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 1: Keep it Classy

Our code works but it's not as nice as it could be - let's class it up a bit!

Task 1.1: Making a Pipe class

Our pipes have a lot of variables each (x, y, flipped, etc.) and they each do a lot of things (blit, move, collide with the bird) so they are a great example of how classes can clean up our code!

First we are going to make our Pipe class at the very top of our code, after our imports. Make a Pipe class and write the __init__ function so that we set the x, y and flipped variables for our Pipe

Hint

Remember that this is what a class looks like in python:

```python
class Pet():
    def __init__(self, name, breed, colour, age):
        self.name = name
        self.breed = breed
        self.colour = colour
        self.age = age
```

Task 1.2: Making some pipes

Now that we have our new Pipe class we can use it to make our pipes for us!

Where we have all our pipe_x, pipe_y and pipe_flipped variables, replace them with our new Pipe class and name them pipe1_object, pipe2_object and pipe3_object. Then everywhere we are using pipe_x, pipe_y and pipe_flipped replace it with pipe1_object.x, pipe1_object.y and pipe1_object.flipped

Hint

To use a class we can use code like this:

```python
pet = Pet("Luna", "dog", "brown", "7")

print("I have a", pet.breed, "named", pet.name)
```

☑ CHECKPOINT ☑

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

☐ You have a Pipe class that stores the x, y and flipped variables

☐ You have made 3 pipes using the new Pipe class

☐ Run your code!

TUTOR TIPS

The code should look like this:

```python
from pygame import *
from random import *

#Made a new pipe class
class Pipe ():
    def __init__(self, x, y, flipped):
        self.x = x
        self.y = y
        self.flipped = flipped

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")
flipped_pipe_image = transform.flip(pipe_image, False, True)

bird_x = 10
bird_y = 250

#Converted variables into the class
pipe1_object = Pipe(200, 250, False)
pipe2_object = Pipe(450, 100, False)
pipe3_object = Pipe(700, 400, True)

while True:

    new_event = event.poll()
    if new_event.type == KEYDOWN and new_event.key == K_UP:
        bird_y = bird_y - 50
    if new_event.type == KEYDOWN and new_event.key == K_DOWN:
        bird_y = bird_y + 50

    #Updated to use the class
    pipe1_object.x = pipe1_object.x - 2
```

```python
    pipe2_object.x = pipe2_object.x - 2
    pipe3_object.x = pipe3_object.x - 2

    background = screen.blit(background_image, (0, 0))
    bird = screen.blit(bird_image, (bird_x, bird_y))

    #Updated to use the class
    if pipe1_object.flipped:
        pipe = screen.blit(flipped_pipe_image, (pipe1_object.x,
-pipe1_object.y))
    else:
        pipe = screen.blit(pipe_image, (pipe1_object.x, pipe1_object.y))
    if pipe2_object.flipped:
        pipe2 = screen.blit(flipped_pipe_image, (pipe2_object.x,
-pipe2_object.y))
    else:
        pipe2 = screen.blit(pipe_image, (pipe2_object.x, pipe2_object.y))
    if pipe3_object.flipped:
        pipe3 = screen.blit(flipped_pipe_image, (pipe3_object.x,
-pipe3_object.y))
    else:
        pipe3 = screen.blit(pipe_image, (pipe3_object.x, pipe3_object.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

    #Updated to use the class
    if pipe1_object.x < -87:
        pipe1_object.x = 800
        pipe1_object.y = randint(100, 500)
        pipe1_object.flipped = choice([True, False])

    if pipe2_object.x < -87:
        pipe2_object.x = 800
        pipe2_object.y = randint(100, 500)
        pipe2_object.flipped = choice([True, False])

    if pipe3_object.x < -87:
        pipe3_object.x = 800
        pipe3_object.y = randint(100, 500)
        pipe3_object.flipped = choice([True, False])
```

Part 2: Blit yourself!

Now that we have a Pipe class, we can make it blit itself and work out whether it should use the flipped image or not.

Task 2.1: Move the images

We are going to need to use the images inside our class when we blit them to the screen. Move the pygame init() and the image loading code to be above where our class is defined.

Task 2.2: Blit function

Inside our Pipe class make a blit function and copy the if and else statements that current control what image to blit into the new blit function.

