Aquarium Pipe Game: Part 1

Aims

- To gain experience determining how OOP can be used to solve a problem.
- To practice using classes, objects, properties, attributes and methods.
- To gain further experience with encapsulation, inheritance and polymorphism.
- To practice debugging code with an IDE.

Download Skeleton Code

Fork the exercise repository to create a copy of the code for you to work on. To do this, first open the following link: `https://github.com/AmeliaKh/AquariumPipeGame`. Second, click on the fork button shown in Figure 1.

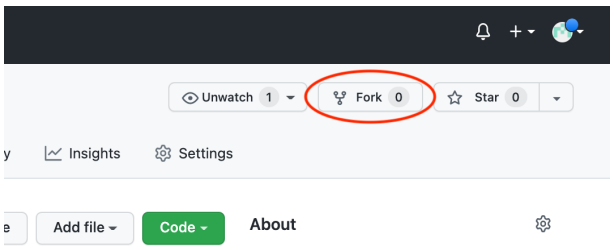

Figure 1: Fork button is circled in red

Next, 'clone' the repository. This will allow you to download the project onto your laptop so you have a local copy. To clone your forked repository, first click the green Code button and then the copy button (these are circled in Figure 2).

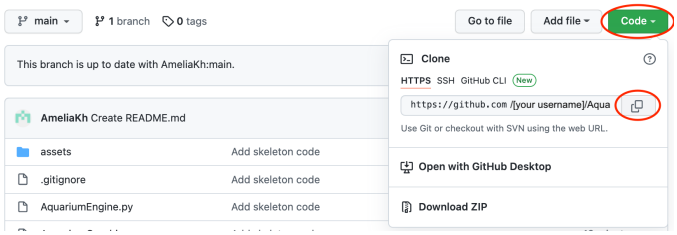

Figure 2: Click the green Code button and then the copy button

Finally, open a terminal window and enter the following command (note the link in the command is what you copied in the previous step):

```
> git clone https://github.com/[your username]/AquariumPipeGame.git
```

Code Overview

Different parts of the code are separated into different files. The graphics for the game is contained in the `AquariumGraphics.py` file. This is responsible for everything visual in your game which includes creating the window, drawing all the rectangles and circles, adding text, etc. You do not need to change this file except in question 13.

The file `main.py` is a wrapper for the rest of the code which initialises the game engine and starts the game loop. This is done as follows:

```
if __name__ == '__main__':
    # Initialise the aquarium game engine
    # Modify the parameters to adjust the game setup
    app = AquariumEngine.Aquarium(
        set_width=580,
        set_height=520
    )
    app.game_loop()
```

`AquariumEngine.py` holds the game engine. This is where all the logic behind the game is kept. Keeping the game logic separate from the graphics helps simplify the code, making it easier for you to make changes now and in the future if you come back to this project.

How the Game Works

The game you will be making consists of several pipes which will be randomly loaded with food for the fish player to catch and eat. The user will need to use their arrow keys to move the fish underneath a tube in order to eat the food dropped down it. You will be given time to add your own creative twist to this game.

The tasks below will help you get familiar with the skeleton code and guide you through building a simple version of the game. After completing this section your game should look like Figure 3.

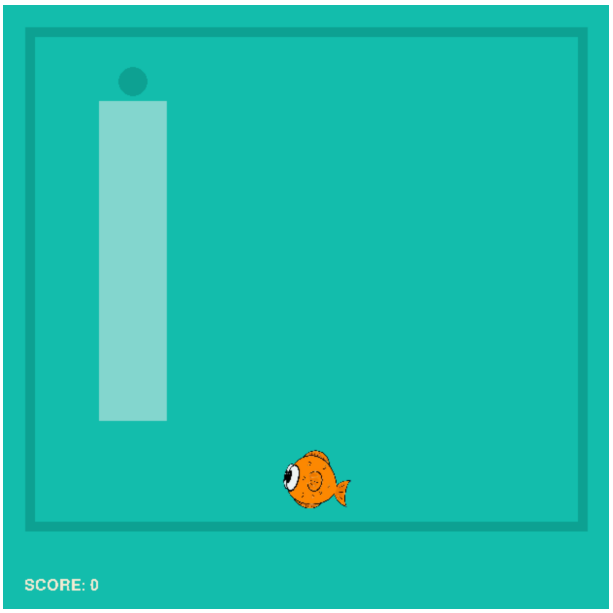

Figure 3: Single pipe with food at the top and player facing left

What to Do

Follow the instructions below. Please note, for some questions (not all) there are comments in the code to guide you to where you should be making the necessary changes.

1. Set the height and width of the fish player to reasonable values so that the fish can fit under the pipe.

   > Hint: use trial and error

2. Set the position of the fish player to a reasonable distance from the bottom of the aquarium and center it horizontally.

3. Fix the code so that the player moves right when the right key is pressed and left when the left key is pressed.

4. Currently, the player's image always faces left. Add some code to change the player's direction to face where it's going.

   > Hint: look inside the assets directory

5. Block the fish from moving outside of the aquarium's border.

   > Hint: look inside `AquariumGraphics.py` to see how the game is padded within the popup window

6. Uncomment `self.pipe.add_food()` so that food is added to the pipe in the game loop.

7. The method referenced above comes from the `Pipe` class. This class also contains a method to move food down the pipe (`move_food()`). Each piece of food is an instance of the `Food` class. `move_food()` moves food objects down the pipe by calling `move_down()` on each piece of food. Fill in `move_down()` inside the `Food` class so the food gets moved down the pipe at a speed of your choice.

8. Right now, food is being added to the pipe continuously. Change the code so that food is only added to the pipe 2% of the time.

   > Hint: import `random`

9. Add some code to `move_food()` which removes food from the pipe if it falls below the game's lower border (i.e `boundary_y`).

10. If food intersects the player (i.e gets eaten), increment the user's score and remove the food.

11. Create a class called `PoisonousFood` which inherits from `Food`

12. Update `PoisonousFood` such that objects of this class drop twice as fast down the pipe as ordinary `Food` objects.

13. Set the colour of `PoisonousFood` to something different to the normal food.

    > Hint: you will need to make changes to the `AquariumGraphics.py` file

14. Change the code so that 20% of food added to the pipe are `PoisonousFood` objects.

15. When the player eats `PoisonousFood` deduct a point.