Implementation of Artificial Ant Colony Algorithm (AACA) in dynamic labyrinth generation for android-based 2D games

Darlis Herumurti¹, Imam Kuswardayan¹, Wijayanti Nurul Khotimah¹, Andhik Ampuh Yunanto², Andi Akram Yusuf²

1 Department of Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia
2 Department of Informatics and Computer Engineering, Politeknik Elektronika Negeri Surabaya

*Email: darlis@if.its.ac.id

Abstract. Playing games can make people feel fun. Players can be more fun if you have a lot of variations, for example, variations of items, maps, or enemies. Map variations can be fulfilled by implementing a dynamic labyrinth generator. In this research, the game has a feature called Dynamic labyrinth generation using the Artificial Ant Colony Algorithm (AACA) method. For deployment, this game is implemented on the Android mobile platform. The evaluation is conducted by testing functionality in system games. The result indicates AACA can generate dynamic mazes and with optimal time with less than five seconds. So, the conclusion from this research that the AACA method can generate dynamic labyrinth quickly and well for Android Game.

1. Introduction
Playing game is an exciting activity. Therefore, playing games can eliminate boredom. Of course, games that can be played quickly, easily and not boring require a lot of variety that can be explored further. These variations can be maps, items, enemies, or other things. For example, every time a player enters the same level, the map that is displayed is always different, or there are rare items that can only be obtained if we have completed certain missions or conditions.

In this year, there are a lot of game application in market place. The types of games that are marketed are also very diverse such as educational games [1], virtual reality [2], survival games [3], and even games that also apply artificial intelligence [4], natural language [5], and text mining [6]. The game is not only 2D-based but there is also a 3D-based both on desktop and mobile platforms [7]. In essence, this game is very popular among all people.

One such variation, namely the varying map requirements can be met by implementing dynamic labyrinth generation using several algorithms [8], some of which are the Kruskal and the Prim Algorithm [9]. In this paper the author tries to use the Artificial Ant Colony Algorithm (AACA) discussed by Dawid Polap for the game 'Ant Cave' [10]. In addition, Salim also study about AACA and has result that the performance of AACA algorithm can solve almost 70% for graph optimization problem [11]. The use of this algorithm can produce a variety of labyrinths, so it is considered suitable to be applied as a dynamic Maze Generator [12]. The aim of this paper is to be able to produce game applications that have different levels of each maze, so playing games becomes more fun.
2. Design

2.1. Ant Cave Game Scenarios
This game is a 2D game with an adventure theme. This game is played by one person. At each level the target of this game is to find the stairs down without losing blood in the form of health points. At each level there are also enemies who also have blood in the form of health points. Players can pass the enemy without fighting with the enemy to avoid unnecessary reduced health points. To pass a certain road the player needs to destroy the enemy by fighting first. Players and enemies will fight automatically, but players can choose to escape the fight. There will be 10 levels where each level will have stairs to go down to the next level. The starting position of the player, enemy and descending stairs will be randomized at each level.

Enemy arrangements are also set at this stage. There are three types of enemies, namely weak, normal, and strong. The enemy designs can be seen in Table 1.

| Level | Weak | Normal | Strong |
|-------|------|--------|--------|
| 1     | 1    | 0      | 0      |
| 2     | 2    | 0      | 0      |
| 3     | 3    | 0      | 0      |
| 4     | 1    | 1      | 0      |
| 5     | 1    | 2      | 0      |
| 6     | 0    | 3      | 0      |
| 7     | 1    | 0      | 1      |
| 8     | 1    | 1      | 1      |
| 9     | 0    | 2      | 1      |
| 10    | 1    | 2      | 2      |

2.2. Rules of the Ant Cave Game
This game has several rules, namely:
1. At each level there will be enemies that appear.
2. Players can move the character by sliding the screen.
3. Players can attack the enemy by moving the player to where the enemy is located.
4. Players can choose to ignore enemies.
5. Players and enemies will have health points, speed, attack, and defense attributes.
6. Health point is the blood of the player and character.
7. If the enemy loses all his health, the enemy will be destroyed.
8. If a player loses all his health then the game will lose.
9. The speed attribute will determine the character that will attack first, the character that has a higher speed will attack first.
10. Attribute attack and defense will affect how much damage will be generated; and the minimum damage value is 1.
11. The design of player and enemy attributes can be seen in Table.
12. Players and enemies will automatically attack in accordance with the attributes of each character.
13. Players can choose to run away when fighting, but players can run away after the enemy is finished and the player attacks or one turn.
14. The player must find a descending ladder in order to enter the next level.
15. The starting location of the player, enemy and descending stairs will be randomized.

