Flappy Bird – The Game development using python

The bird is the 'hero' of this game. The player can make it climb (ascend quickly), otherwise it sinks (descends more slowly). It must pass through the space in between pipes (for every pipe passed, one point is scored); if it crashes into a pipe, the game ends.

We can start by importing the required libraries,

```
import math
import os
from random import randint
from collections import deque

import pygame
from pygame.locals import *


FPS = 60
ANIMATION_SPEED = 0.18  # pixels per millisecond
WIN_WIDTH = 284 * 2     # BG image size: 284x512 px; tiled twice
WIN_HEIGHT = 512
```

Represents the bird controlled by the player.

The bird is the 'hero' of this game. The player can make it climb (ascend quickly), otherwise it sinks (descends more slowly). It must pass through the space in between pipes (for every pipe passed, one point is scored); if it crashes into a pipe, the game ends.

Attributes:

x: The bird's X coordinate.

y: The bird's Y coordinate. msec_to_climb: The number of milliseconds left to climb, where a complete climb lasts Bird.CLIMB_DURATION milliseconds.

Constants:

WIDTH: The width, in pixels, of the bird's image.

HEIGHT: The height, in pixels, of the bird's image.

SINK_SPEED: With which speed, in pixels per millisecond, the bird descends in one second while not climbing.

CLIMB_SPEED: With which speed, in pixels per millisecond, the bird ascends in one second while climbing, on average. See also the Bird.update docstring.

CLIMB_DURATION: The number of milliseconds it takes the bird to execute a complete climb.

```
    super(Bird, self).__init__()
    self.x, self.y = x, y
    self.msec_to_climb = msec_to_climb
    self._img_wingup, self._img_wingdown = images
    self._mask_wingup = pygame.mask.from_surface(self._img_wingup)
    self._mask_wingdown = pygame.mask.from_surface(self._img_wingdown)
```

Initialise a new Bird instance.

Arguments:

x: The bird's initial X coordinate.

y: The bird's initial Y coordinate.

msec_to_climb: The number of milliseconds left to climb, where complete climb lasts Bird.CLIMB_DURATION milliseconds. Use this if you want the bird to make a (small?) climb at the very beginning of the game.

images: A tuple containing the images used by this bird. It must contain the following images, in the following order:

0. image of the bird with its wing pointing upward

1. image of the bird with its wing pointing downward

```
"""
if self.msec_to_climb > 0:
    frac_climb_done = 1 - self.msec_to_climb/Bird.CLIMB_DURATION
    self.y -= (Bird.CLIMB_SPEED * frames_to_msec(delta_frames) *
               (1 - math.cos(frac_climb_done * math.pi)))
    self.msec_to_climb -= frames_to_msec(delta_frames)
else:
    self.y += Bird.SINK_SPEED * frames_to_msec(delta_frames)
```

Update the bird's position

This function uses the cosine function to achieve a smooth climb:

In the first and last few frames, the bird climbs very little, in the middle of the climb, it climbs a lot.

One complete climb lasts CLIMB_DURATION milliseconds, during which the bird ascends with an average speed of CLIMB_SPEED px/ms.This Bird's msec_to_climb attribute will automatically be

decreased accordingly if it was > 0 when this method was called.

Arguments:

delta_frames: The number of frames elapsed since this method was last called.

```
if pygame.time.get_ticks() % 500 >= 250:
    return self._img_wingup
else:
    return self._img_wingdown
```

Get a Surface containing this bird's image.

This will decide whether to return an image where the bird's visible wing is pointing upward or where it is pointing downward based on pygame.time.get_ticks(). This will animate the flapping bird, even though pygame doesn't support animated GIFs.

```
WIDTH = 80
PIECE_HEIGHT = 32
ADD_INTERVAL = 3000
```

A PipePair has a top and a bottom pipe, and only between them can the bird pass -- if it collides with either part, the game is over.

Attributes:

x: The PipePair's X position. This is a float, to make movement smoother. Note that there is no y attribute, as it will only ever be 0.

image: A pygame.Surface which can be blitted to the display surface to display the PipePair.

mask: A bitmask which excludes all pixels in self.image with a transparency greater than 127. This can be used for collision detection.

top_pieces: The number of pieces, including the end piece, in the top pipe.

bottom_pieces: The number of pieces, including the end piece, in the bottom pipe.

Constants:

WIDTH: The width, in pixels, of a pipe piece. Because a pipe is only one piece wide, this is also the width of a PipePair's image.

PIECE_HEIGHT: The height, in pixels, of a pipe piece.

ADD_INTERVAL: The interval, in milliseconds, in between adding new pipes.

```
def __init__(self, pipe_end_img, pipe_body_img):
```

Initialises a new random PipePair.

The new PipePair will automatically be assigned an x attribute of float(WIN_WIDTH - 1).

Arguments:

pipe_end_img: The image to use to represent a pipe's end piece.

pipe_body_img: The image to use to represent one horizontal slice of a pipe's body.

```
    """
    file_name = os.path.join('.', 'images', img_file_name)
    img = pygame.image.load(file_name)
    img.convert()
    return img

return {'background': load_image('background.png'),
        'pipe-end': load_image('pipe_end.png'),
        'pipe-body': load_image('pipe_body.png'),
        # images for animating the flapping bird -- animated GIFs are
        # not supported in pygame
        'bird-wingup': load_image('bird_wing_up.png'),
        'bird-wingdown': load_image('bird_wing_down.png')}
```

Return the loaded pygame image with the specified file name.

This function looks for images in the game's images folder(./images/). All images are converted before being returned to speed up blitting.

Arguments:

img_file_name: The file name (including its extension, e.g.('.png') of the required image, without a file path.

```
pygame.init()

display_surface = pygame.display.set_mode((WIN_WIDTH, WIN_HEIGHT))
pygame.display.set_caption('Pygame Flappy Bird')

clock = pygame.time.Clock()
score_font = pygame.font.SysFont(None, 32, bold=True)  # default font
images = load_images()
```

The application's entry point.

If someone executes this module (instead of importing it, for example), this function is called.

```
bird = Bird(50, int(WIN_HEIGHT/2 - Bird.HEIGHT/2), 2,
            (images['bird-wingup'], images['bird-wingdown']))
```

the bird stays in the same x position, so bird.x is a constant,center bird on screen.

```
if __name__ == '__main__':
```

If this module had been imported, __name__ would be 'flappybird'.

It was executed (e.g. by double-clicking the file), so call main.

Thank you,