International Journal of Research in Engineering, Science and Management
Volume 6, Issue 6, June 2023
https://www.ijresm.com | ISSN (Online): 2581-5792

Intelligent Path-Finding Agent in Video Games Using Artificial Intelligence

Koyya Vishnu Teja[1*], Mallikarjun M. Kodabagi[2]
[1]*Student, School of Computing and Information Technology, REVA University, Bengaluru, India*
[2]*Deputy Director, School of Computing and Information Technology, REVA University, Bengaluru, India*

***Abstract*: This paper focuses on exploring the ability of artificial intelligence models (AI) in video game development for path finding. Path finding is a major component of video games that allows non-playable characters (NPCs) to navigate their complex 3D or 2D environment, to interact with the player realistically. The most frequent context is found in real-time strategy games. In recent years, AI algorithms have been extensively used for path finding and other design-specific works, offering more advanced and effective methods compared to conventional algorithms. This paper will examine the challenges and limitations of using AI for path finding in video games and the need for domain-specific knowledge.**

***Keywords*: Artificial Intelligence, Machine Learning, Reinforcement Learning, Video games, Unity, Unity ML agents.**

1. Introduction

Video games are an incredibly important part of the entertainment industry, with a global video game market of 238.4 billion USD and are expected to reach 372 billion USD in revenue by 2023. Video games have also been shown to have an impact on various aspects such as better education, defense, and simulation programs apart from entertainment. The development and advancement of hardware contributed to their growth and success, enabling more advanced game mechanics, graphics and immersive experiences. In the world of video games, artificial intelligence (AI) is about creating more responsive, adaptive, immersive, and challenging games. From NPCs to procedural character behavior, from path finding to adaptive game play mechanics, AI is revolutionizing the gaming experience. In other words, understanding the role of AI in video games is crucial for anyone looking to explore future technology.

Path finding AI model: Path finding AI is referred to computing an optimal route in a given region between the specified start and goal nodes. Recent developments in path finding lead to more improved, accurate and faster methods and still captivates the researcher's attention for further improvement and developing new methods as more complex problems arise or are being created in AI.

Path finding strategies have the responsibility of finding a path from any coordinate in the game world to another. They include a starting point and a destination; they then find a series of points together which is the path to the destination. These points are the list of positions within the game world that the AI agent is allowed to move. To date, there are many algorithms regarding path finding. The well-known path finding algorithms are A-Star, Dijkstra, Breadth First Search (BFS) and Depth First Search (DFS).

This implementation focuses on hybridized algorithms with path finding algorithms technique and machine learning and other methods, such as multi-agent systems, meta-heuristics, imitation learning and other decision-making methods. Hybridization of these algorithms is needed to maintain their effectiveness while tackling its weakness. This algorithm can be implemented in many games, especially in constructing the non-playable character (NPC) or Game Enemy AI.

2. Literature Review

Research on Artificial Intelligence Algorithm and Its Application in Games (2020) Cundong Tang, Zhiping Wang, Xiuxiu Sima, Lingxiao Zhang. With the in-depth development of intelligent technology, game artificial intelligence (AI) has become the technical core of improving the playability of a game and the main selling point of game promotion, deepening the game experience realm. Modern computer games achieve the realism of games by integrating graphics, physics and artificial intelligence. It is difficult to define the meaning of realistic game experience, but generally speaking, it usually refers to the immersion of the game and the intelligence of non-player characters in the game. As the technical core of improving game playability and the selling point of many commercial games, game artificial intelligence gives players a way to interact with non-player characters in the game, and promotes the realm of game experience to a higher level. Based on this, this paper analyzes the history and present situation of artificial intelligence in game development, and puts forward the possible changes and impacts of artificial intelligence technology based on machine learning on game development in the future.

Research on path-finding and navigation technology in environment (2021) Pie Yi Yin, Chang Yuan Li. The research of path-finding and navigation technology is very important in many fields such as artificial intelligence, military, and video analyzing. It could also be applied in practical applications.

*Corresponding author: vishnu.koyya@gmail.com

This paper studies the navigation grid method and A* path-finding algorithm for VR application scenarios. For complex VR scenarios, this paper proposes an improved solution to the path-finding algorithm, which improves the efficiency and accuracy of the path-finding technology. The improved algorithm is applied to the virtual reality environment to realize the path planning function of the virtual character.

