Girls' Programming Network

Flappy Bird!

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

Amanda Hogan
Renee Noble
Isabella Hogan

A massive thanks to our sponsors for supporting us!

ATLASSIAN

Part 0: Setting up

Intro to Python

Task 0.1: Start Programming

Follow the instructions from the lecture to set up your flappy bird python file!

Hint

Don't forget to ask the tutors for help if you're not sure what to do next

Task 0.2: You've got a blank space, so write your name!

At the top of the file edit the comment to write your name!
Any line starting with # is a comment.

```
# This is a comment
```

Task 0.3: Some scaffolding

Now read through the other comments to see the different steps we'll be doing.

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 1:

- ☐ You should have a file called flappy_bird.py
- ☐ Your file has your name at the top in a comment
- ☐ Your file has a bunch of scaffolding comments
- ☐ Run your file with F5 key and it does nothing!!

Part 1: Pygame Zero Essentials

Intro to Pygame Zero

In this section we are going to get Pygame Zero set up, welcome our players, and create our game screen.

Task 1.1: Print a welcome and the rules

Welcome the player and print the rules!

1. **Display** the following welcome message using some **print** statements in the `# print welcome` section

```
The game is about to start!
Click the mouse to "flap" upwards
Dodge the pipes and the floor
Good luck and have fun!
```

Task 1.2: Get Pygame Zero

PygameZero will give us the tools we need to write our game code. Let's import it!

1. At the top of your file, under the start modules comment, **import Pygame Zero**
2. At the bottom of your file, under the runs everything comments, **make Pygame Zero GO!**

Hint

To import and start Pygame Zero we can use this code:

```
import pgzrun
...
pgzrun.go()
```

Task 1.3: Make a screen

Before we make our game we need to set up the screen to display it on!

1. **Make a screen** that is 800 pixels wide and 600 pixels tall. This should be under the create constants comment

2. **Run** your code! Do you see a big dark box?

Hint

To make a screen that is 100 pixels wide and 200 pixels tall you could use this code

```
WIDTH = 100
HEIGHT = 200
```

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 2:

- ☐ Print a welcome
- ☐ Print the rules
- ☐ Try running your code!

Part 2: Images!

Task 2.1: Get all the images!

We need to load all the images that we need for our game!

1. **Download** the images onto our computer. Go to https://www.girlsprogramming.network/flappy-bird-images and download all the images there.

2. **Double click** on the zip folder to *unzip* the the folder to get to the images

3. **Move the images folder** into the "flappy_bird" folder

Task 2.2: Backgrounds are better!!

Let's start our game by adding the background to the screen! We just downloaded the pic! We'll be using a **function** to **draw** things to our screen.

1. Under the comment "make background" make a new Actor() called `background` with an image of "bg".

2. Directly under that make it so that the actor's x value is 400 and y value is 300

3. **Create a function** call `draw()` under the draw everything to screen comment

4. Inside your function, ***draw*** `background` ***to the screen*** This should be under the set background image comment.

5. Make sure the code you wrote in Step 4 is ***indented inside your function***.

Hint: Make an actor!

To make an actor with an x value of 10 and a y value of 25, it should look like this;

```
myCharacter = Actor("characterImage")
myCharacter.x = 10
myCharacter.y = 25
```

Hint: Draw an actor!

We can use draw, since we already programmed the x, y position for our background Actor.

Here's an example:

```
myCharacter.draw()
```

Task 2.3: Use your draw function!

Pygame Zero always **looks** for a function called **draw**!

1. **Run** your code. Pygame Zero will know to use your draw function.
2. **Did you see** the background image?

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 3:

- ☐ Saved all the images to the same place as your code
- ☐ Made the background appear on the game screen
- ☐ Run your code!

★ BONUS 2.4: Funky Backgrounds!!

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

We can use any image as a background!

1. Go on the **internet** to find a cool image that you want to be the background of the game. The image must be a png

2. Save the image into your **images folder**

3. Where you've set the background image replace **"bg"** with the **name of your file**

Try and make sure the new image is a similar size to the original and if it's too small try to make it bigger

Events & If Statements

Part 3: The Bird is the Word!

Task 3.1: Making a bird

This game is meant to be Flappy Bird but there's no bird on our screen! Let's fix that.

1. **Check** that you have the **bird image file called bird.png** where you saved your code. This should be in your images folder.

2. Create an ***Actor*** using the bird image,
   Store it in a variable **named bird.** Do this after the make bird comment

Task 3.2: Where is the bird?

Set the starting position of the bird. This should be directly under where you first create the bird actor.

1. Set the starting **x value to 160**

2. Set the starting **y valueto 300**

Task 3.3: Show me the bird!

