IMPLEMENTATION OF COLLISION AVOIDANCE SYSTEM ALGORITHM IN NPC GAME 3D

Gugah Alwan Hamanako[1], Adi Sucipto[2]

[1,2]Sekolah Tinggi Multi Media "MMTC" Yogyakarta
[1]alwanhamanako@gmail.com, [2]adi.sucipto@mmtc.ac.id

***Abstract*--Nowadays, with the rapid development of technology, we are given the convenience of finding entertainment such as games. Now game can be quickly run on various media, one of them is on a smartphone. Most smartphones' Operating Systems (OS) are Android and IOS, and currently the most popular is the Android. This research aims to create an Artificial Intelligence of Non-Player Character (NPC) that can avoid every obstacle using the Collision Avoidance System algorithm. The result of the research is an Android game app that applies the Collision Avoidance System that can make AI NPC avoid obstacles. The application of the algorithm to the NPC is made in 3 stages, namely, designing by making a flowchart of the algorithm, then writing a program from the Collision Avoidance System, and finally, testing the AI NPC. The test is carried out by comparing the reactions of NPC 1 and NPC 2 in passing through obstacles when on different paths by being carried out jointly between NPC. Based on the tests on AI NPC, NPC managed to avoid obstacles in front of them and the players, and it is almost 93% successful.**

Keywords: Collision Avoidance System; Obstacle; Games.

I. INTRODUCTION

With the rapid development of technology today, we are given the convenience of finding entertainment, such as listening to songs, watching movies, and downloading games via the internet. Game is a form of entertainment often used as a mind refresher from fatigue caused by our activities and routines [1]. Nowadays, the game can run from various platforms. One of the most popular media is the smartphone. Many smartphones use Android and IOS as Operating Systems. The most popular OS for a smartphone now is Android. Android is a mobile Operating System based on Linux that covers operating systems, middleware, and application [2].

Currently, there are several games of *balap karung (sack race)*, such as "Game17an" and "Kucing dalam Karung". "Game17an" is a games consisting of several games that are usually played to celebrate Indonesia's Independence Day (commonly known as Agustusan). One of the games in "Game17an" is Game "Balap Karung". This game asks the player to wear sacks so it covers their chest and lower body, then they race with other players. In "Game17an" player uses the left and right buttons to change the track of the player in the game. This game's goal is to avoid the 'enemy' and reach the finish line. The game "Kucing dalam Karung" uses the same way to play as "Balap Karung" but the difference in this game uses a cat as the character. Players have to press the screen to make the character jump and press the screen again after the character landing on the ground so the character does not fall. The game will finish if the player is too late to reach the finish line.

Many games of *balap karung* use simple game mechanics, and among of them do not have NPC as an enemy. "Game 17an" and "Kucing dalam Karung" has the NPC in its game. However, in these games, the NPC is just an obstacle that inhibits the player from finishing the game, and the NPC does not have the behavior to avoid another player. Meanwhile, in the game "Kucing dalam Karung", the NPC can only speed-racing with the player and cannot move to another track, and neither can avoid the player.

A Non-Player Character or known as NPC is a character in a computer game that the player does not control. NPCs can be intelligent agents created by embedding Artificial Intelligence (AI) having human-like behavior, such as changing their strategy or mindset in response to the opponent's actions [3]. NPCs can have the ability to make movements or interactions automatically by the computer and are not controlled by the player [4].

There are several algorithms that can be used for NPCs in racing or pathfinding games, namely (1) A* algorithm, which is one of the best algorithms most commonly used in pathfinding, especially for AI in games [5],[6]. The A* algorithm always checks all unexplored locations. When it is explored, it will record all of its neighboring locations for further exploration up to its objective point [7], (2) Dijkstra's algorithm, which is the most frequently used algorithm in finding the shortest route [8], is simple to use by using simple nodes on an uncomplicated road network [9], and (3) The Collision Avoidance System algorithm is an algorithm that functions to carry out an interaction with the nearest obstacle and then avoids it, which is inserted into the Non-Player Character (NPC) [10].

