AI FLAPPY BIRD USING NEAT

A PROJECT REPORT

Submitted by

SHARAN RAM M (210701242)
A.R.SHARVESH (210701243)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE

CHENNAI

MAY 2024

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

BONAFIDE CERTIFICATE

Certified that this Thesis titled **"AI FLAPPY BIRD USING NEAT** " is the bonafide work of "**SHARAN RAM M (210701242), A.R.SHARVESH (210701243)"** who carried out the work under my supervision. Certified furtherthat to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred onan earlier occasion on this or any other candidate.

SIGNATURE

Mr.RAKESH KUMAR M B.Tech., M.E.,

PROJECT COORDINATOR

Assistant Professor (SG)

Department of Computer Science and Engineering

Rajalakshmi Engineering College

Chennai - 602 105

Submitted to Project Viva-Voce Examination held on_____

Internal Examiner **External Examine**r

ABSTRACT

In order to learn the game Flappy Bird, which involves a bird navigating between holes in pipes, this article utilizes the NEAT (NeuroEvolution of Augmenting Topologies) method. Neural networks are optimized using NEAT to control the bird's motions depending on inputs from the game state, progressing from simple to sophisticated structures. The networks are designed to fly as long as far as possible without colliding.

Over generations, the networks undergo selection, crossover, and mutation as a result of recording the position of the bird and the closest pipes as inputs. The outcomes show that NEAT efficiently creates neural networks with strong Flappy Bird gameplay techniques. The effectiveness of NEAT in adapting neural network topologies for challenging control tasks is demonstrated in this work, underscoring its potential for in-the-moment decision-making in dynamic settings.

ACKNOWLEDGMENT

First, we thank the almighty god for the successful completion of the project. Our sincere thanks to our chairman **Mr. S. Meganathan B.E., F.I.E.,** for his sincere endeavor in educating us in his premier institution. We would like to express our deep gratitude to our beloved Chairperson **Dr. Thangam Meganathan Ph.D.,** for her enthusiastic motivation which inspired us a lot in completing this project and Vice Chairman **Mr. Abhay Shankar Meganathan B.E., M.S.,** for providing us with the requisite infrastructure.

We also express our sincere gratitude to our college Principal, **Dr. S. N. Murugesan M.E., PhD.,** and **Dr. P. KUMAR M.E., PhD, Director computing and information science , and Head Of Department of Computer Science and Engineering** and our project coordinator **Rakesh Kumar M B.Tech.,M.E.,** for her encouragement and guiding us throughout the project towards successful completion of this project and to our parents, friends, all faculty members and supporting staffs for their direct and indirect involvement in successful completion of the project for their encouragement and support.

SHARAN RAM M

A.R.SHARVESH

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The objective of the straightforward yet difficult side-scrolling game Flappy Bird is to maneuver a bird between pipe gaps without colliding. This research trains an artificial intelligence (AI) to play Flappy Bird on its own using the NEAT (NeuroEvolution of Augmenting Topologies) algorithm. NEAT is ideally suited for creating efficient gaming tactics since it gradually increases the complexity of neural networks. The position of the bird and the closest pipes are two examples of the game state inputs that the neural networks receive and modify throughout generations to enhance their functionality. This study highlights NEAT's potential for handling complicated control issues with minimum human interaction by showcasing its capacity to evolve resilient neural network topologies for real-time decision-making in dynamic contexts.

The AI gains the ability to guide the bird around obstacles using NEAT, maximizing both the bird's flying time and distance covered. The goal of this study is to demonstrate how machine learning techniques—specifically, NEAT—can be used to effectively master the challenging and dynamic gameplay of Flappy Bird. With the use of evolutionary concepts and neural network encoding of game state information, NEAT enables the creation of intelligent agents that can learn and adjust to intricate gaming situations. The use of NEAT in Flappy Bird not only sheds light on the technology's effectiveness in game production but also highlights its wider applicability in resolving real-world issues involving independent decision-making in dynamic environments. The knowledge and use of machine learning algorithms in gaming and other fields is advanced by this study.

1.1 PROBLEM STATEMENT

The goal of this project is to automatically learn the Flappy Bird game using the NEAT (NeuroEvolution of Augmenting Topologies) algorithm. In the difficult challenge of Flappy Bird, a bird must flap its wings to get over obstacles. The goal is to create a machine learning system that will allow the agent to pick up efficient techniques for optimizing flight time and distance without running into other objects. Training neural networks to regulate the bird's motions in response to real-time visual inputs is the main issue. The objective is to build neural network topologies with adaptive behaviors for effective navigation through the iterative evolution process of NEAT. This research demonstrates how well NEAT works in challenging control scenarios seen in gaming contexts.

1.2 SCOPE OF THE WORK

The goal of this research is to use machine learning techniques to create an autonomous agent that can play the Flappy Bird game proficiently using the NEAT (NeuroEvolution of Augmenting Topologies) algorithm. Implementing NEAT to develop neural networks that regulate the bird's vertical motions in response to inputs from the game state is part of the scope. The position of the bird and the positions of the pipes must be encoded as inputs, the networks must be trained to maximize flying time and distance flown without collisions, and the method parameters must be optimized for effective learning. Metrics for evaluation comprise the agent's performance in terms of barriers avoided and scores attained. Comparing NEAT-based strategies with alternative machine learning techniques for Flappy Bird gaming is another aspect of the study.

