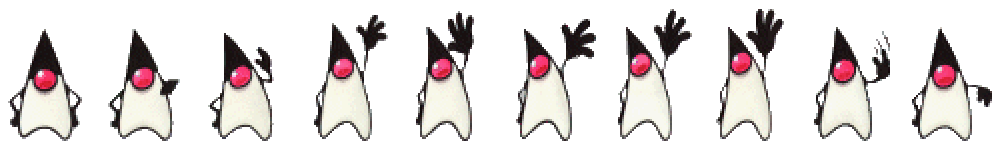
Pygame lesson 3

In this lesson you are going to learn a bit about animating. You will need this image for the first part:
http://openbookproject.net/thinkcs/python/english3e/_images/duke_spritesheet.png



This is an example of a spritesheet – a collection of related images gathered into one file.

Treating the spritesheet like a normal image.
In this example, I have imported the spritesheet just like a normal image. No animation here… Use this code to get yourself to this same point:

```
# An introduction to spritesheets
# sources: http://openbookproject.net/thinkcs/python/english3e/pygame.html

import pygame
pygame.init()

# Set-up variables
WHITE = (255, 255, 255)
size = (700, 500)
x_pos, y_pos= 100,100

# Load the sprite sheet
duke_sprite_sheet = pygame.image.load("duke_spritesheet.png")

# Create a character
guy = pygame.image.load("duke_spritesheet.png")

# Set the width and height of the screen
screen = pygame.display.set_mode(size)
pygame.display.set_caption("Spritesheet experiment")
done = False

clock = pygame.time.Clock()

# -------- Main Program Loop -----------
while not done:
    # --- Event Processing
    for event in pygame.event.get(): # check if key pressed
        if event.type == pygame.QUIT:
            done = True
    # Clear the screen and set it to white.
    screen.fill(WHITE)
    # ---------- Drawing code ----------
    screen.blit(guy, (100, 120))
    # --- This updates the screen and sets the frame rate
    pygame.display.flip()
    clock.tick(60)

# Close the window and quit - it is important to quit the game
pygame.quit()
```

Animating using a spritesheet.
The key to this is only showing part of the image. This is done by making a rectangle to say which part you want to show. It works well for this image (and others like it) because all the ‘frames’ are exactly the same size. I recommend looking for this when animating.

First of all, we are going to need some way to count which frame we are on. I did that using a variable, ‘pos’, added to the variable declaration section at the top of the program.

```
x_pos, y_pos= 100,100
pos = 0 # used for counting through the spritesheet
```

Then I made the code that reveals a rectangle of the spritesheet and cycles through each position:

```
# ---------- Drawing code ----------
screen.blit(guy, (100, 120),(pos*50,0,50,72))
pos = (pos+1)%10
```

Note the rectangle works using four parameters:

- First – the x coordinate of the top left edge of the rectangle
- Second – the y coordinate of the top left edge of the rectangle
- Third – the width of the rectangle (this is why it is so ideal to have them all the same width
- Fourth – the height of the rectangle

