Knowledge-based Authentic AI for a Open-World Game

CSC 581: Knowledge and Usability • June 4, 2014
Final Version

Ian Dunn
California Polytechnic State University
San Luis Obispo, CA
ian@iondune.com

Abstract

Knowledge representation is a fine candidate for implementing a complex and authentic artificial intelligence system for a game, yet many current game systems avoid knowledge-based systems in favor of more traditional artificial intelligence systems and hacks. We explore the flaws of many simple AI systems and discuss the implementation of a knowledge-based authentic AI system for an open-world sandbox game.

Contents

1 Introduction

As modern games push the envelope by bringing more and more realism to the table, a need for highly sophisticated AI has emerged. However, many modern AAA game titles fail to deliver on that mark. Of particular note is the artificial intelligence systems deployed by many open-world sandbox games. Either AI systems in these contexts are extraordinarily difficult to design, or many games either willfully or unwittingly lack any reasonably intelligent character implementation

1.1 Overview

This paper addresses the nature of open-world games and the artificial intelligence systems necessary to drive them. In particular, we are interested in defining a framework for a knowledge-based system operating at incredible scale and solving many common issues with modern artificial intelligence systems.

We will begin with an overview of some related artificial intelligence systems used in AAA game titles which either utilize knowledge-based methods or are deployed in an open-world session. We then provide high-level design documents for an open world game platform (or rather, the elements of such a platform which are relevant to the implementation of an artificial intelligence system) and the knowledge-based artificial intelligence system itself.[1]

1.2 Open-World Games

This paper is primarily concerned with the implementation of an AI system for an open-world game - it is therefore per-

[1] It is important to note that the systems described in these design documents are for theoretical purposes only - no systems have been implemented and no results have been gathered. As design documents, they are written in present tense and active voice, both of which tend to sound like descriptions of existing systems but are still appropriate for a document such as this.

tinent to give a reasonable definition of what an open-world game is.

In terms of terminology, "open world" games have been around for more than 30 years under a variety of definitions and genres. Until recently, the capabilities of such games were fairly limited by graphical and computational boundaries. Modern games in the genre such as *Grand Theft Auto 4* and *5*, *The Elder Scrolls V: Skyrim* (and *Morrowind* and *Oblivion*, predecessors to Skyrim), *Fallout 3*, *Mass Effect* and even to a certain extent some MMOs like *The Elder Scrolls Online*, *World of Warcraft*, and *Guild Wars* have redefined the genre and opened the doors for future titles.

I have come up with three criteria which I believe can be effectively used to describe open-world games, or at least to set goals for what we may hope open- world games to attain.

Scale
The game world should be significantly large. Obviously the term "large" is fairly subjective. The most important aspect is that there should be differing levels of detail within the game - there should be nearby events which have close effects on the player and far events which have larger- scale but less immediate effects. For a game of reasonable scale, there should be more than two such levels of abstraction. For a more in-depth look at what I mean by these levels of abstraction, see section 5.1.

Choice
The player should be free to make choices, not just binary decisions. These choices should include decisions about where to go at what time as well as storyline decisions that affect who the player aligns with and how the story progresses.

Effect
Finally, those choices must have effect. A decision that a player makes should have both immediate and long term consequences.

1.3 Authenticity

The title of the paper refers to "authentic" AI - a reasonable definition of that term is given here.

First, in carrying out the implementation of an authentic AI system design decisions should (as much as possible) be made to mimic reality, not to work around reality in a way that creates desired results.

Second, the AI characters in the resulting system should behave in a way that we might expect real characters behave. In other words, they should pass some fashion of Turing test, in which it is no possible to easily disambiguate AI characters in the game from real players who are simply immersed in the game and "in character."

2 Background

2.1 Theory

Highly theoretical ontological and otherwise knowledge-based AI systems have been used in some places with varying success [6]. The AI system choices employed in all AAA-title open-world games indicates that such techniques are not employed in any real capacity in industry, but the ongoing issues found in those games shows that there is certainly room for improvement.

