CEPAT Journal of Computer Engineering: Progress, Application and Technology
Vol. 2, No. 1, February 2023, pp. 9-17
ISSN: 2963-6728, DOI: https://doi.org/10.25124/cepat.v2i01.5779

Design an enemy non-player character in maze game using finite state machine algorithm

Irfan Ainul Afif Refnaldi[1], Purba Daru Kusuma[1], Ashri Dinimaharawati[1]
[1]Department of Computer Engineering, School of Electrical Engineering, Telkom University, Indonesia

Article Info

Article history:

Received January 26, 2023
Revised February 16, 2023
Accepted February 20, 2023

Keywords:

Game
Educational Game
Non-Player Character
Finite State Machine

ABSTRACT

Educational games are very suitable for learning for children because the gameplay is not so heavy and is based on the same conditions as everyday life, which certainly does not contain bad elements that are inappropriate for children. The output of educational games that will be used for learning must really be considered because the age of kindergarten children is the age at which children's character and manners are formed. In this research, the writer will develop a game with a labyrinth concept which carries the theme "Caring for the Environment". The author adds the Non-Player Character (NPC) feature in the game which requires behavior design using the Finite State Machine algorithm with three working principles State, Event (happening), Action. The result of this research is that all the features in the developed game design have been implemented and function properly, especially the finite state machine method which is applied to NPC behavior. NPC behavior shifts go well with an average score of 4.1 (maximum value = 5), the room restart feature and reducing the player's life when caught by an NPC are going well by getting an average score of 4.6 (maximum value = 5) from respondents who have played Game Maze Cleaner.

Corresponding Author:

Irfan Ainul Afif Refnaldi
Student of Computer Engineering
School of Electrical Engineering, Telkom University
Bandung, Indonesia
Email: irfanainulr@student.telkomuniversity.ac.id

1. INTRODUCTION

In the current era of globalization and modernization, the awareness of Indonesian citizens about the importance of keeping the environment clean is still very low. From all age groups, children, adolescents, and adults still like to throw garbage out of place. Awareness of the importance of keeping the environment clean must be instilled from an early age. One way is by learning character education in schools [1]. Environmental care character building can be done not only by direct practice but can also be taught through education, learning and facilitation materials [2]. It is possible that learning through educational game platforms can also teach students the character of caring for the environment.

Educational games are examples of media that are very suitable for learning. Learning using this educational game is a strategy that is commonly used to invite players to learn while playing in gaining knowledge [3]. Interest in learning is an important aspect to achieve maximum results in learning. With low learning interest, students' desire to learn also becomes low and vice versa. One way to increase student interest in learning is to use a game-based learning model. The game-based learning model is a fun learning model, increases creativity and interest, and improves learning outcomes [4]. Learning using educational games is suitable for kindergarten and elementary school students because the gameplay is not so heavy and is based on

conditions similar to everyday life, which certainly does not contain bad elements that are inappropriate for children. The output of educational games that will be used for learning must really be considered because kindergarten-age children are the age where children's character and character are formed.

Based on the description above the researcher wants to develop an educational game to provide additional options for teachers in an effort to teach environmental care characters to children with the theme "Care for the Environment" with the concept used is a maze game. The systematics of the Maze Game that will be developed is that a player who is a student will enter the labyrinth map and be given a special mission, namely, cleaning up trash scattered in the maze. In the game to achieve the goal so that the game becomes more interesting and the theme of environmental care character education is achieved, the researcher places several Non-Player Characters (NPCs) in the form of skeletons, and pickup objects in the form of garbage that must be picked up or cleaned by players. In developing the behavior of enemy NPC characters, researchers use the Finite State Machine algorithm, which later NPCs will become obstacles or enemies for players in completing missions.

2. METHOD

2.1. Educational Game

Educational games are digital games built for the education sector, supporting the teaching and learning process using interactive multimedia technology [5]. Educational games can also be interpreted as games designed to provide learning to the user, understanding and developing concepts, as well as motivating the user to learn while playing these educational games [6].

