Benefits of Genetic Algorithms in Simulations for Game Designers

University at Buffalo
School of Informatics

By: Jason Jones
7/2/2003

Jason Jones Capstone Project

Benefits of Genetic Algorithms in Simulations for Video Game Designers

Abstract:

This paper investigates the benefits of genetic algorithms, specifically concentrating on their use in simulations. Simulations have been found to be effective in training people in companies, the military, and even in educational institutions. There are skills that are acquired through playing games and simulations. These skills have not been able to be delivered by more traditional training methods. This paper illustrates the use of genetic algorithms in entertainment games as well as training simulations. The information is gathered from expert opinions and the conclusions of the author. All this information leads to the idea that simulations are the best possible training method, and genetic algorithms can enhance them even further. There are problems that still exist however even with the use of genetic algorithms. These problems are mainly the lack of user differentiation. Future research and studies are needed to overcome these challenges.

Jump to Section:

I. Introduction

This section introduces the key aspects of the paper, which include: the industry and its trends, the problem being looked at, and an overview of genetic algorithms and artificial intelligence. Each section is important to understand the focus and drive of the paper. They also lay a framework that will be used in the remainder of the paper. They also show how genetic algorithms work, and why they are important. The current state of game production and the problems that exist will be covered. It will also be important to understand where the industry is heading and what the trends that are occurring currently.

Industry:

This section will go into the current state of the video game industry. It will show how the industry has grown, who the important audiences are, and where it is heading. Many facts and figures are included to illustrate how unique and how much potential that the industry has.

The video game industry has continuously grown and spread across many different variations of people throughout its development. This includes a spread throughout age groups,

gender, and race. The industry generates billions of dollars each year, and sells games and systems to many different nations across the world. An example of this worldwide popularity can be illustrated in South Korea. Starcraft is an online alien strategy game. It is fairly popular in the US and Europe, but in South Korea, it is the equivalent of soccer in Europe or South America. Simply put, every young man has played it, and most everyone else knows the rules. Most of the time you turn on the television in Korea, there is at least one channel broadcasting famous Starcraft matches, or matches from games like Starcraft. In this game, players become famous and held on pedestals much like sports athletes in the US do [12], [40].

Not only in popularity are video games a big business, but also in profits. In 2001, there were more than 221 million units sold. These sales totaled more than $6.35 billion dollars for US game developers. This is according to Interactive Digital Software Association, a trade organization for the industry [1]. The $20 billion computer game industry, in 2002, will grow to a $100 billion-a-year business within a decade [8].

All this increased revenue comes from many people that were not once considered "gamers". These are the people that are starting to join the world of video games. Originally, gamers have been thought of as being teenaged male. However due to games such as *The Sims*, women, older men, and other non-gamers are being attracted in unprecedented numbers to the gaming industry. Roughly 43% of all gamers are women. Also the average age of someone who plays a video game now is 28 [24]. To most people these numbers come as a surprise. The increased average age of a video game user increases the need for more advanced games and NPCs. An adult consumer population of games demands smarter and more challenging opponents than a younger audience would. Therefore it becomes essential to develop intelligent

NPCs. Also with more and more people being drawn towards the allure that gaming provides, it is equally as important to create games that people can enjoy.

It is basic capitalism and the desires of people to create new games that have bigger audiences, and generate more revenue. The game companies are always looking for areas for new ways to make money, and people who play games are always looking for newer games that offer new challenges or new modes of game play. Not only will there be increased revenue from the new games that will challenge players in ways that older games have not, but these new games with intelligent characters will have a prolonged life due to the fact that the characters learn and change as the human user changes. Previous sales have shown that the longer the life of a game, the more popular it will be, and the more money that it will make. The biggest selling games are those with the longest lives. Games that become classics are those that have had huge sales and have endured changes in technology and yet still are popular. An example of this can be found in the game series *Zelda*, which was created by Nintendo. It was and continues to be extremely popular, with ten different editions available, on multiple systems. The idea that makes this possible is a factor that I call, and has been mentioned in discussion groups in the gaming industry as replay ability. This is the belief that a game can be replayed many times and still continue to be entertaining. When a game such as *Zelda*, which has been considered as a classic, has made the profits that it has, this is due to what I believe is its replayability. I have owned several different versions of the game, and all of which I have played over and over due to the uniqueness that the game has provided. When a game changes and is continually different for the user, then I believe it will become a very profitable new type of game.

The size of the video game industry is quite immense. It has generated a lot of money and attracted many users. Many of these users are not young teenage male anymore. The importance

in this is that when an older and more diverse groups of people are playing and buying games, there needs to be a more diverse selection of games.

The section has shown how the industry will continue to grow in both profits and number of consumers. With such a large industry there need to be better games that contain better characters. This is to ensure the continued growth that already is occurring. This is due to the fact that the audience for games has gotten older and more money is being spent for these games. One of the main issues that was brought up in this section was the replay ability factor. If a game can be played multiple times, then it will have a longer shelf life, and hence make more money for the designer.

The Problem Being Looked At:

This section contains the important details of the problem. The details are what is currently being done in games, and what is wrong with it. It also will lead into the following sections that wish to illustrate how the problem can be solved. The main aspect of this section is to state what the problems are that exist in games that this paper is addressing. That problem has to deal with the characters in games and their intelligence. The idea being presented is that genetic algorithms (GAs) can be used to solve this intelligence problem. The section also wishes to address different uses of games, such as training simulations. These simulations also require the use of more intelligent characters to be properly used.

The creation of next generation games is the problem that this capstone project wishes to solve. For quite some time gamers have complained of computer opponents that are too predictable and what's worse, they are far too easy to beat and ultimately dominate [33]. Historically games have been long on graphics and short on artificial intelligence [35]. Video games have reached a plateau, which is described in detail by Chris Crawford [6]. He is an

expert in the video game industry who has written dozens of essays about game design and the state of the industry. He also can be considered somewhat of a pioneer for the industry. In the article, he complains that computer games are "reaching a state of moribund stasis". The industry is becoming fully market-driven, and more importantly games are very expensive to develop. Games also are very often “derivative clones of one another.” One can just look at the vast majority of games coming out today and will notice that they are fundamentally the same as they were in the early days of game development in the 1980’s. Games such as "flight simulators, sports games, graphic adventures, role-playing games, strategy wargames, running-jumping-climbing games, ‘shooters’, puzzle games... The only changes that we have seen in these ten years have been embellishments. The graphics, animations and sound are better. The games have more internal detail, larger worlds, and more complexity. But the basic designs have not changed" [6]. Creating intelligent non-player characters is at the heart of this project. I plan to show how genetic algorithms can possibly solve the problem of creating intelligence in these non-player characters. This project wishes to determine what type of game would be the most beneficial with the use of GAs. It specifically is concentrating on simulation games, but also looking at other possibilities for GAs in other types of games.

There are some other aspects that make this project important. Generating more revenue for game companies has been mentioned, so has creating games for a more adult audience. This audience is one that is more intelligent and has greater demands than younger gamers do. There are other reasons for more intelligent NPCs as well. As gaming grows, more people are paying attention to how games might be used, beyond entertainment. Many times I often wonder, what are games doing to my mind? I have been playing games for many years, taking up most of my time as an adolescent. Do I read books differently because of all the games I have played? Do I

watch TV differently? Do I work differently? These are questions that need to be studied in the future.

There are institutions that are waking up to the fact that games might be used to teach and train people. People will willingly spend twenty or thirty hours in a game environment. It can be hard to get people to pay that much attention to textbooks or seminars. So what if training and education might be used in the same model that games are used in. If intelligent agents were used in games, then these new uses of games can become possible. There are a few examples of new uses for games. The Swedish government has sponsored a site, which tracks the use of video games in military training (www.defencegaming.org). A Korean game design firm has joined with the largest national textbook maker to develop "Demiurges," a massive multiplayer online game based on a grade school education. In February 2003 in Washington DC, managers of schools, hospitals and parks met to determine how games might be used to simulate and develop public policy (www.seriousgames.org) [38].

These new outlets for gaming have an evident need for intelligent characters. They have to respond to all the user inputs, but also have to produce equally intelligent responses for a game to be used productively as an educational tool. Different types of characters are present in every kind of game. There are characters that are friend, enemy, or neutral. The characters and their interactions with the user will be discussed further in the following pages.

Non-player characters are any creatures, or obviously characters, in a game that the human user or player does not control. These characters can either be an enemy or ally, also called wingmen, of the user. These characters can either help or hinder the user, and sometimes are neutral in objective where they neither help nor hinder the user.

The way these non-player characters interact with the user depends on the coding of the game. The developer of the game writes the code that the computer follows and executes certain actions. This is the idea of hard coding everything that the computer can do into the game. This seems ideal because the computer will always have a predetermined action for whatever it may encounter. These encounters are generally with the human user of the games, who by nature is unpredictable and does not always follow a predetermined action. At times there are encounters between several NPCs. Problems occur when predictable and unpredictable characters are put together. The non-player character, being predictable, does not always have a response, or at the very least, an intelligent response to the unpredictable human user. This is due to the vast number of interactions and possibilities that happen during a game.

It is impossible for a game developer to hard code for every possible interaction and situation in a game. The possibilities for these interactions can be endless, depending on what game one is playing. Often times when a non-player character encounters a situation that has not been coded for, it does what the human user would seem to be unintelligent. The move or action of the computer just does not make sense to the user and dulls the value of the game and the gaming experience. The term "scripted" can also be used in exchange with hard coding the computer. When the non-player characters are scripted, their behavior is "stilted and not very compelling" [19]. John Laird has done work on a game that has non-player characters that act more human-like and will lead to better game play [19]. He has used AI bots to simulate this type of behavior. Bots are considered "computer generated artificial lifeforms, with the vast majority being designed for you to combat against" [30]. This type of AI has been described as combat AI. The bots in these games are considered "combat AI designed for multiplayer games" [36].

It has been demonstrated in this section that current games are not up to the users needs. The characters are lacking diverse reactions to their encounters with the human players. Many game designers and game design manuals have suggested that the NPCs should have greater intelligence to take gaming to the next level. This next level is to have fully interactive characters that allow the user to become fully involved in this virtual world.

Currently this is not possible because most games use hard coding to allow the characters to interact with the user. It becomes a painstakingly long process to code every interaction and situation that could occur in a game. The repetition that occurs due to hard coding can dull the experience for the user and decrease the shelf life of a game. Genetic algorithms have been suggested as a solution to this problem in the section. The rest of the paper goes into detail on how to actually use GAs to create intelligent characters.

The main point of this section was to demonstrate the importance of simulations for training and education. They are becoming more important and are being used by governments as well as companies for important training functions. Genetic algorithms can be used in these simulations to make them more realistic and allow for the user to be fully involved in the simulation. GAs allow for more realistic simulations of life, which A-life has illustrated, and will be discussed in greater detail in later sections.

Genetic Algorithms and Artificial Intelligence:

Artificial intelligence (AI) is an important part of any game. It allows programmers to increase the ability of the computer-controlled opponents in games. Developing AI that can compete with and exceed the abilities of human players as an opponent in a game is the long-sought-after goal of researchers and developers. However these opponents, often called non-player characters (NPCs), are not up to the standards demanded by the users and purchasers of

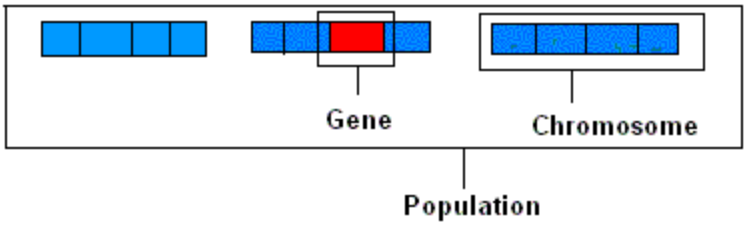
video games [33]. These NPCs can be made intelligent, to the level of a human user, by use of genetic algorithms. This section will illustrate the current trend in developing NPCs. It will also cover how GAs are used within AI. Fuzzy logic and finite-state machines are all covered in this section.

