1

What Is Game AI?

Kevin Dill

1.1 Introduction

Game AI should be about one thing and one thing only: enabling the developers to create a compelling experience for the player. Every technique we use, every trick we play, every algorithm we encode, all should be in support of that single goal.

Wikipedia gives the following as a definition for artificial intelligence (or AI): "the study and design of intelligent agents," where an intelligent agent is "a system that perceives its environment and takes actions that maximize its chances of success" [Wikipedia 12-A]. This is certainly not the only definition for the term—"artificial intelligence" is a term that is notoriously difficult to define—but it does accurately describe much of AI as it is researched and taught in our universities. We will use the term *academic AI* to describe AI as it is commonly taught today.

Film animators often describe the artificial life which they create as the *illusion of life*—a phrase which is believed to have originated with Walt Disney. This is a very different goal. The characters in a cartoon don't necessarily "take actions that maximize their chances of success"; Wile E. Coyote, for example, does just the opposite much of the time. Instead, they seek to engender in the viewer a gut-level belief in their reality (despite the fact that they are obviously artificial), and to create a compelling experience, which is what movies are all about.

Every game is different, and the AI needs for games vary widely. With that said, the goals for a game's AI generally have much more in common with Disney's view of artificial life than with a classic academic view of AI. Like cartoons, games are created for entertainment. Like cartoons, games are not about success maximization, cognitive modeling, or real intelligence, but rather about telling a story, creating an experience, creating the *illusion of intelligence* [Adams 99]. In some cases, the techniques that we need in order to create this illusion can be drawn from academic AI, but in many cases they differ. We use the term *game AI* to describe AI, which is focused on creating the appearance of intelligence, and on creating a particular experience for the viewer, rather than being focused on creating true intelligence as it exists in human beings.

1.2 Creating an Experience

It is often said that rather than maximizing the chances of success, the goal of game AI is to maximize the player's fun. This certainly *can* be the goal of AI, but it probably isn't the best definition. For one thing, much like the term AI, "fun" is a word that is notoriously hard to define. For another, not all games are about fun. Some games are about telling a story, or about the really cool characters that they contain. Others are about creating a sense of excitement, adventure, suspense, or even fear (like a horror movie). Still others are about giving the player a sense of empowerment, making him (or her) feel like "The Man."

The one thing that is universally true is that games are about creating a particular experience for the player—whatever that experience may be. The purpose of Game AI (and every other part of a game, for that matter) is to support that experience. As a consequence, the techniques that are appropriate for use are simply those that best bring about the desired experience—nothing more, nothing less.

1.2.1 Suspension of Disbelief

Players are willing participants in the experience that we are creating for them. They want to buy in to our illusion, and so willingly suspend the natural disbelief they would normally feel for such obviously artificial characters and events. With that said, it is our responsibility to provide an illusion that is sufficiently compelling to enable them to do this—that is, to maintain the player's *suspension of disbelief*. We succeed any time that the user thinks about and responds to the AI as if it were real, even if the underlying algorithm is actually quite simple. We fail any time that some action (or inaction) on the part of the AI reminds the user that the AI is only a machine program, not real. ELIZA—an AI psychologist developed by Joseph Weizenbaum in 1964 [Wikipedia 12-B]—exemplifies both how easy it can be to capture the player's belief with a simple algorithm, and how quickly you can lose that belief when the algorithm misbehaves.

Because they are willing participants in the experience, and because of the way the human mind works, players are actually quite forgiving. As long as the AI produces behavior that is basically reasonable, the player's mind will come up with explanations for the AI's decisions, that are often quite complex—much more complex than what is really going on inside of the AI—but also fundamentally compelling and believable. In fact, to some extent it can be a mistake to build an AI that thinks too hard. Not only can it be a waste of precious developer hours to overengineer your AI, but it can also result in a character that will perform actions that, while sensible to the AI, don't match the player's mental model of what the AI is doing. In other words, the actions make sense if you know what the AI is thinking—but, of course, the player can't know that. As a result, those carefully chosen decisions end up looking random or just plain wrong.

