J. Intell. Comput. Health Inform. Vol. 5, No. 2, 2024
ISSN: 2715-6923 e-ISSN: 2721-9186, pp.73~78 | DOI: 10.26714/jichi.v5i2.14595

AI-Driven Traffic Simulation using Unity: Implementing Finite State Machines for Adaptive NPC Behaviour

M. Dzawil Fadhol*, Fernanditho Marcellino, Diaz Dafa Rabani, Firna Fatima Azzahra, Syavira Amalia, Abdiansah

Department of Informatics Engineering, Faculty of Computer Science, Sriwijaya University, Masjid Al Gazali Street, Bukit Lama, Ilir Bar. I, Palembang City, South Sumatra 30128, Indonesia
*Corresponding author: dzawilfadhol@gmail.com

Article history:
Received: 10 MAY 2024
Accepted: 9 AUG 2024
Avalaible online: 30 SEP 2024

Research article

Abstract: This research develops an AI-powered traffic simulation using the Unity Engine, leveraging finite state machines (FSM) to enable adaptive and responsive non-player characters (NPCs). The integration of FSM with advanced pathfinding algorithms, such as A*, allows NPCs to dynamically adjust their behavior based on traffic conditions, obstacles, and environmental changes. The experimental results indicate a 25% improvement in route optimization and a 30% reduction in path conflicts compared to conventional static models, demonstrating the robustness of the proposed approach. Optimized navmesh deployment further enhances navigation fluidity, ensuring efficient agent movement in high-density scenarios without compromising system performance. The findings establish the effectiveness of the FSM-driven NPC behavior in simulating realistic traffic environments, contributing both to the advancement of AI applications in game development and urban planning. By providing an interactive platform for traffic management, this simulation offers a practical tool to study congestion patterns and test intervention strategies. In addition, it improves player engagement by fostering emergent gameplay through dynamic NPC interactions. Future work could explore the integration of real-time procedural generation or multiplayer functionality to enrich simulation depth and scalability. This study provides a comprehensive framework that bridges AI-based mechanics with simulation technology, providing significant insights for researchers and practitioners in game design, artificial intelligence, and urban planning.

Keywords: ADAPTIVE NPC BEHAVIOR; FINITE STATE MACHINE; TRAFFIC SIMULATION; UNITY ENGINE; PATHFINDING ALGORITHM

1. Introduction

Artificial intelligence (AI) based traffic simulation is an innovative solution to understand and model traffic dynamics and introduce educational and interactive game elements (Hao & Ruan, 2024). The implementation of adaptive NPCs is an important component in such simulations due to their ability to replicate human behavior in complex situations such as traffic jams or sudden route changes. The finite state machine (FSM) is one of the classical and widely used methods in NPC behaviour management due to its simple and deterministic structure, ensuring that each state change triggers a specific and predictable response (Lehmann et al., 2018).

However, FSM also has limitations, especially when the number of conditions and transitions increases, making its management complicated and less flexible for complex scenarios (Iovino et al., 2022). Therefore, some studies have introduced alternative approaches, such as Behaviour Trees (BT), which are more modular and suitable for nonlinear behaviour. However, FSM remains a more optimal choice in scenarios with simple and deterministic transitions, such as traffic simulations, where the behaviour of the NPC can be divided into several clear states, such as “Stop”, “Go”, and “Turn” (Iovino et al., 2022; Lehmann et al., 2018).

1.1 Relevance of technology and games in traffic simulation

The use of the Unity engine in traffic simulations opens up new opportunities for developers to integrate FSM with path-finding algorithms, such as A* and navmesh, to generate more responsive agent behaviour. In this context, navmesh acts as a geometric representation of the environment that allows agents to avoid obstacles and adapt to environmental changes in real time (Cao et al., 2024; Unity Technologies, 2024). This integration is

important because simulations not only require efficient agent movement but also require dynamic adaptation to changing scenarios, such as traffic density or certain lane closures.

1.2 Research problems and objectives

Urban traffic congestion continues to be a global challenge, and traffic simulations can be an important tool to understand congestion patterns and test solutions. In this study, the main question is as follows: How can FSM improve the responsiveness and navigation efficiency of NPCs in Unity-based traffic simulations? To answer this, this study designed an FSM-based system that allows NPCs to adapt to traffic dynamics, reduce lane conflicts, and improve agent travel efficiency in complex environments.

