

Artificial Intelligence and Computer Games

Charles Weddle
Graduate Student
Florida State University
weddle@cs.fsu.edu

ABSTRACT

This paper presents how artificial intelligence (AI) is used in computer games to solve common problems and provide game features. Specifically, non-playing character (NPC) path finding, decision making and learning are examined. Different AI techniques are looked at as to how they help provide a solution to these problems and features in computer games. This discussion is followed by a survey of research articles regarding the different type of AI techniques presented.

INTRODUCTION

One of the more popular uses for computer technology is playing games. The computer game has grown from modest text based adventures all the way to contemporary three dimensional graphical games with complex and immense worlds. Computer games typically find themselves on micro-computer technology but have also been built on distributed parallel systems as well. Computer games have a way of evolving and adapting to their environment. As soon as the computer hardware is developed, computer games somehow find a way to maximize the resources available.

Computer games have several uniquely identifiable systems that when put together provide the expected entertainment. These systems include graphics rendering, playing audio, user input, and game artificial intelligence (AI). Not one of these systems makes a game by itself. Rather, it is the synergy created by all of these systems working together that make a worthwhile computer game.

The topic of this paper is to discuss the different features that game AI provides to a computer game along with the different AI techniques used to provide those features. A survey of research is provided for the AI techniques presented that shows how important academic research is to the advancement of computer games.

GAME ARTIFICIAL INTELLIGENCE

What does artificial intelligence provide to a computer game? Some users only want to know how cool the graphics look. If AI was removed from computer games, the games would be so simple that nobody would want to play computer games anymore! Without the game AI, the aliens would no longer move seamlessly through space or the simulated city painstakingly built would never have any problems and winning would be as easy as starting the game.

There are several ways that AI contributes to modern computer games. Most notably are unit movement, simulated perception, situation analysis, spatial reasoning, learning, group coordination, resource allocation, steering, flocking, target selection, and so many more [8]. Even context dependent animation and audio use AI.

It has taken time for many of the uses of AI to make it into computer games. Computer games are computer processor intensive. The systems of a computer game compete for the use of the same hardware resources. Typically the graphic and audio systems usually win that battle leaving the game AI with the little processing time left.

Computer hardware has become more affordable since computer games were first invented and more time and computer resources have been allocated to the game AI than ever before. In fact, some game developers feel that the current state of the game industry is experiencing diminishing returns on graphics and a competitive advantage is now to be obtained through more sophisticated game AI [8].

Even with more attention being provided to developing sophisticated game AI, most of the techniques used in computer games do not rival the goals or motivation behind the academic research of artificial intelligence. Game developers regard IBM's Deep Blue, the ultimate chess machine, to be one of the few true implementations of a game AI system [8]. But it should be taken into consideration that the everyday computer game has goals that are different from academic research and even from those of Deep Blue.

COMPUTER GAME PROBLEMS SOLVED WITH AI

Three common problems that computer games must provide a solution to are non-playing character (NPC) movement, NPC decision making, and NPC learning. Solving these problems is the responsibility of the game AI. There are many more ways that AI is applied to solve problems and provide features in computer games as mentioned above but these three have been selected because they are typical problems to most computer games and provide an interesting AI discussion.

NPC Movement Using Path-Finding

A computer game must provide a way for a NPC to move throughout the game world. When the monster is on one side of the building and the player is on the other, how does the monster negotiate a path through the building to the player? This needs to be done efficiently even when the player is constantly moving throughout the building. This is the problem of NPC movement.

AI Search Methods are utilized to perform path-finding in computer games. Specifically, the A* algorithm is the most widely used search method for path negotiation in computer games [9]. Game developers like using the A* algorithm because it is so flexible. A* does not blindly search but rather assesses the best direction to explore even if that means backtracking. Ultimately, the A* algorithm will determine the shortest path between two points.

The A* algorithm finds a path between two nodes in a graph. The nodes also store information that is essential to the A* algorithm like graph position. The cost of a node takes into account various factors like how much energy it would take to travel a path. The job of the A* algorithm is to find the shortest path between two nodes with the least cost.

