khazanah informatika

Jurnal Ilmu Komputer dan Informatika

Backtracking and k-Nearest Neighbour for Non-Player Character to Balance Opponent in a Turn-Based Role Playing Game of Anagram

Yosa Aditya Prakosa*, Alfa Faridh Suni
Departement of Electrical Engineering, Faculty of Engineering, Universitas Negeri Semarang
*yosaadityap@students.unnes.ac.id

Abstract-Anagram is a turn-based role-playing game where two players construct words by arranging given letters. A significant aspect of playing a game is the challenge. A good challenge comes from an opponent with a close ability. In a two-player game like Anagram, the second player can be a nonhuman player called Non-Playable Character (NPC). A balanced game is more engaging. Therefore, it is imperative to insert artificial intelligence (AI) into an NPC to make it possess a balance ability. This study investigates the AI algorithm that is the most appropriate to make a balance NPC for Anagram games. We tested three scenarios: Descending AI, Random AI, and AI with k-Nearest Neighbour (k-NN). Descending AI gets an Anagram solution by selecting a word with the highest score from all possible answers. Random AI picks a word randomly from the possible answers, while AI with k-NN chooses a word closest to one of the human players. The results show that Descending AI is the best algorithm to make the strongest NPC, which always gets the highest score, followed by Random AI and AI with k-NN. However, AI with the k-NN algorithm makes the constructed NPC has the highest number of turns at an average of 18, while Descending AI gets 14 turns and Random AI has 15 turns. Looking at the remaining lives at the end of the game, AI with k-NN makes the NPC has 25 lives left, while Descending AI has 59 lives, and Random AI has 48 lives. Less remaining lives suggest that NPC containing AI with the k-NN algorithm matches closer to the human player and therefore is more suitable for Anagram NPC.

Keywords: anagram, artificial intelligence, game, non-player character

Article info: *submitted December 27, 2021, revised May 27, 2022, accepted June 11, 2022*

1. Introduction

One of the rapidly growing computer software technologies is gaming. There are many types of games that have been developed, one of which is word puzzle games or anagrams. An anagram is a game that forms a word by rearranging the letters of another word [1]. The main purpose of anagrams is to form correct words according to the spelling system of a language [2]. One of the innovations that have been developed in anagram games is the gameplay, where players must be able to compose anagram and sub-anagram words to attack enemies [3]. In the research by Kuswardyan et al. [3], anagrams are used as a battle system in turn-based role-playing games (RPGs). Turn-based RPG is a combat system in which attacks from players or enemies have been carried out alternately [4], waiting for input from the player [5]. In turn-based RPGs, the enemy can be a Non-Player Character (NPC) that performs activities automatically [6]. NPCs that have artificial intelligence (AI) can increase user engagement in playing games [6]. Artificial intelligence is a technology that can make decisions by analyzing and using the data available in the system [7].

In the anagram game by [3], NPCs don't have artificial intelligence yet. NPCs that do not have AI will be easily defeated by players, making the game less challenging [6]. On the other hand, NPCs who have an invincible AI will make players desperate so they are not interested in playing the game [8]. Therefore, intelligent AI is needed for NPCs to be able to keep up with players [9]. Several methods can be used to provide intelligence to NPCs, for example, fuzzy [5] and rule base [7]. The rule base method applies the rules according to the situation and actions [10]. An example of applying the rule base is to divide the behavior of NPCs into 3 types, namely descending AI, random AI, and k-NN AI. Descending AI answers

the anagram with the highest value word. Random AI answers with a random word. Meanwhile, k-NN AI answers with words calculated by the k-nearest neighbor (k-NN) classification algorithm. The k-NN algorithm has the opportunity to be an option because it is an adaptive algorithm and can adapt to the players' answers [11]. The k-NN algorithm has better accuracy than the nave Bayes method.

This paper discusses the results of research on the most suitable rule base to be applied to anagram games with the turn-based role-playing game (RPG) genre. The match in question is the ability of the NPC to provide a balanced resistance to the player.

There have been many studies related to the addition of intelligence to NPCs. Research [3] applies a backtracking algorithm to search for anagrams. They proved that the backtracking algorithm was able to find anagram words well.

