

ANNOTATED WORLDS FOR ANIMATE CHARACTERS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Patrick Owen Doyle

March 2004

© Copyright by Patrick Owen Doyle 2004
All Rights Reserved

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Richard Fikes
(Principal Adviser)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Barbara Hayes-Roth

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Terry Winograd

Approved for the University Committee on Graduate Studies.

Abstract

Autonomous intelligent agents designed to appear as convincing “animate” characters are presently quite limited. How useful and how believable they are depends on the extent to which they are integrated into a specific environment for a particular set of tasks. This tight integration of character and context is an inadequate approach in the rapidly-expanding realm of large, diverse, structurally homogeneous but disparate virtual environments, of which the Web is only the most obvious example. By separating a character’s core features, which are invariable, from its knowledge of and abilities in any specific environment, and embedding this latter information in the environment itself, we can significantly improve the character’s believability, its utility, and its reusability across a variety of domains.

This thesis describes the requirements for an agent architecture that can interact with an annotated virtual environment, together with languages for representing information in and about these environments. An example applying this approach to intelligent animate characters is presented and used in two substantially different environments, the Web and a multi-user virtual world.

Author's Note

This thesis came about primarily because I was disheartened by the low level of interactivity and apparent intelligence (social, emotional, functional or otherwise) exhibited by the characters in every large multi-user environment I have encountered. Large, lengthy, and expensive efforts are being made to build massive virtual worlds where people can come together to learn, to socialize, and to play. They are often beautiful; they offer many places to visit, many things to see and do, activities and challenges for the visitors. However, they are populated by a professional cast (so to speak) of characters only slightly more interactive than furniture, and I don't mean particularly smart furniture.

The general trend in computer entertainment over the last several years has been to focus on improving the often spotty intelligence of the computer (as environment, as opponent, as assistant). This trend has not has significant effect in the massively multi-player worlds because of the very limited computational resources that can be devoted to any particular character and the difficulty of producing autonomous believable characters at all.

My belief is that embedding knowledge in the environment is the only way we are going to be able to take many characters with limited resources in large environments and get them to behave believably with respect to the world. This offers the potential of much more believable characters, and much more capable characters as well. They shouldn't just be shopkeepers and passers-by, milling around and making smalltalk. They should be able to participate actively in the life of the world and the experiences of the visitors as friends, playmates, assistants, guides, leaders, opponents, and even just more colorful local color. Hopefully the ideas in this dissertation will help point

the way to getting there.

Acknowledgments

The production of a doctoral dissertation is a demanding, time-consuming, and draining process. I have always been keenly aware that it is also fundamentally a collaborative process: I have succeeded in large measure because of those who have supported and encouraged me on this path, emotionally, socially, and academically. These few brief remarks are an inadequate recompense for their efforts, but luckily neither my expressions nor my feelings of gratitude are bounded by these pages.

First to thank is Barbara Hayes-Roth, my advisor for most of the years I spent at Stanford. She made me welcome here at the university and in her research group, and while freely sharing her considerable experience still actively encouraged me to find my own path. Her casual confidence in my abilities was and is something for which I am especially thankful.

To Richard Fikes, who has been my advisor these last many months, I owe many thanks. His willingness to take me on as a student, despite the very tenuous connection between his research and my own, was no less surprising than the enthusiasm he has displayed during our many productive and enjoyable conversations. The refinements he has contributed to my work, to my thinking, and to its elaboration here, are numerable and wonderfully skilled. I was lucky to be his student, even if for only for a brief time.

Terry Winograd graciously agreed to be on my committee despite the considerable demands on his time. To him I owe my exposure to the wide-ranging and fascinating field of human-computer interaction, which has had an enormous influence on this dissertation and on all my professional creations. The quality of his research and his teaching will always be ideals for me. I am grateful for his incisive comments on my

research. He deserves thanks also for allowing me to join in on his research group's weekly lunches, and also for introducing me, in passing, to the works of Jorge Luis Borges, from which I have derived much pleasure.

The delightful, discursive, and highly informative conversations I've had with Pat Langley qualify him for an unofficial committee position at the very least. Pat's good humor and terrible puns are matched by his vast knowledge of artificial intelligence and I have benefited and suffered from them in turns.

Katherine Isbister has been my friend for several years, and I have had the good fortune to collaborate with her, both in research and in teaching. There is considerable value in knowing someone who has survived the Ph.D. process, done it recently enough that she still remembers what it felt like, and can offer encouragement that it is actually possible to make it through. I think her students at RPI are going to be very lucky.

To Karl Pflieger I owe thanks for advice, guidance, feedback on my ideas and my writing, technical support, and years of TV nights with the gang at his house. All hail Joss Whedon!

Together with Karl, Eyal Amir, Urszula Chajewska, and Pedrito Maynard-Zhang were friends and comrades-in-arms in the Ph.D. process. Our TSG meetings provided critical analysis, practical advice, and much-needed support. They had as much faith in me as I did in them, to my perpetual amazement.

Thanks also go to Heidy Maldonado, who despite being several years younger than I has been at Stanford even longer than I have, though in that time she's earned far more degrees. Heidy is one of the most well-connected researchers I know, and discussions with her about my work and how it relates to what others are doing have been frighteningly enlightening. Her enthusiasm has been simply frightening. To her I wish good luck as she completes her own Ph.D., which I suppose will be the first of several.

Many others in the computer science department deserve mention. Along with the TSG members, Avi Pfeffer and Lise Getoor and I studied for the qualifying exam together. To them and that process I owe whatever breadth of knowledge in AI I possess today. Many others in the department combined friendship with

scholarship and provided the social and intellectual atmosphere we all hoped for when we embarked on a university life: Lee Brownston (who also showed me where to find the culture in the Bay Area), Francois Guimbretiere, Ruth Huard, Danny Huber, Jessica Jenkins, Uri Lerner (I bow to his superior knowledge of *The Simpsons*), Tamara Munzner (and her superior knowledge of science fiction, not to mention her legendary parties), Todd Neller, Aarati Parmar, Daniel Rousseau, Scott Sanner (sadly, an expert on Jimmy Buffett), and Diane Tang especially.

For almost the entire time I've been at Stanford, I have worked at Concentric Network Corporation (later XO Communications). The balance of research and real-world development I did there kept me, I hope, from growing stale in either. Of my friends at XO, special thanks to Rob Field for his good humor and endless frustrating *Magic* games, and to Jeff Powell, who seemed almost as eager as I was to see me graduate.

I would like to thank Anson Lowe, Stanford professor and doctor, who in a literal sense kept me up and working during most of this process. How I got lucky enough to have a doctor who answers email, I don't know.

Professors Kevin Compton and Bert Herzog at the University of Michigan both encouraged me to come to Stanford and see how another university does its teaching and learning. Each man had a profound influence on my early exposure to graduate school, Kevin as my research advisor and Bert as the first professor for whom I was a teaching assistant. I am deeply grateful to them both.

My friends of many years, who knew me long before I came to Stanford and have endured nearly a decade of promises that I'm going to graduate "probably next year," deserve thanks for their friendship and their faith. Especially to Scott Dexter, who got his Ph.D., a professorship and tenure all while I was still working on my thesis; to Jeff Woodside, who got his M.D., got married, had a child, and moved to Texas (a state he deeply loves); to Bonnie Wallace, who has a special ability to distance herself geographically while still staying a close friend; and most of all to David Schairer, true friend since childhood, I give affectionate thanks.

My girlfriend Wendy has waited patiently for nearly two years for me to finish this document. She has given me support when I wanted it and has told me to buckle

down and get to work when I needed it. The sense of exhilaration at finishing the dissertation is tempered by the knowledge that I will finally have to take those swing dancing lessons.

Finally, to Mom and Dad, my sister Kate, and my brother Philip, my love and gratitude. With the help of all the people I've mentioned, and the many others that I haven't, I finished this journey; but without your love and encouragement, I never would have been able to begin it.

Dear friends, reproach me not for what I do,
Nor counsel me, nor pity me; nor say
That I am wearing half my life away
For bubble-work that only fools pursue.
And if my bubbles be too small for you,
Blow bigger than your own: the games we play
To fill the frittered minutes of a day,
Good glasses are to read the spirit through.

And whoso reads may get him some shrewd skill;
And some unprofitable scorn resign,
To praise the very thing that he deplores;
So, friends (dear friends), remember, if you will,
The shame I win for singing is all mine,
The gold I miss for dreaming is all yours.

— Edwin Arlington Robinson, “Dear Friends”

Contents

Abstract	v
Author's Note	vi
Acknowledgments	viii
1 Introduction	1
1.1 Animate characters	2
1.2 Virtual worlds	4
1.3 Many characters, many worlds	7
1.4 Two kinds of designers	9
1.5 Contributions	12
1.5.1 A framework for annotating virtual worlds	12
1.5.2 A character architecture that uses annotations	13
1.5.3 Example worlds and characters	14
1.6 The structure of the dissertation	14
2 Qualities of Animate Characters	17
2.1 Definitions of animate characters	21
2.2 Qualities of character	23
2.2.1 Personality	23
2.2.2 Individuality	25
2.2.3 Emotion	29
2.2.4 Social Relations	31

2.2.5	Expressiveness	32
2.2.6	Believability	33
2.3	Qualities of animacy	35
2.3.1	Goal-directedness	36
2.3.2	Reactivity	37
2.3.3	Situatedness	38
2.3.4	Broad capabilities	39
3	Enabling Better Character Behavior	41
3.1	Acting in the world	43
3.2	Knowledge in the world	44
3.3	Examples	48
4	The Annotation Framework	50
4.1	Design and drama	51
4.2	The design space of annotations	52
4.3	A taxonomy for characterizing annotations	55
4.3.1	The structure of annotations	56
4.3.2	The mechanics of annotation	63
4.4	Claims	64
5	An Animate Character Architecture	66
5.1	Overview of the architecture	67
5.2	Implementation details	70
5.3	Communicating with the environment	70
5.4	The context manager	73
5.4.1	Environment	73
5.4.2	Self	75
5.4.3	Relationships	75
5.5	The emotion system	76
5.5.1	The model of emotions	77
5.5.2	Emotions in the architecture	81

5.5.3	Emotion influences	86
5.5.4	Computation of emotions	88
5.5.5	Expressions	89
5.6	The behavior engine	92
5.6.1	Tasks	92
5.6.2	Behaviors	94
5.6.3	Actions	97
6	Analysis of the Architecture	99
6.1	Goals of our architecture	99
6.1.1	Simplicity	100
6.1.2	Domain independence	101
6.1.3	Believability	102
6.1.4	Annotation sensitivity	103
6.1.5	Building our own architecture	104
6.2	The approach to analysis	104
6.3	Qualities of character	106
6.3.1	Emotion	107
6.3.2	Social Relations	109
6.3.3	Expressiveness	109
6.3.4	Personality	111
6.3.5	Individuality	111
6.3.6	Believability	112
6.4	Qualities of agency	113
6.4.1	Goal-directedness	113
6.4.2	Reactivity	114
6.4.3	Situatedness	115
6.4.4	Broad capabilities	116
7	A Guide for the Web	117
7.1	Motivations	118
7.1.1	(Not) communicating	119

7.1.2	How character interactions help	119
7.2	The interface	121
7.2.1	Communications between the character and the environment	122
7.2.2	Communications between the user and the character	124
7.3	The environment: an art gallery	125
7.4	The annotations	127
7.5	The character: Merlyn	131
7.5.1	Personality and emotions	132
7.5.2	Believability-enhancing behaviors	134
7.5.3	Functional behaviors	134
7.6	Lessons Learned	137
7.6.1	Challenges	140
8	Sidekick for an Adventure Game	142
8.1	Motivations	142
8.2	The environment: a text-based adventure game	145
8.2.1	Properties	147
8.2.2	The game setting	150
8.2.3	The environment's annotations	152
8.3	The character: Harker	153
8.3.1	Goals	154
8.3.2	Personality and emotions	154
8.4	Affective context	156
8.4.1	Distribution of knowledge	157
8.4.2	The annotations	158
8.4.3	Example: spelunking in dark caves	160
8.5	Puzzles	161
8.5.1	Distribution of knowledge	162
8.5.2	The annotations	165
8.5.3	Example: the wizard's workshop	168
8.6	Games	174

8.6.1	Distribution of knowledge	174
8.6.2	Categories of players	176
8.6.3	Applying these categories in Nim	178
8.6.4	The annotations	180
8.6.5	Example: playing a game of Nim	184
8.7	Lessons Learned	187
9	Conclusions	194
9.1	Summary of contributions	194
9.1.1	The qualities of animate characters	194
9.1.2	A framework for annotating virtual environments	195
9.1.3	An architecture for building animate characters	195
9.1.4	Evidence of success	196
9.2	Related Work	196
9.2.1	The Oz Project	196
9.2.2	The IntelliMedia Project	198
9.2.3	The Synthetic Characters Group	198
9.2.4	ELIZA and PARRY	199
9.2.5	Julia	199
9.2.6	Improv	200
9.2.7	Soar	200
9.2.8	The Sims	202
9.2.9	Other Work	204
9.3	Future directions	206
9.3.1	More extensive annotation vocabularies	206
9.3.2	Application to other character architectures	207
9.3.3	Experiment in existing virtual worlds	207
9.3.4	Annotations as character communications	207
9.3.5	Characters as annotated entities	208
9.3.6	Procedural annotations	209
9.4	Conclusions	210

A	The Adventure Game: Annotated Transcripts	212
A.1	Puzzles	212
A.2	Game-playing	220
B	The Web Gallery: Annotations	231
B.1	Elements of art vocabulary	231
B.2	Example of gallery annotations	235
	Bibliography	238

List of Tables

2.1	Qualities of animate characters	40
5.1	Entities and the ways in which they are appraised	80
5.2	Emotions used by the architecture	82
6.1	Qualities of animate characters (reorganized)	107
8.1	Character goals referenced in affective annotations	159
8.2	Notable annotations in puzzle description	168
8.3	Notable annotations in game description	184
B.1	Major elements in the art gallery vocabulary	234

List of Figures

1.1	Virtual worlds	6
3.1	Knowledge in the world	45
5.1	The animate character architecture	67
5.2	An annotation communicated by the MUD environment	72
5.3	An object in the environment that is a source of emotions	87
5.4	Translation of fear influences to fear emotion for heroic character . . .	90
5.5	Translation of fear influences to fear emotion for timid character . . .	90
5.6	Hierarchy of the behavior engine: tasks, behaviors, and actions	92
5.7	Fitness of several behaviors in terms of reproach emotion	95
5.8	The main sense-think-act cycle of the architecture	98
7.1	Components of the interface	123
7.2	The guide control panel	125
7.3	Merlyn in the museum	126
7.4	Merlyn telling a story	132
8.1	A sample interaction in a MUD	146
8.2	Main floor of manor house (used in Section 8.6.5)	151
8.3	Other floors of manor house (used in Section 8.5.3)	152
8.4	Harker reacts emotionally to environment	161
8.5	Harker helps to solve some puzzles	171
8.6	Harker plays a game of Nim	186

9.1 A screenshot from *The Sims* 202

List of Listings

7.1	Extract from art gallery annotation	129
8.1	Annotation describing affective location	160
8.2	Annotation describing guardian puzzle	165
8.3	Annotation describing Nim game	180
A.1	Annotations as Harker helps defeat guardian	212
A.2	Annotations as Harker plays Nim	221
B.1	Annotation describing an artwork in the Web gallery	235

Chapter 1

Introduction

What we look for in a created character is not mere surprise but revelation. The unexpected behavior of a fictional character must be surprising in the way that human beings are surprising; it must tell us something we recognize as being true to life.

— Janet Murray, 1997

The real world is filled with characters. The clerk at the bookstore, the professor in the lecture hall, the coworker in the office, all have their own appearances, personalities, and stories. Our imaginary worlds are also filled with characters, characters from Don Quixote and Sherlock Holmes to Homer Simpson and Daffy Duck, characters that are a purer distillation of the qualities that we hate, love, admire, disdain, embrace, reject, that fascinate us and that compel us. And yet we are always one step removed from them. We cheer for them, but cannot encourage; we sympathize with them, but cannot comfort; we love them but cannot tell them so. These characters interact with one another but they do not interact with us.

One of the great modern fantasies has been to imagine how to bring our characters to life in a more literal sense, and how to make it possible for us to interact with them. The “holodeck” of *Star Trek* is a futuristic device that does just that — it transforms an ordinary room into a perfect replica of any imaginary environment, and populates it with intelligent, interactive characters. It enables the observer to become an active participant in the story.

Obviously, we are not on the verge of installing home holodecks. We are seeing, however, first approximations to such imaginary technology in the form of massive virtual worlds in which players can embody themselves, explore and live, and interact with others, whether they are players or characters controlled by the computer. These worlds have benefited from more than two decades of research on the creation of intelligent, interactive characters and on the construction of compelling virtual environments.

Animate characters have not, as yet, been used effectively in virtual worlds in general. Research on characters has tended to focus either on a character operating outside of any environment, or on a character that is tightly coupled with one specific environment for which it was designed. However, with the growth in the size, complexity, and number of virtual worlds, the question of how to make characters that can effectively be used in these environments is increasingly important.

The central question of this thesis is, how can we effectively enable many characters of different capacities, goals, and designs, to behave believably in large, complex, and unfamiliar environments?

1.1 Animate characters

The goal of artificial intelligence in the last half century is to build a system that has at least the general reasoning abilities of an ordinary human being. A subgoal that has arisen in the last twenty years, informed by and shared with art, psychology, sociology, drama, and literature, is to build something that is both more and less: an animate character, a creature with all the qualities that have fascinated and engaged us in dramatic characters for millennia, but which can interact with us as well.

Research on interactive, intelligent characters has been going on in earnest since the early 1990's. With the Oz Project, Bates and his students ambitiously attempted to make computational versions of all aspects of a dramatic experience — the characters, the environment, and the director [13, 88, 144]. More recent work, while more sophisticated, has tended to specialize, whether on building characters for particular tasks (e.g. teaching [69, 85], assistance with routine work [10, 63], sales [18], free-form

play with puppets [20, 56]), enhancing particular properties (such as speech [27], animation [9, 52], or models of emotion [38, 37, 47]), or developing underlying theories about what makes effective characters [14, 93, 98].

While animate characters are a kind of autonomous, intelligent agent, the goals of these agents are somewhat different. The most significant difference is that the evaluation criteria of animate characters are fundamentally *subjective*. For most intelligent agents, there are clear, measurable criteria for successful behavior. Did the agent successfully perform the required operations? How long did it take? What resources did it consume?

On the other hand, the principal goal of an animate character is to appear believably “alive” to a viewer. Such an evaluation depends heavily on who the viewer is, the viewer’s knowledge of the agent, the context of the agent’s activities, etc., etc. An apparently-drunken character, for example, may be incapable of following even the simplest instructions, but this does not necessarily make it a failure *as a character*, merely as an instruction-following agent. A character that does nothing but sit in a chair and snore, oblivious to the world around it, may be entirely successful while having little or no intelligence behind it. Joseph Weizenbaum’s ELIZA [143] fell into this category: it had virtually no intelligence, but was extraordinarily effective at convincing human beings that it was a functioning psychotherapist.

On yet another hand (we have many), human beings are very sensitive to behaviors that are not lifelike. Coleridge wrote that our acceptance of art requires the “willing suspension of disbelief” — that we are actively, if not necessarily consciously, relaxing the criteria we apply in evaluating what we are experiencing. This suspension, however, is quite fragile, and it is easily shattered, especially by interactive systems. We may have high expectations of accurate performance from a cleaning robot, for example, but we would not be especially surprised if it did not react to the sound of a loud noise. On the other hand, the vacuous character asleep in a chair had better react, or else our suspension of disbelief will be instantly shattered.¹

¹Of course, it is possible to manipulate our expectations by presenting us with characters whose properties confound or confuse them. Floyd, the famous robot from the Infocom game *Planetfall*, was such a character. It was patently a worker robot, but its speech and actions were those of an enthusiastic and playful young child. Thus it was possible to ascribe lapses in performance to its

Put another way, an effective animate character must possess many of the properties of a traditional intelligent agent while at the same time acting in a way that is perceived as believable. It is, in some sense, a combination of a traditional intelligent agent with a traditional dramatic character from art, film or literature.

This imposes significant difficulties. Creating either intelligent agents or believable dramatic characters is no easy task; creating an entity with the properties of both is doubly difficult, scientifically and artistically. The fact that such an animate character must be interactive, at least with its environment and presumably in most cases with human beings as well, makes it all the more problematic. The character must respond believably to what is happening in the world, but it is not possible for a designer to encode recognition of any arbitrary event in the world, nor all reasonable responses.

So far, these problems have generally been addressed by significantly circumscribing either the domain of the interaction, the perceived capabilities of the character, or both. A character in the form of a dog who is not expected to do anything but bark and wag its tail can ignore most events in its environment and most actions on the part of the user without seriously compromising its believability. A character that is fixed in one location, never leaving it and never being exposed to any unexpected perceptual information, can be made to seem highly intelligent and competent with respect to everything it encounters². Enabling characters to act believably across a range of circumstances and environments is going to be an even greater challenge.

1.2 Virtual worlds

A “virtual world” is a computer-based analog of a physical environment. Generally, a user is embodied within the world, and can use that body (called an avatar) to move around in the world and to interact with its contents. Most of these environments

personality, and bizarre fluctuations in its personality to its “robotlike” nature. In fact Floyd was not an intelligent character in the AI sense of the word, but still serves as an excellent example for this issue — see [98] for more.

²The infamous Microsoft Office paper clip character Clippy combines both of these properties; its non-human nature lowers expectations about its capabilities and about its behaviors, while limiting its interaction to popping up with suggestions in circumstances it recognizes or to responding to typed queries makes it seem reasonably intelligent, if annoying

support many users simultaneously, so they are not only physical but social spaces [45, 139].

While these virtual environments have been used for scientific modeling, education, training, and various other purposes, the mainstream domain in which they have received the most attention is, not surprisingly, entertainment. The first of the Multi-User Dungeons (later called Domains) was constructed by Richard Bartle and Roy Trubshaw in the late 1970's [11]. These were text-based environments; all of the contents, actions, and interactions were described in text rather than graphically. Nonetheless, they permitted a wide variety of sophisticated action, and the original MUD spawned many hundreds of subsequent MUD worlds, ranging from medieval dungeons to futuristic space stations to educational environments, including an explorable, interactive model of the human body.

As home computers became more powerful, and technology improved, it was perhaps inevitable that graphics would replace text. What are now called massively multi-player online role-playing games (MMORPGs) evolved in the mid-1980's with games like *Island of Kesmai*, but the current crop of modern games began in the mid-1990's with *Ultima Online*, and has continued with others such as *Everquest*, *Asheron's Call*, *Star Wars Galaxies* — there are dozens in existence now, with millions of players worldwide [72]. These worlds can support thousands of users simultaneously, and are vast in scope.

Figure 1.1³ shows screenshots from several popular contemporary virtual worlds. While the quality of the graphics and the perspective of the representations differ somewhat, they are essentially the same. The goal is to offer many users the opportunity to interact with one another in interesting environments, generally also offering them a range of activities that are not possible in the real world.

Contemporary large-scale virtual worlds can be roughly divided into two categories. First are the purely social worlds. They exist for people to get together and interact with one another, rather than with computer-controlled characters. These worlds are more like virtual playhouses than games; that is, they place emphasis on social interaction and social play, as opposed to goal-oriented interaction with the

³Clockwise from upper left: *There*, *Ultima Online*, *The Sims Online*, *Asheron's Call*



Figure 1.1: Virtual worlds

environment. *The Sims Online* [7] and the more recent *There* [62] are examples of these social worlds.

The second, and far the more common, are the role-playing game worlds. Their origin is in the “pen-and-paper” role-playing games, the most famous of which is *Dungeons & Dragons*. These free-form games allowed players to create characters whom they controlled, and to guide those characters through adventures in fantasy worlds. The game consisted of a set of rules describing how to resolve certain interactions (for example, whether your barbarian’s battleaxe hit the wizard before he could rain fire on the village). Small groups of players could get together to role-play in an adventure, while a GM (game master) described the world, played the roles of all its inhabitants, and refereed the interactions.

The massively multi-player role-playing worlds are environments in which the computer plays the role of the game master. Rather than relying on verbal descriptions and rolling dice to decide what happens, the computer can present a graphical representation of the world, complete with dungeons, spaceships, dragons, tanks, and towns. It also controls the so-called “non-player characters”, the apparently living, breathing inhabitants of these worlds that are not actually being run by players. These worlds would be empty and not terribly interesting without companions, assistants, townsfolk, villains, marauders, dragons, orcs, space slugs, and the like. They provide services, give aid, offer challenges, and generally give a social context for the game and provide most of the motivation for action in the game world. It is with this latter group that we are concerned in this dissertation.

There is a substantively different kind of networked construct that can be thought of as a virtual world, and that is the World Wide Web. While the topology of the Web is not intended to be thought of as representing an environment, and Web pages are not fundamentally interactive, it is still possible to imagine interactions with other people or with animate characters that use the content of the Web as the setting. While the Web is not as complex an environment, its size and number of users make it worthwhile to include it in our discussion. Chapter 7 focuses on how an animate character can interact with a user on the Web.

1.3 Many characters, many worlds

This brings us to the issue with which the dissertation is concerned. There are many large, complex virtual worlds. Most are not only mechanical but social simulations as well. In order for these worlds to achieve their design goals, they need to be peopled with a large number of intelligent, animate characters to play various roles not filled by the humans who visit them. Some of these characters may be created by the designers of these environments, and be controlled by systems within the world. We may also want to permit characters to connect in the same ways that humans do, with access to the same (or similar) perceptual information and with the ability to act in the ways that humans act.

From the point of view of the human visitors to these worlds, any characters they encounter should behave believably. A cornerstone of “behaving believably” is behaving in recognizably lifelike and often apparently intelligent ways with respect to the environment. Characters should appear to have a basic understanding of the structure of the world. They should react appropriately to the events that they experience. They should, depending on their roles, be capable of interacting with the world in interesting and nontrivial ways.

