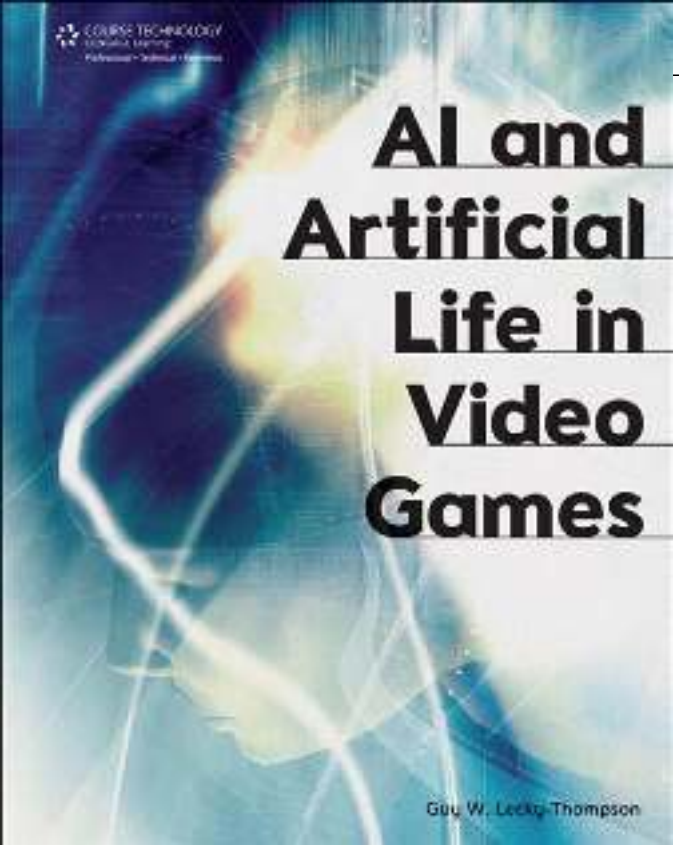
COURSE TECHNOLOGY
AI and Artificial Life in Video Games
Guy W. Lecky-Thompson

AI and Artificial Life in Video Games

Guy W. Lecky-Thompson

Charles River Media
A part of Course Technology, Cengage Learning

COURSE TECHNOLOGY
CENGAGE Learning

Australia, Brazil, Japan, Korea, Mexico, Singapore, Spain, United Kingdom, United States

AI and Artificial Life in Video Games

Guy W. Lecky-Thompson

Publisher and General Manager, Course Technology PTR: Stacy L. Hiquet

Associate Director of Marketing: Sarah Panella

Manager of Editorial Services: Heather Talbot

Marketing Manager: Jordan Casey

Senior Acquisitions Editor: Emi Smith

Project Editor: Karen A. Gill

Technical Reviewer: Mark Morris

CRM Editorial Services Coordinator: Jen Blaney

Copy Editor: Barbara Florant

Interior Layout: Jill Flores

Cover Designer: Mike Tanamachi

Indexer: Jerilyn Sprotson

Proofreader: Gene Redding

Library of Congress Control Number: 2007939377

ISBN-13: 978-1-58450-558-7

ISBN-10: 1-58450-558-3

eISBN-10: 1-58450-616-4

Course Technology
25 Thomson Place
Boston, MA 02210
USA

Cengage Learning is a leading provider of customized learning solutions with office locations around the globe, including Singapore, the United Kingdom, Australia, Mexico, Brazil, and Japan. Locate your local office at: **international.cengage.com/region**

Cengage Learning products are represented in Canada by Nelson Education, Ltd.

For your lifelong learning solutions, visit **courseptr.com.**

Visit our corporate Web site at **cengage.com.**

Printed in the United States of America
1 2 3 4 5 6 7 11 10 09 08

Dedication

This book is for my wife Nicole and my children, Emma and William.

ACKNOWLEDGMENTS

Once again, I owe a debt of thanks to the skills and persistence of the editing team. Karen Gill deserves special mention for her work managing to keep the balance between my initial prose and the finished product (as well as making sure we all did what we were supposed to!). A big thank-you also to Barb Florant for actually editing the prose and making sure that I made sense all the way through.

Mark Morris deserves credit for helping keep me as technically correct as possible. He gave some very valuable insight into some of the techniques and technologies presented in this book.

The behind-the-scenes team also gets a mention for work in preparing the book and making it look as good as it does, especially Gene Redding, for proofreading; Mike Tanamachi, for sharpening my slightly amateur illustrations; Jill Flores, for laying out the final copy; and Jerilyn Sprotson, who gave the book the all-important index.