In Addition, the attribute or capability of player and enemy is shown in Table 2.
2.3. Ant Cave Maze Design

The labyrinth that will be needed by the Ant Cave game is a maze that can be drawn on the Cartesian plane. Every one value of x and y will represent one city that can be passed by ants. Each city will have its own pheromone value. If the pheromone value exceeds the limit it will be considered as a road that can be passed. Examples of the desired maze can be seen in Figure 1. The conditions for a labyrinth are:

1. There are accessible cities that are connected at a distance of 1 or are on all four sides.
2. Cities that can be traversed are cities where the pheromone value exceeds the limit, which is 0.3.
3. The number of cities that are connected at least 30% of the total cities and a maximum of 75% of the total cities.
4. The location of the entrance and the exit will be randomized.

![Figure 1. Labyrinth level design example](image)

2.4. Artificial Ant Colony Algorithm

Initiation, this stage is the stage to determine the size of the map (column x rows), the number of ants, evaporation value, alpha coefficient, beta coefficient, pheromone minimum limit as a road, pheromone initial value. Random Ant Position, at this stage the ant position will be randomly used using the default function of unity, Random.Range(). For each ant the value of x and the value of y that determines the initial position of the ant will be randomized with Random.Range().

The next stage is repetition, the intended loop is that each ant will have its own distance, a list of cities that have not been visited separately, and a separate probability list, but when updating pheromones in each city will be done once for all ants. After updating the pheromone value, the map will be checked whether the labyrinth has been formed with the DFS algorithm when it has formed, then this Artificial Ant Colony Algorithm has been completed. If the labyrinth has not formed and there are still cities that have not been visited, the ant will repeat this loop. But if a labyrinth has not yet been formed and all cities have been visited, the position of the ant will be re-randomized.

Calculating distance, calculating the distance in question is calculating the distance between ants and cities that have not been visited by ants. To calculate the distance used formula (1).
\[ L_{ij} = \| x_i - x_j \| = \sqrt{\sum_{k=1}^{2} (x_{i,k} - x_{j,k})^2} \]  

(1)

Calculating the probability, calculating the probability in question is calculating the probability of each city that will be chosen by the ants. If there are two cities that have the same pheromone value and distance, the selected city will be randomized. To calculate the probability used formula (2).

\[ p^t(x_i, x_j) = \frac{[f^t(x_i, x_j)]^{\alpha} \left( \frac{1}{L_{ij}^t} \right)^\beta}{\sum_{a \in N_i} [f^t(x_i, x_j)]^{\alpha} \left( \frac{1}{L_{ij}^t} \right)^\beta} \]  

(2)

Moving ants, moving ants means that the ant will walk towards the city with the highest probability. It is assumed that the time to walk is 1 time even though there are ants that cover longer distances.

Updating pheromone values, to update pheromones used formula (3). The value of \( n \) in formula (4) is the number of ants in the city \( x_i, x_j \). If there are no ants in the city, the value \( \Gamma_i^t \) is 0 because there are no ants that release pheromones in the city. If there is more than one ant, the value of \( \Gamma_i^t \) is the number of pheromones released by the ants.

\[ f^{t+1}(x_i, x_j) = (1 - \rho) f^t(x_i, x_j) + \Gamma_i^t \]  

(3)

\[ \Gamma_i^t = \sum a \frac{1}{L_{ij}^t} \]  

(4)

Checking the labyrinth with DFS, checking the labyrinth with the DFS algorithm aims to check whether the labyrinth has been formed or not. If the labyrinth has been formed then complete this algorithm. Checking cities that have not yet been visited, checking cities that have not yet been visited aims to determine which cities will be visited next which are also cities that have not been visited by ants. When all cities have been visited by ants, a new iteration will start with a new memory. This memory contains distance, probability, and also the city data that has not yet been visited.

3. Result
The application is tested in a smartphone with the type of Sony Xperia Z1 Compact, Qualcomm MSM8974 Snapdragon 800 processor, 2GB of memory, and the Android operating system 5.1.1. The testing scenario is conducted by testing functionality which using black box testing. Testing functionality is carried out by preparing several test scenarios as benchmarks for testing success.