From the point of view of the worlds’ designers and that of the characters’ creators, these requirements have to be met in some practical way. It should be possible to expand and change the world without requiring significant changes to each character. Characters cannot be expected to have complete knowledge about a world they inhabit.

These are demanding requirements (to say the least). There is no autonomous agent with reasoning ability and knowledge approaching a human being’s; we cannot simply present an agent with a world and expect it to understand what is happening. And even if we did have agents with such vast capacities, that still would not always be enough. People in the real world often do not know how to behave. Sometimes appropriate behavior depends on specialized knowledge that we don’t possess before being confronted with a situation. To find the right gate at an unfamiliar airport, to react when we encounter someone we haven’t met before, to use devices too new or too complex, humans rely on cues and clues from the environment to tell us where to go, how to respond, and what to do.

The key observation and central idea of this dissertation is that this embedding of knowledge in the environment works for agents as well as for human beings. That is,

Distributing knowledge between the character and the environment permits animate characters that are more adaptable and behaviors that are more believable than we could achieve by placing all the knowledge in the character alone.

As we shall see, this idea has several compelling benefits. The designer of a world is the best suited to understand the kinds of reactions visiting characters should

have, or at least the kinds of reactions that they are *intended* to have. The designer can provide knowledge that it would be difficult or impossible for a character to get through pure reasoning alone, even if it has sufficient capacity to do so. Depending on their abilities, all agents have access to this embedded information, offering a consistency of action and reaction that would be problematic to produce otherwise. And this knowledge can be incorporated into the environment as it grows and changes, rather than requiring changes to each individual character to reflect changes in the world, meaning that a character should be able to move from one place to another in a single world, or from one world to another similar world, without becoming apparently disoriented and helpless.

1.4 Two kinds of designers

We have drawn a distinction between the designers of an environment and the designers of the characters that will inhabit it. Who do we imagine these groups are, and what is the relationship between them?

The text-based MUD environments were the most prominent and prevalent example of large, complex virtual worlds for most of the last 25 years. In most of these worlds, the inhabitants were *also* the world designers. You might play a character in an adventure-themed MUD, but at the same time have the ability to write code to create or modify aspects of the adventure, blurring the distinction between creator and player, as though being simultaneously in a play and in the world the play evokes.

As Howard Rheingold writes in *The Virtual Community* [115],

In MUDs...participants also create objects with powers, such as magic carpets that transport their owners to secret parts of the kingdom. Other participants can buy or steal those carpets; people can gain power to make even more useful carpets, but only after amassing sufficient knowledge of the MUDs' lore as well as a formal mastery of the MUDs' world-building languages and meeting certain challenges...

Gaining the power to modify the environment in which the game takes

place is a primary goal for newcomers in both the “adventure” and “social” MUDs....When you log off a MUD, many of the dwellings you have built, the cities you have constructed, the tools and toys and weapons you have created, can be explored or used by other people.

Indeed, some MUDs, such as the famous LambdaMOO⁴ at Xerox PARC, have no explicit game component at all, and are purely social environments in which people explore via their avatars, spend time with one another, and extend the environment, creating an enormous, ongoing, dynamic, collaborative imaginary world.

As might be expected, characters have been built to operate in these worlds. Some have been coded as interactive objects within the environment (e.g., [35]), while others are external programs that connect and communicate in the same way that an ordinary visitor would, and are superficially indistinguishable from them [45, 139].

These MUDs were and are largely free, volunteer-run systems. Their successors, which have arisen in the last decade, are the massively multi-player role-playing games, which are commercial systems, designed, developed and maintained by corporations for profit. One of the more significant ramifications of that distinction is that the for-profit systems do not offer the same level of control to their players. To be more specific, with the exception of *The Sims Online* [7], none of them permit their customers to program the environment in any way. The division between designer of the environment and player of the game is clear and absolute.

Even so, these pay-to-play worlds are trying to find ways to give players more control over the environment. Some months after it opened, *Ultima Online* [134] introduced the idea of purchasing “deeds” to houses, which could then be built and furnished with objects to a player’s taste. It is a far cry from creating your own flying carpet, but a step in that direction, coming with the recognition that the community of players is also a vast potential source of new content. It is precisely that support of dual roles, both creator and audience, that has made both the historical MUDs and the Web successes.

Another recent trend, even in single-player, non-networked games, is the creation of “mods,” or game modifications. An increasing number of the more complex games

⁴Available via telnet at `lambda.parc.xerox.com`.

provide tools that allow their players to generate new content which they can then distribute to one another. The role-playing game *Neverwinter Nights* [8], for example, actually had a fairly short single-player story but exceptionally elaborate tools for generating new graphics, objects, effects, and entire scenarios, of which thousands are now available for download from the Internet.

Some commercial games have taken the approach of publishing an API so that enthusiasts can build their own programs to communicate with the game, but without actually giving them the ability to control the game environment directly. *Doom*, *Descent*, and *Half-Life* are all examples of games that allow external programs to control characters operating inside the game, and which are actively being used in other AI research (e.g., [77, 78, 76]).

We will make our division between character and environment designers along those lines. That is, we assume that there is an API through which a character's designer can have that character connect to the environment and operate within it, and that that API will provide means for a character to perceive and communicate with the environment that might not be available to an ordinary user or player, but that the character's designer does not necessarily have the ability to make changes to the code that describes the environment. The designer of the environment can build it to support characters, can add knowledge accessible through the API that characters can make use of, but does not have control over the characters themselves.

Of course, the designers could be the same group. In the case of the MMORPGs, for example, the characters might actually be implemented within the code of the environment rather than being external programs that only connect to it. Or it may be that the character designers are enthusiasts, who have no connection to the people who built the game world. In any case, our core desire is the same. We wish to make it possible for someone who is adding to or modifying the environment to do so without having to take into consideration the properties or goals of any *specific* character, but only the kinds of knowledge needed by a general class of characters; and we want our character creators to be able to change the personalities, goals, and operational details of those characters without having to take into account the content of any particular environment. Only the API needs to be fixed.

This would permit, for example, researchers with their own utterly different character architectures to experiment with them and compare them in a single world. It would permit a character designer to experiment with that character in worlds that have very different content or players or design goals without having to reprogram the character, assuming that those worlds share the same basic structure. In this way, whether or not the characters are built by the same programmers who created the environment, there can still be many different characters moving and acting in a large and ever-changing, expanding world.

1.5 Contributions

The hypothesis is that we can achieve believable behaviors by distributing knowledge between a character and its environment. The character should possess understanding of the general cases, and the environment will supply information about specific situations. There are three major contributions of the thesis in exploring this hypothesis: a conceptual framework for organizing this information, an agent architecture for implementing these characters, and two worked examples of characters interacting in virtual worlds.

1.5.1 A framework for annotating virtual worlds

We use the word *annotation* to describe the kind of knowledge that is embedded in a virtual world for a character to use. In its broadest interpretation, it might consist of any sort of arbitrary data that could be helpful to an agent, and could be organized and presented in many different ways. However, that kind of description is not especially useful for a world or a character designer needing to understand how to create annotations. Therefore the first contribution of the dissertation is to offer a framework for thinking about and organizing annotations for characters.

The intent is to examine the kinds of knowledge that an animate character (as opposed to any kind of intelligent agent) would need in order to make itself more believable. Some aspects of believability have only to do with expressions of the

character's personality and emotions. Others have to do with demonstrations of the character's functional capabilities — that is, the character has to be able to *do* meaningful things in the world, and do them in ways that are consistent with that personality and those emotions. This knowledge needs to be broken up between the character and the environment. What does the environment need to contain? How general can its annotations be while still being useful to the character? What about characters with different capabilities, or different levels of sophistication at performing their tasks? Are there universal annotations that apply to all characters in any environment? We examine these issues and provide guidelines for authoring usable annotations.

1.5.2 A character architecture that uses annotations

In order for a character to make use of the annotation information embedded in the environment, it must be capable of perceiving them and of communicating with the environment about them. Agent architectures with these abilities may in other ways range from very sophisticated to quite simple, and the uses to which the information is put and the extent to which it is reasoned about may vary considerably.

Our interest is principally in the simpler agent systems. One of the arguments for the annotation approach is that it supports the production of believable behaviors by a large number of characters with very limited computational resources; that is, it scales well. To validate that argument, we have created a character architecture of our own design, one that is reasonably simple and yet capable of making use of annotations to exhibit complex and convincing behaviors.

The other reason we have provided our own architecture, in addition to the need for a system that is annotation-sensitive, is to explore some novel ideas about the ways in which believable behaviors can be generated. Chapter 2 provides a list of qualities characters need in order to be believable, and after Chapter 5 describes the architecture itself, Chapter 6 will examine the architecture's strengths and limitations in terms of those qualities.

1.5.3 Example worlds and characters

To demonstrate the annotation approach, we have built two different virtual environments, and for each we have created a character using our architecture.

The first environment is an art gallery on the Web. A collection of 19th century paintings is organized in a collection of “rooms” and “walls” through which an interested viewer may browse. Annotations describe the details of the artworks (e.g., artists, media, dates of construction, historical significance) as well as the structure of the site (areas, rooms, walls) and tours for exploring the site for visitors of various ages. The character, Merlyn, is an animated graphical agent whose purpose is to explore art galleries with children. As a child clicks through the gallery pages, Merlyn talks about the art, explains the contents, tells stories, plays games, comments with his opinions, and obeys simple commands given by the child.

The second environment is a text-based adventure game, implemented in a MUD world. The adventure game is set in and around the house of a magician to whom the player has come seeking advice. Annotations describe the contents of the locations in the game, their properties, the puzzles in the game and how they must be solved, together with hints on how to proceed, the emotional context of locations and events, and how to play games-within-the-game. The character, called Harker, plays the role of a sidekick and guide to the player. In addition to built-in behaviors for expressing his personality and for obeying certain orders, Harker uses the annotations to react to the environment, to play games, and above all to help solve the puzzles in the adventure.

Note that while the characters have knowledge about the *kinds* of environments they are exploring (art galleries, adventure games), they have no specific knowledge of the particular worlds they are visiting, which is, after all, the point of the annotations.

1.6 The structure of the dissertation

In this chapter we have introduced the central problem of the dissertation together with our approach: characters do not act in a vacuum, but must behave believably

with respect to the world around them; a practical means for achieving this goal is to enhance the environment so that it guides characters in choosing appropriate behaviors.

Chapter 2 details the qualities that an *animate character* must possess, first those of character, which are drawn from drama, art, and psychology, and then those of animacy, which are taken from traditional artificial intelligence work.

Chapter 3 motivates the concept of an *annotated environment*, explaining why this is a valuable idea from the points of view of HCI, psychology and software engineering.

Chapter 4 specifies the framework in which the annotations are built, from the broadly necessary qualities of an annotation to the functional roles they play in an environment.

Chapter 5 introduces the character architecture. The architecture has three fundamental components: the underlying mechanisms of agency, the annotation-integration mechanism, and the behavior engine that sequences the character's actions to produce believable behavior.

Chapter 6 is an examination of our sample architecture in terms of the qualities introduced in chapter 2. It discusses the ways in which the architecture was designed to exhibit those qualities, and how its limitations might be overcome.

Chapters 7 and 8 provide demonstrations of the annotation idea in two virtual environments. Chapter 7 shows how a character called Merlyn guides a user through a Web-based gallery of 19th century artworks. In Chapter 8, an assistant and sidekick called Harker helps a player in an adventure game that takes place in a text-based virtual world. These chapters describe how the characters' personalities were built, how their abilities enable them to operate in the general classes of environments that these specific examples represent, and the nature of the annotations embedded in these environments to facilitate those abilities.

Chapter 9 is a survey of research that is related to the ideas brought together in this dissertation. This includes work on character design from animation, art, drama, and literature, work on affordances and software engineering.

Chapter 10 concludes the dissertation by summarizing what has been presented, discussing the strengths and weaknesses of the annotation idea, and points at future

extensions of this work.

Chapter 2

Qualities of Animate Characters

Believability. That is what we were striving for...belief in the life of the characters.

— Chuck Jones, 1989

All fiction demands the suspension of disbelief; it requires the willingness to accept the reality of unreal situations, characters, and actions. Over the last century, the development of increasingly sophisticated media forms has given storytellers the power to create fictional worlds that seem as lifelike as real ones. But such worlds have been passive ones. The storyteller weaves a tale which is then distributed to an audience that silently watches or hears it. However, computers can now make *interactive* fiction possible. Users can participate in the action and the machine can tailor the world's responses to the user's choices and desires. To fulfill the promise of this technology, there must be interactive characters as believable as traditional scripted ones.

The problem of creating believable, dynamic characters that were nevertheless unreal — that had no physical existence — was first attacked by cartoon animators in the early decades of the 20th century. Their goal was, as ours is, to design worlds filled with actions that “evoke the observer's belief in the desires, the intentions, the volitions of the characters.” [137] To do this, they had to discover the essence of character, those aspects of behavior that had to be emphasized and those that could be ignored.

The most successful early work was done at the Walt Disney Studios [137]. From the silent *Alice* shorts of the early 1920's through the revolutionary *Snow White* in

1937 [141], Disney transformed animated cartoons from a new media form into an art form. Disney animators accomplished this largely by discovering through experimentation what animation techniques led to the construction of visually believable sequences of actions. These practices, such as making sure a character *anticipates* an action before performing it, or *squashing and stretching* an object to emphasize its elasticity, became the fundamental principles of Disney animation.

Interactive computer believability is now roughly in the situation of the wooden, iconic *Alice* tales of the early 1920's. A variety of work has been done in this area, ranging from believable textual dialog [143] to puppetry [56] and populated worlds, both text-based [13] and graphical [20], but so far success has been largely *ad hoc*. We are still in the process of learning the fundamental principles of interactive character design.

Some basic issues are already understood. In order for an observer to accept the plausibility of a character, the character must act in an intelligible and coherent way, and exhibit a personality that is consistent with its actions. The creation of such characters relies upon two bodies of work. The first is the understanding of effective *static* character presentation techniques, such as those embodied in the Disney principles. The second is research developing *dynamic* intelligent agent systems.

Because most traditional media creates a passive storytelling experience, the finished products are unique, fixed, static. They are scripted down to the smallest detail in each frame. The whole goal of the designer is to communicate a specific sequence of ideas in the most effective way, knowing how the story has contributed to the current action and how it will lead into succeeding ones. This allows every detail of presentation to be tailored to the demands of the moment. Presentation — the *form* an action takes — is everything.

Intelligent agents are generally designed to operate in dynamic worlds. They have incomplete information about their environments, limited time in which to act, and a limited set of actions to take. Effectiveness is measured by how appropriate an action the agent can choose to take, given its bounded resources. Rational choice of an effective action — its *function* — is paramount.

The art of building believable characters is the art of balancing the demands of form and function. By deciding upon a set of possible functions and a set of forms each function may take, a design space for behaviors is chosen. The effectiveness of the character depends upon its choosing behaviors that maximize its believability, while still performing necessary tasks.

Before we expand on this idea, a word about terminology. Ever since researchers began developing software programs that could give the appearance of intelligent living beings, a variety of terms have been offered to describe those programs. Believable agents, interactive characters, lifelike characters, synthetic characters, and embodied conversational agents have been suggested, among many others. The central idea behind these terms is that these programs combine two essential qualities: agency (in the artificial intelligence sense of the word), and character (in the dramatic sense of that word).

In this discussion we will use our own term, *animate characters* [57]. We choose this phrase for two reasons. The first is to avoid confounding it with any of the many other existing terms, each of which has its own history and associated preconceptions, and none of which is entirely compatible with our own. Second, and more importantly, we believe it best emphasizes the aforementioned essential qualities of active autonomy combined with individual, distinct personality.

Note that *animate* does not mean *animated*. The great animators of the Disney studios, for example, were concerned with rendering drawings of characters that would “appear to think and make decisions and act of their own volition.” [137] Animate characters, created through the use of artificial intelligence techniques, not only *appear* to make decisions and act, but in a functional sense, actually do. They are, as the term “animate” means, active, mobile — they are “full of life.”

In this chapter we describe the qualities that an animate character should possess. Just as there has been much debate over what to call these characters, there is still considerable discussion over the properties that define them. We are not attempting to resolve that debate here. This list of qualities should not be taken to be immutable, comprehensive, or even canonical. Since this dissertation does not argue that one approach to character creation is better than another, but is instead about how to

improve upon animate characters no matter what approach is used, our list is intended to be a reasonable synthesis of the various proposals that have arisen in the literature, and which can be applied to the whole range of animate characters being developed by us and by others.

Dramatic vs. Biological Inspirations: A Brief Aside

Before examining major definitions for animate characters, it is important to point out that there are two dominant attitudes toward the creation of these characters, the dramatic and the biological. The first, exemplified by the Oz Project at Carnegie-Mellon University [13] and the Virtual Theater Project at Stanford [60], takes the approach that our character designs should be driven by our understanding of what people actually *perceive* as lifelike, whether or not that is in any way actually like life. Wile E. Coyote and the Road Runner are successful characters, despite being entirely unlike real coyotes or roadrunners, because they nevertheless invoke in their observers a recognition of qualities we associate with living things (they exhibit emotion, they think, they plan, they have personalities, and so forth).

The other view, originating at MIT, is that we can build plausible lifelike agents by incorporating into them ethologically-inspired models of sensing and behavior. The MIT Media Lab ALIVE project [20, 21] created a virtual dog called Silas whose behaviors were driven by motivations and goals based on ideas taken from animal ethology. Another notable and effective system with biologically-modeled characters was the commercial simulation *Creatures*, whose imaginary animals were driven by models of genetics, biochemistry, neurobiology, and ecology [53].

Here we tend toward the dramatic view. The biological models are especially appropriate when attempting to simulate certain aspects of real behavior, but as we have observed, successful characters often have little to do with what is real. More to the point, the biological view is concerned primarily with the way in which the behaviors are generated, while the dramatic view is most concerned with the effects those behaviors have on the viewers or participants. We do not just want to build agents that can simulate some properties of animacy; we want to build *characters* that

provoke some very definite and recognizable effects on the humans who see them.

2.1 Definitions of animate characters

In order to motivate our list of qualities for animate characters, we first follow the approach of Franklin and Graesser [46] by examining the definitions suggested by the major research groups in the field.

Loyall writes: “Believable agents are personality-rich autonomous agents with the powerful properties of characters from the arts.” [87] A believable agent has personality, emotion, self-motivation, change, social relationships, consistency, and presents the illusion of life (appearance of goals, concurrent pursuit of goals, parallel action, reactive and responsive, situated, resource bounded, exist in a social context, broadly capable, well-integrated capabilities and behaviors).

According to Blumberg, “...an autonomous animated creature is an animated object capable of goal-directed and time-varying behavior.” [20] A creature must react, be seen as having an independent existence, have choices to make, reveal its intentionality, appear to care what happens to it, adapt, and display variability in movement and response.

Scott Neal Reilly writes that believable agents are “autonomous, interactive agents that have the qualities that have made the non-interactive characters of traditional media believable.” [113] They may not be intelligent or realistic, but they will have strong personalities.

Hayes-Roth and Doyle [57] claim that animate characters redefine traditional agent design problems. In addition to possessing empathy, personality, and a capacity for social relations, their behaviors must be variable rather than reliable, idiosyncratic instead of predictable, appropriate rather than correct, effective instead of complete, interesting rather than efficient, and distinctively individual as opposed to optimal.

Stone and Lester [132] describe animated pedagogical agents as possessing three key properties: timely domain coverage (that is, coverage of the educational topics), contextuality (appropriate explanations for the situation), and continuity (coherent behaviors, pedagogically and believably).

Perlin and Goldberg [108] are concerned with building believable characters “that respond to users and to each other in real-time, with consistent personalities, properly changing moods and without mechanical repetition, while always maintaining an author’s goals and intentions.”

Trappl and Petta describe synthetic characters as needing different abilities in different contexts; “animators might be assisted in the delicate and ephemeral task of ensuring consistent and believable patterns of behavior in yet other settings, achieving e.g. various degrees of agent autonomy can play an essential role in providing effective assistance to users...” [138]

In characterizing believable agents, Bates requires “only that they not be clearly stupid or unreal.” Such broad, shallow agents must “exhibit some signs of internal goals, reactivity, emotion, natural language ability, and knowledge of agents as well as of the micro-world.” [13]

For Foner [45], the critical issues are autonomy, personalizability, discourse (ability to carry on two-way dialog), risk/trust, domain (appropriate domain for the agent), graceful degradation, ability to cooperate, anthropomorphism, and choosing a setting in which users’ expectations for the agent can be met.

Considering these definitions, several themes emerge. First are the properties of traditional agents — autonomy, responsiveness, reactivity, situatedness, and goals. Next comes the ability to function socially; the agent should be able to carry on discourse, understand other agents, be able to cooperate, have a capacity for social relationships and social behaviors. Third is the character’s capacity for behaving believably. It should exhibit a consistent, strong, idiosyncratic personality, interesting variability, clear emotions, an appearance of motivation and goal-directedness, and should do nothing “clearly stupid or unreal.” [13] Fourth are its particular domain behaviors — the character should behave contextually, cover domain topics in a timely way (for a pedagogical agent), it should be effective, and have well-integrated capabilities with the environment.

From these themes we form our list of qualities that animate characters should possess. For other good lists of character requirements, see Elliott and Brzezinski [40], Hayes-Roth and Doyle [57], Blumberg [20] and especially Loyall [87].

2.2 Qualities of character

Whether the character is as simple as the Road Runner or as complex as Hamlet, we take pleasure when — and *only* when — even the surprises in a character’s behavior are causally related to its traits. [81]

An animate character is an unusual combination of art and science. In essence, it is simply a particular kind of autonomous agent (in the traditional artificial intelligence meaning of the phrase), which happens to have certain properties that aren’t ordinarily associated with those agents. However, the goals of building such a character are somewhat different than traditional AI goals. For one thing, the ultimate evaluations are always going to be at least partly subjective. And the sources of inspiration for their behaviors are primarily artistic rather than scientific. As Sengers [125] observes, the creation of animate characters is an artistic endeavor that happens to use artificial intelligence research as one of its tools.

To help clarify this distinction, we break the qualities of animate characters down into two groups. The first, qualities of character, focuses on those properties that are driven by artistic needs. The second, qualities of animacy, examine those qualities that stem from artificial intelligence. The combination defines the kind of thing we are concerned with, and are trying to create.

We begin with qualities of character, the foremost of which is personality.

2.2.1 Personality

Character *always* comes first, before the physical representation. Just as it is with all living things, including human beings. We are not what we look like. We are not even what we sound like. We are how we move; in other words, our personalities. [71]

A character, whether in a novel, a film, a cartoon, or as an autonomous computer program, is an impression created in the mind of an observer by the production of

some sequence of actions¹. Some information is provided (the character waves a pistol, cowers in fear, arches an eyebrow), but it is up to the observer to connect these actions into a coherent whole that describes the character. That whole is the character's *personality*.

Personality can be conceived in two ways. First is to regard personality as a set of inner traits that a character possesses which control its behavior. A character may, for example, be greedy, cold, friendly, timorous, and so on. This is the operational point of view we need to take when building characters. We create a set of behaviors and some preferences for the character to decide how to choose among them, and when the character is running it expresses those behaviors according to those preferences. That is its personality.

However, from the audience's point of view, personality is only *what a character expresses, how the character expresses it, and when it is expressed*. It does not matter what internal mechanism is being applied, it matters only what the end result is. For the audience, what the character is doing is all that there is, and personality is a model that the audience builds around those behaviors to make sense of the character. From the agent designer's point of view, personality leads to action; from the audience's point of view, personality is inferred from action. The most sophisticated animate character in the world, no matter how elaborate its models of personality, emotion, or animation, that does nothing but sits in a corner and mopes, is not going to convey a very deep or interesting personality to an audience. Ideally, behavior is the bridge that allows the audience to replicate the model created by the designer in their own minds.

These two views of personality are both reflected in traditional animation, in which the thinking is done for the agent by the animator. According to Lasseter [80], "Personality is the intelligent application of all the principles of animation." It was the original Disney animators who provided these basic principles [137], which include such rules as *exaggerate the essence of an idea via the design of the action*, *visually anticipate the action to prepare the audience for it*, and *overlap actions, so*

¹A note on terminology: Some authors use the word "action" and some use the word "behavior" to refer to what a character is doing; those terms are used interchangeably in this section, but in later chapters we will draw a distinction between them

one doesn't completely finish before the next begins. Again, the underlying key is that the actions must lead the audience to believe that it has a thought process. “Without a thought process, the actions of a character are just a series of unrelated motions. With a thought process to connect them, the actions bring a character to life.” [80]

Another of the Disney principles that is critical for character builders is *staging*, which means to clearly and unmistakably express a single idea at a time. The audience should not have to struggle to determine what the character's personality is; it should be unambiguously conveyed to us. Chuck Jones, the creator of Wile E. Coyote and the Road Runner, extended this principle with one of his own to the effect that *a character always follows certain rules for what it will and won't do.* These rules helped to guarantee that a character clearly expressed its own unique identity. We will find this idea helpful in later chapters.

For example, the Wile E. Coyote and Road Runner cartoons had a list of rules [71] that included:

- RULE 1. The road runner cannot harm the coyote except by going “Beep Beep!”
- RULE 3. The coyote could stop anytime — if he were not a fanatic.
- RULE 5. The road runner must stay on the road — otherwise, logically, he would not be called the road runner.
- RULE 9. The coyote is always more humiliated than harmed by his failures.

Of course, rules of this sort aren't sufficient to describe the character's personality. They say nothing about how the characters move or what they look like; they only provide the basic structure for action. In order to create an effective character, we need to understand what makes one personality different from another.

