Evolution in an open world RPG

Abstract

In this review we are going to look at the history of table top role playing games (RPG), specifically Dungeons & Dragons and computer RPGs, how their artificial intelligence (AI) has largely remained primitive, how other computer game genres have improved their AI components and what we can learn from that. We are looking for a way to add a more dynamic element to a genre that was largely lost from the transition from its table top origins to the computer, and to take advantage of that fact that the game is running on a computer. We intend to achieve this through creating a simple model of evolution for the open world RPG that will allow for long term environmental consequences to come to fruition. In a table top RPG this process could occur because the imagination of the players is a key component of the game, this is however very hard to move into a computerised form. This is also intended to be a Darwinian evolution process, so the optional solution is what survives. The main goal is to see if we can offer something that is similar to what could be offered by playing a table top, but is still something that could only be done by a computer.

Table of Contents

[Chapter 1: Introduction 3](#_Toc292704993)

[Chapter 2: Role playing games 5](#_Toc292704994)

[2.1 The history of table top RPG. 5](#_Toc292704995)

[2.2 The general features of a table top RPG. 5](#_Toc292704996)

[2.3 The history of computerised RPGs. 6](#_Toc292704997)

[2.4 The general features of a computerised RPG. 7](#_Toc292704998)

[2.4.1 Features of a standard computerised RPG. 8](#_Toc292704999)

[2.4.2 Features of an open world (aka sandbox) RPG. 8](#_Toc292705000)

[2.5 Primitive AI 9](#_Toc292705001)

[Chapter 3: Adaptive behaviours in games 10](#_Toc292705002)

[3.1 Adaptive behaviours. 10](#_Toc292705003)

[3.1.1 Online and offline learning 10](#_Toc292705004)

[3.1.2 Supervised and unsupervised 10](#_Toc292705005)

[3.2 Scripting 10](#_Toc292705006)

[3.3 Dynamic scripting 11](#_Toc292705007)

[3.4 Finite State Machine 13](#_Toc292705008)

[3.5 Neural Networks 14](#_Toc292705009)

[3.6 Genetic Algorithms 15](#_Toc292705010)

[3.7 The Belief-Desire-Intention model 16](#_Toc292705011)

[Chapter 4: Applying evolution to role playing games 18](#_Toc292705012)

[4.1 The need for scripting in any adaptive AI. 18](#_Toc292705013)

[4.2 Discussion on using dynamic scripts. 18](#_Toc292705014)

[4.3 Discussion on Genetic Algorithms 19](#_Toc292705015)

[4.4 Discussion on Belief-Desire-Intention model 19](#_Toc292705016)

[4.5 Levels of realism in simulation games 20](#_Toc292705017)

[Chapter 5: Proposal 21](#_Toc292705018)

[5.1 The system 21](#_Toc292705019)

[5.1.1 The use of existing RPG stats and diet. 22](#_Toc292705020)

[5.1.2 The use of dynamic scripts. 22](#_Toc292705021)

[5.1.3 The finite state machine. 23](#_Toc292705022)

[5.2 An example 23](#_Toc292705023)

[5.3 Limitations 24](#_Toc292705024)

[References 25](#_Toc292705025)

Chapter 1: Introduction

In 1974 a new style of table top role playing game (RPG) was released called Dungeons & Dragons (D&D), this was unlike many other games and RPGs at the time as now instead of a game that was essentially a set of rules and all skill based, D&D’s most important factor was the players imagination. D&D shifted the focus from a typically tactically based set of rules, to thinking more as if their in-game character and the world they were in was real and as such they had to act their part. The way the inhabitants of the fantasy world behaved and changed over time was mostly controlled by one of the players, the Dungeon Master (DM) and only needed to rely on rules in certain situations.

Since RPGs were first computerised in 1982 for the Atari 2600, the artificial intelligence (AI) presented was very primitive to fit within the limited game cartridge size and computational cycles. However since then, all aspects of a consumer based computer or gaming console have improved dramatically and along with that both the graphics and sound quality have improved. Unfortunately the AI for games and especially RPGs have remained quite primitive. As the way the AI worked in Dungeons & Dragons was primarily based on how the Dungeon Master controlled them and only for simple things were rules used. This presented a major problem of trying to get human level intelligence AI for a game made for a limited system.

Other game genres have however improved their AI by varying amounts. Genres such as the first-person-shooter lead themselves to having more complex AIs that can deal with the terrain they are given and take cover and work in teams to flank and out manoeuvre other bots or players. This can be seen in Monolith Productions F.E.A.R.[[1](#Jef11)] where the AI uses a combination of GOAP (Goal Oriented Action Planning) [[2](#Jef04)] and a NavMesh of the current world to intelligently move the AI bots around the world. Unfortunately this AI did not make any use of adaptive behaviours and as such players could learn how to manipulate the bots as they would act in a predictable fashion.

The simulation genre where the player is often in control of a world or people often features a much more important focus on the AI. Games such as Maxis’ The Sims series and Lionhead Studios Black & White series have both been very successful and both have a heavy focus on much more complex and realistic AI and in the case Black & White [[3](#Ric02)] the ability to learn. This genre has some overlap with the traditional table top RPG as the world would be simulated by the DM and the inhabitants of that world often behaved in similar ways to that of characters in The Sims or the divine creatures from Black & White. This brings up the possibility combining the work done by the simulation genre in bringing some realism to the RPG genre; however these current systems do not work with long term and long reaching consequences.

After nearly 30 years of development of the computerised RPG, various aspects of the RPG have taken advantage of the computer in giving a uniquely different experience to a table top version. Such as the immersive graphics and sound which while this takes away the players need on their imagination, it is similar to the difference between reading a book and watching a film, they both have their place. The computer also offers a way for players to play over long distances in real time using the internet to share adventures together. However one aspect that has not taken advantage of that computational power is the AI and the environment; with modern computers being vastly more powerful than the Atari 2600, it is now possible to simulate much more complex AI. It is now possible for the computer to offer something that would not be easy for a human DM to offer, complex adaptive AI that can change and adapt to their environment.