The traditional method of programming NPCs is to use a finite-state machine. This is an AI machine that has been programmed with a set of rules, strategies, and behaviors. As was mentioned earlier, these pre-defined rules, strategies, and behaviors are all coded into the AI, which is put into the game. These actions are alternated each time a certain situation is come across. The AI uses ideas such as fuzzy logic, also called random weighting when making a decision to give the NPC a sense of intelligence. “Fuzzy logic is a superset of conventional (Boolean) logic that has been extended to handle the concept of partial truth -- truth values between ‘completely true’ and ‘completely false’”[17]. Fuzzy logic is used in making machines do a better, more precise job. It uses the idea of partial truth or partial false. However this is not really any sort of learning, or intelligence, it is closer associated with random results. An example of a finite-state machine can be seen in diagram 1.0, which is included in the appendix. It is important to understand that finite-state machines are still used even when involving a genetic algorithm into solving the problem.

There is a need for the computer to learn from its experiences and search for a possible intelligent action or response. Genetic algorithms can carry out this search for best possible solutions. The basic idea is that evolution converges on a solution to a problem. GAs address the challenge of having a program or AI in this case, by having the AI perform tasks, without telling it how to do those tasks. This is done by genetic operators, which include mutation, crossover, and reproduction, all of which will be discussed in greater detail.

Genetic algorithms (GAs) were invented in the early 1970's by John Holland and were fully described in a widely his cited book in 1995 [13]. They apply ideas of genetics and evolution of computation. There are three steps that must be taken for a GA to solve a problem and develop a proper solution. These steps are: define a representation, define the genetic operators, and define the objective function [43]. The objective function is also called the fitness function, which basically is the evaluation function of the solution set.

Genetic algorithms start with a random set of solutions. This initial random set is called a population. Each solution individually in the population is called a chromosome. In programming terms, a chromosome is a genetic data structure. A single bit in the string of characters or symbols in a chromosome is called a gene. They are illustrated in the following diagram:



1.1 Population Example

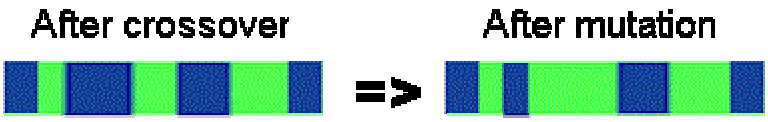
The process in which chromosomes change into new chromosomes, is commonly described as evolving. Every time this process occurs a new generation is created. A generation contains a new population that is different from the previous generation. These new chromosomes in the next generation are called offspring. The new generation can be created in a number of ways. The first way is to use a crossover operator. This is where genes from one

chromosome merge with genes from another chromosome to form two entirely new chromosomes. New chromosomes that only resemble slightly the parent generation are created. Also when crossover is used, chromosomes with low fitness scores are generally changed, or eliminated so that only the good genes are used to create more fit offspring.

The other way is to use a mutation operator. This is the idea where one or more genes in a chromosome are changed so that the resulting chromosome is different from the previous. In nature mutations generally form out of necessity and generally benefit the creature, which the mutation occurred in. However in genetic algorithms mutations are generally bad. Usually they cause the chromosome to not function correctly and render it an unfit solution.

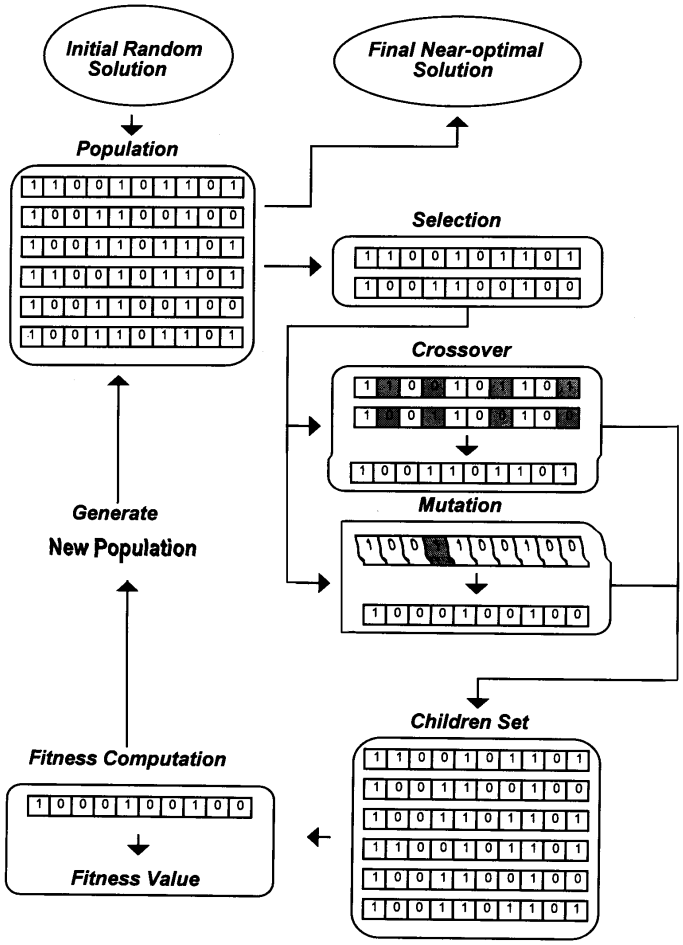
Mutation is used to introduce genes that were not in the parent population, and to "replace genes lost in the selection process so they can be tried in a new context" [16]. A good mutation rate is about two percent of the total population [13]. If the mutation rate becomes too high, then the offspring will lose all of the good genes that the parent generation had that were selected originally. If it is too low then some new genes may never be introduced that could produce the fittest solution possible. These two operators are illustrated in the following diagrams.

1.2 Crossover Example [26]



1.3 Mutation Example [26]

The next generation of chromosomes are evaluated on how well they solve the given problem. Based on how well they act as solutions, they are assigned fitness scores. These fitness scores are determined by the programmer depending on what problem that person is looking to solve. The whole process is illustrated in the following diagram, 1.4 that has a generic genetic algorithm.



1.4 Genetic Algorithm Example [16]

There are different methods for selecting a solution, but generally the higher the fitness score, the better the chromosomes chances of being selected are. These methods for sampling and include, deterministic, remainder stochastic with replacement, remainder stochastic without replacement, stochastic with replacement also called roulette selection, stochastic without replacement, binary tournament selection, which uses the top two fitness scores [16]. The

stronger or fitter solutions are selected to create a new generation, while the weaker, less fit solutions are rejected in order to maintain the population size. This follows the general rules of evolution developed by Darwin, which basically states survival of the fittest.

The cycle of creating new generations, evaluating them, selecting and rejecting continues until either, a most fit solution is found, or the programmer sets a limit on the number of generations that will be formed. Generally the latter is used, as it is difficult to judge what is the fittest solution when more solutions can still be made. In a way it is a sense of predicting when to quit and when to keep going.

The final part of a genetic algorithm that must be addressed is assigning a fitness score. Sample code from a genetic algorithm that illustrates a fitness function can be found in the appendix. The fitness score is represented by *f*. Generally they are scaled before the offspring's parents can be chosen. One of the main methods for assigning a fitness score is linear scaling. Linear scaling requires a linear relationship between the raw fitness f and the scaled fitness f', and is represented by the equation: $f' = af + b$ [10]. Where the coefficients *a* and *b* are selected for each generation to cause the maximum scaled score to be a specified multiple, which is usually double of the population average. This will preserve the genetic diversity in the population of possible solutions. The genetic diversity can be disturbed by an event called premature convergence, "convergent behavior without a guarantee of optimality" [10]. When the extraordinary solutions are left untouched and the GA is allowed to select without interruption, then all of the mediocre solutions would be eliminated. This is not optimal because it does not leave the possibility of one of these mediocre solutions leading to a better solution. Also when only the extraordinary solutions are selected, the population becomes very similar and eliminates the possibility of randomness.

This section has demonstrated the importance of AI and how it can act more realistic with the help of GAs. The traditional methods of programming using fuzzy logic and FSMs was a part of this section. Although FSMs can become predictable, they are still necessary for GAs to work well. Linear scaling was also discussed and how it allows for the preservation of the genetic diversity. The basics of genetic algorithms was the bulk of this section. The importance of crossover, mutation, and selection were all discussed in order to create the fittest solutions. Their ability to handle large solution spaces makes them ideal for the problem at hand. The fitness function and how it allows for each solution to be evaluated, and to determine its fitness.

II. Machine Learning

This section describes the importance of machine learning. It illustrates how important machine learning is, and how it can be used in artificial intelligence. The basic concepts of what is effective learning, and what is not is also included. The section also shows how machine learning can be used in games. There is also a discussion on how important it is in games and how it relates to the main problem of the paper.

What it is:

Machine learning is an important aspect of artificial intelligence. It allows for greater ability and potential of AI in games. This section wishes to give an introduction to what machine learning is. This understanding will be used to clarify how genetic algorithms use machine learning to create better solutions and effectively solve problems.

"In the second decade of the next century, it will become increasingly difficult to draw any clear distinction between the capabilities of human and machine intelligence" [18]. This quote illustrates the importance of learning in machines and what is capable for the future. Machine learning is an important concept in creating intelligent NPCs. When the AI learns about

the game and it's human opponent it obviously becomes more intelligent, and would eventually become a good opponent. In the long run this good opponent would be what the user has been looking for since AI was first used in games.

Despite the fact that the goal of this project and most game developers should be to make the perfect AI, one that challenges users in ways only imaginable, their still needs to be a hint of imperfection present. This is to say that playing against a perfect opponent that makes no mistakes and always wins contains as little entertainment as does a poorly designed opponent that is easy to beat. There is a difficult medium to meet, between too easy and too hard. Either of these choices are not ideal. The AI must be able to learn from its mistakes, while at the same time not become an errorless piece of perfection.

This section has shown how machine intelligence is important for gaming. The AI needs to learn from the user, and change with the user strategies. NPCs need to learn and change with the user. One of the main points is that even though a intelligent opponent is required, there still needs to be a hint of weakness. This is the idea that the NPC cannot be too hard to beat, or else the game will not be enjoyable. The opponent needs to almost win, but in the end fall short and lose.

Application in Genetic Algorithms:

Genetic algorithms are a type of machine learning. This section will discuss their use in machine learning. A machine learns from experience. This experience is acquired through playing many games against human users. The machine has learned when it has changed its behavior. To do this there are a few simple ideas to be considered. First, the solutions that the GA finds that are considered weak, those having low fitness scores are killed off and no longer used. At this point the GA uses only the stronger solutions that work well against the user. In this

sense the NPC is learning, as it eliminates solutions and actions that work poorly against the user, and keeps the ones that work well. More formally, “machine learning is the study of computational methods and construction of computer programs that automatically improve performance based on experience.” [28] When artificial intelligence encounters a particular interaction or action by the human player, it remembers it and learns from it. This is similar to a human being learning from past experiences to make themselves stronger mentally.

Genetic algorithms are considered machine learning due to the fact that they change with the user. As the user changes strategies, the GA will adapt to those changes. They eliminate the weaker solutions and keep only the stronger ones, thus exhibiting intelligence because it has the best solutions for each problem.