2.2.2 Individuality

Each character must be an individual, a unique, distinct personality. As Chuck Jones writes, “*it is the individual, the oddity, the peculiarity that counts...the only things*

worth watching in this or any other world are those that identify and overcome the ordinary.” [71]

Characters can be made individuals in many ways. They may have distinct physical attributes, such as height, weight, color, shape, or size. They may dress or speak differently. Primarily, however, individuality is expressed through the kinds of things the character does, the ways in which it does them, and the frequencies with which it does different things.

An obvious example is the title character of Robert Louis Stevenson’s *The Strange Case of Dr. Jekyll and Mr. Hyde* [131], the story of a scientist who drinks a potion that brings out the darker side of his personality. Dr. Jekyll’s behaviors are dispassionate, scientific, controlled, and beneficent; when turned into the evil Mr. Hyde, they become wild, cruel, and cunning. The same being is capable of all these actions, but the individuals Jekyll and Hyde exhibit entirely distinct personalities through their very different behaviors.

Consider also the seven dwarves from the Disney film *Snow White* [141]. Despite being physically similar, and having similar goals and roles to play in the film, they are each unique personalities. As Walt Disney describes Doc [137]:

Doc is upset by the least little thing, the least little annoyance. He doesn’t know where he is; somebody has to help him out — that is, as far as mental reaction or personality is concerned...He has this habit of getting mixed up, only when he is upset or when he is angered or when he meets Snow White.

while he says of Dopey,

His body should be longer and his legs shorter. You should have the feeling about him that he is wearing somebody’s cast-off coat...Dopey runs as fast as he can and gets ahead of the other dwarfs – turns around and looks back like a kid...He would hold on to his hat when running along with some fellows.

It is possible to give a character a strong personality without making it an individual. That is what a stereotype is. The cackling villain twirling his mustache, the

helpless, beautiful maiden, and the noble prince are all stereotypes. We know what they represent, but do not know who they are as individuals. If a character is to be truly believable, it must surpass those primitive qualities and offer something more distinctive and individual.

Signature, idiosyncratic, and ambiguous behaviors

Take Laurel and Hardy, for example; everybody can see Laurel doing certain things because they know Laurel. — Walt Disney, in [137]

As we have said above, an animate character's personality is apparent to the audience through what it does — through its behaviors. Many of a character's behaviors are likely to be shared by other characters in the environment. All the cowboys in a Western are apt to ride horses, shoot at things, and drink rotgut at the local saloon. For these behaviors, the question is not so much, *does the character exhibit this behavior?* but *how does the character do this in a unique way?*

Personality is also defined by *idiosyncratic* behaviors, which are behaviors common for the character, and *signature* behaviors, which are behaviors unique to the character. I have the the idiosyncrasy that I fiddle with my pen when I'm talking; it is a common behavior of mine. When I'm out walking, I frequently tap my pants pocket to make sure my keys are still there. Indiana Jones has a habit of tugging on his fedora. Hercule Poirot fussily adjusts all the objects around him so that they are neatly lined up. These are behaviors that a character frequently exhibits and which are a large, if unobtrusive, part of that character's routine existence. These help to make characters believable as individuals.

A signature behavior, on the other hand, is one that uniquely defines the character; it is something that that character and only that character does. Bugs Bunny is the only rabbit who says "What's up, Doc?" just as Ebenezer Scrooge says "Bah, humbug!" or Homer Simpson cries "D'oh!" As we observe, "signature behaviors... 'telegraph' to observers the key character qualities of an individual." [57]

Put another way, when you see a particular character, you are likely to see an idiosyncratic behavior, but if you see a signature behavior, you are certainly looking

at a particular character. It is often the case that idiosyncratic behaviors are also signature behaviors, though they clearly do not have to be.

A third type of behavior that is more useful for an animate character designer than an animator or writer is the *ambiguous* behavior. An animator can make an infinite variety of poses and actions for a single character; whatever can be imagined can be done. An animate character, on the other hand, is limited by whatever capabilities were built into it by the designer. Since it is only possible to build so many distinct behaviors into an animate character, it is advisable to have some set of behaviors that can be interpreted in different ways depending on the context in which they are seen. Mr. Spock of *Star Trek* arching his eyebrow is an ambiguous behavior; it is something that could be done for many reasons in many circumstances.

Ambiguous behaviors have the advantage that they can fit in many circumstances and, while conveying little actual information to the audience, can nevertheless give the audience the impression that something meaningful is taking place in the character's mind. By leaving the audience to decide what that is, such behaviors can appear both seamless and appropriate. However, they are rarely interesting, and they certainly violate the principles of clarity and individuality that are part of the bedrock of traditional animation.

To take an example, consider a scruffy, overweight man asleep in an overstuffed chair; let's call him Uncle Bob. As he sleeps, he snores, he twists, he mutters to himself. Occasionally he occasionally wakes long enough scratches an elbow or an ear. Such a character could be built with a minimum of reasoning ability; indeed, behaviors could be chosen nearly randomly. This could be a very believable character, simply because it conveys so little information. These are the things we expect a sleeping man to do. It does not make a particularly compelling character, however.

An interesting counterexample is the famous text-based computer therapist ELIZA, designed by Joseph Weizenbaum in the 1960's [143]. ELIZA was designed as a Rogeanian psychotherapist; in this approach, whatever the patient says is turned into a question. So, for example, if ELIZA's "patient" typed "I'm angry at my mother" ELIZA would ask, "Why are you angry at your mother?" The patient might respond, "Because my mother hates me," producing "Why does your mother hate you?" and

so on.

ELIZA was never intended to be as successful as a character; it was originally meant as a simple experiment in text recognition and parsing. The character had no real intelligence; essentially all it did was recognize the structure of a patient's sentence and reform it as a question — a purely parsing problem. However, much to Joseph Weizenbaum's surprise, it was not only effective as a character, but people were eager to use it as a psychotherapist to tell their troubles to.

Although ELIZA was appeared completely cognizant of the context, and Uncle Bob was entirely unaware of what was happening around him, both characters were effective for the same reason, namely, their contexts were so rigidly circumscribed that almost anything they did made sense. Another character that was successful by narrowly constraining its context was PARRY [29], who played the role of a paranoid schizophrenic.

This discussion emphasizes some of the problems posed to the animate character builder beyond those encountered by a traditional animator or an actor. We must not only create the personality of the character, but every aspect of its existence. An animator can easily make each frame unique from all the others, can emphasize just those elements of the personality or action that are important. An actor has a wealth of knowledge about motion, voice, posture, and human behavior to draw upon when acting. When creating animate characters, we must simulate all of this knowledge from the ground up; we sacrifice control for autonomy and knowledge for flexibility, but finding a suitable balance is an exceptionally difficult and, as yet, incompletely solved problem.

2.2.3 Emotion

The apparent desires of a character, and the way the character feels about what happens in the world with respect to those desires, are what make us care about that character. If the character does not react emotionally to events, if it doesn't care, then neither will we. The emotionless character is lifeless. It is a machine. [13]

Crucial to the creation of a successful character is its ability to express emotions. This does not necessarily require that the character have an explicit model of emotion², but that the character behave in such a way as to lead the observer to suppose that it does have emotions. As Thomas and Johnson observe, “From the earliest days, it has been the portrayal of emotions that has given the Disney characters the illusion of life.” [137]

Earlier, we claimed that personality is revealed through the behaviors of a character. To an extent this is true. However, the character must clearly express emotions as it acts in order for the audience to understand *why* it is doing what it is doing. If we see two men in a dimly lit alley, and one pulls out a gun and shoots the other, we have no understanding of what led to that action. If he is crying, or trembling in fear, or his face is contorted in rage, we are given vital clues about the reasons he has decided to act as he has. A character’s actions tell us what it is capable of; its emotional expressions reveal the thought processes behind those actions.

Besides helping to explain why a character is behaving in a certain way, emotion is the basis for empathy, and empathy is necessary for the observer to have any attachment to the character. As Elliott and Brzezinski ask, how can you feel sorry for an agent that doesn’t lead you to conclude that it’s sad? [40] A successful character must evoke an emotional reaction in an observer, and the principal mechanism for doing this is to exhibit emotion itself³.

Emotions, or at least some means or recognizing emotions in others, are necessary in animate characters that interact with people for another reason. A character must be able to respond in an emotionally-intelligent way to the actions of others. If a human interactor pats an animate character on the head, there is a strong expectation

²There is a growing body of research suggesting that emotions, rather than being an undesirable impediment to rational thought, are actually the fundamental mechanism by which people make decisions [31]. Models of emotion have commonly been used in animate character design for over a decade, but now there is beginning to be serious discussion in AI circles about employing models of emotion in rational agents as well as in animate characters (e.g., [28]).

³As with any rule, there are exceptions. Herman Melville’s *Bartleby the Scrivener* [96] is a short story about an affectless, characterless office worker who sits at his desk, refusing to move or react in any way, responding only “I would prefer not to” when pressed. He is eventually hauled off to prison, where he dies, alone and silent, but the overall effect is one of distaste rather than pathos. His lack of affect makes him seem eerily inhuman.

on the part of the human that the character will exhibit an emotional response, whether it is to smile, to giggle, to frown, or to do something else entirely. Similarly, if the human interactor cries, there is an expectation that the character should perceive this and react emotionally to it in some way. There has been considerable work on this problem already (see, e.g., [37, 38, 41, 47, 110]).

Of course, as [113] and [111] observe, there are significant dangers in using the term “emotion” when discussing the computations of a program. Whether or not the output of the program is intended to engender in a viewer the belief that the agent itself has or exhibits emotions, obviously it does not. However, we will allow ourselves the inaccuracy of saying that the character “has” emotions when we mean that it is exhibiting an expression that is intended to be perceived as an emotion. We will have somewhat more to say on this issue when we discuss the character architecture in Chapter 5.

2.2.4 Social Relations

As human beings are social, so too are characters. With few exceptions, most of a character’s existence is spent interacting with other characters, people, or both. These interactions are one of the principal ways in which a character exhibits its personality and its “thoughts.” Thus one of the qualities an animate character must possess is a capacity for social relations.

In this respect, animate character designers have an advantage. Reeves and Nass [112, 99] have convincingly demonstrated that people’s natural response to computers is to treat them as though they were social entities. Their approach has been to replicate traditional social experiments involving human beings, but replacing some participants with computers. The results strongly indicate that, even in circumstances where it is explicitly made clear that computers have no social intelligence or personality of any kind, people continue to react to computers as though they were social entities. As they describe it, “Modern media now engage old brains. People can’t always overcome the powerful assumption that mediated presentations are actual people and objects...[P]eople respond to simulations of social actors and natural

objects as if they were in fact social, and in fact natural.” They conclude that “individuals’ interactions with computers...are *fundamentally social and natural*, just like interactions in real life.”

There has been considerable subsequent research on how insights from the social sciences can be applied in character design (e.g. [36, 100, 67, 63, 120]). Of course, there is also considerable discussion in the animation and artistic literature on this issue. Johnstone [70], for example, frames social interactions as *status* relations, which are negotiations about the relative social positions of the characters, an idea adapted to animate characters by Hayes-Roth, van Gent, and Huber [57]. Thomas and Johnson [137] claimed that it was only when they began to explore character relationships in their animations at Disney that the more subtle and psychologically revealing aspects of the characters were revealed.

A character must be capable of social interactions; those interactions should differ between any two pairs of characters, as each has its own personality, emotions, and goals; and, given humans’ knowledge of and sensitivity to social relations, this should be seen as an opportunity to add depth and subtlety to the character’s personality.

2.2.5 Expressiveness

A character is how it behaves. [71]

Expressiveness is the crucial artistic quality for animate characters. It does not matter what architecture drives the character or what reasoning the character is capable of, only what it can express to the observer. For a traditional rational agent, the key property is *action-selection*: the ability to choose the most appropriate action at any given time. For characters, this property becomes *action-expression*, the ability to effectively express the current choice of behavior [126].

Much of the discussion in the animation literature revolves around this point. One of the earliest Disney principles called for clearly expressing a single idea before moving on to the next one [80]. Cartoons have an advantage here because they strip away inessential qualities, and can emphasize just those elements that the animator wants the audience to notice. According to Gard, “Through exaggeration, cartoons

are able to elicit very powerful emotional responses from an audience, because cartoon acting is a concentrated version of live acting.” [48]

Some characters, of course, are successful precisely because they are not expressive. The stone-faced villain in the dusty Western street enhances his effectiveness by making it impossible for us to know what he is thinking or planning. Anti-heroes such as Dirty Harry or the Terminator function in the same way. However, they are the exception rather than the rule. In general, for a character to come to life, it must give the audience an opportunity to see it thinking and feeling, and this requires that it effectively express itself.

2.2.6 Believability

A character’s behavior must be believable within its context. That is, given the character’s personality, emotional state, goals, and social and physical environment, the character’s choices of behaviors seem reasonable to an observer. This does not mean “reasonable” in the sense of rational, intelligent, or optimal, but reasonable in the sense of being consistent with the personality of the character. According to Neal Reilly [113], “Believable characters are characters that seem to be alive and that an audience has emotions for or about. Believable does not mean honest, convincing, or realistic.” Or, as Bates [14] puts it, “believability will not arise from copying reality...It is art which must lead the charge here.”

Believability is not about reality. It is about producing in the observer a sense of the reality of the character. Laurel writes, “Computer-based agents, like dramatic characters, do not have to *think* (in fact, there are many ways in which they cannot); they simply have to *provide a representation from which thought may be inferred.*” [81]

There are two degrees of believability. First, which we call *believability in the small*, is that each individual behavior the character exhibits is a believable one in its setting. In general, a character in a burning building who runs for the door is behaving believably. A confused-looking character on a streetcorner who asks a passerby for directions is behaving believably. In contrast, a politician at a news conference who

begins to quack like a duck and flap his arms is behaving very unbelievably⁴.

The other type of believability is *believability in the large*. This means that not only are the individual behaviors themselves plausible, but the string of behaviors taken as whole makes sense to an observer. A character who picks up a heavy rock, inadvertently drops it on his foot, and curses, is behaving believably; but a character who repeats this process over and over quickly seems unnatural. Likewise, a character who walks up to you, who smiles, greets you, angrily shakes his fist in your face, asks about the weather, then begins weeping is behaving in an exceptional and improbable way – even though no *one* of those actions might seem unnatural.

This is the critical and most vexing problem for animate character creators. An author or animator can intelligently choose the sequence of behaviors a character will enact, while the animate character designer must rely on the architecture to make those decisions. Unfortunately, most, if not all, animate characters fail to achieve this second type of believability. The tendency is for architectures to jump from selecting one behavior to another according to whichever behavior is regarded most appropriate at a given instant, but without respect to the overall flow of activity. Sengers calls this *schizophrenia*⁵ [125]. As she writes, “These creatures, while perhaps intelligent in a formal sense, do not appear to have the coherence of behavior over time that we impute to living creatures.”

For rational agents, schizophrenia is not an issue. They are guided according to measurable metrics for optimality, and while the optimal sequence of behaviors may seem bizarre to a human observer, the point is to achieve some functional goal efficiently, not to behave as a human might expect. We can, of course, attempt to frame believability in terms of optimal behavior with respect to certain human expectations. There are two difficulties. The first is that human perception is highly subjective, so no two observers will have exactly the same reaction to the same character behaviors. Subjectivity makes it problematic to find an “optimal” course of behavior. Secondly, people are extremely sensitive to slight variations on expected behaviors. As Jones

⁴We hope.

⁵I feel constrained to point out that this term was not chosen carelessly, and Sengers goes to considerable pains to trace the origins and current usages of the word in psychology and critical theory to justify its application in this context.

writes, “We are far too close to other human beings...we are subconsciously critical of other human beings according to how they deviate from our own behavior or from standards of behavior we approve of.” [71] At best we can hope to produce sequences of behavior that *seem* reasonable to *most* observers.

We will have more to say on the subject of building believable sequences of behaviors in Chapters 5 and 6.

2.3 Qualities of animacy

So far our discussion has focused on the properties that make effective characters. These observations are generally true of any artistic characters, coming from literature, live film, animation, etc. In the creation of computer-controlled *animate* characters, we must also explicitly recognize properties that are implicitly assumed to exist by artists. An artist, whether an actor, a writer, or an animator, has direct and complete control over the manner in which the character will present itself to an audience. The designer of an animate character, on the other hand, may constrain how the character behaves, but is in turn constrained by the autonomy of the character. The qualities of that autonomy meaningful to our discussion of characters form the topic of the remainder of this chapter⁶

Animacy and agency: A second aside

The term traditionally used to describe an autonomous computer program to which observers might reasonably ascribe some degree of “intelligence” is *agent*. We have, as observed earlier, preferred to call our characters *animate* in order to emphasize their intended lifelike qualities, rather than as *agents*, since there are many conflicting interpretations of that term, and under some provided in the artificial intelligence literature, even such simple objects as thermostats meet these criteria [121], which is hardly consistent with our intention. After comparing many popular definitions, Franklin and Graesser [46] synthesize this more helpful one: “An **autonomous agent**

⁶For an excellent, if brief, description of the qualities that intelligent agents in general should possess, see Langley and Laird [79].

is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future.” Yet this still does not reach the heart of what we mean by *animacy*.

Sengers [125] clarifies this morass, at least for our purposes, by observing that agenthood is best thought of as a useful metaphor for understanding certain programs for which it is useful or important to think of them as properties we associate with intelligent beings. To tie our interpretation of animacy with the traditional notion of agency, an animate character is an agent whose behaviors are intended to produce in an observer the impression that it is an intelligent, living being. When we subsequently use the term “agent,” this is the kind of agent we will be concerned with.

Yet another way to view our animate characters is as communications [125]. That is, the animate character is a mechanism for communicating an intention between the designer and the observer or interactor. This is because the agent’s activities are intended principally to be *understood* by the observer, and its fundamental success or failure depends on this communication being effective. It does not “matter,” in some sense, whether the character is actually capable of learning, perception of the environment, planning, or sophisticated reasoning. It only “matters” whether the agent engenders in the observer a sense of its lifelikeness. Obviously, an agent that lacks these sorts of capabilities will likely do poorly in engendering this sense, but the point is that it is not these abilities but the effect that they produce on the observer that is the purpose of the character’s actions.

2.3.1 Goal-directedness

Key to the appearance of animacy and to the idea of agency is that the character is not simply reactive to its environment but is directed by its own internal goals and motivations.

As Thomas and Johnson [137] observe about animation prior to 1930’s, “...the only thinking done was in reaction to something that had happened.” As they write,

“Of course the potential for having a character really appear to think had always been there...but no one knew how to accomplish such an effect. That all changed in one day when a scene was animated of a dog who looked into the camera and snorted. Miraculously, he had come to life!” This becomes one of the guidelines they provide on character creation, “Look for places to show the character thinking.”

Wooldridge and Jennings [147] characterize this property as *pro-activeness*: “[such] agents do not simply act in response to their environment, they are able to exhibit goal-directed behavior by *taking the initiative*.”

A successful character must at least *appear* to be driven by its own goals. Human beings are goal-directed, and we expect to recognize goal-directed behavior in other intelligent entities [112]. And for animate characters, that other key quality of personality is dependent for its communication on goals. As Hooks [61] writes, “The idea of *personality* only makes sense as it relates to movement, and movement is an effort to survive in the world.”

Even in agent architectures that avoid any explicit internal representation of goals (e.g., [25, 26]) observers tend to impute goals to their actions. Following Loyall and others [87, 113, 125] we take the view that it is more effective to foster this perception by permitting the character’s designer to express those goals and the ways in which the character can exhibit them directly.

2.3.2 Reactivity

Reactivity is a property of agents that “perceive their environment...and respond in a timely fashion to changes that occur in it.” [147]. Reactivity requires that the agent have some sufficient mechanism for perceiving the salient features of the environment and for allowing those perceptions to influence subsequent behaviors.

This is another property that we certainly expect to find in characters. Under special circumstances, character can and do exist that take no notice of what is happening around them, or are indifferent to that knowledge; but, in general, a character is revealed to the observer through the way in which it responds to changes in the world.

Loyall [87] emphasizes that, for characters, the reactions must be timely; that is, the character must be not only reactive but *responsive*. An animate character of a man in a ruined castle who sees a horrible apparition and then stands motionless for several minutes before fleeing in shock and horror is reactive, but hardly lifelike. Conversely, responses that come too quickly may also destroy believability of the character. There is a tempo to the behaviors of living beings and animate characters must respect it.

2.3.3 Situatedness

One of the major research areas in artificial intelligence is *planning*, by which is meant the study of formal methods for finding sequences of actions that will achieve certain goals in formal worlds. Deciding what route to take to travel across country, ordering the steps in assembling a bookshelf, or scheduling observatory time on a telescope are examples of this type of problem. It is a particular kind of problem-solving that has long been assumed to be an example of the kinds of intelligence AI researchers are trying to achieve.

The anthropologist Lucy Suchman, in examining AI research on planning, observed that this is only one kind of planning, one rooted in a Western view of what “planning” means. In comparing Western approaches to sea navigation (plot the course you need to follow to reach your destination, then follow that course) to the Micronesian approach (at any particular moment, adjust your heading to lead you in the right general direction), she offered the concept of *situated action* to explain the latter view. As she writes,

...our actions, while systematic, are never planned in the strong sense that cognitive science would have it. Rather, plans are at best viewed as a weak resource for what is primarily an *ad hoc* activity. It is only when we are pressed to account for the rationality of our actions, given the biases of European culture, that we invoke the guidance of a plan. [133]

Thus a *situated* agent is one whose behaviors are directly and primarily influenced by the current circumstances, rather than by preconceived long-term plans. Such an

agent, as Agre and Chapman [1] write, is “...*participating* in the flow of events...[it] must *lead a life*, not *solve problems*.”

A successful animate character must possess this quality of situatedness. Its behaviors must be believable and meaningfully interpretable in the context of the current situation — social situation, physical situation, conversational situation, the present situation in all its respects.

2.3.4 Broad capabilities

An animate character must possess, at least superficially, all of the qualities that we expect to find in living beings. It must, or must appear to see, hear, speak, move, think, feel, learn, etc. Bates [13] uses the phrase “broad but shallow” to describe the requirements for the capabilities of an animate character, as compared to the deep but narrow capabilities that most intelligent agents possess.

In addition to possessing these capabilities, they must be well-integrated within the animate character. A character that switches from listening to thinking or from seeing to expressions of emotion in a jerky or obvious way will readily destroy the illusion of life. It must instead be able to integrate these behaviors smoothly and transparently, a requirement that might not necessarily be made of other kinds of autonomous agents⁷ [66, 125].

⁷Sengers [125] observes that this leads to a fundamental dichotomy in current agent design: on the one hand, breaking an agent into modules that cover individual capabilities is perhaps the only practical means of construction, but at the same time this places fundamental restrictions on the degree to which those capabilities can be integrated. She refers to this as the *atomization* problem, which is the direct cause of the schizophrenia in agent behaviors we described in section 2.2.6.

Quality	Description
Personality	Possessing a recognizably coherent set of traits controlling the character’s behavior.
Individuality	Being distinct from all others through the kinds of things a character does, the ways in which they are done, and the frequencies with which they are expressed.
Emotion	Apparently having desires and feelings about the world with respect to those desires.
Social Relations	The capacity to react to other characters or people, and the ways in which behaviors are chosen to do so.
Expressiveness	Effectively conveying the designer’s intention behind the action to the audience.
Believability	Giving the appearance of being alive; being something the audience cares for or about. Behaving in a way that the audience recognizes as like life in a dramatic sense.
Goal-directedness	Giving the appearance of having goals and acting to fulfill them.
Reactivity	Responding in a timely way to what is happening in the world.
Situatedness	Acting in a way that reflects an apparent understanding of the character’s current context (physical, social, conversational, affective, etc.).
Broad Capabilities	Being “complete” in the sense of having, to some degree, the major components of animacy — being able to sense, think, act (hear, see, speak, move, feel, etc.).

Table 2.1: Qualities of animate characters

Chapter 3

Enabling Better Character Behavior

*All the world's a stage;
And all the men and women merely players...*

— Shakespeare, *As You Like It*, Act II, Scene 7

In the previous chapter, we defined the qualities that an animate character must possess to be successful. It must be functional, which is to say that it must be capable of performing its intended task and performing it effectively; and it must be convincing, so that its actions are not only successful from a utilitarian point of view, but that it convinces human viewers or interactors, or at least does not clumsily dispel their belief, that it is a thinking entity with its own beliefs, goals, emotions, and personality.

Depending on the purposes for which the character was built, its functional goal may simply be to be convincingly lifelike. Characters have been created to serve many ends: some characters are used in games, where they are assistants, informants, and opponents; educational programs have teachers and tutors; applications have wizards and helpers; some software and Web sites use them to sell products and services; and so forth. Nevertheless, the one thing they have in common is that they have been created as characters in order to encourage certain beliefs and attitudes in the people

who interact with them, whether to make them more comfortable, more trusting, more attentive, or more engaged.

This is a difficult problem. A character created for a film, such as Snow White in *Snow White* or Gollum in *Return of the King*¹, can be painstakingly edited, frame by frame, to act precisely as the director intends it to act. There is no autonomy, no chance for the character to make a suboptimal choice. Such characters are animated, not animate by our definition. This is also true of characters in much interactive software. If the interaction is tightly controlled by the designer, the characters in the software may have no intelligence, but simply be puppets executing prescribed actions. Or the range of action may be so narrow that a simple set of rules can dictate how a character should behave.

But what about characters in more open-ended environments, who have more complex tasks to perform? If the designer of the character does not know in advance all of the details of the environment in which the character is going to operate, or the sequence of events it is going to encounter, then the character has to be programmed with sufficient intelligence to make its own choices. The more flexible the circumstances and the more capable the character is meant to be, the more knowledge and reasoning ability the character must have.