Susanto et al. compared 2 classification methods that have high accuracy, namely k-NN and Naïve Bayes [11]. They stated that the k-NN method has higher accuracy than Naïve Bayes. The accuracy of k -NN is at 93.17% while the accuracy of Naïve Bayes is at the level of 78.38%. Higher accuracy is expected to make NPCs more adaptive and dynamic so that they become a balanced opponent for players.

2. Methods

a. Non-Player Character (NPC)

A non-Player Character is an entity in the game that is controlled automatically by a computer, not by humans [12]. NPCs can be friends, foes, or neutrals [13]. NPCs are expected to behave intelligently like humans. He can respond to answers according to the actions of the original player. Intelligent NPCs can be obtained by adding artificial intelligence (AI) to characters [14]. The use of AI on NPCs is done by giving certain algorithms according to the expected intelligent behavior [15]. In anagram games, AI is made in such a way as to create intelligent NPCs in choosing anagram words, so that players feel challenged in playing the game.

b. Implementation of AI on NPCs

Artificial Intelligence in anagram games is applied to be able to search and choose good anagram words so that they are not easily defeated by players. The flow of AI implementation on NPCs can be seen in Figure 1.

c. Word scramble

The first step when the game starts is word shuffling. The scrambled word is a combination of all letters with vowels. It is intended that in every word scramble there are vowels in it. The number of anagram letters used in this study was 6 letters.

d. Composing and matching words with Backtracking

The second step after the words have been scrambled is to generate or arrange words. At this stage, a backtracking algorithm is applied (see Figure 2). If the word is not available it will change to another word (backtrack). Words are compiled from the existing letters and then searched in a database of available dictionaries. The words available in the dictionary are added to the list to calculate the score.

c. Scoring

Each word in the list is scored by converting each letter used to a number. Each letter has a different score (see the scoring guide in Figure 3). The score is between 1 and 10. Frequently used characters are low and rarely used are high. The vowels and letters l, n, r, s, and t have a value of 1, while the letters q and z are worth 10. The results of the score calculation for each word in the list are stored in the score list.

d. Word Selection

Based on the list of scores formed, the process of selecting words is carried out. This study tested 3-word selection algorithms, namely Descending AI, Random AI, and k-NN.

1) Descending AI
   The Descending AI method selects the word from the list that has the highest score. The score list is sorted in descending order and then takes the word that is in the topmost position.
2) Random AI
   The Random AI method selects words from the list randomly. Although it is not necessary, in this research, the process of calculating the score and the process of sorting the data is still carried out. Then the word is selected using random.next() command to ensure a random word selection. The word selected may have the highest score and may have the lowest score or in between.
3) k-NN
   The application of the k-NN method requires observations to determine the correct value of k. The value of k shows the number of neighbors that become the benchmark for selecting groups for new data. The new data is placed in groups based on the distance between the data and other data already in the group. The distance calculation is done by equation (1) which is none other than Euclidean distance [6].

$$d_{ij} = \sqrt{\sum_{k=1}^{n} (a_{ik} - a_{jk})^2} \quad (1)$$

In equation (1), n is the number of attributes and is the distance between data. The smaller the distance value, the more similar the two data. The data that is calculated

for the proximity value is the player's answer score and the available score. The player's answer is calculated for its proximity to all available potential answers, then the answer that has the smallest closeness to the player's answer is selected.

e. Assault

Assault is the process of reducing the live score of players and NPCs. Each opponent is given a live score of 100 at the start of the game. If one of them manages to make a word with a score of 15 for example, then the live score of the enemy is reduced by 15 points, as a form of damage to the opponent. The higher the player's score when answering the anagram, the more damage the opponent takes.

Players get the opportunity to answer anagrams many times in a turn. Each time an answer is given, the opponent takes damage and the live score is reduced. The game ends when the live score of the player or NPC is exhausted (less than or equal to zero).