Purpose in Games:

The purpose of this section is to show the reader how machine learning can be used in games, and why it is important. The idea is to show examples of machine learning in games, and how those examples have made the games better. There will also be a brief discussion on how the user needs relate to the use of machine learning in games.

“If an expert system--brilliantly designed, engineered and implemented--cannot learn not to repeat its mistakes, it is not as intelligent as a worm or a sea anemone or a kitten.” [37] Although I believe this line to be poorly worded, it illustrates the importance of machine learning. A computer can be designed perfectly, but if it repeats certain mistakes against a user that gives the user the upper hand, than it is not considered intelligent. This demonstrates the importance of adapting to each user. Every opponent of the AI will generally have different strategies and approaches to the game that the AI will have to counter. Just because a particular strategy works against one opponent, it does not mean that it will work against all opponents.

The conventional idea is that the machine has to learn that when a strategy does not work, it should not repeat it.

However I see a problem with this idea. Although it does allow the AI to become an excellent opponent against one particular user, it does not make it a great opponent against all users. As was previously mentioned not every person uses the same strategy. So just because a strategy does not work against one person does not mean that it will not work against someone else. There needs to be an understanding by the AI about the differences between users. While it would be difficult to differentiate between users without some sort of login or screen name, it is an important idea to understand the differences between user strategies.

AI today does not possess the ability to differentiate between users. This is possible using a GA, but it would take many games for the AI to realize this. The GA can adapt to the style and strategy of the opponent over many games played, but would not be an instantaneous recognition. For future study it might prove beneficial to develop some sort of GA, or other AI learning technique that could instantaneously differentiate between users and develop the best strategy to counteract their movements. This would allow many users to play against the same AI opponent and all would encounter different tactics by the AI.

Aspects to Avoid:

Machine learning can be faked through some special techniques used by developers. These techniques should generally be avoided, however the current trend is to use them. These techniques also do not allow for fully interactive NPCs. In order to create the desired level of competition from these characters, this section illustrates what aspects should be avoided.

Machine learning is not simply mimicking the human player and copying what the strategy that the human player has. Although this can be a part of it, called modeled behavior, it

generally a different application than GAs. This also should not be the whole learning concept. When this copying occurs too much, an idea called mimicking stupidity happens [27]. This is when the artificial intelligence is taught badly by the player. This will occur if the player is new at the game and unsure of the controls or what to do in the game. The computer would learn bad strategies and techniques. This is why a reset function would have to be present in the computer. The whole learning process would have to be reset so the computer can learn again once the player is more experienced. Another option is to preset a minimum intelligence level for the computer. This way the computer already has a basic strategy and knowledge and could not be taught badly by a new user initially.

The problem that most designers face is that there is a temptation to create the false impression of the computer learning. An impression of learning can be easily implemented by controlling the frequency of errors in tactical decisions made by the AI, and reducing them with 'experience' as the game is played. This experience is gained as the computer plays game after game against the human user. This creates a realistic illusion of an intelligent learning process, but cannot be used unless the desired behavior is already known. However, this is useless for learning to counter player strategy. Obviously because the set behavior is not known so it is impossible to predict the players actions without previous competition against that exact player.

AI can fake learning simply by cheating. The AI can cheat by certain programming by the developer. This cheating comes in several forms. The AI has access to more information than the player and therefore has an unfair advantage. The whole layout of the board, being able to see the human users position and available resources are examples. While this is occurring the player is unable to see or use these resources until they actually discover them. The AI already has knowledge of these things even before the game starts. What also can be done is that the

NPC can have more resources available than the user. It may require half the amount of money in a game for the NPC to buy a weapon or build something than is required for the user. This makes the NPC seem more difficult and smarter, but it is really cheating that makes the NPC that difficult.

Also reaction time can be used to cheat. The AI has a faster reaction than most users do and that also presents an unfair advantage. The processing power of today's CPUs is increasingly getting faster. They are able to process more events and programs than earlier models. This allows the AI to "think" and react much faster than the user is able to. The computer simulates a world, and the computer can have the fastest theoretical reactions in that world. The increased processing power allows the computer to react in this simulated world extremely fast. In games that are highly deterministic, it can also guess at the future with more accuracy, which is a vital part of "reaction" time. Cheating works when the user does not know about it. However if the user does find out about the fact that the NPC can cheat it can become frustrating and disappointing for them. This will eventually cause the person to stop playing the game, and in the long run possibly could result in the boycotting of a certain developer's line of games. I would believe this would happen to a line of games that is widely known to cheat. For instance I have refused to buy sequels or similar types of games from a company due to the fact that the original was obviously allowing cheating. When the computer is allowed to cheat too much, then it receives too much of an advantage and loses some of its entertainment value.

Peter Molyneux [5] has discussed cheating a few times and is quoted as saying, "It's easy to design an AI-driven player that cheats, that can do anything you can but faster. The real trick is making them insanely hard to beat, making them entertaining, and challenging all at the same time." This is the goal of any developer that has been discussed previously. This goal is to create

an intelligent AI without cheating. Users want to play against a NPC that is very challenging, comes close to winning, but in the end the user prevails.

This section has discussed some of the aspects that should be avoided by developers. One of these main aspects would be to avoid cheating. If implemented and the user has no knowledge of, it imitates the existence of intelligence. Cheating can occur by requiring less resources of the NPC to produce the same results as the user. Incredibly fast reaction times have also allowed the NPC to react to situations thousands times faster than the user can. The goal that was stated was to create a difficult opponent, that learns and does not cheat.

III. Applications

This section covers the use and application of GAs in games. It concentrates on how they apply to non-player characters. One subsection discusses some previous examples of research and games that have used GAs. The examples that are given all show how GAs have been used successfully. These examples lead to the conclusion that GAs are best suited for simulation style games and applications. In the final section experts are questioned as to their feelings on GAs in games. They discuss the applications that GAs have been used in, where they have succeeded, and where they may be best used. These opinions also lead to the conclusion that simulation games are the best place for GAs to be used in.

Use of Genetic Algorithms for Non-player Characters: (Top)

From the title of this subsection, it is clear that it covers the use of GAs for NPCs. The current state of NPCs and how GAs can remedy that situation to make the characters more intelligent is discussed. Also some of the shortcomings that GAs may have when being used for

NPCs is also mentioned. There are two approaches to creating intelligent characters, top-down and bottom-up approach. A discussion of their pros and cons concludes the section.

How genetic algorithms work in general has been discussed, but how can they be used for NPCs? They allow for the character to have solutions to the problems and situations that it may encounter in a game to be varied and evolve as the user strategy changes. This is ideal as the user tends to learn new tricks on how to win. The GA would allow the AI to learn these new tricks and accurately find a new solution to counteract that trick or strategy. Currently in games when a user finds a new trick or strategy often it will usually work regardless of the difficulty setting [27]. The AI does not learn from experience and the game becomes redundant.

Genetic algorithms solve optimization problems. “In an optimization problem, one tries to minimize or maximize a global characteristic of a decision process such as elapsed time or cost, by an appropriate choice of degrees of freedom which can be controlled, and under a set of constraints, linked for example to physical limits” [23].

GAs also have the benefit of finding non-intuitive solutions. Solutions of this nature are the best for games. I believe that these solutions provide the best experience for the player. When an NPC becomes repetitive and does the same action or response every time, the game loses entertainment value and becomes boring. However, if the NPC uses new techniques each time, and changes up its actions when dealing with the user, the game changes and becomes a real challenge. It also increases its entertainment value and generally should have a longer shelf life due to its uniqueness.

Intelligent responses by non-player characters are an example of an optimization problem. The genetic algorithm searches for the best, or optimum response for the given situation or interaction for the non-player character. The physical limits to this problem are

represented by the fact that there are only a few actions or responses acceptable for each situation. An example of this is if a human user approached a neutral non-player character in a game, and asked the character a simple question, there are only a few acceptable responses. If the NPC just attacked, or ran away from the human user, that would not make any logical sense. It would alienate the user because human beings like when games simulate real life. When a non-player character does something that does not simulate real life, then the game is not as enjoyable.

Another example is, if a creature is in a "fight or flight" situation. This is a situation in which the NPC decides based on set rules what it should do. The NPC can either fight the human player, or run away and fight it some other time, hence the name “fight or flight”. Instead of simply providing every creature with two set response choices, GAs could be used to vary the intricacy of each creature's response infinitely. This would simulate the individualism of each creature and contribute greatly to the realistic feel of the game. Creatures in games will be able to truly learn from players' tendencies and mistakes and employ complex strategies to counter them. The NPC could utilize its ability to intelligently analyze the big picture of the battle and the details involved.

An example can be found in any basic action game. When a human user attacks a non-player character that is an enemy, there are only a few acceptable or realistic responses. The character can defend itself, run away, or do nothing and be killed. The following is an old attacking algorithm:

```
if(player.position < attack_range)

    fire_weapon();
    else
```

```
run_at_player();
```

In this algorithm, there are only two options for the computer to choose from for the NPC to act. The NPC can either fire its weapon, or run at the player. This is similar to the "fight or flight" model described previously. When the human player gets to a certain distance, the computer than has the two choices mentioned. There is not much intelligence in this though however. There really is not any strategy involved with this. There is not the idea of running and hiding, protecting itself or any other action other than attacking and fighting to its death. This is the basic model that was used for many years in older games.

There are problems however when interactions such as the above example are oversimplified. There are many variables to the above sequence that could occur. How the human user attacked, e.g. what kind of weapon, what kind of attack, and many other details on the whole interaction. The problem that oversimplifying the attack produces is the fact that it dulls the need for genetic algorithms. If every interaction was so simple and had so few acceptable responses, then machine learning would not be needed at all. GAs are an example of machine learning and their importance has been shown in this paper.

However the main problem that this paper is discussing is the fact that interactions between human users and non-player characters are very difficult and complex. There are many variations and dependencies on each interaction that is impossible for a human programmer to keep track of. If the attack example were so simple, than hard coding would always work. With the few possibilities of interactions it would be very easy for programmers to write code for every one of these interactions.

There are two starting points for creating an intelligent NPC. The first is a more top-down, behavior-based approach. Examples of this include *Oz, Petz,* and *Improv*. The other is a more bottom-up, emergent behavior approach such as Creatures, Silas, Sims', and Artificial Fish.

At their most extreme, the top-down approach requires each behavior to be explicitly defined by the programmer. The bottom-up approach depends on low-level mechanisms to cause high-level behaviors to emerge. The argument for the bottom-up approach was that as the size and complexity of virtual characters grew, it would become impossible to create all the necessary behaviors by hand. The top-down approach to creating virtual characters would eventually become too much trouble to use, whereas the bottom-up approach is much more scaleable.

Game Theory:

There are aspects to understanding how characters in games and simulations think. One of the ways is through game theory. This section describes game theory, its possible use in simulations, and a couple problems that have occurred. Game theory can be an important training tool to understand the behaviors and actions of the characters in games. It is important to understand the use of game theory and its application to simulations. This section attempts to solve these problems.

Game theory is an aspect that can be applied to simulations especially those used for training. Game theory is useful in understanding situations where the fitness consequences of an individual's behavior depends in part on the types and frequencies of behaviors exhibited by other characters in the population. It focuses on how people interact and concentrates on their behavior when attempting to achieve their own goals. When the outcome of a situation or "game" depends on the actions or interactive strategies of two or more people, then game theory applies. During the training an employee or student for a particular situation, the strategies of all users effect the outcome of the game. What strategy the user uses depends on the action that the opponent takes. It studies what should each player do, if all the players were doing the best for themselves as they possibly could. This is the case with simulation training. Each player or

character in the game is attempting to achieve their goals and get what is best for them. The person using the simulation is attempting to achieve stability, order, or some sort of higher goal, while the characters in the simulation are attempting to disrupt that order and get what is best for them. As mentioned earlier, game theory would model the behavior of the characters in the simulation, and would attempt to how those behaviors arise. This would be an important aspect in training, because to understand how the opponent thinks and acts makes the job of defeating them much easier.