After a point, this becomes problematic. Partly this is because there are significant limitations to our technical abilities when it comes to building intelligent agents, characters or otherwise. The hard artificial intelligence problems have still not been solved. Partly this is because of the vast amount of knowledge required to make even simple functional decisions in complex environments. Partly it is because of the vast amount of knowledge required to behave in convincingly believable ways, and unlike factual knowledge about the world, which AI researchers have attempted to codify in knowledge bases for computers to use, there are no such formal bases of knowledge about what it means to behave believably².

¹Gollum is an interesting case, as its movement is produced through motion capture of a human actor, and then applied to a graphical embodiment on film. Yet the point is the same — all of the behaviors of the character are directly the result of human decision-making.

²There have been many attempts, especially in the last century, to codify acting methodologies and practices in the theater (e.g., Stanislavsky [130]), which are of some help here. However, there is a level of detail that does not need to be explained to an actor (such as that yawning indicates

In any case, as we develop more complex and sophisticated environments, affording greater ranges of action, adding more knowledge to the character has its limits. Even human beings do not have complete knowledge of the real world or how to behave in it. In fact, by looking at how we address the problem of limited knowledge and complex activity in the real world, we can find some insights into how to handle the problem for characters in virtual worlds.

3.1 Acting in the world

For our purposes, we can look at the way in which people act in the world (or react to it) as an endless cycle of several steps:

- **Perceive the environment.** First of all, we must be able to sense the environment and what is happening within it. Anything that is not available to our awareness cannot influence our behaviors.
- **Recognize what is occurring.** Beyond simple perception, we must have some degree of recognition of the environment or the event. For example, at the end of the film *2001: A Space Odyssey*, the astronaut Frank Bowman is drawn into an alien artifact. The subsequent sequence of lights and sounds is an example of perception without recognition. We know that *something* significant is happening, but it does not signify to us; we have no idea what it is or what it means.
- **Interpret what is occurring.** That is, beyond the bare fact of recognizing something (be it object or event), we must put it into a meaningful context in order for it to be reasoned about. For example, we might perceive and recognize a man who draws a gun and shoots another man with it. However, a knowledge of the context of the action is necessary to evaluate it. Why is the gun being fired? Where is it happening? What are the consequences? We might further say that one *understands* what is occurring if the interpretation one settles upon is a correct interpretation.

fatigue) but which a character will not know unless explicitly told.

- **Act in accordance with what is occurring.** With an interpretation of a situation comes the capacity to reason about that situation, and our actions are the product of reasoning about our knowledge with respect to our goals.

This is (to a first approximation) how human beings function. For characters, of course, the story is somewhat different. All that is available to us of a character are our perceptions of its actions. We may infer that it is following a process like our own (perceiving, recognizing, and interpreting the world before it acts), but this is not necessarily the case, and in fact it is often *not* the case. Yet as we observed in Chapter 2, it is not what (or whether) the character is thinking but what we *think* it is thinking that is important. So long as a character can act in ways that cause us to believe that we are recognizing and understanding its “thoughts,” then it has achieved a degree of success.

However, short of designing a character that is perpetually asleep or in a catatonic state and ignoring the world around it, an animate character that does not act in the world, and act in a way that is recognizable (if not understandable), is not apt to be believable for very long. But in a complex environment, where a character does not necessarily possess all the knowledge needed to understand what is happening or to choose appropriate responses, how can this problem be addressed?

3.2 Knowledge in the world

People often find themselves in complex and unfamiliar environments. It is not possible to function using only the skills we have already learned. Instead, we constantly analyze our world as we interact with it, reinterpreting the environment, its contents, and the procedures we use to manipulate them. The world can either help or hinder this process.

Ordinarily we rely on the world to help us function. We do not memorize the minor variations in the procedures for making an ATM deposit at one bank or another, become experts in geography in order to navigate to distant cities, or study the designs of doors in order to make use of them. We expect to find clues or directions that



Figure 3.1: Knowledge in the world

enable us to perform these tasks as we do them. This information embedded in the environment saves us from having to learn how to do many of the tasks we do. The world shows us what to do: we do it, and then forget about it. As Norman writes, “People routinely capitalize on this fact. They can minimize the amount of material they must learn or the completeness, precision, accuracy, or depth of their learning. People can deliberately organize the environment to support their behavior.” [101] He calls this “knowledge in the world.”

Figure 3.1 shows some examples of knowledge in the world. Some knowledge is embedded in the physical properties of the object, as with the door handle. Knowledge can be conveyed by the placement of objects, as with the guard rail around the painting. And some knowledge is symbolic, as with the signs and monitors. In each case the world is providing us guidance about what to do or how to do it.

The psychologist J. J. Gibson used the term *affordance* to describe a property of a physical object that a person could recognize as permitting certain interactions [51].

A flat tree stump, for example, with its surface a few feet from the ground, affords sitting on, standing on, leaning on, putting other objects on, burning, and so forth. The design of objects and environments by people is a task in which the goal, in part, is to afford certain activities while discouraging or de-emphasizing others. Thus a round door handle affords turning, a button affords pushing, and a protruding loop affords pulling. A printed sign posted at eye height affords reading. Norman's observation is that this idea can be applied to elements of virtual environments (such as computer interfaces) as well as to elements of the real world.

There are several advantages of embedding knowledge in the world. First, such knowledge is immediately retrievable; it is right there in front of us, just when we need it. If the world has been designed properly, all we need to do is look, listen, or touch. Second, it requires no learning. The emphasis is on interpretation rather than memorization. We only need to interpret the world to use it.

Third, by being part of the world, such knowledge accurately reflects the state of the world. Knowledge in the mind may be inaccurate, outdated, or misleading, and its accuracy and detail may fade over time. It also means that it is not necessary to store this knowledge — that is, the memory load imposed upon us is diminished, which is increasingly important as the number of things we have to remember grows. As Brooks observes, “The world is its own best model.” [25]

Lastly, these cues and constraints in many cases make it possible to act almost immediately. Ideally, neither significant amounts of reasoning nor practice are required. A door handle is designed to be easy to comprehend and to use, as are many common physical objects, be they forks, stovetops, light switches, ATM machines, etc.

In [102], Norman uses the term “cognitive artifact” to describe “those artificial devices that maintain, display, or operate upon information in order to serve a representational function and that affect human cognitive performance.” The example of a pre-flight checklist for an airline pilot is given as an example. The artifact has the effect of changing the nature of the pilot's task. In this case, it is changed from one of remembering a sequence of complex operations to one of reading and following a list of instructions. The checklist represents compiled knowledge that is available in the environment and which obviates the necessity of the pilot possessing certain

knowledge and, conceivably, reasoning abilities. This is precisely what we would like to do for our characters.

Of course, this approach has limitations as well. All necessary information must be made available to the user, explicitly or implicitly, which can lead to cluttered, clumsy designs for complex systems. Certainly there is no way to embed all the knowledge needed to fly an airplane within the cockpit, and even an office copying machine is so complex that the guidance its embedded knowledge provides is often barely comprehensible. So while this approach can work for some objects and tasks, as the complexity of the interaction increases, the limitations of the physical environment become insufficient to completely constrain the interaction.

Relying on knowledge in the world is also inefficient. It must be found, examined, and interpreted before it can be used. While a good design can facilitate immediate understanding, thereby reducing the demand on a user's memory and cognition, a bad design can mislead and create precisely the problems it was intended to solve, requiring extra thought and memorization to get around it [103]. In contrast, what Norman calls "knowledge in the head" has already been analyzed and stored and can be applied more quickly and without the possibility of misinterpretation. Such knowledge makes us quick and precise in performing well-understood, familiar tasks.

In the real world, a person's ability to perceive affordances is compromised by the noisy, uncertain nature of the environment. The designer has limited resources to convey the intended use of a thing or place, and must hope that the user correctly interprets his or her intent. A human being must overcome these obstacles in order to make use of the knowledge in the world.

An agent in a human environment is even more helpless. It does not have access to the wealth of unobtrusive but critical information that allows us to perform our tasks. The vocabulary of visual clues, physical constraints, and commonsense knowledge the designer relies upon is not one most agents can understand.

This problem is greatly simplified, however, for agents in virtual worlds. In these environments, designers have considerable control over what knowledge is available to an agent and how it can be made available. Knowledge can be embedded in the world in a literal sense, through representations attached to objects, entities, and locations

in the code describing the world. We call this knowledge *annotations* of the virtual world [34, 33, 35].

Like the affordances of a designed object or an interface, an annotation is a communication about the properties and opportunities for interaction. However, unlike affordances, annotations are intended for the agents (in our case, the animate characters) in virtual environments. Their purpose is to provide guidance to these agents about the content and meaning of the environment and the interactions that the agent may have with it. And being in a virtual environment, annotations can transcend many of the limitations of the design of affordances in the physical world.

3.3 Examples

So far this discussion of character tasks and annotations has been quite abstract. We have in mind, however, some particular problems that we would like to use annotations to solve. Here are some motivating examples to lead us into the next chapter, which will discuss annotations in more technical detail.

- Many environments engender emotional reactions in their inhabitants. A packed airport gate produces, in most travelers, a sense of frustration; a dark, low-ceilinged cave, the roof of a skyscraper, or the thick of a battlefield engender fear; a sunny, warm tropical beach leads to feelings of relaxation and well-being. In a text-based virtual world, these places would be described with words, while a graphical environment would draw them. How is a character to understand the emotional overtones of these places without either natural language understanding or vision on a par with a human being? Even if it did have comparable abilities, that does not necessarily mean that it will come to the correct dramatic conclusion — the reaction that the builder of the environment *wants* the agent to have.
- Similarly, events in the world may be intended to produce certain kinds of reactions in people and characters. A dragon flying down and attacking the village is meant to produce terror, excitement, despair; and a character must have a

way to determine this even if it has never been in that village or encountered a dragon, and even if it does not know what a dragon *is*. If a person kicks the character, it should react with fear, anger, frustration, irritation, and should be able to do so whether it is kicked, slapped, hit, poked, etc., and without necessarily being able to identify each of these as a person does.

- In the cases in which we are interested, the character is interacting with one or more other agents (characters or people). Consider an environment with a game to be played, such as chess, checkers, or skee-ball. How will a character know how to play these games? It may be possible for it to learn through experiment, but that is the province of the human; generally a person will not care to wait while the agent works out what to do, even if it has the capacity to do so. Instead, the person wants to *play* with the character. The character needs to be able to play, and moreover, play (and react to its progress) in a way that is consistent with its personality, and to do so without considerable reasoning and experimentation.
- Think of a parent leading a child through a science museum. The parent may know little or nothing about the contents of the exhibits, and have no foreknowledge about how to interact with them. As they move through the museum, the parent relies on signs, illustrations, and panels of text to explain what an exhibit is about, what it means, how to interact with it, and what the interaction means. The parent can examine these annotations, digest them, explain what is happening to the child, and interact (or help the child to interact) with the exhibits. We would like a character to be able to perform the same role in comparable virtual setting, and with the same lack of foreknowledge of the specifics of each exhibit as the parent has.

Chapter 4

The Annotation Framework

In the previous chapter, we introduced the idea, taken from cognitive psychology, that people ordinarily rely on the features of the world around them to assist them in understanding it and interacting with it. Norman [101] observed that, this being the case, the world can actively be designed to facilitate this process; by carefully creating this knowledge in the world, the environment makes certain interactions and interpretations of the world easier while making others more difficult or more obscure.

Our central insight is that, if humans rely on well-designed environments to explain function and suggest appropriate behavior, autonomous agents, being less intelligent and less perceptive than their human counterparts, must do the same, and that this approach is especially well suited to dealing with the problem of making animate characters seem more believable. Also, the nature of the Internet, in which these virtual worlds exist, readily lends itself to highly-distributed computation, dynamic data sources, and very mobile agents, all of which support our contention that building less intelligent but more adaptable characters for use in many similar environments is a good solution to this problem.

The knowledge we wish to embed in a virtual environment for use by our characters we call *annotations*. In this chapter we outline the properties that it is desirable for annotations to possess, the broad classes into which they can be divided, and the ways in which they can be conveyed to our characters.

4.1 Design and drama

Just as with the design of the physical world, the creation of annotations for characters serves a purpose. However, the purposes are somewhat different, and it is worth highlighting that difference before we begin our discussion of how those annotations are created.

An ordinary person acting in the world has some set of goals guiding his or her behavior. The designed world in which that person operates was created in order to facilitate certain tasks and the acquisition of some knowledge at the expense of (or possibly with the intention of) making other tasks difficult or other knowledge obscure. The goals of the environment's designer and the goals of the person acting in the environment may align, they may conflict, or, as is most often the case, they are in partial agreement, and the resultant action is in effect the product of a negotiation between the designer and the person acting in the world.

Near the other extreme is the relationship between a director and an actor in a film or stage play. In this case, the overall goal — that of creating a dramatic work of art — is shared completely by both. Like the designer in the everyday world, the director creates and manipulates the environment, including the sound, lighting, and props on the stage, to facilitate certain actions. The actor works within these constraints and according to the script shared with the director in order to produce a sequence of actions that is dramatically satisfying to the audience.

The kinds of situations we are concerned about for our characters fall somewhere in between. Like the director and the actor, the designer of the annotations in the virtual world and the creator of the character are both attempting to enable the character to produce interesting, believable behavior that is consistent with the structure of the environment. However, like someone in the everyday world, and unlike an actor, the character is not constrained to obey the wishes of the environment's designer. There is no fixed script that has been agreed upon in advance.

Perhaps the closest analogy is to improvisational, rather than traditional, drama. In improvisational theater, the players do not have all the details of the story before they begin. Instead, they may have a set of ideas, personalities, or props that they

are challenged to incorporate into an interesting story. Some of this information may be available before they begin, but more of it may become available as the action progresses. The interaction between the actors is, on one level, the attempt to act out a story, while on another level it is a negotiation about how to proceed in the narrative. Similarly, the designer of the virtual world has some goals and knowledge that are intended for the character to express to the human observers or interactors, and the character attempts to do so without possessing much or any of that knowledge in advance.

4.2 The design space of annotations

An annotation is not simply markup. An annotation is a communication between the environment and the character. It represents knowledge that bridges the gap between the conceptual world of the environment's designer and the conceptual world of the character's designer. As the environment and the character are two independent systems, there is a tension between supporting the goals of the environment's designer and supporting the goals of the character's creator. These tensions are spread along three major axes which form a space in which the choice of any particular set of annotations must be made.

- **Generality with respect to character**

The designer of a character would like the annotations to be tailored to that specific character's personality and abilities. The more focused the annotations, the more believable the character's reactions will be, in much the same way that a director who is working with an actor at creating the nuances of a particular role in a story is more effective than a director giving general advice about how to act. Annotations about emotional responses to events, for example, would take into account the character's unique personality and would describe exactly the response it should exhibit. Annotations describing what to say about the environment, whether they are lessons, hints, observations, or directions for the character to share with users, could be provided in natural language written in

the character's voice. Even suggestions about nuanced behaviors exhibited by this character and no other could be made by the environment at appropriate times.

However, the environment's designer wants these annotations to be useful to a wide variety of characters with different personalities and abilities — the whole point is to enable many characters to operate in the same circumstances. Annotations so focused on a single character would be essentially useless to any other characters. Thus there is a tradeoff between providing enough specificity for a character to express itself in ways that are consistent with its personality and unique behaviors, and enough generality that many characters of differing temperaments and ways of behavior can make use of them.

- **Generality with respect to world**

A second axis of the space is the generality of the annotations with respect to the world itself. A Web site about American history would need an annotation language very different from one for a Web site about advances in microbiology. Even within a narrower focus, the designer of a museum Web site about sculpture, for example, might prefer an annotation language that can represent knowledge about casting methods, while an art site that deals with oil paintings might prefer to encode information about brush style, the chemistry involved in certain specific colors, and so forth. The designer of the environment would prefer a language tailored specifically to the content of that one environment, while the designer of a character who is meant to navigate many environments would prefer a language that has sufficient flexibility to detail the contents of other spaces.

- **Generality with respect to task or goal**

A third axis of the space corresponds to generality with respect to the character's tasks or goals. At one extreme, the designer of the environment may wish to direct a character to certain behaviors, rather than supporting it in achieving its own goals, which may be somewhat different.

Consider a person at Stanford University who is going to visit a friend in Berkeley, but who does not know the area. The friend could provide point-to-point driving directions, directions of the form *Start at Palm Drive. Drive for 3.8 miles on University Avenue. Take Highway CA-84 east for 7.9 miles...* This information is in a form that permits only one task, which is to drive to Berkeley.

On the other hand, the friend could provide a map of the Bay Area. This would make it possible to drive to Berkeley, but it would also make it possible to drive many other places. This is a more general annotation with respect to task — it permits many tasks to be undertaken, may support many goals, while the point-to-point directions are useful for only one. However, it requires more intelligence on the part of the driver than the point-to-point directions. The map must be interpreted, must be employed to extract the information that those directions provide directly.

Likewise, for our annotations, the tradeoff is between making annotations that answer specific questions straightforwardly but which cannot be used flexibly, and offering annotations that may be used for many purposes but which require more work on the part of the character to be used for any particular purpose.

The other consideration that must be made for the choice of an annotation, beyond the generality or specificity it has for any particular character, world, or task, is how extensible the annotation is. That is, given that we have chosen a point or a region in this space in which to place the annotations to meet our design goals, how easily can we add more knowledge or different kinds of knowledge to those annotations? How easily can we expand that space?

Extensibility is an important point when we consider that not all characters have the same capabilities. Some may only be able to understand the most basic elements of a description, while others may be more sophisticated and be able to make use of more complex information. In the general case, we would like our annotations to support many levels of capacity in their readers. Of course, there may be design considerations that make the more rigid, limited annotations entirely appropriate, but we should be conscious of those limitations when we choose them.

4.3 A taxonomy for characterizing annotations

The purpose of an annotation is to enable an agent to operate more effectively in its environment by providing it with information about the environment and what is happening there that it would be difficult or impossible for it to obtain otherwise. In principle, such an annotation might consist of any kind of arbitrary data that could be helpful to an agent. This sort of broadly inclusive description does not, however, provide any guidance to a designer needing to understand the practical uses to which an annotation may be put.

As a result, we offer an organization of the types of annotations along several dimensions that define those characteristics that are meaningful to the designer and which will help him or her to partition the annotations for the environment in ways that are both conceptually useful and relevant in practice. Here we present the framework that we have developed in creating the two environments described in this dissertation.

Obviously, this is not only not the only possible framework, it is not even the only reasonable one. Our categories are nowhere near as detailed as those used in wider knowledge representation researches, such as Cyc [83] or those of Schank's conceptual dependency framework [124]. Nevertheless, we believe it does emphasize the important differences that any designer, whether working on believable agents or not, will need to consider, without making more fine-grained distinctions that may be unimportant or inapplicable for one pairing of agent and environment or another.

We describe four dimensions. The first three are applicable generally to any character/environment pair. The last is only useful for agents with personality and emotion but is fundamental for that class. For each we provide explanation and some motivating examples.

4.3.1 The structure of annotations

{ **descriptive, directive** }

The first dimension distinguishes between descriptive annotations, which provide factual information with no indication about how that information should be employed, and directive annotations, which provide the character with some explicit direction, suggestion, recommendation or order in regard to its behavior.

Directive annotations assume implicitly that the character will benefit from instructions issued by the environment. The more intelligent the character – the better its sensors, its knowledge, its reasoning abilities – the less direction it is likely to need. Yet as we have observed, even broadly competent human beings routinely rely on directive annotations for acting in the real world, either because they are uncertain how they would want to behave or because they have agreed, explicitly or implicitly, to modify their behaviors according to some external standard. An “applause” sign in a television studio, a stop sign at an intersection, the emergency instruction card in the seat pocket of an airplane, and the arrows on a museum map indicating the stops on an audio tour are all simple examples of directive annotations.

The common “exit” sign nicely demonstrates the difference between a descriptive and directive annotation. Unlike a stop sign, it does not carry any connotation of command. We do not expect people, upon seeing an exit sign, to feel compelled immediately to leave through the exit. Instead it has a more volitional sense: if you would like to leave, here is the place to go. It provides needed information that enables people to perform an operation they might wish to perform, rather than ordering them to act in a particular way. It is descriptive rather than directive.

Notice that this is not a hard division but a continuum. Descriptive annotations may contain information that is intended to be highly suggestive without actually issuing a command. “This is a dark and misty graveyard. It is midnight,” is merely descriptive, but adding that “most characters would be very frightened” is, while not directive, strongly suggestive that the character should exhibit fearful behaviors. At one end of the continuum, the character is provided with the data needed to reach the conclusion about how to act, and at the other extreme it is provided only with

the action it should perform.

Even directive annotations do not *force* compliance. A driver is free to run a red light and risk an accident or a ticket. The real world is filled with directive annotations people routinely ignore, many of them meant to induce behavior that is desirable from the point of view of the annotator but potentially undesirable from the point of view of the character, such as a sign urging us to “Drink Coca-Cola” or the verbal exhortation “Don’t touch that dial!”

As with people, a character does not necessarily have to obey a direction from the environment, even if the results of disobedience are suboptimal or actually harmful. The purpose of these annotations is to indicate clearly the types of behaviors the designers of the environment *believe are appropriate* for the character to exhibit.

Examples:

- *The room is dark.* (descriptive)
- *There are exits to the north, east, and south.* (descriptive)
- *Follow Patrick through the north exit.* (directive)
- *There is an elevator call button here that may be pressed to summon the elevator.* (descriptive)
- *This painting is entitled “Aristotle Contemplating a Bust of Homer.”* (descriptive)
- *Don’t touch the painting.* (directive)
- *It is loud in here.* (descriptive)
- *Cover your ears.* (directive)
- *There is an old wooden bridge across the river here.* (descriptive)
- *The bridge looks unsafe.* (descriptive)
- *Don’t try to cross the bridge.* (directive)

{ entity, relationship, operation, event }

The second dimension characterizes the concept the annotation is describing.

An entity is some objective element in the discourse between the character and the environment. It may be manifest concretely in the environment, as for example a person, a door, a sculpture, an airplane, or it may be an abstract concept that is meaningful for the character in the environment, such as a problem to be solved.

A relationship is a connection among two or more concepts. That a boy is the son of his father, a ship is sailing on the ocean, or a chess piece is part of a chess set are simple examples of relationships among entities. Relationships can also involve operations and events; that setting off the dynamite blows down the door, the sunset precedes the dawn, and starting the car permits driving the car are examples.

Operations are actions that can be performed by the character. Operations come in two forms. The first are those actions that can be performed by any actor in the environment, that is, by any character or any user. These annotations simply make it clear to the character which such actions are accessible and how they are performed. Unlocking a door, pushing a button, reading a book, and shrieking in terror are all examples of such actions. The second type of operation are those actions that are available specifically to annotation-sensitive characters, that allow them to achieve effects that would not be possible through actions available to ordinary users. Requesting a map of the maze in order to assist a user to find the way through, and getting a list of how heavily-trafficked the different parts of the game world are so as to guide users into the less crowded areas are examples of these sorts of operations.

Events are changes in the environment or communications within the environment. An opening or closing door, a flash of lightning, a character leaving the area, or a sudden, inexplicable heightening of tension are all examples of events.

Examples:

- *There is a sharp pin here.* (entity)
- *There is a balloon here.* (entity)

- *The balloon can be popped with a sharp object.* (operation)
- *You have popped the balloon.* (event)
- *The water bottle is in the backpack.* (relationship)
- *An avalanche has sealed the north exit.* (event)
- *There is a telephone here.* (entity)
- *The telephone is on the desk.* (relationship)
- *The telephone can be answered when it is ringing.* (operation)
- *The telephone is ringing.* (event)

{ content, context, purpose, structure }

The last of the general-purpose dimensions captures the designer's intent for the annotation.

Content is the most straightforward type of annotation. Content describes elements and happenings in the environment, without providing any underlying justification for why those annotations are present or when or how they should be employed. The majority of annotations fall into this category.

Context annotations indicate to the character that certain knowledge is only relevant in some circumstances. What operations are supported by the environment, what operations are pertinent, and what information accurately reflects the state of the world are the kinds of things that may depend on context. An annotation telling the character that it can only swim where there is water, that pieces on a chessboard should only be moved when a game is in progress or that sundials only tell time when the sun is out are context annotations.

A purpose annotation informs the character that some elements in the environment were created to achieve a specific end. It may also detail what the character's role is in achieving that end, and provide information about how to accomplish it. In the adventure game environment, which consists of a sequence of puzzles for the

user to solve, a purpose annotation may group a set of areas, objects, and operations together as a single puzzle, providing the character instructions on how to solve the problem or hints to suggest to the user. In the online art museum, one purpose may be to instruct the visitor about a particular artist's works, and this may involve taking a tour of those artworks, pointing out certain features of that art, presenting details about the artist's life, and so forth.

Finally, a structural annotation describes some relationship between a set of annotations. This is not to be confused with an annotation that is a relationship, which describes connections between elements of the environment; structural annotations describe connections between *annotations*.

Examples:

- *The water bottle is in the backpack.* (content)
- *The telephone can be answered when it is ringing.* (context)
- *You are now through the looking glass.* (context)
- *The scroll at the center of the maze must be retrieved.* (purpose)
- *The steps in the process of solving the puzzle are as follows...* (structure)

{ effective, affective }

The final dimension is not fundamental for all characters and environments, but is specific to those combinations where personality or emotion plays a role. It differentiates between affective annotations, which explain emotional content or qualities, and effective annotations, which deal with the purely operational aspects of the world.

The purpose of an affective annotation is to give a character an indication of how it should react, emotionally, to the content of the annotation. Affective annotations might tell a character that the park in which it finds itself is relaxing, that the operation of sliding down the water slide is enjoyable, or that only frightened characters would leave the haunted house before the tour is over.

It might seem as though all affective annotations are by nature directive. They do, after all, supply the character with a subjective impression that the character has a very limited ability to rationalize. An annotation that says “this environment is frightening” can either be adopted as an implied command to appear nervous and fearful or it can be ignored, but without having justifications such as, e.g, the environment is dark, cold, and isolated, together with commonsense knowledge about people’s affective responses, the character cannot reason about the subjective impression.