Typical A* algorithms have three main attributes, fitness, goal, and heuristic or f , g , and h respectively. g is the cost to travel from the start node to some node between the goal. h is the heuristic or estimated cost to get from this node to the goal. f is the sum of g and h , or the total estimated cost of a path going through this node. The A* algorithm also maintains an Open list of the nodes that have not been explored yet and a Closed list of nodes that have been explored.

The following is pseudo code for the A* algorithm [9].

1. Let P = the starting point.

2. Assign f, g, and h values to P.
3. Add P to the Open list. At this point P is the only node on the Open list.
4. Let B = the best node from the Open list (best node has the lowest f-value).
 - a. If B is the goal node, then quit. A path has been found.
 - b. If the Open list is empty, then quit. A path has been found.
5. Let C = a valid node connected to B.
 - a. Assign f, g, and h values to C.
 - b. Check whether C is on the Open and Closed list.
 - i. If so, check whether the new path is more efficient (lower f-value).
 1. If so, update path.
 - ii. Else, add C to Open list.
 - c. Repeat step 5 for all valid children of B.
6. Move B from the Open list to the Closed list and repeat from step 4.

By turning the monster and player into nodes on a graph that represents the building, the A* algorithm can be used to quickly find the shortest path between the monster and the player.

NPC Decision Making Using Bayesian Networks

In the previous example of the monster negotiating a path to the player, a different problem must first be solved before a path is negotiated. Does the monster even know the player is in the building? This is one example of NPC Decision Making.

Since a game develops an artificial world that NPC's exist in, it is completely feasible for the game designers to give full knowledge of the game world to the NPC's. Amazingly accurate decisions could be made by the NPC's with full knowledge of the world they live in. But how much fun would that be if the computer knew your every move? The user would have no chance of winning. Even if the user could win it would probably not be that much fun because frankly the computer would be acting like a computer!

Game AI is needed to make the NPC's act and react in a human like way. When the player enters the building from the south side, the monster cannot sense the player because of the wall between them. If the player enters the building from the north side and trips causing a noise disturbance, then the monster senses the player and starts to negotiate the shortest path as discussed above. The game is afoot!

One AI technique that has been used to implement this is a Bayesian Network. Bayesian Networks provide NPC's the ability to perform complex reasoning in a human like fashion.

Bayes' Theorem is the foundation of probabilistic reasoning [10]. Let's say you wanted to know the probability that the monster had sensed the player if the player had tripped. Baye's Theorem allows the computer to calculate the probability of the monster sensing the player if the player trips.

This expression can be written as;

$$P(B|A) = P(B|A)P(A) / P(B)$$

Where P(B|A) is the probability that the monster would sense the player if the player had actually tripped.

And P(A) is the probability of the monster sensing the player.

And P(B) is the probability of the player tripping.

These relationships or rules, A causes B, can be arranged into a graph in which the graph captures the statements and interrelationships. Probabilities are calculated for each of these interrelationships.

Three types of inference can be performed on a Bayesian Network [10]; Causal Inference, Diagnostic Inference, and Inter-causal Inference.

- Causal Inference, also known as abduction or prediction, states given that A implies B and the value of A is known then the probability of B can be computed.
- Diagnostic Inference, also known as induction, states that given that A implies B and the value of B is known then the probability of A can be computed.
- Inter-causal Inference, also known as explaining away, states given that A and B both imply C and the value of C is known, it can be computed how a change in A will affect the probability of B even though A and B were originally independent.

If the player quickly moves out of range of the monster and stands absolutely still then the monster might stop pursuing the player blaming the original disturbance on the wind. The monster in this case is performing Bayesian inference by explaining away the original disturbance given the new set of player input.

NPC Decision Making Using Fuzzy Logic

Another AI technique used to implement decision making in computer games is Fuzzy Logic. Fuzzy Logic, as its name implies, states that something can be true to a degree rather than being true or false [11]. Fuzzy membership sets are defined that provide information about characteristics of the game environment.

