Implementation Dijkstra's Algorithm for Non-Players Characters in the Game Dark Lumber

Rio Andriyat Krisdiawan, Aji Permana, Eran Darmawan, Fitra Nugraha, Anto Kriswandiyanto
Universitas Kuningan, Kuningan, Indonesia

*rioandriyat@unik.ac.id

Abstract. Forest is a unitary ecosystem in the form of a stretch of land containing biological natural resources dominated by trees in their natural environment, which cannot be separated from one another. The importance of forest preservation is based on law, namely ensuring the existence of forests in a sustainable manner while maintaining sustainability and not destroying the environment and the surrounding ecosystem. However, public awareness in preserving the forest is still lacking. Therefore, technology is needed that can provide information about forests and the impact of illegal logging in an attractive manner. Games are multimedia entertainment that is made as attractive as possible. Action games require players to concentrate and think quickly in order to avoid obstacles. Dijkstra's algorithm looks for the shortest path in a number of steps from which each step selects the minimum weighted value and enters it into the set of solutions. Dijkstra's algorithm implemented on non-player characters in a game to determine the shortest path to the target player. Games built using Unity and developed using the Game Development Life Cycle (GDLC) method. GDLC is a game development process that applies an iterative approach consisting of 6 development phases, starting from the initialization or concept creation, preproduction, production, testing, beta and release phases.

1. Introduction

Game in English which means game is something that can be played with certain rules so that there are winners and losers, usually in a not serious context or for refreshing purposes. Game can be defined as a tool that can attract attention and a good training environment for the real world in organizations that demand collaborative problem solving (Beck & Wade, 2006). According (Krisdiawan, 2018) “Action games are games that feature action games full of shooting scenes, avoiding obstacles that require concentration to reach the highest point”.

According to law number 18 of 2013 article 1 paragraph 1, forest is an integrated ecosystem in the form of a stretch of land containing biological natural resources, dominated by trees in their natural environment communities that cannot be separated from one another. The importance of forest preservation is based on article 3-point b, which is to ensure the existence of forests in a sustainable manner while maintaining sustainability and not destroying the environment and the surrounding ecosystem.

Games with the action genre are one of the most interesting game genres. Playing this action game requires concentration and speed of thought in solving existing obstacles. The character played by the
user or player will face obstacles and fight or avoid the enemy (Non Players Character). Non player character (Non Players Character) is any character in the game that is not controlled by the player, such as monsters, villagers, and animals in the forest. NPCs represent characters in the story or game and have the ability to improvise their actions. It is the opposite of an animated character from an animated film whose actions are written in advance, and for “avatars” in a game or virtual reality, the player-directed actions are performed in real time. The behavior of the non-player character is made as similar as possible to the real world. The more traits the character can perform, the more moves the character can do (Mustika et al., 2020). (Reynolds, 1999) divides NPC behavior into three layers, namely:

- Action Selection
- Steering
- Locomotion

![Figure 1. Hierarchy of Behavioral Motion](image)

The existence of NPCs in a game is one of the important factors and components in modern computer games that can determine whether the game is interesting or not. The concept of an intelligent agent is one of the models used in making NPCs. The autonomous nature of intelligent agents is an advantage in modeling a game NPC. One of the models or artificial intelligence that can be used in determining NPC behavior is Fuzzy State Machine. Non-player characters have 4 types of reference, namely: NPC Partners, Enemy NPCs, Quest NPCs, Story Support NPCs. The four NPCs can be in one game depending on the type and genre of the game played. An algorithm is needed so that the enemy can find out the position of the player's character and know the shortest distance to chase the player's character. One of the algorithms that can be used in overcoming this problem is the Dijkstra algorithm.

Dijkstra is an algorithm that is applied to find the shortest path on a graph or directed path. However, this algorithm can also be used in undirected graphs. Dijkstra's algorithm looks for the shortest path in a number of steps where from each step we choose the minimum weighted value and enter it into the set of solutions (Dijkstra, 1959). Dijkstra's algorithm has iterations to find the point whose distance from the starting point is the shortest. At each iteration, the distance of the points gives the shortest distance. The condition for this algorithm is the side weight which must be negative (Galih & Krisdiawan, 2018).