We deal with this problem by ascribing a normative quality to these annotations. The presumption of the annotations is that there is some standard affective character, akin to the “reasonable person” in the legal world, and that these annotations describe how such a standard character would behave. The character that receives them may filter these impressions through its own personality before deciding what use to make of them. In the case of the frightening haunted house, a character with a particularly timorous or cowardly personality may run away screaming as soon as the tour begins, while an especially brave character would scoff at the possibility of danger.

Obviously this is an imperfect substitute for a character with human-level intelligence, which presumably would be able to form its own impression based on the purely factual aspects of the environment. It is worth observing, however, that in a dramatic context it is not necessarily desirable for each character to draw its own conclusions about how to behave. If the character’s role is to be a character in an interactive story, the creators of the story will have wanted the participants to have a certain kind of experience, and by providing these dramatic clues to the characters, they can guide them to support that goal.

While affective annotations deal with subjective impressions, effective annotations deal with the objective aspects of the environment. Such annotations explain that chairs are for sitting, you must have a key to unlock the door, with a fishing pole you can catch fish for dinner, etc. For a traditional autonomous character, the effective annotations are all that would be desired or needed to function.

One could argue that affective annotations break the separation between the reasoner and the descriptive system, since they are in essence telling the character how

to react to the situation. A more intelligent character, we would hope, would instead receive annotations about the state of the environment and interpret those annotations to judge its own emotional reactions. For example, a sudden rockslide that has the effect of blocking the only exit from the cave should cause the character to reason that certain prospective goals (such as escape) are presently impossible to achieve, giving rise to feelings of anger, frustration or fear. In fact this is precisely how the Em architecture works.

Nevertheless, affective annotations are not an unreasonable way to deal with the problem, and in fact are likely to achieve results at least as good as those we could produce through reasoning. The purpose of entertainment is to engender an affective response from the user. Thus every event produced by the environment was designed by its creators with a specific affective reaction in mind. Emotional annotations are a way for the designer to make the desired reaction explicitly comprehensible to the character. By reasoning, the character may miss the affective *purpose* of the event.

This contrast provides an axis on which we may evaluate these annotations. At one end, the environment provides a model that describes the events that take place within it. This model is at a level suitable for some characters to reason about. At the opposite end, the environment only instructs the character how to behave; in this case, the character is effectively an animate extension of the environment, rather than an independent, reasoning entity.

Examples:

We emphasize again that affective annotations should be read as indicating what a “reasonable” or “ordinary” character would be expected to feel, not that their affects must be expressed or felt by the receiving character.

- *The room is dark.* (effective)
- *The room is frightening.* (affective)
- *There is an old wooden bridge across the river here.* (effective)

- *The operation of crossing the bridge makes you (that is, would make a reasonable character) nervous.* (affective)
- *You have entered the tomb.* (effective)
- *Every area in the context of the tomb makes you nervous.* (affective)
- *Patrick unlocks the cell door.* (effective)
- *The unlocking of the cell door makes you relieved.* (affective)
- *Patrick's unlocking of the cell door gives you a sense of gratitude toward Patrick.* (affective)

4.3.2 The mechanics of annotation

{ passive, active }

Here we draw a distinction between annotations that are embedded in the environment but are not provided to the character until and unless the character requests them, as opposed to annotations that are sent to the character without the character seeking them.

The annotations in the Web art museum we will describe in Chapter 7, for example, are all of the passive kind. It is only when the character queries the server for details on the artworks in the museum that they are provided. Similarly, in the adventure game environment of Chapter 8, most of the specifics of the environment are also passive. However, when changes occur in the world, such as a puzzle being solved, another agent taking an action, or the environment changing its properties, then an annotation is actively broadcast to the character.

Very often the active annotations pertain to distinct events, but not invariably. A slow heightening of dramatic tension could be conveyed to a character through a sequence of active annotations indicating that the properties of the affective context were different, but these would not correspond to any actual change in the environment.

{ static, dynamic }

The last division we wish to mention is between annotations that are static and dynamic. Static annotations, not surprisingly, are unchanging. An annotation that describes the fact that a certain key will unlock a certain door, or one that Abraham Lincoln was born in 1809, are static annotations. These annotations do not represent the state of the world but its unvarying structure.

By dynamic, we mean annotations that do change over time, either because they represent aspects of the world that change, or because they provide different information from one character to another. Dynamic annotations are produced as the result of some computation on the part of the environment. An annotation that describes a room with a door in it must be dynamic if it indicates whether the door is open or closed. A character that queries the environment for its own point-to-point movement directions or for available choices in a game is making use of dynamic annotations.

In our own experiments, the Web art museum is composed almost entirely of static annotations, while the adventure game environment is largely dynamic, as we shall see.

4.4 Claims

In this chapter and the previous one, we have provided an intuition behind the idea of an annotated environment and offered some structure for how we will organize the annotations in our experiments. We close this discussion with three claims about what annotations will allow us to achieve for our characters. We will revisit these claims in the light of our example worlds of Chapters 7 and 8.

Claim: Annotations enable characters to behave believably in unfamiliar environments.

Our first claim is that annotations can provide the kinds of knowledge a character needs in order to react believably to what is occurring in the environment, and to express its personality and its emotions in ways that are consistent with the context

the environment creates. We assert that these annotations can be made general enough to apply to a range of characters who do not share the same personalities or individual behaviors, but specific enough that they can successfully be used to choose how to exhibit those behaviors convincingly.

Claim: Annotations enable characters to be functionally capable in unfamiliar environments.

One of the arguments for using annotations is that in this way we can build characters that can move among many environments with different content but similar interfaces or opportunities for interaction. Our Web museum of Chapter 7 is such a case: there are many Web sites that are laid out like physical museums, and which describe art and artists of various forms and from different places and times. Our second claim is that there are general classes of environments whose individual details are significantly different, as in the case of the Web museum sites or of online multi-player games, but whose general properties and supported activities are similar enough that we can build annotations to describe them all, and that these annotations will enable character who understand the general properties of those activities but not the particular instantiations from world to world nevertheless to be able to act capably in those worlds.

Claim: Annotations are a practical mechanism for achieving these ends.

Our final claim is simply that annotations are, in fact, a feasible way of accomplishing the goal of supporting many characters of limited resources and capabilities in behaving in ways that are believable and functionally meaningful across large environments or between similar but not identical environments.

Chapter 5

An Annotation-Sensitive Architecture for Animate Characters

The central thesis of this dissertation is that annotations are an effective mechanism for enhancing the lifelike qualities of animate characters across a range of environments. We make that argument regardless of the implementation details of any particular animate character architecture. Ideally, this approach should support any character architecture that is capable of perceiving and manipulating the annotations, regardless of how it does so.

In order to experiment with our annotations we have chosen to work with a particular architecture of our own design. The architecture we present here is intended to be a reasonable but straightforward example of that broader class of annotation-sensitive architectures. It was used to build the animate characters in both the text-based MUD adventure game of Chapter 8 and the Web-based art gallery described in Chapter 7.

We begin with a brief summary of the architecture in section 5.1, then turn to a more detailed examination of the individual components. A diagram of the principal components is shown in Figure 5.1.

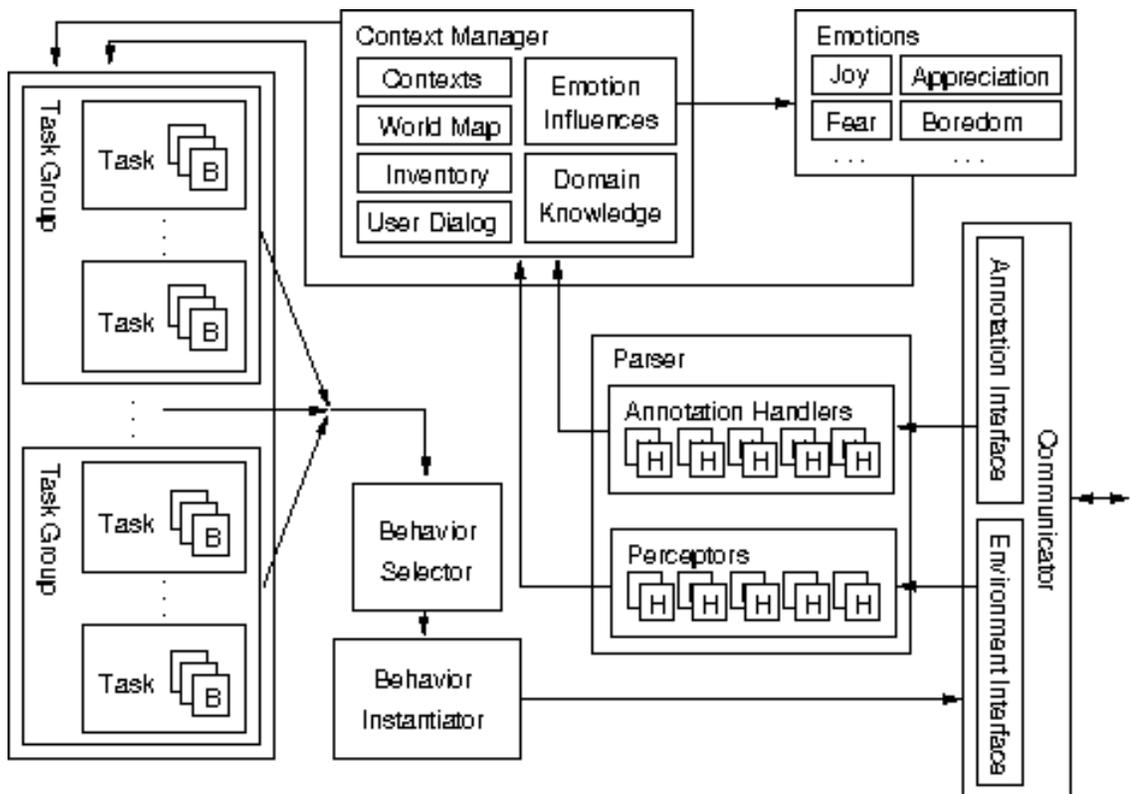


Figure 5.1: The animate character architecture

5.1 Overview of the architecture

The architecture is an extension of a standard reactive agent architecture with some limited planning capabilities. It is unusual in three respects. First, it has the capacity to modify its set of behaviors dynamically, so it may add or remove single behaviors or groups of behaviors during operation. Second, the annotation interface, which operates in parallel with the perception system, permits the agent to make use of additional information embedded in the environment specifically for that purpose. Third, because the purpose of the architecture is to create a character that behaves *believably*, it incorporates a model of emotion that directly influences the choices of behaviors to exhibit, and the choices of behaviors are influenced by other than purely rational considerations, so the character may behave in ways that would otherwise be regarded as suboptimal.

The architecture runs in a traditional *sense-think-act* loop. There are two types of handlers for sensory input. The first, the *perceptors*, recognize patterns in the ordinary stream of information from the environment seen by the character. The second, parallel sensors are the *annotation handlers*, which recognize annotations that the environment communicates to those agents in the environment indicating that they can interpret them. There may be an arbitrary number of both perceptors and annotation handlers in a particular character, and they may be associated with the core architecture itself or specifically with individual behaviors or contexts (described below).

Once a handler has recognized some sensory input, it may take several actions depending on the content. It may modify the character's current state, including emotional (or *affective*) state. It may modify information about the current context (that is, world state, such as the character's location, inventory, locations of other entities in the world, state of those entities, etc.), create or delete contexts, make contexts active or inactive, add or remove behaviors from the lists of active behaviors, or reorganize the existing behaviors. Multiple handlers may respond to the same inputs (or different components of the same inputs).

Once sensory input has been processed, the architecture updates the emotional state of the character. It does this by considering all of the active *emotion influences* that the character has perceived either from the environment or from itself, and computes how the strength of all of those influences translates into the strength of the character's emotions. Then the architecture decides upon a *expression* of the character's emotional state, an incorporation of all its current emotions into a single dominant theme.

Next the architecture's *behavior engine* thinks about the choice for its next behavior. Individual behaviors, which roughly correspond to atomic actions in the world, are organized into *tasks*. A task is a collection of behaviors that share some conceptual or functional properties, so for example the sequence of behaviors needed to retrieve an object from a distant room for the user, or the collection of behaviors used in storytelling, would form tasks. Tasks, like behaviors, may be added, deleted, or modified during execution.

The behavior engine chooses a behavior by polling each active task, asking that task to recommend a behavior for selection and to provide a numerical rating of its overall appropriateness. Each task does this by polling its behaviors, which in turn rate their own fitnesses according to the character's affective state, relevant aspects of the current context, behaviors recently executed, and so on. The task returns a suggested best next behavior to instantiate to the behavior selection module.

The behavior selection module is thus presented with a list of behaviors that are legal to execute and which the individual tasks rank highly as filling their purposes. The function of the behavior selection module is to choose one behavior from among this set, by comparing them according to the rankings provided by the tasks and also considering global concerns that are irrelevant for individual tasks. That it is always more important to follow a moving user than to relate an amusing anecdote, that it is never a good idea to smile immediately after the agent has just done so, that it is better to lecture to the user only infrequently, etc., are examples of such concerns. The result of this deliberation is a single behavior that will be instantiated this cycle.

Having finished thinking, it is finally time for the character to act. The selected behavior is passed to the behavior instantiation module. The purpose of this module is to translate the abstract behavior into an *action* that can be executed in the environment. The instantiation module informs the behavior that it has been chosen. The behavior then returns an action object to the instantiation module, which passes the action to the communications system to be sent to the environment. At that time, the behavior may also make modifications to its internal state or to the character's global state (affective state, context information, etc.) that is relevant to the execution of the action. For example, a character that has chosen to quake in terror may modify its emotions to increase its sense of fear.

The environmental action having been chosen, the agent executes it, and the cycle begins again.

5.2 Implementation details

The character architecture is implemented in Java. As Java is an object-oriented language, the architecture is organized in terms of objects. As much as possible, the various object classes (such as communicators, parsers, perceptors, tasks, and behaviors) are designed to be independent of the details of a particular environment. When this is not possible, subclasses are created to tailor the objects to the environments. For example, the MUD-based character has a communicator object in which the low-level I/O is done across a standard telnet connection, while the communicator of the Web-based character sends and receives XML and HTML via HTTP. Regardless, the interfaces between the various components are identical, so while at a low level the individual characters vary considerably, at an architectural level they are virtually identical.

One of the goals of the architecture is to make it reasonably straightforward to translate a character from one kind of environment to another, without having to rewrite its personality, its emotional model, and its abstract behaviors. Ideally, only the communications system and the actions in the environment into which the behaviors are translated should need to be changed.

We now turn to the details of these architectural components, beginning with the communications system and sensing of the environment.

5.3 Communicating with the environment

The communicator is the character's interface to the environment, sending all actions from the character and receiving all inputs sent by the environment. The communicator continually gathers input until it can form what it recognizes as a single, complete communication from the environment.

When the communicator receives a complete communication, it is passed to the parser responsible for processing that type of communication¹. The parser consists of

¹The architecture supports the concept of multiple parsers, each of which handles a different type of communication; so, for example, a character in a sophisticated 3D environment might have one parser for sounds, another for text, a third for graphical position data, and so forth. In the

sensory handlers that are designed to respond to particular communications. For example, there may be a handler that reacts when the character perceives that someone has kicked it; another that is activated when the character perceives that someone has picked up an object; another when someone asks a question; and so on. There are two kinds of handlers in the parser: a set of *perceptors* for handling ordinary environmental inputs and a set of *annotation handlers* for processing annotations sent by the environment.

In the case of the text-based MUD environment, for example, an ordinary environmental communication is a contiguous block of text terminated by a linefeed character, such as “The pirate leaves north,” or “Harker picks up the shovel,” or “You are standing in a maze of twisty little passages, all alike. There are exits to the east and south.” An annotation is an structured XML fragment; an example is shown in Figure 5.2.

When it receives a communication, the parser cycles through the appropriate list of handlers (perceptors or annotation handlers, depending on whether the communication is an annotation or not), attempting to find a handler that is capable of processing that communication. If it does so, it passes the communication to that handler and the handler executes some code in response. There may be multiple handlers that can process the same communication, in which case they are invoked in the order of their specificity until either one of the handlers consumes the communication or until all handlers have been invoked. If there are no handlers to match the communication, it is ignored².

Handlers are not a part of the parsing subsystem itself, but instead register themselves with the parser from elsewhere in the architecture. Most commonly, handlers are built into the behaviors, tasks or the context manager of the character (about which more below). For example, consider a character that plays games. It has a behavior for asking whether the user would like to play a game with it. When that behavior is invoked, the behavior creates a perceptor that watches for a yes/no type of response from the user to that particular question and registers it with the parser.

implemented characters, however, one parser was sufficient since there was only one primary channel of communication in each case.

²In practice, there are “catch-all” handlers to avoid this eventuality.

```

<?annotation-begin?>
  <annotation>
    <event>
      <game-update id="nim">
        <game-begins/>
        <configuration>
          <position id="bowl-1" value="4"/>
          <position id="bowl-2" value="8"/>
          <position id="bowl-3" value="6"/>
        </configuration>
        <players>
          <player id="Harker"/>
          <player id="Patrick"/>
        </players>
        <players-who-can-move>
          <player id="Harker"/>
        </players-who-can-move>
      </game-update>
    </event>
  </annotation>
<?annotation-end?>

```

Figure 5.2: An annotation communicated by the MUD environment

If the user responds to the question, that handler is triggered, and may activate other behaviors for game-playing and then de-register itself with the parser.

Handlers are extremely flexible. In response to a communication, a handler can modify its own state, make contexts in the character's global context manager active or inactive, add or remove behaviors or tasks, modify the character's emotions or emotional influences on the character, or simply add information to the global context manager for future reference (as in the case of purely descriptive factual information). A handler's control over the character is limited only by its location in the architecture, so a handler associated with the global context manager is able to make a greater range of changes than a handler that is associated with a single behavior, which does not have visibility into the entire structure of the system.

Handlers match communications according to patterns that they provide to the parser when they register. The nature of these patterns depends on the type of

the parser. In the case of the MUD environment, for example, the patterns for the perceptrons are simply regular expressions, since MUDs communicate in highly-stylized forms of English, while the patterns for the annotation handlers are written in XPath, a standard query language for retrieving information from XML documents, since MUD annotations are written in XML. Each parser compares the communications against these patterns using its own matching mechanism³.

5.4 The context manager

The context manager is the primary mechanism by which the character maintains state information, both about itself and about the environment. It is used by the behavior engine to assist in determining what behaviors are legal, which are appropriate, and to provide data for certain behaviors to function (for example, planning a path from one location in the world to another). The three kinds of information it contains are: information about the structure of the environment, including the entities in the environment, their properties, and their organization; information about the relationships that the character has with those entities; and information about the internal state of the character itself.

5.4.1 Environment

In Chapter 4, we divided the objects of annotations into four main types: *entities*, *relationships*, *operations*, and *events*. As the character explores the environment and receives information about it from the communications system, the handlers are triggered. They in turn can call the context manager to request that it build records of these objects for future reference. For example, when a user visits a new page in the Web art gallery, the character receives an annotation describing the artworks displayed on that page, the artists of those works, the elements depicted, and so

³This is another reason for supporting multiple parsers; it would be possible, for example, to support both simple and more sophisticated English-language recognition systems within a single character by providing distinct parsers and handlers for trying different approaches to matching user utterances.

forth. The annotation handlers inform the context manager that there is a new location in the art gallery, and that this location contains these various elements. The context manager keeps a catalog of all of the entities the character encounters in the environment or that are referred to by the annotations it receives.

Much of this information is handled in a generic way by the context manager so, for example, it has no special knowledge of artworks or artists. It is left to the behaviors to reason about whether they are relevant and what use to make of them. However, some kinds of entities and events are privileged and the context manager does additional work when processing them.

The one of these that applies to all our experiments is the layout of the environment. As the character moves about in the environment, it builds up a map of discrete locations (e.g., locations in the MUD or URLs on a Web site) and the connections between them. Whenever the context manager is informed of the existence of a new location or a new connection between locations, it modifies a global map of the world. It also keeps a record of the physical locations of all the entities the character reports upon. In this way the character has a means of navigating or answering queries about the structure of the environment.

In addition to keeping records on the various entities of the environment, the context manager organizes them into *contexts* in order to preserve their relationships with one another. A context is a representation of certain aspects of the state of the environment. Contexts may be active or inactive, and may be added or destroyed by the character as it needs them.

For example, when the character moves into a new location, a context is created that represents what information the context manager has available about that location, including the entities that it contains (including other agents, objects in the inventory, abstract entities such as puzzles that take place in that location or games that can be played), what operations are enabled or forbidden in that location, the affective properties of the location, and so on.

There may be multiple contexts active at one time, and contexts may be nested within one another. For example, the context manager distinguishes between what are called *locations* and *areas* in the annotation language, where an area is a collection of

locations. Thus whenever the character is in a particular location in the environment, there is an active context representing information about that location, and it is a child of a broader context that contains information about the area in which the location is contained. A treasure hunter standing in the dungeon of a haunted castle may see an iron maiden and a rack; these are attributes of the immediate physical context. However, there is a pervasive aura of fear and danger throughout the entire castle, and so a separate context describing those properties exists and is also active as a broader context beyond the current location.

5.4.2 Self

The central mechanism the character has for representing its current state is the emotion system, which is connected to, although strictly speaking independent from, the context manager.

Knowledge of goals is represented only implicitly according to which tasks and behaviors are active. The tasks and individual behaviors may contain state information relevant to their own executions that is not directly available to the context manager.

5.4.3 Relationships

In the course of its existence, the character may enter into stateful relationships with certain entities in the environment, animate or otherwise. The context manager tracks these relationships. Some are trivial, such as the location of the character in the environment or the objects that the character is currently carrying. Others are more elaborate and depend on the kinds of activities that the environment supports. In the MUD game world, for example, there is a component of the MUD context manager that tracks all of the active puzzles in the game and what state those puzzles are known to be in. There is another component that maintains records on all of the games the environment provides for the character to play and, when the character is playing, what the current score is, what moves are available, and so forth.

The social aspects of relationships with other characters or agents, including the

user, are represented through the use of directed emotions such as *reproach* or *appreciation*; see the following discussion of the emotion system for an explanation.

The one kind of information about the character's relationship with human users that is maintained in the context manager is communications directly from the user. When a user says or does something that the character interprets as a command, the character adds it to a list of unfulfilled commands from the user. This list is used by the behavior engine in choosing certain behaviors, creating new behaviors and tasks, and removing them.

5.5 The emotion system

An explicit model of emotions in the architecture is interesting to us only insofar in that it is a useful abstraction for thinking about a certain set of ways in which a character's behaviors may be influenced. It is desirable that these influences should correspond intuitively to the influences of human emotions upon human behaviors. Our goal is not necessarily to reproduce a psychologically plausible model of emotion, but to build a system that has these advantages for us:

- Emotions are a vocabulary in which human beings are intuitively well-versed and it is natural to provide a means for describing the behaviors of our characters in terms of that vocabulary. Thus an explicit model of emotion with corresponding control mechanisms will make it easy (or easier) for the author of a character to create behaviors that correspond to the emotionally-influenced behaviors of living creatures.
- Describing behaviors in terms of emotions will tend to lead creators to produce more believable characters because it is easier to understand how groups of behaviors will interact with one another in terms of emotions than in terms of other, less intuitive, measures⁴.

⁴Such as in those systems that consider emotions as emergent properties of the architecture rather than modeling them explicitly, e.g., the work of Aaron Sloman [128].

- An explicit model of emotion will lend itself to association with the affective (emotional) annotations proposed by the environments in which the characters are intended to operate (see Chapter 4).

As mentioned, psychologically plausibility is not a goal of our model; there are a number of other research projects that are focused on this problem (e.g., [37, 104, 106, 111, 128, 149].) Like Reilly [113], we are more concerned with a mechanism that is easy for authors to use and which will straightforwardly influence a character's behavior in the ways in which they intend.

Here are some examples of the kinds of situations for which we want to be able to describe behaviors easily in terms of the emotions of the character:

1. Harker enters a dark laboratory, filled with strange alchemical equipment and unsettling devices. The laboratory is dominated by a huge, tentacled plant which springs to life as he draws closer. Its tentacles begin to writhe and reach out for him as a vast, gaping maw opens in anticipation of its next meal.
2. Harker and a companion get into a heated argument about how to solve a problem; later, his companion apologizes.
3. Harker is walking in a pleasant garden on a warm summery day. He finds a valuable coin lying on the path in front of him. While reaching for the coin, he steps on a thorn.
4. Harker is walking in a pleasant garden on a warm summery day. He inadvertently steps on a sharp thorn, and while looking down at the source of the pain, sees a valuable coin in the undergrowth.
5. While out walking, Harker recognizes an old friend crossing the street. A car with a careless driver nearly runs Harker's friend over, but he is able to pull his friend to safety in the nick of time.

5.5.1 The model of emotions

Picard [110] lists five requirements for a system that can be said to have emotions:

1. Emotional behavior;
2. Fast primary emotions;
3. Cognitively generated emotions;
4. Emotional experience: cognitive awareness, physiological awareness, and subjective feelings;
5. Body-mind interactions.

though, as she says, not every requirement need necessarily be met to achieve results in practice.

The emotion system we use in this architecture is based upon the model proposed by Ortony, Clore, and Collins [106] (often called simply the “OCC” model). It is a cognitive model in which emotions arise in response to an evaluation of situations and entities against some beliefs and goals possessed by the agent. As a result, the system in this thesis meets all but the second of Picard’s requirements, fast primary emotions, since in effect all emotions are the result of some sort of cognitive appraisal on the part of the agent. We will have more to say about this list in the analysis of the architecture in Chapter 6.

Ortony, Clore, and Collins: A brief summary

The OCC approach⁵ attempts to explain how emotions are generated via a cognitive model in which emotions are regarded as the results of appraisals of the state of the world as compared to the goals the agent holds. The theory is particularly appealing to computer scientists because it was explicitly designed to be implemented and tested computationally, and it has been used or adapted in several computational systems that use emotion [37, 113].