There are problems that I see with applying game theory to simulations however. There are usually three assumptions made, with the first being that the characters or agents are self-interested and rationale, agents are capable of the most sophisticated reasoning, and all of the agents understand the structure of the game. The latter assumption I have no problem with, however the first two do have problems. The agents in the simulation would obviously be self-interested, because they can be programmed as such, however to be rationale can be a problem. As has been stated numerous times, often times the characters in games and simulations do not act realistic, or even rational. AI has not been able to understand all of the human factors, and has not been able act consistently and rational all of the time. There are times when the characters do things that do not seem correct, and would lead to the assumption that they are not rational. This behavior can be seen as irrational, but the AI just does not have the correct response for each instance and therefore does not act in a rational manor. The second assumption that the agents are all capable of sophisticated reasoning can be a problem. It would be difficult to say that the AI is capable of this type of reasoning. The AI can act intelligently, but to say it uses sophisticated reasoning is a stretch. If a GA were used than the AI would be able to search for the best solutions for each problem. The AI made logical steps to find the correct solution,

but is this really reasoning? I would imagine that it could be considered reasoning, and that the AI is capable of making connections and finding solutions logically. However I would not agree that it is capable of sophisticated reasoning. Another problem with game theory is that it has been unable to be applied to complex problems. An algorithm, particularly a GA, has not been found to adequately handle all of the strategies and rules in complex games. Simple games such as the prisoners dilemma, which an example of can be found in the appendix, have been successfully used with game theory. Simulations are obviously very complex and have many strategies and possible character behaviors. Due to this aspect game theory would not currently be able to handle the complex nature of a simulation. To be as effective as possible the simulation should be as complex as the world it is modeling.

Game theory studies optimizing agents, those attempting to achieve what is best for themselves, and which GAs work well with. "A Genetic Algorithm is applied in game theory to find equilibrium points in non-zero sum and non-cooperative situations, and in the game of iterated prisoner's dilemma to explore the possibility of evolving cooperative behavior" [4]. GAs find the best possible solution and then apply it to the problem that is currently being solved.

This section has demonstrated how game theory can be used in simulations. It has discussed its importance in understanding behaviors of characters in games, and how to understand behavior when each character is attempting to do what is best for themselves. Also some of the problems such as acting rational and sophisticated reasoning when dealing with AI have been discussed. The main problem that currently exists with game theory is its inability to be applied to sophisticated games or applications. How GAs are used in game theory was also presented to show their value when dealing with multiple characters. GAs also allow for the most efficient and best possible solutions when dealing with characters that want the best outcome for

everyone. They also allow for random solutions and are able to handle the evolution of strategies throughout a game.

Previous Studies and Examples of Genetic Algorithms in games: (Top)

GAs have been used to create more interactive and intelligent games. This section covers some of previous studies and applications that have used GAs. Some games are discussed, such as GP*Othello, Half-Life,* and some simulations. These examples show how current research and development are using GAs to solve some problems in games. The idea is also to show how current games and simulations have needs that GAs can solve. These problems do not currently have the most effective solutions, and the argument is that GAs can solve these problems more efficiently.

Some games have started to us genetic algorithms to increase the intelligence of NPCs. This is a relatively small and select group. If there was to be continuing research in the field, then it possibly will create more games that use genetic algorithms for AI. The types of problems that GAs can handle better than heuristic search methods has yet to be determined. Generally, the use of genetic algorithms has been kept to simpler games, such as chess, othello, etc. These games pit one opponent against another. They have simple rules and only require the computer to control one NPC.

One such example was done at Columbia [20] and was called the GPothello. The game being used was othello. Function trees were used and were called “dudes”. The experiment used a small population, made the mutation rate high, had many generations, and allowed for “dudes” with lower fitness scores to survive and create offspring. Also different selection types were used to see which one would produce the fittest “dudes”. The idea of the experiment was for the computer to learn using genetic algorithms how to beat Edgar, a smart computer program and

also an expert system. Also the “dudes” played against random move players, which are programs that randomize their moves, but are not considered expert systems. Not only did they try to beat Edgar and the random move players, but they also learned from playing against them.

What was found from this experiment was that the fundamental idea that fitness scores improve as generations evolved. There were thirty-four “dudes” created. Edgar crushed most of the “dudes”. Only one “dude” from the original experiment could beat Edgar, but was 7-43 against random move players. After some modifications to the successful “dude” two of the new “dudes” could beat both Edgar and the random move players consistently. Some concepts could also be found from this experiment. First off, there was no difference found in the selection method. Statistically all of the selection methods turned in the same results. One of the negative results was that the “dude” that was evolved to beat Edgar was over taught to win against Edgar, but not against random move players. The function tree was only taught to beat Edgar and was unprepared for random moves of other players. Changes were made to the code to increase randomness and better results were found, that were illustrated above.

Another example of a game using GAs is a game called *Quake 2*, which was developed by Grey Matter Studios. This game uses genetic algorithms to develop control systems that help the NPCs react to other elements such as obstacles, ammunition, and enemies in the game world. The NPCs basically learn how to beat the human user without the use of cheating. The research for Quake 2 has lead to the production of AI that can predict what a human user will do next. It will predict the most likely move and react to that accordingly. This is similar to playing against a real person who would try to predict another’s moves to win the game. The concept works in the same manor as in chess, where GAs have been programmed to predict the opponents moves to defend against them. Grey Matter and its researches would not give me any information on

exactly how the GA was programmed to produce this predictive nature. In my contact with them, they said they were unable to give out any type of programming or trade secrets for any of the specifics.

Andrew Vardy [42] who did AI research using GAs extensively has admitted, "researchers have been able to generate predictive targeting systems, which shoot where the target is heading." This is similar to the above advancement in *Quake 2*, except this is a targeting system. This is an advancement in the achievement of creating intelligent NPCs. This is the idea of the computer predicting what the user will do next. While not totally a form of intelligence it is a step in the right direction. Predicting the path of a user does not illustrate intelligence unless the prediction is correct. If the NPC shoots inaccurately more of the time than not, then this obviously would not be considered advancement. The targeting system works by using a predictive shooting mechanism that evolved over many generations. It would determine which direction a target was heading in and fire 3.75 degrees ahead of it. This is certainly not optimal. It would be better to change this angle depending on the speed and relative distance of the target. Still this behavior evolved without anyone having to actually program it in, so that illustrated the ability of GAs.

The biggest achievement however has come from a game called *Half-Life*, which was developed by Valve Software. This is a first person shooter (fps) game that has made huge strides to advance game AI, and especially GAs. This game uses the most advanced AI in the field when it was released. Enemies exhibited squad- and flocking behavior simulated from actual real-life animal flocking patterns, and organized attacks with fellow "monsters", which are all NPCs, by assessing their chances of winning in an all-out battle. This achievement incorporated into *Half-Life* was made possible by including a notion of self-preservation. This is

the will to live and fight another day, a response that had been common among human users, not NPCs until the release of this game. Previous games had used NPCs that simply used pathfinding and attacking the player, often running directly into the enemy's gunfire. "Pathfinding is a graph search to find an optimal (lowest cost) path from the current node to the desired destination" [46]. Pathfinding is accomplished in games by A*. This is a search algorithm that is used to find the shortest path through a search space to a goal state using a heuristic method. The AI that was included in *Half-Life* was considered "combat AI" [36]. After the game's release and ensuing popularity, most games in the same genre included similar AI in their games to accomplish a similar effect.

Also included in *Half-Life* are "emotional states" for each enemy. They include scared, aggressive, confident, etc. For instance, if significantly outnumbered, the NPC might panic or retreat to find back up. This is somewhat similar to the "fight or flight" state of NPCs. The basic concept is the same, but in actuality the two are quite different.

What *Half-Life* uses is more advanced than "fight or flight". There are alternatives to just fighting or fleeing. The characters can attack the human player, until their health reaches a certain point than retreat. The NPCs can group together and attack the user based on a learned behavior or strategy. The basic idea that *Half-Life* involves is that there are a multitude of variations to the "fight or flight" scenario. Some may be just slight changes that are generally classified as fight or flight. An example of such is a NPC retreating to a location that contains many other NPCs that are all on the same side. While this is fleeing, there is a strategy and a sense of intelligence behind this.

There is another type of game called *Creatures*, developed by Creature Labs, used genetic algorithms. In this game, users were able to train and breed fantasy-like mammals whose

behavior was controlled by the integration of a neural network [7]. This uses a model of biochemistry and an artificial genome with crossover and mutation. This game is a great introduction to the world of artificial life technology and virtual life simulation. One can create life, evolve a species, and teach the creatures adaptive tricks. The user can also create an infinite number of different creatures, and through the theories of genetics, no two will ever be the same. The creatures that can be created in this game can be taught. One can teach the creatures to eat, speak, and interact with the user and other creatures.

Another aspect to consider in the use of genetic algorithms, which *Creatures* could be considered a part of, is artificial life (Alife). Artificial Life is a field of scientific study that attempts to model living biological systems with the use of complex algorithms. Genetic algorithms have been used in Alife. Alife also attempts to build systems that exhibit some of the key properties of life, such as adaptability and robustness in complex environments. It is different from AI, which has traditionally focused on cognitive and intellectual abilities of human beings. Alife has been used as an entertainment source, as *Creatures* has illustrated. Alife has also illustrated the benefits of genetic algorithms in creating NPCs that are not predictable and seem alive. “When people talk about wanting great A-Life (or great AI, for that matter), what they really want is the experience of interacting with something truly alive" [39]. This is the feeling of playing against an opponent that exhibits intelligence and interacts with a human user in more than just one set way. To be alive, or in terms of Alife to act alive, the computer must act obviously like a living creature. This to me would mean to act unpredictably and unscripted.

The popularity and recognition that *Creatures* has received is reasonable evidence that users are looking for new types of experiences beyond traditional computer games. This is mostly due to the fact that the characters in this game are always changing and can act as creative

as the user trains them to be. Older games use characters that are not as original and creative as those used in many Alife games. The greater challenge and entertainment value for the user lies in the games that have these types of characters. The key selling point and main reason for their popularity is the idea that they possess fully interactive environments.

Virtual Petz, by PFMagic, is also an example of Alife. The goal of the developers was to create the best and most interactive "illusion of life" available on a personal computer. To do this the designers combined a direct interaction interface with very active and intriguing characters. These characters had many lifelike qualities about them. These well thought out characters allowed users to easily forget they were just playing a game, and form relationships with the characters that they created, and that they can adopt in the game. This connection would lend itself useful in a simulation as users get emotionally tied into the game, and thus makes it more believable. If using the simulation for a training device, when it is most believable and the users have emotional connections, then it can be most effective.

As recently as a month ago Clemens Lode has implemented a StarCraft GA to determine build order. The GA is called the Starcraft calculator and build order optimizer [22]. This is an optimization problem that determines the best order to build up an army or base in the game. It works by “simulating an abstract StarCraft:Broodwar environment and then calculates the time a certain build order needs. By comparing different build orders, randomly changing them and taking the best build order for the next generation, the build order is improved step by step. StarCraft build orders fit perfectly for this algorithm called 'Hill-Climbing'.” The idea behind this is that each build order that is used is timed and evaluated depending on its efficiency. The program will find the most efficient order for the user.