For example, a humidity membership set might be defined to categorize the level of humidity. Given a humidity value, the membership set would be able to answer if it were very humid or not humid at all. Another membership set might be defined that describes the level of sunshine.

A rule set is established for a certain behavior for a NPC that uses the humidity and sunshine membership sets. For example, the NPC rule set might define an attack behavior. Given input into the humidity and sunshine membership sets, the behavior of the NPC can be determined when a player is encountered. For example, if it is very humid and it is not sunny then attack the player. Or, if it is not humid and it is sunny then greet the player.

The membership sets for humidity or sunshine might have ranges of degree that overlap each other creating a fuzzy truth. An input value of .75 into the humidity membership set might be humid or it might be very humid. The rule sets can account for this and possibly test for both conditions.

NPC Learning

The previous discussion described how NPC's can make decisions using Fuzzy Logic and Bayesian Networks. That's great and can make for human like NPC's but these methods are predictable. Given a set of input the NPC will react in a certain pre-defined way every time! This predictability can be disguised or hidden with a large enough set of inputs but nevertheless the game will produce the same results given the same inputs.

The idea of machine learning is what computer games would love to have. The unpredictable nature of a monster given the same set of inputs time and time again is the ultimate goal. Before long, the monster will just be standing at the north door waiting for you to enter! You don't have a chance!

AI Genetic Algorithms have been applied to computer games to try and implement learning in NPC's [12]. Someone reading about genetic algorithms might think they accidentally picked up a medical science book on genetics. The terms used to describe genetic algorithms include chromosome, genotype, gene, crossover, and mutation. This is definitely mad science!

A genetic algorithm works in the following way [12].

1. Create a first generation population of random organisms.
2. Test them on the problem that is being solved and rank them according to fitness. If the best organisms have reached our performance goals then stop.
3. Take the best performers and mate them by applying genetic operators such as crossover and mutation. Add a few brand-new random organisms to the population to introduce new variety and help ensure against convergence on a local maximum.
4. Loop to step 2.

Genetic Algorithms try and build the perfect specimen and are very complex. This AI technique has not found itself into many modern computer games because it takes a lot of computer resources and time to evolve a specimen or NPC into something worthwhile.

RESEARCH SURVEY

In the previous section four AI techniques were presented that are used in computer games. These four techniques are Path Finding, Bayesian Networks, Fuzzy Logic, and Genetic Algorithms. These AI techniques help a computer game provide non-playing character path finding and decision making as well as learning.

Computer games are a practical application of AI but the discovery of new and better AI techniques comes from the AI research that has been and continues to be conducted. A very small selection of the many available AI research articles is presented. The research articles are presented in the following groups; Path Finding, Fuzzy Logic, and Genetic Algorithms.

AI Path Finding

Path finding is an important part of a computer game because NPC's have to be able to know the shortest path to a goal, for example the player. The most popular way to compute the least cost and shortest path in computer games is by using the A* algorithm.

The A* algorithm works well and is extremely efficient but has some drawbacks. The article "Path finding for human motion in virtual environments" [1] has found that the paths produced by A* may not necessarily produce human like paths. Rather jagged paths are produced that may or may not be what a human would do. This article presents a path finding algorithm for complex three dimensional virtual environments that produces smooth paths.

The algorithm presented considers goal directed motion that can be subject to constraints such as obstacles, cost, etc. Physical dimensions of the human model are also a parameter into this algorithm. Whether the human model can or cannot overcome an obstacle in the path depends on its physical capabilities. For example, a tall human model might not be able to crawl through a small opening.

The article "Greedy algorithms in disordered systems" [3] presents three classes of graph algorithms and their application to percolation and interface problems in disordered systems. Search, Minimal Path, and Minimal Spanning Tree algorithms are presented and classified as greedy because these algorithms take into consideration graph locality only.

Most interestingly is Dijkstra's minimal cost path algorithm. The pseudo code for Dijkstra's algorithm for seeking the minimal cost path between site s and site i follows;

Begin with $pred_{ss} = 0$, $d_{ss} = 0$ and $d_{si} = \infty$ for $i \neq s$.