According to the theory, agents’ emotions are the product of a three-step process: the agent encounters a situation, the agent appraises the situation, and emotions arise

⁵Please note that this is a *very* brief summary of a quite elaborate model, many elements of which are simplified or ignored in our architecture. See [106] for the full treatment, including the complete list of emotions they propose.

in accordance with its appraisal. The theory provides a structure according to which a situation is appraised, and categorizes emotions as combinations of basic outputs from that structure.

More specifically, situations are divided into three types of entities: *events*, *agents*, and *objects*. The appraisal mechanism is organized into three corresponding types: events are appraised according to *goals*, agents according to *standards*, and objects according to *attitudes*.

Central to the appraisal of events is the *goal structure*, which may be envisioned as a lattice, connecting goals and subgoals that stretch from some small set of high-level concerns, such as “lead a happy life” or “please my parents,” to a wide range of lower-level and more immediate challenges, such as “graduate from college” or “get an A in CS121.”

Goals are broadly categorized into three types. *Active-pursuit* goals (or **A**-goals) represent things that the agent wants to get done. *Interest* goals (**I**-goals) are things the agent wants to see happen. They differ from **A**-goals in that they ordinarily represent things that the agent does not have any direct influence over or which are not usually actively pursued, although they may spawn **A**-goals whose purpose it is to improve the likelihood of their occurring. Preservation goals (to maintain one’s health, friendships, well-being, etc.) are an example of **I**-goals.

The third class of goals are the *Replenishment* goals or **R**-goals. They are distinguished from **A**-goals and **I**-goals principally in that they are goals that must be repeatedly and cyclically achieved, as opposed to **A**-goals that disappear when realized and **I**-goals that have no cyclical quality. Keeping the car filled with gasoline, emptying the mailbox, or washing the dirty dishes are examples of **R**-goals. The urgency of such goals supposedly increases over time since they were last satisfied.

In addition to goals, the appraisal structure contains two other components. First are *standards* by which agents are appraised, which represent not what the agent *wants* to be true but what the agent believes *ought* to be true about itself or others. Standards are necessary to permit the agent to evaluate behavior, social norms, and other regularities in the world that are at best implicitly related to its goals. An evil old man who goes about stealing candy from babies is likely to be appraised

negatively according to the standards of a heroic character, for example.

The third component of the appraisal structure are the agent's *attitudes*, according to which objects are appraised. Attitudes reflect the agent's liking or disliking of objects or entities independent of its standards or goals. Liking vanilla ice cream, Buster Keaton films, or time-domain reflectometry are examples of attitudes.

The valenced appraisal of the entity types according to their corresponding appraisal components is captured by three *intensity variables*. These variables are central to the appraisal process, and they moderate the strength of the agent's affective response to the situation. The variable associated with reaction to events is *desirability*; for agents, it is *praiseworthiness*, and for objects, *appealingness*.

The several triads proposed by the theory can be placed in one-to-one correspondence with one another:

Entity	Appraisal Component	Intensity Variable
events	goals	desirability
agents	standards	praiseworthiness
objects	attitudes	appealingness

Table 5.1: Entities and the ways in which they are appraised

Evaluation of a situation is performed with respect to all three components, and in fact multiple evaluations may occur, since there may be multiple appraisals that are appropriate for each entity.

Each appraisal of an event always occurs with respect to some *focal goal*. That goal and its subgoals each have some value to the agent, and the desirability of the event can be computed according to the values of those goals and the extent to which it facilitates or inhibits their completion. The more desirable an event, the more intense the positive event-based emotions; the more undesirable, the more intense the negative event-based emotions.

There may be several simultaneous evaluations of an event with widely different desirabilities according to different, possibly contradictory, goals. [106] provides the example of a man in financial trouble who learns that a favorite wealthy aunt has

died, leaving her money to him. This event is *desirable* with respect to the Active-pursuit goal of getting out of debt, but *undesirable* with respect to the Interest goal of maintaining the well-being of loved ones.

Also, the desirability of a goal's success may be disproportionate with the undesirability of its failure. Given a focal goal of taking a trip to Tahiti, it is desirable, but hardly cause for celebration, that the agent's car starts so that it can drive to the airport, but the failure of the car to start jeopardizes the entire trip and is highly undesirable — as undesirable as the failure of the focal goal itself.

Corresponding to the appraisal of events by their desirability, agents are appraised according to their praiseworthiness. The appraisal compares an agent's actions to certain standards, such as *one ought to do good* or *one ought to do one's best*. Failure to act can also be considered, so an agent may be blameworthy (that is, negatively praiseworthy), for example, for seeing that a child is about to injure itself and doing nothing to prevent it.

Finally, and most simply, objects are appraised by their appealingness according to the agent's attitudes.

5.5.2 Emotions in the architecture

We have chosen to implement eleven of the twenty-two emotions in the OCC model: joy, fear, satisfaction, fears-confirmed, relief, pride, reproach, appreciation, liking, and disliking. We have also added one emotion of our own, boredom. Table 5.2 summarizes these emotions.

In order to make use of the system of emotions in the annotations of the environment, we assume that there are certain general goals, desires, and attitudes that are shared by many or all agents in the environment, such as **preservation-of-well-being**. These are referred to in the affective annotations provided by the environment to help justify those annotations. Table 8.4.2 describes the general goals we employ, and specific instances are discussed in Chapters 8 and 7.

We now give a brief explanation of the individual emotions and the ways in which

Emotion Type	Description
Joy	An event makes it more likely that one of the agent's goals will succeed.
Fear	An event may occur that will make it more likely that one of the agent's goals will fail.
Fears-Confirmed	A goal that the agent feared would fail <i>did</i> fail.
Satisfaction	A goal that the agent believed would succeed <i>did</i> succeed.
Relief	A goal that the agent feared would fail succeeded.
Pride	The agent executes a behavior consistent with its standards of behavior.
Appreciation	Another agent acts in a way that the agent finds praiseworthy (in accordance with its standards).
Reproach	Another agent acts in a way that the agent finds blameworthy (inconsistent with its standards).
Anger	Another agent acts in a way that makes it more likely one of this agent's goals will fail.
Liking	Agent is contemplating an object (or other agent) toward which it has a positive attitude.
Disliking	Agent is contemplating an object (or other agent) toward which it has a negative attitude.
Boredom	(Non-OCC) The absence of any events is causing the failure of the agent's goal to be active.

Table 5.2: Emotions used by the architecture

they arise, after which we will describe the architectural mechanisms that create and manipulate them.

Event-Driven Emotions

The event-driven emotions are joy, fear, fears-confirmed, satisfaction, and relief. They are generated in response to *events* in the world, whose *desirability* is evaluated with respect to the character's goals.

JOY

Joy is a pleasurable emotion engendered by a desirable event. For example, a poor man who wins the lottery experiences joy. Note that this does not necessarily have to be a present event; it is possible to experience joy at the recollection of a past event or even at the anticipation of a future event. Joy is primarily influenced by desirability of the event, but according to the OCC model can also be influenced by its unexpectedness (winning the lottery may produce a more intensely joyful response than receiving an equally large, but anticipated, sum of money for another reason) and its proximity in time.

FEAR

Fear is what is called a *prospect-based emotion*, since it has not to do with the occurrence of an event but the contemplation of an undesirable event that may occur. A character standing on a bridge whose supports are snapping is likely to experience fear at the prospect of falling to its death. The intensity of fear is due principally to the undesirability of the contemplated event, but is also influenced by its perceived likelihood.

FEARS-CONFIRMED

Fears-confirmed is a *confirmation* emotion, meaning that it arises as the result of the occurrence of an undesirable event that the character had contemplated. That is, the emotion of fear about a possible undesirable event gives way to fears-confirmed when the event actually does occur. The intensity of the emotion can be influenced by the strength of the original fear emotion, so that the more afraid the character was before the event, the stronger the reaction after it happens. Other influences include the amount of effort expended by the character to avoid the event or prevent it from transpiring, and the degree to which the undesirable event was realized (the extent of a stockbroker's loss on the stock market, for example).

SATISFACTION

Satisfaction is the mirror image of fears-confirmed; it arises when a desirable event that a character has contemplated actually does occur. As with fears-confirmed, it is influenced by the strength of the original emotion (which in the OCC model is *hope*, although we do not explicitly use hope in the architecture), the amount of effort the character expended to bring the event about, and the degree to which the character was successful.

RELIEF

Relief is the alternative to fears-confirmed. It arises when an undesirable event the character has contemplated does *not* occur; that is, it is avoided or prevented. As with fear, intensity is influenced by the intensity of the original fear emotion, as well as the effort the character expended and the degree to which the undesirable event was avoided.

BOREDOM

In experimenting with our characters, it became clear that one kind of emotional attitude we wanted to be able to express was boredom. This was appropriate when the user was not acting, and it enabled the character to react, either by expressing boredom, offering suggestions of things to do, attempting to provide assistance to a user who might be confused about how to proceed, and so forth. In terms of the OCC model, we describe boredom as an event-driven emotion that occurs with the *absence* of any events, desirable or otherwise, which causes the failure of an implicit goal of the character to be actively engaged with the user in the world.

Agent-Driven Emotions

These emotions arise when the character evaluates agents (itself or other agents) according to its standards regarding how agents should act. Note that the term “agent” in this context can be interpreted very broadly, extending even to inanimate

entities, so that it is possible to be angry with the weather when it rains or with a rock when one trips over it.

PRIDE

Pride occurs when the character acts in a way that is consistent with its standards. A hero who rescues a child from a burning building is behaving consistently with his beliefs, and experiences a feeling of pride as a result. The intensity of the pride depends upon how praiseworthy the character perceives the action to be, how much of a deviation from ordinary and unremarkable behavior it is, and what the OCC model calls the *strength of the cognitive unit with the agent*, by which they mean the strength of the character's identification with the entity that performed the act. In most cases, the character is the actor producing the sense of pride, but it is possible for the character to be part of a larger agent (so that, for example, the character may feel pride when his team wins a football game because the character "identifies" with that team).

APPRECIATION

Appreciation is the emotion produced when another agent acts in a way that the character finds praiseworthy. The intensity of the emotion, like that of pride, depends upon the degree of praiseworthiness, and also upon the extent to which the behavior deviates from expected behavior.

REPROACH

Reproach is the opposite of appreciation, and occurs when another agent acts in a way that the character finds blameworthy (negatively praiseworthy).

ANGER

Anger is a compound emotion; it occurs when another agent acts in a way that produces an undesirable event. This leads to a sense of reproach toward that other agent, combined with displeasure about the event itself.

Object-Driven Emotions

These emotions arise when contemplating objects, which the character finds appealing or unappealing according to its attitudes. They can also be applied to other agents (as opposed to the *actions* of other agents, which meet or violate a character's standards).

LIKING

Liking is the emotion experienced when contemplating an object that the character finds appealing. Intensity is influenced by the degree to which the object is appealing, and can also be modified by the degree of familiarity with the object.

DISLIKING

Disliking is simply the opposite of liking an object.

5.5.3 Emotion influences

A character's emotions can be influenced both by aspects of the world and by the results of its own cogitation or action. A slap on the face, the rumble of an oncoming train, or an audit notice from the IRS are all examples of emotion stimuli, as are contemplating one's own failures or deciding to forgive an opponent. When a source of emotion is perceived by the character, the architecture creates an *emotion influence* to describe it. The influence is a persistent object in memory with the following properties:

- *Type* — the type of the emotion (e.g., fear)
- *Source* — the event, object, or agent itself (such as a creaking bridge)
- *Source-Type* — an event, object, or agent
- *Cause* — the reason the emotion is being generated by the source (such as a threat to the goal `preservation-of-well-being`)
- *Strength* — numerical value from 0.0–100.0

```

<location> ...
  <inventory> ...
    <object id="guardian">This huge...thing...is evidently meant
to guard the workshop. It has a bizarre combination of waving
leaves, tentacles, and mouths with very sharp, needle-like teeth,
and it appears somehow to be rooted in the hardwood floor. Since
it's also about eight feet tall, I wouldn't advise going anywhere
near it.
      <emotion-sources>
        <emotion-source source="guardian" source-type="object"
          cause="threat-to-self-preservation"
          type="FEAR" strength="40.0"/>
      </emotion-sources>
    </object>
  </inventory>
</location>

```

Figure 5.3: An object in the environment that is a source of emotions

Influences persist from one cycle to the next. Influences from events and agents, however, are assumed to be transient. A character frightened by a loud clap of thunder, or pleased at the result of the baseball game, will not continue to be influenced by that event for long. These influences gradually decay over time, weakening on each cycle, until they reach a strength of zero, at which point they are removed.

Figure 5.3 shows an example of an object in the environment that is a source of emotions. In this case from Chapter 8, there is a strange plant-creature guarding the laboratory in the adventure game. It is intended to provoke fear in characters that encounter it. The single emotion source it contains is a fairly high source of fear due to a threat to one of our assumed “standard” goals, **preservation-of-well-being**⁶.

The figure also points out some discrepancies between the OCC model and the implementation. The environment does not explicitly have the concept of a prospective event, so there is no way to indicate that something *might* take place that could affect the character. Thus the prospect-based emotions are associated instead with the other entities in the environment as the potential sources of these events. Notice

⁶The `emotion-source` annotation does not specify a source-type because it is already indicated by the fact that the guardian is an object.

in this case also that the guardian is treated as an object rather than as an agent; this is because it is not, strictly speaking, an intelligent and active creature, but only a static element of a puzzle that the character has to solve, and not capable of interacting in the ways that true agents in the MUD can interact.

As mentioned, influences can come both from the environment and from within the character itself. If the character chooses to execute a behavior that may change its own emotional outlook, as part of the process of execution that behavior can create a new influence which is fed to the emotion system and added to the list for consideration on subsequent cycles. The behavior emotion is produced through a self-generated emotion influence that is added to each cycle on which nothing occurs in the environment.

Emotions from the environment either come from annotations which will provide all of the above information needed to create the influence, or can come as the result of perceiving an aspect of the environment that does not have an annotation describing it. In those cases, we add a particular class of perceptor called emotion perceptors that recognize the ordinary environmental output (such as, for example, “Bob kicks you”) and the perceptor then provides a description of the influence to the emotion system.

5.5.4 Computation of emotions

On each cycle, the strength of every emotion is recalculated by the architecture. This is done by considering every existing emotion influence. First, the source and type of the influence are examined. If the source of the emotion is present to the character (for example, if the source is a location object, and the character is in that location), then the strength of that influence is added to a running total for influences of that type. When all the influences of one type (such as fear) are collected, they are passed through a translation function to determine what the strength of the emotion itself would be. The translation function is where the character’s personality comes into play; it is parameterized so that different characters have different responses to influences of varying types.

The translation function we use comes from [110], where Picard offers the following sigmoidal function to describe the intensity of an emotion as a function of its inputs:

$$y = \sum_x \frac{g}{1 + e^{-(x-x_0)/s}} + y_0 \quad (5.1)$$

where x is the input stimulus and y is the output emotion strength. The other factors control the shape of the curve according to the desired properties of the emotion and of the agent's personality. g modifies the gain of the function, changing the amplitude of the curve. Picard suggests that this can be tied to the agent's state of arousal, so that an agent in a high state of arousal may have a strong reaction to a mild stimulus, while agents in low states of arousal may be little affected even by profound stimuli. x_0 shifts the curve to the left or right, changing the strength of the stimulus needed to provoke a response, while y_0 shifts the curve up or down. s controls the slope of the curve; low values of s produce a curve has a sharp, nearly-vertical transition region, corresponding to agents that undergo radical emotional shifts at narrow boundary regions, while a higher value of s produces a more gradual curve.

For each emotion, the character defines a different parameterization of this general function. A heroic, fearless character, for example, may use a function such as shown in Figure 5.4 for translating the strength of all fear influences into its own fear; so it takes an enormous total strength of fear influences to produce much fear in the character. A timid character might use a function such as described in Figure 5.5.

5.5.5 Expressions

In order that a character's behaviors be believable, not only must the individual behaviors chosen on each execution cycle seem believable themselves, but the sequence of behaviors taken as a whole must be believable. Unfortunately, architectures in which behaviors are chosen one at a time in an endless loop, always choosing the best behavior for the current moment in time, can produce good individual behaviors but sequences that seem incoherent. A character that is both very angry and very afraid, for example, might toggle back and forth between shouting and threatening the agent that has made it angry and cowering and weeping in fear at the prospect of

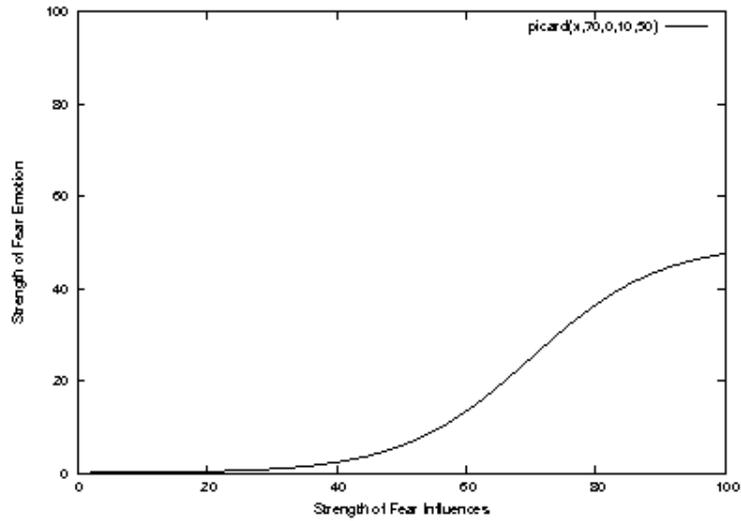


Figure 5.4: Translation of fear influences to fear emotion for heroic character

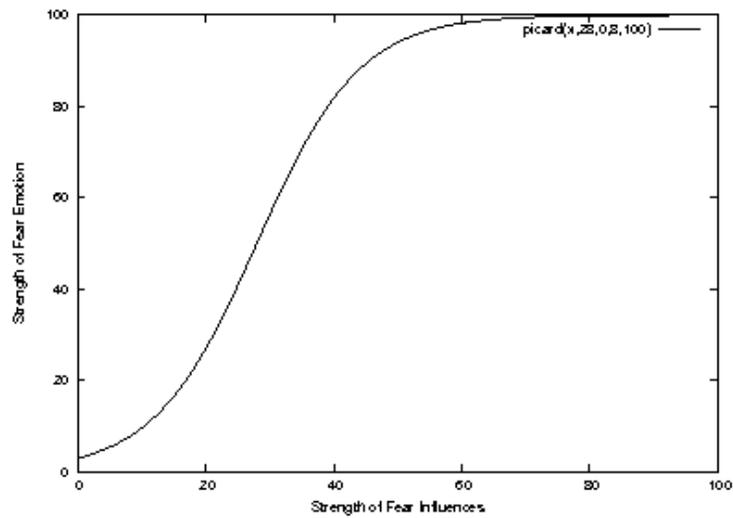


Figure 5.5: Translation of fear influences to fear emotion for timid character

some terrible event. This is not believable behavior in an ordinary human being, as no matter how intense the different emotions are, under usual circumstances one will dominate for a time, and then another will come to the fore, as will the expressions of those emotions, rather than bouncing back and forth from one action to the next. As discussed in section 2.2.6, Sengers [125] calls this the problem of *schizophrenia* in agent architectures.

One of the ways in which the architecture addresses this problem is by introducing the mechanism of *expressions*. On each cycle, the character's set of emotions and their strengths are evaluated, and on the basis of that distribution, a single dominant expression is chosen. This expression, in turn, influences the set of behaviors that can be considered for execution. An expression is intended to represent the dominant theme across the character's emotions, though expressions are not necessarily in one-to-one correspondence with any particular emotion. That is, the set of expressions is peculiar to a particular character, while the set of emotions applies across any character implemented in the architecture; all characters have a fear emotion, but one might have expressions of petulance and irritation, while another might use expressions of balminess or perhaps obsessive introversion. When a new expression is chosen, the system explicitly moves into a *transition* expression between the old one and the new (such as `transition-from-petulance-to-anger-expression`), and there are certain behaviors that are only permissible during these transitions.

There are two major reasons for using expressions. The first is that they do provide a way to minimize the tendency to bounce back and forth between conveying one emotion and another; instead, one dominant expression of emotional state is chosen for a time, and then an explicit opportunity for transition into another expression is given, before the new expression is adapted.

The second reason is that expressions are another way of describing when behaviors are appropriate, in addition to emotions. A behavior may be appropriate when a character's dominant expression is one of, for example, petulance (perhaps this is the way the character expresses moderate anger combined with mild fear when other emotions are weak). They may also be entirely inconsistent with emotions, so a certain character may hold an expression of pleased calm when, in fact, enraged.

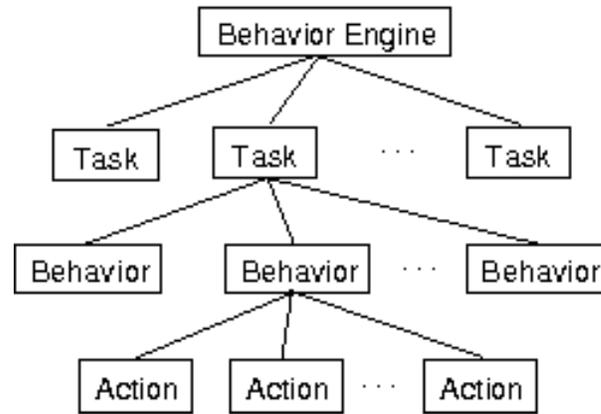


Figure 5.6: Hierarchy of the behavior engine: tasks, behaviors, and actions

Expressions are essentially a way of grouping portions of the space defined by all the character’s emotions.

5.6 The behavior engine

At long last we come to the behavior engine, the part of the architecture responsible for selecting and executing behaviors. Figure 5.6 shows the structure of the engine.

5.6.1 Tasks

At the highest level, the possibilities for action are organized into *tasks*. A task corresponds to a group of similar behaviors or behaviors that are collectively useful for accomplishing a particular end. In the case of the Harker character in Chapter 8, there is a **Following** task, which consists of various movement behaviors the character uses in order to stay in the same location as the user.

A task may contain only a single behavior or it may have a large group of behaviors. Individual behaviors may conceivably appear in multiple tasks; the behaviors associated with movement, for example, appear in several tasks, as they are applied for different purposes in those tasks.

Tasks are decoupled from one another; they do not have “awareness” of one another. In this way it is relatively straightforward to introduce or remove tasks from the character without raising complex integration issues. It also permits the creation of new tasks during execution whose properties are dependent upon information provided by the annotations in that environment.

Most tasks are persistent, intended to operate throughout the lifetime of the character. Others are temporary by design. Another example from the Harker character is the `RetrieveItem` task. When the user asks Harker to fetch an object that has been seen in the environment but is not in the current location, Harker responds by creating a new task designed specifically to solve that problem. It contains a sequence of behaviors first for going to that location⁷, then picking up the object, then returning to the user’s location, and finally giving it to the user. Once complete, the task removes itself from the engine.

Creating these temporary tasks is the way in which the character architecture can perform a kind of rudimentary planning. The structure of the temporary task, the parameters of its behaviors, represent the character’s plan for addressing a particular need. It is also a mechanism for implicitly creating goals, which are not explicit in the architecture. A task can be created dynamically which contains the behaviors and tests necessary to execute a new goal, and eliminated when that goal is satisfied.

Some tasks are marked as *uninterruptible*, or may be uninterruptible at some times. The behaviors of uninterruptible tasks are always chosen by the behavior engine. Only one active task may be marked uninterruptible at any one time, to avoid contention.

Tasks are each given a type; in the current architecture the two types are *functional* and *believability-enhancing*. The purpose of assigning types is to make it possible for the behavior engine to ignore certain types tasks under conditions in which they may not be appropriate; for example, a character that is following a rapidly-moving user does not need to consider the believability-enhancing tasks but should instead

⁷The character queries the context manager, asking it to return a sequence of directions for reaching the location of the object from the current location, and a simple path planning algorithm computes the course, if one is possible. Otherwise, the character responds that it doesn’t know how to make the trip.

concentrate on the functional tasks (and specifically on following the user).

Each task may have some state information associated with it. The `RetrieveItem` task, for example, tracks the object to be retrieved and the plan for reaching the object's location and traveling back to the user's location. The `SolvePuzzle` task records the puzzle object, whether it has been solved, the current step in its solution and current state.

5.6.2 Behaviors

A character is defined principally by the set of *behaviors* that it can exhibit. Behaviors correspond conceptually to single meaningful actions in the environment or, put another way, a behavior is intended to communicate a single idea to the viewer. Smiling, dancing, opening a door, answering a question, picking up an object from the floor — these are all examples of behaviors. The character begins with certain behaviors built into it, and others may be added or removed according to the annotations the agent receives from the environment.

Every behavior defines some *preconditions*, a *fitness function*, and some set of associated actions into which it will be translated if selected.

When the behavior engine runs, it queries each task for a recommended behavior. The individual tasks, in turn, query each of their behaviors, asking for a numerical rating of how appropriate that behavior is in the current cycle. The first thing the behavior does is test its preconditions to see whether it is legal to execute. The preconditions may be arbitrary tests on the context manager, the values of the emotions, the current expression, or any state information associated with the behavior or the task. Frequently, the behavior should always be considered and the precondition test simply returns true. A simple counterexample are expression transition behaviors that should only be considered when, for example, the character is transitioning from an expression of boredom to fear.

If the preconditions are met, the behavior computes its fitness. Again, the fitness function may make arbitrary queries of the context manager, emotions, etc. In practice, the architecture defines a set of parameterizable functions for use here, making

it easier to author new behaviors and to promote some consistency between the ways in which different behaviors rate themselves. For determining fitness with respect to emotions, for example, there are `LinearFit`, `ExponentialFit`, `WeightedAverageFit`, `BellCurveFit` and other such fitness functions. Figure 5.7 shows how the fitnesses of very simple emotional response behaviors are computed by various fitness functions using the character's reproach emotion.