. According to Hurd and Jeunings, there are several criteria for educational games that must be met in designing good educational games. The following are some of the criteria for educational games [6]:

1. Overall Value: The overall value of the game is focused on the length of the game and the design of the game.
2. Usability: The important points of a game that will be built are easy to use and easy to access.
3. Accuracy: The success of the description of a game at the planning stage to be built can be realized in the execution process of making a game.
4. Appropriateness: The design and content of the game can adapt well according to user needs.
5. Relevance: As an educational game, the output of playing the game must educate and provide learning to the target user.
6. Objectives: The success of the user in learning and applying the value of the educational game being played.
7. Feedback: The feedback function is to find out the strengths and weaknesses of the games that have been built.

2.2. Game Development Life Cycle

GDLC is a guideline that governs the course of the game making process which consists of 6 processes namely initiation, pre-production, production, testing, beta, release.

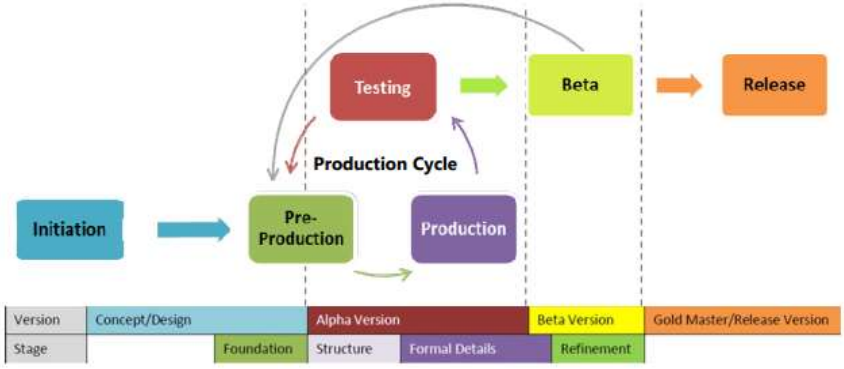

Figure 1. Game Development Life Cycle [7]

Figure 1 displays the process sequence of the game development life cycle, (1) **initiation,** the beginning of developing, a game developer discussing ideas about what kind of game to develop; (2) **pre-**

production, the process of making game designs, perfecting concepts and documentation of game designs, making prototypes of games; (3) **production,** implement game designs, concepts, and other aspects into protype build games; (4) **testing,** testing of prototype builds by developers includes usability test, and functionality test; (5) **beta,** external testing to test the feasibility of the game and detect various errors and complaints from the testing party; (6) **release,** the final build of the official game is released if it has passed beta testing [7].

In this game the GDLC process is as follows, (1) **initiation**, build an educational game with the theme of environmental concern, with gameplay to clean up trash in the maze; (2) **pre production**, making interface designs on the balsamiq wireframes application and designing the behavior of npc objects; (3) **production**, implementing designs that have been made into the GmL language in the game maker language application; (4) **testing**, conducting functionality tests focused on changing npc behavior; (5) **beta**, conducted a test in collaboration with 10 respondents to find out whether the game being built was feasible to play; (6) **release**, this game aims to meet the developer's graduation requirements, this game has not been officially released to the public.

2.3. Maze Game

The maze game is a game that aims to find a GOAL or a way out, by going through several paths on the game map. In the game there are several labyrinth paths that branch out, and not every path is the right path to reach the GOAL because there are trap paths or paths that are blocked by barrier walls, we usually refer to as dead ends [8]. Maze game maps are usually square and rectangular, but it is possible to make maps with other flat shapes.