One of the algorithms that can be embedded in the NPC is the Collision Avoidance System. This algorithm functions to carry out an interaction with the closes obstacles, then avoid them. Milak conducted similar research in 2019 and applied Artificial Intelligence to Non-Player Characters using the Collision Avoidance System algorithm and Random Number Generator in the 2D game "Balap Egrang". In this research, the Collision Avoidance System algorithm is used by NPC to avoid obstacles. This game format is 2D and uses side-scrolling to play the game [11].

So far, scientific articles show that no sack race game implements the Collision Avoidance System algorithm on NPCs. The 3D game "Balap Karung" in this research implements this algorithm. The NPC is made so that it can have artificial intelligence that can avoid obstacles as well as player characters. This AI. can provide challenges for players in completing the game.

II. METHOD

A. Life Cycle Development Stage

There are three stages involved in the creation of this production work, namely the pre-production stage, the production stage and the post-production stage. The creation process begins with the search for ideas, concepts and data which will then be used as mechanics in the game to be produced. The development of these ideas and data are carried out by conducting research on sack race game and the rules contained in this traditional game.

From the results of this research then it is developed into a game mechanic and documented into the creation of a Game Design Document which is a document from the initial concept to the design in making games “Balap Karung”. All the concepts that are made in the Game Design Document will then be discussed with team members in the production process. At this stage you can find out the level of difficulty in the production process so that if there is a game design concept, whether it comes from the mechanics, visuals and programmers; that are not appropriate, it can be reduced.

At the production stage, work has entered the respective job description sections. In accordance with what is discussed in the initial design stage. At this stage the game mechanics and features discussed in the previous stage are made by making a flowchart of the NPC algorithm, then starting to be converted into program form (Fig. 1).

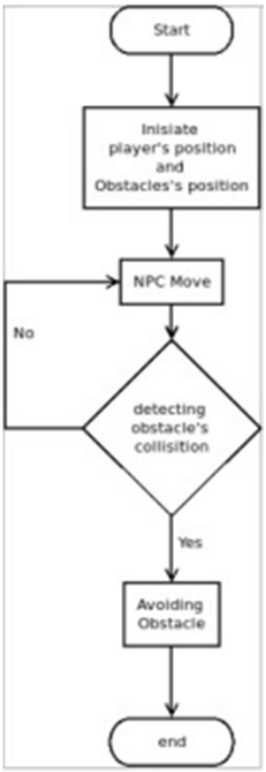

Fig. 1. Flowchart Collision Avoidance System

If there are changes to mechanics or features that are inappropriate for the game, discussions will be held with the game designer. Every game mechanic program, NPC features and algorithms are made on Unity and Visual studio software.

Then the post-production stage, at this stage testing of the mechanics and games that are made is based on the design that was carried out in the previous stage. The process includes testing the black box NPC game independently to check whether the NPCs made are in accordance with those in the Game Design Document, conducting testing on individuals or groups to find out reviews and evaluations of the strengths and weaknesses of NPCs, revising the evaluations that are carried out in the previous stages, and publishing the final results of the games that are carried out in accordance with the previous stages. Targeting ages 13 and over, the end result of this game production works can be accessed through the game sharing website, itch.io via the link https://ghamanako.itch.io/balap-karung.

B. Collision Avoidance System Algorithm

Fig. 2 shows pseudocode algorithm collision avoidance system [11]. The stages of the Collision Avoidance System algorithm in that figure are as follows:

1. The algorithm will detect obstacles or obstacles that are ahead.
2. Two possible obstacles that must be passed
3. The obstacle is in front of the right, so the algorithm will read and dodge to the left.
4. The obstacle is in front of the left, so the algorithm will read and dodge to the right.

```
Start
        If Obstacle true
                then avoid active
        Else avoid nonactive
 Offset
        If obstacle is in right
                Set avoid active then going left
        Else
                If obstacle is in right
                        Set avoid active then going left
                Else
                        move Forward
                Endif
        Endif
 Endif.
```

Fig. 2. Pseudocode Algorithm Collision Avoidance System [11]

This algorithm uses Collision Detection, which is the process of detecting collisions between two objects. Collision Detection is also useful for determining the position of one object with another so that no objects penetrate one another [10].

