

# An Overview of Different Artificial Intelligence Applications in Games

Xuantao Chen<sup>a</sup>

*Quality School of International, Bitao Building, 8-5 Taizi Road, Shekou, Shenzhen, Guang Dong, China*

**Keywords:** Artificial Intelligence, Games, Ad Hoc Behaviour Authoring, Tree Search, Machine Learning.

**Abstract:** With huge improvements being made in Artificial Intelligence (AI) in recent years, the power of AI is gradually revealed to a greater extent. Game development is complicated and challenging due to the requirement of knowledge in different fields and the gigantic amount of effort to complete it. Therefore, its complexity undoubtedly creates an application scenario for the powerful AI. In this paper, an academic perspective of game AI was presented by reviewing several fundamental AI methods used in the majority of games such as ad hoc authoring, tree search, and machine learning. Besides this, the paper also mentioned several specific AI algorithms or methods such as procedural generation techniques and constraint satisfaction problems, methods that were applied to some common game genres. The research value of the paper arises with the collision of the two hot complex topics, and its trend will become more popular in the future without doubt. A conclusion on the topic is reached with some open problems mentioned, which demand new methods and improvements.

## 1 INTRODUCTION


Games and Artificial Intelligence (AI) have a long history. Since the late 20<sup>th</sup> century, people have been trying to apply AI to games, and they have done well. Some landmark achievements of AI applications in games happened in 1994, 1997, and 2016, with AI programming Chinook, Deep Blue, and AlphaGo defeating the world champions in different games respectively (Fan, Wu, Tian, 2020). Chinook, developed in the 1990s, is one of the most representative precursors in such fields as it is the first time in history that a computer program won a world championship.

On 15 August 1994, a checkers match was played by the world checkers champion Marion Tinsley and Chinook, in Boston, Massachusetts. The first six games were drawn, with only two potential winning chances shown. Right before the next round of games, Tinsley claimed that he had a health issue, so the match was canceled, and Tinsley was sent to the hospital. Subsequently, Tinsley was diagnosed with cancer, and he resigned his title of world champion (Schaeffer, Lake, Lu, Bryant, 1996). This marked a historical moment as it was the first time that a non-human character defeated a world champion human

in the game domain even though it is not considered as a glorious victory for Chinook. Nevertheless, a few challengers came after Tinsley's capitulation to challenge this champion machine, but they failed to win. With Chinook's strength being proved, the human concept of AI development in the aspect of the game was solidified, which laid the foundation for the future AI's domination over the field.

Chinook's use of tree search was one of the key factors that helped him win the match. To be more specific, the algorithm is called the alpha-beta search algorithm that is in a depth-first manner (Schaeffer, Treloar, Lu, Lake, 1993). With studies suggesting that a deeper search will translate into stronger play, it indicates that the algorithm helps Chinook to determine the optimal move in the game, thus defeating Marion Tinsley and other challengers (Thompson, 1982).

Speaking of tree search, it is known that AlphaGo, another famous AI that created a landmark in AI games, also involved applications of tree search as one of its core components. Similar to alpha-beta search algorithm, the Monte Carlo Tree Search (MCTS) of AlphaGo is another tree search algorithm that helps AI agents determine which step to execute in a game through simulation process. More analysis

<sup>a</sup> <https://orcid.org/0009-0004-5537-7355>

and explanation of MCTS will be mentioned later in this paper.

With the rapid development of electronics and programming in recent years, there are now many more advanced AI techniques beyond just traditional tree searches. Developers have come up with new AI algorithms and invented new advanced AI methods with improved ability (Bakkes, Spronck, Herik, 2009). However, in order for researchers to understand the basic principles laid behind, this article will mainly focus on the most fundamental algorithms of game AI that laid the basis for the future development of game AI. Some examples such as ad hoc behavior authoring, tree search, and machine learning are shown in Figure 1, which are the methods that the newly invented game AI are based on. Furthermore, this paper also summarized two designated AI algorithms or methods specifically used in some common game genres: procedural generation techniques and constraint satisfaction problems. With such a review, researchers, scholars, or developers will have a clear understanding of the logic behind game AI algorithms and how different AI helps in the game industries.