2.2 Halo

There have been a number of publications and articles written about the Halo AI system. [4][5] The main components of their system are a behaviour tree, a perception model, and a scheme for storing memories. The latter two are good examples of knowledge and reasoning techniques in use. [2] The behaviour trees are somewhat unsatisfactory since they are designed to be specifically modified and carefully planned. In the context of a game such as Halo that makes sense, though, since it is a story-driven game with roughly linear gameplay.

Design Principles
Isla's paper on the Halo 2 AI system [4] contains a list of four consequences of complexity and four sources of complexity which, when combined and rephrased, make an effective list of seven design principles for AI systems (and perhaps for software in general), listed below:

- **Coherence**: Actions should stop and start at appropriate times and dithering should never occur.
- **Transparency**: It should be possible to a certain degree (except where specifically and intentionally made impossible) to determine the internal state of an AI character through external observation of it's actions.
- **Run-time**: As in all software systems, there is a constraint on allowable runtime. This is particularly pertinent to open-world games where a large number of actors must simultaneously operate in real time.
- **Mental bandwidth**: It should be possible to reason about the internal state and calculations of the system without relying on a computer.
- **Usability**: From a design standpoint, it should be possible to manipulate actors and characters to create stories.
- **Variety**: We must be able to represent a variety of different possible characters which behave differently in different situations.
- **Variability**: Individual characters should have variety in their responses to different scenarios.

2.3 Skyrim

Skyrim has two artificial intelligence systems under the name "Radiant" which are of some interest. The first, Radiant AI, is a system for establishing character personalities by assigning them tasks to perform on a daily basis. Characters also maintain a relationship with the player which affects their reactions to player actions. For example, characters who are friendly to the player may invite the player to dinner if they barge into the character's house in the late evening, whereas characters which are not friendly may be upset by this occurrence.

On the surface these systems seem promising but their implementations are heavily scripted and reliant on preexisting planning and forethought. There are also a considerable number of character quirks which have given Skyrim a reputation for humorously dumb AI.

2.4 Minecraft

No discussion of open-world game systems would be complete without mention of *Minecraft*. On the surface, Minecraft appears to be an open world game because of the significant amount of freedom afforded to players both in game choices and even in style of gameplay. However, I content that Minecraft is not in fact an open-world game by the above given definition because it fails to attain the **scale** attribute.

Many players of Minecraft tout the impressive scale of the game because the worlds which are procedurally generated are theoretically quite expansive. However, deeper analysis shows that the scale of Minecraft is actually quite limited. While the world size in terms of raw horizontal distance is quite impressive, this massive size is only possible through a severely limited vertical scale in which the highest mountains rise only 60 meters above sea- level, and the deepest oceans are only another 60 meters below the surface.

In general, the horizontal scale around the player is also quite limited. Without any sophisticated level-of-detail system, view distance is severely low and only allows players to see objects that are fairly nearby. Coupled with the previously discussed vertical limitations, the world (from a player's perspective) is actually quite small.

In terms of implementation, Minecraft simply ignores scale as defined in this paper's introduction. There is only a single level of abstraction employed by the simulation engine - events which are nearby occur, and anything which is beyond that "nearby" threshold is cached out and entirely static until it is revisited. This is entirely unsatisfactory from a world-simulation standpoint, and disqualifies Minecraft as an open-world game. [2]

3 Existing Systems

3.1 World

The herein described artificial intelligence system is meant for use in developing an open-world sandbox game.

Editability

One facet of this game is that random procedural generation is used for creation of the world, environment, and all characters, structures, and objects within the game. The result of this stipulation is that no manual level design controls may be used. The artificial intelligence system must be implemented on an entirely automated platform. On the one hand, this appears to impose an unfortunate limitation on the design of the system. It is actually, however, conducive to our goals. Consider:

- In an open-world/sandbox environment, the player is free to make modifications to the world. Pre-calculated or designed elements would therefore be invalidated. It is easier to formulate a system that establishes all of its system elements given at least partially arbitrary input than to attempt to recalculate on the fly based on similarly arbitrary modification.
- Adding design-ability as a requirement to an artificial intelligence system is actually an encumbrance - see the above seven principles of system design and the Halo AI paper.[2]

Therefore, as one of our system goals we will not require any manual controls or editors available to designers.

