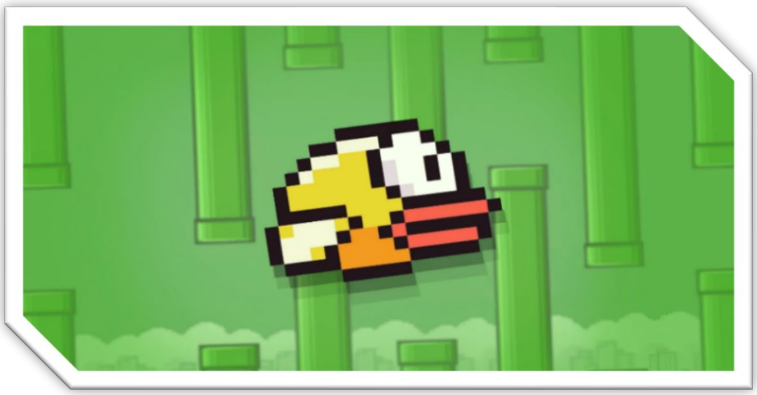
FLAPPY BIRD GENERATIONS

A GENETIC APPROACH TO TURNING DIGITAL DISASTER INTO EVOLUTIONARY VICTORY



NOVEMBER 2025

Contents

Foreword:

In this illustrious era, when engines whir with contemplative purpose and brass-bound contraptions dare to imitate the very whims of Nature, you are hereby beckoned to breathe life into a most extraordinary enterprise: the education of a flock of mechanical birds. Once mere flittering playthings of digital amusement, they now return as earnest scholars of the air, striving to master the noble art of not crashing headlong into obstacles of their own simulated destiny.

Your charge is nothing less than to endow these innocent aviators with something resembling judgement — a delicate undertaking comparable to convincing one's pocket watch to develop opinions on philosophy. Through the mysterious rites of selection, mutation, and heredity, you shall shepherd generation after generation of these metallic fledglings, hoping that, by some beneficent twist of algorithmic fate, they may acquire the wisdom to flap at the proper moment and decline such folly as plummeting earthward.

Should your avian protégés soar with grace through narrow apertures, accept your triumph with dignified pride. But should they dash themselves upon the merciless geometry of their world, resist the temptation to collapse in despair upon your typewriter; for it is well known that the errors of one's own code possess a peculiar talent for lingering longer than any ill-fated bird ever could.

Advance, then, with earnest curiosity, a stout heart, and a sensible reluctance to trust any calculation performed past midnight. May your algorithms find elegance, your evolutionary schemes bear fruit, and your debugging be as swift and bloodless as the Queen's own correspondence.

1. Introduction:

In the spring of 2013, a deceptively simple mobile game quietly emerged from the mind of a solo *Vietnamese* developer. *Flappy Bird*, created by *Dong Nguyen* of *.Gears Studios*, presented a minimalist premise: a tiny pixelated bird flaps forward, and the player must tap the screen repeatedly to navigate it through narrow, vertically aligned pipes [1]. What made the game remarkable was not its sophistication, but rather its merciless difficulty and addictive charm, all wrapped in a retro, 8-bit aesthetic inspired by classic arcade games.

Despite its humble origins, Flappy Bird exploded into global virality during early 2014. Originally released on iOS in May 2013 and later on Android, the game saw a sudden surge in popularity in January of that year, eventually becoming the most downloaded free app on the iOS App Store [2]. At its peak, it was reported to bring in around **US$50,000 per day** in ad revenue – an astonishing sum for a one-man indie project [3].

Yet, the cult of *Flappy Bird* was as fraught as it was fervent. Many players reveled in the frustration, endlessly sharing high scores, screenshots, and memes. Critics, however, slammed its punishing difficulty and apparent lack of depth; some argued it was *not even fun*, but simply a compulsive, time-wasting distraction. The debate only intensified as its creator made a deeply surprising move: in February 2014, Nguyen *removed* the game from both the App Store and Google Play, citing guilt over its addictive nature [4]. The game's sudden disappearance did nothing to diminish its legacy. In fact, it fueled a frenzy: devices with Flappy Bird installed were resold at exorbitant prices, and countless clones flooded the app marketplaces [1]. In less than a year, Flappy Bird had become more than a game – it was a cultural phenomenon, a case study in virality, and a cautionary tale about the unanticipated consequences of mobile success.

