Documentation of the project

Szigethy Virág & Szigeti Mark

First steps

I searched for multiple python GUI libraries, but in the end I chose the pygame library because I found it interesting and useful. So my first few hours from the development were spent exploring the library and the documentation itself, which can be found here: https://www.pygame.org/docs/

Getting the original textures

After some google searches I found the original textures for the game. Then for testing I placed the background and the bird in a normal position to check the images. And now comes the hard part.

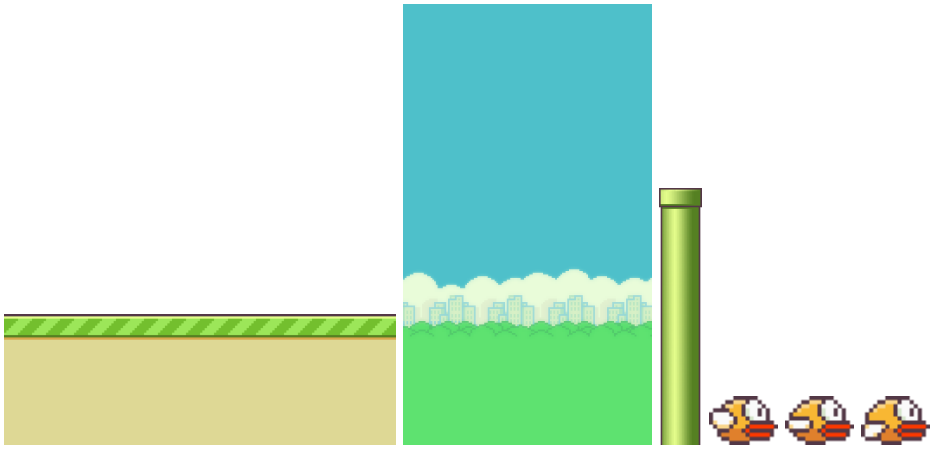

Figure 1. Image files from the original game

Recreating the game

After watching some videos about the game (unfortunately I do not own the original game) I tried to recreate what I saw. First I made a ground moving animation. My idea was to print the image 2 times next to each other, so when the left image slowly runs out from the view, the other one slowly fills the void, creating a continuous animation. Next I added the pipe images, which are generated in pairs (top and bottom), and moving to the left with the same speed as the ground. When a pipe pair leaves the view, a new pair will be generated, creating endless waves. With pipes and the bird, it was time to create the bird control (press any key to jump) and program the bird and the collision physics. In pygame getting the image hitboxes is not a hard thing, so it was not a huge challenge. As a final touch I added a small animation sequence to the bird, to make it more similar to the original game.