This brings up the fact that for more realistic AI that is based on a nature process of evolution to be able to adapt to its environment that there is going to be a need for a way of limiting the changes of the creatures lest the algorithm figures out that simply by exponentially increasing the strength of the creature means it survives better. This situation would be of course game breaking so depending on the algorithm used, there needs to be a limiting factor to the changes that can occur either through hard limits of the game or through a process that fits in with the natural process.

This review will look at the transition of RPGs from table top to their modern computerised form and how the AI has not changed much in the past 30 years. How other genres have taken AI much further along such as first –person-shooters and simulation games and how RPGs can take advantage of work done in those areas for the parts that overlap between genres.

# Chapter 2: Role playing games

While the genre of RPG is very diverse, we are going to focus on the history of the typical modern RPG that has given us the standard class based characters, typical RPG stats and levelling, and quest lines.

## 2.1 The history of table top RPG.

The origin of nearly all modern RPGs can be traced back to Dungeons & Dragons, first released in 1974 by Gary Gygax and Dave Arneson. Dungeons & Dragons deviated from the other RPGs available at the time which were primarily military simulations. These military simulations consisted of the players controlling armies of units and were often short lived campaigns with a simple set of rules in a small world. Dungeons & Dragons however gave the players control of a single unit that was their avatar in the fantasy world and everything else in the world is controlled by a specific played called the Dungeon Master (DM). This shifted the focus from the generally short campaigns played in a small section of a world with a narrow goal, to a more open ended world that is far more dynamic now that it the product of the DM players imagination; such that they controlled the flow of the adventure with it [[4](#Mon03)]. This was an important change as now the game had a user specific goal and it was the DMs role to guild the other players through a world or to complement what they wanted to do and to make it feel like a real place. This means that player actions can have long lasting consequences that may or may not be immediate or obvious.

## 2.2 The general features of a table top RPG.

As Dungeons & Dragons has a storying telling element and is controlled by the imaginations of the players, it does have rules, lore and a certain style for how the game is played. Most games at the high level fall into two categories, either a linear story or an open ended world. Linear stories are generally shorter but this is not always the case, but they are generally the product of the DM telling a story of some kind, and depending on how narrow the story is, the players actions will affect how the story progresses and the actual ending. In an open world setting, the players generally go find an adventure and when an appropriate time presents itself, the DM will add in a story that may just be a short misadventure or perhaps a long drawn out journey and many of these can happen within a given game. Also depending on the open world game, there is no definite end unless one in agreed upon as even death in most of these games is temporary and reversible. Whilst in a short linear story, the game ends when the players reach the end of that story, although a linear story can still have subplots.

Another major feature of these games is the non-player-characters, often referred to as NPCs. These NPCs often have knowledge about the situation or turmoil in the game world and depending on the personality of these NPCs, the players may have to extract the information out of them. As the NPCs are controlled by the DM, the players can interact with them as they would a human and form friendships, rivalries or become enemies. There are no exact rules for how a DM controls these NPCs, only that they should appear reasonably realistic and have motivations for any particular actions they perform. This fact that such an important factor in a typical Dungeons & Dragons game is primarily controlled by the imagination of a human player doesn’t translate well into a computerised form.

Players are generally able to perform whatever actions they want and the DM in turn generally should react to these actions by having the game world react accordingly. Such as, should a particular player be playing an evil character they may poison a town’s water supply, which to us as humans we can predict the potential outcomes of such an action. However to a computer unless this action was accounted for by the game’s developers it would probably not even be an action that could be performed, let alone having any meaningful outcome. This is another aspect that does not translate well into a computerised form, the ability to perform any physical action that affects the player or the game world at large.

Another important factor when playing these games compared to a computerised form, is the fact that each player interacts with the world in turn. Generally only if there is a particularly lively event, each player tells the DM (and the other players) what actions they are performing, to which the DM then has to calculate the outcome of these actions and respond back to the players with the consequences.

## 2.3 The history of computerised RPGs.

The first commercially available computerised RPG was Starpath’s Dragonstomper for the Atari 2600 released in 1982. This game had many similarities of a Dungeons & Dragons game, where the player controls a single unit in a world and must improve their characters stats and equipment in order to advance through the story. It has many features of a table top game such as similar numerical game rules, NPCs, a story and a large game world for the player to explore.

## 2.4 The general features of a computerised RPG.

As by the nature of a computer, many if not all of the numerically based rules have been successfully translated into a computer form for the use in RPGs. And as these rules were originally designed to be calculated by human players within the context of playing a game which means they cannot be too complicated or laborious to calculate, they can be calculated quite easily by a computer at a much higher rate than by a human. This has an important consequence that unlike a table top RPG that is played in turns and can at times take quite a while to complete a combat encounter, that the games could be played in real-time, where the computer takes advantage of different stats of different creatures to calculate how often they can attack and how often the players can attack. It is also important to note that not all computerised RPGs are played real-time, there are still many out there that choose to be played in a turn-based fashion to give the players’ time to think about what they are going to do, however this is by design rather than a limitation of the medium.

These computerised RPGs also still feature NPCs, however they are generally much more limited that what one would expect from a table top RPG. They are generally there just for selling items to the player or for providing some simple and often inane dialog, usually in a town that could be populated with many NPCs; only a select few will be of any importance. This is mostly due to the limitations of the medium, where getting an NPC to act like a human beyond repeating scripted dialog and walking around scripted paths can add a lot of extra effort. However some games have implanted a system typically referred to as Living World, where the NPCs have lives, where the go to sleep at night, wake up and go to their jobs, mingle with other NPCs and talk about daily life and then go back to their homes at night. While much of this is still scripted and the NPCs don’t actually have needs, it can still add a nice level of realism to a village or city, rather than a town filled with people who are stuck in one spot and who only repeat the same few lines of dialog.