Update the function so that it's using self instead of self.x, self.y and self.flipped instead of pipe.x, pipe.y and pipe_flipped. Instead of saving the blit to a variable called pipe, save it to self.rect

Task 2.3: Go and blit!

Now where we blit the background and the pipes replace all the if statements for blitting the pipes with pipe1.blit() etc.

Also replace in the collision ifs to use pipe.rect

☑ CHECKPOINT ☑

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

☐ When a pipe goes over the left edge it comes back on the right edge

☐ When the pipe reappears on the right edge it's a random height

☐ Try running your code!

TUTOR TIPS

The code should look like this:

```python
from pygame import *
from random import *

#Moved init and images above the class
init()

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

class Pipe ():
    def __init__(self, x, y, flipped):
        self.x = x
        self.y = y
        self.flipped = flipped

    #Added blit function
    def blit(self):
        if self.flipped:
            self.rect = screen.blit(flipped_pipe_image, (self.x, -self.y))
        else:
            self.rect = screen.blit(pipe_image, (self.x, self.y))


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

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

bird_x = 10
bird_y = 250

pipe1_object = Pipe(200, 250, False)
pipe2_object = Pipe(450, 100, False)
pipe3_object = Pipe(700, 400, True)

while True:

    new_event = event.poll()
    if new_event.type == KEYDOWN and new_event.key == K_UP:
        bird_y = bird_y - 50
    if new_event.type == KEYDOWN and new_event.key == K_DOWN:
        bird_y = bird_y + 50

    pipe1_object.x = pipe1_object.x - 2
    pipe2_object.x = pipe2_object.x - 2
    pipe3_object.x = pipe3_object.x - 2

    background = screen.blit(background_image, (0, 0))
    bird = screen.blit(bird_image, (bird_x, bird_y))

    #Use the blit function instead
```

```python
pipe1_object.blit()
pipe2_object.blit()
pipe3_object.blit()

display.update()

#Update to use class
if bird.colliderect(pipe1_object.rect):
    print("Game Over!")
    quit()
    break

#Update to use class
if bird.colliderect(pipe2_object.rect):
    print("Game Over!")
    quit()
    break

#Update to use class
if bird.colliderect(pipe3_object.rect):
    print("Game Over!")
    quit()
    break

if pipe1_object.x < -87:
    pipe1_object.x = 800
    pipe1_object.y = randint(100, 500)
    pipe1_object.flipped = choice([True, False])

if pipe2_object.x < -87:
    pipe2_object.x = 800
    pipe2_object.y = randint(100, 500)
    pipe2_object.flipped = choice([True, False])

if pipe3_object.x < -87:
    pipe3_object.x = 800
    pipe3_object.y = randint(100, 500)
    pipe3_object.flipped = choice([True, False])
```

Part 3: Do it all!

Our Pipe class can do it all if we tell it how! Now we're going to tell it how to move the Pipe and how to check for collisions

Task 3.1: Get moving

In our Pipe class, make a function called move. This is where we're going to get the pipe to move itself.

Inside the new move function minus one from the pipe's x variable. Then move the if statement from the main game that decides whether the pipe has gone over the edge into this function and update it to use self.x, self.y and self.flipped

Task 3.2: I like to move it move it

We can now replace our old pipe moving code and replace it with our new pipe.move() function. We can also delete all the if statements that move the pipes back to the start from our code and clean it up.

Return & Arguments

Task 3.3: Collisions

Make a function in our Pipe class called collides_with that has one argument called bird.

Inside this function we will return whether or not the bird has collided with our self.rect

Now we can replace the `bird.colliderect` code with `pipe.collides_with(bird)` for all of our pipes

Hint

To use make a function that has an argument and returns something we can use this code:

```python
class Pet():
    def __init__(self, name, breed, colour, age):
        self.name = name
        self.breed = breed
        self.colour = colour
        self.age = age

    def sit(self, used_name):
        if used_name == self.name:
```

```
            return "Sitting"
        else:
            return "That's not my name"
```