1. Find Labeled node i with smallest $d_{si} + c_{ij}$, where j is unlabeled. If none exist, exit.
2. Assign distance label $d_{sj} = d_{si} + c_{ij}$ and predecessor index $pred_{sj} = i$ to node j . Return to 1.

The article states that Dijkstra's algorithm is $O(|N|^2)$ for generic implementations and $O(|E|\log(|E|))$ for general implementations which maintain a heap of the sites at the growth front.

The article "Path planning in construction sites: performance evaluation of the Dijkstra, A*, and GA search algorithms" [2] presents, as the title suggests, a comparison between different path finding algorithms. The objective of the research was to analyze path finding algorithms that can generate optimal paths through construction sites when taking into consideration shortest path, safety and visibility. The article states that falls, falling materials, collapses, and electrical accidents are among the most frequent types of accidents on construction sites. Optimal paths through hazardous areas might prevent some of these types of accidents.

The article compares the two deterministic and greedy search methods, Dijkstra and A*, along with a Genetic Path Finding Algorithm. Both Dijkstra and A* are quadratic time algorithms, $O(n^2)$, where n is the number of vertices in the graph. The three areas for evaluating each algorithms performance are shortest path, safety, and visibility. Accuracy, search strategy, and execution time are also considered.

The results of the research showed that the Dijkstra and A* algorithms produced the shortest paths. All three algorithms were successful in producing safe paths by avoiding dangerous areas on the construction site. The Genetic Algorithm produced the most visible path by avoiding areas with narrow pathways or restricted views. The greedy approach, taking the next node without taking into consideration if it is the best choice overall, that the Dijkstra and A* algorithms use always chose the narrow pathways to produce the shortest path, which lead to bad visibility.

AI Fuzzy Logic

Fuzzy logic helps build plausible human like behavior in NPC's in computer games. Fuzzy logic has been used to help make decisions in other applications as well. The article "Adaptive intelligent control of aircraft systems with a hybrid approach combining neural networks, fuzzy logic and fractal theory" [5] presents another application that uses fuzzy logic.

The article presents an approach to handle nonlinear dynamic system control in aircraft stability and control that uses neural networks, fuzzy logic, and fractal theory. The application built a fuzzy decision procedure to select the appropriate mathematical model of which represents all possible dynamic behaviors of the system.

The article gives two examples of nonlinear problems that might occur. Inertial coupling that is a gyroscopic effect that occurs in high roll-rate maneuvers of modern high speed airplanes and flutter effect that is the dynamic instability of an elastic body in an air-stream.

The article goes on to list the fuzzy inference system built for making decisions in this system and concludes that the models do represent the aircraft dynamics in flight. The article also suggests that the neuro-fuzzy-fractal approach is a good way of solving similar problems.

Not all fuzzy logic is the same! The article "Linear partial information with applications" [4] presents a mathematically more simple and more applicable fuzzy logic model than that of fuzzy models based on fuzzy sets-theory. The simplicity in LPI-fuzziness is achieved by not using membership functions. Rather the fuzziness in decision making is made linear by establishment of linear restrictions for fuzzy probability distributions and normalized weights. The article

concludes that LPI-fuzziness is better for decision making than that of other fuzzy logic models and presents several applications of the LPI-fuzziness theory.

AI Genetic Algorithms

A desirable goal in computer games is to achieve NPC learning. Genetic Algorithms optimize through evolution. The best specimen is evolved over time. The article "Genetic algorithm learning and evolutionary games" [6] links Genetic Algorithm theory to evolutionary game theory. The article presents the canonical Genetic Algorithm where each repeated turn of the algorithm consists of variety generating and variety restricting processes. The canonical Genetic Algorithm has one variety restricting process which is Selection. Selection decreases the number of selections in the genetic population.

This article concludes that economic genetic algorithm learning can be shown to be a specific form of an evolutionary game. The results of the research show that Genetic Algorithm learning is an evolutionary two step process; the movement of populations towards an evolutionary stable state and then getting out of the evolutionary stable states. Also, the results show that Genetic Algorithm learning can use the techniques found in evolutionary game theory.