Many examples of the use of GAs have been given in this section. Some entertainment games have been mentioned that use GAs for certain problems. These problems are generally small issues that GAs can handle. The problem that has been found that using GAs for such complex problems as creating intelligent characters has not been successful. This is due to the difficulty of creating an evaluation process that can encompass all of the strategies that are present in entertainment type of games. GAs have been successful in simple games, such as chess and othello, as well as simulation games. *Virtual Petz, Black &White,* and other Alife games have been very successful at using GAs to create fully interactive characters that exhibit intelligence. These have lead to my idea that GAs can be successful within simulation training type of games. The abilities that create intelligent characters in Alife games, are the same abilities that are needed in training simulations to make them fully interactive.

Expert Opinions: (Top)

Opinions in the field of artificial intelligence and game design have come to a general conclusion on the use of genetic algorithms for NPCs. Most believed that while there is hope for a new design of a genetic algorithm, they lack the speed and sophistication to be able to handle a task of creating intelligence in NPCs. While genetic algorithms have never been fully programmed into a game, there have been some successful uses of genetic algorithms, as has been mentioned earlier. However a fully enhanced NPC made intelligent by genetic algorithms does not seem possible in most expert's views. This section outlines many experts' opinions of GAs in games. They cover both the positive and negative aspects of their use.

In an interview conducted with Patrick Doyle [9], a Ph.D. student at Stanford University who has done extensive research into creating intelligent NPCs. His research did not use any sort of learning mechanism, be it genetic algorithms or neural networks, etc. Instead he is

experimenting with a fixed agent architecture to achieve his goal. He felt that genetic algorithms would be far too problematic for NPCs that are pretty high-level. These NPCs help solve puzzles and talk with the user, as opposed to NPCs that are primarily combatants in FPS games.

As for his ideas on how to start to create intelligent NPCs, he suggested using an agent that was purely reactive, and just had a list of rules like "if there's a wall in front of me right now, turn left". That might be a reasonable framework from which to start building an algorithm's components. While this is not a sense of learning, it is a start to create solutions that contain intelligent responses to situations that can occur in the game.

In an article by Steven Woodcock [45], who has 16 years of ballistic missile defense work building massive real-time war-games and simulators. He wrote that in 2001, "there was much more focus on making the more traditional approaches (state machines, rules-based AI's, and so on) work better. The reasons for this varied, but essential aspect, boiled down to variations on the fact that these approaches are better understood and work "well enough." Developers seemed to want to focus much more on how to make them work better and leave exploration of theory to the academic field." This trend continued in 2002 and on through today as developers use these traditional methods because they work for the games right now. The problem with this belief is that the non-player characters in today's games are not as intelligent as the users would desire. The traditional methods have reached a level of sophistication that they cannot overcome. This is not sufficient because the characters do not act as intelligent as a human player would, which has been illustrated earlier.

Woodcock also continued in his article to say "there wasn't a single developer at the Game Design Conference 2000 that reported using GAs in any current projects, and most felt that their appeal was overrated. While last year's group had expressed some interest in

experimenting with using GAs to help with game tuning, the developers who had tried reported this year that they hadn't found this to be very useful. Nobody could think of much use for GAs outside of the well-known "life simulators" such as the Creatures and Petz series." The simulation of life seems to be a very popular area for GAs. It has helped to create unpredictable responses by the NPCs that could not be created with traditional methods.

To further this point Andrew Vardy [41], who is a Ph.D. student at Carleton University in Ottawa Canada, added, "GAs created some interesting creatures, but one could have fairly easily designed similar creatures by hand." He found this to be true in a simulation that was supposed to create computer opponents that exhibited "continuous novelty". He also mentioned that "getting GAs to work on any really hard problem is always more difficult than you expect. They offer the promise of finding great solutions to problems with practically no effort. Yet, given our current understanding, effort is definitely required."

While this view tends to shut the door on any possibility of GAs working to create interesting and intelligent NPCs, there is some hope. Vardy felt that while it was not possible yet, that being the key word, to use for NPCs intelligence GAs could be used in a limited role. This role has yet to be identified unfortunately due to lack of research and interest into the use of GAs for NPCs.

Greg James [15] who is involved in the gaming industry in Canada, and has an extensive knowledge of GAs, and their use in games also had some useful insight. He knew of a couple examples that I was unaware of. He believed that "there was not anyone who is using GAs for commercial games at all at this point…there was a rumor that Sid Meier's Alpha Centauri used GAs as part of the game tuning process "prior" to release… and that The Sims

uses a simulated annealing-like hill climbing process using happiness gradients to deal with the motivation of the Sims."

James' opinion of GAs for non-player character intelligence was similar to the other experts' views. He felt that "the thing about GAs in game programming is that they're most likely being programmed by game programmers, which means they will not end up appearing in the traditional research literature." His personal feeling of GAs are that they are "impractical to deal with a solution space as large as "NPC intelligence" and are better used to work out the details of their behavior once you have decided what it is you actually want them to do. For example, is the NPC playing Ghost Recon, Starcraft, Diablo II, or Worms 2? One GA probably would not be able to solve any one of those problems completely."

Most of the experts opinions were pretty consistent. They all believed that in entertainment games that GAs were not a practical way of creating intelligent characters. The problem was far too big for GAs to handle. The main difficulty was creating an evaluation function that could properly identify all of the strategies that are necessary for the NPC to be a successful opponent to the user. The consensus seemed to be that the minor details, specific to each particular problem are where GAs could be used. An example was the use of a targeting system. Many also agreed that GAs have been, and could be very successful in simulation games. Some also thought that training simulations would be an excellent arena for the use of GAs.

IV. Authors Views

This is the final section of the paper. It concludes all of the information that has been presented and demonstrates how it can be effectively used. These sections reflect the author's recommendations to solve the problem and his interpretations. The purpose here to show how all

the previous information can be used effectively to solve the problem that may not have been presented in other literature. Then finally a conclusion that wraps up the main arguments is presented. All of the subsections represent the authors views and reflections.

Analysis and Interpretations: (Top)

Many avenues have been explored on how to create intelligent characters in games. Many of these have failed to fully achieve their goal. This section will demonstrate what the author concludes on all of the information that has been presented. Also the interpretations of all this information will be discussed as it applies to the main problem that the paper is attempting to solve.

A lot can be concluded from extensive expert interviews and information gathered from research involving genetic algorithms. The first main idea that can be concluded is that genetic algorithms are not able to make NPCs intelligent in complicated games. This is partly due to the fact that it can be difficult to write the fitness function. This is the crucial part of any genetic algorithm. The key is to understand what will be good rules, and what are bad ones. As has been explained there are vast amounts of solutions and rules that must be accounted for when a complicated video game is being used. This is where the major problem lies, the encoding of all these strategies.

First however one should look at the fitness function itself. It is written so that it must evaluate each of these possible solutions to evolve them to the fittest one. GAs are a slower search method and that may slow the process down, and in the end cause for a solution that is not the fittest one possible. The problem with advanced games is that there are so many rules to account for. The AI must also handle all of the strategies of the game. This way the NPC will be able to play and perform like a human user would. To enter a strategy into a GA, there must be

an analysis of each by the user, and then code those into the GA. By this I mean that each strategy receives a certain score. A simple example can be found in chess.

In chess each piece can make a certain type of move. The pawn can only move one square, the bishop can only move diagonally, and so on and so forth for each piece. Each move that a particular piece can make is given a certain score. The programmer determines if the move is a good one or a bad one, and assigns a score for this. The good moves are given a positive fitness score, whereas one that is considered bad would be given a negative score. Not negative in the sense that it is below zero, but in the sense that it would be a low score.

The main problem, which was mentioned previously, that has to be overcome to enable the use of GAs for NPCs, is to be able to encode all the strategies and rules into the bits. This process is easier when using a simple game like tic-tac-toe, chess, or othello. However when a complex game is used, such as Quake, Half-Life, or any other advanced game the process becomes much more difficult. With the simple games it is easy to understand and identify all of the rules and strategies used. However when using an advanced game there are many rules that have to be followed, and there are even more strategies that are needed in order to create an intelligent NPC.

To overcome this complexity, there is a slight possibility of dealing with the encoding of all of the strategies and rules. The main way is to make each strategy as simple as possible. If one is using a fps, then a score can be assigned for each move or action that a NPC makes. When fighting against a human player, a headshot would be assigned the highest score, because it is the deadliest, and the most effective way of beating the player. A body shot will not receive as high a score as a headshot will. A miss fire would receive a very low score. So the idea is that the higher scoring solutions would be the fittest. Therefore during the selection process the solutions

with low scores, such as miss firing would be eliminated. The NPC would become a tougher opponent and would be better at defeating the player. Then the next step would be to encode the strategy for avoiding and surviving against a player. It would have to be broken down much like shooting at the player was into simple rules.

This will be a time consuming process though. When every strategy and rule is broken down it takes a lot of effort and time to account for all of them. It also creates a situation in which it is tough to judge whether all the time spent is worth all of the effort. Does simply making the NPC a more accurate shooter, or making it better able to hide, create the sense that it is more intelligent? This could be debated for either side. I believe that if the NPC has a better aim, and also is tougher to hit and find, then it is more intelligent. When looking at human players, they are considered smarter and better opponents when they are well played and difficult to beat at a game. If this same idea is applied to NPCs, then if they have the same characteristics they will be considered more intelligent. With the NPCs also acting with better characteristics, it will create the sense of playing against a human opponent and create will lead to increasingly random and individualized responses, giving every NPC a realistic personality and behavior pattern in the context of the game.

However just breaking down the rules and strategies into simple strands is not entirely the solution. The NPCs have a goal state, which is the end goal for each action or simulation. They also have a procedure or strategy to reach that goal that is already designed. This would also lead to the same conclusion that the problem lies not in the fitness function, but rather in the encoding of the strategies. To do this, each NPC should be modeled as a finite-state machine (FSM) (fig 1.5). Problems can arise from this type of modeling as well. At times FSM can be predictable, which is one of the problems that has been occurring currently in games, and that this paper is

trying to overcome. However during the description of GAs it was mentioned that FSMs are still needed to allow the GA to carry out its processes.

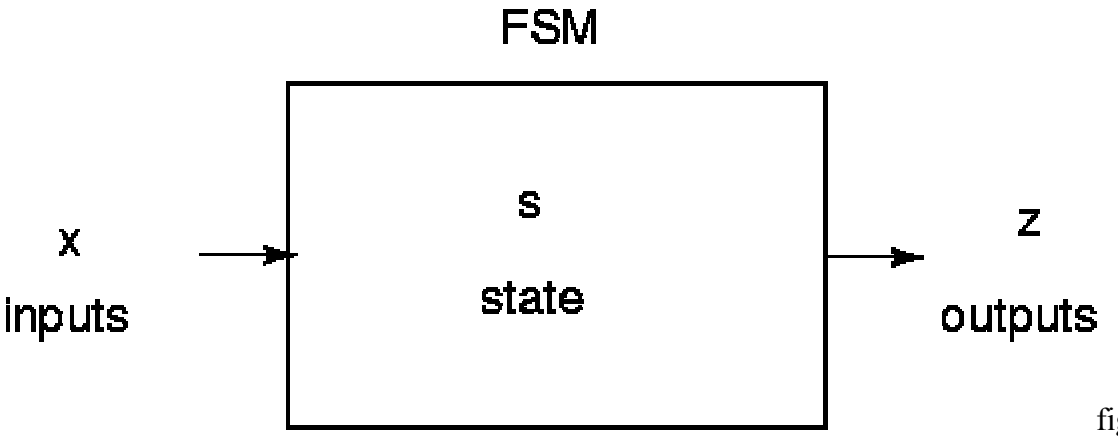


fig 1.5

A problem with GAs is that they are problem specific, meaning that they only work well for a specific type of problem. They cannot be generalized and applied to many different problems. The fitness function that is written must be specific for the solutions that it is testing. In order to score a solution, the fitness function must be coded for a specific problem in order for the GA to work properly.