☑ CHECKPOINT ☑

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

☐ Our Pipe class now has move and blit functions

☐ We're using the move and blit functions instead of the old code

☐ Run your code!

```python
            self.flipped = choice([True, False])

    #Added collides_with function
    def collides_with(self, bird):
        return bird.colliderect(self.rect)


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

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

bird_x = 10
bird_y = 250

pipe1_object = Pipe(200, 250, False)
pipe2_object = Pipe(450, 100, False)
pipe3_object = Pipe(700, 400, True)

while True:
    new_event = event.poll()
    if new_event.type == KEYDOWN and new_event.key == K_UP:
        bird_y = bird_y - 50
    if new_event.type == KEYDOWN and new_event.key == K_DOWN:
        bird_y = bird_y + 50

    #Use the move function instead
    pipe1_object.move()
    pipe2_object.move()
    pipe3_object.move()

    background = screen.blit(background_image, (0, 0))
    bird = screen.blit(bird_image, (bird_x, bird_y))

    pipe1_object.blit()
    pipe2_object.blit()
    pipe3_object.blit()

    display.update()

    #Use the collides_with function here
    if pipe1_object.collides_with(bird):
        print("Game Over!")
        quit()
        break

    if pipe2_object.collides_with(bird):
        print("Game Over!")
        quit()
        break

    if pipe3_object.collides_with(bird):
        print("Game Over!")
        quit()
```

```
        break

    #Deleted the ifs here because the move function does it now
```

Part 4: A list of Pipes

Our code looks great now! But we can make it even better by making a list to keep track of our pipes

Task 4.1: Listmaker

After we make all our pipes before the while loop let's put them in a list called pipes.

Hint

You can make a list like this:

```
pets = ["Luna", "Emmy", "Saphira"]
```

Task 4.2: Move each pipe

Where we have our code that moves the pipes we can replace it with a for loop that loops through each pipe and tells it to move.

Hint

To loop through a list we can use code like this:

```
for pet in pets:
    print("I love", pet)
```

Task 4.3: Blit each pipe

Where we blit each pipe let's replace it with a for loop that loops through each pipe and tells it to blit, just like we just did with moving them

Task 4.5: Check each pipe

Make a variable before the while loop called over and set it to False - we're going to use this to make sure we end the game properly when the game should be over.

Where we check each pipe for collisions, let's replace it with a for loop that loops through each pipe and checks if it's collided with the bird. If it has do the usual "game over" code and also set over to True

Then write another if statement that checks if over is true, if it is then `break` again

☑ CHECKPOINT ☑

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

☐ We have a list of all 3 of our pipes

☐ We're using for loops to move, blit and check for collisions

☐ Run your code!

TUTOR TIPS

The code should look like this:

```python
from pygame import *
from random import *

init()

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

class Pipe ():

    def __init__(self, x, y, flipped):
        self.x = x
        self.y = y
        self.flipped = flipped

    def blit(self):
        if self.flipped:
            self.rect = screen.blit(flipped_pipe_image, (self.x, -self.y))
        else:
            self.rect = screen.blit(pipe_image, (self.x, self.y))

    def move(self):
        self.x = self.x - 2
        if self.x < -87:
            self.x = 800
            self.y = randint(100, 500)
            self.flipped = choice([True, False])
```

```python
    def collides_with(self, bird):
        return bird.colliderect(self.rect)


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

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

bird_x = 10
bird_y = 250

pipe1_object = Pipe(200, 250, False)
pipe2_object = Pipe(450, 100, False)
pipe3_object = Pipe(700, 400, True)

#Added a list of the pipes
pipes = [pipe1_object, pipe2_object, pipe3_object]

#Added over variable
over = False

while True:

    new_event = event.poll()
    if new_event.type == KEYDOWN and new_event.key == K_UP:
        bird_y = bird_y - 50
    if new_event.type == KEYDOWN and new_event.key == K_DOWN:
        bird_y = bird_y + 50

    #Using a loop for this now
    for pipe in pipes:
        pipe.move()

    background = screen.blit(background_image, (0, 0))
    bird = screen.blit(bird_image, (bird_x, bird_y))

    #Using a loop for this now
    for pipe in pipes:
        pipe.blit()

    display.update()

    #Using a loop for this now
    for pipe in pipes:
        if pipe.collides_with(bird):
            print("Game Over!")
            #Don't forget to set over to true
            over = True
            quit()
            break

    #When the game is over we need to break the big while loop
```

```
if over:
    break
```

Part 5: All that glitters is coins

Now that we know all about classes we can make a class for coins that give us points!