2. Methodology

2.1 Game Development Model

The system development method used in this research is to use the Game Development Life Cycle (GDLC) work methodology. GDLC is a game development process that applies an iterative approach consisting of 6 development phases, starting from the initialization / concept creation, preproduction, production, Testing (Alpha testing, Beta testing), and release phases (Krisdiawan, 2018). From these 6 phases, they can be grouped into 3 main processes, namely:

- The initialization process which consists of concept and design,
- The production process consists of Pre Production, Production and Testing (Alpha and Beta)
- Release.
The following is an explanation of the 6 phases described previously, including:

a. Initialization. In this process, planning is carried out regarding the rough concept of the game to be made, such as analyzing the game to be made, making a simple description of the game. And the output of this stage is a rough game concept, namely the type of game to be made, the game scenario, the game character, the story in the game, the target player, the platform used and determining the suitable game engine in game development.

b. Preproduction. At this stage the researcher revises the game concept that was made in the previous stage, and makes the game design more perfect, the game design made focuses on defining the game genre, gameplay, game mechanics, storyline, character design, obstacle design, storyboard, UML (Unified Modeling Language) diagram modeling and documentation.

c. Production. This stage is the core stage in the game-making cycle, this stage includes the creation of game assets including making 3D models for the Character Player, making 3D models for Non-Player Characters both in human and animal forms, making 3D Environment models such as soil, stone, trees and other supporting 3D models, and creating animations for any 3D model that can move, writing program code, animation, sound effects, and integration between game assets and code. The game design that was made in the previous stage is perfected at this stage, all aspects that have been designed are implemented into the composing aspects of the game.

d. Alpha Testing. This test is carried out by the team to check whether there are still bugs or errors and can allow the addition or reduction of features. If there are still bugs or errors and you are required to add features or reduce features, the team will fix them immediately. The method used in this internal testing process is the Black box and White box testing methods.

e. Beta Testing. Testing is done externally by a third party, this is done to test whether the game being built is acceptable and to detect various errors and complaints from third party testers. Beta Testing is outside the production cycle, but if the result of this testing has the potential for error, the researcher will repeat the production cycle. The method used in this external testing process is the UAT (User Acceptance Test) method.

f. Release
This stage is the final stage where the games that have been created and tested in the previous stage are ready to be released and published.

2.2 Flowchart System Game
The following is an overview of the process of players playing the dark lumber game on the scene stage using a flowchart. Figure 3 explanation of the flowchart of game system design.

a. When the player enters the scene stage, the system will check the checkpoint value to position the player character according to the checkpoint position. If the player has played the stage before and has touched the checkpoint, the system will position the player character at the last checkpoint, but if the player has never played the stage, then the checkpoint value = 0 and the player character is positioned at the beginning of the stage.

b. When the player runs a mission, the player character has the possibility to meet the NPC, if the NPC sees the player's character, the NPC's dijkstra algorithm is run to find the closest distance to
the destination node, but if the NPC doesn't see the player's character or the player can avoid the NPC, the players can continue to complete the mission and head to the end of the stage to open the next stage.

c. If the player's character is caught by the NPC, the system will display a failure condition scene and the player will repeat the game.

d. When the player character successfully evades the NPC and manages to touch the end of the stage, the system will count the number of missions that were successfully completed, if all missions are successfully executed, the system will display a successful scene and the next stage player will open, but if the mission has not been completed, then the system will display a scene of failure conditions and the player repeats the game.

