ISSN No: 2350-1278 (Print)
Peer Reviewed & Refereed Journal
National Research Journal

AI TECHNIQUES FOR GAME DEVELOPMENT

Kavita Kumari

Assistant Professor, Department of Computer Science, Khalsa College for Women, Civil Lines, Ludhiana

Gagandeep Kaur

Assistant Professor, Department of Computer Science, Khalsa College for Women, Civil Lines, Ludhiana

ABSTRACT

As the graphics race subsides and gamers grow weary of predictable and deterministic game characters, game developers must put aside their “old faithful” finite state machines and look to more advanced techniques that give the users the gaming experience they crave. More intelligent, adaptive, and realistically behaving characters will be the next big thing in the industry, rather than more polygons, better textures, or faster frames per second. This paper covers the numerous artificial intelligence approaches that are now in use by game creators in addition to those that are new to the field. The techniques covered in this paper are finite state machines, scripting, agents, flocking, fuzzy logic, decision trees, neural networks, genetic algorithms, and extensible AI. This paper introduces each of these methods that can be applied to games.

Keywords Artificial Intelligence, Computer Games, Fuzzy Logic

1. INTRODUCTION

Modern computer games use a wide variety of artificial intelligence (AI) methods. The most often used methods include flocking, agents, scripting, and finite state machines. These techniques are well-established, simple and have been successfully employed by game developers for several years. Additionally, the use of fuzzy logic in AI for games. Finally, there are a few game developers who are venturing out to try exciting and new techniques, which have not been possible until recently, due to processor constraints. These techniques include decision trees, neural networks, and genetic algorithms. This review will explore each of these techniques and their applications in industry and computer games.

2. LITERATURE REVIEW

This paper aims to introduce all of these approaches that can be used in AI games, with a special emphasis on the various approaches. (Duarte et al., 2020)

Game AI, the intersection of artificial intelligence and gaming, has experienced significant growth and development over the past few decades. As gaming technology has evolved, the sophistication of AI systems is implemented within games. In this, we review the historical context of game AI, its significance as a benchmark for AI progress, and recent advancements in key application areas [1].

In modern computer games, artificial intelligence (AI) techniques play a vital role in creating immersive and engaging player experiences. A variety of AI techniques are employed by game developers, ranging from traditional methods like finite state machines and scripting to more advanced approaches such as neural networks and genetic algorithms. This review aims to explore each of these techniques, discussing their applications in industry and computer games, as well as their advantages, limitations, and current usage in games [2].

The concept of flocking behavior, originating from Craig Reynolds' seminal paper "Flocks, Herds, and Schools: A Distributed Behavioral Model," has been widely adopted in the development of video games to create realistic and cohesive group movement among non-player characters (NPCs). This literature review explores the application of flocking algorithms in various gaming scenarios, ranging from simulating natural phenomena like bird flocks and fish schools to orchestrating tactical movements of military units in strategy games [3].

Yannakakis (2012) this essay investigates Redefining "game AI" is necessary, as the term has become outdated over a decade after the initial research on AI's application in computer games and the creation of a new AI domain. Non-player characters (NPCs) at various levels of control—from pathfinding and navigation to decision-making—have traditionally been the focus of game AI tasks. It appears, however, that the traditional challenges of game AI have been well addressed through the use of sophisticated AI approaches, not necessarily following or inspired by advances in academic practices. This is evident from commercial-standard games developed over the last 15 years and current game productions.

3. FINITE STATE MACHINES

A finite state machine has been used in old games like PACMAN as well as in newer games such as TOM RAIDER. The reason for this is that they are easy to program, easy to understand and debug, and general enough to solve any problem. An FSM divides a game object's behavior into logical states in such a way that each type of behavior is represented by a specific state.

First, we need to understand what FSM is and how it is created before we can use it as AI in games. An FSM is a state machine that implements finite states. FSM is usually designed using a graph where node(vertex) is represented as states and edges are represented as transition lines. For our NPC, a state can represent an action or a state. Every state has a connection to at least one other state, allowing some state to reach it. Now, transition is the process by which we change from one state to another and requires a connection. If one or more of the predetermined conditions are satisfied, the state changes. Lastly, we must ensure that we begin at any state when constructing the FSM and that no state is unreachable. The game loop keeps FSM updating by checking if the transition condition is met or not. environment will be for the player. For game AI, the possible ways in which to use an FSM are endless. It could be used to manage the game world or maintain the status of the game or game object. An example is modeling unit behavior in a real-time strategy game. Alternatively, it could be used to parse input from the human player or even to simulate the emotion of a non-player character.