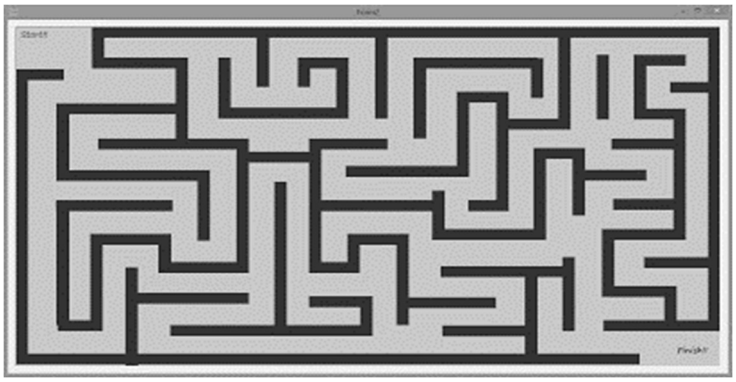

Figure 2. Map on Maze Game

When the game starts the player will be at the starting point, then the player must choose and go through the right labyrinth paths so that the player can find the finish point. The player can be said to have won when the player reaches the finish point before the allotted time span runs out. In some Maze Games, Developers or Game Masters deliberately add additional features to make the game feel more interesting, for example, such as the life limit feature given to players, placing several Non-Player Characters (NPCs) on maze paths that can interact with player avatars, and to adding the level of difficulty to the game there is also a feature that adds a labyrinth path change feature at each predetermined period of time.

2.4. Non-Player Character

Non-Player Character (NPC) is a character in a game that moves automatically or is not controlled by the player, NPC can be in the form of animals, monsters, and in human form like villagers. NPCs have a behavior that is inversely proportional to the player's avatar or main character, where avatars are given direct orders by players in real time, while NPC behavior has been determined or scripted by game developers and game masters. The designed NPC behavior can be said to be very good when it is made very similar to the characters in the real world, where the more behavior is designed, the more movements NPC can make [9] [10].

Non-Player Character is a type of autonomous agent for use in computer animation and interactive media, for example games and virtual reality. These characters have a role in the storyline of the game and can improvise their actions [10] [11]. The division of NPC behavior into three layers according to Raynold (1999) namely action selection, steering, and locomotion [10].

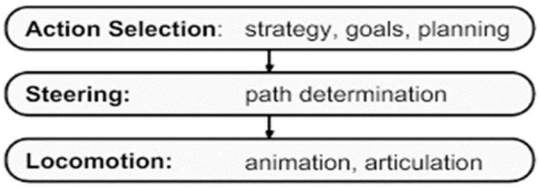

Figure 3. Movement Hierarchy of NPC Behavior

NPC plays an important role in a game because the presence of NPC can determine the level of attractiveness, and interactive or not a game.

2.5. Finite State Machine

Finite State Machine (FSM) is a control system design methodology that describes the working principle of the system by using the following three points, namely State, Event, and Action [12]. Here are some other definitions of a Finite State Machine or also known as a Finite State Automata:

1. FSM is a computing device with input and output of the string type where one of the two values can be accepted and rejected [13].
2. Finite State Automata is a system with a mathematical model with discrete input and output forms. This system can be in one of a number of internal configurations called states [13].

The system in FSM can move or transition to another state if it gets certain feedback or events, either from external devices or components contained in the system itself. The transition from one state to another is also accompanied by a response or action taken by the system when responding to the input that occurs. Actions taken can be simple actions or involve a series of complex processes [12] [13]. The representation of a Finite State Machine is usually depicted in a state diagram, to make it easier to understand the working system of a complex control system [14].

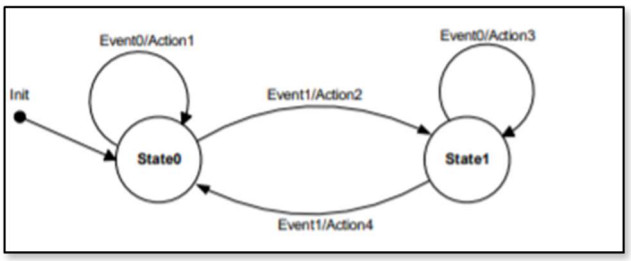

Figure 4. Simple State Diagram

The diagram above displays a simple work system from a Finite State Machine, when the system is turned on, the system will switch to State0 and produce Action1 with input Event0, and if what happens or the input is Event1 then what will be executed is Action2 then the system moves to State1. and so on [14]. The advantage of using fsm in NPC design is that it has a working system that is very suitable for use in changing NPC behavior which is influenced by player movements and surrounding objects.

2.6. NPC Design

The design flow for the behavior of the Non-Player Character in the Maze Game that will be built can be seen in the state chart Figure 5.

