Girls' Programming Network

Flappy Bird!

Extensions

This project was created by GPN Australia for GPN sites all around Australia!

This workbook and related materials were created by tutors at:

Sydney, Canberra and Perth

Girls' Programming Network

If you see any of the following tutors don't forget to thank them!!

Writers

Sasha Morrissey
Renee Noble
Courtney Ross
Joanna Elliott
Amanda Meli
Adele Scott
Sheree Pudney
Caitlin Bathgate
Alex McCulloch

Testers

Emily Aubin
Samantha Rampant
Leanne Magrath
Melody Wong
Vivian Dang
Annie Liu
Rosey Stewart
Natalie Bartolo
Jeannette Tran
Gaya Pilli
Cindy Chung
Stephanie Chant
Sam Criddle

Extension: Gravity

The original Flappy Bird game had the bird go down on its own with gravity and you had to keep it up by flapping the wings

Task 1.1: Fall Bird!

Just like how we make the pipes move from right to left by adding to their pipe_x we are going to do the same thing but we're going to add to the bird_y

Add 1 to bird_y after you update all the pipe_x variables

Task 1.2: Different planets!

Different planets have different amounts of gravity! Try changing how much you are adding to the bird_y variable and see how that changes the game

☑ CHECKPOINT ☑

If you can tick these off you have finished the extension!

- ☐ Your bird falls down if you don't press any buttons
- ☐ Your bird still goes up when you press the up button
- ☐ You have played with different numbers for gravity

Extension: Flappy Wings

The game is called Flappy Bird but there isn't much flapping going on! Let's animate the wings so that they flap!

Task 1.1: Download the images

First we need to download the images that we need. We have 2 images to download, one with the wings up and one with the wings down.

Once you've downloaded the images load them into pygame and call them bird_wings_up_image and bird_wings_down_image

Hint

Make sure you save the images to the same place as your code

Task 1.2: Keeping track

So that we can know what our wings are doing let's make a variable before the while loop called wings_position and set it to "up"

Task 1.3: Lift the wings

Now where we blit our bird image we are going to replace it with an if statement.

If the wings are up then we should blit the bird_wings_up_image and update the wings_position to be "down"

Task 1.4: Lower the wings

Now we need to put the wings down by adding an elif statement right after the first.

If the wings are down then we should blit the bird_wings_down_image and update the wings_position to be "up"

Hint

Make sure you use an elif for the second statement otherwise the wings will always be up

Task 1.5: Too fast!

Those little wings are flapping so fast we can hardly see them! Let's count to 10 before we change the position so that our bird doesn't run out of puff!

Make a variable called count before the while loop and set it to 0. Now every time we blit the bird (for both positions) we add one to the count and check if the count is greater than 10. If it is then we can update the wing_position and set the count back to 0

☑ CHECKPOINT ☑

If you can tick these off you have finished the extension!

- ☐ The bird flaps its wings as the game runs
- ☐ Try running your code!

Extension: Faster and faster!

In case the game isn't tricky enough, let's make it get faster as it goes.

Task 1.1: Set the speed

To keep track of the speed let's make a variable before the while loop called speed and set it to 2 because that's the speed that the pipes are currently moving.

Where we minus 2 from the pipe_x instead minus speed. Make sure you do this for all 3 pipes.

Task 1.2: Faster faster faster!

Now that we have a variable we can make the speed change!

After we set the new pipe_x position we can add one to the speed which will make our pipes move faster and faster

Task 1.3: Too fast!

Wow! That's really fast! Our speed is getting too big too quickly so we need to add a number that is smaller than one.

Try adding 1/800 to the speed instead of adding one. You can play around with this number to find what you like best

☑ CHECKPOINT ☑

If you can tick these off you have finished the extension!

- ☐ The pipes start of at a slower speed and slowly get faster
- ☐ Run your code!

Extension: Score!

Our flappy bird game is pretty good but there's no way to keep score and see how well you're doing.

Part 1: Keeping time

Task 1.1: What's the time?

Pygame can tell us how long it's been since the game started! Let's use that as our score. The longer you play the game the higher your score is.

In our if statements where we check if the bird has collided with a pipe and end the game let's add a score that is the time the game has been running divided by 1000. It should look like this when you're done:

```
Game Over!
You lasted: 3 seconds!
```

Hint

To get the number of milliseconds from pygame you can use this code:

```
time_in_milliseconds = time.get_ticks()
```

☑ CHECKPOINT ☑

If you can tick these off you can move on to Part 2!

