**Manipulating Images**

Note: Each lesson in this unit assumes you have completed the previous lesson. We won’t be writing code from scratch for these lessons, just building on the previous code. If you are not sure where things go, or why it isn’t working, please ask your teacher, or refer to previous lessons.

In the last lesson we drew a picture of a ball on the screen. The coordinates of the ball was set using two variables:

ballX = 0

ballY = 0

This places the top left of the picture at coordinates (0,0).

Notice that the **origin** for our window is at the **top left**. This is a little different than in math class, where the origin of the XY plane is at the middle or bottom left of a graph. The reason why the origin is at top left is historical: when we type on the computer, we start the page at the top left and make our way down the page, so it makes sense to make the top left corner the origin in computing.

Transferring an image is called a Bit BLT (pronounced “bit blit”), which comes from “Bit BLock Transfer”. This is the line that does this:

screen.blit(ball, (ballX, ballY)) # draw the ball

What is a bit block transfer?

When transferring a picture (a block of bits) onto the screen, many pixels need to be copied over, one pixel at a time - an intensive procedure for the **CPU** (the Central Processing Unit - the computer’s brain). Python’s **blit** function is written in a low level language to speed the process up. Also, to make things easier on the CPU, the **GPU** (graphical processing unit) does the job instead. The GPU is a separate chip that takes care of graphics calculations only so the CPU can take care of other things.

**The Transform Class**

You have probably learned about transformations in math class. Common transformations include reflection about an axis, rotation and translation. Python has a class called the Transform class. Inside the Transform class are several functions that allow us to transform images:

1. **pygame.transform.scale()**

This function returns a scaled image. It takes two parameters:

**surface**: the image you want to scale (e.g. ball)

**size**: a tuple representing the final size (e.g. (20,20))

Example newpic = pygame.transform.scale(pic, (50,50))

Notice the tuple (coordinates) must be in parentheses.

1. **pygame.transform.flip()**

This function takes an image and returns the flipped transform. It has three parameters:

**surface**:the image to flip

**xbool**: True to flip about the x axis

**ybool**: True to flip about the y axis.

This example will flip a picture upside down:

**newpic = pygame.transform.flip(pic, True, False)**

1. **pygame.transform.rotate()**

Two parameters: surface and angle in degrees.

**newpic = pygame.transform.rotate(pic, 30)**

**Exercise:**

Create a new image (or take a .png from the internet) and try flipping it, scaling it and rotating it in various ways.

NOTE:

You may use MS Paint to create your own bitmaps. Later on in this course it may be useful to have a better image editor to work with. If you are taking Comm Tech then you probably have access to Photoshop and other editing software. If not, the image editor I recommend is “paint.net”. It is similar to Photoshop only it is free. Get it here:

<http://www.getpaint.net/>