Often the only way the world will react to the player outside of combat or dialog is through scripted events, this is again due to limitations of the medium. Where if the player started a fire in the corner of a town in any given RPG; most of the time nothing would happen, unless the developers had put in checks for such an event.

However there have been aspects of these games that have taken advantage of the computer, such as the graphics, sound and networking. These have allowed for the games to be quite immersive and while some would argue that this is simply taking away the use of your imagination, it is similar to the difference between reading a book and watching a film, they each have their own place. The other aspect networking has also made it possible that many people can play the same RPG together over the internet which again would previously be either very tricky to do using phones or video streaming and would also be limited to a few people. This has brought about a particularly popular sub-genre the MMORPG (Massively Multiplayer Online Role Playing Game), these are typically played by thousands of players at once in a very large game world.

Despite these advancements in computerised RPGs, the AI side has not changed much in these years. If you look at a modern computerised RPG you will still find that many creatures are in one of a few states, such as ‘Idle’, ‘Attacking’ or ,’Fleeing’. NPCs have also not changed much in these years except in the dialog side, many games do now offer complex dialog trees and the player’s character’s abilities can have an effect on what the NPC may be willing to tell the player. These can range from the character’s bluff skill, to their ability to intimidate which could both offer a different set of dialog from the NPCs; these however are still scripted in but they are an improvement.

### 2.4.1 Features of a standard computerised RPG.

While in a table top RPG the difference between a linear story and open world RPG is typically not so clear cut, within the context of a computerised RPG the distinction is clearer, although not in all cases.

What is considered the standard computerised RPG is for the most part a linear story RPG that is primarily a story telling device. The player generally has very little ability to stray from the normal path and what the player has to do next; or rather what they can do next is made reasonably clear. This does however mean that the stories can be much more involved as the writer’s do not have to take into account every possible action the players may make ahead of time, unlike a DM who can react to players as they perform the actions.

Some of the major games of note in this series are the BioWare’s games Baldur’s Gate game series and Dragon Age: Origins which are both known for their story telling ability.

### 2.4.2 Features of an open world (aka sandbox) RPG.

Open world RPGs in a computerised form are for the most part like their table top counterpart, the player chooses a character they wish to play as and set out into a world. Depending on the game they may have a clear main story with a single or multiple endings, or they may have several different stories that all result in a different endings. These are typically defined by the fact that the player has much more freedom to go about what they do and can go where-ever they want within the game world for the most part. However because of this freedom they do tend to have issues with giving the player direction and in telling a story. This is why they generally go for a more ‘sandbox’ feel, where the players can just do whatever they feel like within this game world and are able to create their own fun through interacting with the world however they want.

Some games of note from this series are Dwarf Fortress, Minecraft, New World Computing’s Might and Magic game series and Bethesda’s The Elder Scroll game series. Dwarf Fortress and Minecraft both only use randomly generated worlds and for the most part let the players create their own stories out of their experiences in those worlds, which gives the players a lot of freedom in what they do, but for the most part have no provided story.

Might and Magic, and The Elder Scrolls games however both offer worlds where the player is given some background to the game world and then set free in it. They can then choose to follow the main story or they can simple explore the world and find all its secrets.

## 2.5 Primitive AI

In none of these computerised RPGs however has there been much work in furthering the AI that was introduced from some of the early computerised RPGs; however this is not the case with all game genres. Some genres have advanced their AI considerably over the years to create agents that are able to navigate complex terrain, form goals and create opinions based on their experiences with the world.

# Chapter 3: Adaptive behaviours in games

## 3.1 Adaptive behaviours.

Adaptive behaviour is the ability to adapt to changes in game environment or player’s actions at run-time in order to appropriately deal with the new situation; preferably in interesting ways that are not easily predictable [[5](#Pie06)]. These adaptive techniques are often classified as online unsupervised learning, however this does not mean that the basis for these AI’s cannot be seeded using offline supervised techniques beforehand.

### 3.1.1 Online and offline learning

Offline learning, in the context of developing a piece of software, is where the AI is taught during the development stage and kept static once it is released. Online learning is where the AI learns while the program is running after it has been released [[6](#Eth04)]. Online learning does not mean that offline learning cannot be applied during development time so that the AI has a good starting point for when the user first uses the software. The important factor of online learning is the AI’s ability to adapt to changes; however how this is implemented can be computationally expensive.

### 3.1.2 Supervised and unsupervised

Supervised learning is the task of inferring a solution from a set of supervised training data and examples; once the AI has been trained it can then be tested on real world data. Unsupervised learning takes place on real world data [[6](#Eth04)].

Both the online/offline and supervised/unsupervised properties have an important impact on the learning algorithm that is used and how it is used. As in an offline supervised learning environment, the algorithm can take it’s time to find an optimal solution for the current situation, however if there is no online learning part the algorithm will still not be adaptive.

## Scripting

The most commonly used method for adding a way of giving any game a basic level of AI is to hand code exactly how the AI should behave through the use of scripts. A script can be anything from a simple set of rules to complex algorithms running in either native code or in a run-time interpreted language. Scripts generally form the backbone of all AI in directing their exact actions and retrieval of information from the game world and work using a combination of inputs from the world, the current state the script is in and a set of rules for those states [[7](#Rog77)]. Scripts allow for the programmers to give the AI very precise instructions on exactly how, when and where an action should be performed, which depending on the game can result in optimal performance. However if the game is such that an optimal performance by the computer players results in a near unbeatable opponent, then the player is not going to continue playing and then the fact that the AI is perfect is moot. This has the result that the programmers have to artificially limit the abilities of the AI in order to level the playing field to such a level that the AI only keeps track of a certain number of units at a time, can only perform a certain number of actions within a second or how fast they react to events in the world, such as spotting an enemy. However in their simplest form have no memory, no way of adapting over time and are usually quite predictable by the player; however unless the scripts themselves are extremely complex, they are for the most part very computationally efficient.