Figure 3. Flowchart System Game

2.3 Dijkstra's Algorithm
(Siang, 2019) Dijkstra's algorithm is one of the shortest distance search algorithms that is more efficient when compared to other algorithms, but its implementation is more difficult. To perform the search for the shortest distance using Dijkstra's algorithm, for example, G is a directed graph labeled with points V (G) = {v1, v2, ..., vn} and the shortest route to be searched is from v1 to vn. Dijkstra's algorithm starts from point v1. In its iteration, this algorithm will look for a point with the smallest
number of weights from point 1. The selected points are separated into a set of solutions and the points will not be checked again in the next iteration. Example:

\[ V(G) = \{v_1, v_2, ..., v_n\} \]

\[ L = \text{The set of points } \in V(G) \text{ that have been visited.} \]

\[ D(j) = \text{Total point weights } \in V(G). \]

\[ W(i,j) = \text{Line weight or distance between points } v_i \text{ to } v_j. \]

\[ W^*(1,j) = \text{The smallest number of path weights from } v_1 \text{ to } v_j. \]

---

**Figure 4.** Flowchart Dijkstra's Algorithm

The steps carried out by the Dijkstra algorithm are in accordance with the flowchart in Figure 4. Is:

a. Determine the starting point \( V_s \) and the destination point \( V_d \).

b. Since the starting point of \( V_s \) will be the starting point of the search, give a weight value of 0 (\( D(V_s) = 0 \)) and because the other points have not been visited and the weight values are not known,
give an infinite weight value \( D(V(G)) = \infty \), enter all points other than the starting point into the set of unvisited points \( C = V(G) \).

c. Calculate the weight value from the starting point \( (V_s) \) to the neighbor or nearest point \( (j) \) using the formula \( D(j) = W(V_s,j) \).

Determine the point with the minimum weight value using the formula \( V_k = \min (\infty, D(j)) \). Enter the points with the minimum weight value \( (V_k) \) into the set of points that have been visited \( (L) \) and delete the points with the minimum weight value \( (V_k) \) from the set of points that have not been visited \( (C) \). The point with the minimum weight value \( (V_k) \) will be the starting point to find the next shortest distance.

d. Calculate the weight value from the departure point that has been determined in the previous stage \( (V_k) \) to the neighboring point or the nearest point \( (j) \) that has not been visited using the formula \( D(j) = D(V_k) + W(V_k,j) \in C \).

Determine the new departure point \( (V_k) \) from the above calculation using the formula \( V_k = \min (\infty, D(j)) \). Enter the point with minimum weight \( (V_k) \) into the set of points that have been visited \( (L) \) and delete the point with the minimum weight value \( (V_k) \) from the set of points that have not been visited. The point with the minimum weight value \( (V_k) \) will be the point of departure to find the shortest distance to the next point that has not been visited.

e. Check if there are still points that have not been visited \( (C) \), if they are still there then repeat step 3, if all points have been visited and have the smallest weight value, then determine the shortest distance from the starting point \( (V_s) \) to the destination point \( (V_d) \) through the point which has the shortest weight value. \( W*(V_s, V_d) \).

3. Result and Discussion

3.1 Calculation of Dijkstra's Algorithm

Figure 5. Example of Dijkstra's Algorithm Calculation

Figure 5. Example of Dijkstra's Algorithm calculation is an example of a case when determining the shortest route between the Non Players Character and the player character. Based on the example above, it is determined that the starting point of the quest is the NPC (Non Players Character) and the destination point is the player character. The following is a table of the closest distance search results in accordance with the example above:
Table 1. Shortest distance search result table

| Iterasi | Titik yang belum dikunjungi (C) | Titik yang sudah dikunjungi (L) | Vs     | V(G) |
|---------|---------------------------------|---------------------------------|--------|------|
| 1       | (NPC, n1, n2, n3, n4, n5, n6, KP)| (NPC)                           | n1     | n1   |
| 2       | (n1, n3, n4, n5, n6, KP)        | (NPC, n2)                       | n2     | n2   |
| 3       | (n3, n4, n5, n6, KP)            | (NPC, n1, n2, n3)              | n3     | n3   |
| 4       | (n4, n5, n6, KP)                | (NPC, n1, n2, n3, n4)          | n4     | n4   |
| 5       | (n5, KP)                        | (NPC, n1, n2, n3, n4, n6)      | n6     | n6   |
| 6       | (KP)                            | (NPC, n1, n2, n3, n4, n5, n6)  |       |      |

