DAFTAR PUSTAKA

Astuti, D., Anggi, A. F., Kusuma, A., & Syah, F. (2022). Application of the Finite State Machine Method to Determine the End of the Story Based on User Choice in Multiple Role Playing Games. In *International Journal of Computer and Information System (IJCIS) Peer Reviewed-International Journal* (Vol. 03). https://ijcis.net/index.php/ijcis/index

Budaya, M., Bangsa, T., Septa, B. A., & Saifudin, A. (n.d.). Penerapan Algoritma Finite State Machine pada Game Horror 3D untuk. *Maret*, *4*(1), 2622–4615. http://openjournal.unpam.ac.id/index.php/informatika

Darmawan, A. P., & Rubhasy, A. (n.d.). KOMBINASI ALGORITMA A STAR DAN FINITE STATE MACHINE PADA AI ENEMY GAME HORROR 3D ESCAPED. *Kumpulan JurnaL Ilmu Komputer (KLIK)*.

Farrel, Y., Aris Gunaryati, dan, Teknologi Komunikasi dan Informatika, F., Nasional Ps Minggu, U., Jakarta Selatan, K., & Khusus Ibukota Jakarta, D. (n.d.). *Farrel, Fauziah, dan Gunaryati-Perbandingan Algoritma Kombinasi Antara Algortitma A* Pathfinding Navmesh dengan Algoritma Waypoint Pathfinding Pada Game Tower Defense Rise of Machine KOMBINASI ALGORITMA A* PATHFINDING NAVIGATIONAL MESH DENGAN WAYPOINT PATHFINDING PADA GAME TOWER DEFENSE RISE OF MACHINE*.

Febriyanto Riski, E., Triayudi, A., & Mardiani, E. (2020). Implementation Of A Star Algorithm In Android Based Alien Shooter Games. *Jurnal Mantik*, *4*(1). https://iocscience.org/ejournal/index.php/mantik

Goldstone, Will. (2009). *Unity game development essentials*. Packt Pub.

Ljung, K. (2015). *Effects of Field-of-View in First-Person Video Games A Study on Camera Field-of-View in Relation to Game Design*.

P.S, G. G. (2015). Platform Comparison Between Games Console, Mobile Games And PC Games. In *SISFORMA* (Vol. 2, Issue 1).

Safira, L., Harsadi, P., & Harjanto, S. (2021). Penerapan Navmesh Dengan Algoritma A Star Pathfinding Pada Game Edukasi 3d Go Green. *Jurnal Teknologi Informasi Dan Komunikasi (TIKomSiN)*, *9*(1), 17. https://doi.org/10.30646/tikomsin.v9i1.540

Schell, J. (n.d.). *The Art of Game Design: A Book of Lenses*.

Wahyu, E., #1, H., Nur, A., #2, R., Fauzan, M., & #3, A. (2019). *JEPIN (Jurnal Edukasi dan Penelitian Informatika) Penerapan Finite State Machine pada Battle Game Berbasis Augmented Reality*.

Skripsi Ganjil 22/23

ORIGINALITY REPORT

15%	15%	5%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	etheses.uin-malang.ac.id Internet Source	4%
2	core.ac.uk Internet Source	1%
3	Submitted to Universitas Brawijaya Student Paper	1%
4	qadrisa.wordpress.com Internet Source	1%
5	www.researchgate.net Internet Source	1%
6	publishing-widyagama.ac.id Internet Source	<1%
7	docplayer.info Internet Source	<1%
8	pt.scribd.com Internet Source	<1%
9	repository.its.ac.id Internet Source	<1%

Jurnal Ganjil 22/23

ORIGINALITY REPORT

9%	5%	6%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Abdul Rafiq, Tuty Asmawaty Abdul Kadir, Siti Normaziah Ihsan. "Pathfinding Algorithms in Game Development", IOP Conference Series: Materials Science and Engineering, 2020 Publication	1%
2	media.neliti.com Internet Source	1%
3	Submitted to Napier University Student Paper	1%
4	Submitted to University of Greenwich Student Paper	1%
5	Submitted to University of Westminster Student Paper	1%
6	Submitted to Padjadjaran University Student Paper	<1%
7	jurnal.itscience.org Internet Source	<1%

A STAR NAVIGATIONAL MESH AND FINITE STATE MAHINE ALGORITHM COMBINATION IN AI NPC ENEMY GAME 3D SPY IN LIFE

Muhammad Elang Perkasa Putra Dolok Saribu[1)], Fauziah[2)] *, Ben Rahman[3)]
[1)2)3)]Universitas Nasional, Jakarta, Indonesia
[1)]elangperkasa2001@gmail.com, [2)]fauziah@civitas.unas.ac.id, [3)]benrahman@civitas.unas.ac.id,