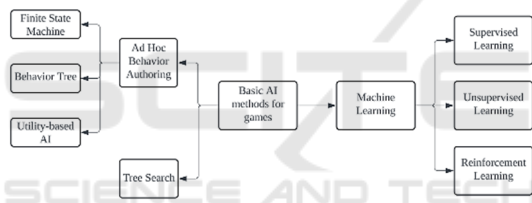


Figure 1: Representative AI methods in games.

## 2 FUNDAMENTAL AI ALGORITHMS IN GAMES

In general, building and controlling a non-player character (NPC) is always considered one of the primary goals when developing video games. With AI, such an essential task can be easily accomplished.

### 2.1 Ad Hoc Behaviour Authoring

Ad Hoc behaviour authoring methods are the most popular AI methods used to control the non-player characters in games. Some of the most represented algorithms include finite state machines (FSM), behaviour trees (BT), and utility-based AI.

FSM, as its name indicates, is a model of computation with a finite number of sequential states or conditions. The main purpose of this model is to change from one state to another based on the inputs

and the current state, which is very simple and therefore, such model had been applied to most of the video games in the world (Lee, 2014). However, there are some defects in the FSM model, namely lack of reactive and hard to cope with complicated scenario, which led to the creation of BT.

Just like FSM, BT also has transformation between a limited number of behaviours and relies on the current state and situation to decide what action or behaviour should be taken. It breaks down desired behaviour into hierarchy tree structures, with nodes in each branch representing a specific decision point and its corresponding action. It is its tree structure that allows developers to break down complex behaviours into smaller, more manageable modules, making it easier to design and maintain the whole system (Colledanchise, Parasuraman, Ogren, 2019).

In order to further alleviate the modular limitation of FSM and BT, utility-based AI was invented. Unlike FSMs and BTs, the utility-based AI methods check all available options rather than one at a time, make an analysis between them, and determine the most appropriate output. To do this, the method made an objective comparison between actions that the NPC can take and the game demands by assigning normalized values to all possible actions that the NPC can take at that moment. As for output, the action with the highest value is usually chosen as the most suitable decision. Therefore, utility-based AI has an apparent advantage over complex scenarios, indicating its flexibility and extensibility.

### 2.2 Tree Search

The vast majority of tasks people ask AI to do can be transferred into a problem, which can be solved by AI through searching for the best-fit solution. Tree Search algorithms can be considered one of the simplest while efficacious algorithms that can be used to complete such a task.

Tree Search is a tree data structure and has a root node known as the starting point. Then, it proceeds as it explores each and every single node where each node represents a state. When a node is reached, the tree will have to decide which branches to execute based on specific criteria since there are a variety of paths with different operations and transitions to the next corresponding step.

One famous representative of the Tree Search algorithm is MCTS mentioned in the introduction, which is highly applied in various games and has achieved gigantic success. For instance, it is well-known that MCTS played a crucial role in the Google DeepMind AlphaGo that defeated the world Go game

champion in 2016 (Silver, Huang, Maddison et al, 2016). The key component of MCTS can be divided into 4 steps:

1. Selection: starting from the root node that represents the current game state, MCTS repeatedly selects child nodes based on a selection policy, such as the Upper Confidence Bound (UCB) formula, until it reaches a leaf node, a state that is unexpanded.

2. Expansion: Following the previous step, MCTS adds new nodes, known as the child nodes, to the leaf nodes.

3. Simulation: MCTS then plays a random simulation with random decisions until a terminal state is reached. Once the terminal state is reached, it sends a result of how well it performs with score being calculated.

4. Backpropagation: The result of the simulation is then propagated back to the starting point, updating the statistics of the visited nodes and corresponding data.

The algorithm repeats these four steps numerous times, gradually building up the search tree and guaranteeing the best solution.