The one thing that we absolutely must avoid at all costs is *artificial stupidity*—that is, selecting an action that looks obviously wrong or just doesn't make any sense. Common examples are things like walking into walls, getting stuck in the geometry, or ignoring a player that is shooting at you. Even some behaviors that actual humans would display should be avoided, because those behaviors *appear* inhuman when performed by an AI-controlled character. For example, humans quite frequently change their minds—but when an AI does so, it often gives the impression of a faulty algorithm rather than a reevaluation of the situation.

One solution to the problem of artificial stupidity is simply to make the AI better—but player expectations can be so diverse that it is hard to meet all of them all of the time. As a result, a variety of other approaches have been used. In some games—zombie games are a good example of this—the characters are deliberately made to be a little bit stupid or wonky, so that their strangeness will be more acceptable. In others, the characters use short spoken lines, sometimes called "barks," to clue the player in to what's going on. For example, they might yell "Grenade!" or "I'm hit!" These aren't really used to communicate with other AI characters (we do that by passing messages in the code), but rather to explain their actions to the player. Some games (such as *The Sims* or *Zoo Tycoon*) go so far as to put icons over the characters' heads, signaling what's going on internally. *Creatures*, a game still known 15 years after its release for its ground-breaking AI, even used a "puzzled" icon when a creature changed its mind, to show that the resulting change in behavior was deliberate.

1.2.2 Reactivity, Nondeterminism, and Authorial Control

Much discussion has been given to the various architectures and which is the best for game AI. Indeed, an entire section of this book is dedicated to that purpose. The first thought that one might draw from Academic AI is to build an AI with a heuristic definition of the desired experience, and then use machine learning to optimize for that experience. There are several problems with this approach, but the most obvious is that the experience is typically something defined by a game designer—who may or may not be a programmer—using squishy human language terms. How do you write a heuristic function to maximize "fun" or "excitement" or "cool attitude?"

That isn't to say that heuristic functions are useless—in fact, utility-based approaches to AI are one of the more common approaches, particularly for games with more complex decision making (e.g., strategy games, and simulation games like *The Sims* or *Zoo Tycoon*). It is critical, however, to retain *authorial control*—which is, to say, to ensure that the AI's author can tune and tweak the AI so as to ensure that the desired experience is achieved. If we relinquish that control to a machine learning algorithm, it becomes much more difficult to ensure that we get the results we want.

There is a competing need, however, which is that we want our characters to be *reactive*—that is, able to sense the environment and select actions that are appropriate to the subtle, moment-to-moment nuance of the in-game situation. Reactivity and authorial control are not mutually exclusive. You can build a system that is reactive, but because you control the way that it evaluates the situation when making its decisions it still provides authorial control. Controlling a reactive AI is more complex, however, because you as the developer have to think through how your changes will alter the AI's decision making, rather than simply changing what the character does directly

There isn't a single right answer here. Some games (such as strategy games or games like *The Sims*) require more reactivity, while other games (such as *World of Warcraft*) make a deliberate design decision to have a more heavily scripted AI that delivers a carefully

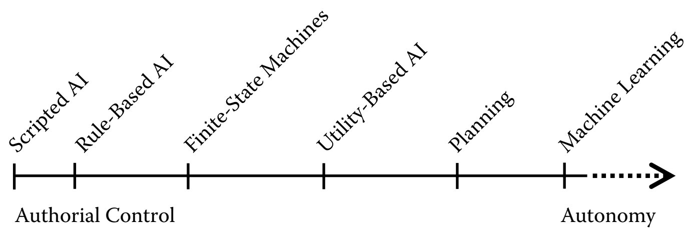

Figure 1.1

The tradeoff between authorial control and reactivity for certain popular AI architectures.

crafted—but highly predictable—experience to the player. Neither is wrong—there are excellent games of each type—but the experience being delivered is different, and that's something that you need to think about when choosing your approach to AI.

There are a variety of architectures that are popular for game AI, many of which are discussed later in this book. Some provide better reactivity, while others allow more direct authorial control. Figure 1.1 shows a rough graph that gives some indication of this tradeoff for several of the most popular game AI architectures. Bear in mind, however, that each architecture has its own advantages, disadvantages, and idiosyncrasies. This is intended as a rough guideline only.