Submitted : xxxxx | **Accepted** : xxxx | **Published** : xxxxx

Abstract: The development of game technology has experienced rapid development and the emergence of types of games that are of interest, one of which is entertainment games, entertainment games have the Stealth genre where players are ordered to avoid fighting to minimize noise and look for enemy information in the shadows. The development of this Stealth video game must coincide with the development of an Artificial Intelligence NPC (Non Player Character) that can interact as an enemy in the game environment. NPCs must be made able to have the ability to chase and detect players. The purpose of this research is to find out what methods and algorithms can be used to detect players. So one method that can be used is the A* Navigational Mesh algorithm using a Finite State Machine. The A* Navigational mesh algorithm is used as a Pathfinding algorithm to determine patrol paths and also chase players. The finite state machine method is used for the interaction of input situations from players and as the logic that decides what the NPC should do to the players. The results of this study from the application of the algorithms and methods found that the three can be combined well. The results obtained are from 300 recorded data frames, in this process the algorithm has 2 results obtained from Level 1 and Level 2. At Level 1 the highest value is on frame 143 and has a total resource value of 2.1% while Level 2 has the highest value on frame 240 and has a total resource value of 5.1%. This is quite large, resulting in benchmark results at Level 1 rendering as many as 2425 frames with a time of 81,281 s, has an average fps value of 29.8 FPS and has the highest FPS value of 60 FPS. At Level 2 it renders 4991 frames with a time of 232.015s, has an average fps value of 21.5 FPS and has the highest FPS value, namely 60 FPS, which means that the device used is not optimal to play the game.

Keywords: A Star; Navigational Mesh; Finite State Machine; Artificial Intelligence;

1. INTRODUCTION

The development of technology is indeed an undeniable phenomenon, in almost every aspect of human life using technology, including game technology which is experiencing rapid development with the emergence of various types of games that are of interest to the players, be it educational, serious, and entertainment games(Astuti et al., 2022). In serious games there is the Stealth genre where players are directed to avoid combat, minimize noise, and look for enemy information in the shadows. In the stealth genre, it can be combined with various themes, one of which is Spy. In this Spy, the player will have the main goal of taking information or belongings of the enemy, which means the player is in the enemy environment and the second goal is to escape from the enemy environment. therefore a Non Player Character (NPC) or Artificial Intelligence (AI) is needed from the enemy which must have the function of finding and capturing players by utilizing pathfinding and waypoints(Alvin et al, 2022.).

*name of corresponding author

In previous research, testing the A* algorithm with Navmesh. The results obtained in the test have 300 recorded frames, frame 21 has a total resource of 0.6% while the benchmark results render as many as 26134 frames in 517,531 s, meaning that the previous test went well without draining the required power(Farrel et al., 2022).

This study combines three algorithms, namely Navigational Mesh (Navmesh) A-Star (A*) as pathfinding waypoints and Finite State Machine as input for interaction situations in Non Playable Characters (NPC). These three types of algorithms will be monitored by using the unity profiler software and the Unity Console Debugger to see the response time on the use of pathfinding scripts and CPU resource usage(Alvin et al., 2022).

2. LITERATURE REVIEW

In the previous research, the combination of the A* pathfinding and waypoint algorithms with a finite state machine was successfully carried out. But haven't used a navigational mesh that will be used as pathfinding(Alvin et al., 2022).

In other studies that have been done before, comparing the A* algorithm and the navigational mesh with the results of the A* algorithm which is better than the Navigational mesh. But it has not been combined with a Finite State Machine which will be used as a description of the behavior and working principles of the system(Farrel et al., 2022).

In another study, he has calculated the response time to the A* algorithm, the results of which apply the A* algorithm to affect the travel time required for the NPC to the player's position(Safira et al., 2021).

3. METHOD

The research process is carried out with steps arranged for the smooth running of the research process and reducing obstacles. The steps in designing the SPY In LIFE video game are; determine tools for application design, literature review, application development process, and application testing process.

Device Requirements

Analysis of application requirements is needed to process a project or application that will be made when making games. Following are the hardware and software specifications used in making this application.

Table 1 Environmental Needs Analysis

Hardware:	Intel Core I3 9100F 3.60GHz 20 GB RAM Nvidia GeForce GTX 1650 Super 4 GB SSD 1Tb
Software:	Adobe Photoshop 2023, Adobe Illustrator 2023, Adobe Premiere Pro 2023, Adobe After Effect 2023, MediBang Paint Pro, Studio One 6, Autodesk Maya 2023, Unity 2021.3.12f1, dan Microsoft Visual Studio 2019
operating system:	Windows 10

Unity Engine

Unity engine is a game engine software used to develop 3D or 2D-based games. Even though it has the basic programming language C#(Goldstone, 2009). Unity Engine can support various programming languages, from C++, JavaScript, to Python(Goldstone, 2009). Researchers will use the C# programming language. Unity Engine also supports various platforms ranging from Android, IOS, PS3, PS4, PS5, XBOX, and Windows-based computers.