Format

The format of the world will be a volumetric implicit model so that non- functional features such as cliffs, overhangs, and caves can be easily supported. There will be an additional hierarchical model of objects which exist on and in the terrain.

Significant level-of-detail algorithms will be required to support the rendering of such constructs at any significant scale. Similarly, those same level-of- detail aspects will need to be incorporated into the design of the AI system to support the same scale.

Physics and Collision

Obviously a sophisticated system will be needed to support the rigid-body physics and intelligent collision detection required to implement such a game. I will leave the details of such an implementation out of this paper; however, we will assume some required aspects in order to make a discussion about the more interesting AI features possible. Those requirements are:

- Line-of-sight/visibility detection: From arbitrary points in the environment, we should be able to query in reasonable time both a visibility check of a specific object and a general collection of objects which are within line-of-sight.
- Feedback navigation: Controllable actors should specify their movement as a velocity and direction. Continual feedback must be available in the form of a scalar representing movement success. Zero or close-to-zero values represent entirely inhibited movement while large values indicate partially inhibited or entirely uninhibited movement.
- Basic object circumnavigation: If movement of an actor is inhibited by any obstruction which could practically be stepped over or climbed, it should immediately engage such an action, noting the reduce speed in the previously described feedback value.
- Lay-of-the-land survey: Tying into the circumnavigation and line-of-sight principles, the AI system should be able to survey an area from a particular vantage point and make a reasonable estimation of which areas are impassable and which may likely slow down the actor. Note that this is distinct from a path- finding system. Instead, a rough heat-map is generated.

[2] Readers who play *Dwarf Fortress* may be disappointed to find that I make no discussion of that game within this paper. The simple truth is that I am unfamiliar with the inner-workings of this game. Simply playing this game is a massive undertaking - to attempt to understand the inner-workings of its complexity is another thing altogether. For those who are familiar with the game, it is probably safe to say that it suffers tremendously from a certifiably terrible interface system. It may be possible that many of the larger complaints levied in this paper are addressed by *Dwarf Fortress*, but most of us will never know, simply because we lack the wherewithal to actually play it.

The reader may note that I have specifically left out path-finding from the list of required features. Path-finding is, however, often found as an integrated feature in a physics or collision engine. It is specifically left out from here. I make this omission because I find existing techniques for path-finding fundamentally flawed. See further elaboration in section 4.1.

3.2 Game

It goes without saying that some game system must exist on which to build this AI system. The AI system must be tightly integrated with the game components, and I will therefore be introducing some components as part of the following discourse on the system design.

However, some general ground rules should be established. The game consists of a large number of characters who interact via standard interactions such as talking and trading along with some non-standard interactions such as combat and negotiation. At the core level, each character is mostly represented by its health, position, emotional condition, and whatever knowledge it has about the environment it is in. Obviously the last two items are rather loaded; their specific definition is described later in the paper.

4 Problems with Traditional AI Systems

There a number of issues present in many current game AI systems, even among AAA titles touted for their advanced AI. Not all of the following issues exist in all titles, but they are certainly common issues. I believe these problems have become prevalent largely because of two reasons:

- Hack culture: A lot of the game industry is driven by what I like to call "hack culture". The need for incredible performance in the face of limited hardware and a feature-hungry customer base makes a lot of game design about not "how well can we do it" but "how well can we fake it". This is especially prevalent within graphic development.
- Suspension of disbelief: The nature of games often requires a certain level of suspension of disbelief. Nobody ever thinks to question what business a plumber has jumping on turtles and rescuing a mushroom princess from a dinosaur, though nearly everyone plays and enjoys the Super Mario games. Users expect games to follow a very specific set of rules (arguably, that is the definition of a game) rather than exactly mimic something from reality. Arguably, this distinction marks the fine line between simulation and game. However, the introduction of sandbox and open-world games starts to blur this line. Certainly there are some features which should remain fantastical, but there are others where a more realistic touch is appropriate. I believe that AI should be considered among the latter.