Now we're ready to add the bird to the screen!

1. Go to your **draw function**

2. **After** the draw actors comment, ***draw*** your bird the same way you drew your background.
   Make sure this is indented inside your function.

Task 3.4: Make the bird fly

We need to make the bird move downwards on the screen

1. Make a new **function** called `update()` after the "updates everything" comment

2. You need to **increase** the bird's y value **by 1** by adding one to its value under the updating bird comment

Hint

We need to save a new y value to bird. We can do this in a similar way to how we originally set it.

e.g. `myCharacter.y = myCharacter.y + 5`

Task 3.5: Falling off the screen

When our bird touches the bottom of the screen, we want it to end the game.

1. **Import** a new module called `sys` directly under where you `import pgzrun`
2. In the update function under the "bird hits bottom of screen" comment you need to test if the bird's y value is **more** than the **height** of the screen (600)
1. If it is, print `"Game Over!"` then you need to put the line `sys.exit()` to immediately **stop the game.**

Hint

To test if one thing is greater than the other you need an **if statement** similar to this:

```
if myNum > otherNum:
    print("Higher")
```

Task 3.6: Moving the bird

Our bird should "flap" upwards when the mouse is clicked.

1. Make a **new function** called `on_mouse_down()` after the moving comment
2. Inside the function **decrease** the bird's y value by 50

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 4:

- ☐ Run your code!
- ☐ There is a background still on the screen
- ☐ There is a bird on the screen and its falling

☐ When you click the bird moves back up

★ BONUS 3.7: Funky Bird!

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

We can use any image as our bird!

1. Go on the **internet** to find a cool image that you want to be the **character** of the game. Must be a **png**
2. Save the image into your **images folder**
3. Where you've set the bird's image replace **"bird"** with the **name of your file**

Try and make sure the new image is a similar size to the original as otherwise the game may be too hard

Part 4: Pipes!

We'll be making some pipes to make the game playable

Task 4.1: Make a pipe

Let's make a pipe!

1. Create a new actor called `topPipe1` with an image of `"top"`. This should be under the make pipes comment
2. Set it's **x value** to be 266
3. Set it's **y value** to be 0

Hint

Look at your code for your bird actor if you've forgotten how to make one

Task 4.2: Updating it!

Now we need to make it so it shows up and so it moves to the left when the game is run.

1. In the draw function under where you're drawing the bird **draw** the topPipe1 actor
2. Go to the **update function** under the updating pipes comment
3. **Decrease** topPipe1's x value by 1

Task 4.3: Now what?

Now that we have our new pipe we need to make some more. We're going to start with 5 more (three groups of a top and bottom pipe).

1. The **first bottom pipe** should have an **x value** of 266 and a **y value** of 800
2. The **second top pipe** should have an **x value** of 532 and a **y value** of -200
3. The **second bottom pipe** should have an **x value** of 532 and a **y value** of 560

4. The **third top pipe** should have an **x value** of 798 and a **y value** of -120

5. The **third bottom pipe** should have an **x value** of 798 and a **y value** of 710

The bottom pipes should be called something like `bottomPipe1` and should have an image of `"bottom"`.

Task 4.4: Now what? pt.2

Now that we have our new pipes we need to update and draw each one

1. Under where you drew topPipe1, **draw** every other pipe

2. Under where you decreased topPipe's x by 1, **decrease** every other pipe's **x value** by the same amount

Hint

This section is very repetitive it might be helpful to copy and paste the sections of code you need to use

Task 4.5: Colliding!

Run your code and see what happens.

If the bird hits the pipes nothing happens right? Let's make that work.

1. Go to the update function, under the **"bird hits pipes"** comment

2. Make an if statement that tests if the bird has **collided** with topPipe1 and if it has, print `"Game Over!"` then **exit** the game like when the bird hits the bottom of the screen.

3. Make similar if statements for **each pipe**

Hint

To test if one actor has hit another the code should look like:

```
if myCharacter.colliderect(myWall):
    print("Hit!")
```

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 5:

- ☐ The game has some moving pipes
- ☐ The game stops when the bird hits the ground
- ☐ The game stops when the bird hits the pipes

Part 5: A better game!

Task 5.1: Wrapping around!

Our pipes move from right to left but drop off the end of the screen let's make them move to the right side when that happens

1. Under where you **update** topPipe1
2. Write an **if statement** to see if topPipe1's **x value** is less than -44
3. If it is, set it's x value to be the **width** of the screen (800)
4. Now repeat this for **every pipe**

Task 5.2 Scoring

We're going to make it so the player can earn some points to make the game more fun.