1.3 AIM AND OBJECTIVES OF THE PROJECT

The goal of this research is to create an autonomous agent that can beat the Flappy Bird game by using the NEAT (NeuroEvolution of Augmenting Topologies) method. The goal is to develop a neural network-based system that can negotiate the complexity of the Flappy Bird environment with the least amount of human interaction by utilizing the evolutionary concepts ingrained in NEAT. This research trains the agent to efficiently manage the bird's motions, optimizing for flight time and obstacle avoidance, in an effort to push the frontiers of machine learning in gaming. The objective is to demonstrate the potential of NEAT in evolving intelligent behaviors in difficult gaming scenarios through thorough experimentation and analysis, offering important insights into the capabilities of evolutionary algorithms for real-world applications beyond conventional domains.

The objective of this research is to create an autonomous agent that can perform well in the Flappy Bird game by applying the NEAT (NeuroEvolution of Augmenting Topologies) method. To do this, a simulation platform based on the Flappy Bird gaming environment must be implemented. Additionally, NEAT must be integrated in order to grow neural networks that govern the bird's vertical motions. The evolved networks are going to be taught to maximize flight length while reducing collisions with obstacles by storing relevant game state inputs, such the placement of pipes and the position of birds. Additionally, the NEAT-based agent will be evaluated in comparison to human players and other baseline methods in order to shed light on how effective NEAT is in creating intelligent agents for use in gaming environments.

1.4 RESOURCES

A significant amount of secondary research, including conference reviews, standard papers, business journals, white papers, certified manuscripts, and analyst insights, has driven the project. Due to the intricacy of implementing NEAT on Flappy Bird, substantial resources are required for its effective implementation. Important materials required for the project include:

• Workstations (PCs, laptops, etc.) with sufficient equipment to provide effective material collection and study.
• Unrestricted internet access for resource retrieval and online exploration.
• Free access to all university buildings, including the lab, which makes it easier to read a wide range of materials, including technical manuscripts for machine learning projects like Flappy Bird with NEAT and academic resources (tutorials, programming examples, bulletins, publications, e-books, journals).

1.5 MOTIVATION

This research is motivated by the goal of utilizing machine learning to address challenging problems in artificial intelligence and gaming. With its complicated dynamics and deceptively easy gameplay, Flappy Bird is a perfect platform for testing the capabilities of algorithms such as NEAT (NeuroEvolution of Augmenting Topologies). We hope to explore the fields of neural network optimization and evolutionary computation by using NEAT to master Flappy Bird and uncover the algorithm's capacity for learning and adapting to changing conditions. This project not only broadens our knowledge of machine learning but also creates opportunities for the development of autonomous systems that can resolve issues in the real world.Our goal is to demonstrate how AI can revolutionize gaming and open doors for creative thinking across a range of industries.

CHAPTER 2

LITRETURE SURVEY

In [1] A large number of algorithms to generate behaviors of game agents have been developed in recent years. Most of them are based on artificial intelligence techniques that need a training stage. In this context, this paper proposes a minimal training strategy to develop autonomous virtual players using the NEAT neuroevolutionary algorithm to evolve an agent capable of playing the Flappy Bird game. NEAT was used to find the simplest neural network architecture that can perfectly play the game. The modeling of the scenarios and the fitness function were set to ensure adequate representation of the problem compared to the real game. The fitness function is a weighted average based on multiple scenarios and scenario-specific components. Coupling the minimal training strategy, a representative fitness and NEAT, the algorithm had a short convergence time (around 20 generations), with a low complexity network and achieved the perfect behavior in the game.

In [2] [3] This research explores the application of the NeuroEvolution of Augmenting Topologies (NEAT) algorithm in training an artificial intelligence (AI) model to play the popular mobile game Flappy Bird. The primary objective is to investigate the effectiveness of NEAT in enabling an AI agent to learn and adapt to the dynamic challenges presented by the game environment. The Flappy Bird game serves as an ideal testing ground for evaluating the adaptability and learning capabilities of the NEAT algorithm.The methodology involves implementing the NEAT algorithm within the game framework, allowing the AI to learn from scratch without prior knowledge of the game dynamics. Various parameters and configurations are fine-tuned to optimize the learning process. The results indicate the AI's ability to navigate through the game obstacles and improve its performance over time.In addition to detailing the NEAT algorithm and its application in the Flappy Bird context, this paper provides insights into

the learning process, challenges encountered during implementation, and a discussion on the obtained results. Comparative analyses with other AI approaches highlight the unique advantages and contributions of the NEAT algorithm in the domain of gaming AI.The findings contribute to the growing body of research on AI in gaming and showcase the potential of evolutionary algorithms, specifically NEAT, in training agents for complex and dynamic environments. The implications of this research extend beyond the realm of gaming, offering valuable insights into the adaptability of evolutionary algorithms in real-world scenarios.

In [4] discusses of Video games and mobile games are a large part of modern society in which multiple people can compete against each other. Video games have a predefined set of rules and a goal which make them fun and challenging. This also makes them a great playground for different reinforcement learning algorithms, machine learning algorithms which attempt to maximize some reward by training themselves as they explore the environment. Defining the rules and rewards of a real-world environment can be challenging, which makes video game a great place to learn about these algorithms.

In [5] discusses about The objective of conduct the current research is to understand the deep relationship between gaming along with how artificial intelligence (AI) is being incorporated into these fields. It is clear that gaming and AI have a natural connection, and the introduction of AI only strengthens this connection. This study aims to highlight the potential benefits of using AI in gaming and simulation, as well as the challenges that may arise in these contexts.

