Girls' Programming Network

Flappy Bird!

FOR TUTOR EYES ONLY

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!!

Part 0: Setting up

Task 0.1: Making a python file

Open the start menu, and type 'IDLE'. Select IDLE 3.5.

1. Go to the file menu and select 'new file'. This opens a new window.
2. Go to the file menu, select 'Save as'
3. Go to the Desktop and save the file as 'flappy_bird.py'

TUTOR TIPS

- Make sure they saved it as .py or they won't get syntax highlighting
- Suggest they save it somewhere they can find easily, e.g. Desktop

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

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

```
# This is a comment
```

☑ 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

☐ Run your file with F5 key and it does nothing!!

TUTOR TIPS

The code should look like this (no bonuses):

```
# <the student's name>
```

Part 1: Making a game

Task 1.1: Print a welcome and the rules

Welcome the player and print the rules!

Use a print to make it happen when you run your code:

```
The game is about to start!
You can move the bird up and down using the arrow keys
Dodge the pipes to win
Good Luck!
```

TUTOR TIPS

You may need to remind them to use the F5 key to run the program

Hint

Want to print multiple lines at a time? You can use three sets of quotes instead of one, to make your strings go over multiple lines

```
print("""
Print
Three
Lines
""")
```

Task 1.2: Get Pygame

We need to import Pygame and start the game!

At the start of your code import and start Pygame

Hint

To import and start Pygame we can use this code:

```
from pygame import *
init()
```

Task 1.3: Make a screen

We want to make a screen to show our game!

Make a screen that is 800 pixels wide and 600 pixels tall

Hint

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

```
screen = display.set_mode((100, 200))
```

☑ CHECKPOINT ☑

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

☐ Print a welcome

☐ Print the rules

☐ Shows a dark screen that is the same size as the background image

☐ Try running your code!

TUTOR TIPS

Students may use multiple print statements or multi-line strings

The code should look like this (no bonuses):

```
# <the student's name>
from pygame import *
init()

print("""
The game is about to start!
You can move the bird up and down using the arrow keys
Dodge the pipes to win
Good Luck!
""")

screen = display.set_mode((800, 600))
```

Part 2: Images!

Task 2.1: Save all the images!

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

First we need to download the images onto our computer. Go to girlsprogramming.network/flappy-bird-images and download all the images there.

Make sure you save all the images to the same place that you've saved your program!

Task 2.2: Load 'em up!

Now we need to load all the images we've just saved!

You should have a background image, a bird image and a pipe image.

Hint

To load an image called "cat.png" you could use code like this

```
cat_image = image.load("cat.png")
```

Task 2.3: Show that background!

The black box we have right now is pretty boring! Let's make our game show the background image!

Blit the background image to the screen and then update the screen to make the image show up.

Hint

Have a look at your cheat sheet and if you've forgotten how to blit and update the screen.

Also, remember that we want our background to line up to the top left of the screen so we want to use the coordinates 0 and 0

☑ 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!

The code should look like this:

```python
from pygame import *
init()



print("The game is about to start!")


screen = display.set_mode((800, 600))

#added all the images
background_image = image.load("bg.png")
bird_image = image.load("bird.png")
pipe_image = image.load("pipe.png")

#Blit and display
screen.blit(background_image, (0, 0))
display.update()
```

★ **BONUS 2.4: Funky Backgrounds!!**

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

We can use any image as a background! Go on the internet and find another image to use as a background! Save it in like we did with our flappy bird images, load it and use it instead of our regular background image

Note: If your new background image is a different size you might need to change the size of the image to fit the screen.

Part 3: Birds and Pipes

Task 3.1: Where's that bird?

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

First we want to save the coordinates of our bird into variables so that we can change them later. Make two variables at the start of your program called bird_x and bird_y and make them equal 10 and 250

Task 3.2: Show me the bird!

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

After we blit the background to the screen but before we update the display add a line of code to blit the bird to the screen.

Hint

Remember that the computer puts things on the screen in the order that you blit them, so if you put the bird on first, it will end up behind the background and we won't see it!

Task 3.3: What about the pipes?

Now we have the bird but no obstacles for them to flap around!

Do the same thing you did in tasks 3.1 and 3.2 but for the pipes instead. For the pipe_x and pipe_y variables use the numbers 250 and 250

Task 3.4: Just one pipe?

Just having one pipe isn't very interesting. Do the same thing you did in task 3.3 and make a pipe2 and a pipe3. For their coordinates use these numbers:
Pipe2 = 500, 100
Pipe3 = 750, 400

☑ CHECKPOINT ☑

If you can tick all of these off you can go to Part 4:
☐ There is a background still on the screen

☐ There is a bird on the left side of the screen

☐ There are 3 pipes on the screen

☐ Run your code!

★ **BONUS 3.4: Funky Bird!**

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

We can use any image as a bird! Go on the internet and find another image to use as a bird! Save it in like we did with our flappy bird images, load it and use it instead of our regular bird image

Note: make sure your new bird image isn't too big or you won't be able to get around the pipes!