Real-time Virtual Simulation Platform for Multi-UVA hunting target using Deep Reinforcement Learning (2021) Kangrui Zhu, Qun Zong, Ruilong Zhang. As one of the most useful flying objects, quadrotor UAVs are gradually being applied to the military field in the form of "swarms". However, an urgent problem is how to improve the intelligence of UAVs at a low cost. In this paper, we propose a hunting scenario for multi-UAV in an urban environment. Firstly, we build the game model of a multi-UAV hunting target and design the reward function. Secondly, we introduce an improved multi-agent deep deterministic policy gradient (MADDPG) against a game AI target. Then, we build a virtual platform to simulate the urban combat environment based on the Unity3D game engine. The ML-Agents Toolkit is used to develop a real-time simulation data interface to achieve low-cost model training. Finally, simulation results from Python and the virtual environment can demonstrate the effectiveness of the proposed hunting scenario.

ABMU: An Agent-Based Modelling Framework for Unity3D (2020) Kostas Cheliotis. The field of Agent-Based Modelling (ABM) has expanded significantly since its emergence, with a plethora of ABM development frameworks available to researchers today. However, relatively few frameworks are found to support 3D models and furthermore, the majority of them are often limited to 3D visualizations of the underlying 2D models. At the same time, many systems of interest are identified that can significantly benefit from being simulated in three dimensions. In response to this, a potential candidate platform for the development of 3D ABMs is identified in Game Engines (GEs), as they often support 3D graphics and provide a programming back-end for coding model logic, however, no significant frameworks exist for ABM development in GEs. This paper presents the Agent-Based Modelling Framework for Unity3D (ABMU), an ABM framework developed for use within the widely used GE Unity3D.

Comparison Between A and Obstacle Tracing Pathfinding in Gridless Isometric Game Lailatul Husniah; Rizky Mahendra; Ali Sofyan Kholimi; Eko Cahyono. Pathfinding algorithms have commonly used in video games. City 2.5 is an isometric grid-less game which already implements pathfinding algorithms. However, current pathfinding algorithm unable to produce optimal route when it comes to custom shape or concave collider. This research uses A* and a method to choose the start and end node to produce an optimal route. The virtual grid node is generated to make A* works on the grid-less environment. The test results show that A* be able to produce the shortest route in concave or custom obstacles scenarios, but not on the obstacle-less scenarios and tight gap obstacles scenarios.

3. Components

A. *Unity's ML Agent Tool Kit*

The Unity Machine Learning Agents Toolkit (ML-Agents) is an open-source project that enables games and simulations to serve as environments for training intelligent AI agents. Implementing ML agents in Unity 3D, including the set up the environment, collecting data, training the agent, and testing its performance will be explained.

B. *Project's Game Environment*

The game environment is a 3D maze structure with a plane game object as the ground and multiple cube game objects as the walls of the maze. The maze is designed in such a way that the players can navigate in multiple ways to any point from any point. All the wall game objects are then given a parent game object to have similar behavior, varying in size and arranged in a way that presents a challenging game environment. Attach the necessary components Box Collider for the wall game object, and Plane Collider for the plane game object. Checking the trigger check box allows the wall game object to be detected when there is a collision between game objects. Adding a tag to the game object will help identify the particular game object which is a standardized method to verify the collision. The size of the game environment matters when taking the agent's training time and process. The larger the game environment, the longer training iterations it takes. The maze environment has a scale of (1, 1, 1) and contains nine walls/obstacles. Each wall in the maze has a length ranging from 1 to 3 unity world units, while its width ranges between 0.5 to 1 unity world unit.

C. *Player*

In this paper, the player is referred to as a game object which is per-programmed. The player game object is to be controlled by the human player in real-life game scenarios. For evaluation purposes, the player's movement is made automatic. Rigidbody and Collider components are attached to the player game object such that physics shows an effect on the player game object. The automated player is programmed to move to a destination point before the AI agent collides with it.

D. *Agent*

In this paper, the agent is referred to as a game object which is developed and trained using AI. They are trained to perform specific tasks and improve their performance over time, making them adaptive to dynamic and complex game environments. With agents, game developers can create NPCs, enemies, and other game elements that can learn and evolve based on player behavior and other environmental factors. The agent is a capsule game object with a collider component attached, for collision sensing. This agent is to perform a specific task, its behavior is to be defined. The agent used is a machine-learning agent powered by a path-finding technique with a calibrated rewards system. Machine learning agents using Unity's ML-Agents require a Behavior Parameters component to be attached to it in order to define their task. Once the agent is defined, observations can be collected to train the agent. This involves running the game and recording the agent's actions and observations. In this paper, the agent is trained to find the shortest path against the automated player in an entirely unknown dynamic game environment. The trained artificial neural network brain model is then given the agent to perform the learned task in real time.