In [6] discusses One of the most popular subjects being investigated in AI nowadays is game learning. Addressing such issues require proper domain specific knowledge.So one such game was developed ie., flappy bird where the agent learns itself on how to

avoid the obstacles and also tries to maximize the score based on the rewards and punishments it receives. No prior knowledge was given to the agent regarding the environment. Instead of utilizing raw pixels, the agent was trained using domain-specific features such as the bird's speed, the distance between pipes, and the height of the pipes, which significantly simplifies the feature space and avoids the need for deeper models to automatically extract underlying data. The agent was trained using the NeuroEvolution of Augmenting Topologies(NEAT) algorithm and is talked about in this paper.

In [7] Evolutionary algorithms are a subfield of artificial intelligence, which, even if deep learning is the leading factor nowadays, gained more attention in recent years, due to the increase of processing power. One of the most popular topics in this field is Neuroevolution, since is combines the advantages of evolutionary algorithms with the advantages of neural and deep neural networks. In this paper, we demonstrate how can NeuroEvolution of Augmenting Topologies (NEAT) be used for game playing.

CHAPTER 3

SYSTEM DESIGN

3.1 GENERAL

In this section, we aim to elucidate how the various components operate synergistically when organized and arranged together. This holistic integration is visually represented through a flowchart presented below, offering a comprehensive understanding of the interconnected workflow.

3.2 SYSTEM ARCHITECTURE DIAGRAM

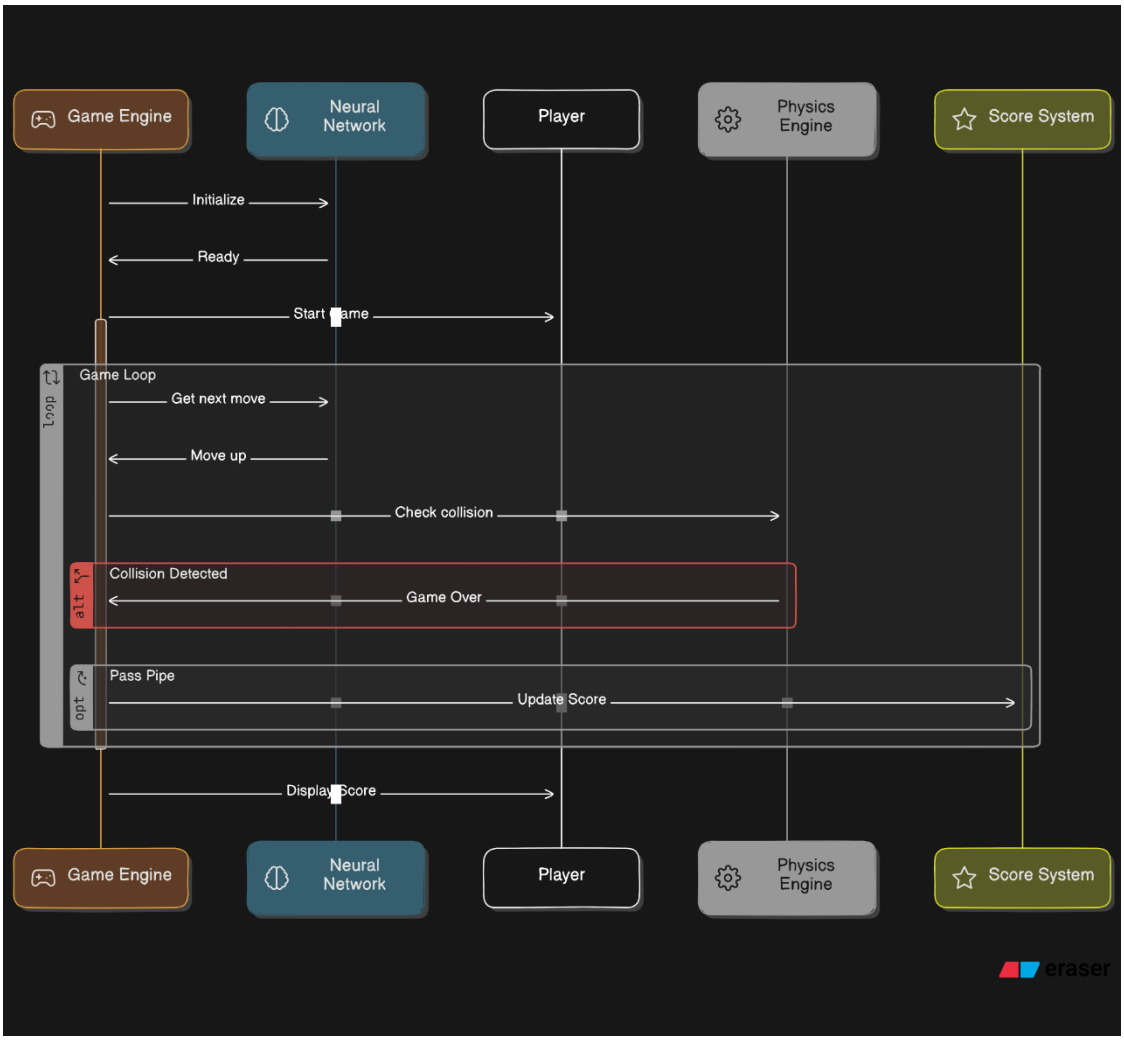


Fig 3.1: System Architecture

3.3 DEVELOPMENTAL ENVIRONMENT

3.3.1 HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the system's implementation. It should therefore be a complete and consistent specification of the entire system. It is generally used by software engineers as the starting point for the system design.