A* Algorithm

The A* algorithm is the most popular algorithmic technique to use because of its accuracy and performance. This algorithm is used to find out the path between the two shortest nodes(Safira et al., 2021). This algorithm has been used in many video game genres such as RTS (Real Time Strategy) RPG (Role-Playing Games), Racing games, and turn based strategy games. This algorithm was first developed by Hart, Nilsson and Raphael in 1967 to solve many problems(Alvin et al., 2022). In the context of video games. This algorithm is often used and

*name of corresponding author

implemented in NPC (Non Player Character) AI to guide finding a way or to catch players. The visualization of the A* algorithm can be seen in Figure 1 as follows:

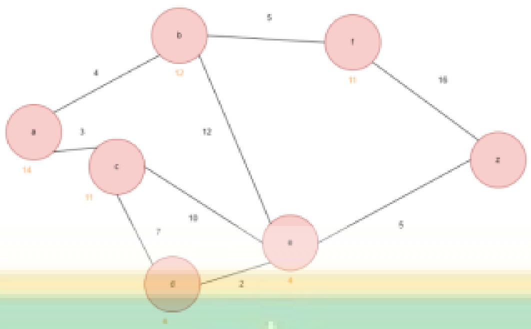

Figure 1. A* Pathfinding

Numbers marked in orange are heuristic numbers and numbers marked in black are weight numbers. The A* algorithm is included in the heuristic algorithm because of its characteristics to keep looking for the smallest value to provide the most optimal solution. A* uses the heuristic function f(n) to determine the nodes, and the value of f(n) is obtained by the formula(Febriyanto Riski et al., 2020).

$$f(n) = g(n) + h(n) \qquad (1)$$

The function $g(n)$ is the price needed to reach the node target of the starting node. The $g(n)$ function will calculate the price that has been obtained to reach the target node. The $h(n)$ function is a heuristic value used for estimation from node to target node. When the grid has a resistance $f(n)$ it will take the estimate and take the lowest price to get optimal results.

Navigational Mesh

Navmesh (Navigation Mesh) is a pathfinding algorithm that has been integrated by the Unity Engine. Navmesh consists of several different components, these components consist of the navmesh surface, and the navmesh agent. The navmesh surface is the path that will be generated automatically by Navmesh when the user bakes the area you want to make as a path(Farrel et al., 2022). The following visualization of the implementation of Navmesh can be seen in Figure 2 below.

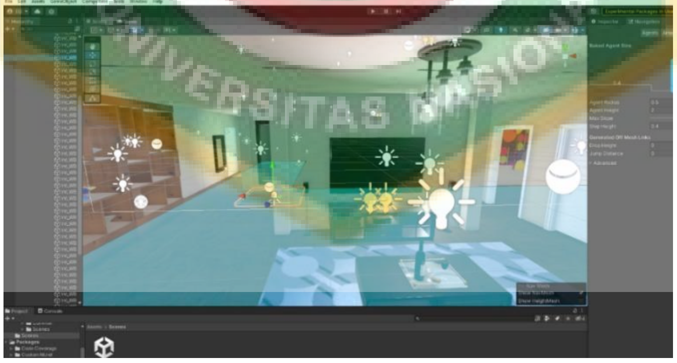

Figure 2. display of the polygon map that has been generated by navmesh

*name of corresponding author

Finite State Machine

Finate State Machines (FSM) is a control system design methodology that describes the behavior or working principles of the system using states, events and actions. In other words, the system will enter a state or state when it gets certain input actions or events(Pukeng et al., 2019). As shown in Figure 3 as follows:

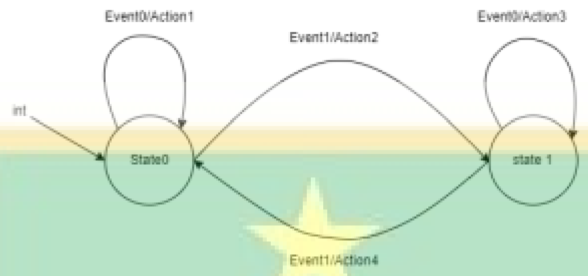

Figure 3. Basic State Machines

Figure 6 shows 2 states and two inputs and four different outputs. When the game is run it will transition to State0, in the state the system will produce action1 if an event0 input occurs, whereas if event1 occurs then action2 will be executed then the system will transition to State1 and so on.

Planning

At the design stage of the Video Game SPY In LIFE is covered with making all the structures of the game that will be displayed in the final product.

1. Title Page

   This Stealth Game uses the logo in Figure 7. This game takes the title "SPY In LIFE" which can be played on Desktop Devices with the Windows operating system.

2. Game Flow

   The game flow or plot of the video game being developed is summed up in a flowchart in Figure 4 below.