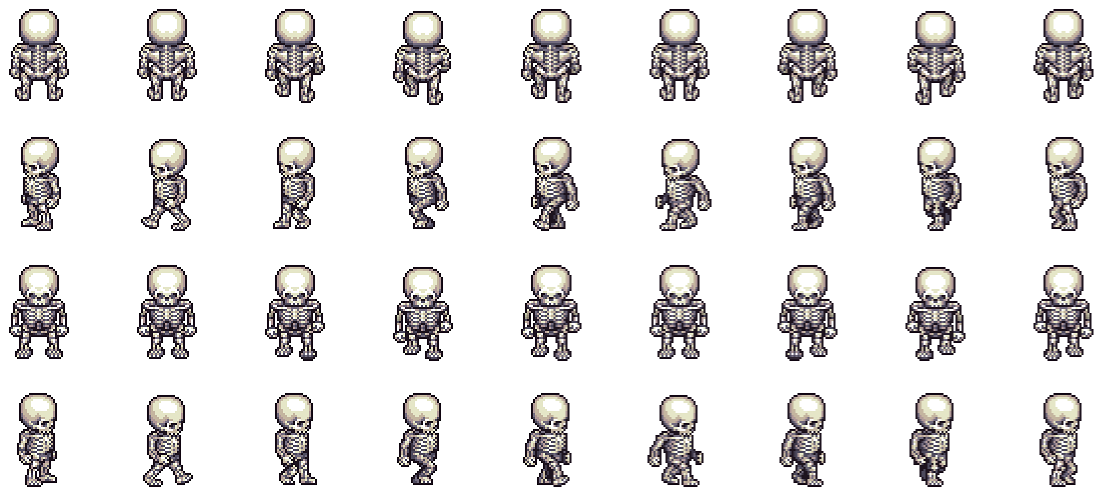
The use of ‘pos’ (which starts at 0) means we move along 50 pixels at a time.

The next part cycles through the frames of the spritesheet. The %10 means that when we get to the last frame, we go back to the start.

Run this code and you will have an animation!!!

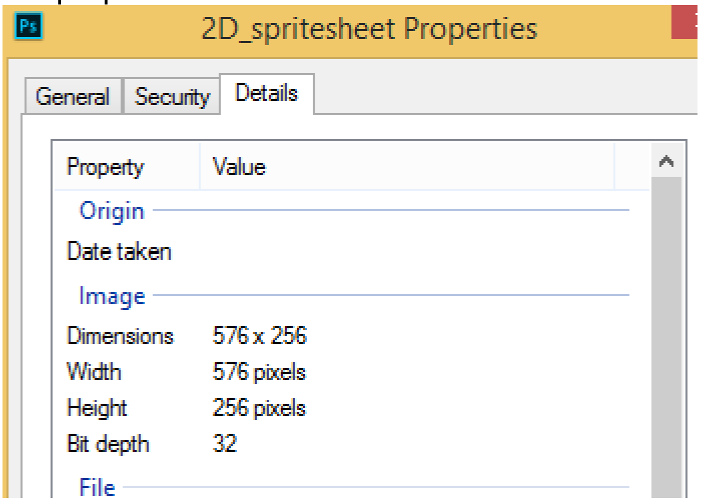
Challenge
Make it so you can control the character with mouse arrows.

Working with a 2D spritesheet
This spritesheet can be used to give a more ‘realistic’ sense of movement. When we are moving to the left, we can make the sprite point to the left, etc.



Fetch a copy for yourself from here:
https://gamedev.stackexchange.com/questions/127767/getting-sprites-from-a-spritesheet-with-rows-and-columns
Or here: https://i.stack.imgur.com/C3ZwL.png

I need to figure out how wide and tall each of the frames are. I did this by opening up the properties of the file:



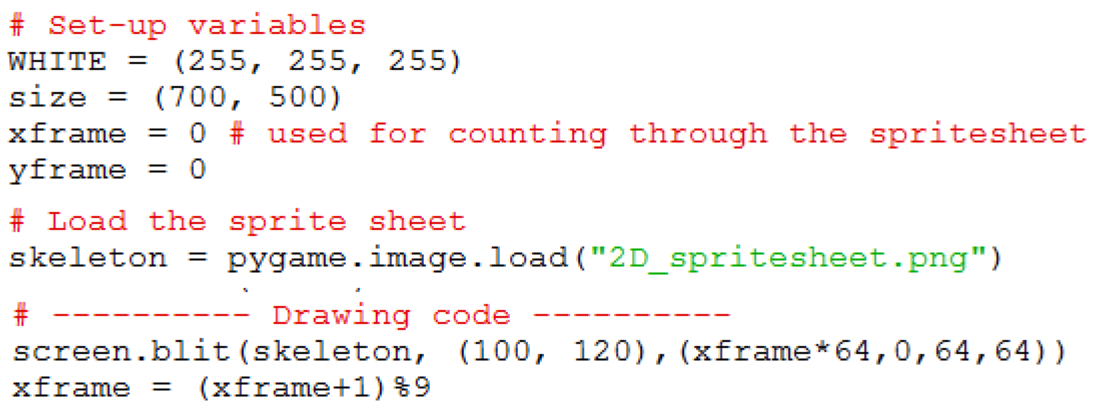
2D_spritesheet Properties

Property	Value
Origin	
Date taken	
Image	
Dimensions	576 x 256
Width	576 pixels
Height	256 pixels
Bit depth	32
File	

Width – each frame is 576/9 = 64
Height – each frame is 256/4 = 64

The same thing can be done with Photoshop, but in this case, this method is easier.

I will modify the code from above to get this spritesheet into my program. Here is my first attempt.

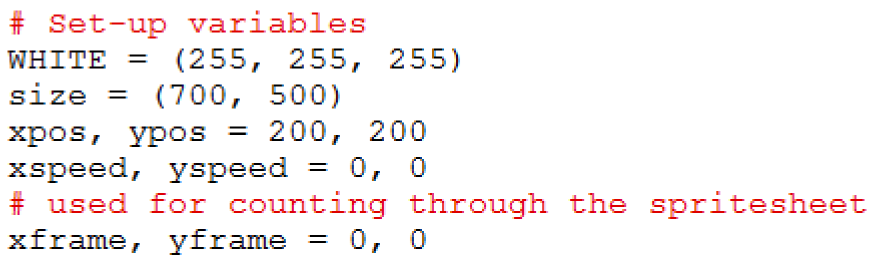


```
# Set-up variables
WHITE = (255, 255, 255)
size = (700, 500)
xframe = 0 # used for counting through the spritesheet
yframe = 0

# Load the sprite sheet
skeleton = pygame.image.load("2D_spritesheet.png")

# ---------- Drawing code ----------
screen.blit(skeleton, (100, 120),(xframe*64,0,64,64))
xframe = (xframe+1)%9
```

This version uses the bottom row of the spritesheet. My goal is to change row depending up on which arrow key is pressed. I will need some variables to move the character.



```
# Set-up variables
WHITE = (255, 255, 255)
size = (700, 500)
xpos, ypos = 200, 200
xspeed, yspeed = 0, 0
# used for counting through the spritesheet
xframe, yframe = 0, 0
```

I went and got the code from an earlier tutorial for moving the sprite in response to the keyboard. After some fiddling around, here is what I got to (see below).

```
# -------- Main Program Loop -----------
while not done:
    # --- Event Processing
    for event in pygame.event.get(): # check if key pressed
        if event.type == pygame.QUIT:
            done = True

        elif event.type == pygame.KEYDOWN: # It was an arrow key.
            if event.key == pygame.K_LEFT:
                xspeed = -3
                yframe = 1
            elif event.key == pygame.K_RIGHT:
                xspeed = 3
                yframe = 3
            elif event.key == pygame.K_UP:
                yspeed = -3
                yframe = 0
            elif event.key == pygame.K_DOWN:
                yspeed = 3
                yframe = 2
    # Clear the screen and set it to white.
    screen.fill(WHITE)
    # ---------- Drawing code ----------
    xpos += xspeed
    ypos += yspeed
    screen.blit(skeleton, (xpos, ypos),(xframe*64,yframe*64,64,64))
    xframe = (xframe+1)%9

    #update variables
    xspeed, yspeed = 0, 0

    # --- This updates the screen and sets the frame rate
    pygame.display.flip()
    clock.tick(10)
```

This works, but I had to fiddle around with the ‘KEYDOWN’ options to get the effect I wanted from the ‘yframe’ variable.

You can find loads of spritesheets online. Try some of your own and see what you can make.