Of note, machine learning is placed on the graph merely due to its popularity in academic circles. Very few games have used machine learning for their core AI. Also, behavior trees have been deliberately omitted from this graph. This is because the performance of a behavior tree depends greatly on the types of decision-making components that are used. If these components are all simple, such as those originally envisioned by Damian Isla [Isla 05], then behavior trees fall in much the same space as finite-state machines. One of the great strengths of a behavior tree, however, is that each node can contain whatever decision-making logic best fits, allowing you to use the most appropriate architecture for each individual decision.

Another complicating factor when selecting an AI architecture is nondeterminism. For many games, we want to add a certain amount of randomness to our characters, so that they won't be predictable (and quite possible exploitable) by the player. At the same time, we don't want the AI to pick an action that is obviously wrong, so we need to ensure that our random choices are all still reasonable. Some architectures are more conducive to adding a bit of randomness into the mix (in particular, behavior trees and utility-based architectures handle this well), so that is another factor that you may need to take into account when designing your game.

1.2.3 Simplicity and Scalability

Both the need for authorial control and the avoidance of artificial stupidity require that the configuration of game AI be an iterative process. Configuring an AI so that it will handle every possible situation—or at least every likely one—while delivering the author's intent and compelling, believably realistic behavior is far too difficult to get right on the first try. Instead, it is necessary to repeatedly test the AI, find the worst problems, correct them, and then test again.

Brian Kernighan, codeveloper of Unix and the C programming language, is believed to have said, "Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it" [Kernighan]. This goes double for game AI. Any change to the code could have unintended side effects. That is, you may fix a bug or a balance issue in one place, only to cause a more subtle issue somewhere else. A simpler underlying algorithm means that you can hold more of the AI in your head. As a result, you will be able to more fully envision all of the side effects of a change, your development will be safer and more rapid, and the final result will be more highly polished (and, for lack of a better term, more "fun" to play).

If you look at the sorts of decision-making algorithms commonly used in games—finite-state machines, scripting, behavior trees, weight-based random, even goal-oriented action planning—the algorithms themselves are quite simple. The configurations built on top of those frameworks may be complex indeed, but the underlying code is simple, easy to understand, and easy to trace through and debug.

There is a caveat, however. Many simple algorithms (finite-state machines being the archetypical example) scale poorly as the AI grows. In the case of finite-state machines, the number of transitions grows exponentially with the number of states. Clearly, this becomes unmanageable quickly. Thus, a truly elegant architecture is one that is not only simple to understand, but also simple to use—which, among other things, means that it must scale well.

1.2.4 Tricks and Cheats

Much has been said about cheating with respect to game AI, but it often seems that we can't even agree on what "cheating" is. Is it cheating to make the AI character a little bit more powerful than the player? What if I give them an item to justify that bonus? Is it cheating to give the AI character in a strategy game an economic bonus, so that they can buy units more cheaply? What if I allow the player to pick the size of the bonus, and call it "difficulty level?"

There is a great story, which I recently confirmed in conversation with Bob Fitch, the AI lead for Blizzard's strategy game titles [Fitch 11]. Apparently the scenario AI for the original *Warcraft* would simply wait a fixed amount of time, and then start spawning a wave of units to attack you. It would spawn these units at the edge of the gray fog—which is to say, just outside of the visibility range of your units. It would continue spawning units until your defenses were nearly overwhelmed—and then it would stop, leaving you to mop up those units that remained and allowing you to win the fight.

This approach seems to cross the line fairly cleanly into the realm of "cheating." The AI doesn't have to worry about building buildings, or saving money, or recruiting units—it just spawns whatever it needs. On the other hand, think about the experience that results. No matter how good or bad you are at the game, it will create an epic battle for you—one which pushes you to the absolute limit of your ability, but one in which you will ultimately, against all odds, be victorious.

Of course, there's a dark side to that sort of cheating, which is that it only works if the player remains ignorant. If the player figures out what you're up to, then the experience they have is entirely different—and nobody likes being patronized. Unfortunately, these days, particularly with the advent of the Internet, players are quite a bit harder to fool (and quite a bit less forgiving) than they were in 1994.