III. Result and Discussion

The "Balap Karung" game is a 3D mobile game that can be categorized into the casual genre. Furthermore, it carries the theme of a traditional Indonesian game often played during Indonesia's Independence Day celebrations, namely *Balap Karung* with the game mechanics modified to suit the casual game genre but not off the mark from the traditional game *Balap Karung* itself.

There are three playable characters with three lanes in the game. The players can switch to any lane when confronted with an obstacle or get ahead of the enemy. The difference in the levels played is in the level of enemy difficulty, the QTE (Quick Time Elapse) area, which is getting smaller. The obstacles must be confronted, and the distance to the finish the line must be achieved, and players can also share player wins after the game on social media.

A. NPC Programming

This program works for NPCs to avoid obstacles and players (Fig. 3). The Collision Avoidance System Algorithm program, located in the void update() section, will execute the program per frame, which will continue to execute program commands as long as the program or game is running.

```
transform.Translate(Vector3.forward*Time.deltaTime *
    speed);
Ray CenterRay = new Ray(transform.position,
    transform.forward);
Ray LeftRay = new Ray(transform.position -
    (transform.right * 3), transform.forward);
Ray RightRay = new Ray(transform.position +
    (transform.right * 3), transform.forward);
if (Physics.Raycast(CenterRay, out hit, range, layer)) {
    Debug.Log(hit.point.x);
    _isObstacle = true;
if (hit.point.x >= 2.0f  ) {
    transform.position += Vector3.left * LaneDistance;
    Debug.Log("kiri");
} else if (hit.point.x <= -2.0f  ) {
    transform.position += Vector3.right * LaneDistance;
    Debug.Log("kanan");
}
```

Fig. 3. List Program

The NPC will move to avoid obstacles if there is an obstacle in front and the obstacle is in the

specified position. Because this game has three lanes, the obstacle position is determined in advance in the program, so the results of switching lanes from the NPC are optimal. The position of the obstacle are taken in the program so it can run is the x-axis position of the obstacle. Also, to focus more on the obstacle, the condition of the layer where the obstacle is located is added so that it only takes the position of the obstacle object in the specified layer. It also applies to avoid players. Furthermore, in this void update, there are also three conditional functions for NPCs to be able to move positions, such as moving to the left lane, moving to the right lane (Fig. 4), and moving to the right or left lane.

B. Validation Test

This program works for NPCs to avoid obstacles and players (Fig. 3). The Collision Avoidance System Algorithm program, located in the void update() section, will execute the program per frame, which will continue to execute program commands as long as the program or game is running.

Fig. 4. NPC success avoid an obstacle

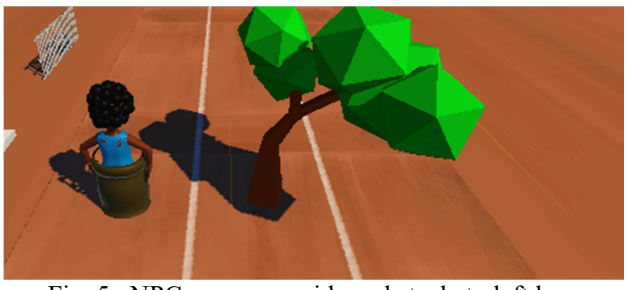

Fig. 5. NPC success avoid an abstacle to left lane

The results from the Table I – III carried out ten experiments at various levels in the game to test the Collision Avoidance System algorithm that was applied to the two NPCs, which initially had their paths. The results of the experiment have a success percentage of 93% out of a total of 30 trials. The algorithm goes well in avoiding the obstacles in front of it. However, several experimental results show that the NPC not avoiding and instead crashing into obstacles, namely when NPC 1 and NPC 2 are on the same path. Then one of the NPCs changed lanes, but when they moved lanes in front of them, there was an obstacle, causing the NPC to glitch by changing positions quickly because the path the NPC was moving to was blocked by NPC 2, so that NPC 1 hit an obstacle (Fig. 6). It happens because the algorithm uses distance to detect colliders, so there are obstacles if the distance is too close when moving from one lane to the next. The algorithm does not read the collider obstacle.