4.1 Uncanny Knowledge

Often, AI characters will behave in ways that are only possible through the existing of an omniscient knowledge base. I classify such behaviour as the result of "uncanny knowledge" since it results in characters behaving as though they have incredibly precise knowledge of some things we may expect to be unknown.

Psychic Guards
Perhaps the most notable and plain example of this issue is from *The Elder Scrolls IV: Oblivion*, a 2007 open-world sandbox RPG set in a medieval/fantasy world. An oft-complained topic from the heyday of this game were the so-called "psychic guards". Players would quietly commit a crime in some dark, secluded corner of the world, only to have guards instantly burst into the room to make an arrest.

Path finding
A more commonplace and ubiquitous example of this issue comes in the form of standard path-finding algorithms. Games with good path-finding are touted for the ability of characters to navigate from point A to point B successfully and without getting stuck or otherwise halted. The problem exhibited here is that characters somehow have omniscient knowledge of the environment. The same algorithms that let a character walk through the streets of a well-known town will lead that same character out of an unknown labyrinth as though they had built it themselves.

System Knowledge vs. Character Knowledge
This problem manifests itself from a failure to classify system knowledge as opposed to character knowledge. If properly implemented, the AI system should know equally well how to navigate out of the labyrinth as it does the streets of a city. But from a character's perspective, these two knowledge sets are entirely disparate. This, I believe, is the fundamental flaw with common path- finding systems. Most work by decomposing the entire world into some sort of graph data structure and performing an offline search. This can produce lifelike behaviour when the environment is simple or otherwise expected to be known by a character, but makes little sense in the context of true knowledge representation.

Another name for system knowledge is the "canonical view".

In this described system, we discard the concept of system knowledge entirely. This serves to solve the uncanny knowledge problem, but also comes with an additional added benefit. As the system becomes more complex, it becomes more and more difficult to keep an updated and accurate representation of the system knowledge. Moreover, such a representation becomes increasingly useless when a large number of characters each have their own differing perspectives of the world.

One hallmark argument in favor of system knowledge, however, is memory. Supporting knowledge representation for each and every character is likely to exceed possible memory capacity in short order and is definitely not a scalable solution. Considerable focus on the remainder of this paper will center on establishing data structures and compression methods for reducing the memory requirements of a character knowledge implementation.

4.2 Oblivious Bystander

Another common issues from modern AI systems is what I call the "oblivious bystander" effect. This problem manifests

itself in two different ways. First, there are systems in which characters remain entirely unaware or at least unresponsive to events that occur around them as long as they are not specifically targeted. Consider a mugging or assassination occurring in this streets while characters walk idly by as though nothing is happening. Systems exhibiting this behaviour lack a model for determining whether events are visible to characters, or lack a model for determining character responses to arbitrary events.

Even games which seem to implement appropriate bystander AI, however, are often still guilty of a slightly more subtle version of the oblivious bystander effect. As an example, characters in *Grand Theft Auto V* will react by running in fear if they are near to scene of a crime. However, this is really just an example of a system in which characters respond to events that tag them. A discrete crime event occurs, so all characters in a specified area react. Unless the event is of a predetermined and clearly defined sort, characters will not respond to it. Non-scripted events do not produce such reactions. If the player behaves absurdly but in an unscripted manner, character reactions do not occur or are nonspecific.

4.3 Canonical View

As previously discussed as an issue related to uncanny knowledge and path finding, the canonical view or system knowledge is the presence of knowledge which is gathered by the omniscient system and is considered absolute. Such knowledge poses problems in both acquisition and utilization. First, rules must exist for establishing the exact interpretation of the existing world state. Second, rules must exist for applying that data to individual characters with unique perspectives.