Scripting has been used to great effect in many games such as BioWare’s RPG Neverwinter Nights (NWN). NWN has proven itself a very competent RPG based on D&D’s 3.5 edition rules and is a good example of scripting, as it provides a feature rich scripting engine that can be used for creating events in the game and for controlling the AI. This is an example of a scripted AI that is complex enough to deal with many different situations and to provide an opponent that will in many cases use its own abilities to good effect. NWN’s AI still has the inherent flaw of any scripted AI that it is however predictable and will not change tactics if what it is currently trying is not working well and that particular creature may actually have the abilities that would be more affective that in given situation.

## Dynamic scripting

The next logical step with scripting is to be able to change what script is being executed at run-time based on the AI performance, this is possible through the use of a technique called dynamic scripting [[5](#Pie06)]. Dynamic scripting provides a way for the AI to learn and adapt to different situations based on its performance, this is achieved through keeping a database of different available rules and each AI agent keeps its own set of weights associated for each rules. These weights are then updated at run-time based on the AI’s performance. At the start of an encounter, these weights can either be all equal or they can be assigned a random value. Depending on the implementation of the system the way in which the weights are used to create the script that is actually used will change, however the simplest way is to simply sort the weights from highest to lowest, although this can lead to issues where if the situation changes quickly the rules used will remain the same for several passes until the lower weight rules overtake the previously highly weighted rules. So another way to use the weights is to have them based on a probability, the weights with the higher values are then executed more often, still leaving the opportunity for the AI to discover a rule that would perform even better for the current situation if its current script is not working as well as it could.

The performance of a script is determined by its fitness function which calculates how well the script performed during the last encounter or other important time frame; this is then used to adjust the weights of the rules that were used to make up the final script. The total amount of weight is always maintained, and each script has a *Wmax* and *Wmin* associated with them which determine the maximum and minimum values that the weight can have. When a rule is rewarded after it performed well in the last encounter, the other rules need to have their weights reduced by the same amount that keeps the total weight in balance. The same is also true when a rule is punished when it did not perform well, and its weight is reduced, the other rules need to have their weights increased. The reason behind increasing the weights of the rules that were not used is that, since the scripts there were used did not perform well, perhaps these other ones will [[8](#Lad08)]. The reward and punished values also have a maximum and minimum.

The weight adjustment formula in its general form:

Where *Pmax* and *Rmax* are the maximum the weights can be adjusted due to a punishment or reward, and where *b* is a break-even value where the weights will remain unchanged if *F* the fitness value equals *b*.

In a given encounter after the script has been created by the system, the overall performance is can be determined. If the script performed well, all the rules that were used have their weights increased and the remaining weights are decreased by the same amount. Again the reasoning is that since the script performed well without those other scripts, chances are those rules are not important [[8](#Lad08)].

Also presented in [[5](#Pie06)] is an example of adding dynamic scripting to NWN, where they have taken the static scripts provided with the game and broken them down into a rulebase with some variation to give each script a certain use condition. They present a mock combat situation of two teams of four combatants, a rouge, wizard, cleric and fight; one team is controlled by the original scripts and the other by their dynamical scripts. While at first the static team proved successful, the dynamic team quickly adapted from their randomly weighted scripts into a set of scripts that proved to be much more effective. This was done to demonstrate that should a human player always play using the same tactics that the dynamic scripts would adapt, putting them at a disadvantage.

Another technique called Reinforcement Learning (RL) behaves in a very similar way to dynamic scripts, in that it also has a set of actions that the AI can perform and each action has a weight assigned to it and is altered based on the perceived performance of using a given action. RL differs mostly in how the actions are evaluated and how the set of possible actions that the AI will then perform next is determined.

## Finite State Machine

Finite state machines (FSM) when in the context of a game AI are a way of separating different groups of actions based on what state the AI is currently in. The finite state machine as a whole is made up of two important elements, the states and the transition. These different states often reflect simple high level actions such as ‘Idle’ or ‘In combat’, these different states often define what script is currently in use. The transitions define the conditions it takes to go from one state to another; this can often be as simple as ‘See enemy’ takes the AI from the ‘Idle’ state to the ‘In combat’ state.

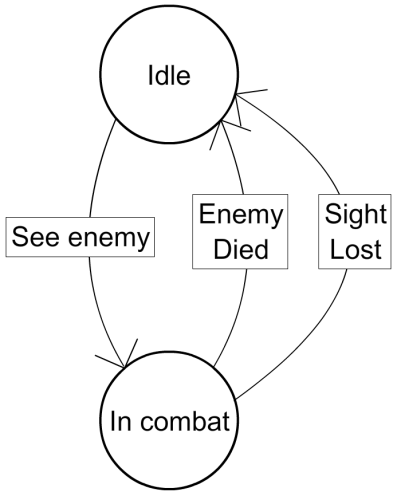


Figure 1: A simple FSM that shows the transitions between two states.

FSMs have been heavily used in games as a simple way to determine which script to run, while not adaptive, more complex FSMs can be designed in such a way that they can function with a basic memory and as such the second time the same action occurs the state can change which script is being used. Because of this, developers have used FSMs because they are simple to use, debug and implement.

One example of a game that has used FSMs is id’s Quake game series [[9](#id)]. They show how the game AI can be represented by a series of different states, all of which are related to combat.

## Neural Networks

Neural networks (NN) attempt to simulate the biological process that occurs with large networks of neurons found in a brain [[10](#Hop82)]. Each of these artificial neurons contains a weighted factor for each input and a condition for firing the neuron’s output and a weight for the value of the neuron’s output. These neurons are often grouped into layers, with one set for accepting the input and collecting the output and a varying number of hidden layers in between. All output from the neurons from one layer are connected to all other neurons in the next layer. These, when combined into a large network can be trained to match input data to training data, and to adjust their weights based on amount of error found between the expected output and the network’s output. There are several different learning algorithms that are used when training a neural network; the simplest and most commonly used is back propagation [[11](#JrA75)].