Finally, a big thank you to Emi Smith for helping deliver the project from inception to publication.

My family, once again, has stuck with me through the process of creating the book. My wife Nicole and children, William and Emma, all helped me, either knowingly or unknowingly, one way or the other, to write it.

About the Author

Guy W. Lecky-Thompson is an experienced author in the field of video game design and software development, whose articles have been published in various places, including Gamasutra and the seminal G*ame Programming Gems*. He is also the author of *Infinite Game Universe: Mathematical Techniques*; *Infinite Game Universe, Volume 2: Level Design, Terrain, and Sound*; and *Video Game Design Revealed*.

CONTENTS

PREFACE

Welcome to *AI and Artificial Life in Video Games*, a book intended to challenge as well as inform designers/programmers who want to add artificial intelligence (AI) and artificial life (A-Life) to their next video game creation. This is not a textbook on AI, nor is it an attempt to write a definitive work on A-Life. Both fields are continually expanding, but some solid ground rules are already in place, which we can leverage in creating better games. AI helps build better games. It can increase difficulty while also helping the player with the actual mechanics of the game. In other words, AI helps the machine to act in a way that will challenge the player, as well as help him by reacting intelligently to his actions.

By a similar token, A-Life can add an extra dimension of playability and immersion, while also providing some unpredictability to the nature of standard AI behavioral models. AI deals with the game's actual thinking; A-Life deals with making the environment more lifelike.

Some games are simply not possible without AI—chess, for example—whereas other games are built upon the premise of A-Life—*Creatures* being perhaps the first and most well known. Either way, any game, at many phases of video game development, can use the techniques in this book.

Be warned, however, that this is not a book about AI in video games, as used in the industry today. Instead, it is a look at techniques that can be used to add AI and A-Life easily and cheaply to video games, without extending the current AI techniques beyond their current capacities too much. As Mark Morris, of UK games developer Introversion, states,

"So why aren't we seeing all these wonderful techniques in games now? Is it the fault of the men in T-shirts or those in white coats, and what do we need to do to ease the passage of research from the lab to production-level video games?"
[http://forums.introversion.co.uk/introversion/viewtopic.php?t=1216]

Despite the huge advances made in processor power and associated resources in both the PC and console gaming world, over the past two decades, AI in games has not advanced one iota. 99 percent of commercial games on the market in 2008 use a combination of the basic AI techniques from games of the 1980s—namely, state machines, rule-based systems, and planning techniques.

Surprisingly, AI and A-Life can be used not only to model behaviors *inside* a game, but they can also be used to help create tools, generate code, and test the game under development. Any real-time system that has to control multiple complex variables can benefit from developmental AI and A-Life, as well as use AI and A-Life techniques to properly and completely test the systems.

This book does introduce the basic AI building blocks, which are entirely necessary in the creation of video games, but it also delves into more advanced techniques, which have only been exploited in a few games to date. What I want to make clear right now is that techniques in this book could be applied to any of the mainstream games and *would* probably enhance them immeasurably.

The industry, however, has been slow to adopt the best AI and A-Life techniques, something that I hope can be influenced in part by my writing this book. As Mark Morris says,

> *"We constantly criticize the academics and say that their techniques would not work on real games, yet when they ask for some source code, we tell them that there are "IPR" issues or that we do not have time to work with them."*

This points to one of the problems that this book seeks to address: AI and A-Life do not get a look in because of the protectionist nature of the industry. For the record, Introversion is active in bridging the gap between the academic and speculative side and the mainstream "real games" side of the industry. This studio always seems to have time to help out.

To address this issue, this book has been constructed to try to show the current state of the art (which has not changed for 20 years, in some cases) and then add to those techniques in a way that produces something that bridges that gap. This book is organized into chapters that give the reader a path through the techniques, covering things that might be familiar, and using them as building blocks for those things that almost certainly will not be.

Chapter 1, "Introduction," delves into the possibilities of using AI and A-Life, while also presenting a quick refresher course on what they actually are. The intention is to establish some high-level definitions of the concepts that we will work with specifically in video games, rather than attempt to discuss all-encompassing scientific definitions that can be used in any situation.

Having digested this, Chapter 2, "Using Artificial Intelligence in Video Games," covers the various techniques and genres in which AI has been used, as well as points the way toward possible future uses. Some of the successes and failures of AI implementation are documented; we will look at what has worked in the past and discuss how the past can be leveraged in the future.