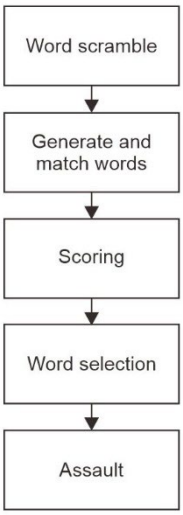

Figure 1. Implementation of AI on NPCs

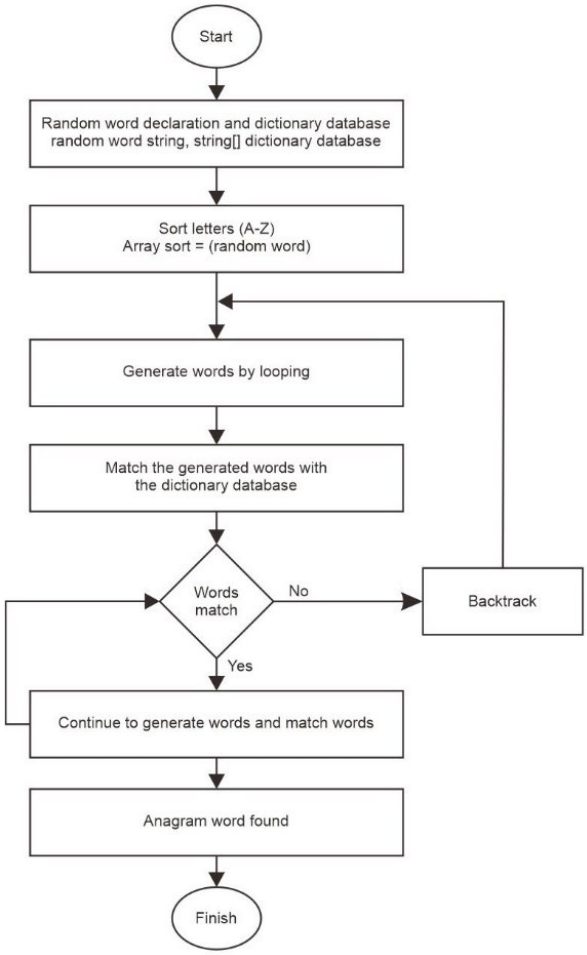

Figure 2. Flowchart generates and matches words

Figure 3. Anagram letter conversion guide

3. Result and Discussion

This section describes the results of game simulations, game testing, and performance comparisons of Descending AI, Random AI, and AI k-NN algorithms.

a. Game simulation

For this research, we designed the game and developed it using Unity 3D. Figure 4 shows a screenshot of the running game, where human players (on the left) are playing against NPCs (on the right). On the display, various display boxes show the status of the game being played. At the bottom is a time display that shows how long one anagram step was answered. The left and right sections show the words that have been answered by players and NPCs. A bit in the middle of the live performance of the two players. At the bottom right there is a display of the number of turns that have been played.

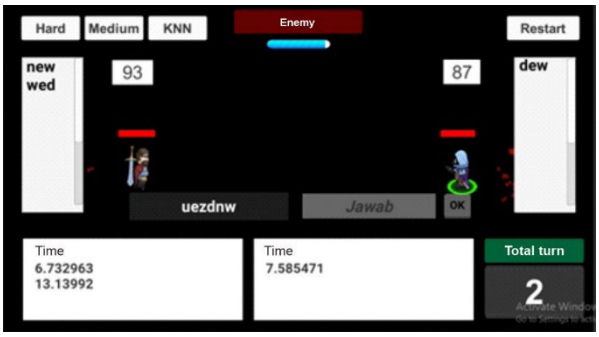

Figure 4. Display when the game is played

b. Game testing

Data was collected by asking 10 to play the game. The ten people consisted of 5 ordinary gamers and 5 teachers who had good English skills. Each person is asked to play the game 3 times, each against an NPC controlled by Descending AI, Random AI, and AI k-NN algorithms.

c. Game with Descending AI

The results of the game simulation between players and NPCs controlled by the Descending AI algorithm are shown in table 1. The table shows the time it takes for the NPC to answer the word, the number of turns required in the game, and the remaining live belonging to the NPC.

Table 1. NPC Descending AI game data measurement

Player	Answering Time	Total Turn	Life Remaining
Gamer 1	5.2 s	15 *turn*	67
Gamer 2	6.1 s	14 *turn*	73
Gamer 3	5.8 s	13 *turn*	65
Gamer 4	5 s	20 *turn*	66
Gamer 5	6.6 s	13 *turn*	66
Teacher 1	6.4 s	13 *turn*	49
Teacher 2	7.6 s	13 *turn*	52
Teacher 3	7.3 s	18 *turn*	53
Teacher 4	6.8 s	10 *turn*	48
Teacher 5	5.5 s	15 *turn*	56

Based on the data presented in table 1, the average length of the answer is calculated, the average number of turns, and the average remaining live for NPCs. The calculation results are presented in table 2.