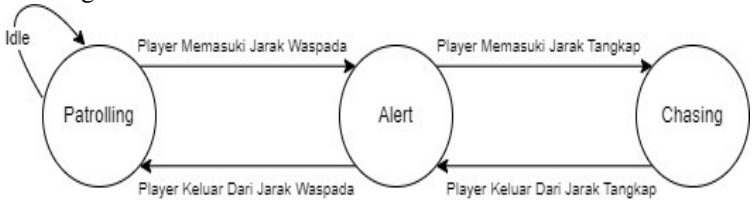

Figure 5. State chart of NPC Behavior in the Maze Game.

Based on the state chart in Figure 5 the flow of designing enemy NPC behavior in the Maze Game is:

1. When the game starts the NPC will enter an idle state or a neutral state then enter a patrolling state
2. In state patrolling the designed behavior of the NPC is that the NPC will walk straight around the map path and change direction when it collides with an object
3. When the player meets and enters the alert radar distance from the enemy NPC, the NPC enters the alert state
4. State alert is a state where the NPC changes state from patrolling to chasing
5. When the alert state is active, and the player stays close to the NPC, the chasing state will be active
6. The behavior of the NPC in the chasing state is that the NPC will chase according to the direction of the player's movement, if the player is caught the player's life will decrease and the room will be restarted
7. If while being chased the player manages to get away from the NPC, the NPC will again enter state alert and state patrol.

3. RESULTS AND DISCUSSION

3.1. Implementation of the game and NPC

In this Maze Game, NPCs are placed on room levels, where each level has more than two NPCs. The NPC will immediately appear on the map when the game starts, then the NPC will move randomly around the labyrinth map (patrolling). When the player is close to the NPC, the NPC will chase and catch the player. The player's health count will decrease if they collide with NPCs.

Figure 6. (a) Home Menu Interface, (b) Room Level 2 Interface, (c) Room Level 3 Interface, (d) Room Level 1 Interface

3.2. Implementation of NPC's Source Code

The source code created to build NPC behavior as expected uses the Finite State Machine algorithm and the Game Maker Language programming language, for an explanation of which can be seen in Figure 7 – Figure 9.

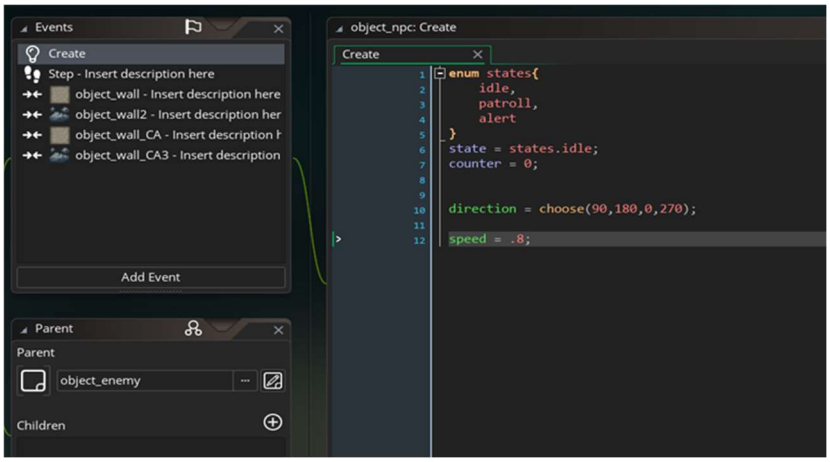

Figure 7. Source Code NPC

Figure 7 is the code for declaring the variables states, idle, patrol, alert with the data type enum. Declare state variable with states idle initial value, counter variable with initial value 0. Give npc random motion direction (up, down, left, right) with code direction = choose (90,180,0,270), and npc movement speed of 0.8 in Game Maker Studio 2 counter.