TABLE I
Validation Test of Level 1

Test	NPC (1)	State	Descrip-tion	NPC (2)	State	Descrip-tion
1.1	Left lane	Move to middle lane	Success avoid the obstacle	Right lane	Move to middle lane	Success avoid the obstacle
1.2	Middle lane	Move to right lane	Success avoid the obstacle	Middle lane	move to right lane	Success avoid the obstacle
1.3	Right lane	move to middle lane	Success avoid the obstacle	Right lane	move to middle lane	success avoid NPC
1.4	Middle lane	move to right lane	Success avoid the obstacle	Middle lane	move to right lane	Success avoid the obstacle
1.5	Right lane	move to middle lane	Success avoid the obstacle	Right lane	move to middle lane	success avoid NPC
1.6	Middle lane	move to right lane	Success avoid the obstacle	Middle lane	move to right lane	Success avoid the obstacle
1.7	Right lane	move to middle lane	Success avoid the obstacle	Right lane	move to middle lane	Success avoid the obstacle
1.8	Middle lane	move to right lane	Success avoid the obstacle	Middle lane	move to right lane	success avoid NPC
1.9	Right lane	move to middle lane	Success avoid the obstacle	Right lane	move to middle lane	Success avoid the obstacle
1.10	Middle lane	move to right lane	Success avoid the obstacle	Middle lane	move to right lane	Success avoid the obstacle

TABLE II
Validation Test of Level 2

Test	NPC (1)	State	Descrip-tion	NPC (2)	State	Descrip-tion
2.1	Left lane	move to middle lane	Success avoid the obstacle	Right lane	move to middle lane	Success avoid the obstacle
2.2	Middle lane	move to right lane	Success avoid the obstacle	Middle lane	move to right lane	Success avoid the obstacle
2.3	Right lane	move to middle lane	Success avoid the obstacle	Right lane	move to middle lane	Success avoid the obstacle
2.4	Middle lane	move to right lane	Success avoid the obstacle	Middle lane	crush to obstacle	fail avoid obstacle
2.5	Right lane	move to middle lane	Success avoid the obstacle	Middle lane	move to right lane	Success avoid the obstacle
2.6	Middle lane	move to right lane	Success avoid the obstacle	Right lane	move to middle lane	Success avoid the obstacle
2.7	Right lane	move to middle lane	Success avoid the obstacle	Middle lane	move to right lane	Success avoid the obstacle
2.8	Middle lane	move to right lane	Success avoid the obstacle	Right lane	move to middle lane	Success avoid the obstacle
2.9	Right lane	move to middle lane	Success avoid the obstacle	Middle lane	move to right lane	Success avoid the obstacle
2.10	Middle lane	move to left lane	Success avoid the obstacle	Right lane	move to middle lane	Success avoid the obstacle

TABLE III
Validation Test of Level 3

Test	NPC (1)	State	Descrip-tion	NPC (2)	State	Descrip-tion
3.1	Left lane	move to middle lane	Success avoid the obstacle	Right lane	move to middle lane	Success avoid the obstacle
3.2	Middle lane	move to right lane	Success avoid the obstacle	Middle lane	move to right lane	Success avoid the obstacle
3.3	Right lane	crush to obstacle	fail avoid obstacle	Right lane	crush to obstacle	fail avoid obstacle
3.4	Right lane	move to middle lane	Success avoid the obstacle	Right lane	move to middle lane	Success avoid the obstacle
3.5	Middle lane	move to left lane	Success avoid the obstacle	Middle lane	move to right lane	Success avoid the obstacle
3.6	Left lane	move to middle lane	Success avoid the obstacle	Right lane	move to middle lane	Success avoid the obstacle
3.7	Middle lane	move to right lane	Success avoid the obstacle	Middle lane	move to right lane	success avoid NPC
3.8	Right lane	move to middle lane	Success avoid the obstacle	Right lane	move to middle lane	success avoid NPC
3.9	Middle lane	move to left lane	Success avoid the obstacle	Middle lane	move to right lane	Success avoid the obstacle
3.10	Left lane	move to middle lane	Success avoid the obstacle	Right lane	move to middle lane	Success avoid the obstacle