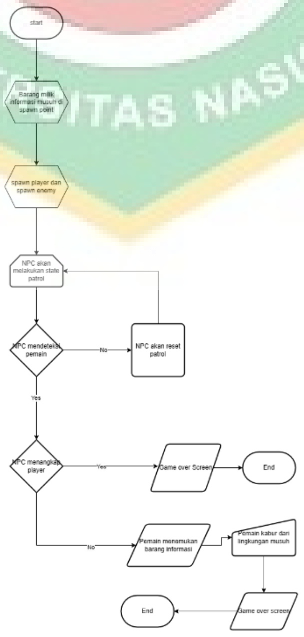
start
Barang milik informasi musuh di spawn point
spawn player dan spawn enemy
NPC akan melakukan state patrol
NPC mendeteksi pemain
No
NPC akan reset patrol
Yes
NPC menangkap player
Yes
Game over Screen
End
No
Pemain menemukan barang informasi
Pemain kabur dari lingkungan musuh
Game over screen
End

Figure 4. Flowchart of the GameFlow

*name of corresponding author

Players will be placed in a location that is safe from NPCs at the start of the game. Players are assigned to find objectives or items and look for exits after completing all the existing objectives. when the player successfully completes all objectives and successfully escapes the win game display will appear. and when the player fails and is caught by the enemy, the game over display will appear.

3. Layout Level

Layout level is a design how a level will be set in a video game. The level to be created will have 2 levels. The level design was made using Autodesk Maya 3D software which was made in the initial stages using traditional images.

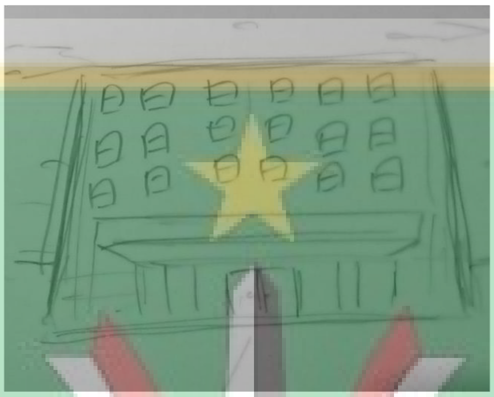
Figure 5. level 1 concept

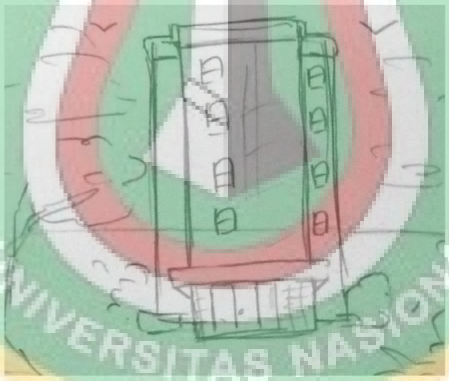
Figure 6. Level 2 concept

4. Artificial Intelligent

The design of AI NPC (Non Player Character) is designed using the FSM (Finite State Machine) method which can be seen in Figure 7.

$$\begin{tikzcd}
\text{Patrol} \arrow[loop left, "\text{patroling to waypoint}"] \arrow[r, bend left, "\text{Player Found}"] & \text{Chase} \arrow[l, "\text{Player Escaped}"'] \arrow[d, "\text{Player is captured}"'] \\
& \text{Capture} \arrow[loop below, "\text{Game Over Screen}"]
\end{tikzcd}$$

Figure 7. Finite State Machine from the AI NPC game SPY In LIFE

In Figure 11 is the design of the AI NPC state machine, there are 3 states; namely state patrol, chase, and capture. In state patrol, AI will head towards the waypoint that has been placed according to the pattern given. The AI will repeat the activity until the AI sees the player. When the AI sees that the player will change to a state, the Chase will chase the player continuously until the player manages to escape from the AI, but if the AI catches the player, it will move state to capture which results in a game over screen, which means the player has failed to complete the objective. When the AI doesn't catch the player, it will return to Patrol state. this condition will continue to repeat until the end of the game and the player manages to escape from the level.

Video Game Development

All data obtained from the literature review and previously designed mechanics will be tested for implementation, so that the next stage is the development stage by combining all designs in terms of level, mechanics, UI (User Interface) and the algorithm to be used(Schell, n.d.).

Testing

The testing process is carried out to ensure that each the designs and algorithms that have been implemented and developed in the game have worked as intended. At this stage, in addition to running the game fully according to the objectives that have been determined, the process of monitoring resources on the device is also carried out using the unity console and MSI afterburner. This test is conducted to determine the performance of the algorithm applied and the performance of the application.

4. RESULT

In the results stage, all designs made previously have been developed into a game unit and have been exported into an .exe file which can be installed and run on a desktop platform with the Windows 10 operating system. The testing stage is carried out by running the application and running benchmark software and looking at the whole performance of the application.

Devices Used

To carry out the test, an appropriate environment is needed to run the test. In this research test. The following table specs the environment used to run tests on the developed Escaped video game.