Task 5.1: Make a coin class

After our Pipe class, make another class called Coin. It only needs to keep track of an x and y coordinate.

Then where you make all your pipes, also make a Coin at coordinates 400 and 150

Task 5.3: Show me the coin

Download the coin image and load it into pygame as coin_image. Then write a blit function inside the Coin class to blit the coin_image to the screen.

After we blit all the pipes, call the Coin blit function to make the coin show up on the screen

Hint

Remember to save your images in the same place you save your program

Task 5.4: Keep it moving

The coin should move along the screen with the pipes. It should also move back to the far right if it goes off the screen and pick a random y coordinate just like the pipes. Make a move function that does this and then after the code that moves all the pipes also move the coin

Task 5.5: Get the points

To keep track of our points make a new points variable before the while loop and set it to 0.

Make a collected_by function that takes one argument called bird. Make it check if the bird has collided with the coin. If it has then move the coin to the far right and randomly pick a new y coordinate and return True. If it hasn't then return False

After we have checked for all the pipe collisions, check if the coin has been collected by the bird. If it has then add one to our points.

When the game ends, print out how many points we got.

CHECKPOINT

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

☐ There is now a coin in your game that moves with the pipes

☐ If you hit the coin you get a point and the coin moves back to the right edge and has a random y coordinate

☐ If you don't hit the coin it moves back to the right edge and has a random y coordinate

☐ At the end of the game it prints out how many points you got

☐ Run your code!

TUTOR TIPS

The code should look like this:

```python
from pygame import *
from random import *


init()

background_image = image.load("bg.png")
bird_image = image.load("bird.png")
pipe_image = image.load("pipe.png")
flipped_pipe_image = transform.flip(pipe_image, False, True)
coin_image = image.load("coin.png")

class Pipe ():

    def __init__(self, x, y, flipped):
        self.x = x
        self.y = y
        self.flipped = flipped

    def blit(self):
        if self.flipped:
            self.rect = screen.blit(flipped_pipe_image, (self.x, -self.y))
        else:
```

```python
            self.rect = screen.blit(pipe_image, (self.x, self.y))

    def move(self):
        self.x = self.x - 2
        if self.x < -87:
            self.x = 800
            self.y = randint(100, 500)
            self.flipped = choice([True, False])

    def collides_with(self, bird):
        return bird.colliderect(self.rect)


#Created the Coin class
class Coin ():

    def __init__(self, x, y):
        self.x = x
        self.y = y

    def blit(self):
        self.rect = screen.blit(coin_image, (self.x, self.y))

    def move(self):
        self.x = self.x - 2
        if self.x < -70:
            self.x = 800
            self.y = randint(100, 500)

    def collected_by(self, bird):
        if bird.colliderect(self.rect):
            self.x = 800
            self.y = randint(100, 500)
            return True
        return False


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")


bird_x = 10
bird_y = 250

pipe1_object = Pipe(200, 250, False)
pipe2_object = Pipe(450, 100, False)
pipe3_object = Pipe(700, 400, True)
```

```python
pipes = [pipe1_object, pipe2_object, pipe3_object]

#Made a coin
coin = Coin(400, 150)

#Made a points variable
points = 0

over = False

while True:

    new_event = event.poll()
    if new_event.type == KEYDOWN and new_event.key == K_UP:
        bird_y = bird_y - 50
    if new_event.type == KEYDOWN and new_event.key == K_DOWN:
        bird_y = bird_y + 50

    for pipe in pipes:
        pipe.move()

    #Move the coin
    coin.move()


    background = screen.blit(background_image, (0, 0))
    bird = screen.blit(bird_image, (bird_x, bird_y))

    for pipe in pipes:
        pipe.blit()

    #Blit the coin
    coin.blit()

    display.update()

    for pipe in pipes:
        if pipe.collides_with(bird):
            print("Game Over!")
            quit()
            over = True
            break

    #Check if the coin is collected and update the points
    if coin.collected_by(bird):
        points = points + 1

    if over:
        break

#Print out our point total at the end
print("Points:", points)
```