Table 3.1 Hardware Requirements

COMPONENTS	SPECIFICATION
PROCESSOR	Intel Core i5
RAM	8 GB RAM
GPU	NVIDIA GeForce GTX 1650
MONITOR	15" COLOR
HARD DISK	512 GB
PROCESSOR SPEED	MINIMUM 1.1 GHz

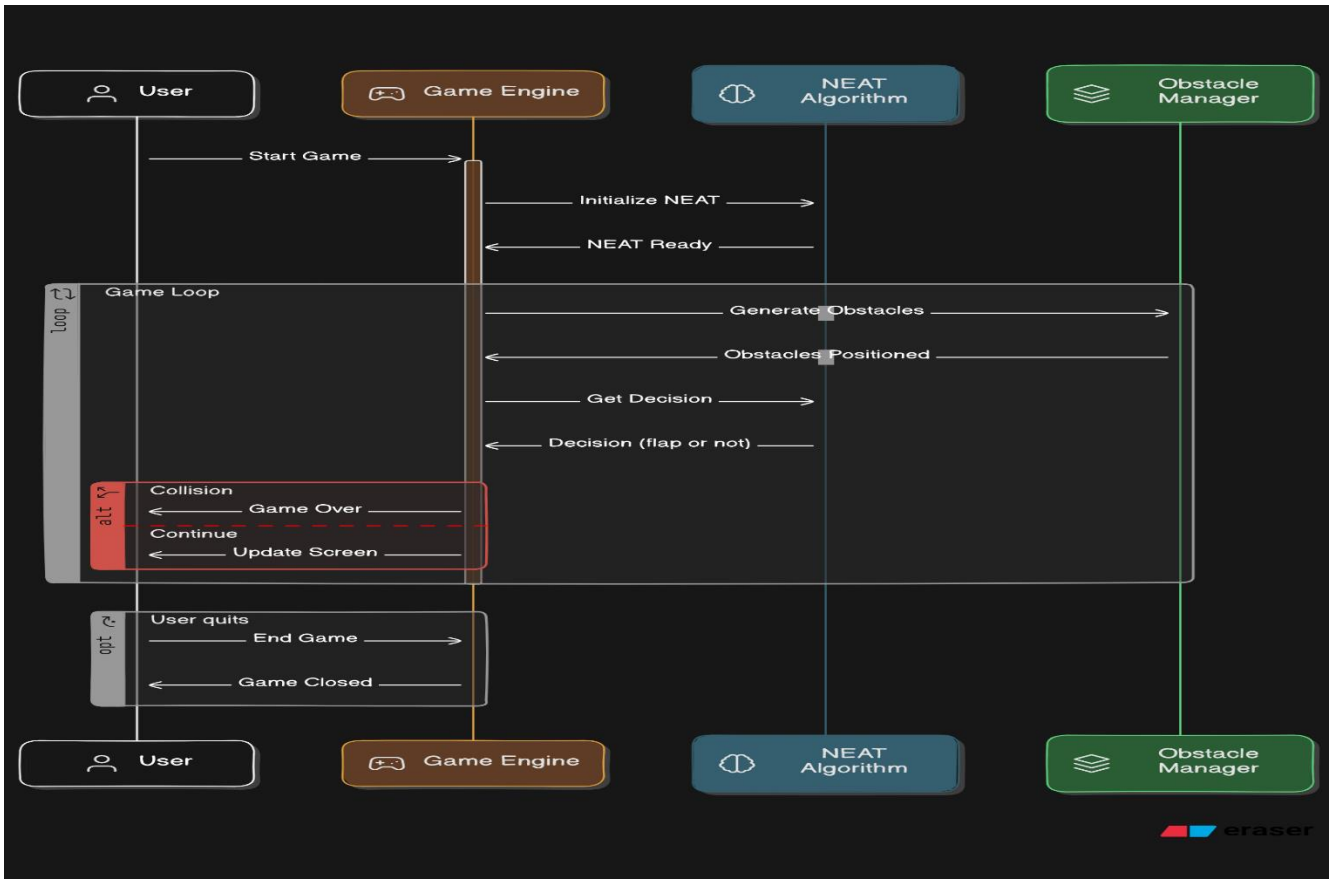
3.3.2 SOFTWARE REQUIREMENTS

The software requirements document is the specifications of the system. It should include both a definition and a specification of requirements. It is a set of what the system should rather be doing than focus on how it should be done. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating the cost, planning team activities, performing tasks, tracking the team, and tracking the team's progress throughout the development activity.**Python IDLE,** and **chrome** would all be required.

DESIGN OF THE ENTIRE SYSTEM

3.3.3 SEQUENCE DIAGRAM

The sequence diagram outlines the operation of our AI-powered lawyer assignment system, commencing with clients submitting legal queries, which are then processed using preprocessing techniques and NLP. Predictive modeling aids in matching queries with lawyers from a legal database, with assigned lawyers promptly notified. Upon acceptance, clients receive confirmation of the assignment, initiating legal proceedings. Continuous monitoring and performance evaluations are conducted throughout, integrating feedback from clients and lawyers to enhance system efficiency.



3.2 sequence diagram

CHAPTER 4

PROJECT DESCRIPTION

4.1 METHODOLODGY

Game Environment Setup : Use Pygame to create a Flappy Bird simulation.

NEAT Integration : Define neural network architecture for game control by implementing NEAT-Python.

Fitness Function: Determine how long and far networks can fly without colliding.

Training : Use NEAT's genetic operators to evolve networks across generations after initializing them with random weights.