For our project, reviving Flappy Bird offers us a powerful sandbox for exploring how a simple rule-based system can give rise to deeply engaging behaviour, and how we can layer genetic algorithms and perceptrons on top of it. By evolving agents that learn to navigate the same unforgiving obstacles that Flappy Bird once presented to tens of millions of players, we not only honour a landmark of mobile gaming history but also gain hands-on experience with key concepts in artificial intelligence and evolutionary computation.

2. Game description

Overview:

Flappy Bird was meant to be a simple game yet being one of the most iconic and frustrating games of its time. The core idea of the game is to keep the bird flying as much as possible, while avoiding the green pipes [5].

The game is an *arcade-style* game, in which the player controls a bird called ***Faby*** which moves persistently to the right. The task is to navigate Faby through *pairs of pipes* (for context see *Figure 2.1*) with *equally sized gaps*, placed at *random heights*!

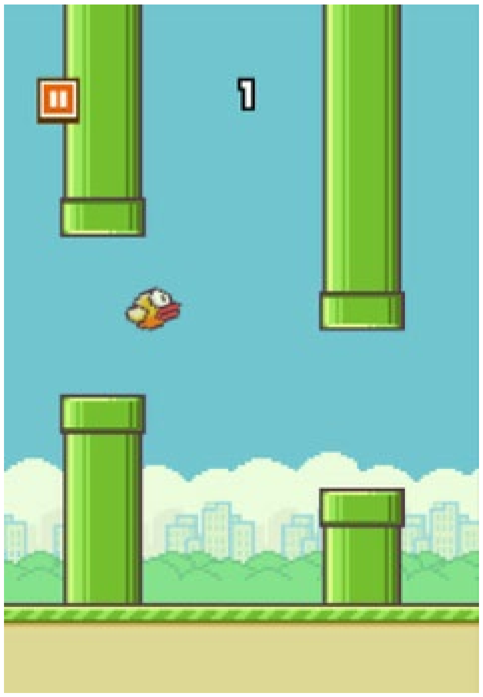


Figure 2.1. – Faby after passing the first pair of pipes.

Faby is affected by gravity, therefore without any input from the player, it will automatically descend until it hits the ground. It only ascends when the player taps (or click in your case) on the screen, making Faby flap its wings. Each flap will raise the bird with a certain amount.

Score:

Each successful pass through a pair of pipes awards the player one point. Colliding with a pipe or with the ground ends the gameplay. During the game over screen, the player is awarded with one of the four medals, displayed in *Figure 2.2*.

Figure 2.2. – Medals awarded for every 10 points

Medal	Points
Bronze	10+
Silver	20+
Gold	30+
Platinum	40+

Screens:

The game has four main screens. The first three can be seen in *Figure 2.3*.