Table 2 Environment Used

VGA	Nvidia GeForce GT 920M 2 GB
PROC	Intel Core I5-5200U 2.2GHz
MEMORY	8GB DDR3L
HARDISK	500 GB
OS	Windows 10
DirectX	Ver 11
Testing Software	Unity Profiler and Stats viewer, MSI Afterburner

*name of corresponding author

Application View

1. Main Menu View

   The main menu is the first display that players will be able to interact with in the video game implementation as can be seen in Figure 8.

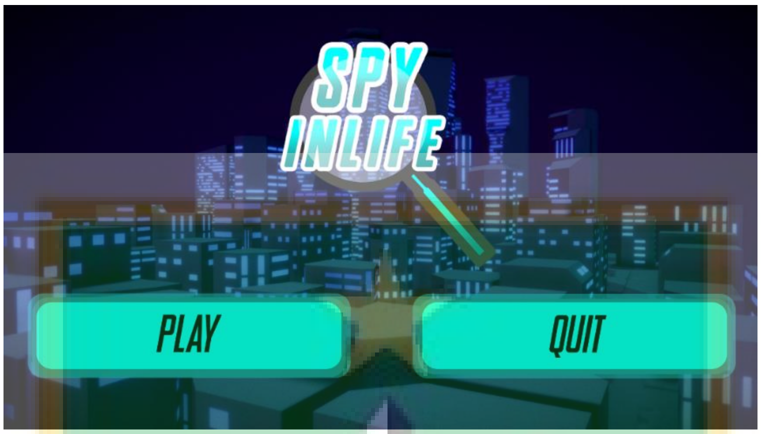

Figure 8. Main Menu View

   The main menu displays the title of the game as well as 2 buttons that players can press which will take them to different functions. When the player presses the "Quit" button, exits the application and presses the "Play" button, they will be directed to the level selection menu.

2. Game View

   The gameplay display is the main view of the mechanics that players will use to complete the given objectives. The objective given in the video game "SPY In LIFE" is to find items to serve as evidence of the crimes the officials have committed. There are many items used as evidence of a crime, namely briefcases, laptops, documents, oil, cellphones, and a photo. After collecting the objectives that have been designed, the player must find a way out. Players can walk using the “W”, “A”, “S”, and “D” keys which indicate a direction respectively. The "E" button functions to take objective items, the "P" button to pause the game, the "Shift" button to run, the "Space" button to jump, and the "Ctrl" button to crouch. As well as using the mouse to look around in the game.

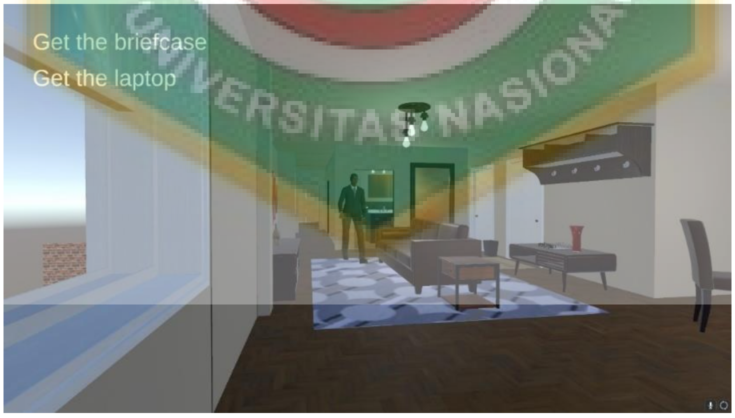

Figure 9. Gameplay View

3. Win View

   The win display is a display that will appear when the player successfully completes the objective of the game. The menu that will display the win display is in the form of 2 buttons, namely "Restart" and "Main menu". The restart button will load the game scene starting from the beginning again. The main menu button will load the main menu.

*name of corresponding author

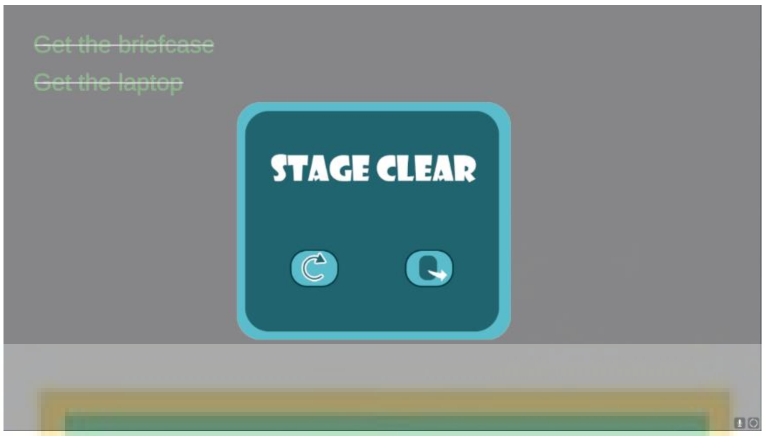

Figure 10. Win View

4. Game Over View

The Game Over display is the display that will appear when the player has been captured by an enemy NPC. The displayed menu will stop the game and bring up 2 buttons, the "retry" button will reload the level and reload the existing game scene. The "home" button will bring the player to the main menu scene.