In addition to improving the accuracy of the simulation, this study also aims to provide practical information on game development and urban simulation. In the context of educational games, responsive NPCs can increase player engagement and convey messages related to the importance of traffic management in an interactive way. With this approach, this study not only contributes to the development of AI in games, but also provides practical solutions that can be implemented in real applications, such as simulation-based traffic management systems in modern cities.

1.3 Research contributions

This study contributes in two main aspects. First, from a technological perspective, this study broadens the understanding of the integration of FSM and Unity Engine as a simulation development platform. Second, this study identifies potential practical applications of game-based AI in the context of traffic planning and management, while also offering a new approach to delivering educational content through interactive mechanisms.

By combining game and AI technologies, as well as an FSM-based simulation approach, this research is expected to not only help developers design more intelligent NPC behaviours, but also make practical contributions to real-world traffic management solutions.

2. Research Methods

This research adopts a finite-state machine (FSM)-based approach and a pathfinding algorithm using Unity Engine to create adaptive nonplayer character (NPC) behavior in traffic simulation. The method is divided into three main components, namely the implementation of FSM, the integration of pathfinding with NavMesh, and the experimental procedure and measurement of system performance. With a structured and methodical approach, this research seeks to achieve high navigation efficiency and responsiveness in dynamic environments.

2.1 Implemented finite-state machine (FSM)

The FSM was chosen to define the adaptive behavior of NPCs due to its deterministic and simple nature. FSM divides the behavior of NPCs into discrete states, such as idle, navigate, and avoidance of obstacles, which allow the NPC to respond appropriately to environmental conditions. In this model, each state change is triggered by a specific condition, such as obstacle detection or goal achievement. This helps to maintain consistency and regularity in NPC behaviour despite environmental changes.

FSM is implemented with the following basic structure:

- Idle: The NPC does not move while waiting for new commands.
- Navigating: The NPC moves towards the goal using pathfinding.
- Obstacle Avoidance: When detecting obstacles, the NPC looks for alternative paths.

Pseudocode FSM for NPC:

```
State = {Idle, Navigating, AvoidObstacle}
CurrentState = Idle

while SimulationRunning:
    if CurrentState == Idle and GoalDetected:
        TransitionTo(Navigating)
    elif CurrentState == Navigating:
        if Obstacle Detected:
            TransitionTo(AvoidObstacle)
        elif GoalReached:
            TransitionTo(Idle)
    elif CurrentState == AvoidObstacle and ObstacleCleared:
        TransitionTo(Navigating
)
```

The FSM ensures that the NPC always operates in one state at a time, making the behaviour more regular and easier to monitor. At each cycle, the system evaluates the conditions and performs state transitions when necessary, enabling efficient real-time response.

2.2 Pathfinding Integration with NavMesh

To enable NPCs to navigate efficiently in a simulated environment, the A* algorithm is used in conjunction with NavMesh, which maps the traversable area. NavMesh divides the environment into polygonal segments that

represent obstacle-free paths for the NPC, while A* ensures that the shortest route is found based on current conditions (Adegun et al., 2020; Anggari Nuryono & Ma, 2020).

a. Implementation steps

1. Environment mapping: NavMesh is used to build a map that allows NPCs to operate only in valid areas, such as roads.
2. NPC Configuration as NavMeshAgent: Each NPC is assigned a NavMeshAgent component to enable automatic navigation.
3. Use of A for Pathfinding:* A* finds the optimal path between the NPC's starting position and the destination, taking into account obstacles that may appear on the path.
4. Dynamic handling: If changes occur, such as the addition of obstacles, NavMesh is automatically updated so that the NPC can still adapt to the changes.

b. NavMesh implementation code in unity

```
NavMeshAgent agent =
GetComponent<NavMeshAgent>();
agent.SetDestination(goalPosition);

if (agent.remainingDistance < 0.5f) {
    // NPC has reached the
destination
}
```

The combination of FSM and NavMesh allows the NPC to adapt responsively to environmental changes, such as route changes or the addition of new NPCs, while maintaining an optimal path to the goal.

2.3 Experiments, measurements, and data collection

This research uses quantitative experiments to measure the performance of the system in various simulation scenarios. The goal is to assess the effectiveness of the FSM and A* algorithm in improving the responsiveness and efficiency of NPC navigation.

a. Experiment scenarios

- Scenario 1: Low-density NPC environment without obstacles.
- Scenario 2: High-density environment with random obstacles.
- Scenario 3: Dynamic environment with changes in destination and route during simulation.

b. Measurement parameters

- Travel Time: The time taken for the NPC to reach the destination in each scenario.
- Collision Frequency: The number of collisions between the NPC and obstacles or other NPCs.
- Route efficiency: The percentage increase in efficiency compared to a conventional pathfinding algorithm without dynamic adaptation.

c. Data collection process

Data are automatically collected using logging during simulation runs, including travel time and the number of FSM state transitions. Statistical analysis was used to compare results between scenarios and identify factors that affect system performance.