Evaluation : Evaluate the functionality of evolved networks in the Flappy Bird game. Evaluate outcomes to determine how effective NEAT is in playing games on its own.

4.2 MODULE DESCRIPTION

For independent play, this module incorporates the NEAT (NeuroEvolution of Augmenting Topologies) algorithm into the Flappy Bird game. Neural networks are evolved using the NEAT algorithm to regulate the bird's vertical motions in response to inputs related to the game state. Integrates the Flappy Bird gaming world into the system, giving the neural networks instantaneous input.

NEAT Implementation: To maximize gaming tactics, NEAT is used to generate neural network topologies, beginning with basic structures and progressively becoming more complicated. Fitness Evaluation: Establishes fitness requirements for evolution by evaluating neural networks according to the length of flight and the distance covered without collisions. Oversaw the neural network's selection, crossover, and mutation across several generations in order to enhance performance.

With the use of NEAT, this module makes it easier to create and assess AI agents that can play Flappy Bird on their own.

CHAPTER 5

RESULTS AND DISCUSSIONS

5.1 OUTPUT

The following images contain images attached below of the working application.

Example instance of generating a request and receiving the response:

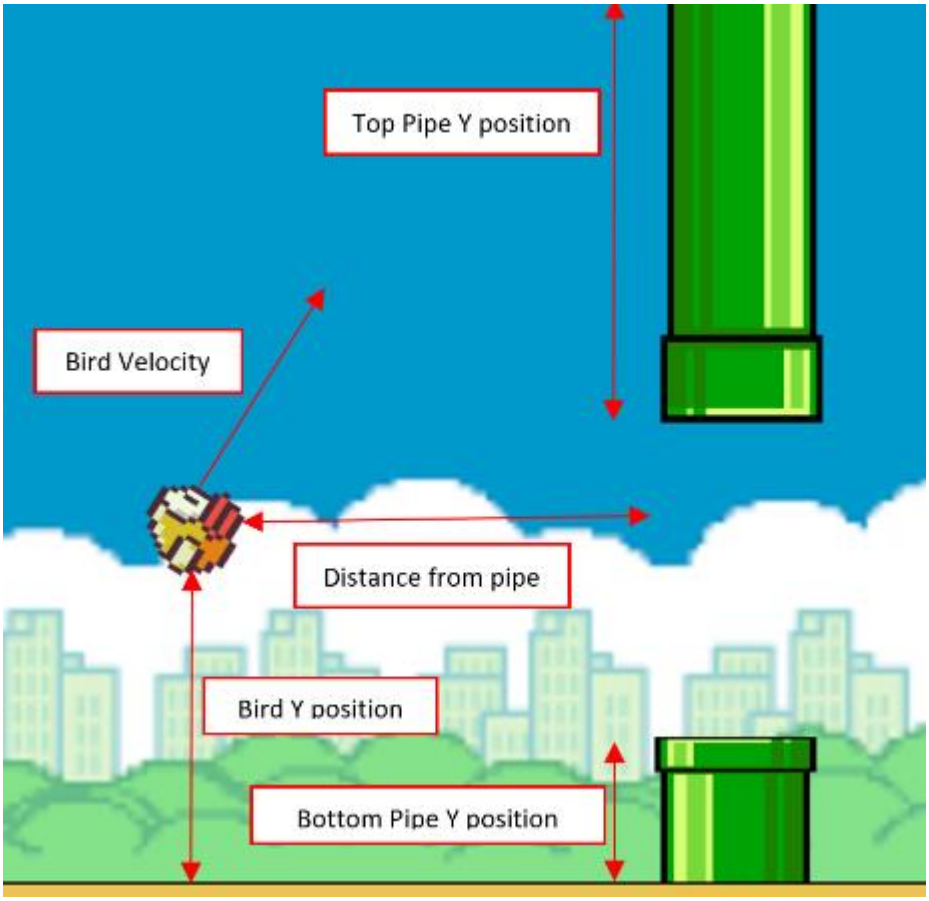


Fig 5.1: concept of game

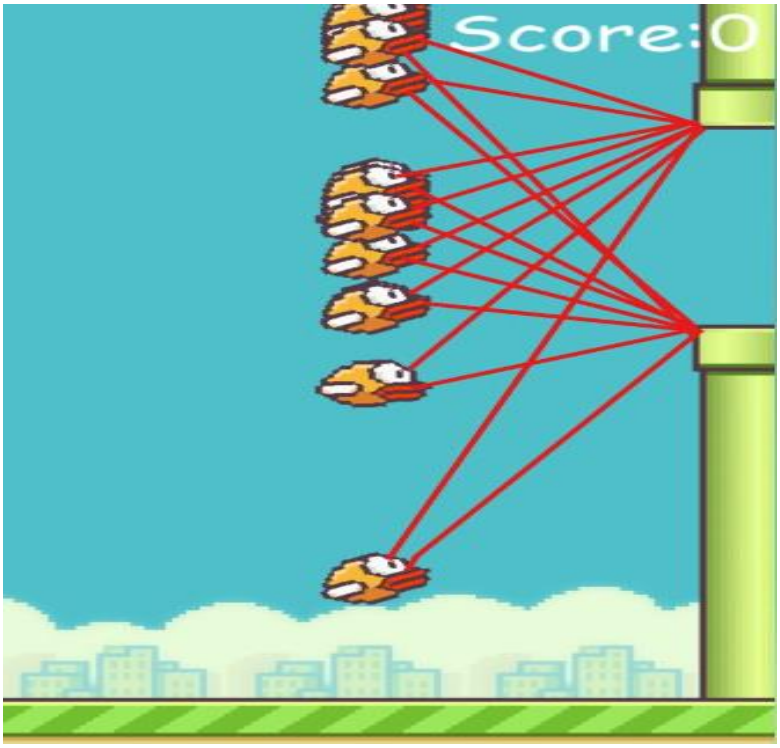
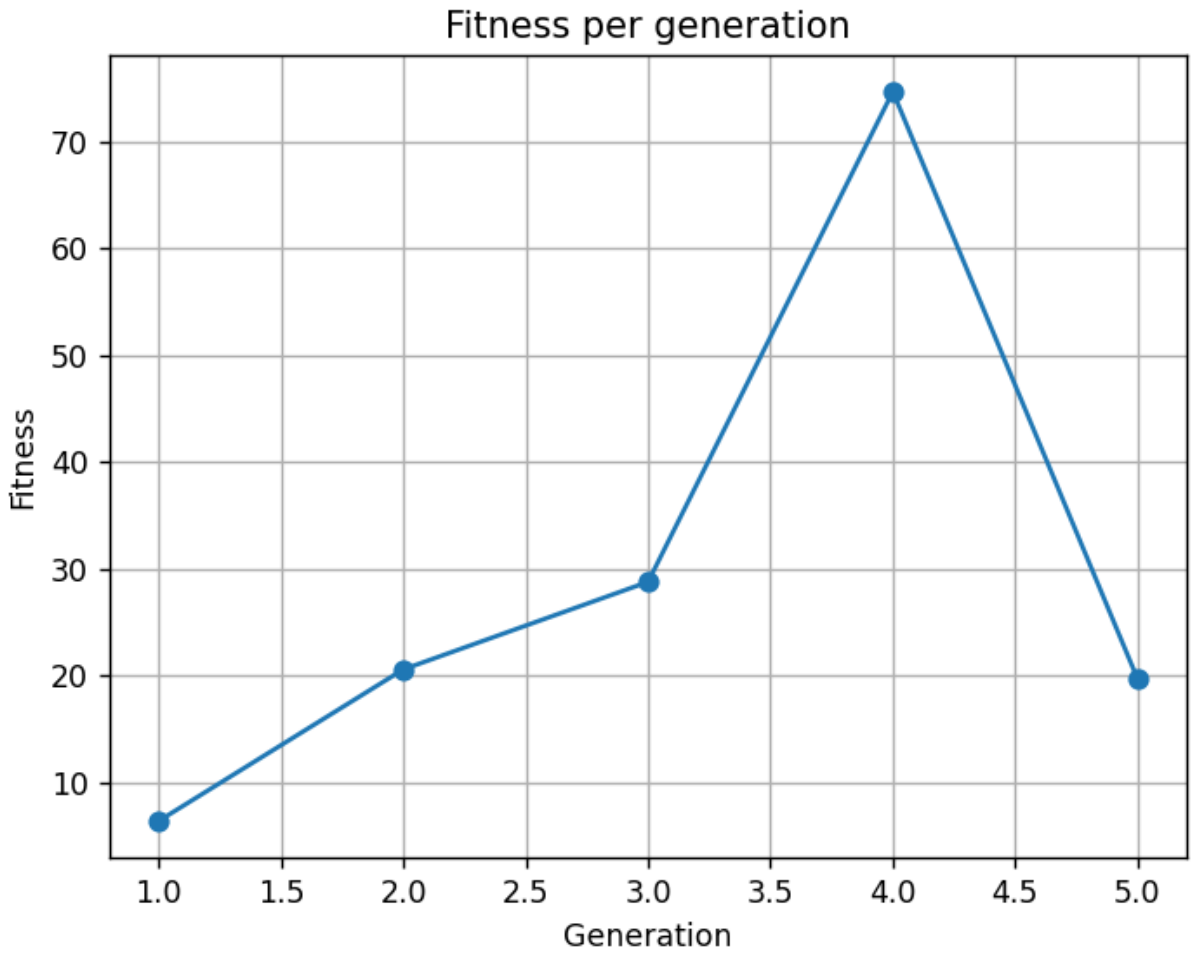


Fig 5.2: working of game

Prediction:


Fitness per generation
Fitness
70
60
50
40
30
20
10
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Generation

5.2 RESULT