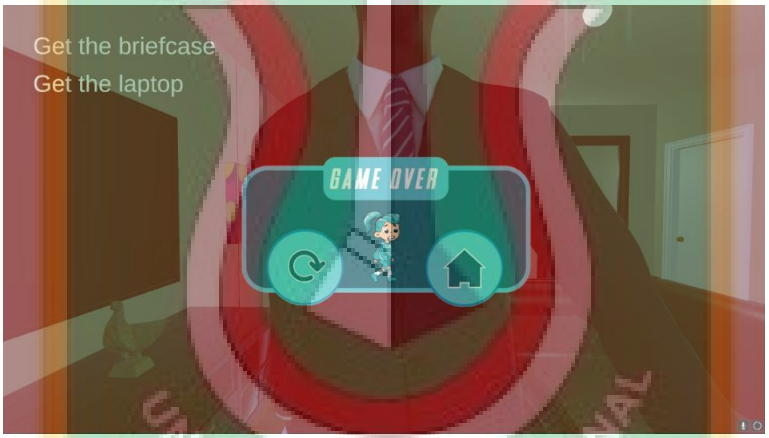

Figure 11. Game Over View

Algorithm Testing Results

The testing process will be divided into 2 stages, namely with the Unity Profiler benchmark and using MSI Afterburner. The Unity Profiler has a complete function to view resource usage from running games, processes in the form of render time, script response time, etc.

Table 3 Usage on the A* pathfinding waypoint FSM level 1 script

Frame:	143/300
Total:	2.1%
Self:	1.7%
Calls:	2
GC alloc:	1.2Kb
Time ms:	0.82
Self ms:	0.66

Testing at level 1 is carried out by looking for the highest resource spike in the use of the finite state machine script from 300 data frames recorded in application testing. In table 3 there are different types of measurements. The frame section has a value of 143/300 which indicates that out of all 300 frames recorded, the most resource usage is in frame 143. The total measures the length of time spent by Unity. total measures the length of time spent

*name of corresponding author

by unity on the function and is written as a percentage. The results obtained in this total measurement are 2.1% of the total running function, this is quite large considering that it only occurs in one frame.

Self measures the amount of time used by unity to run the functions of the script but does not measure the amount of time used by unity to call the functions of the script, the measurement results are also calculated as a percentage. From the results obtained from table 3. unity spends a different amount of time than the total, which is 1.7% and the results are smaller than the total, but it's still quite large because Unity only runs the script for one frame. Calls measure how many functions this script is called in one frame. In the Calls section in table 3, the finite state machine script function is used 2 times because it calls 2 functions in the first update, namely the path waypoint and field of view from AI.

GC alloc measures how much memory of the script has been allocated by unity. From this test the results obtained from GC alloc are 1.2 Kb, this shows that the finite state machine script eats up the memory in unity. Time ms measures the total time spent by unity on the script function and is calculated in milliseconds. From the test results in table 3. Time ms has a time value of 0.82 ms in one frame. Self ms has a measurement system similar to Time ms, but Self ms does not calculate the sub-function functions that are called on the finite state machine script. The measurement results in table 4 are 0.66 less than the time value of Time ms.

Table 4 Usage on the A* pathfinding waypoint FSM level 2 script

Frame:	240/300
Total:	5.1%
Self:	5.1%
Calls:	4
GC alloc:	2.5Kb
Time ms:	0.83
Self ms:	0.83

Then testing at level 2 is carried out by looking for the highest spike resource in the finite state machine script from 300 data frames recorded in application testing. In table 4, the frame section has a value of 240/300 which indicates that out of all 300 frames recorded, the most resource usage is in frame 240. The results obtained for this total measurement are 5.1% of the total running functions. These results indicate getting better results. from the test results in Table 4 level 1.

Self which functions to measure the amount of time in unity has a total of 5.1%, the result is the same as the total. The result obtained by Calls is 4. This is due to having 4 AIs, more AIs will make the number of functions called. On GC alloc has 2.5 KB of memory used. From the test results in Table 4. Time ms has a time value of 0.83 ms in that one frame. Self ms has the same value as Time ms which is equal to 0.83 ms.