AI is treated as an entry point to A-Life; there is a place for both in many gaming genres, but discussing A-Life without AI is bound to leave out some useful techniques. In "Uses for Artificial Life in Video Games" (Chapter 3), you will realize that most games will necessarily encompass a mixture of AI and A-Life. This chapter also shows where A-Life can be implemented in the video game development paradigm, and to what ends. The key is to understand its potential before trying to decide where A-Life can be used.

"The A-Life Programming Paradigm" (Chapter 4) dissects A-Life and shows how the various, existing theories can be applied to implement A-Life in video games. This requires a slight change in the way that behavioral modeling and decision making are approached. Traditional AI prescribes where A-Life will allow behavior to emerge.

Part of this understanding requires that the essential A-Life building blocks are present in the video game. Chapter 5, "Building Blocks," will take you through some key fundamentals relating to implementation of A-Life algorithms coupled with traditional AI. In this way, we can marshal the power that A-Life offers.

Part of this power is brought to bear in Chapter 6, "The Power of Emergent Behavior." In-game characters are given the freedom to have their behavior emerge, rather than be restricted to prescribed patterns of behavior that can be easily spotted and subverted. Emergent behavior is more than just the sum of its parts, however. It can be used at many different levels—from single-entity modeling to entire collections of in-game entities.

These collections can be used for the game proper but also have a role in "Testing with Artificial Life" (Chapter 7). Theoretical and practical advice is offered for testing systems using A-Life techniques. Video games, like all real-time systems, need extensive testing through modeling players' behaviors, and A-Life is an excellent substitute for real players when used appropriately. Development time is not wasted, either. In many cases, A-Life is used to model a real player that can be used as an in-game adversary. Chapter 7 will address how this can be achieved and give some specific examples.

We then extend these testing examples in Chapter 8, "Several A-Life Examples," which dissects some of the actual technology that has been used in games. The hope is that designers can borrow some of these techniques for their own games while discovering how the theories could evolve further.

"Multiplayer AI and A-Life" (Chapter 9) uses the previous material to show how AI and A-Life techniques can be employed in multiplayer gaming development. With the explosion of the Internet and addition of high-speed access, multiplayer and massively multiplayer game universes are rapidly becoming the norm. AI and A-Life can make multiplayer games run better and be easier to maintain, as well as offer some great in-game features. In some cases, their use can also help reduce the need for real humans to perform some of the high-level tasks associated with running a multiplayer game online.

However, many cutting-edge techniques remain in the informal laboratory. Even games such as *Creatures* were not considered mainstream. In Chapter 10, "The Application of A-Life Outside the Lab," we will examine how these novel, processor-sapping AI and A-Life technologies can be designed in, and not just bolting onto games.

After all, the goal of adding AI and A-Life is to make the game better. Exactly what "better" entails depends entirely on your point of view. AI and A-Life can help immersion, making a game more accessible. They can help make better enemies that are more capable, harder to combat, less fallible, more unpredictable, and generally more lifelike. For example, AI can be added to a car to help the driver (player) control it. The player's weapons can have AI built in, and A-Life can be used to manage the minions (entities) controlled by the player. An entire army can be governed by algorithms, making A-Life a formidable tool that the player can use to win the game. The same algorithms can be used to enable an army to learn from the player and adapt itself to the player's style.

There is also ample scope to get the A-Life painfully wrong. It does go wrong, and the results tend toward the disastrous. Part of the problem is that we are trying to give the game as much flexibility as the player has and, in doing so, we cannot be certain that it will not misbehave.

Much like the computer Joshua in David Bischoff's novel *WarGames*, some AI and A-Life applications in video game environments will lose their entities' leashes, enabling them to run a little wild. The danger is that once let off the leash, we cannot get them back under control. We can only trust that the necessary safeguards are in place to marshal deviant behavior.

For example, there is nothing worse for a player than believing the game is against them—or worse, cheating—and that their own trusted helpers (the car, army, or soccer team) are deliberately trying to turn the tide to their own advantage. The result of poor AI is unquestionably simple: the game will fail to convince the player but might feasibly be a commercial success thanks to other aspects of its design or IP.

Using the techniques presented in this book does not necessarily safeguard against getting AI and A-Life wrong on occasion, but it should help you understand where the pitfalls are, learn how to build in those safeguards, and develop correct AI and A-Life, as well as be in a position to correct any errors. After all, "intelligent," self-correcting code cannot be far behind, can it?