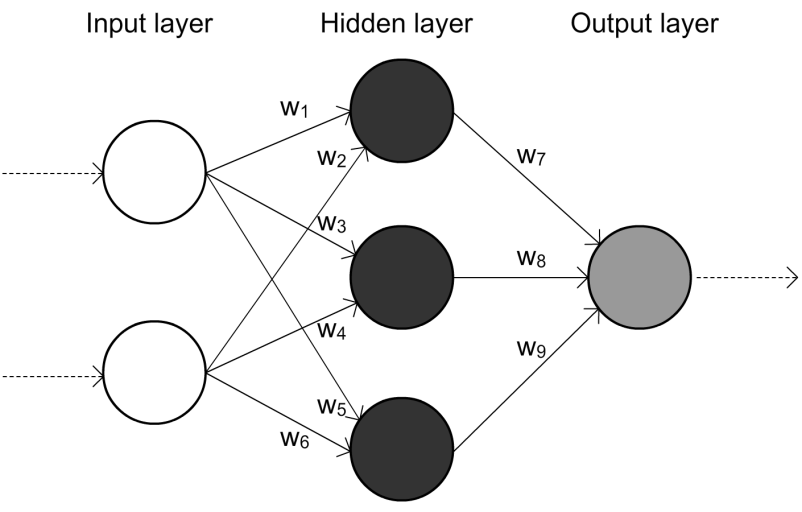


Figure 2: Simple neural network with one hidden layer. Showing how each connection has an associated weight.

While NNs are not often used in games for online learning, it has been successfully used in offline supervised learning. Such as Colin McRae Rally 2.0, the second game in a long series of realistic rally car driving simulators. These games focus on creating a realistic environment for which the cars are to be driven in where everything about the cars and the track is simulated. This presented a tough challenge for the developers of the AI to create something that could navigate these tracks proficiently and as such they turned to using a NN that was trained to drive and handle the roads and then used higher level rules for additional driving skills such as overtaking and recovering from a crash [[12](#Jef01)].

NNs offer a good way of adapting to a situation and learning to deal with complex input and output. However neural networks in the context as working similar to a gene pool where different genes are going to have a varying importance on whether or not a creature survives in a given environment; additionally in the context of a gene pool, neural networks are going to have less variation between different individuals resulting in poor environmental adaptation should something major occur.

## Genetic Algorithms

Genetic algorithms (GA) provide another way to replicate biological processes to search for a solution to a problem. This is achieved by using a population (aka gene pool) of potential solutions, each encoded into a string of data called a chromosome [[13](#Joh92)] [[14](#Dav89)]. These chromosomes can simply be an array of values used to represent different parameters for a function or more complex data. Through natural selection where the selection is based off a fitness function that determines the genetic fitness of a chromosome. In the context of a game, the chromosome could be split up into an array of bytes that are used to control frequency or order of AI agent actions. The performance of these actions is used to determine the fitness of each chromosome. The algorithm then moves to the reproduction phase where based on the fitness of each chromosome, the ones with the highest fitness are chosen to be bread together to form the next generation of chromosomes; the two main operators at work during reproduction are crossover and mutation. The crossover operator at random will choose a crossover point where the data between two chromosomes are swapped. After crossover a mutation operator is applied which has a small chance to corrupt a small amount of data, this usually involves simply swapping a single bit somewhere in the new chromosome. Of course, depending on implementation, the number of crossover and mutation points can be more than one. Due to these crossover and mutation operators, the new child chromosomes have a chance to be both better in some aspects and worse in others. Due to the additional random nature of how the child chromosome come about, the algorithm can find novel solutions to the problem, as an important factor in the genetic algorithm is that it does not need to know why a chromosome is more fit, only that it is.

This random nature of the algorithm is both an advantage and disadvantage, in that one chromosome may result in a novel and interesting solution to the problem and in that same generation you may end up with a very poor performing chromosome. This chance of having a poor performing chromosome is generally what leads to game developers not using GAs as an online adaptive solution, however this also depends on how the GA is used to affect the AI.

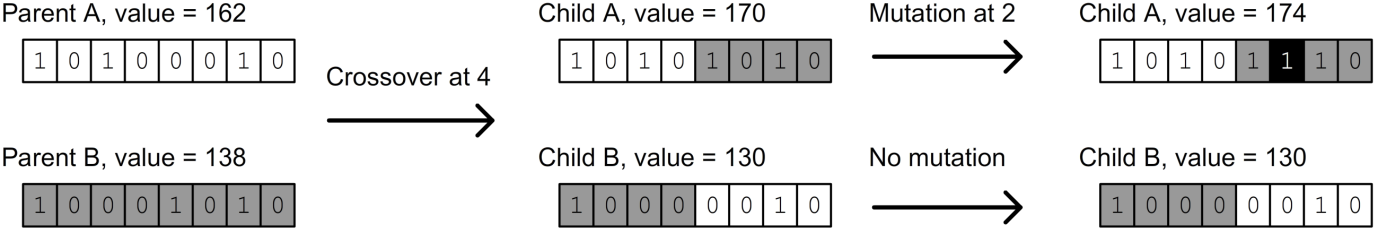


Figure 3: Shows the process of crossover and mutation between two chosen parents and how the values for the child  
 chromosomes come about. This shows a very simplistic data encoding model where the actual value of the  
 chromosome is directly based off the bit stream. This can work in many cases, however as shown the values  
 can fluctuate rather than trending towards a particular value.