It is difficult to determine for a given world how to evaluate such criteria without introducing a natural bias.

5 Solutions

In the following paragraphs, I outline my solutions to the previously described problems and, generally, introduce the principles and systems of the implementation.

5.1 Bigger-Picture Model

The bigger-picture model is the system used to allow scalability in the described AI system. The basic principle of the system is that each character must have its own knowledge model in order to correctly simulate complex scenarios. The quality of such a simulation is directly linked to the size of the knowledge representation utilized. We therefore have a memory problem, and a system which is entirely unscalable.

Local Events

To address this issue, we utilize what is called the "bigger-picture" model. Consider a simulation of a continent which contains many nations, many cities, and many citizens. Our simulation consists of players who are experiencing and are involved in the world. For the characters nearby the player, we must consider the knowledge model of each individual in order to accurately portray the situation. Two peddlers are making a trade, so we must analyze their motivations and emotions so that an interaction can take place. Perhaps a fight breaks out, and nearby characters must intervene to break it up. All of this simulation must occur at a finite scale so that it can be observed by the player. The thoughts, emotions, and actions of each involved individual are important to the accuracy of the simulation.

Remote Events

Consider now a similar fight taking place three blocks over from the player. The player's interaction with this event is far more limited. Perhaps if the fight becomes loud the player will be able to hear it happening. Maybe the player will go down that street at a later time and hear talk of what occurred. All of these interactions occur at a different level of abstraction than the previously discussed events.

While the events that occur around the player may directly involve them, the events that occur at a slightly more remove level will often only indirectly involve the player. We may consider this a difference of one level of indirection.

Of course it is always possible that these remote events may trigger events with local consequences. It may even be possible that individual interactions are key to determining whether such an escalation takes place.

Consider that the two arguing peddlers may happen to be of two different rival clans. The argument may now escalate into a brawl, and from there into a riot. Soon the player has become directly involved in this interaction since the growing scale of a riot may engulf them.

However, if we can devise a general model for representing these remote events at an abstract level such that only the important elements are observed – either through ambient effects, historic significance, or escalation into a larger event with more widespread effect; then, we have introduced a scheme for reducing the memory and computation overhead of representing extensive knowledge structures on a large scale.

Another way to consider this model is to think in terms of "leak". Interactions are only important in the effects they have on the player. Events that take place in close proximity leak nearly everything since all aspects of the actions are directly observable. We must therefore employ a fully or nearly fully accurate model of simulation to this event. However, a remote event leaks only some information - only the aspects which may directly affect the player or indirectly produce some cause at a later date. We may therefore use a less specific and detailed simulation of the event. Interactions which are event more remote leak less volume and less often. Part of the bigger picture model relies on accurately determining what information leaks from a given interaction.

Lazy Evaluation

Our scheme for implementing the bigger-picture model is similar to the concept from programming language design called "Lazy Evaluation." In the programming language context, lazy evaluation is a system for detecting calculations which are described by the language but not used and avoiding unnecessary work in performing these computations. In general, the problem is similar to a reachability problem - if evaluation will never require a value, it is safe to not determine that value. If at some point in the future that value is needed, however, we can return to the computation and calculate the value.

In the context of our AI system, this means that any event which is at a level of abstraction above the player's current experience context can potentially be put aside in terms of simulation evaluation until the effects of that event are required, either because of escalation or a change in the player's location. Obviously such a scheme is reliant on a reliable way to compute the possible escalation of an event.

Terminology

We devise the following terms for representing the concrete relationships between events.

Interactions: The smallest composable piece of temporal information in the AI system. Consists of any two characters and some exchange of information or material. Note that the exchange of information described can be applied at a highly abstract level. Two characters sharing a glance as they walk through the street is considered an interaction.

Events: A composition of interactions. Some representation for determining related interactions and combining them into a single element. An event is also hierarchical - events can be combined to form a single, larger event.