Another type of cheating is purely informational. That is, does the AI have to sense a unit in order to know that it exists and where it is located? The problem is that, while doing line-of-sight checks for visibility is fairly straightforward, remembering what you saw and using that to predict future events is quite a bit harder. In other words, if I see a unit but then it goes out of sight, how do I remember that it exists? How do I guess its location? If I see a unit of the same type later on, how do I know whether it is the same unit or a different one? Humans are fairly good at this sort of thing, but doing it well requires a combination of opponent modeling, intuition, and sheer guesswork. These are things that computers are notoriously bad at.

Unfortunately, for many types of games it is critically important that the AI have a reasonably good ability to predict things like the locations of resources, and enemy strengths and positions. If the player gets this wrong then they will lose—but that is an acceptable sort of a loss to most players. "I just never found anything I could work with," or "You tricked me this time—but next time I'll get you!" They start another game, or reload from a save, and if anything the challenge pulls them right back into the game. If the AI gets it wrong, on the other hand, then the player will win easily, never experiencing any significant challenge at all. They won't be thinking, "Well, the AI probably just had bad luck." They'll be thinking, "Man, that is one stupid AI." Once they start thinking about the AI as "stupid," the experience you were chasing is almost certainly lost.

At the end of the day, the decision of whether or not to write an AI that cheats is a relatively simple one. You should make the AI cheat if and only if it will improve the player's experience—but bear in mind that if you cheat and get caught, that in itself will change the player's experience, and typically not for the best. In *Kohan 2*, an RTS that received accolades for its AI, we had two subtle cheats. First, when exploring, we gave the AI a random chance every 30 seconds or so to cheat and explore an area where we knew there was something good to find. This helped us to avoid games where the AI just didn't happen to find any of the nearby resources early enough in the game. The second was to keep track of the approximate amount of enemy strength in an area (but not the specific locations of units). This allowed us to assign reasonable amounts of strength to our attack and defend goals, and to avoid sending units off on a wild goose chase. None of the reviewers caught on to either cheat, and in fact many of the intelligent behaviors they ascribed to the AI were, as suggested earlier, really just side effects of the ways in which we cheated.

1.3 Conclusion

Academic AI can be about a great many things. It can be about solving hard problems, recreating human intelligence, modeling human cognition in order to learn more about how our brain works, optimizing performance in complex operational environments (for instance, for an autonomous robot), or any of a host of other extremely challenging and worthwhile pursuits. All of these things are hard and worth doing—but the solutions that apply to them don't necessarily apply to games.

Game AI should be about one thing and one thing only: enabling the developers to create a compelling experience for the player—an experience that will make the player want to spend time with the game, and want to buy the expansion packs and sequels, that will inevitably result if you succeed.

The rest of this book is packed with tips, tricks, techniques, and solutions that have been proven to work for our industry. Game schedules are tight, with little margin for error and few opportunities for extensions if the AI isn't working right when it is time to ship. Furthermore, simply building a compelling AI for a game is challenge enough. We have neither the time nor the inclination to tackle the hard problems where we don't need to, or to reinvent solutions that have already been found. With those challenges in mind, hopefully some of the approaches found within will work for you as well.

References

[Adams 99] E. Adams. "Putting the Ghost in the Machine." Lecture, 1999 American Association for Artificial Intelligence Symposium on Computer Games and AI, 1999.

[Fitch 11] B. Fitch. "Evolution of RTS AI." Lecture, *2011 AI and Interactive Digital Entertainment Conference*, 2011.

[Isla 05] D. Isla. "Handling complexity in the Halo 2 AI." *2005 Game Developer's Conference*, 2005. Available online (http://www.gamasutra.com/view/feature/130663/gdc_2005_proceeding_handling_.php).

[Kernighan] B. Kernighan. Original source unknown. Available online (http://www.software-quotes.com/printableshowquotes.aspx?id = 575).

[Wikipedia 12-A] Wikipedia. "Artificial Intelligence." Available online (http://en.wikipedia.org/wiki/Artificial_intelligence, 2012).

[Wikipedia 12-B] Wikipedia. "ELIZA." Available online (http://en.wikipedia.org/wiki/ELIZA, 2012).