This will be a problem for any video game developer. If GAs were more general than one could be used in many different games. However due to their ability to be able to only handle one problem, research and development must be put into creating a GA that can handle each problem that they could be used for in a game. This research and development costs a lot of money to the game developing companies. It is obviously beneficial to create a game that has as little development costs as possible, and still satisfy all of the user's needs for the game.

Games are very expensive to develop and design. Times have changed so that no longer can the "garage" programmer sit at home and create a game that will be popular, and create large profits. It is extremely expensive to design state-of-the-art graphics, and more importantly even more expensive to design the AI for the game. This has created an industry in which only the big

companies can afford to create games. It also has created a market in which companies are becoming less likely to take chances on ideas that do not follow the mainstream. When companies spend large amounts of money on a game, and it flops, then that company is less likely to take chances on other games considered projects. These project games are ones that do not follow the design of other games produced by competing companies that have been popular.

Costs of Development:

The exact costs and capital that companies allow for any given project is kept secret. This is obviously confidential information that companies are not willing to give out. Even after scouring through many postmortem reports of games, all exact costs were kept secret. One can estimate generally the expenses of implementing AI into a game. Considering most teams have one dedicated AI programmer, who have very good salaries, estimated at well over $100,000 per year, and a few designers that get partly involved in the AI. These designers make average salaries, but over a course of two to three years, there becomes a lot of money needed for the upfront costs. The high-end games developed by the large companies generally are in the two to three year range [15]. The average game by a smaller company was generally in development for twelve to eighteen months. Also to be included is the fact that AI has a strong impact testing, which is not cheap by any stretch of the imagination. This adds more money to the total expenses. The total expenditure for all this time and effort is generally in the millions of dollars.

Online Gaming:

To offset these AI development costs and the use of AI in general, some developers have tried to create online games. Online gaming has recently become a trend in the gaming industry. Many games have attempted to create a sense of real world action happening in a virtual world environment. There is a draw for the user, playing against friends and family is a feeling that is

not comparable to playing against a computer. This is the feeling that game developers have tried to build a game for. There are games that are being developed that are meant only to be played online against other people. A single player version of the game is left out. The idea behind this being that if people are playing against other people, then there is no need of NPCs, especially ones that require the development of an intelligent agent. *The Sims* online had this very idea in mind during its creation. Every single character in the game would be controlled by a human player and eliminate any sort of NPC [21].

This idea would have problems when designing and implementing a simulation that was meant for training. Generally the people using the program would be beginners, or at the very least have little experience at what they were being trained in. to be effective as a training simulation the opponents of the users should be experienced and quality opponents. When only trainees are using the program then there is a lack of experience and the simulation is not as effective as should be. The characters in the simulation should be NPCs and not controlled by humans, to allow for the characters to be challenging. It would be impractical to hire someone strictly to control the adversary in the simulations to compete against the trainees. The NPCs would bring a sense of order and realism if they were able to act intelligently.

Having all the character being controlled by human users has some other problems, both economically and realistically. While it was promoted as being the game that would attract a large number of people, estimated at 1 million subscriptions per month to play the game online, *The Sims* online has not lived up to that potential. Six months after its release, there are only 97,000 subscribers and the game software itself has plummeted in price [32]. While the author credits the fact that the game was not designed for the right audience, there are other issues with the poor sales. The author claims that the game and its producer, EA, were counting on a large

audience of people not considered gamers to buy the game. These people are called the casual gamer. The idea was that these people were not willing to pay a monthly subscription fee to log on and play with other people. Perhaps the fact that sometimes it is just easier and more practical to play a single player game against the computer is the reason for the low sales of the game? Often times, single player mode has more entertainment value for the user, especially when the game is designed for people who would not be experts or experienced playing games. They may feel apprehensive about joining a network of people who may or may not be better than the user is.

Another problem with online gaming in general is the connection to the Internet question. Despite the fact that broadband connections, such as DSL and cable modems are increasing in sales, the vast majority of homes still use 56k dial up modems. These slower connections slow down online gaming and are less advantageous for users to play games online. I would predict that a majority of experienced gamers mostly have a broadband connection. Approximately 31% of home internet users have broadband connections at the end of march 2003. This is an increase from 24% the previous year [14]. with the increased graphics and amount of users allowed to play in a particular game at any one time, it is becoming essential to have a broadband connection. Those with dial-up connections are unable to play these newer games at the same speed as broadband users. Often times lagging occurs, which is when the game seems to pause or go very slow, due to the fact the connection is not fast enough to send and receive all of the information. Until broadband becomes a majority and a cheaper resource, online gaming will not overcome the rut that it is currently in. According to In-Stat, there will be 29.6 million home networks that have broadband by 2006, compared to the 6.5 million home networks that currently exist in North America. The main reason cited for this growth was due to online

gaming [25]. These statistics illustrate a need for a broadband connection to play games online. Until broadband becomes a majority, then game designers should concentrate on single player mode to accommodate dial-up users.

Single player mode therefore must be a part of every game since many users have been shown to be unwilling to buy into strictly online games. Also “the infrastructure of the very Internet that designers are relying on cannot easily or effectively handle the burden of intensive, real-time interaction, especially when using "bleeding edge" 3D graphic engines” [11]. The technology is just not there yet to be able to fully handle a massively multiplayer game. However the technology has improved, since the article quoted was written in 1997, and 3D engines and the Internet itself has improved drastically since then. The basic premise that online games have not done well also still applies today as it did in ’97.

Other online games in addition to *The Sims Online,* that have failed are *Earth and Beyond* and *Motor City Online*. Many of the popular online games that are considered successful, are put into a category called massively multiplayer games that support thousands of players at any single moment. These successful games include, *Ultima Online, Everquest,* and *Asheron’s Call*, all have thousands of different characters. While the human user controls most of these characters, there are many roles that exist in these virtual worlds that need to be filled by the characters.

As mentioned before, every character in a game may not be used by a human player. In many fantasy-medieval role-playing games there are characters that exist that are not desirable for people to be. People want to play characters in games that are the adventurer, soldier, or hero. They do not want to be the blacksmith, carpenter, or any other menial character. People have normal jobs and lives in the real world. They want to play characters that they would probably

never get to be in their real life. Due to this fact there is a constant battle for the most elite characters in games, and the inability to find people to play characters that are less desirable. When this occurs, there are two generalizations that can be made. The first is that people will not subscribe to an online game for fear of undesirable characters, and secondly that the game will not be complete unless all characters are being used by a user.

To overcome this obstacle GAs could be used for intelligence of the NPCs in online gaming as well as single player games. Patrick Doyle [10] has written a paper on creating intelligent NPCs in online gaming. The author feels they are important to increase the realness and entertainment of these online games. He also believes that they allow the users to stay in character and act naturally in the virtual world. These intelligent NPCs also would be the supporting roles in the games that are the less desirable characters as perceived by the human user.

The first major hurdle is the fact that comprehending every aspect that encompasses intelligence is very large. Not only does the GA have to understand every variable, but so does the programmer. These variables are the appropriate responses to their corresponding situations. When a human user performs a certain action, the NPC must be able to act intelligently in order for the time and money spent developing the GA to be considered worthwhile.

That in and of itself it a whole other problem. What exactly is intelligence, and is it a universal standard? A formal definition is the "capacity for understanding and for other forms of adaptive behavior; aptitude in grasping truths, facts, and meaning" [31]. Many people would agree with this meaning, but can it be applied to AI too?

There is not any single universal standard for understanding exactly what intelligence is. Every person probably has his or her own idea of what intelligence is. The same is true for

people's opinion of what intelligence in AI is. Passing the Turing test supposedly illustrates intelligence in machines, but not everyone would agree with that. Some would argue that conversing is not solely a measurement of intelligence, which I am in agreement with. If the AI can convince, or fool people into thinking that it is human, than it should be considered intelligent. One could converse with an actual person, or at least attempt to, and not be able to understand them. This would hypothetically make them unintelligent. However I would argue that not only should the AI have a conversation, but also perform some sort of act as well to verify its intelligence.

This act, as would apply to this paper would be to play a game and be able to play like a human. There are many programs that can play and beat human players in chess. That however does not make these machines intelligent. Their massive computing power gives them an unfair advantage. Deep Blue which beat world chess champion Garry Kasparov in 1997 had an unfair advantage. The machine could process 200,000,000 moves per second, compared to Garry's two moves per second [35]. If the AI were to play a more advanced game, one that has a lot of unpredictable scenarios, such as a massively multiplayer online game that is constantly changing. This constant change would negate the superior processing power that the AI possesses because it requires an understanding of the current situation to act intelligently. The NPC would have to act and converse much like a human user might do. Doyle [10] has argued that people do not act naturally or rationally in the virtual world. This creates the problem of having NPCs that act very rigid and stiff in order to supply a sense of order and direction in the game.

The advantages that Deep Blue had over human world chess champions illustrates that an adequate AI opponent has been produced. However as has been mentioned earlier, this is mostly due to the processing power and cheating modes that AI can do that humans cannot. Many game

developers believe the idea is to make games that can match the player's ability by altering tactics and strategy. This is in contrast to the idea of improving the ability of opponents, which is not just raising the standard difficulty level feature, which is not all that uncommon in many games. However I believe that just altering tactics is not the solution to the problem. Games are meant to be played on many occasions and for long periods of time. When a person who plays a game on an average of 20-30 hours a week plays against a NPC that just alters its tactics, there will be repetition in the responses of the characters. This is not a display of intelligence, but rather a similarity to previous game strategies of altering tactics.

Programmers must keep in mind the needs of the user. They want an original, challenging, and yet an opponent that can be defeated. The game becomes dull and less entertaining when a user can predict accurately exactly what the NPC is going to do at every situation. This is even evident in many recent game releases that are marketed as containing very advanced AI. A game such as *Return to Castle Wolfenstein* (RTCW), which was produced by Activision was an example of a game that just alters its tactics. Alex Champandard [2] reviewed the AI in the game, and found it to be "very good, though there's obviously room for improvement. First and foremost, a better handling of the characters perception is in order." The author is commenting on the idea that the NPCs only react to sounds and movements of the user, and not other NPCs.

The main problem with the game is the fact that NPCs reactions are randomized and at times are seen on numerous occasions. An example can be seen in one type of NPC in *RTCW* in which they have "one behavior that causes them to roll on the floor to take cover, or to position themselves for a shot. On more than one occasion, I've noticed them roll straight into the wall" [2]. This illustrates the not only the NPC exhibiting one behavior, but also problems in the

programming itself. There are so many aspects in game programming to cover that it is easy to overlook a particular event, which would be easily identified by the user. The randomness of reactions by NPCs can make the game dull at times. To overcome this problem, the developers have varied unit placement, by either using a probabilistic knowledge-based placement, or by manually defining each position that a unit can be placed in. Also the NPCs reactions change depending on the tactics used by the user.

Recommendations: (Top)

This section differs in what it presents from the previous section because it gives recommendations to solve the problem. It is the author's point of view on how to solve the problem effectively and what ideas should be looked at for future study. Beyond on how to just solve the problem, this section includes a discussion on different types of simulation projects that have used GAs.

