

Extension: Projectile motion!

The bird doesn't currently fall according to physics, so to make our game easier to play and look better we will be implementing better motion

Task 1.1: The projectile motion formula

$$y_t = y_0 - ((v_0 \times \sin(\theta) \times t) - (\frac{1}{2} \times g \times t^2))$$

Where:

y_t = the value of y at t seconds

y_0 = the value of y at 0 seconds

v_0 = the velocity at 0 seconds

$\sin(\theta)$ = sin of the initial angle the projectile is "thrown at"

t = the time in seconds

g = gravity

Task 1.2: Using the formula!

Let's implement this

1. Go to directly under where the bird is created
2. Make a variable for:
 - y_0 ← The bird's original y value
 - v_0 ← Should be 80 (You can play around with this after it works)
 - t ← Should start at 0
 - g ← 9.81
3. Inside the update function, after you make these variables global, you will need to increase t by 0.1 (the function is run roughly 10 times per second). Where you're currently updating bird's y value you will need to substitute in the formula
$$\text{bird.y} = y_0 + (0.5 \times g \times t^2)$$

The formula has changes a bit from the original as the bird falls at an angle of 0° and $\sin(0)$ is 0 so z bit of the formula cancels out

Hint

To use exponents (powers) in python it should look like:

```
squared = x**2
```

Remember to make a variable global in a function, you need to write this at the top of the function:

```
global myVar
```

Task 1.3: Using the formula pt. 2!

Now that the bird is falling more gracefully test the code. Hm.. something looks wrong

This is because we're not resetting any of the values we need to when the bird moves upwards when we click. In the `on_mouse_down()` function you need to reset `t` to 0 and you need to set `y0` to the bird's `y` coordinate after it has moved upwards

★ BONUS 7.4: Faster and Faster!

Waiting for the next lecture? Try adding this bonus feature!!

Now that you have mastered basic projectile motion, play around with some of the values and make the bird's descent faster or slower. You could even try importing the `math` module to use `sin` and change the initial projectile angle