|  |  |
| --- | --- |
|  | **Wydział Finansów i Zarządzania** **Kierunek: Informatyka** |

Dawid Miśkiewicz

(numer albumu: 81794)

Temat pracy

The Comparison of Evolutionary Algorithms, Genetic Algorithms, and AI in Flappy Bird Development

Opiekun merytoryczny:

Mgr inż., Sebastian Bień

Wrocław 2024

Table of content

Introduction

1. Flappy Bird Game  
1.1 Flappy Bird Game – basic information

1.2 Tools and technologies used for development

1.3 Structuring the game logic – Creating sprites and animations

1.3.1 Player (Bird):

1.3.2 Obstacles (Pipes):

1.4 Implementing Game Physics

1.4.1 Simulating gravity and player movement

1.4.2 Collision detection between the bird and pipes

1.4.2.1 Mask Creation

1.4.2.2 Mask Offsets

1.4.2.3 Overlap Detection

2. Evolutionary Algorithms

2.1 What it is

2.1.1 Definition and inspiration from natural selection.

2.1.2 Key components: population, fitness, selection, crossover, mutation.

2.2. How does it work

2.2.1 Initialization of the population.

2.2.2 Evaluation of individuals using a fitness function.

2.2.3 Selection of parents for reproduction.

2.2.4 Generating offspring through crossover and mutation.

2.3 Example of use

2.3.1 Solving combinatorial problems (e.g., traveling salesman problem).

2.3.2 Evolving game-playing agents.

2.4 Advantages and Limitations

3. Genetic Algorithms

3.1 What it is

3.2. How does it work

3.2.1 Encoding solutions into a genetic format.

3.2.2 Fitness evaluation and ranking of individuals.

3.2.3 Genetic operators: selection (e.g., roulette wheel, tournament), crossover, mutation.

3.3 Example of use

3.3.1 Feature selection in machine learning.

3.3.2 Evolving strategies for board games.

3.3.3 Parameter optimization in engineering problems.

4. Comparison of Algorithms  
4.1 Evolutionary Algorithms in Flappy Bird  
4.2 Genetic Algorithms in Flappy Bird  
4.3 AI Approaches in Flappy Bird  
4.4 Strengths and Weaknesses of Each Approach

5. Results and Analysis  
5.1 Performance Comparisons  
5.2 Optimization Efficiency  
5.3 Game Performance Evaluation

CONCLUSIONS

BIBLIOGRAPHY

TABLE OF FIGURES

SUMMARY

Introduction

1. Flappy Bird Game

This chapter provides an overview of the development of the Flappy Bird game, highlighting the tools and technologies implemented, the design and implementation of the game architecture, and the core mechanics that govern its functionality.

1.1 Flappy Bird Game – basic information

Flappy Bird is a game developed in 2013 by a Vietnamese programmer Dong Nguyen under his company DOTGEARS COMPANY LIMITED. The game was released in May 2013 on the IOS platform and in January 2014 on the Android platform. At the start of 2014, the game gained an unexpected spike in popularity, leading to mixed reviews caused of its difficulty. The objective of the game is to pass through pipes with a **sprite** using only one button to jump. The player’s score is determined by the amount of pipes passed.

1.2 Tools and technologies used for development

The app was developed in the programming language Python using the game engine Pygame in an integrated development environment Visual Studio Code. The following technologies were used to achieve the goals set in the project:

* Python 3.11.9
* Pygame 2.4.0
* neat-python 0.92

1.3 Structuring the game logic – Creating sprites and animations

The main game loop is managed by main() and manual\_play() functions. Inside the loop, two steps occur. The first one – Event Handling is where the loop processes events such as window events for exiting the game and key presses for jumping the bird.