1. Make a variable called **score** that starts at zero in your create constants section
2. At the top of the **update function** above the updating bird comment make the variable **"global"**
3. Increase score by one each time a **top pipe** wraps around the screen

Hint

To make a variable global you need to write

```
global myVariable
```

Task 5.3: Now what?

Now that we are detecting when the game is over and we're tracking a score, we should print the final score when the game is over.

1. In the **if statement** that checks whether the bird hits the bottom of the screen, before the sys.exit(), **print** out the score
2. In the **if statements** that check if a bird has hit a pipe, before the sys.exit(), **print** our the score

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 6:

- ☐ Your game ends after the bird collides with the wall or pipes
- ☐ Your pipes are wrap around the screen
- ☐ You are printing a score when the game is over

★ BONUS 5.8: On screen score!

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

We are currently printing the score once the game has finished, but we should try to display the score by drawing it onto the screen in the draw function using code like:

```
screen.draw.text(f"My Number is: {myNum}",topleft=(5,5),
color=(255,255,255)
, fontsize = 30)
```

Play around with the formatting given here. If you want some more formatting options you can go to this link: https://pygame-zero.readthedocs.io/en/stable/ptext.html

6. Extension: Game Over!

We're going to make a screen that tells you the game is over

Task 6.1: Making a game over variable!

We need to store whether it is game over or not

1. In the create constants section under the other variables, create a boolean variable called **gameOver**

2. Initially set gameOver to be **False**

Hint

To make a boolean variable it should look like:

```
myFlag = True
```

Task 6.2: Using the flag

We need to store when the game is actually over.

1. At the top of the **update function** above the updating bird comment make the **gameOver** variable **"global".**

2. Go through the **update()** function.

3. Every line that says sys.exit() should be replaced with a line that sets **gameOver** to be **True**

Task 6.3: Making a game over screen!

If the game is over, it should display a different screen.

1. In your **draw()** function, before the set background image comment, write an **if statement** that tests if gameOver is **True**

2. If it is, **fill the background** with a solid colour and **display text** that says something like; “Game Over! Your score was:” then their score

3. Make your code to **set the screen** and **draw the actors** be inside the **else** section of this if.

4. Go to the **update()** function and make an if statement so that everything that’s currently in the update function only happens if the game isn’t over.

Hint

To display a single colour as a background the code looks something like:

```
screen.fill((0,0,0))
```

To write text to the screen the code looks something like:

```
screen.draw.text(f"Num:
{myNum}",center=(200,100),color=(255,0,0), fontsize = 30)
```

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 7:

☐ Your game shows a basic game over screen when it should

★ BONUS 6.4: Funky Screens!

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

Now that you have your basic game over screen you can change some things and add images or whatever you want to make the coolest game over screen.

For some more text formatting go to this link:
https://pygame-zero.readthedocs.io/en/stable/ptext.html

For some other screen design code go to this link:
https://pygame-zero.readthedocs.io/en/stable/builtins.html#screen

7. Extension: More pipes!

Right now the game is a bit repetitive because we are only looping three sets of pipes so why don't we change it a little so we can add more

Task 7.1: Making some more pipes

Below where you are making your first sets of three pipes let's make some more.

1. The **fourth top pipe** should have an **x value** of 1064 and a **y value** of -70
2. The **fourth bottom pipe** should have **x value** of 1064 and a **y value** of 790
3. The **fifth top pipe** should have **x value** of 1330 and a **y value** of -190
4. The **fifth bottom pipe** should have **x value** of 1330 and a **y value** of 610
5. The **sixth top pipe** should have **x value** of 1596 and a **y value** of -50
6. The **sixth bottom pipe** should have **x value** of 1596 and a **y value** of 750

Task 7.2: Fixing the rest of the code

Now to fix the rest of the code

1. Under where you're drawing your six other pipes make sure you draw these new pipes
2. Under where you're checking to see if the bird hits your other pipes make sure you check the same for these new pipes
3. Under where you're updating each of your previous pipes make sure you do the same for these new pipes.
4. Every time you wrap a pipe around to the other side of the page, instead of resetting the pipes' x value to the width of the screen, reset the pipes' x value to 1552 so the new pipes don't overlap with the old ones.

☑ CHECKPOINT ☑

If you can tick all of these off you can check out some of the other extensions and bonuses

☐ You have a working game that allows the bird to move, hit the pipes and increase the score

☐ Your game has 6 sets of pipes at different heights

★ BONUS 7.3: Interesting pipes

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

Now you have the basis of the game plus some extra pipes you should play around with some of the values we've given you like the speed the pipes and the bird moves, or maybe the initial coordinates of things. See if you make the game more fun or difficult. You could also add more pipes if you want. If you add more pipes though you need to remember to reset their x values to (266 * the number of pipes) - 44