Genetic Algorithm learning is demonstrated in the article "Computers play the beer game: can artificial agents manage supply chains?" [7]. The paper presents a phenomenon in industry practice called the bullwhip effect where the variance of orders increases upstream in a supply chain. The research replaces human players with artificial agents playing the Beer Game to see if these agents can learn and discover good and efficient order policies in supply chains. The artificial agents learn rules in the game by a genetic algorithm.

The following algorithm presented in the article defines the procedure for the artificial agents to find the optimal order policy;

1. Initialization. A certain number of rules are randomly generated to form generation 0.
2. Pick the first rule from the current generation.
3. Agents play the Beer Game according to their current rules.
4. Repeat step 3 until the game period is finished.
5. Calculate the total average cost of the whole team and assign fitness value to the current rule.
6. Pick the next rule from the current generation and repeat steps 2, 3, and 4 until the performance of all the rules in the current generation have been evaluated.
7. Use a genetic algorithm to generate a new generation of rules and repeat steps 2 through 6 until the maximum number of generation is reached.

The research conducted in this article performed several experiments pitting the artificial agents against undergraduate and MBA students. The results of the research found that the artificial agents out performed the students every time. It should also be noted that the undergraduate students out performed the MBA students on a consistent basis!

The article concludes that the artificial agents are able to track demand, eliminate the bullwhip effect, and discover the optimal policies when playing the Beer Game.

CONCLUSION

This paper presented a survey of research articles regarding Artificial Intelligence and described how these AI techniques are used in computer games. AI is an important part of every computer game and is as equally important as computer graphics and computer audio. Without game AI, playing the game would not be any fun!

REFERENCES

- [1] S. Bandi, D. Thalmann, Path Finding for human motion in virtual environments, Computational Geometry Volume 15, Issues 1-3, February 2000, pp. 103-127.
- [2] A.R. Soltani, H. Tawfik, J.Y. Goulermas, T. Fernando, Path planning in construction sites: performance evaluation of the Dijkstra, A*, and GA search algorithms, Advanced Engineering Informatics, Volume 16, Issue 4, October 2002, pp. 291-303.
- [3] P.M. Duxbury, R. Dobrin, Greedy algorithms in disordered systems, Physica A: Statistical Mechanics and its Applications, Volume 270, Issues 1-2, August 1999, pp. 263-269.
- [4] E. Kofler, Linear partial information with applications, Fuzzy Sets and Systems, Volume 118, Issue 1, 16 February 2001, pp. 167-177.
- [5] P. Melin, O. Castillo, Adaptive intelligent control of aircraft systems with a hybrid approach combining neural networks, fuzzy logic, and fractal theory, Applied Soft Computing, Volume 3, 2003, pp. 353-362.
- [6] T. Riechmann, Genetic algorithm learning and evolutionary games, Journal of Economic Dynamic and Control, Volume 25, Issues 6-7, June 2001, pp. 1019-1037.
- [7] S. O. Kimbrough, D.J. Wu, F. Zhong, Computers play the beer game: can artificial agents manage supply chains?, Decision Support Systems, Volume 33, Issue 3, July 2002, pp. 323-333.
- [8] P. Tozour, The Evolution of Game AI, AI Game Programming Wisdom, Charles River Media, Inc., Hingham, MA, 2002
- [9] J. Matthews, Basic A* Pathfinding Made Simple, AI Game Programming Wisdom, Charles River Media, Inc., Hingham, MA, 2002
- [10] P. Tozour, Introduction to Bayesian Networks and Reasoning Under Uncertainty, AI Game Programming Wisdom, Charles River Media, Inc., Hingham, MA, 2002
- [11] T. Alexander, An Optimized Fuzzy Logic Architecture for Decision-Making, AI Game Programming Wisdom, Charles River Media, Inc., Hingham, MA, 2002
- [12] F. D. Laramée, Genetic Algorithms: Evolving the Perfect Troll, AI Game Programming Wisdom, Charles River Media, Inc., Hingham, MA, 2002