Level of Indirection: The distinction between events which require one defined abstraction versus a lower or higher abstraction. When multiple events are composed into a larger event, that larger event has a singularly higher level of indirection. This classification is vital to the performance of the system. As level of indirection increases, the level of detail required for simulation decreases, as does the criteria for leaking information.

Translation: The term for leakage of information from an event. Whichever details of an event become important to the application of other events become translated when the information is shared.

Escalation: The term for a significant change in an event which produces either mass translation or a level of indirection change. Note that while the word seems to imply uplifting, escalation actually refers to a reduction in level of indirection.

Experience Context: The current environment of the player, consisting of all visible events and interactions at whatever level of indirection the system has chosen such that the effects of said events and interactions is known, or can be known, by the player.

Locality

Note that by necessity, physical locality is not the only criteria by which events are subdivided. In general, it may be more effective to subdivide the world based on semantic organizational structure. For example, at the large scale it is advantageous to consider countries as individual pieces; at a larger scale, we might do the same for continents. Such subdivisions make logical sense for our implementation, but may not always follow exactly how a subdivision of similar scale but using raw position would subdivide the same area.

6 Knowledge Representation

In this section we discuss the basic units of our knowledge based system. Specifically, we define the five general types of knowledge we must represent and give some examples and implementation details for these types.

The types are Experiences, Facts, Beliefs, Personality Traits, and Emotions.

6.1 Experience

An experience is perhaps one of the most heavy units of knowledge we must represent, but it is also one of the most vital to the defined system. An experience is defined as a record of one character's memory of a certain event in the context of that event's effects on the player. This knowledge unit plays a large role in the decision making process for a character, as described in the following subsection. Given an extraordinary number of characters that we must simulate and a potentially large duration in which experiences may stack up, careful consideration must be placed on how to store this data.

Nature vs. Nurture

The question of "nature versus nurture" is of ongoing debate. The essential question as is relevant to our discussion is whether a predefined set of behavioural characteristics can be used to accurately predict a character's response to any arbitrary event and the degree to which acquired experiences will affect responses. In the interest of not taking a stance on an issue that is outside the scope of this paper, and in pursuit of a scheme for representing interesting character interaction, we split the issue directly in half by utilizing both a defining set of characteristics and a sum of experiences with equal weight to determine character responses.

Experience Trees

It may seem intuitive to encode the set of a characters experience as a linear stream of events sorted chronologically. We choose to instead represent a character's recalled experiences in a tree format with less emphasis on the temporal organization of that data. There are two reasons for making this distinction.

First, it is a more accurate representation of realistic thought processes. People do not remember their lives as a list of chronologically sorted events; instead, memory functions like a tree of related recollections which are triggered by experiences and other memories.

Second, this representation allows us to utilize the exact same bigger-picture representation and lazy evaluation scheme in representing character experiences.

Representation

In principle, the representation of experiences is closely tied to the following two knowledge types, facts and beliefs. Experiences also consist of some unit of time (though this time representation should be non-absolute, in accordance with our principle of avoiding excessive use of system-knowledge). Our representation will also incorporate some elements of emotion.

6.2 Facts

Facts are a unit of basic knowledge that a character believes to be true. An example of a fact might be the path between two locations. In this sense, a fact is "irrefutable". This doesn't

mean that it is absolutely correct, but instead that its correctness is a binary question. Facts are generally things which were true at one point, or which may have been considered by any reasonable observer to be true. It is possible that a fact may no longer be true even if it originally was, but this distinction should occur unbeknownst to the character.

A more general way to consider this definition of a fact is that two characters should be able to settle an argument over the veracity of a fact by making an observation which immediately corrects both character's representation of that fact. In principle, this means that facts should consist mostly of small and simple ideas related to the observable world.

6.3 Belief

In polar opposite to facts are beliefs. Beliefs are closely related to facts but are usually more complex in scope. In particular, any notion of preserving the veracity of such information is entirely forgotten.

In terms of representation, beliefs have an associated confidence factor and, more importantly, some notion of a reason for why that belief is held.