Part 4: Make things happen

Events & If
Statements

Task 4.1: Make a game loop

Let's make a while loop to run our game inside!

Keep all of our image loading and coordinate variables *outside* the loop, and put all of the blit and display update code *inside* the loop

Hint

To make a while loop that runs forever it should look like this:

```python
# Stuff out of the loop is here
print("I only happen once")

while True:
    # Stuff inside the loop is here
    print("I print over and over and over again!")
```

Task 4.2: Events!

To make our game do things, we need to check if an event has happened!

Let's make a variable called new_event and store any event that has happened there

Hint

To store an event that has happened we use code that looks like this:

```python
new_event = event.poll()
```

Task 4.3: Now what?

Now that we have our new event let's check and see if it's someone pressing down the up arrow!

Make an if statement that checks if the event is a KEYDOWN and if the key being pressed is the up arrow key. If it is, let's print out "Going up!"

Hint

To check if someone has pressed the left arrow I can use code like this:

```python
if new_event.type == KEYDOWN and new_event.key == K_LEFT:
    print("Pressed Left")
```

Task 4.3: Move that bird!

When they press the up key we want the bird to move up, so let's update the bird's y coordinate when the up key is pressed.

Inside the if statement you just wrote, after the print, update the bird_y variable to be itself minus 50

Hint

Remember that to make something go up we make the number smaller!

Task 4.3: Going down!

Now our bird goes up when we press the up key! Let's do the same thing but make the bird go down when we press the down key.

You can copy and paste the if statement you just made and change the key to down, the print statement and make sure it adds 50 instead of minusing

Hint

Make sure your if statements end up lined up like this:

```python
if name == "Alex":
    print("I like your name")
if name = "Renee":
    print("Your name is great!")
```

Task 4.4: Going left!

Now our bird goes up and down when we press the correct key! Let's do the same thing but make the bird go left when you press the left key

When you're moving up and down you need to change the bird_y variable, but when you're moving left and right you need to change the bird_x variable!

Task 4.5: Going Right!

Now our bird goes up, down and left when we press the correct key! Let's do the same thing but make the bird go right when you press the right key

☑ CHECKPOINT ☑

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

☐ It prints "Going up" when you press the up key

☐ The bird moves up when you press the up key

☐ It prints "Going down" when you press the down key

☐ The bird moves down when you press the down key

☐ It prints "Going left" when you press the left key

☐ The bird moves left when you press the left key

☐ It prints "Going right" when you press the right key

☐ The bird moves right when you press the right key

TUTOR TIPS

The code should look like this:

```
from pygame import *
init()

print("The game is about to start!")

screen = display.set_mode((800, 600))

background_image = image.load("bg.png")
bird_image = image.load("bird.png")
pipe_image = image.load("pipe.png")
```

```
bird_x = 10
bird_y = 250

pipe_x = 200
pipe_y = 250
pipe2_x = 450
pipe2_y = 100
pipe3_x = 700
pipe3_y = 400

#Added the game loop
while True:

    #Polling for the latest event
    new_event = event.poll()
    if new_event.type == KEYDOWN and new_event.key == K_UP:
        print("Going up")
        bird_y = bird_y - 50
    if new_event.type == KEYDOWN and new_event.key == K_DOWN:
        print("Going down")
        bird_y = bird_y + 50
    if new_event.type == KEYDOWN and new_event.key == K_LEFT:
        print("Going left")
        bird_x = bird_x - 50
    if new_event.type == KEYDOWN and new_event.key == K_RIGHT:
        print("Going right")
        bird_x = bird_x + 50

    background = screen.blit(background_image, (0, 0))
    bird = screen.blit(bird_image, (bird_x, bird_y))
    pipe = screen.blit(pipe_image, (pipe_x, pipe_y))
    pipe2 = screen.blit(pipe_image, (pipe2_x, pipe2_y))
    pipe3 = screen.blit(pipe_image, (pipe3_x, pipe3_y))
    display.update()
```

★ **BONUS 4.4: Stop that game!**

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

At the moment we can never quit the game. Let's add another if statement that will let us quit the game if we press the Q key

Part 5: Win or lose!

Task 5.1: Bang!

Right now we can run right into the pipe and nothing bad happens! Let's change that so that when you hit a pipe you lose the game

Add an if statement after we update the display that checks if the bird has collided with the first pipe and print out "Game over!" if it does. Then repeat for the two other pipes.

Hint

To check if two things have collided we can do it like this:

```
if thing1.colliderect(thing2):
    print("BANG!")
```

Task 5.2: Stop the game!

When we hit the pipe we want the game to stop!

After the print statement `quit()` the game and `break` the loop

Task 5.3: Is that the edge?

We want to announce that the player has won when they reach the other edge of the screen!

Write an if statement that checks whether the bird has gone off the other edge of the screen and print out "Winner!" if they do, then `quit()` the game and `break` the loop

TUTOR TIPS

It might seem counterintuitive to quit the game first and then break. The reason we do this is so that we don't need to quit the game outside of the loop. If we break first then the get thrown out of the loop and would need to quit pygame outside the loop (which is a little confusing)
If we quit and don't break then the loop continues and pygame will throw errors when it tries to load images because there is no game anymore

☑ CHECKPOINT ☑

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

☐ When your bird hits a pipe the game ends and it prints "Game Over!"