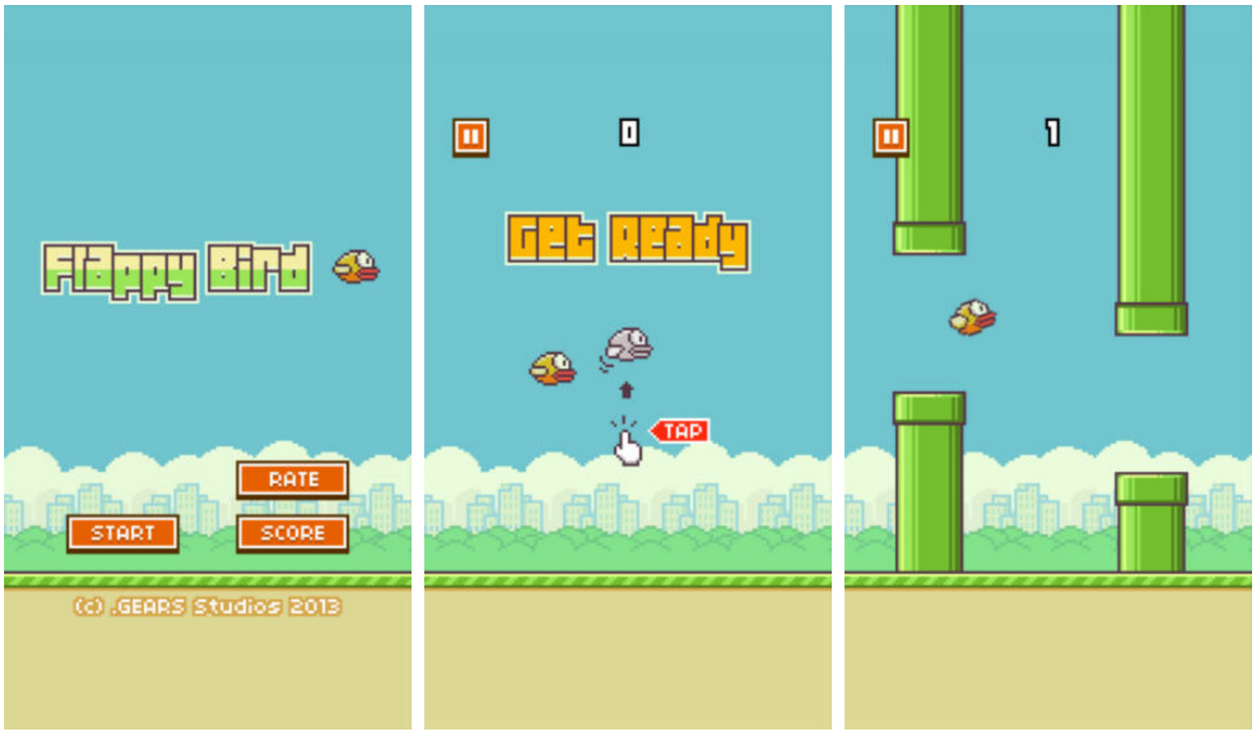


Figure 2.3. – the first three screens of Flappy Bird.

The first screen is the ***title screen***, displaying the title of the game, the player, the copyright of the .GEARS company who made it, and 3 buttons for *rating* the game on App Store/Play Store, displaying *the highest score* in the game, and for *starting* the game.

The second screen is the ***tutorial screen***, which made Flappy Bird to be one of the easiest games to play with, although it required a lifetime to master it.

Finally, the third screen represents the ***game screen***, where the whole action is going on.

Whenever Faby is touching a pipe or the ground, the fourth screen (see Figure 2.4.) will appear on top of the third screen, displaying the *current score*, the *best score* ever recorded and 2 buttons – the first button being the *replay button* and the second being the *share button*, just in case the player wants to proudly announce its performance.

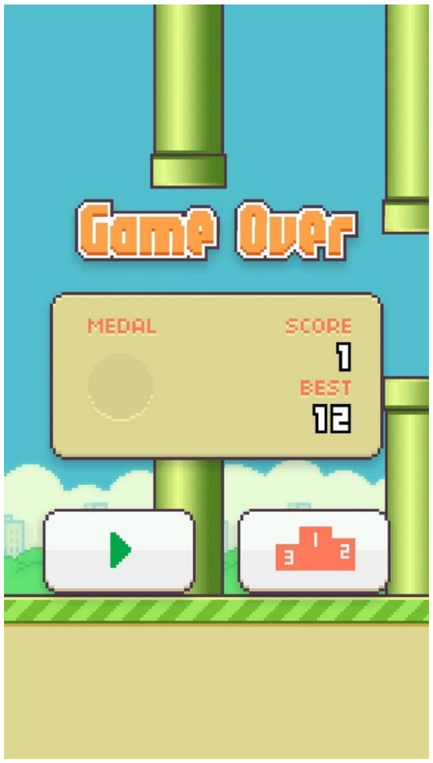


Figure 2.4. – The game over screen

3. Project requirements

For this project, your task is to implement Flappy Bird.

As in the previous project, the game will have a manual mode, which will represent the classic gameplay already presented, and the autonomous mode in which the bird will have a mind on its own.

Implement the ***title screen*** to display, as in the original game, the title and the character in midflight. Add also *your own copyright*. Add 2 buttons for the two game modes (manual and autonomous), together with a button for highest score. This last button will display a smaller popup screen which will display the highest scores in the manual game and in the autonomous game.

Adapt the text in the tutorial screen to be something relevant based on the selected mode. It needs to be something, not necessary identically with the original. So be creative.

Manual Mode:

In the manual mode, the player will be able to play Flappy Bird. Either on Click or by pressing the space bar or whatever key will you prefer, the user will be the one controlling how Faby flies.

Regarding the ***jump*** height of each flap, be free to consider as much as you feel right for it. Do not exaggerate thought 😊.

The ***pipes*** will be at random heights and as the game progresses you can add random distances *between* them, after a score of 30+.

For bonus points and if you have time, you can create ***moving pipes***, both vertically and horizontally, to make the game harder as the score progresses. Such behavior can be seen in this video [6](focus on the pipe placements, not on the Super Mario elements which are outside our goal).