MCTS has not only applied to Go or other classic board games, but it has also applied to modern board games and video games that are way more complex. For example, researchers' implementation of MCTS in Settler of Catan had outperformed previous heuristic game AI, providing a challenging opponent for players (Chaslot, Bakkes, Szita, Spronck, 2021). Besides this, some other scholars have deployed MCTS on RTS games such as StarCraft to create AI agents with multifaceted strategic decisions that are more effective, achieving performance comparable to that of human players (Uriarte, Ontañón, 2021). Overall, it is easy to see the shadows of Tree Search in various games, stating its importance and relevance towards game industries.

## 2.3 Machine Learning

Machine learning, as its name implies, is a field of artificial intelligence that enables computers and systems to learn and improve, where the system will adapt to the new circumstances gradually as it explores. There are three major categories of machine learning: supervised learning (SL), unsupervised learning (UL) and reinforcement learning (RL).

In (SL, humans feed data to the system, where the inputs and their corresponding outputs are labelled and provided. The system is likely to learn the relationship between input and output, creating a set of rules that will predict the expected output. Unlike SL, the dataset fed to UL is unlabelled and only

contains input. The system will make observations and find the association of inputs according to their features. Totally different from SL and UL of analysis, RL works well in the field of decision-making, especially sequential one. The system or agent is able to interact with an environment and determine what is deemed as good or bad behaviour in the context of the game objective. It receives rewards or penalties as it executes and adjusts its behaviour accordingly. The RL model focuses more on delayed rewards, which maximizes the cumulative long-term rewards, rather than simply the immediate rewards. Therefore, researchers have applied RL models to games that emphasize such features, such as StarCraft (Zhao, Shao, Zhu et al, 2016).

As seen, machine learning algorithms rely heavily on data to train models and make predictions or decisions. And so, for the data of games, video data such as replay of players would provide gigantic value to the training model. Many organizations such as DeepMind and Facebook have released a great amount of replay data sets to train their model and try to mimic the player's behaviour. As long as there is sufficient data, whether it is SL, UL, or RL, it can strongly enhance the game experience of players by creating challenging AI agents.

## 3 SPECIFIC AI ALGORITHMS FOUND IN SOME COMMON GAME GENRES

In addition to the simple task of applying AI to the work of NPCS, AI can also help humans in many more areas of the game. Indeed, some specific AI has been applied to a wide range of video game genres to accomplish different tasks such as procedural generation techniques for content generation in sandbox games and constraint satisfaction problems for a further realistic experience in simulation games respectively.

### 3.1 Procedural Generation Techniques

In Sandbox game, creating a random game world is an indispensable step. Procedural generation techniques are powerful tools in computer programming that can accomplish this task by its function of automating the creation of certain data according to algorithms. One of the major AI algorithms of procedural generation techniques that work in Sandbox games is noise function, such as Perlin Noise and Simplex Noise.

Created by Ken Perlin in 1983, Perlin noise is a procedural noise algorithm that generates a smooth, continuous, and random pattern, which makes the texture look more realistic even though it is generated by AI. It accomplishes such a task as it calculates the noise patterns by performing a series of actions such as interpolating between a grid of random gradient vectors and combining multiple octaves of noise.

On the other hand, also introduced by Perlin in 2002, Simplex noise served as an alternative to the early Perlin noise, with a higher efficiency and smoother noise pattern in complex terrain. Simplex noise has a faster gradient evaluation that allows it to extend to a higher dimension (Gustavson S, McEwan I, 2022).

### 3.2 Constraint Satisfaction Problems

Normally, in simulation games, problems will always arise as time progresses. For NPC to act like humans is also an important factor in determining whether a simulation game is good or bad. Therefore, this led to the introduction of Constraint Satisfaction Problems, a technique involving a series of problems that match the above key points, into simulation games.