☐ Your game prints how many seconds you lasted when the game ends.

☐ Run your code!

Part 2: Play again

Task 2.1: Waiting

If we want to keep track of our scores over multiple plays it might be good to be able to play again right away without having to rerun our code!

Let's make a "waiting" variable to keep track of whether or not we are waiting to play again. Before the while loop make a variable called waiting and set it to False. Then in the code where we print "Game Over" set waiting to True (make sure to do this for all your pipes)

Task 2.2: No really, we're waiting!

The game isn't really waiting yet so let's fix that.
If we are meant to be waiting let's show a waiting screen. You can download the image here: <URL>

Where we load all our images, load the new waiting screen and call it waiting_image. Then add an if statement right at the beginning of the loop (before we check for new events) that checks if we should be waiting. If we are meant to be waiting then blit the waiting image onto the screen at position 0,0 and update the display.

We don't want the rest of our code to run while we're waiting so add a continue.

Hint

Remember to save all your images in the same place as your code

Task 2.3: I don't want to wait forever!

Now we need to check if the player wants to play again or quit.

After we update the display inside our new if statement but before we continue, check for a new event. Then add an if statement to see if the player has pressed the q key, if they have we should `quit()` the game and `break`

Add an if statement to check if the player has pressed the enter key. If they have then we should set waiting back to False. We also need to reset the game by setting all the bird and pipe variables back to their original values (e.g. bird_y should equal 250 etc.)

Hint

The enter key is K_RETURN

Task 2.3: Those scores aren't right!

Remember how we get the score? We are just seeing how long it's been since the game started, and we can't reset that every time we start a new game.

Let's keep track of what time the game started so that we can work out how long the game is taking. Think of it like if you look at the clock before you run a lap and the time is 12:03 and then if you look at the clock again when you finish your lap and the time is 12:05 you know that it took you 2 minutes to run the lap.

Make a start_time variable before the while loop and make it equal the current pygame time. Then when we work out the score we should get the current time and minus the start time and then divide it by 1000

We also need to update the start time every time we start a new game (so the clock resets). In the if statement that lets us play again where we reset all the variables we should also reset the start_time to be the new current time.

Hint

Remember you can get the current pygame time using this code:

```
current_time = time.get_ticks()
```

To work out how long it's been in seconds since we started it could look like this:

```
score =  (time.get_ticks() - start_time)//1000
```

☑ CHECKPOINT ☑

If you can tick these off you can move on to Part 3!

- ☐ When the game ends you get a screen asking you to play again
- ☐ If you press q on the waiting screen the game quits
- ☐ If you press enter on the waiting screen the game starts again and everything is in its starting position
- ☐ When you play multiple games the scores are right every time
- ☐ Run your code!

Part 3: High Score

Task 3.1: Who's winning?

Now that we can play over and over again we can keep a high score of all of the games we play!

Make a variable called high_score before the while loop and set it to 0. Then whenever we get someone's score add an if statement that checks if the score is greater than the high score and if it is print out that we have a new high score and what it is. Then update the high_score to be equal to the score that we just got.

☑ CHECKPOINT ☑

If you can tick these off you can move on to Part 4!

- ☐ Every time you set a new high score the game prints it out
- ☐ Run your code!

Part 4: High Score Store

Files

Task 4.1: Long term storage

Right now our high score is only kept until we quit the game and then it's gone. Let's store the score in a file so we can keep it every time we play.

Download the highscore.txt file and save it in the same place you've saved your program

Task 4.2: A bit of light reading

Let's read our high score from the file instead of setting it to 0 at the beginning.

Open the file, then set high_score to equal what is read from the file (don't forget to convert it to a number!), then print what the current high score is.

Hint

You can read from a file using code like this:

```
with open("age.txt") as file:
    age = int(file.read())
```

Task 4.3: Write it down

Now we need to make sure we store our high score back in the file for next time.

After the while loop, open the file again in write mode and write the high_score into the file.

Hint

You can write to a file using code like this:

```
with open("age.txt", "w") as file:
    file.write(str(age))
```

☑ CHECKPOINT ☑

If you can tick these off you have finished the extension!

☐ If you play multiple games, quit and then run your code again it will remember your high score