4. AGENTS

Intelligent agents are software agents that perceive their environment and act in that environment in pursuit of their goals. Examples of intelligent agents include autonomous robots in a physical environment, software agents with the internet as their environment, or synthetic characters in computer games and entertainment. Agents usually integrate a range of competencies, such as goals, reactive behavior, emotional states and consequent behaviors, natural language, memory, and inference. Agents are central to the study of many problems in AI, such as modeling human mental capabilities and performing complex tasks. Games are ideal environments for agents as they provide realistic environments in which only limited information is available and where decisions must be made under time and pressure constraints. Generally, agents in games are sets of FSMs that work on their particular problems and send messages to each other. Alternatively, an agent could be a set of fuzzy state machines, neural networks, genetic algorithms, or any combination of some or all of these techniques. Important decisions that need to be made when designing an agent are the architecture and whether the agent is to be reactive, goal-directed, or some combination of the two. A purely reactive agent is suited to highly dynamic environments where little information about previous actions and states is necessary. At the other extreme, a purely goal-directed agent is suited to a static environment where planning and considering previous moves are highly desirable. For example, a monster in a first-person shooter or a role-playing game would be more suited to simply reacting to what is currently happening in the game. However, an agent governing the strategy for the AI in a strategy game needs to carefully plan its moves depending on what has happened so far in the game. A good architecture for a real-time strategy agent is necessary to ensure success.

5. SCRIPTING

A scripting language is any programming language created to simplify any complex task for a particular program. The scope of a scripting language can vary significantly depending on the problems it is designed to solve, ranging from a simple configuration script to a complete runtime-interpreted language. Scripting languages for games, such as Quake's QuakeC or Unreal's UnrealScript, allow game code to be programmed in a high-level, English-like language. They are designed to simplify some set of tasks for a program and hide many complicated aspects of a game, thus allowing non-programmers, such as designers and artists, to write scripts for the game. During development, the designers use scripting to implement stories, while artists use scripting to automate repetitious tasks, do things that the computer can do better than humans, and add new functionality. After the game is shipped, mod groups and hobbyists write scripts if the scripting system has been exposed to the public. However, as with FSMs, scripting languages are deterministic and they require the game developer to hard-code character behavior and game scenarios. Therefore, the developer must anticipate and hard-code each of the situations the player might be in. The uses of scripting languages in games vary from simple configuration files to entirely script-driven game engines. Creating events and opponent AI for the game's single-player mode are among the frequently used applications. Also, in single-player mode, they can be used to tell the story of the game and control the player's enemies. A first-person shooter game could use scripting to create a monster's AI. Alternatively, a real-time strategy game might use scripting to define how spells function or to define a quest or part of the game story. Furthermore, in massively multiplayer online games, scripting can be an extremely useful tool (MMOGs). Scripting can be used to conceal the specifics of managing several servers in a server farm in massively multiplayer online games (MMOGs), making it easier to send network events to a client. Also, scripting languages can handle saving an object's state automatically. In role-playing games, scripting can be used to define simple conversation trees for a non-player character. Finally, a scripting language could even be a complicated object-oriented language that controls every aspect of gameplay. Games that have successfully used scripting, whether it was a custom-made scripting language or an off-the-shelf language, include Black & White, Unreal, Dark Reign, and most of the games developed by BioWare.

6. FUZZY

Logic Fuzzy Logic is unlike traditional Boolean logic in that it allows intermediate values to be defined between conventional values such as yes/no or true/false. Consequently, "fuzzy" values such as 'rather hot' or 'very fast' that are used to describe continuous, overlapping states can be used in an exact mathematical way. Fuzzy Logic Systems have an easy-to-understand structure. Fuzzy logic has the advantage of allowing decisions to be based on inaccurate or incomplete data that is not amenable to Boolean logic. Fuzzy logic is powerful because it can express a concept with a limited set of fuzzy values, unlike Boolean logic, which requires hard coding of each state and transition. It is possible to apply fuzzy logic to the domains of behavioral Fuzzy logic can be used for highly nonlinear problems, when processing expert knowledge is necessary, and when there is no simple mathematical model that can solve the problem. However, fuzzy logic is not ideal when conventional methods yield a satisfying result, when there is an existing mathematical model that already solves the problem, or when the problem is not solvable. In short, if there is already a simple solution that satisfies a problem then there is no need to complicate things. However, if the problem is non-linear or there is no simple solution, then fuzzy logic may be appropriate. selections and input/output filtering and has been used in tools for controlling subway systems, industrial processes, household and entertainment electronics, and diagnostic systems.

7. FLOCKING

Flocking is an AI technique for simulating natural behaviors for a group of entities, such as a herd of sheep or a school of fish. Flocking, also known as swarming or herding, was developed by Craig Reynolds in 1987 as an alternative to the conventional method of scripting the paths of each bird individually. For a large number of individual objects, scripting was laborious, prone to mistakes, and difficult to modify. In flocking, each bird in the flock is an individual that navigates according to its local perception of its environment, the laws of physics that govern this environment, and a set of programmed behaviors. Flocking assumes that a flock is simply the result of the interaction between the behaviors of individual birds. Also, flocking is a stateless algorithm, which means that no information is maintained from update to update. Each member revaluates its environment at every update cycle. This reduces the memory requirements and allows the flock to be purely reactive, responding to the changing environment in real-time. Flocking has been used with great success in a variety of commercial titles. It can offer a powerful and effective tool for realistic environment creation and unit motion that the player can explore. For example, in a real-time strategy or role-playing game, flocking can be used to allow groups of animals to wander the terrain more naturally and for realistic unit formations or crowd behaviors. For example, groups of swordsmen can be made to move realistically across bridges or around obstacles, such as boulders. Alternatively, in first-person shooter games, monsters can wander the dungeons more believably, avoiding players and waiting until their flock grows large enough to launch an attack.