This page intentionally left blank

CHAPTER

1

INTRODUCTION

In This Chapter

- Defining Artificial Intelligence
- Knowledge Management
- AI Put Simply
- Defining Artificial Life
- Top-Down versus Bottom-Up Intelligence

This chapter will broadly introduce the concepts of artificial intelligence and artificial life, in particular, as applied to video games. There are some general concepts to be aware of before trying to tackle the rest of the book, and we cover these in this opening chapter.

AI and A-Life can be embodied by a quote from David Bischoff's novel *WarGames*, in which one of the characters attempts to explain how AI leads to a decision-making process.

> *"If you put your foot in the fire, and it got burned, would you put your hand in?"*

It illustrates the point that, in a system of connected parts, such as the human body, we can use experience in one area to predict the outcome of a similar event in another area. In the case of the character in the novel, a connection between two specific games is being explained—neither of which can be won.

The inevitable outcome of one is stalemate, and the logical conclusion that Bischoff is trying to get across is that, in a similar game with higher stakes, stalemate is still inevitable. Another choice quote from the same book reads:

> *"The only winning move is not to play."*

The context of these two quotes gives us the basis for both AI and A-Life. On the one hand, we use AI in games to direct behavior, while on the other hand, sometimes A-Life can produce solutions that AI, by itself, cannot achieve.

Traditional AI systems (state machines, planning systems, and so on) direct behavior in an explicit way. It is only when we augment these static systems with holistic and heuristic ones that A-Life begins to replace pure AI. This results in emergent solutions that often cannot be achieved with pure AI techniques as used in video games currently on the market and in development.

It is possible to avoid A-Life completely and tie many AI systems together, as can be seen in many "modern" video games, but the result is a limited emergence, which can be every bit as predictable, over time, as traditional AI solutions.

It is only when we look at each in turn that we can decide which one is most appropriate for a given game universe. This chapter and the next two should help you begin to make "intelligent" choices for a game project you might have in mind.

Defining Artificial Intelligence

For our purposes, a loose, informal definition of artificial intelligence will suffice. After all, this book is a study of the application of AI and A-Life in video game design, development, and testing—not a discourse on AI and A-Life as scientific fields. However, remember that they are just that—scientific concepts that imply two things:

- They are incomplete, and
- Advances are still being made.

In other words, as this book is being written, researchers are uncovering new AI techniques and A-Life paradigms. The rudiments of AI may have been worked out and documented, but these are surely just the tip of the iceberg. A-Life is still a field in its infancy, and there are sure to be leaps and bounds made as machines become more capable.

Put all this together, and we begin to realize that not only is a formal scientific definition unnecessary for our purposes, it is also incomplete. Hence, an informal definition will be more than sufficient; we could label it an *applied* definition that will take into account how AI will be used, rather than being an all-encompassing theoretical explanation of what AI is.

Artificial Intelligence as a Reasoning System

One aspect of AI is that it makes decisions based on information from existing models. It is a decision-making mechanism that might result in action or inaction and will be based on the presence or lack of information of some kind.

So, we could look at an AI module as something that receives an input, analyzes that input, and delivers an output. The analysis of the input is the "intelligent" aspect of the system. Rather than delivering a prescribed output blindly, without regard to any input, we create a model that attempts to map one to the other (output to input) within the confines of the problem domain.

In simple video game terms, this is like creating a game in which the player shoots upward at marauding aliens. The aliens can also bomb back—and in one of two modes, via a:

- Prescribed pattern, or
- In-game knowledge-driven decision architectures.

In the prescribed pattern mode, the aliens drop bombs on the player at regular intervals—say, every 10 seconds—regardless of the player's relative location. The mode using in-game knowledge-driven decision making will drop a bomb at a regular interval *if* the player is about to move into a position where the bomb has a chance of hitting him.

The latter solution uses in-game information and applies it to supply a more natural solution.

The model for deciding whether a bomb should be dropped or not has to take many variables into consideration—the positions of other aliens, the player, other bombs, closeness of the player, self-preservation, and so on. The more sophisticated the model, the more "intelligent" we say that it is.

So, in this case, AI uses information in a specific model to arrive at a decision. The subsequent action results from the decision-making process that uses observation of the state of play as the input information. The model is prescribed, but flexibility is allowed in the decision-making process.

If we want to model a different behavioral process, we need to create a different model with its own input information and output actions. We can implement a

movement model, for example, that manages the position of the alien in relation to other aliens, the player's craft, and any other bombs/bullets in play.