Table 2. Average NPC Descending AI game data

Player	Average Answering Time	Average Total Turns	Average NPC Life Remaining
Gamer	5.74 s	15 turn	67
Teacher	6.72 s	14 turn	52

Table 2 shows that NPCs need more time when playing against teachers, which on average takes 6.72 seconds compared to 5.74 seconds when playing against ordinary gamers. NPCs get fewer turns, which is 14 turns against teachers compared to 15 turns against regular gamers. Furthermore, the remaining live NPCs when playing against teachers are a smaller number than the remaining when against ordinary gamers.

d. Game with Random AI

The results of the game simulation between human players against NPCs who work using the Random AI algorithm are shown in table 3. Then the average data is presented in table 4.

Table 3. Random AI NPC game data measurement

Player	Answering Time	Total Turn	Life Remaining
Gamer 1	6.5 s	14 *turn*	57
Gamer 2	7.4 s	18 *turn*	43
Gamer 3	8.3 s	13 *turn*	49
Gamer 4	7.4 s	14 *turn*	65
Gamer 5	6.6 s	15 *turn*	76
Teacher 1	6.4 s	16 *turn*	29
Teacher 2	6.3 s	19 *turn*	19
Teacher 3	6.9 s	14 *turn*	47
Teacher 4	6.8 s	14 *turn*	58
Teacher 5	7.3 s	17 *turn*	43

Table 4. Random AI NPC game data average

Player	Average Answering Time	Average Total Turns	Average NPC Life Remaining
Gamer	7.24 s	15 *turn*	58
Teacher	6.74 s	16 *turn*	39

NPCs with Random AI algorithms take longer to answer when facing regular gamers. The number of turns

obtained by NPCs is 16 turns when facing teachers, which is more than the turns obtained when facing ordinary gamers. Furthermore, the life that NPCs have when the game is over is 39 when facing teachers, which is smaller than the rest when facing ordinary gamers.

e. Games with k-NN

Game simulations between human players and NPCs run with the k-NN algorithm are shown in table 5. Game data shows the average time to answer words, the number of turns, and the remaining lives of the NPCs at the end of the game. On two occasions playing against the teacher, it was seen that the remaining live was 0 which meant that the NPC had lost.

Table 5. NPC game data measurement with k-NN

Player	Answering Time	Total Turn	Life Remaining
Gamer 1	7.4 s	20 *turn*	54
Gamer 2	5.8 s	21 *turn*	31
Gamer 3	6.6 s	19 *turn*	28
Gamer 4	7.1 s	15 *turn*	33
Gamer 5	5.3 s	20 *turn*	37
Teacher 1	6.5 s	15 *turn*	0
Teacher 2	6.2 s	22 *turn*	22
Teacher 3	6.9 s	19 *turn*	16
Teacher 4	5.2 s	17 *turn*	0
Teacher 5	6.7 s	17 *turn*	26

Furthermore, the data in table 5 are averaged and presented in table 6. The calculation results show that the teacher as a player is a formidable opponent for the NPC. In general, NPCs need a little more time to respond when playing against teachers. The number of turns obtained is less when facing the teacher. Even the live remaining NPC is smaller when playing against the teacher.

Table 6. Average NPC game data with k-NN

Player	Average Answering Time	Average Total Turns	Average NPC Life Remaining
Gamer	6.44 s	19 *turn*	37
Teacher	6.3 s	18 *turn*	13

f. NPC performance for various algorithms

The data in table 2, table 4, and table 6 become the basis for comparing the performance of NPCs when controlled by various algorithms. The three tables present the average NPC game data in the form of average answering time, the average number of turns, and the average remaining live when NPCs are controlled by Descending AI, Random AI, and AI with k-NN algorithms.

Table 7. Comparison of Answering Time

Player	NPC Answering Time		
	Descending AI	Random AI	AI k-NN
Gamer	5.74 s	7.24 s	6.44 s
Teacher	6.72 s	6.74 s	6.3 s
Average	6.23 s	6.99 s	6.37 s

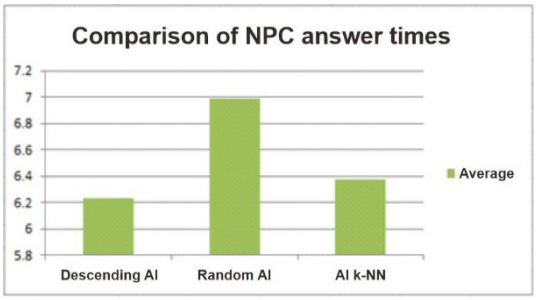

Figure 5. Comparison Graph of Answering Time Descending AI, Random AI, and AI k-NN