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

The next step is physics which is handled by the Bird class in the update\_position() method, where gravity and jump mechanics are applied to the bird. The bird’s position is updated for each frame based on velocity that is adjusted with gravity. Pipes move with constant speed and their height is random, as defined in the Pipe class.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Czcionka

Opis wygenerowany automatycznie

1.3.1 Player (Bird):

The bird is represented using three images (in flight) that have been loaded using pygame.image.load() and scaled to fit inside the window. Animation is controlled in the animation() method inside the Bird class, which shows different bird images according to a counter.



Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Czcionka

Opis wygenerowany automatycznie

The bird is then drawn with a rotation to simulate tilting during movement. The rotation is handled by pygame.transform.rotate().

1.3.2 Obstacles (Pipes):

Pipes are represented by an image (pipe.png) and have a top and a bottom section. The Pipe class creates, moves, and draws these pipes. Pipes are portrayed through the animation() method of the Pipe class.

Obraz zawierający tekst, Czcionka, zrzut ekranu, numer

Opis wygenerowany automatycznie

Pipes move across the screen in the update\_position() method by decreasing their x position and new pipes are periodically created.

1.3.3 Background and Base:

The background (BG.png) and the base (base.png) are drawn in the draw\_window() function. The base’s horizontal movement creates a scrolling effect, and when it moves out of the screen, it reappears on the opposite side. This is done by the Base class:

Obraz zawierający zrzut ekranu, woda

Opis wygenerowany automatycznie

Obraz zawierający zrzut ekranu, żółty, Prostokąt, zieleń

Opis wygenerowany automatycznieObraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

1.4 Implementing Game Physics

1.4.1 Simulating gravity and player movement   
 In the game, the motion of the bird is governed by a simplified kinematic equation. Said bird’s motion is handled in the PlayerBird class, where gravity and upward movement are defined by the equation presented below. The gravitational effect on the bird’s vertical position (y) is calculated as:

Where:

is the bird’s initial upward velocity of the bird. It is set to -10.5 units when the bird jumps.

is a constant downward gravitational acceleration. The acceleration is approximated to 3 units /

it the time step, tracked as tick\_count.

The position change is calculated as:

Obraz zawierający tekst, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

To guarantee smooth motion and prevent excessive falling speed, the velocity is capped at a maximum of 16 units:



The bird’s vertical position (self.y) is then updated by adding to its current value:  


When the bird jumps, its velocity is set to an upward value, resisting gravity temporarily. The jump() function resets the parameters to simulate the jump effect:

Obraz zawierający tekst, Czcionka, zrzut ekranu, numer

Opis wygenerowany automatycznie

The bird tilt is also adjusted to reflect its vertical movement visually. When the bird is rising, the tilt is positive, simulating a nose-up effect. Thus, the tilt takes on a positive value during ascension, creating a nose-up impression, and negative during descent, tilting gently downwards until reaching :

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

This guarantees that it appears realistic concerning the effect of gravity in the game.



1.4.2 Collision detection between the bird and pipes

The collision detection mechanism provides an accurate detection between the player (bird) and the game obstacles (pipes). Pixel-perfect collision is achieved by using masks to give a binary representation of the visible pixels of game objects.

Obraz zawierający zrzut ekranu, Prostokąt, sztuka

Opis wygenerowany automatycznie

1.4.2.1 Mask Creation

Masks are generated for the bird as well as both the bottom and top pipes.

Obraz zawierający tekst, Czcionka, zrzut ekranu, linia

Opis wygenerowany automatycznie

1.4.2.2 Mask Offsets

Offsets are calculated to cancel offsets from the bird’s mask to the pipe's masks. The offsets are taking into account the relative positions of the bird and pipes:

Obraz zawierający zrzut ekranu, Czcionka, tekst, linia

Opis wygenerowany automatycznie

1.4.2.3 Overlap Detection

The overlap() method is used to determine whether any pixels in the bird’s mask overlap with the pipe masks:

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

If any overlap is detected, a collision has occurred:

Obraz zawierający tekst, Czcionka, zrzut ekranu, numer

Opis wygenerowany automatycznie

This method ensures that collisions are accurately detected, even with tricky shapes such as a tilted bird.