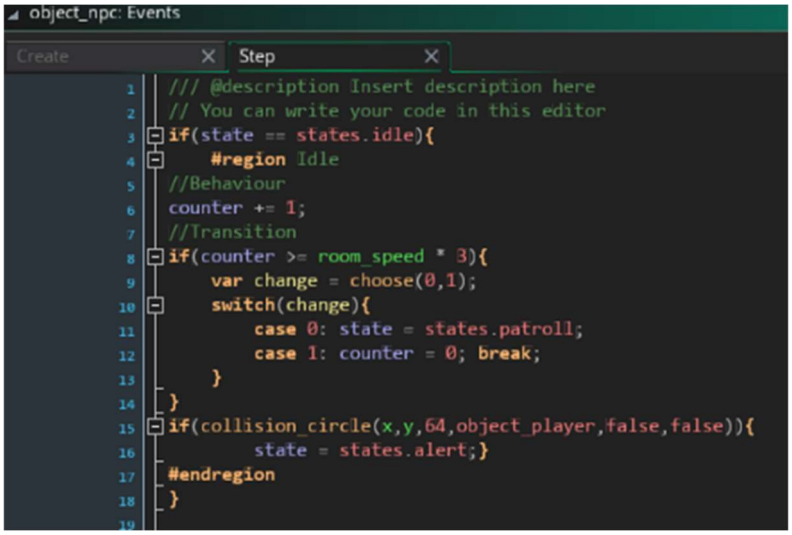

Figure 8. Source Code NPC

Figure 8 is the code for the command on idle states, if the active state is states idle then the counter value will increase by 1, if the counter value is greater than room speed (which has been set to 3 seconds) then the variable will transition with two choices between states patrol or return to its initial condition, namely the counter value = 0. In this code, it is also set that the npc will have a collision circle (a player's monitoring circle). If the player enters the npc's collision circle with a radius of 64 pixels, the code will run the states alert command.

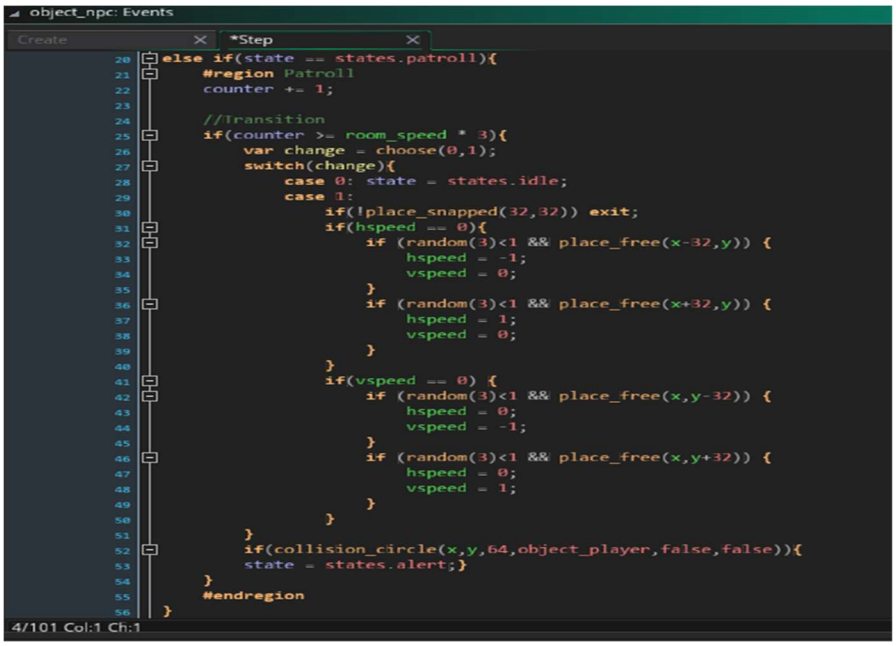

Figure 9. Source Code NPC

Figure 9 explains that if the active condition of the code is states patrol, the counter value will increase by 1, if the counter value is large, the room_speed code executes a switch case command, either returns to idle states or executes commands to the npc to move automatically on the labyrinth map. If the npc is unlocked or detected on the 32x32 pixel game maker grid then no command will be executed (place_snapped annotation). If the horizontal speed of the npc is equal to 0 then a random counter value of 3 is given, which means 1 out of 3 times the possibility of the next command being active (explanation random(3)<1) then checks if the npc can move to the left (place_free(x-32) , y)) then, executing the npc command will move to the left (hspeed= -1, vspeed= 0) and so on to order the npc to move to the right, up, and down. In this code, it is also set that the npc will have a collision circle (a circle of supervision of the player). If the player enters the npc's collision circle with a radius of 64 pixels, the code will run the states alert command.