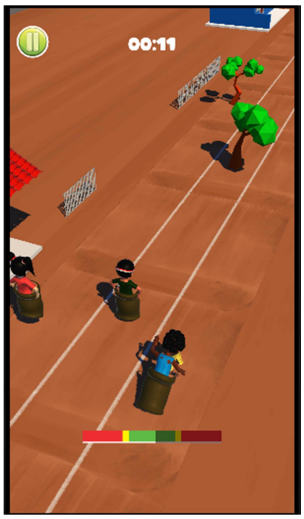

Fig. 6. NPC hit an obstacle

IV. Conclusion

Based on the results of the Collision Avoidance System algorithm research on NPC that is carried out, it can be concluded that the game "Balap Karung" is successfully produced which can run on a mobile platform. It provides a new experience in games with the casual action genre with the theme of traditional Indonesian games, namely sack race.

Furthermore, the successful implementation of the Collision Avoidance System algorithm on NPCs to avoid obstacles, the test results of the Collision Avoidance System algorithm applied to NPCs have succeeded in making NPCs avoid existing obstacles as well as players in the game "Balap Karung". The test was carried out by comparing the reactions of NPC 1 and NPC 2 in passing an obstacle on different paths by carrying it out together between NPCs.

The game "Balap Karung" is successfully

produced, but some players still find some bugs that appear in the game, so some improvements are needed in this game. It is also necessary to reduce the NPC's ability to avoid obstacles because when the game was tested by several people, it was found that the NPS was still quite difficult to beat

V. REFERENCES

[1] M. Ridhoi, *Cara Mudah Membuat Game Edukasi Dengan Contruct 2*. Maskha, 2018. Accessed on 11 Juni 2022. [Daring]. Available in http://archive.org/details/CARAMUDAHMEMBUATGAMEEDUKASIDenganContruct2.

[2] I. Y. Supardi, *Semua Bisa Menjadi Programmer Android Case study*. Elex Media Komputindo, 2014. [Daring]. Available in https://books.google.co.id/books?id=ouVyDwAAQBAJ.

[3] I. Umarov and M. Mozgovoy, "Believable and Effective AI Agents in Virtual Worlds: Current State and Future Perspectives," *Int. J. Gaming Comput.-Mediat. Simul.*, vol. 4, no. 2, pp. 37–59, Apr 2012, doi: 10.4018/jgcms.2012040103.

[4] C. W. Reynolds and others, "Steering behaviors for autonomous characters," in *Game developers conference*, 1999, vol. 1999, pp. 763–782.

[5] A. Candra, M. A. Budiman, and R. I. Pohan, "Application of A-Star Algorithm on Pathfinding Game," in *Journal of Physics: Conference Series*, 2021, vol. 1898, no. 1, pp. 012047.

[6] E. B. Aydoğan and Y. Atay, "Unity Based A* Algorithm Used in Shortest Path Finding Problem for Helicopters," in *2021 International Conference on Control, Automation and Diagnosis (ICCAD)*, 2021, pp. 1–5.

[7] X. Cui dan H. Shi, "A*-based Pathfinding in Modern Computer Games," *IJCSNS*, vol. 11, no. 1, pp. 125, 2011.

[8] Risald, A. E. Mirino, and Suyoto, "Best routes selection using Dijkstra and Floyd-Warshall algorithm," in *2017 11th International Conference on Information & Communication Technology and System (ICTS)*, 2017, pp. 155–158. doi: 10.1109/ICTS.2017.8265662.

[9] X. Cao, X. Li, X. Wei, S. Li, M. Huang, and D. Li, "Dynamic programming of emergency evacuation path based on Dijkstra-ACO hybrid algorithm," 电子与信息学报, vol. 42, no. 6, pp. 1502–1509, 2020.

[10] F. Bevilacqua, "Understanding steering behaviors," *Envotatotuts Envato Pty Ltd*, 2012.

[11] A. S. Milak, E. W. Hidayat, and A. P. Aldya, "Penerapan Artificial Intelligence pada Non Player Character Menggunakan Algoritma Collision Avoidance System dan Random Number Generator pada Game 2D 'Balap Egrang,'" *J. Teknol. Inf. dan Ilmu Komput.*, vol. 7, no. 5, Art. no. 5, Okt 2020, doi: 10.25126/jtiik.2020711816.