2. Evolutionary Algorithm

This chapter describes the technology utilized in the project and the design principles followed by the author while developing and implementing evolutionary algorithms for the thesis.

2.1 What is it?

An evolutionary algorithm is a computer-based approach to problem-solving modeled on natural evolution through the application of computer science and artificial intelligence. It simulates certain biological mechanisms, namely reproduction, mutation, and recombination, in order to solve these problems. Such algorithms are based on Darwin's conception of natural selection whereby weaker solutions are sorted from stronger ones and the more fit alternatives are preserved and bred, generation after generation. The ultimate goal is the development of solutions that are optimal enough to fulfill the desired purpose.

2.1 Key components: population, fitness, selection, crossover, mutation.

The fundamental components of evolutionary algorithms are:

Population: Set of candidate solutions that evolve with time. In our case, we have the population existing as a list of BirdControl objects.

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

Fitness Function: A quantitative measure for the assessment of an individual's quality or performance in the population. For example, the fitness of a bird is rated on its performance in the simulation.



Selection Mechanism: A process of identifying and ranking individuals with high fitness for reproduction. In this study, the population is sorted by fitness in descending order.



Crossover: Genetic operation whereby two parents exchange some but not all characteristics in the formation of offspring. The approach employs a simple crossover that weighs features from one of the parents.

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

**Mutation**: A random or semi-random change applied to individual candidate solutions for creating diversity and thereby avoiding premature convergence. The mutation function is used to alter weights with a certain probability.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

2.2. How does it work

2.2.1 Initialization of the population.

The algorithm begins by initializing a population of candidate solutions. In this study, each solution represents the neural network weights of a simulated bird agent. These weights are initialized randomly to ensure diversity in the initial generation.

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

2.2.2 Evaluation of individuals using a fitness function.

The evaluation starts from an initial population depending on the candidate solutions. In this case, each solution is just the weight of a neurally driven simulated bird agent. Initially, all the candidate solutions undergo random initialization, so as to secure some degree of diversity in the first generation.



2.2.3 Selection of parents for reproduction.

Individuals with higher fitness are more likely to be selected for reproduction. This selection mechanism ensures that good genes are carried over to the next generation.



2.2.4 Generating offspring through crossover and mutation.

Reproduction involves creating offspring by combining traits of selected parents using crossover and mutation. These operations introduce variation and enable exploration of the solution space.



2.3 Example of use

2.3.1 Solving combinatorial problems (e.g., traveling salesman problem).

Evolutionary algorithms are well-suited for solving combinatorial problems, such as the traveling salesman problem (TSP). By evolution on the order of the sequences of cities, EAs are capable of efficiently minimizing the total travel distance in very large-size instances.

2.3.2 Evolving game-playing agents.

In the case of evolving game-playing agents, the agents gradually learn how to overcome obstacles using increasingly efficient methods from one generation to the next, which is reflected in their higher overall performance. The algorithm's ability to optimize the neural network weights controlling the decision-making process of birds offers promising perspectives for the use of intelligent systems.

2.4 Advantages and Limitations

Evolutionary algorithms are characterized by their strengths and numerous applications, making them invaluable in several fields of engineering and optimization problems. However, specific challenges should be taken into consideration. This section provides a brief description of the major benefits and limitations experienced during the implementation and tests on evolutionary algorithms in this study.

2.4.1 Advantages

* Versatility: Capable of addressing a wide range of optimization problems.
* Global Search Capability: Effectively explores complex solution spaces, reducing the likelihood of being trapped in local optima.
* Adaptability: Easily integrates with various problem domains, including game AI and combinatorial optimization.

2.4.2 Limitations

* Computational Cost: High resource requirements for large populations or complex fitness evaluations.
* Convergence Challenges: Risk of premature convergence, necessitating careful tuning of algorithm parameters.