Positive outcomes were obtained when the Flappy Bird game was subjected to the NEAT algorithm. The evolving neural networks showed notable improvements in gameplay performance throughout the course of subsequent generations. While early generations crashed often and behaved erratically, networks started to guide the bird through pipes more reliably by the 50th generation. The top-performing networks reached high scores by generation 100, on par with expert human players. Neural networks that could adjust to the game's dynamics were produced as a consequence of the evolution process being successfully driven by the fitness function, which was dependent on flight length and distance. These findings validate NEAT's capacity to learn and apply complicated control tactics in an intense gaming environment on its own, underscoring the possibility of wider uses in real-time decision-making tasks.

CHAPTER 6
CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION

The efficacy of the NEAT method in developing neural networks that can handle intricate control tasks is demonstrated by its application to the Flappy Bird game. NEAT effectively created techniques for the bird to navigate over pipe gaps by gradually increasing network complexity, optimizing for longer flight duration and distance. Strong performance is demonstrated by the evolved networks, highlighting NEAT's capacity to resolve dynamic real-time decision-making issues with little assistance from humans. The promise of neuroevolution approaches in gaming AI development and other fields that require adaptive control solutions is demonstrated by this effort. To further confirm NEAT's adaptability and effectiveness, further research should include improving evolutionary parameters, optimizing input encoding, and applying NEAT to more challenging games or real-world situations.