This paper has presented numerous amounts of information on how GAs are not well suited for NPC intelligence in more advanced games. However A-life has provided proof that GAs can be successful in certain aspects of intelligence in NPCs. They have been shown to create intelligent characters in a few popular simulation games. These characters were able to learn and change as the game progressed. They also created life-like adaptations to their human counterparts, adaptations that NPCs were previously unable to attain. These adaptations were similar to learning a player's style or ability and learning to defend or work with that style.

GAs worked well in these A-life games and simulations because there are not a lot of strategies and rules to encode. Life often proves itself to be unpredictable and thus a simulation of such would be even less predictable. This creates a sense of a virtual world void of rules, and one in which there are few strategies to deal with. I believe that A-life is the driving force for

GAs. This field allows the most flexibility in rules and strategies and therefore requires lower amounts of encoding. This is beneficial for the game designer who is generally pressed for time and money. In the game design world when deadlines get pushed back, the consumer finds other sources of entertainment and can make one's delayed product less successful upon its release. Also completing a game design under budget is always a positive step towards assurance of working on future projects. The underlying idea is that when looking to create fully interactive and intelligent characters, simulation games are the prime candidate. GAs have been shown to work well and adequately allow characters to handle every situation encountered in a game.

Although most designers have given up attempting to use GAs for NPCs, the simulation games have given hope. Simulations in general have been extremely successful using GAs. The walking robot project is an excellent example. The plan was to use a GA to design a walking robot that would be able to walk over rough terrain. A GA was used to see if it was able to possibly come up with a design that a human would not be able to. The original design was very awkward looking and did not seem to be an appropriate design for the project. However when the design was implemented it worked perfectly. It was a design that no human could have possibly come up with. This illustrates the power of GAs to produce simulations that are both powerful and can be fully integrated. This is important due to the fact that current methods for simulation were unable to match the ability of a GA.

The training and education simulation games have started to become more widely used due to their abilities. These simulations can be used to shape public policy. People are trained in for situations on what to do, if that particular situation was to occur. When the simulations are intended for peacekeeping, healthcare, and even hazardous chemical cleanup, the illustration of the possibilities of simulations are exhibited. The idea is that people are willing to play games

repeatedly and for many hours at a time. They almost have an addictive sense about them. People can have the sense of wanting to play, almost like a craving. The simulations are not intended for entertainment, but when developed in the same sense as an entertaining type game, they can have the same effect.

This craving feeds the simulation training game. If a designer can have a game that people are willing to sit and play for many hours, while at the same time secretly, or at times openly, being used to train people to perform their jobs more effectively. People would instinctively react to the situations that they encounter, and would be more prepared for those situations when simulation training is used prior to that event. The training simulations have been proven to be effective for the military, healthcare systems using a program called *Simhealth* developed by Thinking Tools Inc., and educational administrators [34]. The military has been extremely successful using simulations. The first simulations used were those for pilots. They proved to be a good teaching arena prior to actually setting foot into the plane. It also was more cost effective due the decrease in risk of damage to the plane and people when an inexperienced pilot was being trained. Simulations also have been shown to allow soldiers to process information on a screen better than those who did not have experience with the games. Also studies have shown that people in the military who have played video games, and more importantly been trained with simulations, have been shown to do better on performance tests than those who have not played or been trained using simulations [3].

The critics of these simulations have often discussed the fact that simulations sacrifice some of their realness in order to make their entertainment value higher. Also mentioned was the fact that the technology that the simulations are built on are not equivalent to the technology used during the real event. The latter criticism has been, and is currently being solved as more

complex and sophisticated technologies are used in game development. Processor speed has increased on desktop PC's, as well as the speed of much of the other hardware used in these PC's.

GAs I believe can be used to make the simulations as accurate as possible and still allow for a sense of entertainment. This entertainment is what makes using simulations so ideal. When the simulation can be viewed as both entertaining and educational simultaneously, then it becomes a valuable commodity. The important aspect to consider when building a simulation is to allow the user to be able to impact and alter the ending. The characters in the simulation must also be able to adapt to what the user attempts as well. Negative as well as positive consequences of actions must also be present in order for the simulation to be effective [34].

To work effectively in a simulation the GA must allow for a large population. This is to ensure random results and greater unpredictability. Using a fsm would not be effective for a simulation. Over time and many successive times the simulation is played, the solutions would repeat themselves and would not be as effective. If repetition occurs then the user forms a general strategy. For instance when a user is playing *Virtual U* [29], where the user learns how to run and operate a university, which is especially useful for employees of a university looking to someday move up to a position of more responsibility. If the user enacts a particular rule or act, the faculty/staff/student body will not always act the same way. Each of these sets of people, faculty/staff/student would not all have the same reaction at every university. So therefore if the AI in the simulation always uses repetition, as having the same response to each reaction of the user, then the simulation is not very useful. To be an accurate simulation the game must allow for completely random solutions to each user inquiry.

The NPCs in these simulations make the game the most believable. It is believable when the characters in the game act naturally and consistent with their roles. It is important to consider how GAs allow for believable NPCs in simulations. GAs can find solutions and answers to responses brought by the user. However, GAs I think should be used in conjunction with another type of AI. For example, I would use a GA to evolve the smartest creatures or objective, depending on the type of simulation, from a population. This is different from direct intelligence, as it is a method for making smarter creatures, not for actually controlling how the NPC "thinks".

Also to be considered when creating an effective simulation, it should be winnable, or at least the user should be able to accomplish all of the goals at the outset of the game. To achieve this the fitness function should be written so that the evaluation of each solution is not as stringent. This would allow for less difficult solutions to survive. The user should not always be able to win or succeed, and especially not easily. However to be an effective educational tool, the user should accomplish the goal. If the fitness function is too harsh, then only the very difficult solutions will survive, and if the user fails to accomplish the goal repeatedly, they would become frustrated. Hence this makes the simulation a poor game, and an even worse educational tool. There is a very thin line between too easy and making the solutions too hard. It would require a good amount of usability testing and learning experiments.

Another issue that makes GAs a good fit for simulations was due to the sluggish nature of GAs. So that inhibits on its ability to control the "thinking" process of the NPC. They are slow because it generally takes many generations to accumulate an acceptable amount of solutions to be a worthwhile character in a simulation. The GA often converges extremely slowly, and playing 1000 to 100000 times at a particular game to get the AI trained is not an option. It would take a long time to come up with a smart enough NPC to be a challenge. So there really needs to

be some other method of letting this happen. These other methods have been done by way of using cheating and other methods that are not considered learning. Getting the computer to play 10000 times against itself would be a lot better solution to the problem. In other words, it is not worth the effort of making the AI adaptable in many types of games, because the training will mean little to nothing. Another thing that makes the process slow is that the GA often makes few, if any good mutants. The computer in these cases will be extremely easy to beat, but the difficulty will vary a lot on this. If the NPCs are easy to beat, how can they be useful for simulations?

They can be useful because when a simulation is being used for training purposes, generally it will be used many times. Each trainee will use the program several times for many hours. When one figures in the amount of trainees that would use a particular simulation, when used for large companies or organizations, then the number of games that the AI has to learn is quite high. It would not take long for the AI to be well trained and to be able to handle extremely intelligent NPCs.

It would be easy to argue that for a simulation to be effective it should not need to be trained. However I believe that given the proper amount of times it would take to properly train and for the AI to be educated, using a GA, it would be more effective method at creating intelligent NPCs in simulations than any other way. Also the simulation should evolve as strategies in the field that it is designed for change. New trainees each year will have new techniques and strategies for dealing for certain problems in their field. This evolving strategy is what would make GAs ideal to use with simulations. As each new strategy is used against the simulation, the AI can learn these strategies and combat them over time so that the same strategy does not always work.

This is what separates a game from an educational simulation. In games generally there are few and many times only one user for each game. However when a simulation, such as *Simhealth* and *Virtual U* are used, there tends to many more users for each. Each simulation can be used many times by every person that is desired to be trained. All this can be achieved by using the same simulation that uses NPCs with specialized abilities in that particular field. One of these abilities would be to strategically utilize the resources they are given [34].

These abilities are best illustrated in the gaming industry by the traveling salesman problem. This problem illustrates the need not for the best fit, but rather the winning fit. This winning fit may not be the best possible solution, however it effectively solves the problem. It also is more cost effective than a best fit would be. The bottom line when designing a game always comes down to staying under budget and meeting the deadline. The winning fit allows for both of these objectives to be met.

The biggest problem that has arisen and is unsolved at this time is the lack of user differentiation. This is the problem that games, and GAs would have in determining the difference between two different users. In an entertainment game this is not such a problem, due to the fact that only a limited number of people will probably play it. When a large company or government organization uses a training simulation for its new employees, then there would probably be hundreds if not thousands of people using the same program. As mentioned it would take the GA too long to determine the difference between each user. A possible remedy for this is to create user identification or some type of login. This would allow the program to know which user is playing and have an accurate strategy available for that person.

While this solution solves a short-term goal, it is filled with problems. The main problem is that a login demonstrates the programs inability to understand human factors. The main factor

being demonstrated here is that at times humans are not consistent. Ideas and thoughts in people are changing constantly. The use of a login makes the assumption that each particular user will not change strategies. This is not the case though. The program would be of little use if it only used the same strategy for each user. This is similar to the current situation in games. Using a login also nullifies the use of a GA because it does not allow the program to adapt and change with the user. GAs have been shown to converge slowly so each person would have to use the simulation a considerable amount of time.

If the training is complex, then the slow convergence of GAs is allowable. However if training is for a simple problem that the user probably would not need to use the simulation for very long, then GAs would take too long to accurately find random and intelligent solutions to the problems that are occurring in the simulation.

The technology is not available yet to allow a simulation to change and adapt to the user quickly and instantaneously. The long-term research goal would be to create some sort of learning agent that would allow the AI to instantly learn the best solutions and strategies to counter what the human user is doing. This instantaneous learning would unleash the vast potential in simulations for training. The simulations would be able to develop scenarios and solutions that the human programmers were not able to think of. This would allow the people being trained to be prepared for any possible situation that could occur, and in the long run produce employees that were more effective in their jobs.

Conclusion: (Top)

Gaming can be seen as a form of entertainment and as an educational tool at the same time. In the business world, training in the form of games can be used for new employees. To have a fully effective game the characters involved need to be intelligent, and able to converse

adequately with the human user. The level that is adequate depends on who is designing the game and what particular abilities in the NPC they want.

The type of gaming that is described here is the simulation genre. This is the idea where the game imitates what occurs in real life. They can allow for effective training and be cost effective. To create NPCs that will allow for this effectiveness it is necessary to use GAs. They will allow for the simulation to change as the user's strategies and abilities change. As simulations are becoming more commonplace in the business world to adequately train employees, the simulations need to keep pace with the demands of the organizations requiring them. To this end the companies developing these simulations should consider using GAs to create effective and intelligent NPCs as has been illustrated.

The future goal of research in intelligent NPCs is to create opponents for the user that can be downloaded off the Internet that are experienced and can be valuable teachers for a particular situation. These would be valuable characters that can change and evolve as it acquires more experience. These traits would seem to lend themselves attainable by GAs. These characters would be useful in not only simulations, but also many other types of games such as first person shooters, role-playing games, and even massively multiplayer online games. Video game and simulation development companies need to visualize their own reflections and contemplate what exactly the user wants. It is becoming increasingly a constant request to have intelligent NPCs that both challenge and stimulate the user.