Table 7 shows the average data for answering NPCs when playing against gamers and teachers. The average answer time is presented visually in Figure 5. It can be seen that the fastest answering time is given by the NPC when controlled by the Descending AI algorithm, which is 6.2 seconds and the longest time is obtained when controlled by the Random AI algorithm, which is 6.99 seconds. The difference in the speed of answering is not significant enough when viewed from the variation in the data presented in Table 1 and Table 3.

Table 8. Comparison of Total Turn

Player	Total Turn		
	Descending AI	Random AI	AI k-NN
Gamer	15 *turn*	15 *turn*	19 *turn*
Teacher	14 *turn*	16 *turn*	18 *turn*
Average	14 *turn*	15 *turn*	18 *turn*

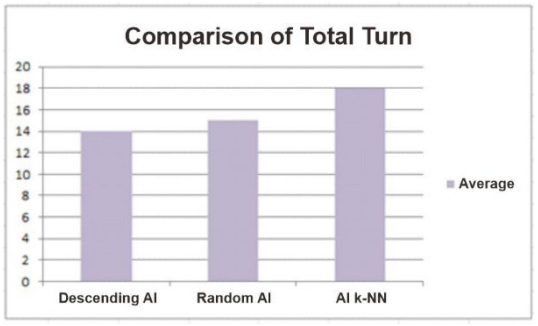

Figure 6. Comparison Graph of Descending AI Turns, Random AI, and AI k-NN

Table 8 presents data on the average number of turns that NPCs get when playing against gamers and teachers. The average number of turns is then presented in Figure 6. It can be seen that the highest number of turns is obtained by NPCs when controlled using the k-NN algorithm, which is 18 turns. While the least number of turns is obtained when the NPC is controlled using the Descending AI algorithm, which is 14 turns. This difference is significant when viewed from the variation of the data in table 1 and table 5.

Table 9. NPC Life Remaining Comparison

Player	NPC Life Remaining		
	Descending AI	Random AI	AI k-NN
Gamer	67	58	37
Teacher	52	39	13
Average	59	48	25

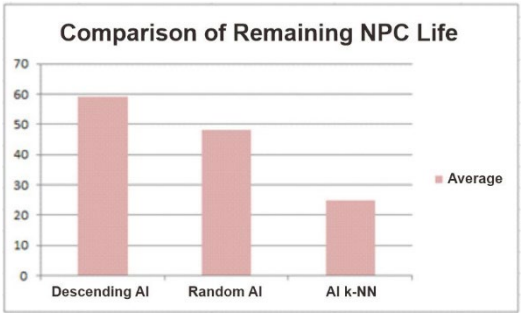

Figure 7. Comparison Graph of NPC Descending AI, Random AI, and AI k-NN

Table 12 shows the average remaining live that NPCs have at the end of the game when facing gamers and teachers. The average game data against the two types of players is presented graphically in Figure 7. It can be seen that the NPCs with Descending AI dominated the game and only lost a few lives with the remainder at 59. The NPCs with AI k-NN had an average remaining live of Rp. 25 which shows that NPCs can have a chance of winning but not very much. The data in table 7 confirms that NPCs with AI k-NN lost after losing all lives.

Based on the test data, it shows that NPCs with Descending AI cannot be defeated and tend to dominate the game. The game ends quickly, with an average of 14 turns with an average remaining live of 59. Each time answering an NPC with Descending AI only takes an average of 6.2 seconds.

The test results show that NPCs with Random AI still dominate the game relatively. The game ends in 15 turns with an average remaining live of 48. Test data shows this NPC has never been beaten even though it took the longest to come up with an answer. NPCs with k-NN do not dominate the game even though statistically still win more games. Games against NPCs with k-NN averaged over 18 turns with an average remaining live of 25.

The ideal NPC can keep up with players [16]. The NPCs worth implementing aren't the ones that dominate the game and can't be beaten. A good NPC is not an easy one to beat in every game. Therefore, NPC with k-NN in the Anagram game is the best choice among the AI algorithms tested in this study.