The interpretation of a confidence factor for a belief is closely tied to the personality attributes of the owning character.

6.4 Personality

Arguably, personality is the most difficult of the discussed concepts to codify. Our separation and distinction between personality and the experience tree makes this somewhat less complicated. We opt to represent personality as a set of defined attributes with numeric values.

For more on methods for discretizing personality traits into computer-friendly formats, see Evans [3] which represents personalities using conditionals.

6.5 Emotion

We chose a scheme for representing emotion which is similar to that for representing personality - a predefined set of numeric values. This is additionally some overlap between the fields for representing personality and the fields for representing personality, though the application is sometimes quite different.

7 Interaction Representation

We must also come up with a scheme for representing interactions that occur between actors in the simulation.

For the most part, the temporal nature of these interaction means that we do not actually require a method for storing them - often, an interaction is merely an instance of something which was computed by the simulation engine. There are some components for which representation detail is necessary, however.

7.1 Language

Perhaps the largest component of an interaction space between characters in a game or simulation is their communication. Special attention must therefore be placed on the medium for this communication, or language.

We therefore choose a representation of language that allows for multiple languages (and the complexity of interaction that arises at language barriers), and even multiple dialects of those languages.

It is important to keep this language representation as simple as possible, however. There is a potential rabbit hole of complexity within the field of linguistics which we could dive into in attempting to devise a universal representation of language and communication. Instead, we establish the core concepts necessary to implement our requirements of multiple languages and dialects.

Our language definition therefore consists only of a mapping of arbitrary symbols to our previously defined knowledge representation units. Dialects may be represented by having languages which use slightly different mappings and foreign languages may be represented by an entirely different mapping.

8 Machine Learning

The most crucial questions to answer when devising this system are:

- How do we know what events might escalate?
- How do we know what they escalate into?

Remember that high level representation of events is supposed to be an accurate representation of what would actually occur if we were to individually simulate the events on the lower level of indirection. Direct simulation of those events is therefore the best way to establish a set of rules for determining the large scale effects of events.

We utilize machine learning to establish a set of rules for determining the possible escalation schemes for an event or set of events. This system is designed to be used as a long-term preprocess step that establishes an initial dataset for deployment as well as a live background process that lives and grows with the simulation while the player is playing.

The high-level description of the system is as follows. We initialize a training data set of characters and run simulations at the smallest possible level of indirection (in the theoretical experience context). The results of this simulation are used to simulate the next level of indirection up by composing the input and output of each individual locality. The process can then be repeated to establish an escalation model for the entire knowledge tree.

9 Conclusions

It is difficult to draw conclusions from a design document. The design principles in the above specification are based on three main influences:

- Analysis of common shortfalls found in existing AI systems.
- An attempt to represent actual events and interactions as truthfully as possible ("Authenticity")
- Level of detail techniques, in particular a focus on hierarchical representations, lazy evaluation, and separable simulation

In principle, the above description is sound. If there is one thing we have learned through analysis of existing AI systems, though, it is that authentic and usable AI systems are *hard to make*. I believe that the described system is novel in design and may successfully address many of the current issues in AI system design, but only an implementation will truly tell.

References

[1] Matt Bertz. The technology behind the elder scrolls v: Skyrim, jan 2011.

[2] Max Dyckhoff. Decision making and knowledge representation in halo 3. *Bungie Publications*, aug 2008.

[3] Richard Evans. Representing personality traits as conditionals.

[4] Damian Isla. Gdc 2005 proceeding: Handling complexity in the halo 2 ai. *GDC*, mar 2005.

[5] Damian Isla. Building a better battle: Halo 3 ai objectives. *Bungie Publications*, mar 2008.

[6] Antonio Snchez-ruiz, Stephen Lee-urban, Hctor Muoz-avila, Beln Daz-agudo, and Pedro Gonzlez-calero. Game ai for a turn-based strategy game with plan adaptation and ontology-based retrieval , 2007.