The following is an explanation of the steps taken in finding the shortest distance according to table 1. above:

a. At this stage, the weight value initialization for each point in the example above, the NPC point is given a value of 0 because it is the starting point of the search, and the other points are given an infinite value (∞).

b. In iteration line 1, the calculation of the value of the weight of the closest point from the starting point of the NPC, the point of the NPC is connected to points n1, n2, and n3. Then the calculation results are obtained as follows:
   - Weight value n1: D (n1) = 0 + 3 = 3
   - Weight value n2: D (n2) = 0 + 2.5 = 2.5
   - Weight value n3: D (n3) = 0 + 4 = 4

c. After getting the results as described above, then determine the smallest weight value (Vk). Vk = min (∞, 3, 2.5, 4) = 2.5 From the results above, it can be determined that the next point of departure is point n2 with a weight of 2.5, point n2 is removed from set L and entered into set C.

d. In iteration line 2, the calculation of the weight value from the point of departure Vk = n2 to the nearest point that has not been visited, n2 is connected to point n6. Then the calculation results are obtained as follows: Weight value n6: D (n6) = 2.5 + 4 = 6.5

e. It can be seen in iteration line 2, the weight values that have been obtained are the point weight values n1 = 3, n3 = 4, and n6 = 6.5. To determine the point with the minimum weight value, the following calculations are performed: Vk = min (∞, 3, 4, 6.5) = 3

f. From the results above, it can be determined that the next point of departure is point n1 with a weight value of 3, point n1 is removed from set L and entered into set C.

g. In the 3rd iteration line onwards, the calculation is carried out as in stage 3, until all points are visited and get the smallest weight value between the starting point of the NPC and the destination point of the KP. From table 3.1 it can be seen that the smallest weight value from the starting point of the NPC to the destination point KP is 9. The paths or paths used are NPC, n1, n6, n5 and KP. W*(Vs, Vd) = (0+3+3+2+1)=9

3.2 Story Line Game

The dark lumber game is an action game with an action platform subgenre. The player plays a role as a teenager who is forced to become a worker in a dark lumber wood factory and intends to run away, on the way the player is required to complete the mission, which is to turn off the electric panel as a source of energy to illuminate the workers in illegal logging, this mission is done to thwart this happening, illegal logging and ensuring the forest remains beautiful and shady. When carrying out a mission, the player will be faced with an enemy (Non Player Character) who is on patrol, the player must avoid the enemy and pass all the obstacles, if the player is caught by the enemy, the mission is considered a failure, but if the player manages to get past the enemy and successfully completes the
mission, then the mission will be considered successful and the player can continue the journey to the next stage.

3.3 Story Board

**Table 2. Storyboard Scene 1**

| Scene 1 | Turn off the electric panel | 1/5 |
|---------|------------------------------|-----|
| Location  | In front of the electric panel lever |   |
| Action   | The player walks closer to the electric panel lever until an indicator appears that says "Press S to turn off the electric panel", if the clue appears, the player can turn off the electric panel by pressing the "S" keyboard, when the panel is off, the light illuminates the workers who are doing it. deforestation will die and deforestation will stop. At that time, an NPC will appear on patrol. |   |

**Table 3. Storyboard Scene 2**

| Scene 2 | NPCs are chasing player characters | 2/5 |
|---------|-----------------------------------|-----|
| Location  | Near the electric panel, where the NPCs are patrolling |   |
| Action   | If the player enters the NPC's vision range, the NPC will chase the player, the animation of the NPC which was originally running animation will change to a running animation, the NPC's speed from 1 will change to 3.2 to allow the NPC to chase and capture the player character, if the player character is caught, then the mission will be considered a failure, but if the player's character manages to avoid the NPC's pursuit, the player can continue completing the mission. |   |

**Table 4. Storyboard Scene 3**

| Scene 3 | The NPC successfully captures the player's character | 3/5 |
|---------|------------------------------------------------------|-----|
| Location  | Near the electric panel, where the NPCs are patrolling |   |
| Action   | When the NPC chases the player, the distance between the NPC and the player will get |   |
closer, if the distance between the NPC and the player character is less than or equal to 1 meter (1 float), the system will assume that the player's character is caught, the NPC animation from the original running animation will change, becomes a jumping animation while hitting, and the player character animation that originally ran will change to a falling animation, after which the system will display a scene of failure conditions.

