**Animation Part III**

In this lesson, we will take a character and animate it. We start by taking a bitmap:



Save this bitmap in your programming directory as “mario.png”. The bitmap contains seven sprites of a famous character. We will load the bitmap and then split it into separate sprites and put them into an array. Here’s how:

1. Load the bitmap

For this, we start from scratch. Here is our basic code. Copy, paste and save it.

import pygame

width = 820

height = 640

size = width, height

screen = pygame.display.set\_mode(size)

greenish = 25,255,2 # RGB colour code

screen.fill(greenish)

marios = pygame.image.load("mario.png") # 7 Marios

marioX = 50

marioY = 200

screen.blit(marios, (marioX, marioY)) # draw the bitmap

gameOn = True

while gameOn:

for event in pygame.event.get():

if event.type == pygame.QUIT:

gameOn = False

pygame.display.flip()

pygame.quit()

1. Split bitmap into individual sprites

A small animated character is called a **sprite**. We will create an array of seven sprites. Recall we split the elephant bitmap into four pieces in lesson 39. We will have to do this again with Mario. These are the steps:

First, take the line I’ve highlighted in red out. We will replace it with these steps:

* Get the width and height of the bitmap

marioWidth, marioHeight = marios.get\_size()

* Get the width of one mario by dividing by 7:

marioWidth = marioWidth/7

* Create an empty array for the sprites

marioPic = []

* Split the bitmap into seven pieces. I’ve started the code, you finish it:

For i in range(7):

marioPic.append( # you’ll have to go to lesson 39 to see how this is done!

1. Draw a sample sprite to make sure it works

screen.blit(marioPic[3], (marioX, marioY))

Now we have an array of sprites at our disposal. To get them to move, we will draw them, one after the other, in our game loop. Here’s the idea:

1. Inside the game loop, draw the sprite. We will add this to the KEYDOWN section:

gameOn = True

while gameOn:

for event in pygame.event.get():

if event.type == pygame.QUIT:

gameOn = False

elif event.type == pygame.KEYDOWN:

screen.blit(marioPic[1], (marioX, marioY))

pygame.display.flip()

pygame.quit()

When you run and press any key, the picture shifts, a bit. But not much happens because we are redrawing the same picture inside the game loop. We must replace the index[1] with a variable that changes. Do you remember how to make a counter?

1. Create an animation counter

Our counter will count from 0 to 6 and then back to 0. Above the game loop, put this line:

walkCounter = 0

Now inside the if KEYDOWN statement, the counter is our index to keep track of which Mario sprite we are drawing:

elif event.type == pygame.KEYDOWN:

screen.blit(marioPic[walkCounter], (marioX, marioY))

# put line in here that makes the walk counter go up by one.

The line after the screen.blit() function is where we make the counter count. Add that line.

Once you have this working, run the code. Mario should “walk” when you press any key.

1. Reset the walk counter

As you may have noticed, every time the counter gets to 7, the program crashes. Why is this?

There are only 7 sprites, numbered 0 to 6. So we need to reset the counter to zero when it reaches 7. Write an IF statement that does this.

1. Get rid of the past Marios

Now it works, but there is a big mess. We need to erase the previous pictures. The sledgehammer method of doing this is to simply redraw the background before redrawing your sprite.

Put this line IMMEDIATELY BEFORE you draw the sprite in the game loop.

screen.fill(greenish)

This gives you the basics. Some suggestions for future game considerations:

Make the keys work so that you don’t have to continually press them to move.

Add background elements (say a tree) that move toward Mario as he walks. This is called scrolling.

Make Mario walk forward instead of staying in one place.