6.2 FUTURE ENHANCEMENT

In order to further test the evolving neural networks, future improvements to the Flappy Bird game that employ the NEAT algorithm could concentrate on making the gaming environment more varied and complicated. The strength and flexibility of the AI might be enhanced by adding dynamic impediments, altering pipe speeds, and adjusting gravity effects. Furthermore, the AI could be able to balance several performance indicators, such scoring and collision avoidance efficiency, by implementing multi-objective optimization. To improve the AI's performance even further, real-time learning features that allow neural networks to change and adapt while a player is playing might be implemented. The application of these developed methods to related games may be made possible by investigating transfer learning .

APPENDIX

SOURCE CODE:

```
import pygame
import neat
import os
import time
import random
import matplotlib.pyplot as plt

pygame.font.init()

WIN_WIDTH = 500
WIN_HEIGHT = 800
BIRD_IMGS = [pygame.transform.scale2x(pygame.image.load(os.path.join("imgs", "bird1.png"))),
             pygame.transform.scale2x(pygame.image.load(os.path.join("imgs", "bird2.png"))),
             pygame.transform.scale2x(pygame.image.load(os.path.join("imgs", "bird3.png")))]
PIPE_IMG = pygame.transform.scale2x(pygame.image.load(os.path.join("imgs", "pipe.png")))
BASE_IMG = pygame.transform.scale2x(pygame.image.load(os.path.join("imgs", "base.png")))
BG_IMG = pygame.transform.scale2x(pygame.image.load(os.path.join("imgs", "bg.png")))
STAT_FONT = pygame.font.SysFont("comicsans", 50)

WIN = pygame.display.set_mode((WIN_WIDTH, WIN_HEIGHT))
```

```
class Bird:
    IMGS = BIRD_IMGS
    MAX_ROTATION = 25
    ROT_VEL = 20
    ANIMATION_TIME = 5

    def __init__(self, x, y):
        self.x = x
        self.y = y
        self.tilt = 0
        self.tick_count = 0
        self.vel = 0
        self.height = self.y
        self.img_count = 0
        self.img = self.IMGS[0]

    def jump(self):
        self.vel = -10.5
        self.tick_count = 0
        self.height = self.y

    def move(self):
        self.tick_count += 1
        d = self.vel * self.tick_count + 1.5 * self.tick_count ** 2

        if d >= 16:
            d - 16
        if d < 0:
```

```
            d -= 2
        self.y = self.y + d
        if d < 0 or self.y < self.height + 50:
            if self.tilt < self.MAX_ROTATION:
                self.tilt = self.MAX_ROTATION

        else:
            if self.tilt > -90:
                self.tilt -= self.ROT_VEL

    def draw(self, win):
        self.img_count += 1

        if self.img_count <= self.ANIMATION_TIME:
            self.img = self.IMGS[0]
        elif self.img_count <= self.ANIMATION_TIME * 2:
            self.img = self.IMGS[1]
        elif self.img_count <= self.ANIMATION_TIME * 3:
            self.img = self.IMGS[2]
        elif self.img_count <= self.ANIMATION_TIME * 4:
            self.img = self.IMGS[1]
        elif self.img_count == self.ANIMATION_TIME * 4 + 1:
            self.img = self.IMGS[0]
            self.img_count = 0
        if self.tilt <= -80:
            self.img = self.IMGS[1]
            self.img_count = self.ANIMATION_TIME * 2
```

```
        rotated_image = pygame.transform.rotate(self.img, self.tilt)
        new_rect = rotated_image.get_rect(center=self.img.get_rect(topleft=(self.x, self.y)).center)
        win.blit(rotated_image, new_rect.topleft)

    def get_mask(self):
        return pygame.mask.from_surface(self.img)



class Pipe:
    GAP = 200
    VEL = 5

    def __init__(self, x):
        self.x = x
        self.height = 0

        self.top = 0
        self.bottom = 0
        self.PIPE_TOP = pygame.transform.flip(PIPE_IMG, False, True)
        self.PIPE_BOTTOM = PIPE_IMG

        self.passed = False
        self.set_height()

    def set_height(self):
        self.height = random.randrange(50, 450)
        self.top = self.height - self.PIPE_TOP.get_height()
```

```
        self.bottom = self.height + self.GAP

    def move(self):
        self.x -= self.VEL

    def draw(self, win):
        win.blit(self.PIPE_TOP, (self.x, self.top))
        win.blit(self.PIPE_BOTTOM, (self.x, self.bottom))

    def collide(self, bird):
        bird_mask = bird.get_mask()
        top_mask = pygame.mask.from_surface(self.PIPE_TOP)
        bottom_mask = pygame.mask.from_surface(self.PIPE_BOTTOM)
        top_offset = (self.x - bird.x, self.top - round(bird.y))
        bottom_offset = (self.x - bird.x, self.bottom - round(bird.y))

        b_point = bird_mask.overlap(bottom_mask, bottom_offset)
        t_point = bird_mask.overlap(top_mask, top_offset)

        if b_point or t_point:
            return True

        return False


class Base:
    VEL = 5
    WIDTH = BASE_IMG.get_width()
```

```
    IMG = BASE_IMG

    def __init__(self, y):
        self.y = y
        self.x1 = 0
        self.x2 = self.WIDTH

    def move(self):
        self.x1 -= self.VEL
        self.x2 -= self.VEL
        if self.x1 + self.WIDTH < 0:
            self.x1 = self.x2 + self.WIDTH
        if self.x2 + self.WIDTH < 0:
            self.x2 = self.x1 + self.WIDTH

    def draw(self, win):
        win.blit(self.IMG, (self.x1, self.y))
        win.blit(self.IMG, (self.x2, self.y))


def draw_window(win, birds, pipes, base, score):
    win.blit(BG_IMG, (0, 0))
    for pipe in pipes:
        pipe.draw(win)
    text = STAT_FONT.render("Score:" + str(score), 1, (255, 255, 255))
    win.blit(text, (WIN_WIDTH - 10 - text.get_width(), 10))
    base.draw(win)
    for bird in birds:
```

```
        bird.draw(win)
    pygame.display.update()


def eval_genomes(genomes, config):
    birds = []
    nets = []
    ge = []
    for _, g in genomes:
        net = neat.nn.FeedForwardNetwork.create(g, config)
        nets.append(net)
        birds.append(Bird(230, 350))
        g.fitness = 0
        ge.append(g)

    base = Base(730)
    pipes = [Pipe(600)]
    win = pygame.display.set_mode((WIN_WIDTH, WIN_HEIGHT))
    run = True
    clock = pygame.time.Clock()
    score = 0

    while run:
        clock.tick(20)
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                run = False
                pygame.quit()
```

```
            quit()
    pipe_ind = 0
    if len(birds) > 0:
        if len(pipes) > 1 and birds[0].x > pipes[0].x + pipes[0].PIPE_TOP.get_width():
            pipe_ind = 1
    else:
        run = False
        break

    for x, bird in enumerate(birds):
        bird.move()
        ge[x].fitness += 0.1
        output = nets[x].activate(
            (bird.y, abs(bird.y - pipes[pipe_ind].height), abs(bird.y - pipes[pipe_ind].bottom)))
        if output[0] > 0.5:
            bird.jump()
    add_pipe = False
    rem = []
    for pipe in pipes:
        for x, bird in enumerate(birds):
            if pipe.collide(bird):
                ge[x].fitness -= 1
                birds.pop(x)
                nets.pop(x)
                ge.pop(x)

            if not pipe.passed and pipe.x < bird.x:
                pipe.passed = True
```

```
                add_pipe = True
            if pipe.x + pipe.PIPE_TOP.get_width() < 0:
                rem.append(pipe)
            pipe.move()
        if add_pipe:
            score += 1
            for g in ge:
                g.fitness += 5
            pipes.append(Pipe(random.randrange(600, 800)))

        for r in rem:
            pipes.remove(r)

        for x, bird in enumerate(birds):
            if bird.y + bird.img.get_height() >= 730 or bird.y < 0:
                birds.pop(x)
                nets.pop(x)
                ge.pop(x)

        base.move()
        draw_window(win, birds, pipes, base, score)

    return True  # Added a return statement for the plot


def run(config_path):
    config = neat.config.Config(neat.DefaultGenome, neat.DefaultReproduction,
                                neat.DefaultSpeciesSet, neat.DefaultStagnation,
```

```
                    config_path)

    p = neat.Population(config)
    p.add_reporter(neat.StdOutReporter(True))
    stats = neat.StatisticsReporter()
    p.add_reporter(stats)

    winner = p.run(eval_genomes, 5)
    # Plotting the graph
    generations = range(1, len(stats.most_fit_genomes) + 1)
    time_taken = [c.fitness for c in stats.most_fit_genomes]
    plt.plot(generations, time_taken, marker='o')
    plt.title('Fitness per generation')
    plt.xlabel('Generation')
    plt.ylabel('Fitness')
    plt.grid(True)
    plt.show()


if __name__ == "__main__":
    local_dir = os.path.dirname(__file__)
    config_path = os.path.join(local_dir, 'config-feedforward.txt')
    run(config_path)
```