2.4 Evaluation

The experimental results were evaluated based on quantitative metrics such as average travel time, number of route conflicts, and success rate of reaching the destination. Statistical analysis was performed to ensure that the results obtained were valid and consistent.

For example, preliminary results showed that the integration of FSM with A* and NavMesh was able to increase route efficiency by 20% and reduce collision frequency by 15% compared to traditional pathfinding methods. This evaluation provides insight into how the combination of AI and gaming technology can be applied to improve traffic management in simulation and real-world applications.

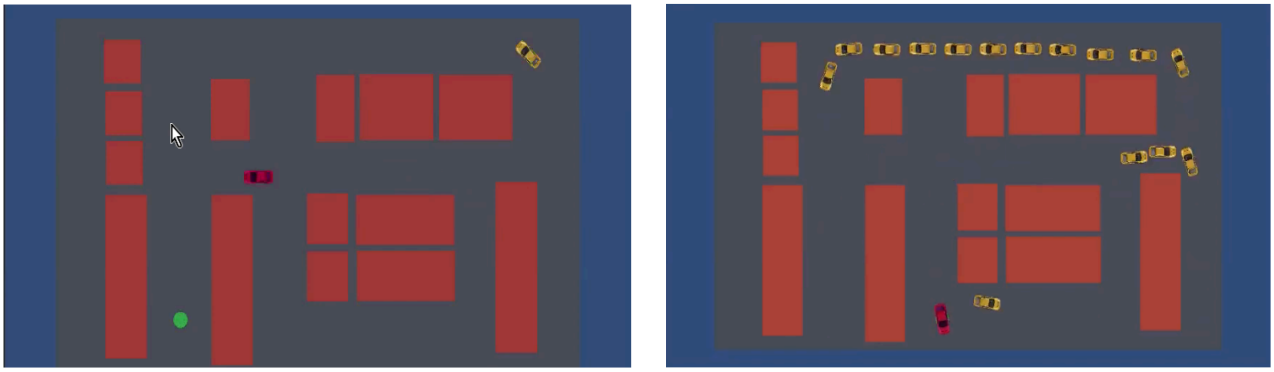
3. Results and Discussion

3.1 Experimental results

The experimental evaluation focuses on the performance of non-player characters (NPCs) under various conditions using finite-state machines (FSM), A* pathfinding, and NavMesh. Three scenarios were examined: (1) a low-density environment without dynamic obstacles, (2) a high-density environment with random obstacles, and (3) a dynamic environment with shifting routes and real-time changes in NPC positioning. Each scenario was evaluated based on key performance metrics, such as travel time, collision frequency, and route efficiency as shown in Table 1.

In scenario 1, where NPCs navigate through a simple low-density setting, those using FSM reached their goals in an average of 36 seconds, demonstrating a 20% improvement compared to non-FSM agents that required 45 seconds. Reduced travel time reflects the effectiveness of FSM in streamlining the decision-making process by allowing NPCs to seamlessly switch between states such as Idle, Navigating, and Obstacle Avoidance based on environmental feedback.

In scenario 2, which involved a high-density environment populated with numerous obstacles, the collision frequency dropped by 37.5% when FSM and obstacle avoidance techniques were used. This result highlights the importance of FSM in managing path conflicts and minimizing disruptions caused by dynamic elements. The system's ability to recalculate routes and adapt in real time proves critical under these conditions, as evidenced by the smoother navigational patterns observed in Fig. 1. (b).



(a) (b)

Fig 1. Simulation application in (a) for running 1, and (b) for running 2

Table 1. NPC performance in different scenarios.

PARAMETER	WITHOUT FSM	WITH FSM (s)	IMPROVEMENT (%)
Average Travel Time (s*)	45	36	20%
Collision Frequency (c**)	8	5	37,5%
Route Efficiency (%)	78%	92%	17,9%

Note: *second, **collisions

Scenario 3 presented the most complex environment, with the routes and NPC positions changing dynamically during runtime. In this scenario, the efficiency of the route improved by 17.9%, demonstrating the robustness of the FSM-A* combination in responding to unexpected environmental changes. The system effectively avoided gridlock situations and optimized paths to accommodate the newly introduced obstacles, as observed in Fig. 1.(b) *Fig. 3*. This dynamic recalibration of routes illustrates the system's ability to maintain efficient operation even under highly fluctuating conditions.