3. Genetic Algorithms

3.1 What it is

A genetic algorithm (GA) is an evolutionary algorithm inspired by natural selection and genetics, it provides a basis for finding the optimal solution or near-optimal solutions for complex problems. In the context of this study, the problem involves the optimization of an AI agent (the bird) capable of playing the Flappy Bird game with extreme effectiveness. The GA receives training through a process imitative of biological evolution, wherein a population of candidate solutions (genomes) is taken through successive generations of evolution. In our implementation, an evolutionary algorithm called NEAT (NeuroEvolution of Augmenting Topologies) employs an interesting way of evolving neural networks. Derived from biological inspiration, NEAT stretches the boundaries of basic neuron weight evolution by including the evolution of the structure called topology within the neural networks. Because of this, NEAT is able to discover, progressively, solutions that configure quite adaptable and efficient neural networks. The solutions in NEAT are represented as genomes that encode neural network structures and their associated weights. A genome in NEAT contains a collection of genes representing the interconnections and properties of neurons. This encoding scheme is directly mapped onto the neural networks controlling the movements of the flying objects.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

3.2. How does it work

The process of evolving the bird’s neural network follows stages such as encoding solutions, fitness evaluation, and genetic operators.

3.2.1 Encoding solutions into a genetic format.

Each bird’s neural network is characterized as an aspect of a genome-relating collection of genes for describing the neural network structure and weight parameters. In NEAT, this genome incorporates both the topology (nodes and connections) and the weights (parameters) of the neural network. The network decides a bird's action in the game on the basis of its sensory inputs.

Obraz zawierający tekst, zrzut ekranu, Czcionka, oprogramowanie

Opis wygenerowany automatycznie

3.2.2 Fitness evaluation and ranking of individuals.

Based on how long they live through the game, each bird (individual) is assigned a fitness score. The longer each bird can survive, as measured by passing successfully through pipes, the better its fitness. If a bird collides with a pipe or falls to the ground, its fitness is penalized.

Obraz zawierający zrzut ekranu, tekst, Czcionka

Opis wygenerowany automatycznie

3.2.3 Genetic operators: selection (e.g., roulette wheel, tournament), crossover, mutation.

Neat uses three primary genetic operators: selection, crossover, and mutation. The selection process essentially selects the best individuals to pass their genes. Crossover is the joining of genetic material from two parents to form offspring, while mutation introduces random changes in the genome for the sole purpose of creating diversity in the population.



The population is created with a specific configuration.



Run for 10 generations, where selection and mutation occur.

3.3 Example of use

In this code, the genetic algorithm is used to create an AI capable of navigating the Flappy Bird environment. This application serves as a good example of tasks such as feature selection, strategy evolution, and parameter optimization performed by genetic algorithms.

3.3.1 Feature selection in machine learning.

Even if this can hardly be counted as a typical machine learning problem, the evolution applied to the neural network of the bird can be considered the optimization of inputs (vertical position of the bird and distances from the pipe) to the actions, that is, whether the bird should stay still or jump. This can be related to feature selection, where the system is trying to find the best representation of input features for making decisions.



3.3.2 Evolving strategies for board games.

The evolving of strategies is conceptually similar to the AI systems that play board games exactly like how AI systems evolve the strategies to play chess or Go. The bird strategies are evolved in the task of traversing pipes, while the AI systems solve it in a practically equal approach. Each bird strategy is governed by a neural network that learns to "decide" when to jump based on varying inputs in the environment.

Obraz zawierający Czcionka, tekst, zrzut ekranu, Grafika

Opis wygenerowany automatycznie

3.3.3 Parameter optimization in engineering problems.

The evolution of the bird controller neural network is like parameter optimization in engineering problems. Neural network weights are varied over generations to get the bird to stay alive longer. One can think of it as optimizing design parameters in an engineering context in order to attain better performance under certain conditions.