In addition, this research also evaluated by system test. The first test is to find out if different labyrinths are formed at each level. And the second test is to find out whether players can finish the game and whether players can complete one level of the game. The labyrinth formed can be seen in Figure 2 until Figure 6. And the result of functionality can be seen in Table 3.

| Testing Code | Name                                         | Result   |
|--------------|----------------------------------------------|----------|
| PF001-1      | Labyrinth Testing (Scenario 1)               | Success  |
| PF001-2      | Labyrinth Testing (Scenario 2)               | Success  |
| PF001-3      | Labyrinth Testing (Scenario 3)               | Success  |
| PF002-1      | Rules Testing (Scenario 1)                   | Success  |
| PF002-2      | Rules Testing (Scenario 2)                   | Success  |
| PF002-3      | Rules Testing (Scenario 3)                   | Success  |
Table 3 shows about functionality result and divided by two testing. The first test is Labyrinth testing which testing to check formed labyrinth in the system. This testing can success if the labyrinth is formed dynamically. Furthermore, the second testing is Rules testing which to check the user or player when playing the game. This testing can success if the user can complete one level. From two testing, the result show that all scenario is success. It means the application is not bad in this research and the system can run smoothly and dynamically. But in this research, there are still many shortcomings because it is still the first stage of development.

4. Conclusion
From the observations during the work process starting from the process of design, implementation, and testing that has been carried out on the application, this research has several conclusions. The use of Artificial Ant Colony Algorithm can be used to generate different or dynamic labyrinths and with an optimal time which only five seconds for 2D games based on Android. The game scenario has been implemented in the game 'Ant Cave' in the form of levels contained in the game. Game rules have been implemented in the 'Ant Cave' game. The advice given to develop this game is that this game still requires further development, especially in terms of setting the level or level of difficulty and also in terms of appearance.
Reference

[1] A. A. Yunanto, D. Herumurti, I. Kuswadayan, R. R. Hariadi and S. Rochimah, “Design and Implementation of Educational Game to Improve Arithmetic Abilities for Children,” in 12th International Conference on Information & Communication Technology and System (ICTS), Surabaya, Indonesia, 2019.

[2] D. Herumurti, A. Yuniarti, P. Rimawan and A. A. Yunanto, “Overcoming Glossophobia Based on Virtual Reality and Heart Rate Sensors,” in 2019 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT), Bali, Indonesia, 2019.

[3] I. Kuswadayan, D. Herumurti, R. R. Hariadi, M. Wildianurahman, A. A. Yunanto and S. Arifiani, “Survival Education for User on Unknown Islands using Simulation Games,” in 12th International Conference on Information & Communication Technology and System (ICTS), Surabaya, Indonesia, 2019.

[4] A. A. Yunanto, D. Herumurti, I. Kuswadayan and S. Rochimah, “Intelligent System for Agent in Educational Game Using Dynamic Gram Similarity,” in 2018 International Seminar on Application for Technology of Information and Communication, Semarang, Indonesia, 2018.

[5] A. A. Yunanto, D. Herumurti, S. Rochimah and I. Kuswardayan, “English Education Game using Non-Player Character Based on Natural Language Processing,” in The Fifth Information Systems International Conference, Surabaya, Indonesia, 2019.

[6] A. A. Yunanto, D. Herumurti and I. Kuswardayan, “Kecerdasan Buatan Pada Game Edukasi Untuk Pembelajaran Bahasa Inggris Berbasis Pendekatan Heuristik Similaritas,” Jurnal Sistem dan Informatika (JSI), 2018.

[7] A. A. Yunanto and S. Rochimah, “Systematic Literature Review Terhadap Evaluasi Perangkat Lunak Tentang Serious Game,” Jurnal Informatika, 2017.

[8] W.-C. Chang, Y.-D. Chiu and M.-F. Li, “Learning Kruskal’s Algorithm, Prim’s Algorithm and Dijkstra’s Algorithm by Board Game,” in International Conference on Web-Based Learning, Jinhua, China, 2008.

[9] Wamiliana, M. Usman, Warsono and Warsito, “Computational Aspects of Some Algorithms for The Multiperiod Degree Constrained Minimum Spanning Tree Problem,” in Journal of Physics: Conference Series, 2019.

[10] D. Polap, “Designing Mazes for 2D Games by Artificial Ant Colony Algorithm,” 2016.

[11] S. Bouamama, C. Blum and J.-G. Fages, “An algorithm based on ant colony optimization for the minimum connected dominating set problem,” Applied Soft Computing, vol. 80, pp. 672-686, 2019.

[12] Z. Hu, K. Wan, X. Gao and Y. Zhai, “A Dynamic Adjusting Reward Function Method for Deep Reinforcement Learning with Adjustable Parameters,” Mathematical Problems in Engineering, 2019.