E. Gameplay

The trained agent is expected to learn to navigate through the complex maze environment by avoiding the obstacle which is the "wall" game objects and reaching the target using the computed vector3 unity world points. The agent should be able to identify the distance between itself and the target and take appropriate actions accordingly. The agent should be able to collide with the target when it is close enough and wait at the given vector 3 points if the target is not close enough. The agent's reward system should be designed in such a way that it encourages behaviors that lead to the successful completion of the task. The training of the agent may take some time, depending on the complexity of the task and the size of the dataset used for training. The agent should gradually improve its performance as it gains more experience through training. Once the agent is trained, it should be able to navigate through the maze environment efficiently and reach the target while avoiding the walls.

F. Unity 3D Game Engine

Unity 3D is a game engine used to create 2D and 3D video games. It is a cross-platform game engine where one can create video games for mobile devices, consoles, and desktop computers. It provides a wide range of features and tools for game developers, including a powerful game engine, scripting tools, graphic APIs, a physics engine, an asset management system, a state management system, and a user interface system. Developers create complex game environments using C#, C++, and JavaScript. It also offers support for virtual reality (VR) and augmented reality (AR) development.

G. Materials

In Unity, a Material is an asset that determines how a 3D object is rendered. It defines the texture, colour, and shading of the object, and can be used to create a wide variety of visual effects. In this project, we are taking colour to visually classify the training of the AI agent. Each material created is attached to each and every game object which is in the game environment to distinguish visually.

H. Prefabs

In Unity, a prefab is a per-fabricated game object that one can reuse multiple times in the game. It's a template of a game object that includes all the components, properties, and settings that have been made to it. Every game object which is intended to be in the Unity game environment is made as a prefab for better reuse.

I. Python

Use of Python 3.9.13 is appropriate regardless of the latest Python packages. The latest Python packages are not compatible with dependencies that machine learning might use. The use of Python 10.x or Python 11.x would not show the expected results.

J. PyTorch

PyTorch is an open-source machine learning library used for developing and training machine learning models. It provides tools to build various types of neural networks, including convolution neural networks (CNNs) and recurrent neural networks (RNNs). A very specific version of PyTorch is used which is compatible with Python 3.9.13 version. PyTorch wouldn't work with the latest Python versions such as Python 10.x or Python 11.x.

K. TensorBoard

TensorBoard is a web-based visualization tool used in machine learning frameworks such as TensorFlow, PyTorch, and Keras. It provides real-time monitoring and debugging of machine learning models during training and evaluation, helping researchers and developers to better understand and optimize their models.

4. Implementation

A. Base Methodology

The most common artificial intelligence is way-point navigation by carefully placing points (nodes) in the game environment to move the game-controlled characters between each point. The major drawback of this method is that these way-points need to be manually set up, and it is time-consuming work. Meanwhile, these way-points will depend upon the environment; different environments require different configuration way-points. In addition, the number of way-points and the location of way-points are also different due to the platform. The A-Star algorithm technique uses path scoring to determine the best path from the starting node to the destination node. To actually score each node, A-Star basically adds together two components $g(n)$ and $h(n)$ giving us $f(n)$. First, it looks at the cost to move from the initial node to any next consecutive node $g(n)$. Next, it looks at the cost to move from the consecutive node to the final node $h(n)$. The equation shows the equation used for scoring any given node.

$$f(n) = g(h) + h(n)$$

B. Hybridization

The A-Star Path finding technique acts as a decision-making rule for designing the reinforcement learning reward system for the agent. The Agent is rewarded with the total cost $f(n)$ which is the sum of $g(n)$ and $h(n)$ where $g(n)$ is the cost of the path from the starting node to node n, and $h(n)$ is the estimated cost from node n to the goal node. The function $f(n)$ estimates the total cost of the cheapest solution through node n. The agent is penalized if the agent is trying for an expensive path instead of the cheapest path. Based on the rewards the agent gains, the agent tries to lessen the value of the total cost $f(n)$ hence finding the best path to the destination up on training. The agent computes 60 times per second (for every frame). The agent is provided with other heuristics and rewards apart from the A-

Star technique to deal with the complex environment.