Automatic Mode:

Now is the interesting part. The Autonomous Mode provides each bird with an artificial brain capable of making decisions. Using ***genetic algorithms***, these birds will evolve across generations, learning when to flap and how to adapt to increasingly challenging environments.

A rudimentary implementation of this project can be seen here [7], done by Max Rohowsky. His implementation is just the basic project, implementing the genetic algorithm and the brain of the bird. You need to add the feel and look of the game on top of it.

The Brain – a single layer perceptron:

The brain used in this case will be represented by a simple ***perceptron***, shown in *Figure 3.1*.

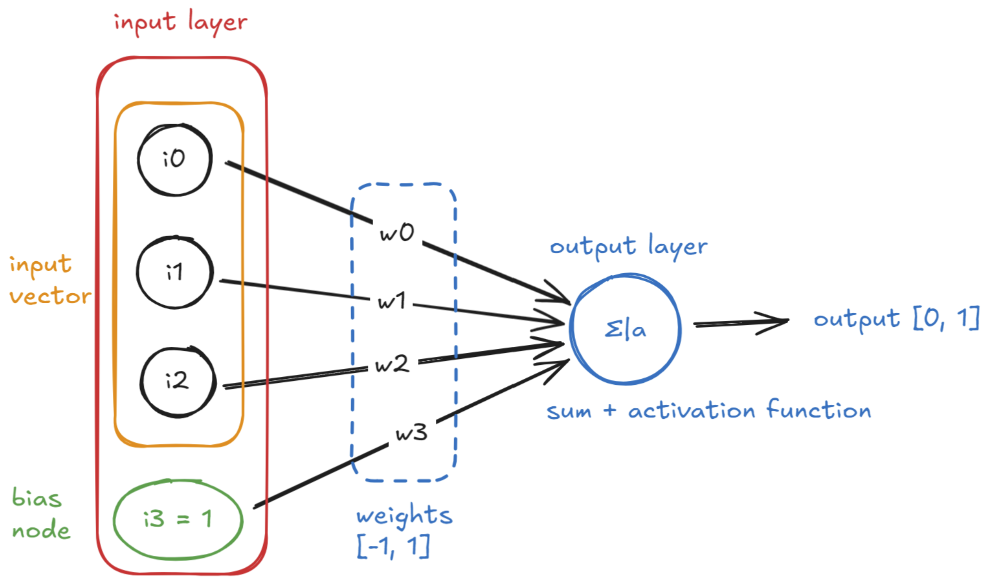


Figure 3.1. – A single layer perceptron, used by the autonomous birds.

The whole idea of having a brain is to take decisions. In case of Faby, it only has to take one decision at a certain amount of time: ***Flap(1)*** or ***No Flap(0)***. Therefore, we can model this process using *a function of multiple variables*, which outputs 1 or 0 based on its input. The coefficients w_i of each variable (called weights) are the unknowns which will be trained (or in this case, evolved) to better produce the desired outcome (flap or no flap).

As in the case of the natural nervous system, the bird will have 3 senses, as can be seen in *Figure 3.2*:

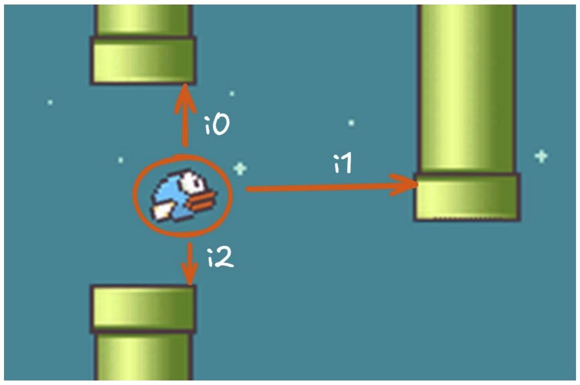


Figure 3.2. – The three senses of autonomous Flappy Bird

- i_0 represents the distance between the bird and the above pipe
- i_1 represents the distance between the bird and the next pipe
- i_2 represents the distance between the bird and the below pipe

These quantities will represent the input for the bird's brain (a single layer perceptron) and in addition to it, there will be a fourth input called the ***bias***. Think of it like the personality of the bird.