Figure 10. Source Code NPC

Figure 10 is the display code for the state alert (and chasing) command, giving orders to the npc if the player enters a collision circle with a radius of 64 pixels then the idle state will be active in the sense that the npc will get ready to chase the player, and if the player has entered If the collision circle is 32 pixels in size, the NPC will chase towards the player (direction = point_direction(x,y,object_playerx,object_playery). If the player leaves the collision circle radius, it will run states idle.

3.3. Alpha Testing

To ensure that all the functions and features of the NPC run well, an Alpha test was carried out with the white box method to ensure that the developed NPC runs according to the Developer's wishes. For the

conditions to be tested, namely NPC idle, NPC patrolling, NPC Chasing, Player managed to get away from the NPC's pursuit, Room restart.

Table 1. Alpha Test Table

Number	States	Input	Expected Results	Output	Description
1	Idle	Placement of the number of NPCs at each level	NPC appears according to the specified number	NPC appears according to the specified number	Compatible
2	Patrolling	NPCs move randomly	NPC automatically walks on the labyrinth map	NPC automatically walks on the labyrinth map	Compatible
3	Chasing	The NPC chased the approaching players	NPCs chase approaching players and if caught will reduce the player's life	NPCs chase approaching players and if caught will reduce the player's life	Compatible
4	Player managed to get away from the pursuit of the NPC	NPC repeats state patrolling again.	NPCs are moving randomly again on the labyrinth map	NPCs are moving randomly again on the labyrinth map	Compatible
5	Room restart	Repeat and refresh the maze map	Player is placed in the initial position with the total life that has been reduced, the state of the map will be randomized again	Player is placed in the initial position with the total life that has been reduced, the state of the map will be randomized again	Compatible

From table 1 the results show that the development of NPC behavior is in accordance with what has been designed by the developer. Development of NPC behavior can be indicated success because NPC behavior developed includes 5 factors (idle state, patrol state, chasing, player run away from NPC, room restart) based on the test results of these five factors are compatible between the expected results and the output.

3.4. Beta Testing

the test was carried out based on a questionnaire that was given to 10 student respondents with an age range of 19 -24 containing 6 questions that focused on NPC behaviours that had been designed.

Table 2. Npc Questionnaire Data Likert Scale

No	Respondent's Name	Question						Total
		x1	x2	x3	x4	x5	x6	
1	Wirna	5	5	5	4	4	4	27
2	Hafid	4	4	3	3	5	3	22
3	Ilham	4	3	3	4	3	3	20
4	Restu	5	5	5	5	5	4	29
5	Fajri	5	5	4	5	5	4	28
6	Angga	5	4	3	4	5	5	26
7	Teguh	4	5	3	5	5	4	26
8	Zidan	4	5	5	4	5	5	28
9	Fajar	4	4	3	3	4	4	22
10	Farhan	5	4	5	4	5	5	28

x1-x6 = number of the question

Table 3. NPC Questionnaire Average Score

No	Question	Average Score	Max Score
1	Are NPCs Patrolling When the Game Starts?	4.5	5
2	Will NPCs Chase Players When Players Approach?	4.4	5
3	What is the NPC's Movement Speed When Chasing Players?	3.9	5
4	How Smooth is the Switching of NPC (Patrol-Chasing-Back Patrol) Behavior?	4.1	5
5	When the Player is Successfully Caught, Will the Room Level Restart and the Player's Life Decreases?	4.6	5
6	Will the game become more difficult when the NPC's capture range is increased and the NPC's speed is increased?	4.1	5

$$\boldsymbol{Average\ Score} = \frac{\boldsymbol{total\ score\ of\ a\ question}}{\boldsymbol{amount\ of\ the\ respondents}} \quad (1)$$

Table 2 shows the data on respondents' answers to the questionnaire that has been given, and Table 3 shows the questions and the average score obtained for each question based on the answers from the respondents with a maximum value limit = 5.