Table 5. Storyboard Scene 4

| Scene 4 | The player character pushes a wooden box |
|---------|------------------------------------------|
| Location | Near the barrier to the second electrical panel |
| Action | The player character has the ability to push the wooden box by pressing the keyboard Left Shift + Right Direction to push towards the right, and Shift Left + Left Direction to push towards the left, this wooden box can be used as a step to jump and reach the barrier wall. |

Table 6. Storyboard Scene 5

| Scene 5 | The player character touches the end point of the stage |
|---------|--------------------------------------------------------|
| Location | End stage |
| Action | Each stage has an end point or end point marked with a game object containing a script function that can determine whether the player has successfully completed the mission or not, when the player touches this end point, the system will count the number of missions with the number of missions completed, if The number of missions completed is the same as the number of missions, the system will display a successful condition scene and the player can proceed to the next stage, but if the number of missions completed is less than the number of missions, the system will display a failed scene and the player will repeat the current stage. this. |

3.4 Implements System

Figure 6. Game play menu display and view
4. Conclusion
a. Dijkstra's algorithm can be implemented in the dark lumber game to regulate the movement of the Non Player Character in finding and determining the shortest distance to the destination point.
b. The dark lumber game displays educational content on the Splash Screen scene in the form of a 3-dimensional animation containing information about forests and forest destruction.
c. The dark lumber game will also display educational content when the player fails or succeeds in completing the stage, if the player fails to complete the stage, the game system will display a 3-dimensional animation containing information about forest damage due to illegal logging, if the player succeeds in completing the stage, the system will display a 3-dimensional animation containing information about the shady forest.

References

[1] Beck, J. C., & Wade, M. (2006). The kids are alright: How the gamer generation is changing the workplace. Harvard Business Press

[2] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. Numerische Mathematik, 1(1), 269–271.

[3] Galih, E. C., & Krisdiawan, R. A. (2018). Implementasi Algoritma Dijkstra Pada Aplikasi Wisata Kuningan Berbasis Android. Nuansa Informatika, 12(1). https://doi.org/10.25134/nuansa.v12i1.1344

[4] Krisdiawan, R. A. (2018). Implementasi Model Pengembangan Sistem Gdlc Dan Algoritma Linear Congruential Generator Pada Game Puzzle. Nuansa Informatika, 12(2), 1–9. https://journal.uniku.ac.id/index.php/ilkom/article/view/1634/1211

[5] Krisdiawan, R. A., Ramdoni, R., & Permana, A. (2020). RANCANG BANGUN GAME TREASURE OF LABYRINTH DENGAN ALGORITMA BACKTRACKING BERBASIS ANDROID. NUANSA INFORMATIKA, 14(1), 46.

[6] Krisdiawan, R. A., Rohmana, M. F., & Permana, A. (2020). Pembuatan Game Runaway From Culik Dengan Algoritma Fuzzy Mamdani. Buffer Informatika, 6(1), 33–40.

[7] Mustika, I. M. F. N., Osmond, A. B., & Ansori, A. S. R. (2020). Membuat Pergerakan Non-player-character (npc) Menggunakan Algoritma Dijkstra. EProceedings of Engineering, 7(1).

[8] Reynolds, C. W. (1999). Steering behaviors for autonomous characters. Game Developers Conference, 1999, 763–782.

[9] Siang, J. J. (2019). Matematika Diskrit dan aplikasinya pada ilmu komputer.

[10] Sunarto, D., & Krisdiawan, R. A. (2018). RANCANG BANGUN GAME KUMBANG KUM OID MENGGUNAKAN ALGORITMA A* (STAR) BERBASIS ANDROID. NUANSA INFORMATIKA, 11(2).