In essence, the model creation is a shortcut in the AI process. We substitute a solution (model) of our own, rather than provide the system with the ability to work out a solution for itself—a system known as "universal AI," where information (data) is acquired and applied to different situations. Universal AI, therefore, is not within our grasp for applications such as video games, if at all. But certainly, with the processing power available in current equipment, intelligent models can be used to enable the system to exhibit intelligent behavior.

These models are therefore prescribed, and the final behavior will be deterministic, and therefore somewhat predictable, even if it presents a challenge to the player. In our aliens example, the player will quickly recognize the model being used and adapt his own model of "dive and fire" to counteract it while avoiding any bombs being dropped.

The (human) player is equipped with problem-solving capabilities that enable him to apply previous knowledge to new situations and come up with simple solutions.

A deterministic model, which is based on rules, is created by one person (the game designer) and is solved by another person. In other words, the AI implementation is the product of one mental model and can therefore be solved by another, once the key is found.

In this kind of approach to AI, people create models to mimic intelligent (in this case, human) behavior by analyzing a specific problem. Just as we took the alien-bombing model through several steps to arrive at a solution (drop a bomb when over the player), video game designers and programmers will similarly create models (algorithms) that process system information in order to solve in-game problems.

Each model is represented by an algorithm in the resulting AI implementation. Programmers may talk in terms of an algorithm that represents the model that has been arrived at as a description of behavior that should manifest itself in the game.

The transition of the human in the game (the designer) to the alien dropping bombs is what results in the model that is used as a template for the alien behavior. This is an example of solving an in-game problem (albeit a simple one) using human behavior that is mapped in to the game space. This can be extended by applying universal AI.

One of the features of universal AI that can make the behavior more complex is the ability to adapt those models. If the game is intelligent enough to notice that the player has found a way to counteract the model in use and change its behavior accordingly, we can call this "real intelligence." A model that represents this new behavior might be dubbed "real AI."

Our first model dropped a bomb only when the player was directly below the alien; but this, as was noted, is easily counteracted. If, though, the alien notes that the closer it is to the player, the greater the chance that the player is hit, it might adapt its reactive model to be more proactive. This might lead the alien to move closer to the player when the player's craft approaches the line of fire. To arrive at this new behavioral model, the alien needs to know that it can move downward as

well as from side to side. It also needs some information regarding range as well as location—if the player's craft is directly below or if it is moving toward the alien or away from it.

This information needs to be gathered together and a decision made—move left, right, down, fire, and so on—but these calculations are not strictly binary. The original model was based on a simple yes/no response. Is the player directly below? If yes, then fire. If not, then do not fire.

We can build in adaptability that considers data that is not based on binary decisions, and we can create algorithms to manage the models being using to exhibit intelligent behavior. Although it might look like universal AI, this will result in models and behaviors that are still applicable only in one problem domain—the video game.

The result is that the algorithms that represent the models become part of a layered system in which new algorithms must be found to manage the application of these lower-level behavioral abstractions. It becomes a hierarchical system that mimics the way that behavior has been observed in real organisms.

So, AI in this case will be the application of models to a given problem domain. To a certain extent it is still predictable, and the player will discover ways to subvert the so-called intelligent behavior of the aliens, but it is as close as we can often get to in-game intelligence.

As we shall see later in this book, A-Life might provide a quick answer to this problem. Some algorithms that we can devise will allow for this kind of model adaptability to present a more natural, less predictable, and sometimes more intelligent model that the player will find harder to combat or more natural to interact with.

Appearance versus Actual Intelligence

Whether adaptable or not, some models are so sophisticated that it becomes almost impossible to tell whether the object on the other side of the interface is intelligent or not. The Turing Test, devised by Alan Turing in 1950, is an experiment to test the capability of a machine to exhibit intelligence. One embodiment of the Turing Test requires that a series of judges interrogate, via a keyboard and screen, another party, without knowing if the other party is human or machine.

It is the job of each judge to try and work out, within a specific subject area, whether the other party is human or a program designed to act like a human. If the judges are fooled, then the computer is said to have passed the Turing Test; that is, the judges cannot tell for sure whether the other party was a machine or a human.

However, take that system and put it into a different domain, and it will quickly become apparent that it is not intelligent. The appearance of intelligence is often enough for us to decide that the behavior exhibited is intelligent. We cannot tell the difference; so we happily accept that the being is possibly human.

One such case in game design mythology is related by Richard Garriott, manager of the *Ultima Online* project. He was walking around the newly created *Ultima* world in his guise as Lord British, when he came across an NPC (non-player character) going about its prescribed business.