The single layer perceptron, presented in *Figure 3.1*, is just a mathematical model which can be written as:

$$f(i_0, i_1, i_2, b) = w_0 i_0 + w_1 i_1 + w_2 i_2 + w_3 b$$

The *input* can be seen as a vector:

$$I = \begin{bmatrix} i_0 \\ i_1 \\ i_2 \\ b \end{bmatrix}$$

The same is valid for the *weights,* considering the bias term to.:

$$W = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix}$$

Therefore, the equation of the perceptron will be written as follows:

$$f(I) = W^T I$$

The output layer, represented by only one neuron, will perform this linear multiplication of each variable, and the result will be passed further to an *activation function*.

Without an activation function, the result of the perceptron will be linear, which is not useful in complex situations. Therefore, the output from the linear multiplication will be passed to a nonlinear function, introducing more complexity which is desired in order to better model nonlinear situations.

For this project, the nonlinear/activation function will be the sigmoid function $\sigma(x)$:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

which has the property of outputting a value between (0, 1), ideal for computing decisions with a certain probability.

Thus, the result of the perceptron can be written as follows:

$$result = \sigma\big(f(I)\big) = \sigma(W^T I)$$

According to the graph of the sigmoid function presented in *Figure 3.3* – also known as the logistic curve – any input less than 0, will produce a 0, while any input greater than 0 will produce a 1. Therefore, the bird will jump with a certain probability, or will not jump, based on other probability.

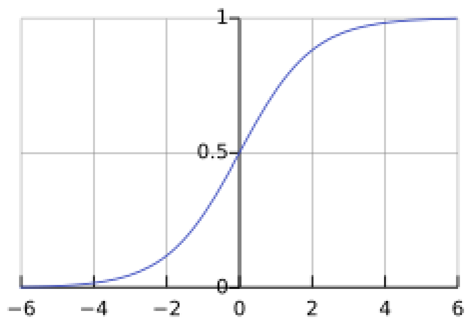


Figure 3.3. – The sigmoid function graph

The Decision

Let's focus on the decision-making process to whether the bird should jump or not. This decision should be made based on a certain ***threshold***. In addition to it, the bird cannot jump if it is in the middle of another jump! A jump can be performed only when the bird is falling and when the output from the perceptron is above a certain threshold.

Therefore, implement a state (a Boolean variable) that describes whether the bird is in the middle of the jump, i.e. is flapping, or not.

The Natural Selection Process:

1. *Speciation*

Start the training process by spawning an initial population of N birds that have the same brain, but with slightly different weights and initial conditions. After all the birds go extinct, i.e. they bump into a pipe or into the ground, comes the ***speciation*** part.

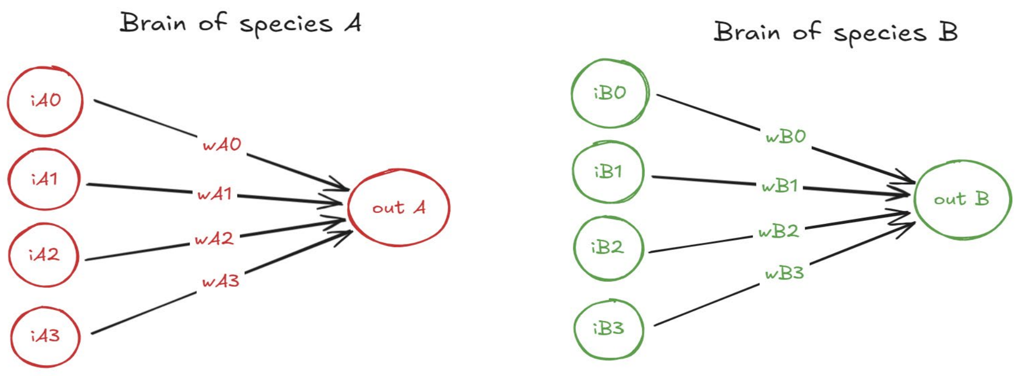


Figure 3.4. – The brains of two bird species

Each bird will be grouped into species (or classes) based on the similarities of their corresponding weights. For the example above (see *Figure 3.4*), the total difference between the species A and the species B will be computed using the following formula:

$$TotalWeightDifference = |w_{A,0} - w_{B,0}| + |w_{A,1} - w_{B,1}| + |w_{A,2} - w_{B,2}| + |w_{A,3} - w_{B,3}|$$

or in a more general case:

$$TotalWeightDifference = \sum_{i=0}^{N-1} |w_{A,i} - w_{B,i}|$$

The smaller the number, the smaller the difference between two brains.

