y + 65 \* i

plat = PlatformTile(x = xval, y = yval, hrd = hrd)

self.tiles.append(plat)

Each time this for loop iterates, it will make a PlatformTile with coordinates xval and yval. The if-else statement will modify the coordinates so that a straight line of PlatformTile sprites will be created; if the direction is HORIZONTAL, xval is increased by the width of a PlatformTile sprite (85 pixels, the width of platform.png). Likewise, if it is VERTICAL, yval will be increased by the height of a PlatformTile sprite (65 pixels). This finishes the \_\_init\_\_ method.

Now, add an activate and deactivate method so we can add each PlatformTile to the screen or remove all of them:

def activate(self):

for plat in self.tiles:

plat.activate()

def deactivate(self):

for plat in self.tiles:

plat.deactivate()

Lastly, go back to the \_\_init\_\_ method of Game. We will create a Platform object, then activate it:

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

games.screen.background = Game.BACKGROUND

**platform = Platform(x = games.screen.width/2 – 85,**

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

**direction = “HORIZONTAL”,**

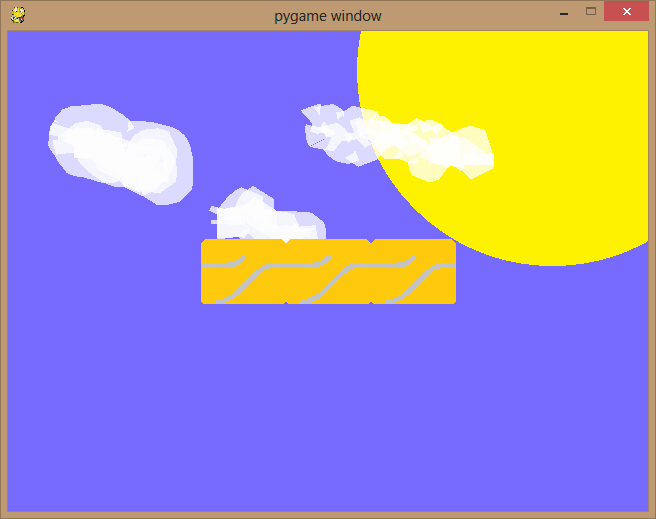
**hrd = True,**

**num = 3)**

**platform.activate()**

games.screen.mainloop()

If you run the program, it should look like this:



You can see the three PlatformTiles in the middle, created by using the Platform class.

This is the first step toward making Pharaoh-Nhiet!