4. CONCLUSION

Based on the development of the Maze Cleaner game and the testing that has been done, it can be concluded that, all of the functions (State Idle, Patrolling, Chasing, Player run away from NPC, Room Restart) of the NPC Behaviors can run successfully based on Alpha Testing conducted by developer. NPC behavior shifts go well with an average score is 4.1 (maximum value = 5), the room restart feature and reducing the player's life when caught by an NPC are going well by getting an average score is 4.6 (maximum value = 5) from respondents who have played Game Maze Cleaner.It can be concluded that the development of NPC behavior using the Finite State Machine can run well and according to the author's design.

In the future work suggestions that can be made to improve the game are developers can add more than one type of NPC, add an attractive behavior to NPCs such as chat pop ups and player attack features, and developer can create a boss level NPC or friendly NPC who can give a mission to player. For the whole game features developer can improve the animation with better assets, provide data storage function, provide more varied gameplay, and add a back sound setting button.

REFERENCES

[1] N. Efendi, R. S. Baskara and Y. Fitria, "Implementasi karakter peduli lingkungan di Sekolah Dasar Lolong Belanti Padang," *Jurnal Komunikasi Pendidikan,* vol. 4, no. 2, 2020.

[2] Ismail and M. Jen, "Pendidikan karakter peduli lingkungan dan mejaga kebersihan di sekolah," *Guru Tua : Jurnal Pendidikan dan Pembelajaran,* vol. 4, no. 1, pp. 59-68, 2021.

[3] A. Yulianti and E. Hariadi, "Pemanfaatan media pembelajaran berbasis game edukasi menggunakan aplikasi Construct 2 pada mata pelajaran komputer dan jaringan dasar," *Jurnal IT-EDU,* vol. 5, pp. 527-533, 2020.

[4] Z. Khusniah, Y. Linguistika and E. Ahdhianto, "Analysis on students learning interest improvement through game-based learning model in fractional learning material of mathemathics at the fifth-grade students of SDN PW 01," *Primary: Jurnal Pendidikan Guru Sekolah Dasar,* vol. 11, no. 2, pp. 613-622, 2022.

[5] N. I. Widiastuti and I. Setiawan, "Membangun game edukasi sejarah Walisongo," *Jurnal Ilmiah Komputer dan Informatika,* vol. 1, no. 2, pp. 41-48, 2012.

[6] Nalendra and R. Buyung, "Pembuatan game anak-anak kindergarten " Seek and Seek "," *Naskah Publikasi STMIK AMIKOM YOGYAKARTA,* 2011.

[7] R. Ramadan and Y. Widyani, "Game development life cycle guidelines," in *International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, 2013.

[8] Astawa and S. I Gede, "Penggunaan metode kecerdasan buatan runut maju dalam memecahkan permasalahan game labirin," *Jurnal Ilmu Komputer,* vol. 5, no. 1, pp. 37-46, 2012.

[9] I. M. F. N. Mustika, A. B. Osmond and A. S. R. Ansori, "Membuat pergerakan non-player-character (NPC) menggunakan algoritma Dijkstra," *e-Proceeding of Engineering,* vol. 7, no. 1, pp. 1498-1503, April 2020.

[10] C. W. Reynolds, "Steering behaviors for autonomous characters," 1999.

[11] M. B. Nendya, S. G. Gunanto and R. G. Santosa, "Pemetaan perilaku non-playable character pada permainan berbasis role playing game menggunakan metode finite state machine," *Journal of Animation and Games Studies,* vol. 1, no. 2, pp. 185-202, 2015.

[12] M. F. Rahadian, A. Suyatno and S. Maharani, "Penerapan metode finite state machine pada game "The Relationship"," *Jurnal Informatika Mulawarman,* vol. 11, no. 1, pp. 14-22, February 2016.

[13] A. F. Pukeng, R. R. Fauzi, Lilyana, R. Andrea, E. Yulsilviana and S. Mallala, "An intelligent agent of finite state machine in educational game "Flora the Explorer"," *Journal of Physics: Conference Series,* vol. 1341, no. 4, 2019.

[14] I. Setiawan, "Perancangan software embeded system berbasis FSM," 2006.

[15] A. Pho and A. Dinscore, "Game-Based Learning," *Tips and Trends Spring,* 2015.

[16] C. F. Vara, "Labyrinth and maze: video game challenges," *Space, Time, Play,* pp. 74-77, 2007.