This paper has attempted to demonstrate the fact that current methods of designing games are not ideal. GAs have been have not successfully accomplished the goal of more intelligent characters in entertainment games. Their main achievement has been minor details, such as target systems, and attack formations. Their greatest impact has been on the simulation game.

GAs have been successful in creating realistic and fully functional games that contain characters that evolve and can adapt to the user. *Black and White, Virtual Petz,* and *Creatures*, are all examples of entertainment simulations that GAs have worked well on.

The idea is that GAs can be applied to training simulations to create fully interactive programs that can account for any type of situation and be able to accurately depict the real world. They are able to handle a larger solution space and therefore more accurately create random and intelligent reactions and responses to the human opponent. This is useful for training due to the nature of training itself. By this I mean that people receive a lot of information very quickly when being trained, and therefore there is a lot of situations and instances that can occur in a simulation. The GA would be able to handle all these problems and find accurate solutions for each.

Using simulations as training has been shown to be a very effective method for teaching employees, as well as students. Universities, the military, and many other large companies have begun to use training simulations and all have found them to be useful. These organizations are not getting the full potential that simulations can provide. If GAs are used then the simulations can handle more problems and more accurately simulate the real world. The problem is that GAs can not differentiate between different users, and therefore take a long time to adapt to each users strategy. At times this can be inefficient, and therefore impractical to use for training. The ideal situation is to have AI that can understand different strategies, adjust to them, and counteract those strategies instantaneously.

Current technology is not yet available to create this type of AI however. Future research could lead to the development of AI that has these abilities mentioned, and thus create an ultimate training simulation. This would be a tremendous tool that could produce the best

employees and students possible. These people would be able to learn more efficiently and retain more of the information that they were given. Since many people are willing to play games, and enjoy them, employee satisfaction at the beginning of a job would be higher. This would lead to better production and output from the employees.

I believe that the future goal of simulations should be to be implemented into grade school education. Children adapt easily to games and can play them for many hours at a time. Studies have shown that children who grow up playing games possess abilities that other children do not have. These abilities are valuable when the child reaches adult age and begins to work. These abilities are better information retention, better spatial skills, and better inductive reasoning skills. Employers look for these skills, and if simulations are used in grade school, then these abilities have been honed for many years.

There must be a careful balance between too much violence and sex in games, and more educational driven simulation style gaming present. The idea of extensive violent game playing can lead to violent people has been studied, and while there are not concrete connections between the two, limiting the amount of violence in a simulation should be taken into account. This is especially true when designing simulations for children.

I believe that this type of AI that can understand user strategies quickly is within reach. It will unleash a type of simulation that could supplant current training methods. In the long run this could save companies money. While the upfront costs will be high, which include buying the software and implementing it, there will be lower costs with requiring staff to train people. In grade schools teachers would have a tool that would fully enhance all of the concepts that they teach. Students could not only learn theories but also be able to implement them by themselves with the use of a simulation. This would create people that are more adept at handling different

situations. People trained with simulations would be more likely to handle more problems effectively as well as finding creative new solutions to problems once thought of as impossible.

References: (Top)

1. Cass, Stephen. "Mind Games." 27 Nov. 2002. IEEE Spectrum. 11 Apr. 2003 <http://www.spectrum.ieee.org/WEBONLY/publicfeature/dec02/mind.html>.

2. Champandard, Alex. "Return to Castle Wolfenstein: AI Analysis." 7 May 2002. AI-Depot. 23 May 2003 <http://ai-depot.com/GameAI/Wolfenstein.html>.

3. Chang, Alicia. "In Study, Video Games Hone Skills." Buffalo News. 29 May 2003: Section A, Page 8.

4. Chughtai, Meliha. Determining Economic Equilibria using Genetic Algorithms. London: Imperial College, 1995.

5. "Computer Games getting Smarter." 24 Mar. 2001. BBC News. 28 Mar. 2003 < http://news.bbc.co.uk/2/hi/entertainment/1237848.stm>.

6. Crawford, Chris. "Computer Games are Dead." 1996. Interactive Entertainment Design. 15 April 2003 < http://www.erasmatrazz.com/library.html>.

7. "Creatures: Official Site." 18 Oct. 2002. Creature Labs. 18 April 2003 <http://www.creaturelabs.com/>.

8. Deutsch, Claudia H. "TECHNOLOGY; Some Colleges Take Games Seriously." NY Times. 2 Apr. 2002: Section C , Page 4 , Column 1.

9. Doyle, Patrick. "RE: Research Questions." Email to the author. 16 Apr. 2003.

10. Doyle, Patrick. "Virtual Intelligence from Artificial Reality: Building Stupid Agents in Smart Environments." Stanford University. 10 Oct. 2001.

11. Fannon, Sean Patrick. "Where We Should Be Going With Online RPGs." 19 Sep. 1997. Gamasutra. 28 May 2003 <http://www.gamasutra.com/features/19970912/online_r.htm>.

12. Gluck, Caroline. "South Korea's gaming addicts." 22 Nov. 2002. BBCNews. 15 Mar. 2003 < http://news.bbc.co.uk/2/hi/asia-pacific/2499957.stm>.

13. Holland, John H. Hidden Order. Reading, Mass.: Perseus Books, 1995.

14. Horrigan, John B. "Pew Internet Project Data Memo." 18 May 2003. Pew Internet and American Life Project. 28 Jun. 2003

< http://www.pewinternet.org/reports/pdfs/PIP_Broadband_adoption.pdf>.

15. James, Greg. "RE: Research Questions." Email to the author. 3 May 2003.

16. Jaramillo, Jorge. Genetic Algorithms for Location Problems. (Thesis) Buffalo, NY: State University of New York, 1998.

17. Kantrowitz, Mark. "What is Fuzzy Logic?" 15 Apr. 1993. Carnegie Mellon University. 13Mar. 2003 < http://www-2.cs.cmu.edu/Groups/AI/html/faqs/ai/fuzzy/part1/faq-doc-2.html>.

18. Kurzweil, Raymond. The Age of Intelligent Machines. Cambridge, Mass: MIT Press, 1992.

19. Laird, John E. "Research in Human-level AI using Computer Games." John Laird's Artificial Intelligence & Computer Games Research University of Michigan. 13 Mar. 2003 <http://ai.eecs.umich.edu/people/laird/gamesresearch.html>.

20. Lee, Wei-Ang & Ogasawara, Naho & Pen Kang Yi, George. "GPOTHELLO MACHINE LEARNING TEAM." Columbia University. 17 Apr. 2003 < http://www1.cs.columbia.edu/~evs/ml/OthelloStudProj/ Yi%20George/index.html>.

21. Levine, Robert. "The Sims Online." Wired Nov. 2002: 176 – 179.

22. Lode, Clemens. SCC 1.02. 3 Apr. 2003. Claw Software. 5 May 2003 < http://www.clawsoftware.de/us/starcalc.htm>.

23. "Methods" MaBOS. 28 Mar. 2003 <http://www.mabos.com/mab-opt.htm>.

24. "Media Center." Interactive Digital Software Association. 17 Apr. 2003 <http://www.idsa.com/pressroom.html>.

25. Motive Communications. "Home Networking: The Challenge and Opportunity for Broadband Service Providers." 2003 < http://library.motive.com/upload/public/WP_HomeNetworking_87.pdf>.

26. Obitko, Marek. "Crossover and Mutation." 1998. Czech Technical University in Prague. 28 Jun. 2003 < http://cs.felk.cvut.cz/~xobitko/ga/cromu.html>.

27. Palmer, Nick. "Machine Learning in Game Development." AI-Depot. 28 Mar. 2003 <http://ai-depot.com/GameAI/Learning.html>.

28. Palu, Sione. "The use of Java in Machine Learning." Gamelan. 22 Apr. 2003 <http://www.developer.com/java/other/article.php/10936_1559871_1>.

29. “Press Release.” 27 Jun. 2000. Virtual U. 30 May 2003 < http://www.virtual-u.org/releases.html>.

30. Quake Terminis. “What are Bots?” 2003 < http://www.quaketerminus.com/bots.htm>.

31. Random house American Dictionary. New York, NY: 1972 p633.

32. Ratan, Suneel. “Sims Flop Dogs Game Developers.” Wired Jun. 2003: 86 - 88.

33. Rouse, Richard. Game Design, Theory, and Practice. Wordware Publishing, 2001.

34. Sawyer, Ben. “Enhancing Simulations, Models and Their Impact Using Interactive Game Design and Development Practices and Technology.” Digitalmill, Inc. 2003.

35. Schaeffer, Jonathon. “A Gamut of Games.” University of Alberta. 26 Aug. 2001.

36. Schreiner, Tim. “Artificial Intelligence in Game Design.” 15 Jul. 2002. AI-Depot. 21 May 2003 <http://ai-depot.com/GameAI/Design.html>.

37. Selfridge, Oliver. “The Gardens of Learning: A Vision for AI.” AI Magazine. Summer 1993: Vol. 14 Num. 2, 36-48.

38. “Serious Games White Paper Released.” 28 Feb. 2003. DigitalMill. 18 Apr. 2003 < http://www.dmill.com/pressmedia/dmpr022802.html>.

39. Stern, Andrew. "Virtual Petz: A Hybrid Approach to Creating Autonomous, Lifelike Dogz and Catz." Proceedings of the Second Intl. Conference on Autonomous Agents. May 1998.

40. Vance, Ryan. “Online Gaming in South Korea.” 22 Nov. 2002. TechTV. 17 Jun 2003 < http://www.techtv.com/extendedplay/reviews/story/0,24330,3408689,00.html>.

41. Vardy, Andrew. “RE: Research Questions.” Email to the author. 12 Apr. 2003.

42. “Video Games Getting Tougher.” 12 Sep. 2001. BBC News. 28 Mar. 2003 <http://news.bbc.co.uk/2/hi/science/nature/1538092.stm>.

43. Wall, Matthew. “Programming Genetic Algorithms.” 23 Mar. 1996. University of Pittsburgh. 19 May 2003. <http://www2.sis.pitt.edu/~shirtle/galibnotes.html>.

44. West, Neil. “The Way Games Ought to be.” Next Generation Magazine Jun. 1998: 104-5.

45. Woodcock, Steven. “Game AI: State of the Industry.” Game Developer Magazine. 1 Nov. 2000. < http://www.gamasutra.com/features/ 20001101/woodcock_02.htm>.

46. Yap, Peter. "Research Interests." 30 Jan. 2003. University at Alberta.
11 Apr. 2003 <http://www.cs.ualberta.ca/people/grad/petery.html>.

V. Appendix

Fitness Score Sample Code:

```
// Population has been sorted according to fitness
int iPopMin = member[POPULATION_SIZE - 1].fitness;
int iPopMax = member[0].fitness;
// Calculate the scaling factors
double dMean = 0.0;
for (int i = 0; i < POPULATION_SIZE; i++)
dMean += member[i].fitness;
dMean = dMean / POPULATION_SIZE;
double delta = 0.0;
double a = 0.0;
double b = 0.0;
double scale = 0.0;
if ( iPopMin >
((SCALE_FACTOR * dMean - iPopMax) / (SCALE_FACTOR - 1.0))
)
{
delta = iPopMax - dMean;
a = (SCALE_FACTOR - 1.0) * dMean / delta;
b = dMean * (iPopMax - SCALE_FACTOR * dMean) / delta;
}
else
{
delta = dMean - iPopMin;
a = dMean / delta;
b = -1 * iPopMin * dMean / delta;
}
// Scale the population (scaling of the population)
for (i = 0; i < POPULATION_SIZE; i++)
member[i].fitness = member[i].fitness * a + b; (this is the evaluation function) [14]
```