For a few moments, Garriott engaged the NPC in conversation and could not tell from the first few questions whether it was human or machine. The fact that they were both characters talking through a text interface, and could both have been human, helped create the illusion that the NPC was really another player or another member of the design team. It was only the NPC's inability to relate exactly what part of *Ultima* it was working on that led Richard Garriott to conclude he was conversing with an algorithm rather than a human. However, he was fooled, if only for a moment.

Similar implementations of conversation algorithms, such as the famous Eliza, just follow prescribed patterns of interaction. They take input from the user (player), check for key words that they can respond to or rearrange the user's input, and provide output. Eliza, for those readers who have not come across this piece of AI software, was a chatbot modeled after a therapist. Through a series of questions and clever reversal of the human players' responses, it was capable of carrying out a limited but reasonably authentic conversation. It failed, however, in the Turing Test, to convince the judges that it was not a machine because the AI mechanisms implemented followed a static model that could not deal with new situations and rigidly stuck to the preprogrammed game plan devised by its human creator.

Certain prepared phrases can also be used as an attempt to guide the user through the conversation in an intelligent way. For example, if Eliza encounters a phrase that it does not understand, it is likely to counter with a phrase that forestalls decision-making and solicits additional, perhaps understandable information this time:

"Can you tell me why you …?"

If the answer does not help, then Eliza might respond with:

"Please go on."

The mixture of phrase-spinning and prompt statements, combined with an ability to parse the input into the appropriate parts of speech, gives the appearance of intelligence. While it is unlikely that Eliza would pass the Turing Test, for video game use and within a restricted environment, this level of intelligence is more than suitable for providing a sense of immersion for a willing player.

The same kinds of abstract theoretical behavioral models can have multiple uses, as we shall see later on. An Eliza-style speech-processing algorithm is an implementation of a challenge-response model. This model can be adapted to, say, fighting games, where question and answer become attack and defense, combined with renewed attacks based on some of the player's own moves.

The algorithms will be different in each case, because they address very different problem spaces, but they share some commonalities in the abstract model that the algorithms seek to represent in concrete terms. In other words, starting with an abstract model, multiple uses can be foreseen, in multiple problem spaces. The

reuse of the theory and models can only help a game studio create better AI and A-Life for future titles.

As in the Turing Test, the goal is to fool all the people all of the time. Video game players are easy marks—they want to believe. Only the rudiments of intelligence are sometimes required for them to feel as if they really are up against a clever opponent.

Video game designers must remember at all times that the limits on hardware mean that we need to leverage as little intelligence as we can get away with. In other words, the amount of intelligence that we design into a game has to be directly related to the game genre. The amount of processing power available to AI routines is dependent on the needs of other game aspects, like graphics, sound, and game universe management routines.

However, note that some of these can be augmented by the use of AI and A-Life and use less processing power in the bargain. Actually, a mixture of AI and A-Life will probably be needed—something we will look into as we progress.

The reapplication of AI models to make adaptable musical modes that use feedback from the current game status representation and player's (or enemies') movement to alter the music is one example. Another might be to use A-Life techniques to "breed" visual elements that look natural and can be processed at a reduction in processing power.

In theory, at least, it ought to be possible to take an innovative shading engine like that used to render high polygon visual models from low polygon representations (as in the CryEngine) and augment it to provide consistently natural results. The basic low resolution polygon model becomes just a template onto which the features of in-game characters can be rendered, and AI/A-Life techniques used to create the actual rendering deviations from some standard model.

To recap, our first tentative definition of intelligence is the application of knowledge within a system, using a model to arrive at decisions that exhibit intelligent behavior. This is generally good enough for video game applications. The Eliza program and Turing Test operate on the principle of the application of knowledge within a system. This knowledge is acquired through interaction (the Eliza "conversation") or from part of a database of stored knowledge.

The game 20 Questions is another example of applying acquired knowledge through interaction and stored knowledge. Questions are asked, with the answers narrowing down the options, eventually revealing the target object. It is a simple enough game for a human, as long as we know what questions to ask. For a machine, we can build a model that helps in asking the right questions and store (temporarily) enough information to enable the computer to arrive at a possible answer. Both humans and machines stand a good chance of getting a right answer—and both can also make mistakes.

Knowledge in a video game can take many forms. There is the game universe and the objects contained within it, the game rules, the actions of other beings, and the value of objects within the game universe. Each being in the game universe has to be able to work with that knowledge in order to act and react within the confines of the