A research paper into using supervised offline GAs to train an AI that could deal with tricky situations presented in an RTS that scripted rule based AI’s cannot, showed that it did not take much time for the trained AI to outperform the scripted AIs. These situations include (in the context of Wargus an open source engine of Blizzard’s Warcraft II RTS) the Footman rush, where the player will attack early on with the weakest land based unit in order to gain advantage and Knight rush, an attack comprising of the most powerful land based unit as soon as possible. These two strategies are often used by players to overwhelm enemy AI because their usual strategy is to build an economy and focus on expansion in order to be in a good position later on in the game. This exploit the flaw in the scripted AI that it is assuming how the player is going to play, and if the player uses a strategy outside of that, the AI typically does not perform very well [[15](#Mar)]. Another problem with this exploit is that it can be hard for a developer to come up with a strategy to counter such an attack; the trained GA on the other hand was able to find a solution. This does have the unfortunate limitation that this had to be done offline and not online.

## The Belief-Desire-Intention model

The Belief-Desire-Intention (BDI) model for an agent in a system attempts to model human reasoning [[16](#Bra99)] such that (as the name implies) the agent is able to form an intention based on what it believes and what it desires. Both of what it believes and desires is susceptible to changing over time as the agent is either taught and has a way of self-learning about the world that it is in. As each of the belief, desire and intention components are separate systems, which can each be represented by different data structures that suit it best.

A popular game that has successfully made use of the BDI model is Lionhead studios Black & White. One of the game elements is your divine creature which interacts with the game world and its inhabitants. The creature uses a BDI model based on the work of Michael Bratman and work done by the developers where they added an opinion aspect to the BDI model [[3](#Ric02)]. This allowed you as the player and the creatures own experiences with the world, to learn. Such as learning that eating people was a good or bad thing to do, or that destroying buildings was a good or bad thing to do, it can either determine if the action was good or bad from you rewarding or punishing it; but it can also learn based on how it felt after it did the action. Naturally good creatures tend towards feeling bad at the loss of life and as such will feel bad after eating someone and as such will tend to not do that from then on, unless you the player intervene. This made interaction with the creature much more interesting and as such once you felt that your creature would not do anything that would negatively affect your godly image (be it good or evil), you could let them loose on the world and they could interact with it as they saw fit.

# Chapter 4: Applying evolution to role playing games

How can we apply evolution to RPGs? So far we have not seen much evidence of adaptive behaviours of any kind being applied to RPGs except in with the one case of a research paper into using dynamic scripts for Neverwinter Nights (NWN). Seeing this, can we use that work to apply dynamic scripts to an evolutionary side of RPGs.

## 4.1 The need for scripting in any adaptive AI.

While scripts tend to solve exact problems quite well, they do not lend themselves to solving problems where the situation is uncertain or when there could be multiple solutions to the same problem. However when broken down into smaller scripts, which are essentially just actions they become much more applicable for other adaptive AI techniques. So we are going under the assumption that all our AIs are going to use scripts in some form, whether they are just simple ‘attack’ commands or more complex motions such as ‘find food source’. This kind of breaking down of scripts is already the process used in dynamic scripts when it is being applied to an existing scripted based system.

## 4.2 Discussion on using dynamic scripts.

If we wanted to use dynamic scripts for the purpose of evolving creatures in an RPG what would be the advantages and disadvantages. Dynamic scripts, when implemented well, provide a computationally efficient and quick way for AI agents to learn and adapt to a given situation using the scripts at their disposal. This is achieved through the use a large rule base that contains all the possible actions that an agent can perform and the actual dynamic script, which is a string if these rules put together. The order of these rules is determined through a weight assigned to each rule in the rule base. The weight each rule is then adjusted after a certain important event that can be used to determine the effectiveness of that rule, such as in a combat situation, the performance can be determined by how well the agent performed in combat. If the dynamic script does not perform well then rules that were previously not used have a higher chance of being used to create the next script which may perform better.

While dynamic scripts work great for short term adaptation, they do not lend themselves to any sort of genetic like structure, where related individuals will have similar tendencies unless coded by the developer to do so.

4.3 Discussion on Genetic Algorithms

If we want to use GAs to evolve the behaviours of creatures within an RPG how could it be done using GAs. GAs are in essence a heuristic based search method that uses well performing solutions to create a better solution and takes advantage of random mutations to potentially find novel solutions to a problem. The way a potential solution is measured is with a fitness function that determines how well a solution performed. Solutions with the highest performance are cross-breed together to create new solutions that should hopefully have better performance that either parent. This cross-breeding with well performing solutions and a mutation operator gives the next generation of solutions the best chance to utilise the best of the previous generation and the potential to discover a new approach through the mutations.

However this process can be very slow if none of the current solutions are very good, then the next generation is going to have to rely on a good mutation to occur in order to bring something new to the gene pool. Also with a low mutation rate and tight fitness function, the gene pool can get stuck in local maxima that may not reflect the most optimal solution. However in the context of a game that is supposed to reflect evolution this is an acceptable outcome as it may take some extreme external influences to really push that population to adapt. Also being in the context of a game, local population extinction is an acceptable outcome if it only happens occasionally show the GA fail to find a salutation that keeps the population alive. However should this occur frequently then the game may have a problem with large populations of game agents disappearing, which indicates a general collapse of the ecosystem or that an external event had a very large impact. This can be hard to deal with as this may be the intended result and as such should be left alone, or it could be a failing of the game engine and should be dealt with.

Also the GA by themselves do not deal with adaptation in the short term but rather long term trending towards a better surviving population; which can be hindered by individuals that may have good traits but were unable to deal with short term issues (such as a temporary food shortage).

## 4.4 Discussion on Belief-Desire-Intention model

The BDI model attempts to recreate many aspects of an intelligent biological being, by simulating what it believes, desires and intends to do. The system as a whole works very well for encompassing many aspects of an adaptive agent as it can learn, retain memories and form goals based on the given situation. It is however a rather complicated system and is outside the scope of this thesis in terms of practicality for implementing it within the given time. It is however one of the few more complex AI systems that has been successfully adapted for a mainstream game that was very successful and primarily because of the AI component and is therefore worth mentioning as an important adaptive AI method.