REFERENCES

[1] Cordeiro, Matheus G., Paulo Bruno S. Serafim, Yuri Lenon B. Nogueira, Creto A. Vidal, and Joaquim B. Cavalcante Neto. "A minimal training strategy to play flappy bird indefinitely with NEAT." In *2019 18th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pp. 21-28. IEEE, 2019.

[2] Jogekar, Ravindra, Mayur Kadu, Bhavarth Patel, Jaskaran Kaur Rai, Tejas Kawale, Karan Sapolia, and Priyanka Thakur. "NEAT Wings: Algorithmic Strategy for Intelligent Flappy Bird Navigation." In 2024 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS), pp. 1-7. IEEE, 2024.

[3] Asha, V., Arpana Prasad, C. R. Vishwanath, K. Madava Raj, AR Manoj Kumar, and N. Leelavathi. "Designing A Popular Game Framework Using Neat A Genetic Algorithms." In 2023 International Conference on Artificial Intelligence and Applications (ICAIA) Alliance Technology Conference (ATCON-1), pp. 1-5. IEEE, 2023.

[4] Selvan, Jerin Paul, and Pravin S. Game. "Playing a 2D game indefinitely using NEAT and reinforcement learning." arXiv preprint arXiv:2207.14140 (2022).

[5] Thurler, Leonardo, José Montes, Rodrigo Veloso, Aline Paes, and Esteban Clua. "Ai game agents based on evolutionary search and (deep) reinforcement learning: A practical analysis with flappy bird." In Entertainment Computing–ICEC 2021: 20th IFIP TC 14 International Conference, ICEC 2021, Coimbra, Portugal, November 2–5, 2021, Proceedings 20, pp. 196-208. Springer International Publishing, 2021.