The next step in testing this application is to benchmark the MSI Afterburner application. Benchmarks are useful for seeing the performance of running applications. The following are the results of the SPY In LIFE application benchmark:

Table 5 Benchmark SPY In LIFE Level 1

Rendered Frame:	2424 Frames
Rendered Frames Complete Time:	81.281s
Average Framerate:	29.8 FPS
Maximum Framerate:	60.5 FPS
1% Low Framerate:	0.5 FPS
0.1% Low Framerate:	0.5 FPS

The benchmark results in table 5 have the same trial parameters as the SPY In LIFE game which only uses the Finite State Machine algorithm which includes all algorithms in it. From the trial process, the results are as shown in table 5. In the Rendered Frame section, the rendered frame produced when benchmarking is 2424 frames. Remdered Frames Complete Time is the time taken when taking frames that have been benchmarked which has a result of 81,281 s. Next is the Average Framerate that is obtained when doing benchmarks, namely 29.8 FPS. The Maximum Framerate section has a result of 60.5 FPS which means a high framerate when running games. The parameter 1% Low framerate has a result of 0.5 FPS and 0.1% Low Framerate has a result of 0.5 FPS.

Table 6 Benchmark SPY In LIFE Level 2

*name of corresponding author

Rendered Frame:	4991 Frames
Rendered Frames Complete Time:	232.015s
Average Framerate:	21.5 FPS
Maximum Framerate:	60.8 FPS
1% Low Framerate:	0.3 FPS
0.1% Low Framerate:	0.3 FPS

Benchmark results in table 6 level 2 have greater results than benchmark results carried out in table 6 level 1. The results of this test are due to the large number of Assets and AI provided at level 2. The Render Frame generated in the level 2 testing trial has as many as 4991 frames. The resulting Rendered Frames Complete Time is 232.015 s. Next is the Average Framerate you get, which is 21.5 FPS. The average framerate results obtained are smaller than the results given in table 6 level 1, this is fairly reasonable considering the total result of using the script is very high. The Maximum Framerate section has a result of 60.8 FPS, which means a high framerate when running games. The 1% Low framerate parameter has a result of 0.3 FPS and 0.1% Low Framerate has the same result, namely 0.3 FPS.

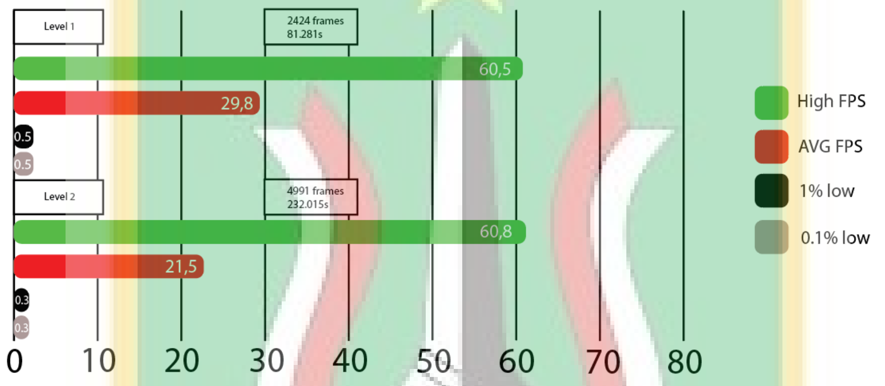

Figure 16. Comparison of benchmark results at both SPY in LIFE game levels

Figure 16 is a graph comparison of the two levels in the game SPY In Life. The benchmark results in Figure 16 have 4 colors that represent different parameters. The green color represents the highest FPS obtained in the game application. The red color represents the average FPS obtained in the results of the game application. The black color represents the number of frames that can be rendered by the computer at a slow speed of 1%. The gray color represents something similar to black but with a slower speed of 0.1%.

From the benchmark results in Figure 16 it shows that at level 1 the highest FPS result is 60.5 FPS, while level 2 has almost the same result as level 1, namely 60.8 FPS. On average FPS obtained, Level 1 gets an average FPS of 29.8 FPS and for Level 2 it has a lower result of 21.5 FPS. The difference in the average FPS at level 1 and level 2 is quite a lot, but this is considered reasonable because it has a difference in the results obtained when testing the usage script.

At the 1% low parameter, at Level 1 you get an FPS value of 0.5 FPS, while at Level 2 it has 0.3 FPS, the difference between the two results at Level 1 has a bigger FPS result. The last one is the 0.1% parameter where at Level 1 you get an FPS of 0.5 FPS and at Level 2 you get the same FPS, which is 0.3 FPS.

NPC State Machine Test Results

Testing on State Machine NPC regarding functions that have been programmed to AI in the game. Testing on the NPC State Machine is carried out by manual testing by carrying out all activities that can change the state of the AI. The test results of the State Machine NPC can be seen in table 7.