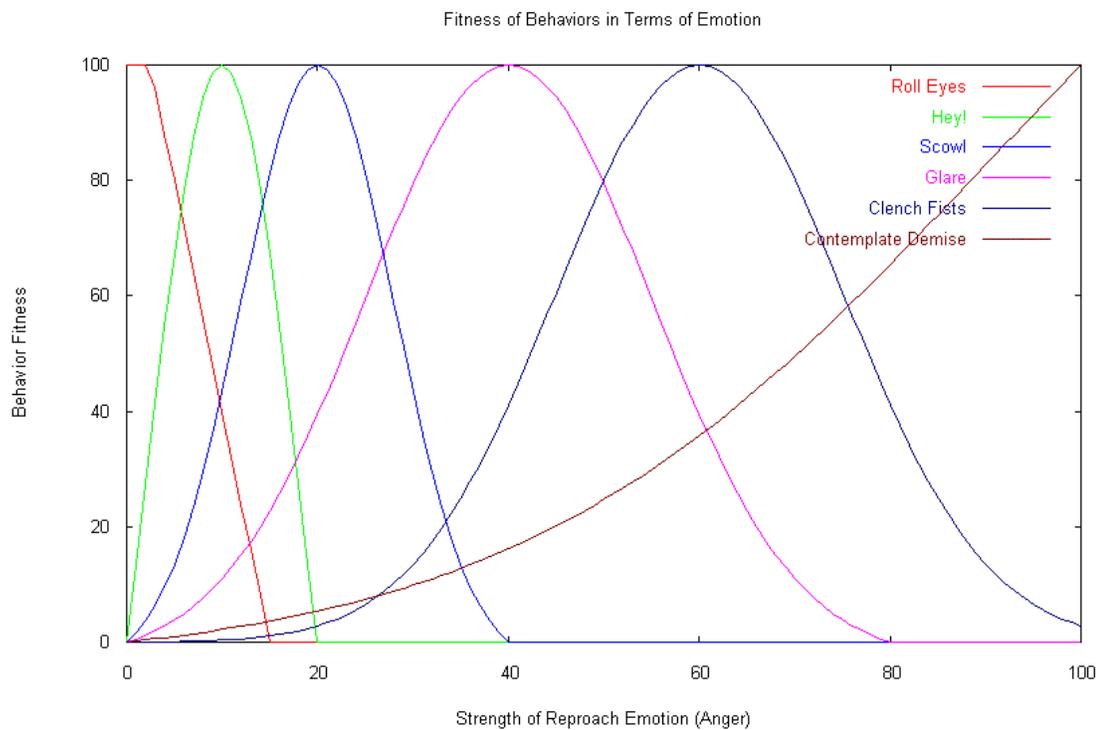


Figure 5.7: Fitness of several behaviors in terms of reproach emotion

Each behavior has associated with it one or more actions. A behavior may have multiple actions if we wish to vary slightly the way the behavior is executed in the world, but these variations do not fundamentally change what the behavior does. This provides a mechanism for having, for example, several ways of yawning without creating several distinct yawning behaviors. Instead we have one behavior with several possible actions.

The group of actions may be organized. The architecture provides three types

of groups: the **ActionBag**, from which actions are chosen randomly; the (poorly named, in retrospect) **ActionSet**, in which actions are chosen according to a bell-curve distribution on a single variable (usually an emotion), **ActionRange**, where each action defines a part of the range of an input variable as the values where it is appropriate (though multiple actions may overlap the same range, in which case a random selection is made between them), and the **ActionSequence**, in which actions are listed in an ordered sequence, so that when the behavior is translated into an action, the next unused action of the sequence is chosen.

Behaviors may also have associated with them annotation handlers or perceptors which they add to the parser when they are created. This is permitted so that a character with a behavior that has specific circumstances under which it should be active or inactive or whose nature should change if something specific occurs in the environment can take those steps.

There are several options that may be set on behaviors. The first of these is a *recency penalty* function. When a behavior is executed, it is marked with a time stamp. The recency penalty takes the difference between current time and the last time the behavior was executed, and returns a penalty on the fitness function for the behavior. The purpose is to offer a generic way of weighting some behaviors so that they do not occur often in rapid succession. The function may take any form and is defined when the behavior is created.

Behaviors may also be marked, either as *transient* or *terminal*. A transient behavior is only executed once and then is destroyed (it is removed from the list of behaviors for its task). A terminal behavior is used when a task is created that is meant to execute a sequence of behaviors and then to end (such as the **RetrieveItem** task). When the behavior marked as terminal is executed, it is the notice to the behavior engine that all behaviors in the task have been completed, and the engine destroys that task.

5.6.3 Actions

Once the engine has selected a behavior, it informs the behavior object that it has been chosen. The behavior object responds by returning an action object that describes how to execute the behavior in the character's environment. That is, a `Boredom` behavior may return an action containing the string `yawn` or `emote Harker looks around.` which can be interpreted by the environment.

When a behavior creates an action object, it may also associate with the action a special handler called an *action processor*. When the character executes the behavior and passes the action to the environment, if there is an action processor, it will call the processor object. By this callback mechanism it is possible to do cleanup operations or make adjustments that correspond to changes in the character's state as a result of performing the behavior. This might include deleting behaviors or tasks, creating emotion influences, changing state information in the behavior or the behavior's task, and so forth.

It is necessary to divorce behaviors, which are abstract and apart from the environment, from actions, which are concrete and executed within the environment, because the nature of the environment or the particular details of execution of an action may change over time. Our MUD-based agent needs to be able to go from one MUD, where for instance the action for saying something loudly is `shout` to another where the same effect is achieved with `yell` without this minor difference requiring modification of the underlying behaviors themselves. More significantly, this allows the basic personality behaviors of an agent to be reused across radically different environments (a MUD and the Web in our demonstrations) with only the corresponding actions needing to be changed.

1. Parse all new communications through the annotation handlers and perceptrons.
2. Recompute emotions based on active emotional influences.
3. Determine the appropriate current expression.
4. Choose a behavior for this cycle:
 1. Query each task for a recommended behavior.
 2. Each task queries each of its behaviors for its evaluation of its fitness in the current circumstances.
 3. Task chooses among the behaviors with highest fitness.
 4. Behavior engine chooses among task recommendations with highest overall fitness and consistent with additional factors the engine imposes.
5. Convert behavior into an action that can be executed in the environment.
6. Send the action to the environment.
7. If there is a post-action processor associated with the behavior, execute it.

Figure 5.8: The main sense-think-act cycle of the architecture

Chapter 6

Analysis of the Architecture

This dissertation introduces the idea that animate characters can benefit from knowledge embedded in the environment as annotations. To explore that idea, we have created characters and annotations as well as environments in which they interact. In order to implement our characters, we chose to build our own animate character architecture.

Chapter 5 describes the character architecture we've built. In this chapter, we discuss why it was constructed as it was, and how it implements the qualities of animate characters we offered in Chapter 2.

6.1 Goals of our architecture

Our primary goal in this thesis was to demonstrate the value of the annotation approach. As a result, we wanted an architecture that would make such a demonstration clear and compelling. While many other existing systems could conceivably have been adapted to make use of our annotations, we felt that the additional complexity and confusion associated with such reuse would only have confounded the basic issue.

The character architecture needed to do two things for us. First, it had to be capable of producing successful animate characters. Second, it had to be suitable for using those characters in our annotated environments. Broadly, there were four main criteria we needed to satisfy: simplicity, domain independence, believability,

and annotation sensitivity.

6.1.1 Simplicity

Our first criterion was that the architecture be relatively simple. Since the overall goal is to demonstrate how annotations can be useful to a character, the more straightforward the architecture is the more clearly we can show how the annotations affect its deliberations and the quality of its behaviors. As architectures become more complex and sophisticated, they also become more opaque, and it becomes increasingly difficult to follow the internal processes that lead to behavior. Simplicity lends itself to clarity in examining how the annotations affect behavior.

Simpler systems are also likely to be the most commonly used with annotations. One of the arguments in favor of the annotation approach is that there will be many characters interacting with the annotated environment, and that therefore it makes sense to distribute certain knowledge in the environment only once, rather than replicating it many different characters. However, as the number of characters grows, the sophistication of the average character is likely to drop.

The most obvious existing case for applying annotations is the current crop of massively multi-player role-playing games (MMORPGs), enormous virtual worlds that are capable of supporting thousands of simultaneous players. The level of intelligence of the interactive non-player characters is extremely low, and the addition of a layer of annotations would significantly improve their believability as well as their task capabilities. However, the number of these creatures is enormous as well (perhaps an order of magnitude above the number of players), and therefore the kinds of resources that can be devoted to a single experimental research architecture are simply not available to each agent. Therefore, even if augmented with a capacity for reading and making use of annotations, these characters would still necessarily be driven by fairly simple architectures. Thus simplicity is also beneficial in arguing for the practicality of the annotation approach.

A simple architecture is also, in some sense, the worst case. It has the most constrained abilities. We have seen (and will discuss again in Chapter 9) that even

humans routinely make use of knowledge embedded in the environment, and we can reasonably expect that the more sophisticated the agent, the more sophisticated the knowledge can be and the broader the range of applications to which it can be put. So rather than experimenting with more elaborate traditional agent architectures (e.g. [5, 24, 75, 97, 127]), we wish to show that even relatively simple characters can benefit significantly from annotations, and that even in these cases the interactions between the characters and the annotations are not trivial master-slave relationships but negotiations that lead to interesting results.

6.1.2 Domain independence

The second criterion is that the architecture be domain independent. It should be possible to author characters that can interact in domains with significantly different content and different interfaces. We set out to build two contrasting environments, a Web space that was primarily visual and disembodied, and an adventure game in a text-based and object-oriented MUD. We needed an architecture that would make it possible to create characters for each environment.

Domain independence implies two requirements. First, that there is a clear division between “mind” and “body.” That is, the elements that are responsible for memory, abstract reasoning, and decision-making should be separated as completely and cleanly as possible from the mechanisms that perceive and act in the environment. This permits us to replace one body with another — to change the details of perception and action without having to make significant changes to the mind of a particular character. It is desirable that we could move a character from one MUD, for example, to another with slightly different formatting and commands without having to alter its mind. We have argued the importance of this division elsewhere [57].

Second, there should be minimal assumptions about how the environment works. The architecture should not require that actions be atomic, that they are guaranteed to succeed, that the environment be static, and so forth. The one assumption of this sort that is built into our architecture is the concept of *locations*, that the environment

is divisible into discrete locations that can be identified, though the granularity of these locations is configurable. The discussion in Section 8.2.1 about the properties of MUD environments gives a sense of the broad range of worlds in which the architecture should be able to operate.

This is not a particularly difficult criterion for most agent architectures to satisfy, as generally they are designed with domain independence in mind. (Some systems do make stronger assumptions about the nature of the environment than others so, for example, the subsumption architecture of Brooks ([25, 26, 24] has a strong bias towards processing continuous low-level signals rather than high-level representational information.)

6.1.3 Believability

Annotations as a mechanism for conveying information about the environment apply equally well to general-purpose agent architectures as they do to animate characters. However, our goal is specifically to explore how they can benefit characters dramatically; that is, how they can aid characters in being more effective as characters. Our third criterion is that the architecture be able effectively to implement not just agents, but animate characters; the distinction is made clear in the discussion of qualities of character of Chapter 2.

Notice that this does not preclude the use of any one of a range of powerful and popular agent architectures, such as Soar [75, 119], Icarus [127], or Prodigy [97]. However, much of what we regard as critical to successful animate characters does not exist in these kinds of systems. They do not have as an architectural component any model of emotion, for example, nor of personality, social relations, etc. Since these are precisely the kinds of information that our annotations discuss, it seems to us appropriate to prefer an architecture into which these elements were incorporated by original design, rather than one in which they will have to be grafted on.

Moreover, the architecture should be organized in such a way that it is relatively easy to a change a character's personality or to create a new personality. Given one character of certain functionality, it should be straightforward to modify it to produce

another with different emotional reactions to the environment, for example. That is to say, the qualities of character, which are independent of the behaviors of the character, should also be modifiable without necessitating changes to those behaviors. It should also be possible to modify personality and emotion explicitly. Some architectures that explore emotion, such as MINDER [111, 128], regard emotions as emergent properties of the interactions of the components of the system. This is precisely the sort of thing we wish to avoid; while it may be true for real biological systems, and therefore worthwhile to model in an agent system, it would make the authoring of dramatic characters exceptionally difficult. We want an architecture that makes it easy to build a recognizable personality and easy to transform that personality into another one.

There are several architectures that have been developed specifically for authoring characters (e.g., [6, 9, 18, 21, 20, 56, 85, 92, 116]); by far the most extensively studied is Hap, developed by the Oz project at Carnegie Mellon University [15, 13, 88, 87], which subsequently extended it with an emotion system, Em [114, 113] and mechanisms for improving behavior expression and transitions among behaviors [125]. Hap certainly satisfies our needs as regards believability and domain independence (it also was used in both text-based and graphical environments), but is considerably more complex than we needed for our experiments, and also did not provide the support we needed for our final criterion, annotation sensitivity.

6.1.4 Annotation sensitivity

The final and most obvious criterion is that the architecture support the handling of annotations. It should have mechanisms for recognizing, processing, storing, and manipulating annotations. It should enable a character builder to add handlers for new annotation types, to tie the perception of those annotations to reactions in the architecture, to store those annotation in an internal database, and make queries and manipulations of that database as a part of the decision-making process.

Ideally, by “annotation sensitivity” we do not mean simply the vocabularies we have chosen for our environment, or even the representation language that we use

(XML), but should enable designers to add handlers for a variety of languages without particular difficulties.

6.1.5 Building our own architecture

We set out with four criteria for an architecture: simplicity, domain independence, support for believability, and annotation sensitivity. Moreover, the resulting system should be capable of satisfying each of the qualities of animate characters (both the qualities of character and the qualities of animacy) described in Chapter 2, a summary of which we repeat in Table 6.1.

No existing systems met all of these criteria. There was also the strong suspicion that any advantage we would gain through the use of an existing system would be more than balanced by the need for significant modifications to make it suitable for the experiments we planned. As a result, rather than adapting another architecture to fit our needs, we chose instead to implement a smaller but tailored system that had precisely the qualities we were seeking. This had the added benefit of allowing us to explore some character design decisions of interest to us but which had not been made elsewhere.

6.2 The approach to analysis

Our arguments in favor of the phrase *animate character* notwithstanding, what we have described in Chapter 5 is a type of agent architecture, of which many have been developed since the 1950's. The difficulty in evaluating such architectures is that, unlike many research products in AI, they are *integrated* systems, with many interacting components, and as Langley and Laird observe [79], this makes evaluation more difficult than for individual structures and methods. Partly as a result of this difficulty, there are even now no standard formal measures for evaluating agent architectures, although there has been considerable discussion in recent years on how to approach this problem (e.g., [55, 74, 79, 135, 140, 147, 148]) and there is even a pending book specifically on the problem of evaluating embodied characters [107].

Nevertheless, it is possible to make qualitative measures across some generally-agreed upon dimensions for these systems. Generality, versatility, taskability, rationality, optimality, efficiency, and scalability is a representative set offered by [79]. They offer in some sense a crude first approximation to the general goal, which the production of agent architectures that are “intelligent,” although, of course, there is considerable debate about precisely what that means.

This problem is compounded for character architectures, where the goal is not necessarily to produce intelligence but instead to produce characters that are “believable.” Like intelligence, believability is an ill-defined term. However, while it may be possible to develop some objective standards for measuring or comparing intelligence of agents, believability is also a highly subjective concept, and depends subtly and critically upon qualities unique to each observer, such as personality, mood, social and cultural background, knowledge of the domain, context of the interaction, and so on. What is quite believable to one observer may seem highly implausible and unreal to another, which is one reason why, for example, adults rarely care to watch the same television programs as their very young children. Formal attempts have been made to assess the believability of animate characters (see [114, 113] for an example) but have been subject to considerable criticism for the very reasons mentioned above.

We are faced with an additional difficulty in the evaluation of our architecture. The entire point of the dissertation is to examine how an animate character can benefit through the use of annotations in the environment. We do not claim, and have no particular interest in claiming, that our architecture is superior to many others along the dimensions described above, or even superior with respect to the qualities offered in Chapter 2. Instead, we simply wish to argue that it is a reasonable example of the broad class of possible animate character architectures for exploring the issues surrounding annotations. We are agnostic about *which* character architectures interact with an annotated environment; in fact, one of the main reasons for taking the annotation approach is to support character architectures of various strengths and goals. However, we must have an example¹, so we have contrived to produce a simple

¹As Lincoln said, when urged to replace George McClellan as the General-in-Chief of the Union armies, “Well, *anybody* will do for you...but I must have *somebody*.”

and standard one.

We laid the groundwork for justifying this “reasonable example” claim in Chapter 2, where we presented a list of qualities of animate characters that are generally accepted by the research community. That is, any reasonable instance of an animate character architecture we might encounter (or, more to the point, that an annotated environment might encounter) should be expected to have those qualities.

In the rest of this chapter, we will examine our architecture through the lens of that list from Chapter 2. Our intent is to show that it implements them sufficiently well to be thought of as this reasonable example of an animate character architecture. We will then use this architecture to implement characters for interacting in the domains of Chapters 7 and 8, where we will argue that the kinds of annotations we are exploring are precisely the kinds that will be meaningful and useful for other builders of such environments for characters.

There is another reason why we are taking this approach. An obvious alternative would be to attempt formal tests with users who would interact with our characters in our environments, and evaluate for us how believable they are (or how effective, how useful, how recognizably emotional — there are many such questions that would be interesting to ask). However, results of such a test would not significantly strengthen the central argument of the dissertation. Demonstrating that with our architecture we can build specific personalities that perform well in our example environments does not necessarily mean that other personalities would do as well, or that other environments would be as congenial. We want to argue that annotations are a mechanism that permits characters to enhance their believability. Showing that our architecture itself produces believable characters doesn’t advance that argument.

6.3 Qualities of character

Just as the larger components of the architecture are built from smaller and more concrete elements, so we will examine here the qualities of character beginning with the more fundamental (emotion and social abilities) and proceeding to the more general (personality and believability) as shown in 6.1, a slight deviation from the

Quality	Description
Emotion	Apparently having desires and feelings about the world with respect to those desires.
Social Relations	The capacity to react to other characters or people, and the ways in which behaviors are chosen to do so.
Expressiveness	Effectively conveying the designer's intention behind the action to the audience.
Personality	Possessing a recognizably coherent set of traits controlling the character's behavior.
Individuality	Being distinct from all others through the kinds of things a character does, the ways in which they are done, and the frequencies with which they are expressed.
Believability	Giving the appearance of being alive; being something the audience cares for or about. Behaving in a way that the audience recognizes as like life in a dramatic sense.
Goal-directedness	Giving the appearance of having goals and acting to fulfill them.
Reactivity	Responding in a timely way to what is happening in the world.
Situatedness	Acting in a way that reflects an apparent understanding of the character's current context (physical, social, conversational, affective, etc.).
Broad Capabilities	Being "complete" in the sense of having, to some degree, the major components of animacy — being able to sense, think, act (hear, see, speak, move, feel, etc.).

Table 6.1: Qualities of animate characters (reorganized)

order presented in Chapter 2.

6.3.1 Emotion

An effective character must act in such a way so as to convey to an observer that it has emotions, those emotions are influenced by what happens in the world, and that it acts in accordance with them. Note that this does not necessarily mean that the character architecture has a concept of emotions, only that they are recognizable in the observed behavior of the character.

Emotions are a shorthand that people use to describe complex cognitive states within ourselves. We use this shorthand when thinking about the reasons why we act in the world or react to it as we do. Since we are very comfortable with using these terms for describing the behaviors of real people, it is entirely natural that the author of a character should want to be able to use them when describing its behaviors.

The architecture supports this way of thinking in several ways. First, it contains an explicit and extensible model of emotions, based on the model of Ortony, Clore, and Collins [106]. The emotional state of the character is made up of this set of distinct emotions, each of which has a valenced value. One of the ways in which the fitness of a character's behaviors can be described is in terms of the character's emotional state.

In addition to the current emotional state, the architecture also tracks all of the influences on the character that lead to that state. A character is not simply angry, for no reason; it may be angry because, for example, it has just been struck by another agent. It may be angry because the picnic has been rained out, a favorite heirloom has been broken, it has slammed its finger in the door, or perhaps simply being tired makes it more irascible. Thus influences can be created either as a side effect of the execution of the character's own behaviors, or as a result of perceptions of events in the environment. The architecture keeps a separate record for each influence, and combines them according to the character's personality (see below) to produce the current emotional state. Both the state and the influences can be used in choosing behavior. Lastly, the architecture keeps a history of past states and influences so that they can also be used in choosing how to behave.

A character's designer has, through experience and introspection, an intuitive understanding of how emotions are expressed through behavior and how emotions are influenced by events in the world. The designer understands at that level how the character's emotions should be expressed and influenced. The architecture allows perceptors to create influences and influences to affect the fitness of behaviors in precisely the terms that the designer is familiar with. By incorporating emotions in the architecture, we encourage the creation of characters that are effective at expressing emotion.

6.3.2 Social Relations

Our feeling is that characters in open-ended interactions are most effective when they are acting in subservient roles; that is to say, when they are following the user's lead and occasionally intervening. Since our interest has tended to focus on characters that act as guides, companions, assistants, etc., our emphasis has been on the interaction between a single character and a single user, rather than on character-character or character-many user interactions. Thus the architecture has fairly limited support for social relations. It comes in two forms.

First is through a subset of the emotions that focuses on relationships with other agents. The ANGER, APPRECIATION, and REPROACH emotions model the character's stance toward some other agent². This is commonly the user, although other agents in the environment can give rise to these emotions as well. The emotion influences indicate which other agent is the source of an emotion so if, for example, another character were to provide a valuable clue to our character, it could add an influence on its sense of APPRECIATION toward that character.

There is also a basic mechanism for tracking the significant interactions between the user and the character. When the user utters some dialog that should provoke a non-trivial response in the character (as when, for example, the user asks for directions to some location, tells the character to fetch something from elsewhere, or asks for assistance with a puzzle) that utterance is added to a dialog history that the character can use when choosing subsequent behaviors.

Thus the character's author can model social relationships through the creation of emotion influences, the influence of those emotions on behaviors, and the selection and reaction to certain kinds of dialog from the user.

6.3.3 Expressiveness

An expressive character, in this context, is not one that is loudly demonstrative or that makes broad, sweeping gestures. Expressiveness means the capacity to effectively

²These are only a subset of the emotions proposed by Ortony, Clore, and Collins; there are others that we chose not to use.

convey to the audience the behavior that the character has chosen. As Sengers has observed [125], it isn't what your character chooses to do that is important, it's what the audience perceives the character to be doing that matters.

While an important quality of animate characters, expressiveness is to a large extent determined by the nature of the behaviors that the character's author decides to create, rather than by the properties of the architecture. The architecture is capable of supporting multiple simultaneous channels of communication between the character and the environment, and it does not place restrictions upon the kinds of commands that the character can execute; as we argued, domain independence was an important consideration for our architecture. However, this being the case, it does not have a mechanism for favoring certain kinds of "more expressive" behaviors over others.

The architecture does provide some assistance, however. On each cycle, the emotion influences are used to generate the strengths of the emotions themselves. The emotions are then used (together with other state information, if the designer wishes) to choose a single dominant *expression* for the character. The fitness of behaviors may then depend on the expression as well as on the emotional state. It is up to the designer of the character to provide the selection of expressions and the rules for deciding which one is dominant at what time. This enables the designer to translate internal emotions into external expressions that may be quite different so, for example, a very fearful character may exhibit an expression of anger instead; a combination of weakly active emotions may lead to an expression of apathy or indifference (or whatever other terms the designer wishes to employ).

Also, following an idea of Sengers [125], if a new expression is chosen that is different from the expression on the previous cycle, the character temporarily moves through a *transition* expression from the one to the other. This allows certain behaviors to become active or at least preferred during these transitions, which permits more smooth movement between one set of behaviors and another. A character moving from an expression of apathy to one of intense agitation, for example, would be well served to exhibit some intermediate behaviors that demonstrate that movement.

The expression mechanism assists the designer in clearly choosing a single dominant expression of the character's personality at any one time, and organizing behaviors according to how well they support that expression. This helps to make the character more appropriately expressive; not in the individual behaviors, which are up to the designer to create, but over long sequences of behaviors, by choosing behaviors that follow a single theme rather than rapidly switching among several.

6.3.4 Personality

From the point of view of the architecture, the definition of a personality is tantamount to the definition of an individual, but there are some distinctions. A personality can be thought of as more abstract, describing the broad properties of a character, while its individual traits distinguish it from all others, even those sharing the same personality. Bugs Bunny was modeled on Groucho Marx, for example; while their personalities are the same, they are recognizably distinct individuals.

The principal way in which personality is broadly defined is through the parameterization of the function that translates emotion influences into the values of the emotions. Recall from Section 5.5.4 that the strength of an emotion x on each cycle is computed as a function of all of the influences on that emotion, In_x , according to

$$Em(x) = \sum_{i \in In_x} \frac{g}{1 + e^{-(i-i_0)/s}} + y_0 \quad (6.1)$$

The choices of parameters define how an individual responds to particular emotional stimuli; whether, for example, fear influences actually lead the character to be afraid at all, or whether even the tiniest insult leads to an explosion of rage. By varying this function an entirely new personality can be produced, even without making any other changes to the character.

6.3.5 Individuality

Given that the broad outlines of a character are defined by its personality, what makes it a unique individual? There are several supports for this in the architecture.