8. NEURAL NETWORKS

Artificial neural networks (NNs) are structures akin to human brains that can learn various features from training data. Neural Networks (NNs) can model highly complex real-world and game scenarios given a large amount of data. When it comes to game agent design, NNs overcome some of the drawbacks of traditional AI methods. Furthermore, NNs are self-adaptive and adapt well to game environments that change in real-time. Choosing the variables from the game environment that will be used as inputs is the most labor-intensive part of developing an NN. This difficulty is because there is a wealth of information that can be extracted from the game world and choosing a good combination of relevant variables can be difficult. Also, the number of inputs needs to be kept to a minimum to prevent the search space from becoming too large. As a result, it makes sense to begin with the most important variables and add more as needed. Choosing inputs that are poor representations of the game environment is the primary reason for failed applications. Some examples of applications that NNs are being used for are predicting sales, handwritten character recognition for PDAs and faxes, stock market forecasting, credit card fraud detection, Pap smear diagnosis, protein analysis for drug development, and weather forecasting. This list demonstrates how NNs can be successfully applied to a wide range of tasks and how their potential is only constrained by the imagination.

9. DECISION TREES

Decision tree learning is a method for approximating discrete-valued target functions. It is one of the most widely used and practical methods for inductive inference. Additionally, decision trees are robust to noisy data and missing values. Consequently, they are a standard tool in data mining. Because decision trees' learned rules are easy to understand and their training and evaluation processes are effective, they are typically chosen over other techniques. A decision tree acts as a predictor or classifier for classifying a particular example into one of a given set of classes. Every example consists of a set of attribute-value pairs that describe an instance. Decision trees, like biological trees, consist of a single root that branches into multiple subtrees, each of which has its subtrees, until it ends in leaves.

10. GENETIC ALGORITHMS

A Genetic Algorithm (GA) is an AI technique for optimization and machine learning that uses ideas from evolution and natural selection to evolve a solution to a problem. A GA works by starting with a small number of initial strategies, using these to create an entire population of candidate solutions, and evaluating each candidate's ability to solve the problem. Gradually, more effective candidates are evolved over several generations until a specified level of performance is reached. The possible applications of GAs are immense. Any problem that has a large enough search domain could be suitable. In an extremely complex search space, traditional search and optimization techniques are too slow to find a solution. However, a GA is a robust search method requiring little information to search effectively in a large, complex, or poorly-understood search space. GAs are also useful in nonlinear problems. Many applications can benefit from the use of a GA, once an appropriate representation and fitness function has been devised. An effective GA representation and meaningful fitness evaluation are the keys to the success of GA applications. The appeal of GAs comes from their simplicity and elegance as robust search algorithms as well as from their ability to quickly find effective solutions to challenging high-dimensional problems. GAs are useful and efficient when domain knowledge is scarce or expert knowledge is difficult to encode to narrow the search space, when no mathematical analysis is available, and when traditional search methods fail.

11. CONCLUSION

Some game developers have built various degrees of extensibility into their game AI and made it accessible to the user community. These games provide some functionality through which the user can modify or develop customized AI for the game. Also, customizing the actions and reactions of game characters increases the replay ability of the game and gives the player a greater "stake" in their characters. Furthermore, games that have successfully implemented extensible AI usually have large online user communities dedicated to providing tutorials and documentation, swapping their creations, and showing off their prowess in writing their AI. Games such as Quake and Unreal have cult followings of people who enjoy coding and trading their game bots.

REFERENCES

Belova, K., & Belova, K. (2021, November 5). *How Artificial Intelligence (AI) Is Used in Game Development*. PixelPlex. https://pixelplex.io/blog/how-ai-enhances-game-development/

Bourg, D. M., & Seemann, G. (n.d.). *AI for Game Developers*. O'Reilly Online Learning. https://www.oreilly.com/library/view/ai-for-game/0596005555/ch04.html

Duarte, F. F., Lau, N., Pereira, A., & Reis, L. P. (2020). A survey of planning and learning in games. *Applied Sciences*, *10*(13), 4529.

IEEE Transactions on Computational Intelligence and AI in Games publication information. (2013, June). *IEEE Transactions on Computational Intelligence and AI in Games*, *5*(2), C2–C2. https://doi.org/10.1109/tciaig.2013.2266802

Sweetser, P., & Wiles, J. (2002). Current AI in games: A review. *Australian Journal of Intelligent Information Processing Systems*, *8*(1), 24-42.

Yannakakis, G. N. (2012, May). Game AI revisited. In *Proceedings of the 9th conference on Computing Frontiers* (pp. 285-292).