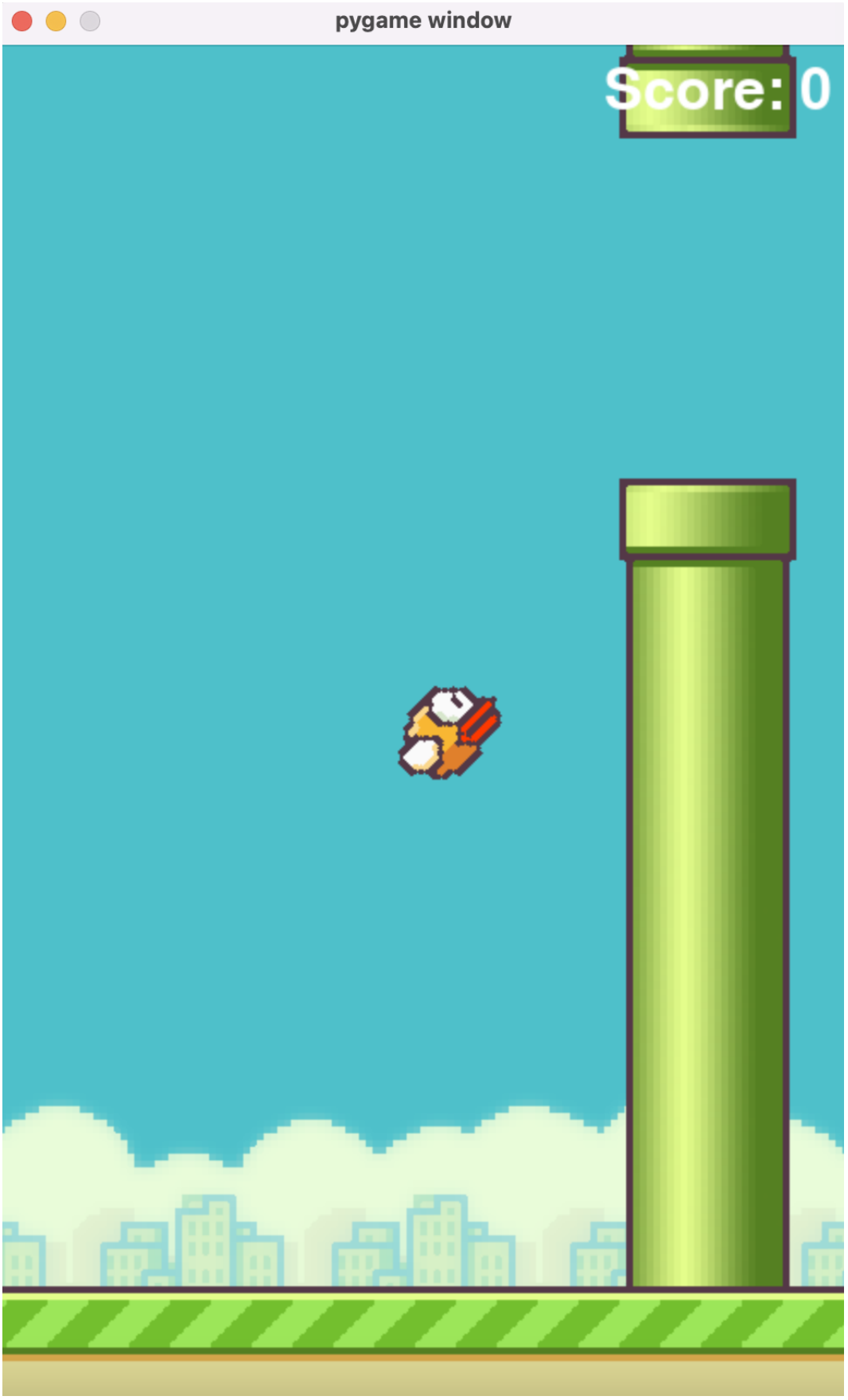

Figure 2. Screenshot from the recreated game played by a human

The Artificial Intelligence

When I started to investigate what technique we should use, it quickly turned out that in most cases people use the so-called Neat method. I watched several youtube videos and investigated Neat's github repository as well (luckily there are examples of different problems solved with this algorithm). Neat stands for NeuroEvolution of Augmenting Topologies and it has its own Python library. In practice Neat creates artificial neural networks which perform as an evolutionary algorithm.

Neat algorithms can perform better than a basic neural network, since during the algorithm, all the networks, the models change to achieve better performance. Each bird gets its own network and from those networks, we select and keep the best ones, so the next generation is created from the previous best one including some small random mutations for each individual bird to keep the evolving process going. This way, the learning starts with random states, and after each generation the population of the birds will perform better and better. With python-neat, we create feed-forward neural networks we will work with.

The network's parameters are stored in an outer config file. Here the most important parameters are the fitness criterion (in our task we select the one with the maximum value), and the population size (which gives the starting number of birds at every generation). Another useful thing can be the fitness threshold, which can stop the simulation at a given point. In the code, genomes represent the set of genes. The genes can be either node genes which are the actual single neurons we are working with, or connection genes which specify the connection between the node genes.

At the end of the optimization, we chose 20 for the size of the populations and a tanh activation function.