## 4.5 Levels of realism in simulation games

Simulating an ecosystem with populated with different dynamic objects and adaptive AI agents can very easily blow out into being an incredibly complicated system, however there are examples of games that have achieved a level of realism without the games becoming unmanageably complex. One example of a popular game that has achieved this is Maxis The Sims game series, where the player is in control of a varying number of people who each have their own needs and desires, and you must provide for them. Many of these needs and desires are simple representations of what we as humans may have, such as hunger, sleep, hygiene, toiletry, entertainment and several others. These can be often satisfied by a large number of different objects that are however often all quite similar. Such as hunger can be satisfied by eating any kind of food, the only difference is the amount of hunger that is satisfied by a single food item.

This simplifies the AI as it does not need to then categories different foods into different types of foods that the characters can and cannot eat. Each character has a genetic makeup that affects their high level behaviours such as ‘Computer Whiz’ and ‘Evil’, as well as their appearance. These genes are selected by the player when creating their characters, but are then passed onto the characters children when they have a baby with a partner. As the selection for these characters is often based off the player’s decision, the game does not need to have any complex system for choosing which characters are best for breeding, but rather let the player decide, or in the case of characters that the player is not in control of, it is fairly random. Shelter is also not so much an issue, as while the character will prefer to sleep in a bed under a roof, there is nothing that is going to affect their chances of reproduction by sleeping outside without a bed.

This again because the genetics side of The Sims is based primarily on higher level behaviours and appearance rather than survival traits. This does however bring to light how the creators of The Sims have taken a complex system and made a relatively simple model of it, such that the characters in the games can appear to be fairly real without having to simulate everything.

# Chapter 5: Proposal

From looking at the discussion of both the dynamic scripts and GAs, I propose a system that uses both, a dynamic script system for adapting to short term issues where the weights are biased by the genetic algorithm that can select for individuals that end up surviving best. In addition, since each dynamic script is fitted for a given situation, each situation is broken up into different states of an FSM to allow for different states to adapt to the certain needs of that situation. Dynamic scripts have also already been proven to work within the context of an RPG even if it was not from the commercial side, it was still successfully implemented in the paper [[5](#Pie06)].

## 5.1 The system

The system as a whole is a hybrid system of dynamic scripts for adapting to different specific situations, a genetics aspect that controls biasing for the weights in the dynamic script and an FSM for controlling the state which the AI is currently in. The main goal of the genes is to simply affect and direct high level behaviour without it directly controlling the behaviour. This should mean that regardless of what mutation occurs within the genes, the resulting dynamic scripts should still be relatively valid and if given enough time, the dynamic scripts can adapt around any bizarre changes. This should only be a problem in the cases of extreme mutation, however in that case that is also an acceptable outcome, as extreme mutations that affect behaviour are also found in real life. The genes should affect behaviours such as aggressiveness, the likeness for being either a herbivore or carnivore as the main ones, but all other important weights that can be associated with a weight in a dynamic script should also be linked to a gene value. The fitness function for the genetic algorithm itself should adapt primarily based on survivability, which begs the question, how do you test survivability outside seeing how long a creature survives in the wild. The driving force for change in the genes however will be evolutionary pressure (aka selection) where under tough situations, it is only the ones with advantageous genes will survive and the rest will not last long [[17](#Gra96)].

The main goal behind this system is for it to be a process that could have been done by a DM in a table top RPG, but we are taking advantage of the power that a modern computer has in being able to process many creatures every second. And while this may not be something that every DM would do during a typical D&D campaign, it is something that can uniquely be done by the computer to offer a different experience than is available through a table top. Similar to how the graphical immersion and network play is currently offered that is not available through a table top game.

### 5.1.1 The use of existing RPG stats and diet.

Another important factor in the genetic algorithm is the use of the standard RPG stats like strength, dexterity and intelligence. The creatures diet (affinity as a herbivore or carnivore) should control the creatures food energy intake, which in turn affects how much energy a creature has, a creature that has high strength, high dexterity or is simply large in size will need more energy. Different food sources should have different energy benefits; a herbivores food source will generally provide less energy but does not require the creature to be particularly capable in combat. And for the purpose of this exercise plant sources will be abundant as they are not part of the same evolutionary system; they will however be limited making large populations unsustainable. A carnivore’s food supply however provides much more energy but does require the creatures to be able to hunt. While these processes are based off nature, they do not have to reflect it accurately, they simply need to be present in order to get similar behaviour. This limiting factor of diet [[18](#DSM62)]also prevents a breed of super creatures springing up as soon as a mutation occurs that results in a creature that has a very high survivability because it has no troubles getting food. However the other limiting factor is the fitness function which is choosing for survivability, this should also prevent a single population from diverging greatly from the other populations in the local ecosystems.

### 5.1.2 The use of dynamic scripts.

The mechanism that is actually controlling the actions of the creatures will be the dynamic scripts which are primarily designed to do that exact operation. These dynamic scripts are usually controlled by adjusting weights on the rules used to create the full script based how well the script performed and if that rule was used to get that performance. These weights are to be biased based off different genes, such as the diet gene or an aggression gene which will cause certain actions to appear more often in the final scripts than they otherwise would. This should allow for the creatures that have a particularly poor set of genes for a given situation to be able to adapt, unless their genes are completely against them, then they are not fit to survive in that environment anyway and natural selection will play its role in not giving that set of genes a high survivability rate.

Each of these scripts for different situations will have a different fitness function assigned to them, so a combat script will want to inflict as much damage as possible whilst receiving as limit damage as possible. A hunting script on the other hand will want to get as much food energy as possible in as little time as possible. While these two may have some overlap in some areas, they are overall being used for very different situations and will generally require fairly different actions.