4. Conclusion

Results and discussions show that the NPCs with Descending AI dominate the play. The NPC ends the game in an average of 14 turns and saves 59 remaining lives. The less dominant NPCs are those controlled by k-NN. The latter finishes the game in 18 turns with 25 remaining lives. NPCs with k-NN can be defeated by human players, while NPCs with Descending AI and Random AI are unbeatable. Therefore, k-NN is the best choice of the three algorithms tested in this study as controller of the Anagram NPC.

Reference

[1] M. A. Rahman, "The Effectiveness of Anagram on Students ' Vocabulary Size," *J. IAIN Palangkaraya*, no. December, pp. 129–139, 2016.

[2] C. T. Panagiotakopoulos and M. E. Sarris, "'Playing with words': Effects of an anagram solving game-like application for primary education students," *Int. Educ. Stud.*, vol. 6, no. 2, pp. 110–126, 2013, doi: 10.5539/ies.v6n2p110.

[3] I. Kuswardayan, R. Rahman, and N. Suciati, "Design and Implementation of Random Word Generator using Backtracking Algorithm for Gameplay in Ambrosia Game," *Int. J. Comput. Appl.*, vol. 158, no. 6, pp. 27–30, 2017, doi: 10.5120/ijca2017912822.

[4] R. D. Pertiwi, "Planning of Making the Introducing Edifice Game," no. December, pp. 12–13, 2014.

[5] D. Ratanajaya and H. A. Wibawa, "Implementasi Kecerdasan Buatan dalam Menentukan Aksi Karakter pada Game RPG dengan Logika Fuzzy Tsukamoto," *Khazanah Inform. J. Ilmu Komput. dan Inform.*, vol. 4, no. 2, p. 82, 2018, doi: 10.23917/khif.v4i2.6744.

[6] M. I. A. Putera and D. H. Murti, "Peningkatan Kecerdasan Computer Player Pada Game Pertarungan Berbasis K-Nearest Neighbor Berbobot," *JUTI J. Ilm. Teknol. Inf.*, vol. 16, no. 1, p. 90, 2018, doi: 10.12962/j24068535.v16i1.a710.

[7] M. Abdi, D. Herumurti, and I. Kuswardayan, "Analisis Perbandingan Kecerdasan Buatan pada Computer Player dalam Mengambil Keputusan

pada Game Battle RPG," *JUTI J. Ilm. Teknol. Inf.*, vol. 15, no. 2, p. 226, 2017, doi: 10.12962/j24068535.v15i2.a671.

[8] V. Vorachart and H. Takagi, "Evolving fuzzy logic rule-based game player model for game development," *Int. J. Innov. Comput. Inf. Control*, vol. 13, no. 6, pp. 1941–1951, 2017, doi: 10.24507/ijicic.13.06.1941.

[9] G. Grund, P. M. Nilsson, M. Larsson, O. Olsson, T. Foughman, and V. Gustafsson, "Realistic NPCs in Video Games Using Different AI Approaches," no. June, 2016.

[10] M. Gwon, E. Goh, C. Sohn, and A. R. B. Ai, "The VR Trip Simulator with Multi Networking of Rule-based Model," vol. 13, no. 22, pp. 15754–15757, 2018.

[11] E. bagus Susanto, M. K. Triawan adi cahyanto, and pd M. Reni umilasari S, "Analisis Perbandingan Algoritma Naive Bayes Dan k-NN Untuk Klasifikasi Multi Dataset," no. 1410651097, 2019.

[12] H. Warpefelt, *The Non-Player Character: Exploring the believability of NPC presentation and behavior*, no. 16. 2016.

[13] R. D. Agustin, "Komponen Konsep dan Desain Game," *J. Ilm. Teknol. Inf. Terap.*, vol. III, no. 2, 2017.

[14] E. Siswanto and A. F. Suni, "Aksi Penyerangan Non-Player Character (NPC) Menggunakan Metode Naïve Bayes Pada Shooter Game," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 8, no. 6, 2021, doi: 10.25126/jtiik.202183804.

[15] F. B. Adi, M. Hariadi, and I. K. E. Purnama, "Simulasi Perilaku Tempur Pada Sekumpulan NPC Berbasis Boid," 2016.

[16] M. Kopel and T. Hajas, "Implementing AI for Non-player Characters in 3D Video Games," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10751 LNAI, no. January 2018, pp. 610–619, 2018, doi: 10.1007/978-3-319-75417-8_57.