Determining whether the two brains belong to the same species, you need to set a threshold value. If the total weight difference between the two brains is larger than the threshold value, then a new species will be created. Else, they belong to the same species.

After the speciation process, all the birds will be grouped into species – see *Figure 3.5*.

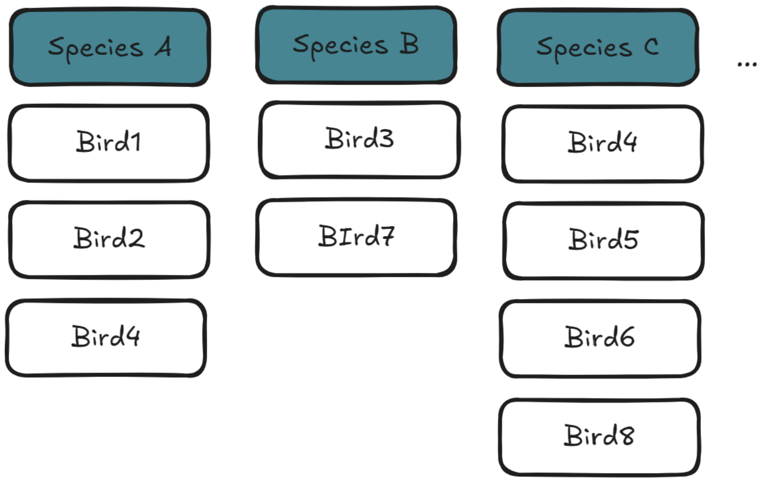


Figure 3.5. – The bird speciation process

2. Fitness score

Fitness refers to the ability to stay alive. The fittest will go the furthest.

There are two levels of fitness:

- Player fitness score: how long one bird survived on its own
- Species fitness score: the average fitness of all birds in that species.

The species fitness score prevents a species with one genius bird and ten foolish ones from being unfairly rewarded. We need to judge the whole species by its collective performance.

3. *Sorting the fittest*

Once every bird has perished, you need to sort everything in descending order of excellence:

- Species are sorted from strongest to weakest based on their species fitness
- Players inside each species are sorted based on their individual fitness scores.

These rankings will provide who deserves to reproduce and which species should go further.

4. *Generate Children for the Next Generation*

Now you need to create a new wave of birds. The first step is to keep the champion from each species, by cloning the very best player of each individual species, without changing anything. This will preserve the greatest success of its generation.

Next, the remaining places of its species will be filled with mutations of a randomly selected bird (other than the champion) to whom you need to apply smaller random mutations on its weights. These small random mutations represent small innovations, sometimes disastrous and sometimes miraculous or somewhere in between. Nevertheless, with enough generations, these tiny changes accumulate until one descendant evolves the uncanny instinct to flap exactly when needed. Max is implementing this part in great detail [7], and for implementation insights, watch his video.

Summary

- Implement both the manual mode and the autonomous mode.
- Follow the video done by Max [7] and use it as a guide.
- Structure your code in an organized manner, use intuitive class names, functions and variables. You can structure your code better than Max did in his video, so please try to do that.
- Add relevant skins, sprints and a personal touch to the game – be creative here and make it interesting. There are a lot of variants of Flappy Bird.

References

[1] S. Perez, "Developer Behind “Flappy Bird,” The Impossible Game Blowing Up The App Store, Says He Just Got Lucky," *TechCrunch,* 2014.

[2] G. Lotan, "A Data-Driven Take on Flappy Bird’s Meteoric Success," *Medium,* 2014.

[3] R. Singel, "The rise and fall of Flappy Bird," *Wired,* 2014.

[4] R. Padilla, "'Flappy Bird' Creator: Game Was Pulled Because It Became an 'Addictive Product'," *MacRumors,* 2014.

[5] V. Ingenito, "Flappy Bird Review," *IGN,* 2014.

[6] pipocaVFX, "Flappy Bird - High Score 999! impossible!," YouTube, 16 02 2014. [Online]. Available: https://www.youtube.com/watch?v=YHH2101OFfI.

[7] M. Rohowsky, "Building the Flappy Bird A.I. without LIBRARIES | Genetic Algorithm | Python/PyCharm," YouTube, 05 02 2023. [Online]. Available: https://www.youtube.com/watch?v=zsGvCwaaMOI.

[8] A. Moscaritolo, "Flappy Bird Tops App Store Charts, Headed to Windows Phone," *PCMag,* 2014.