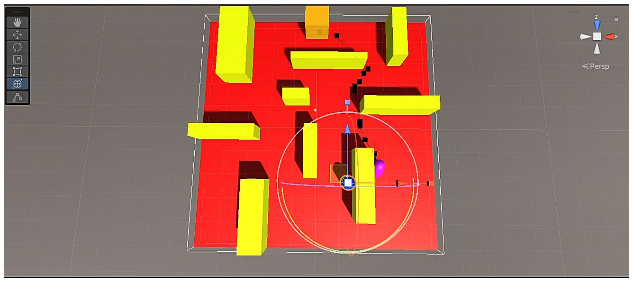
Fig. 1. Environment turned “red” when the agent collided with the wall (penalized)

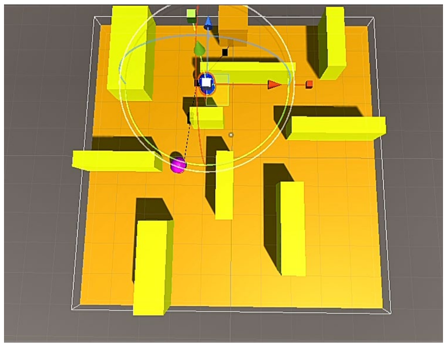
Fig. 2. Environment turned “orange”. Agent getting rewarded consistently and probably reached the predicted node

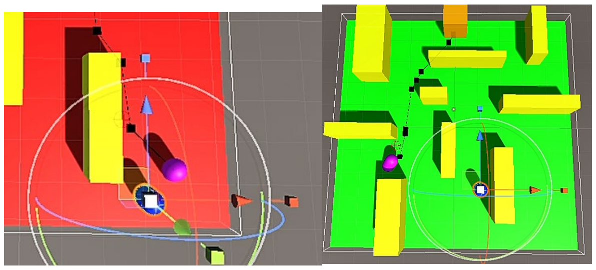
Fig. 3. Environment’s transition from “red” to “green”. Agent successfully collided with the moving target game object finding the path to the target

5. Graphical Results

After the agent is trained, its performance is tested in the game environment. This involves running the game and evaluating the agent's behavior and decision-making skills. The agent's performance can be improved by refining its behavior and fine-tuning its parameters. The learning results are evaluated in TensorFlow.

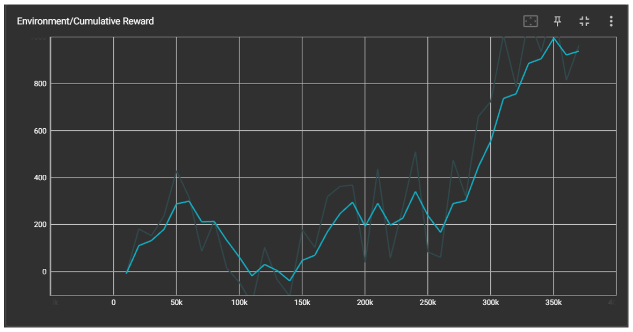

Fig 4. The cumulative reward graph tends to increase as the agent gets trained

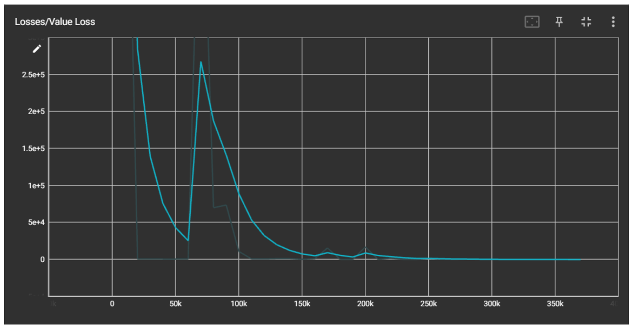

Fig 5. The value loss graph is expected to decrease as the agent gets better with its environment

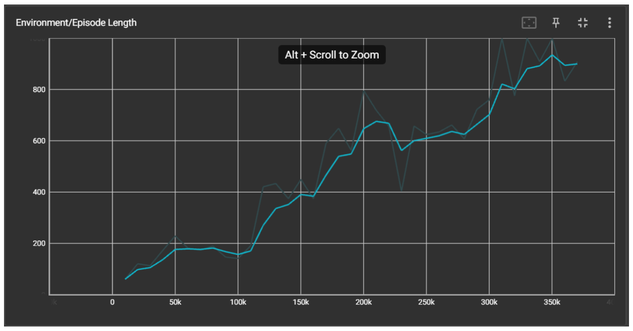

Fig. 6. The episode length graph is expected to increase as the agent gets skilled with its performance

6. Conclusion

This work has demonstrated the hybridization of the path finding technique which is supervised learning with reinforcement learning that can be used as a powerful AI agent in video games. This study also concludes that this algorithm is still used widely in many shortcoming studies related to path finding, especially in game development. The most common artificial intelligence in a game is way point navigation by carefully placing checkpoints with cost in the game environment to move the game-controlled characters between each point. Finally, this project proposes a more general dynamic AI model which can solve the random obstacle avoidance problem during finding the shortest path. Moreover, the training and improvisation of this AI agent in a complex area are challenging. In searching for the shortest path to reach the target virtual human. This trained model can be used as a dynamic NPC or Enemy AI which will then be able to successfully navigate the complex environment making real-time decisions, acting based on the rewards received and adapting when needed.

