Flappy Bird Extensions!

Extension 1: Printing Flappy Bird!

We can print words to the screen as well as displaying images! Let’s print the words Flappy Bird to the screen so the user can’t forget what game they're playing.

Extension 1.1

1) Store what colour you’d like to use in a variable called `colour`. You’ll need to use the RGB values.
2) Store what size you’d like to use in a variable called `font`.
3) Store where you’d like the text to appear in a variable called `location`.
4) Blit the text to the screen!

Hint: *Your cheat sheet will help you remember the format that these variables require!*

Extension 2: Print the user’s score!

Now that we have Flappy Bird up and running, wouldn’t it be nice to tell the user how many pipes they managed to get past? Let’s print the final score to the screen!

Extension 2.1

1) In the `for` loop where you create the list of pipes, add a new variable called `pipe[‘number’]`, and store the value of `i`.
2) When a collision is detected, print the value of `pipe[‘number’]` of the pipe that was collided with to the screen.

Extension 3: Make Flappy Bird Fall!

In the real Flappy Bird game, the Flappy Bird is constantly falling! Update the game so your Flappy Bird does the same.

Extension 3.1

1) Set the initial value of the variable `moving` to `down`.
2) In the code where you check to see what arrow key was pressed, update the code so if no key is pressed the value of `moving` is `down`.

Extension 4: Faster Flappy Bird

Some of our users are so good, they can get past all the pipes. Let's make the game harder for them by speeding up the game the longer they play.

Extension 4.1

1) Create a variable called `speed` and give it a value of `1`.
2) Update the code that makes the pipes move across the screen, and subtract `speed` from `pipe['x']`.
3) Create a new `if` statement where:
    a) If the user has passed less than 5 pipes, the value of `speed` is `1`.
    b) If the user has passed more than 5 pipes, but less than 10 pipes, the value of `speed` is `2`.
    c) If the user has passed more than 10 pipes, but less than 15 pipes, the value of `speed` is `3`
    d) If the user has passed more than 15 pipes, the value of `speed` is `4`.

Is the game harder now? Play around with value of speed to make the game harder or easier!

Extension 5: Forwards and Backwards

Sometimes the pipes are coming at Flappy Bird at just the wrong speed! Let's make the Flappy be able to move forwards and backwards as well.

Extension 5.1

1) Update the `if` statement that checks what key has been pressed.
    a) If the right arrow has been pressed, set `moving` to `forward`.
    b) If the left arrow has been pressed, set `moving` to `back`.
2) Update the `if` statement that checks what the value of `moving` is.
    a) If the value of `moving` is `forward`, increase the value of `bird_x` by `1`.
    b) If the value of `moving` is `back`, decrease the value of `bird_x` by `1`.

Have fun sending Flappy Bird all over the screen!

Extension 6: Falling Obstacles

Avoid pipes is easy! Let's have some obstacles fall across the screen as well!.

Extension 6.1

1) Store the obstacles picture in a variable called `obs_image`.
2) Create a new `list` called `obstacles`
3) Create a `for` loop that generates the value of `obstacle['x']` and `obstacle['y']` for each obstacle you want to create.
4) Inside the game loop, create a `for` loop that blits the `obstacles` to the `screen`.
5) Update the value of `obstacle['x']` and `obstacle['y']` for each `obstacle` so that they fall diagonally across the screen
6) Detect a collision with the `obstacle`! If Flappy Bird hits an obstacle, it's game over!

***Hint:** If you get stuck, look back at what you did in Part 3 of the first workbook!*