*name of corresponding author

Table 7 State Machine Test Results on AI NPC

State	Condition	Action	Result
Patrol	Bool canSeePlayer is false	AI will go to the predetermined waypoint	Succeed
Chase	Bool canSeePlayer is true	Chasing players according to the position of the player	Succeed
Capture	Enemy and player is Colliding	Game over screen and the game will immediately stop	Succeed

DISCUSSIONS

Researchers use the A Star Navigational Mesh and Finite State Machine algorithms to create AI in a Stealth game entitled "SPY In LIFE". The first process is the research flow starting from looking at device requirements and software requirements that will be used to make this stealth genre game that will be made and the configuration of each algorithm used. When processing there are UI Main menu, Level Select, Background Story, Intro Video and Outro Video, concept maps for Levels 1 and 2 as well as AI which will be created using a combination of the A Star Navigational Mesh Algorithm and the Finite State Machine. To find the results, researchers will use the Unity Profiler software on the Unity Engine and use MSI afterburner to see the benchmarks for Game SPY In LIFE. The results on CPU usage at each level have quite a difference because each level has different amounts of Assets and AI. The result of CPU usage obtained at Level 1 is 2.1% while at level 2 is 5.1%. This also results in a difference in the average FPS obtained at each level when the SPY In LIFE game benchmark uses MSI Afterburner. The average FPS results obtained at level 1 are 29.8 FPS while Level 2 is 21.5 FPS. The FPS results obtained are quite small because the large CPU usage makes the game very heavy, in the sense that the device specifications used for testing do not run optimally.

5. CONCLUSION

Based on the SPY In LIFE Video Game test results obtained from manually monitoring the A* Navigational Mesh algorithm using the Finite State Machine on AI NPC, there are conclusions that can be drawn: 1. with a combination of the A* Navmesh Algorithm using a finite state machine. NPCs can explore levels that have been made without colliding with other 3D Assets that are considered to be blocking the intended path. NPCs can also chase players dynamically when players move without any problems, 2. Monitoring results show that the combination of the A* Navmesh algorithm using a Finite State Machine uses CPU resources at a level of 2.1% and at Level 2, which is 5.1%, which is quite large even though it only covers 1 frame, 3. The results obtained from the benchmarks performed on Game SPY In LIFE saw that the CPU resource results were quite large which greatly affected the performance of the game. The SPY In LIFE Level 1 game managed to render 2424 frames in 81.281s, has an average FPS of 29.8 FPS and has the highest FPS, namely 60 FPS. At Level 2, it manages to render 4991 frames in 232.015s, has an average of 21.5 FPS and has the highest FPS, namely 60 FPS. The difference in the test results is caused by the difference in the number of assets and AI provided. In this game, it can be interpreted that the device used is not optimal in running the game. The development of Artificial Intelligent (AI) for Enemy that chases players has been successfully developed using the Finite State Machine method and can run well as expected.

6. REFERENCES

Astuti, D., Anggi, A. F., Kusuma, A., & Syah, F. (2022). Application of the Finite State Machine Method to Determine the End of the Story Based on User Choice in Multiple Role Playing Games. In *International Journal of Computer and Information System (IJCIS) Peer Reviewed-International Journal* (Vol. 03). https://ijcis.net/index.php/ijcis/index

Darmawan, A. P., & Rubhasy, A. (n.d.). KOMBINASI ALGORITMA A STAR DAN FINITE STATE MACHINE PADA AI ENEMY GAME HORROR 3D ESCAPED. *Kumpulan JurnaL Ilmu Komputer (KLIK)*.

Farrel, Y., Aris Gunaryati, dan, Teknologi Komunikasi dan Informatika, F., Nasional Ps Minggu, U., Jakarta Selatan, K., & Khusus Ibukota Jakarta, D. (n.d.). *Farrel, Fauziah, dan Gunaryati-Perbandingan Algoritma Kombinasi Antara Algortitma A* Pathfinding Navmesh dengan Algoritma Waypoint Pathfinding Pada Game Tower Defense Rise of Machine KOMBINASI ALGORITMA A* PATHFINDING NAVIGATIONAL MESH DENGAN WAYPOINT PATHFINDING PADA GAME TOWER DEFENSE RISE OF MACHINE*.

Febriyanto Riski, E., Triayudi, A., & Mardiani, E. (2020). Implementation Of A Star Algorithm In Android Based Alien Shooter Games. *Jurnal Mantik*, *4*(1). https://iocscience.org/ejournal/index.php/mantik

Goldstone, Will. (2009). *Unity game development essentials*. Packt Pub.

Pukeng, A. F., Fauzi, R. R., Lilyana, Andrea, R., Yulsilviana, E., & Mallala, S. (2019). An intelligent agent of finite state machine in educational game "flora the Explorer." *Journal of Physics: Conference Series*, *1341*(4). https://doi.org/10.1088/1742-6596/1341/4/042006

Safira, L., Harsadi, P., & Harjanto, S. (2021). Penerapan Navmesh Dengan Algoritma A Star Pathfinding Pada Game Edukasi 3d Go Green. *Jurnal Teknologi Informasi Dan Komunikasi (TIKomSiN)*, *9*(1), 17. https://doi.org/10.30646/tikomsin.v9i1.540

Schell, J. (n.d.). *The Art of Game Design: A Book of Lenses*.

*name of corresponding author