Constraint Satisfaction Problems is a powerful problem-solving technique that finds solutions to problems that involve satisfying a set of constraints or requirements. The fundamental principle behind Constraint Satisfaction Problems is to represent the problem as a set of variables, each with a domain that represents the possible value stored, and a set of constraints that limit the values (Barták, Salido, Rossi, 2010). The goal is also set but must be accomplished under constraint.

Constraint Satisfaction Problems fit well with simulation games since problems emerge at any time in this kind of game. In order to act like humans (that's what simulation games are looking for), the constraints in Constraint Satisfaction Problems will limit the behaviour of the NPC and govern the game world. These constraints can capture the complex and often conflicting issues that humans face in the real world. This flexibility allows the simulation to produce further nuanced and realistic scenarios, thus enhancing the player's experience.

## 4 CONCLUSIONS

With the summary of previous section had implies, games are an ideal domain of AI. In this paper, different basic AI methods and algorithms as well as AI applications in games are summarized and

presented. However, there are still many open questions about the application of AI in such fields. Some of the potential challenges are listed below:

- Whether the ai can become a qualified teammate and cooperate with different types of players, so that the player's experience would rise.
- Whether the AI's output in some algorithms may be too predictable, which will reduce challenge and excitement for players.
- Whether humans can create a universal game AI with a unified framework, resulting in a lower cost of game development.

Apparently, different AI methods are most likely relevant to each other, with each based on the other's ability to extend new AI methods. Therefore, it is suggested that enrolling the combination of various AI methods that already exist in order to learn from each other's strong points is an essential and ineluctable thing to do for development of new AI methods or algorithms.

## REFERENCES

- Bakkes, S., Spronck, P., Herik, J., 2011. Rapid and Reliable Adaptation of Video Game AI. *IEEE Transactions on Computational Intelligence and AI in Games*, 34(1): 30-31.
- Barták, R., Salido, M.A. & Rossi, F., 2010. Constraint satisfaction techniques in planning and scheduling. *J Intell Manuf* 21, 5–15.
- Bayliss, J. D., 2012. Teaching game AI through Minecraft mods. 2012 IEEE International Games Innovation Conference, Rochester, NY, USA, pp. 1-4.
- Chaslot, G., Bakkes, S., Szita, I., & Spronck, P., 2021. Monte-Carlo Tree Search: A New Framework for Game AI. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 4(1), 216-217.
- Colledanchise, M., Parasuraman, R., Ogren, P., 2019. Learning of Behavior Trees for Autonomous Agents. 2018 IEEE, *IEEE TRANSACTIONS ON GAMES*, 11(2).1
- Fan, X., Wu, J., Tian, L., 2020. A review of artificial intelligence for games//Artificial Intelligence in China: Proceedings of the International Conference on Artificial Intelligence in China. Singapore: Springer Singapore, 298-303.
- Gustavson, S., McEwan, I., 2022. Tiling Simplex Noise and Flow Noise in Two and Three Dimensions. *Journal of Computer Graphics Techniques* 11(1).
- Lee, M., 2014. An Artificial Intelligence Evaluation on FSM-Based Game NPC. *Journal of Korea Game Society*, vol. 14, no. 5. Korea Academic Society of Games, pp. 127–135.

- Schaeffer, J., Lake, R., Lu, P., & Bryant, M., 1996. CHINOOK The World Man-Machine Checkers Champion. *AI Magazine*, 17(1), 21.
- Schaeffer, J., Treloar, N., Lu, P., & Lake, R., 1993. Man Versus Machine for the World Checkers Championship. *AI Magazine*, 14(2), 28.
- Silver, D., Huang, A., Maddison, C. J., et al., 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529(7587): 484–489.
- Thompson, K., 1982. Computer Chess Strength. In *Advances in Computer Chess 3*, ed. M. R. B. Clarke, 55–56.
- Uriarte, A., & Ontañón, S., 2021. Game-Tree Search over High-Level Game States in RTS Games. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 10(1), 73-79.
- Zhao, D., Shao, K., Zhu, Y., et al., 2016. Review of deep reinforcement learning and discussions on the development of computer Go. *IET Control Theory A* 33(6):701–717.