### 5.1.3 The finite state machine.

This leads into the use of the FSM, which is used to control the transfer between different stats of the creature. The FSM will also need its own set of scripts for some situations, as in a combat situation, it may be required to reset the combat chain should a new enemy appear and the creature should reassess the situation at hand. However for the most part the FSM will control the change between states and the conditions for the states to change, this can be something high such as while a creature is in the ‘looking for food’, if it sees a potential food source, then it should move into the ‘hunt prey’ state. This can allow for each dynamic script to properly adjust to the situation that it needs to deal with at hand, without having to worry about how it needs to deal with completely different situations too often.

## 5.2 An example

If we take the example of a small game world that has enough plant life to support a population of rabbits which are primarily herbivores, which in turn is enough to support a population of wolves who eat the rabbits. The rabbits are generally not very strong but are quite fast, which means in order for the wolves to eat they must be able to keep up with the rabbits in order to hunt them. The wolves hunting puts pressure on the rabbits to adapt in some form, of course if the population of the rabbits is high enough that in general the overall population is not a dire situation then there is much less pressure applied. However if we assume in this example that the rabbit population is low enough that the wolves are indeed putting evolutionary pressure on the rabbits. This gives the rabbits a few possible courses of action, to either be faster again or to breed more quickly. Should the rabbits become faster, this in turn will put pressure on the wolves as they are their only food source. The wolves will have to either become faster or potentially should the genes allow for it, to eat some plant life. As we are not going to worry about the actual nutrition available from different food sources, only that plants will generally give less energy than meat will, this means that the wolves that do eat plants will survive. Should times get really tough this may mean that the entire population of wolves turn to eating plants, which may not be completely accurate in relation to nature, it also is not impossible. This would require that while in the ‘looking for food’ state, that over time as the ‘find animal to eat’ rule continues to fail to result in any food, the ‘find plant to eat’ would eventually get a chance to execute assuming that the negative biasing of the wolves genes do allow it to be run at some point. Should the genes allow it, the wolves which are able to survive on plants will continue to do so, and perhaps even on the other side of the gene spectrum there are some wolves which are fast enough to catch rabbits will continue to do so. In nature however wolves are pack animals and as such not all of the wolves will be required to be able to hunt, however for the sake of simplifying this already complex system, we are not going to worry about different social structures depending on the creature involved.

## 5.3 Limitations

While the system itself is built on the well-known structures of FSM, dynamic scripts and GAs, it does have limitations. One of the biggest one is of scalability, as to get the intended result the system as a whole needs to be simulating at all times, this means that the larger the game world and more agents in the world, the more computationally expensive everything is. At best the system will scale linearly through the use of good spatial partitioning between different agents and efficient use of memory.

This is primarily a design and implementation issue, where a given game world may be small enough that this isn’t an issue. Or the game world may require that the AI is implemented in such a way that it does not negatively influence the speed at which the game runs.

Another limitation is the complexity of the system, or rather the minimum complexity required. As the system as a whole requires that in order to get any variances as a result of evolutionary pressure, the system needs to be complex enough that different environment require different adaptations, otherwise a single ‘decent’ genetic structure will survive well enough in all situations that there is no need for selection.

These limitations will be explored in further detail in my continued research into the topic.

# References

x

|  |  |
| --- | --- |
| [1] | Jeff Orkin. (2011, Mar.) Jeff Orkins' website. [Online]. <http://web.media.mit.edu/~jorkin/goap.html> |
| [2] | Jeff Orkin, "Applying Goal-Oriented Action Planning to Games," in *AI Game Programming Wisdom 2*, Steve Rabin, Ed. Hingham, Massachusetts, United States of America: Charles River Media, 2004, ch. 3.4, pp. 217-227. |
| [3] | Richard Evans, "Varieties of Learning," in *AI Game Programming Wisdom*, Steve Rabin, Ed. Hingham, Massachusetts, United States of America: Charles River Media, 2002, ch. 11.2, pp. 567-578. |
| [4] | Monte Cook, *Dungeon & Dragons: Dungeon Master's Guide*, 1st ed.: Wizards of the Coast, 2003. |
| [5] | Pieter Spronck, "Dynamic Scripting," in *AI Game Programming Wisdom 3*, Steve Rabin, Ed. Boston, Massachusetts, United States of America: Charles River Media, 2006, ch. 8.5, pp. 661-675. |
| [6] | Ethem Alpaydin, *Introduction to Machine Learning*, 2nd ed.: The MIT Press, 2004. |
| [7] | Roger Schank and Robert Paul Abelson, *Scripts, Plans, Goals, and Understanding*. United States of America: Psychology Press, 1977. |
| [8] | Manuel Ladebeck, Apply Dynamic Scripting to "Jagged Alliance 2", September 2008. |
| [9] | id. Quake 1 Engine Source. [Online]. <ftp://ftp.idsoftware.com/idstuff/source/q1source.zip> |
| [10] | John J Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *PNAS*, vol. 79, no. 8, pp. 2554-2558, April 1982. |
| [11] | Jr Arthur Earl Bryson and Yu-Chi Ho, *Applied Optimal Control: Optimization, Estimation and Control*, Revised ed.: Taylor & Francis, 1975. |
| [12] | (2001, April) Generation5 interviews Jeff Hannan. [Online]. <http://www.generation5.org/content/2001/hannan.asp> |
| [13] | John Henry Holland, *Adaptation in natural and artificial systems.*, 2nd ed.: The MIT Press, 1992. |
| [14] | David Edward Goldberg, *Genetic algorithms in search, optimization, and machine learning.*, 1st ed.: Addison-Wesley, 1989. |
| [15] | B.Sc. Marc Ponsen, Improving adaptive game AI with evolutionary learning, 2004. |
| [16] | Michael E Bratman, *Intention, Plans, and Practical Reason*. Standford, California, United States of America: CSLI Publications, 1999. |
| [17] | Graham Bell, *Selection: The Mechanism of Evolution*, 1st ed.: Springer, 1996. |
| [18] | D. S. Miller and P. R. Payne, "Weight Maintenance and Food Intake," *The Journal of Nutrition*, no. 78, pp. 255-262, 1962. |

x