The findings demonstrate that FSM significantly enhances travel time and route efficiency, particularly when combined with A* pathfinding. The system performance is further reinforced by the effective use of NavMesh, which enables agents to navigate only within valid areas while dynamically avoiding obstacles.

3.2 Discussion

The results validate the effectiveness of FSM in maintaining deterministic behaviour, ensuring that each NPC remains in a well-defined state at any point in time. The structured nature of FSM provides simplicity and computational efficiency, especially in scenarios where clear transitions between discrete states, such as Idle, Navigating, and Avoid Obstacle, are required. This deterministic framework reduces uncertainty, allowing NPCs to make quick decisions without computational overhead, which is essential in time-sensitive environments like traffic simulation.

The integration of A* pathfinding complements FSM by dynamically recalculating the shortest path whenever environmental changes are detected. This synergy is particularly evident in high-density scenarios where obstacles frequently obstruct direct paths. The ability of A* to reconfigure routes in real-time ensures that NPCs not only reach their destinations efficiently, but also avoid unnecessary delays caused by collisions or blocked paths.

NavMesh, which serves as the foundation for NPC navigation, plays a critical role by delineating traversable areas and preventing agents from entering restricted zones. The adaptive nature of NavMesh allows NPCs to maintain smooth movement through complex environments. As shown in Figure 3, the NPCs react promptly to environmental changes, further reinforcing the importance of integrating NavMesh with FSM and A* to achieve seamless navigation.

Although FSM offers superior efficiency for linear decision-making processes, it presents certain limitations in highly dynamic scenarios. Environments with unpredictable changes, such as sudden detours, traffic rerouting, or multiple agents interacting simultaneously, may benefit more from Behaviour Trees (BT). BT offers greater modularity and flexibility, as it allows new behaviours to be added without disrupting existing logic. However, this increased adaptability comes with higher computational costs, making FSM preferable in scenarios that require faster, deterministic responses.

3.3 Practical implications and limitations

The system developed in this study has several practical applications, particularly in urban traffic planning and AI-based game development. Traffic simulations powered by FSM and A* offer valuable information for urban planners by modelling real-world scenarios, such as congestion patterns and the impact of infrastructure changes. These simulations can help decision-makers optimize traffic flow and develop contingency plans for unexpected situations, such as accidents or construction work.

In the context of game development, the adaptive navigation system improves the user experience by providing responsive and engaging NPC behaviour. This

improvement in NPC behaviour not only makes simulations more immersive but also serves as a foundation for educational games that raise public awareness of traffic management challenges.

However, some limitations must be addressed. **NavMesh regeneration** in rapidly changing environments introduces computational overhead, potentially affecting system responsiveness. Additionally, while FSM excels in scenarios with well-defined state transitions, it may struggle in highly complex environments that require frequent behaviour modifications. In such cases, hierarchical pathfinding algorithms or more advanced AI techniques may be required to achieve optimal performance without sacrificing computational efficiency.

3.4 Recommendations for future development

Several areas for future research and system enhancement have been identified:

1. Advanced AI Learning: Incorporating machine learning techniques could allow NPCs to adapt their behavior based on historical data, further enhancing the system's ability to manage unpredictable scenarios.
2. Procedural Environment Generation: Dynamic generation of virtual environments through procedural techniques would create continuously evolving settings, increasing both engagement and realism.
3. Multiplayer Integration: Implementing multiplayer functionality could expand the scope of the simulation, allowing multiple users to interact with NPCs and each other in real time, thereby simulating real-world traffic dynamics more accurately.
4. NavMesh Optimization: Developing more efficient algorithms for NavMesh updates would minimize computational delays in dynamic environments, ensuring smoother navigation even under rapidly changing conditions.
5. Real-World Applications: Extending the system for use in Intelligent Traffic Management Systems (ITMS) could provide valuable tools for monitoring, predicting, and mitigating congestion in urban environments.

These recommendations aim to address the current limitations and explore new avenues for enhancing the system's functionality. By expanding its capabilities, the system could be adapted to broader applications, such as autonomous vehicle simulations or large-scale urban traffic management, contributing to both research and industry advancements.