☐ You win the game when you go off the rightmost edge

TUTOR TIPS

The code should look like this:

```python
from pygame import *
init()

print("The game is about to start!")

screen = display.set_mode((800, 600))

background_image = image.load("bg.png")
bird_image = image.load("bird.png")
pipe_image = image.load("pipe.png")

bird_x = 10
bird_y = 250


pipe_x = 200
pipe_y = 250
pipe2_x = 450
pipe2_y = 100
pipe3_x = 700
pipe3_y = 400
```

```python
while True:
    new_event = event.poll()
    if new_event.type == KEYDOWN and new_event.key == K_UP:
        print("Going up")
        bird_y = bird_y - 50
    if new_event.type == KEYDOWN and new_event.key == K_DOWN:
        print("Going down")
        bird_y = bird_y + 50
    if new_event.type == KEYDOWN and new_event.key == K_LEFT:
        print("Going left")
        bird_x = bird_x - 50
    if new_event.type == KEYDOWN and new_event.key == K_RIGHT:
        print("Going right")
        bird_x = bird_x + 50

    background = screen.blit(background_image, (0, 0))
    bird = screen.blit(bird_image, (bird_x, bird_y))
    pipe = screen.blit(pipe_image, (pipe_x, pipe_y))
    pipe2 = screen.blit(pipe_image, (pipe2_x, pipe2_y))
    pipe3 = screen.blit(pipe_image, (pipe3_x, pipe3_y))
    display.update()

    #Added if statements to detect collisions
    if bird.colliderect(pipe):
        print("Game Over!")
        quit()
        break

    if bird.colliderect(pipe2):
        print("Game Over!")
        quit()
        break

    if bird.colliderect(pipe3):
        print("Game Over!")
        quit()
        break

    #Added if statement to detect when the bird goes over
    if bird_x >= 800:
        print("Winner!")
        quit()
        break
```

Part 6: Flip it!

In the real Flappy Bird some of the pipes hung down from the top. Let's update our game so that ours does that too!

Task 6.1: Turn it upside down

First we need to make an upside down version of our pipe_image.

Use the pygame transform to flip our image upside down and save it as flipped_pipe_image

Hint

To flip an image upside down we can use code like this:

```
bat_image = image.load("bat.png")
hanging_bat_image = transform.flip(bat_image, False, True)
```

Task 6.2: Pick a pipe

We're going to make pipe3 be the one that is upside down. Where we blit the pipe_image for pipe3, use the flipped_pipe_image instead

Task 6.3: Too low!

With the pipe coordinates the way they are it looks a bit silly because it's still poking up from the floor instead of hanging down from the ceiling. We can fix this by using -pipe3_y (notice the minus sign!) when we blit the image to the screen!

☑ CHECKPOINT ☑

If you can tick all of these off you can do the extensions:
☐ The last pipe is flipped upside down and hangs from the ceiling

TUTOR TIPS

The code should look like this:
```
from pygame import *
init()
```

```python
print("The game is about to start!")

screen = display.set_mode((800, 600))

background_image = image.load("bg.png")
bird_image = image.load("bird.png")
pipe_image = image.load("pipe.png")
# Added a flipped pipe image
flipped_pipe_image = transform.flip(pipe_image, False, True)

bird_x = 10
bird_y = 250

pipe_x = 200
pipe_y = 250
pipe2_x = 450
pipe2_y = 100
pipe3_x = 700
pipe3_y = 400

while True:
    new_event = event.poll()
    if new_event.type == KEYDOWN and new_event.key == K_UP:
        print("Going up")
        bird_y = bird_y - 50
    if new_event.type == KEYDOWN and new_event.key == K_DOWN:
        print("Going down")
        bird_y = bird_y + 50
    if new_event.type == KEYDOWN and new_event.key == K_LEFT:
        print("Going left")
        bird_x = bird_x - 50
    if new_event.type == KEYDOWN and new_event.key == K_RIGHT:
        print("Going right")
        bird_x = bird_x + 50

    background = screen.blit(background_image, (0, 0))
    bird = screen.blit(bird_image, (bird_x, bird_y))
    pipe = screen.blit(pipe_image, (pipe_x, pipe_y))
    pipe2 = screen.blit(pipe_image, (pipe2_x, pipe2_y))
    #Update pipe3 to use the flipped image and the negative y coordinate
    pipe3 = screen.blit(flipped_pipe_image, (pipe3_x, -pipe3_y))
    display.update()

    if bird.colliderect(pipe):
        print("Game Over!")
        quit()
        break

    if bird.colliderect(pipe2):
        print("Game Over!")
```

```
        quit()
        break

    if bird.colliderect(pipe3):
        print("Game Over!")
        quit()
        break

    if bird_x >= 800:
        print("Winner!")
        quit()
        break
```