References

[1] Hart, P. E.; Nilsson, N. J.; Raphael, B. “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”, SIGART Newsletter 37: 28–29.
[2] Alexander Nareyek. AI in Computer Games [OL]. http://www.ai-center.com/publications/nareyek-acmque ue04.pdf
[3] RemcoStraatman, William van der Sterren, ArjenBeij, Killzone’s AI: dynamic procedural combat tactics [OL], http://www.cgf-ai.com/docs/straatman_remco_killzone_ai.pdf
[4] Andrew Lupponw. Hierarchal AI [OL]. http://www-cs-students.stanford.edu/~amitp/Articles/HierarchalAI.html
[5] Sun Shudong, Lin Mao, “The coordination path planning of multiple moving robots based on GA”, Automation Journal, 2000, 26(5):672-676.

[6] Mahmoud Tarokh, “Hybrid Intelligent Path Planning for Articulated Rovers in Rough Terrain”, Fuzzy Sets and Systems, 2008,159(21):2927-2937.

[7] Ye Tao, Chen Haikui, Yang Guosheng, “A new method of Global robot navigation and obstacle avoidance in Unknown Environment”, Robo, 2003,25(6):516-520.

[8] GAO Qingji, YU Yongsheng, HU Dandan, “Feasible path search and optimization Based on an improved A * algorithm”, China Civil Aviation College Journal, 2005,23 (4): 42-44.

[9] L. delaOssa, J. A. Gamez, and V. Lopez, “Improvement of a car racing controller by means of Ant Colony Optimization algorithms,” in IEEE International Journal of Machine Learning and Computing, vol. 2, no. 1, February 2012 17 Symposium on Computational Intelligence and Games, 2008, pp. 365-371.

[10] S. Fujii, T. Nakashima, and H. Ishibuchi, “A study on constructing fuzzy systems for high-level decision making in a car racing game,” in IEEE Congress on Evolutionary Computation, 2008, pp. 3626-3633.

[11] T. Nakashima, and S. Fujii, “Designing high-level decision-making systems based on fuzzy if-then rules for a point-to-point car racing game,” Soft Computing, vol. 14, no. 5, pp. 529-536, March 2010.

[12] J. Togelius, P. Burrow, and S. M. Lucas, “Multi-population competitive co-evolution of car racing controllers,” in IEEE Congress on Evolutionary Computation, 2007, pp. 4043-4050.

[13] J. Togelius, and S. M. Lucas, “Evolving controllers for simulated car racing,” in Proceedings of the Congress on Evolutionary Computation, 2005, pp. 1906-1913.

[14] J. Togelius, and S. M. Lucas, “Evolving robust and specialized car racing skills,” in Proceedings of the IEEE Congress on Evolutionary Computation, 2006, pp. 1187-1194.

[15] A. Agapitos, J. Togelius, and S. M. Lucas, “Evolving controllers for simulated car racing using object oriented genetic programming,” in GECCO ’07: Proceedings of the 9th annual conference on Genetic and evolutionary computation. New York, NY, USA: ACM, 2007, pp. 1543–1550.

[16] Y. C. Hui, E. C. Prakash, and N. S. Chaudhari, "Game AI: artificial intelligence for 3D path finding," in TENCON 2004. 2004 IEEE Region 10 Conference, 2004, vol. 2, pp. 306-309.

[17] J. -Y Wang, and Y. -B Lin, "An Effective Method of Pathfinding in a Car Racing Game,” in the 2nd International Conference on Computer and Automation Engineering, 2010, pp. 26-28.

[18] D. M. Bourg, and G. Seemann, AI for Game Developers, O’REILLY, 2004, ch. 7.

[19] S. Rabin, AI Game Programming Wisdom 4, Charles River Media, 2008, ch. 2.

[20] N. J. Nilsson, Principles of Artificial Intelligence, Tioga Publishing Company, Wellsboro, PA, 1980, pp. 366-381.

[21] B. Hamboeck, “XNA or game development for everyone – restructuring the game part2,” .Net Developer’s Journal, vol. 6, pp. 14- 29, 2008.