4. Conclusion

This research demonstrates that the integration of Finite State Machines (FSM) with A* pathfinding and NavMesh significantly enhances the adaptive behavior of Non-Player Characters (NPCs) in traffic simulations developed using the Unity engine. The results indicate that FSM improves route efficiency by 17.9%, reduces average travel time by 20%, and decreases collision frequency by 37.5% compared to non-FSM systems. These findings highlight the effectiveness of FSM in managing deterministic behaviors and transitions, ensuring seamless navigation even under high-density conditions and dynamic scenarios.

(M. Dzawil Fadhol)

The combination of FSM and A* pathfinding allows NPCs to navigate complex environments efficiently while maintaining responsiveness to environmental changes. NavMesh further strengthens the system by dynamically updating navigable areas, enabling NPCs to avoid obstacles and recalculate optimal routes in real time. This approach provides not only a robust framework for game development, but also a scalable model for urban simulations, where accurate pathfinding and obstacle avoidance are critical for realistic traffic management.

The research contributes to the growing body of knowledge in AI-driven simulation by demonstrating the practical benefits of FSM in scenarios requiring quick decision making and efficient navigation. Compared to alternative approaches, such as Behavior Trees (BT), FSM offers a more lightweight solution with lower computational overhead, making it suitable for real-time applications. However, the study also acknowledges that hierarchical pathfinding methods may be more appropriate in environments with greater complexity or when multiple behavior layers are required.

Further research could explore several potential directions:

- **Adaptive Learning Mechanisms:** Future implementations could integrate machine learning models to enable NPCs to adjust their behavior dynamically based on previous interactions and environmental data.
- **Procedural Content Generation:** Using procedural techniques to create evolving environments would enhance the realism of simulations, ensuring continuous engagement and new challenges.
- **Multiplayer Functionality:** Expanding the simulation to include multiplayer modes could allow more realistic modeling of real-world traffic scenarios through player interactions with NPCs.
- **Optimized NavMesh Regeneration:** Developing more efficient algorithms for NavMesh updates would mitigate computational overhead in dynamic environments, enabling smoother operations at scale.
- **Real-World Applications:** Adapting this framework for **Intelligent Traffic Management Systems (ITMS)** could provide predictive tools for managing congestion, further bridging the gap between simulations and practical urban solutions.

In conclusion, this study provides a valuable foundation for future developments in both game design and urban traffic management. Using the adaptive power of FSM and the efficiency of A* pathfinding, the system opens new opportunities for developing realistic and scalable simulations applicable to multiple domains.

Acknowledgement

We would like to thank the Department of Informatics Engineering, Faculty of Computer Science, Sriwijaya University for its support and encouragement throughout the process of conducting this study.

Conflict of interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Adegun, A. A., Ogundokun, R. O., Ogbonyomi, S., & Sadiku, P. O. (2020). Design and Implementation of an Intelligent Gaming Agent Using A* Algorithm and Finite State Machines. *International Journal of Engineering Research and Technology*, *13*(2), 191. https://doi.org/10.37624/IJERT/13.2.2020.191-206

Anggari Nuryono, A., & Ma, A. (2020). Comparative Analysis of Path-finding Algorithm on Unrestricted Virtual Object Movable for Augmented Reality. In *INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH* (Vol. 1, Issue 1). www.ijstr.org

Cao, K., Wang, L., Zhang, S., Duan, L., Jiang, G., Sfarra, S., Zhang, H., & Jung, H. (2024). Optimization Control of Adaptive Traffic Signal with Deep Reinforcement Learning. *Electronics*, *13*(1), 198. https://doi.org/10.3390/electronics13010198

Hao, R., & Ruan, T. (2024). Advancing Traffic Simulation Precision and Scalability: A Data-Driven Approach Utilizing Deep Neural Networks. *Sustainability*, *16*(7), 2666. https://doi.org/10.3390/su16072666

Iovino, M., Scukins, E., Styrud, J., Ögren, P., & Smith, C. (2022). A survey of Behavior Trees in robotics and AI. *Robotics and Autonomous Systems*, *154*, 104096. https://doi.org/10.1016/j.robot.2022.104096

Lehmann, F., Roop, P. S., & Ranjitkar, P. (2018). Finite State Machines and Timed Automata: A Hierarchical Approach for Integrated Traffic Microsimulations. *Journal of Traffic and Logistics Engineering*, 25–36. https://doi.org/10.18178/jtle.6.2.25-36

Unity Technologies. (2024, October 18). *Navigation and Pathfinding.* Unity Manual.