Most obviously, through the set of behaviors that the character can execute and the ways in which they are performed. One individual can sing, another can dance; and even within those categories, each will sing or dance in individual ways, which are expressed through the ways that the behaviors are authored. In the text-based MUD environment, one character may sing by sending *:warbles an Italian aria cheerfully* while another, even with an identical personality type, transmits *:belts a Broadway tune at the top of his lungs*. The fitness of an individual behavior can be described, in part, by the values of current emotions, by the active emotion influences, by the state of social relations, and by the dominant expression, all of which also uniquely identify the character as an individual.

The individual is also defined by the choice of expressions within the architecture. As we described earlier, on each cycle the set of emotions is examined and a single dominant expression is chosen that best exhibits the character's overall state of mind. The choice of set of expressions, and the rules for translating emotions into expressions, also determine the individual's personality. An individual who translates fear into an expression of anger is very different from one who translates nearly every emotion into expressions of fearfulness; likewise, these are different from a character whose major expressive themes are always upbeat and cheerful.

Further, an individual is defined by how the perceptions of the environment modify the internal state, and particularly how they create new emotion influences. Nervous characters, upon encountering what are typically fear-provoking situations (such as a rockslide blocking the only entrance from a cave) may all react similarly, so some of these rules could be ascribed to personality; but a character with a morbid fear of snakes may react with terror, while a highly nervous herpetologist may find nothing frightening in a snake he knows is not poisonous.

6.3.6 Believability

We have made much in this dissertation of the importance of "believability," the sense the character conveys that it is alive and real in a dramatic sense. Unfortunately, as we have also observed, this is a very subtle and highly subjective term. It depends

upon design choices both at the lowest levels and time spans (how do you describe how the character arches its eyebrow?) to the highest and longest (how did the character perform in helping you to solve the game?).

The architecture has no *direct* mechanism for producing believable characters. As we have described, it does have support for creating emotions, emotional influences, describing how those influences affect a particular character, controlling how a character reacts to other socially, and assisting the designer in focusing on a set of broad expressions and in smoothly transitioning among them. The sum of these, together with the details of the individual character's behaviors, may or may not produce a believable result. A poor designer might choose inappropriate expressions, define poor functions for translating influences into emotions, or simply author inconsistent and unappealing behaviors, and this would lead to an unbelievable character. Believability is the product of effective use of the architecture, rather than of any specific property of the architecture itself.

Of course, the architecture was designed with the creation of believable animate characters in mind, and we will show by example in subsequent chapters that it is possible to produce what we consider to be believable results. However, it is reasonable to imagine (though by no means certain) that by the addition of more capabilities — for example, learning, more sophisticated natural language processing and generation, models of the beliefs and knowledge of other agents, etc. — that it would either be easier to produce believable characters or that characters believable in a wider range of circumstances or time could be produced. We maintain, though, that we can and have achieved a sufficient degree of believability for our purposes using the architecture as it exists now.

6.4 Qualities of agency

6.4.1 Goal-directedness

This is arguably the quality for which the architecture has the weakest support, since it does not have an explicit notion of goals. However, it is important to observe

that goal-directedness in a character does not necessarily mean that the character *has* goals, in any real or formal sense, but only that it conveys to the observer the impression that it does and that it is choosing behaviors in accordance with those goals.

Goals can, however, be created and pursued implicitly. Broadly speaking, the goals of the agent are implicit in the organization and definition of *tasks*, which are collections of behaviors focused on a particular purpose. The goal *keep the player entertained* might contain a collection of behaviors for telling stories, expressing opinions, and various types of amusing behavior. Since the architecture permits the creation and destruction of tasks dynamically, it is also possible to create a new task whose behaviors and their constraints embody a more narrow goal that arises during the character's execution. In the adventure game environment, for example, if the player asks the character to go retrieve an item from a distant location, a new task is created with the necessary behaviors to plan to reach that location, get the item in question, return to the player, and hand it over. In this way it is possible to achieve a degree of goal-directedness in the character at the task level.

6.4.2 Reactivity

Reactivity means two things for a character: first, that it respond to what is happening in the environment, and secondly, that it does so in a timely manner. A character that responds but does so only after considerable delay is not likely to be believed.

In order for the character to respond, it must be able to perceive what is happening in the environment. The parsing subsystem allows an author to write handlers both for annotations the character will receive and for ordinary communications from the environment. Those handlers can manipulate the character's emotional state and the knowledge in the context manager. More importantly, a handler can be tied directly to a behavior so, for example, a behavior can parameterize itself and set its fitness according to a recent percept. In this way the behavior engine can offer behaviors that are tailored to particular percepts from the environment, which permits highly reactive actions.

Most of the architecture's time is spent in the selection of the next behavior. As a result, the behaviors need to be organized so that it is possible to avoid computing the fitness function of each behavior on each cycle. This is done first by arranging them into a hierarchy; at the highest level are *task groups*, which classify broad kinds of jobs the character has³, down to *tasks*, and finally to behaviors.

Individual behaviors have both preconditions, which test whether or not they are even appropriate to consider, and fitness functions, which return numerical ratings for their appropriateness on the current cycle. Preconditions are intended to be fast checks, while the fitness functions may take more work to calculate. Tasks and task groups may also have preconditions. For example, if the user with whom the character is exploring the adventure game is not in the same location as the character, the entire **Believability** task group is immediately eliminated from consideration (since there is no point in performing if there is no audience). This approach saves the character from computing fitness for all but the behaviors that are actually worth considering.

6.4.3 Situatedness

A situated character is one whose behaviors arise in response to its present perception of the environment. This is in contrast to the deliberative approach of creating an internal model of the world and then reflecting upon that internal model to decide upon a course of action. Traditional planning agents have tended to use this latter approach; they form a long-term goal and then examine the model to determine how to satisfy the goal. For example, such an agent may make a many-steep plan to pick up an empty soda can, to go down to the elevator, take the elevator down to the first floor, go from there to the kitchen and drop the can in the recycling bin. An execution monitoring system may cause it to replan when it encounters obstacles, as when the kitchen door is closed or the recycling bin is full. However, the general approach is one of working out a plan of action and then executing that plan. The virtue of a situated stance is that the agent is not constantly monitoring the world to

³By default the architecture has two task groups, one for **Believable** behaviors that are only meant to influence observers' impression of the character, and one for **Functional** behaviors that actually affect the state of the world around them.

guarantee that it meets its expectations, but is more opportunistically choosing its actions in light of the situation in which it finds itself.

As with other qualities, situatedness is more a result of the designer's choices than of the architecture. Situatedness in our architecture, it should be pointed out, is a property that individual tasks and behaviors may or may not possess, rather than necessarily of the system as a whole. It is certainly possible to build behaviors or entire tasks that do not make use of any environmental percepts in their preconditions or fitness functions, in which case you would produce a character that was essentially disconnected from what was happening around it. However, this would be a bizarre and rather unintuitive use of the architecture, since the structure emphasizes the concept that behaviors are appropriate in certain circumstances, and those circumstances should be described in terms of the state of the world as well as the state of the agent.

6.4.4 Broad capabilities

Finally, a character should be broadly capable. Since it is meant to interact with a human user, it must be able, or seem to be able, to emulate the major capabilities that we expect to see in living beings, qualities such as seeing, hearing, speaking, moving, thinking, feeling, etc.

The architecture permits these kinds of capabilities, and offers support for several. There are subsystems for parsing input from the environment, both annotations and ordinary communications. There is a memory where the character can record facts about the environment and its relationship to it. There is an emotional model that allows the character to represent feelings. All of these things can be used in reacting to the environment and in deciding how to behave, and the communication module permits arbitrary actions in the world, whose nature is up to the character's author. While more sophisticated abilities, such as learning, planning, abstract reasoning, etc., would be possible to implement within the architecture, there is no direct support for them.

Chapter 7

Demonstration 1: A Guide for the Web

The Wart did not know what Merlyn was talking about, but he liked him to talk. He did not like the grown-ups who talked down to him, but the ones who went on talking in their usual way, leaving him to leap along in their wake, jumping at meanings, guessing, clutching at known words, and chuckling at complicated jokes as they suddenly dawned. He had the glee of the porpoise then, pouring and leaping through strange seas.

“Shall we go out?” asked Merlyn. “I think it is about time we began lessons.”

— T. H. White, *The Once and Future King* [145]

In this chapter, we examine the first and simpler of the two environments we built to test the annotation concept. The interface is Web-based, and has two parts, a graphical control panel by which the user communicates to the character, and a graphical representation of the character, which communicates with the user via animations and synthesized speech. That interface is used to connect to a sample Web art gallery containing medieval and 19th century artworks. Our character, Merlyn, has the task of interacting with a single user exploring the art gallery by providing background and commentary on the artworks. The interface and the character are designed with child users in mind.

We begin with a discussion of research in this type of environment, which has some interesting properties not only for exploring annotations but for the use and development of animate characters in general. We will then turn to the technical details of the interface and of the sample environment.

7.1 Motivations

According to Schank and Abelson [123], the bulk of human knowledge and memory is communicated and encoded in story form. Although there has been a long-standing interest in how interface agents might bring social and narrative perspective to information resources [105], there is little explicit use of adaptive storytelling in the repertoire of those agents. Most of the interface agents in use today are only embodied alternatives to traditional menu- or prompt-driven mechanisms for performing simple tasks such as searching for files or providing context-sensitive help. Although they possess bodies and offer some social context, these utilitarian agents lack intelligent believability [10, 118]. There has been active research on pedagogical agents that do exhibit intelligent behavior, though they create narratives only in the sense that their pedagogical requirements impose a structured sequence of behaviors (e.g. [117, 132]). Finally, there has been work on the creation of stories through the interaction between the user and agents that are characters in the story [15, 59]. In these systems, however, story is the purpose of the interaction.

In contrast, our interest is in the explicit use of narrative and social context as ways to help users organize familiar ideas, learn new material, and more effectively engage with the content. In particular, we are interested in situated stories — stories that are tied closely to the environment — that can illustrate it and bring it to life for the visitor. There has recently been a growth of interest in this issue [2, 16, 63, 65, 109], and the term “narrative intelligence” has been introduced to describe systems that have these properties [93]. We approach this problem by creating an interface agent that can act in the human role of “guide” to a digital location.

7.1.1 (Not) communicating

One can think of communication as comprised of three components: context, structure, and content. Context provides a sensible interpretation of content; “Fire!” means vastly different things when shouted in a movie theater or on a battlefield. Structure provides a framework within which to organize content; the linear nature of a story, the hyperlinks on a Web site, or the relationship of notes in a musical score constitute structure. Content is the message, the information to be conveyed, but without structure or context, this information is not a communication.

The Web is a vast and growing body of content, but it has serious deficiencies in the other parts of communication. There is minimal structure; Web sites offer trails of hyperlinks among their documents and into other sites, but this structure is implicit in navigation and often not obvious to the user.

There is even less context on the Web than there is structure. It is possible to follow a hyperlink to a new site providing very different information from a different source, and to the non-technical user this change may be invisible. Without knowing sources, we do not have an intelligent way to evaluate the information we see, nor to categorize it with respect to our other knowledge. A Web page by the American Medical Association about the value of aspirin in treating heart attacks may be quite different from a page provided by a pharmaceutical company, a general practitioner, a suffering patient, or a faith healer. We rely on context for the meaning and quality of information, and although graphical banners and catchy URLs may help, the Web is fundamentally a decontextualized space.

One way to address these communication problems is by introducing an intelligent character as a mediator between the user and a Web page. Characters provide a social context and a narrative structure in which to place Web content.

7.1.2 How character interactions help

There are many possibilities for improving these communications deficiencies in the Web; why should we use characters rather than some other, more traditional interface enhancements?

There is considerable research suggesting that people unconsciously treat computers as though they were social entities [99, 100, 112]. This is a strong justification for social interfaces: if people naturally think of their interactions in social ways, rather than forcing them to adapt to interaction conventions convenient for the computer, we should have the computer adapt to our social conventions. In addition to feeling more natural, this allows us as designers to play to users' aptitudes. Social interaction is comfortable because it is very familiar and because it is highly error-tolerant; most software interfaces are uncomfortable because they are neither.

Social interactions also naturally provide a context for a communication — the social context. People do not just want information; they want to have a sense of how it is relevant to them. They need information that helps them make use of and digest what they are learning from a human, social point of view. Why does this matter and to whom? Why should I in particular care? How will it help me and why might I have a special connection to this information? Through tone of voice, inflection, facial expression, gesture, and physical appearance, we give each other valuable clues for evaluating the content of what we are saying. A tweed-coated professor calmly lecturing to students about the possibility of alien life will be received much differently than a wild-eyed fellow wearing a tinfoil hat and shouting on street corners. By presenting a character with a personality and a social role in relation to the information, we take advantage of peoples' social strategies for establishing context — their goals, how they align their interests in relation to those of the speaker, how they model how to respond, what's relevant to them and what's valuable — that is, their social filters.

By adding characters that are layered above the Web interface, we are also creating a persistent context. The character travels from page to page with the user, so even in the absence of context or through shifting contexts in the browser, the user has a continuous, evolving interaction with the character and, through the character, with the site content.

The other way in which characters address the communication weaknesses of the Web is by providing a narrative structure to the interaction. As we shall see, a character can be capable of representing a Web page's content through dialog and

story form. Schank and Abelson [123] argue that narrative is how most human knowledge is transferred.

Not only can a character reinterpret the contents of a single page in a narrative style, but the persistence of the character means that the whole course of a Web experience can be recalled and reasoned about in narrative terms. Dialog with the character is a way to maintain persistence of memory, and to draw relationships between what was encountered, what is being examined, and what future options might be appropriate.

By supplanting the existing interface, which is convenient for computers but less so for humans, with a mechanism that humans find comfortable, familiar, and useful, we believe we can improve the efficacy of communication on the Web, and thereby the quality of the user's experience on the Web.

In addition to providing a more familiar interface, another advantage of providing a character as a guide to the Web is to raise the level of interactivity. Rather than expecting the user to sit passively and read the text, we can provide a lifelike agent that will speak, gesture, move about, ask questions, and generally raise the level of interactivity of the site. This hopefully increases the user's focus on the material, and therefore both the level of enjoyment and the degree of retention. Lester et al. [84] have demonstrated that the mere presence of an animated character, whether it provides domain details or not, has a salutary effect on children's experiences in educational computer programs.

7.2 The interface

The interaction between the user, the character, and the environment, takes place through an interface with two components. The first component is a standard Web browser. The browser window displays the content of any art gallery that the user is visiting. It also displays, in a persistent frame at the bottom of the window, a graphical control panel which the user employs to issue commands to the character.

The second component is the animated body of the character. Written as a

Java applet using Microsoft Agent¹, the character operates independently of any site the user is visiting. The character body may move around the screen, play various animations (e.g., reading a book, scratching his chin, clapping his hands, pointing and waving at things, etc.²), and talk to the user in a synthesized voice. The character operates in parallel with the browser interface, and it exists throughout the interaction, independent of which or how many gallery sites the user visits. Figure 7.1 shows the organization of the interface.

7.2.1 Communications between the character and the environment

When the session begins, the user loads the character by going to Merlyn's home page. This page is split into two frames: the gallery viewing frame, initially empty, and the control panel frame. The control panel frame, which persists through the session, loads the character applet. In addition to displaying icons for communicating with the character, the control frame contains JavaScript code that mediates between Merlyn and the viewing frame.

The Web pages of the museum, which appear in the viewing frame, have two kinds of control code embedded in them. First, when a new museum page is loaded, it makes a JavaScript call to the control frame, telling it there is a new page. The control frame then calls the Java applet with the name of the new page. The applet performs some bookkeeping, then calls the Web server directly with a request for the annotations corresponding to the new page. It is assumed that each for each page of the gallery Web site, the character can ask for and receive an annotation document that explains the content of that page.

The other control information in the Web pages is hidden in the artworks. For

¹The architecture of Chapter 5 is only slightly modified to run as an applet rather than as an application; it is in all significant respects the same system. The Microsoft Agent package is only an animation front-end; it has nothing to do with the architecture.

²There is a clear distinction between the body of the character and the mind of the character. The Microsoft Agent software makes it relatively straightforward to replace one body with another, since a body is nothing more than a collection of animations stored in a particular file. We simply chose a Microsoft Agent body that was compatible with the personality of the character we wanted to build, but most of its behaviors would function perfectly well if replaced with another body.

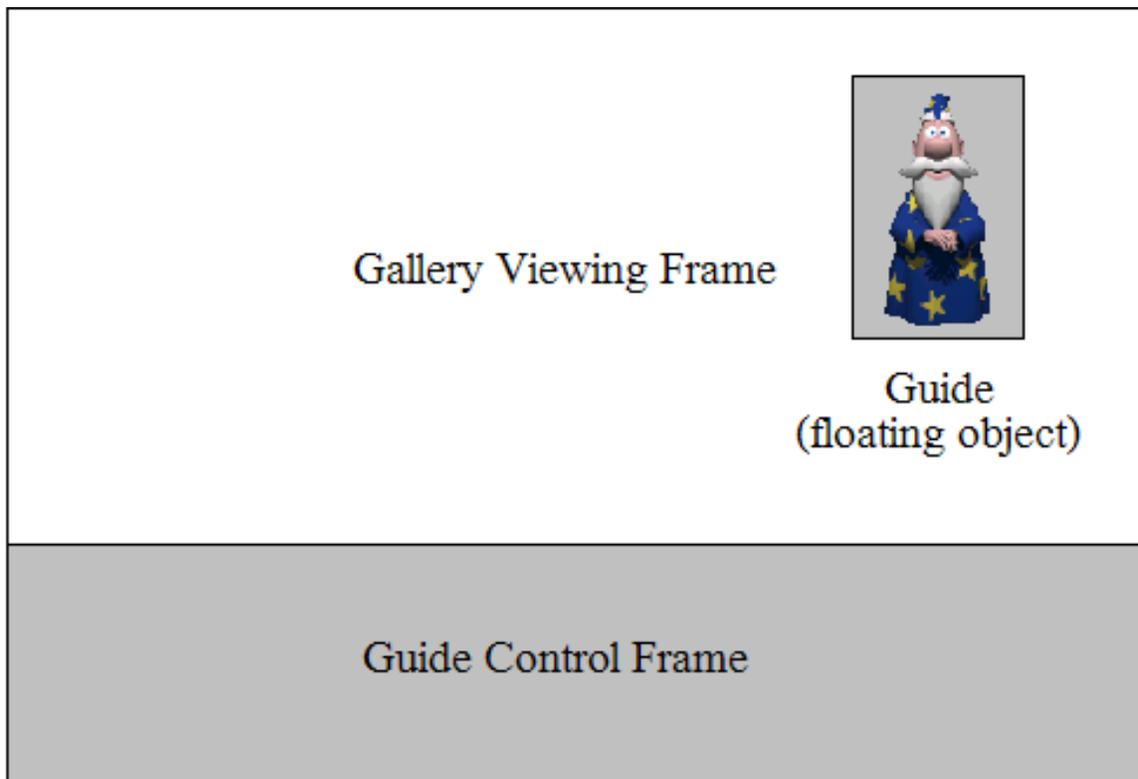


Figure 7.1: Components of the interface

the agent to use the entity annotations, which provide details about specific parts of an artwork, it must know where those parts are and when the user is selecting them. We handle this by making the artworks HTML imagemaps, where the hotspots of the imagemap correspond to the entities defined in the annotations. When the user clicks on a hotspot, the Web page makes a JavaScript call to the control frame, telling it the name of the artwork and the name of the entity the user selected. The control frame passes this information to the agent.

Ideally, there is a complete division between the HTML, which contains the surface representation of a site, and the annotation markup, which contains its deep meaning. One of the goals of this approach, after all, is to allow site builders to construct annotations independently of the page layouts. In practice, however, we must embed a small amount of control information in the Web page. This is because, due to the

fundamental limitations imposed by the browser and of (D)HTML and JavaScript, the character is unable to sense what the user is doing on the page without being told.

The museum Web pages can determine whether the control frame is available, so that if a user visits the museum without an agent, no function calls are made. This is in keeping with the design goal of making the site usable even when there are no agents present. The overhead for this control code is 1-2 lines of code on each page for loading, and an additional line for each entity in the imagemap. Thus while it is philosophically unsatisfying that it is necessary to have code in each gallery page for priming the character, the actual amount is quite small.

7.2.2 Communications between the user and the character

The character converses with the child user through motion (such as pointing at a piece of art on the Web page) and through synthesized speech that also appears in bubbles above its head. However, in keeping with the point-and-click nature of the Web, the user communicates with the character through a control panel of icons at the bottom of the browser window. The user can issue these instructions:

1. **Tell me about the painting.** If there is information available, Merlyn will talk about the title of the painting, the artist, its era, and provide any other details he finds in the environment.
2. **Open my scrapbook.** The scrapbook is a pop-up windows that contains snapshots of each of the paintings that the user has seen, and provides a place for the user to make notes about those paintings.
3. **Let's play a game.** Merlyn will play a game with the child, asking him or her to find certain things depicted in the current artwork or how it or the entities within it relate to other art they've seen in the museum.
4. **Let's make a story.** This is a stub for a tool that allows the child to write stories about the paintings he or she has seen, aided by suggestions from the



Figure 7.2: The guide control panel

character.³

5. **Tell me a story.** Merlyn will take the child on a tour of one or more paintings in the gallery that are connected by some theme, telling part of a story at each painting.
6. **Last part.** If the character is telling a story, this indicates the child wants to go back and hear the last part of the story over again.
7. **Stop the story.** Commands the character to stop narrating whatever story it is telling.
8. **Next part.** If the character is telling a story, move on to the next part.

Figure 7.2 shows the control panel. Items on the control panel are lit up or dimmed depending on whether they are currently available. We describe how these functions work within the character in 7.5.

7.3 The environment: an art gallery

Having created a character and an interface for visiting art galleries on the Web, we next needed to build galleries to visit. Our example art museum Web site contains two galleries: the *Nativity Exhibit* houses medieval and Renaissance religious art revolving around the birth of Christ, while the *Pre-Raphaelite Exhibit* features art painted in 19th century England, both by and in response to the artistic style of the

³This tool was not fully implemented.

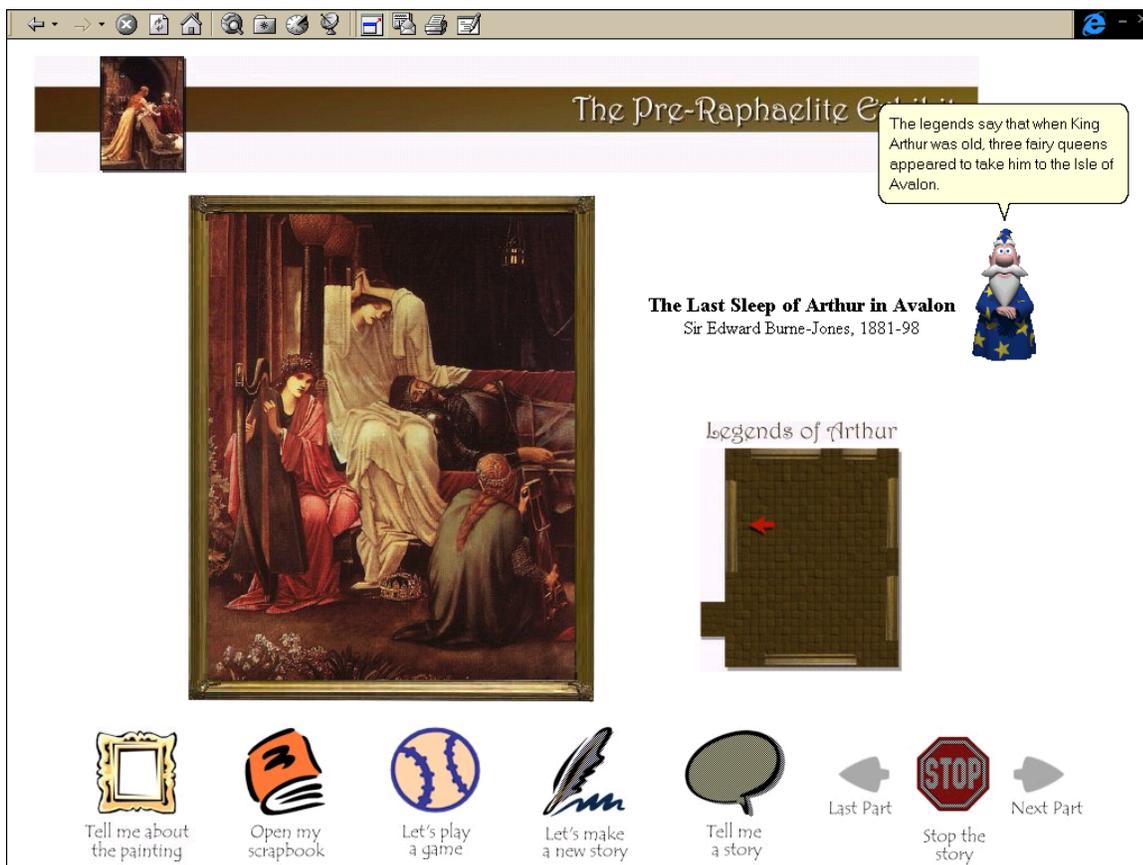


Figure 7.3: Merlyn in the museum

pre-Raphaelite movement. (These topics were chosen because of our familiarity with and interest in them, rather than for any particular pedagogical purpose.)

The museum is organized by analogy to a physical environment. It is broken up into the two galleries, and each gallery is composed of a set of rooms. Rooms in turn are built up of walls. Each Web page represents either an overhead view of an entire exhibit gallery or of a portion of a wall in one of its rooms. We are representing one piece of art on each wall, and we are using only paintings (as opposed to sculpture or other artifacts). When in a room, a small map in the lower corner of the page indicates the shape of the room, where the paintings are located, and which painting the user is viewing. The map is used to navigate from painting to painting and from

room to room within the gallery. Figure 7.3 shows Merlyn describing a painting in the Legends of Arthur room within the Pre-Raphaelite gallery.

The painting takes up most of the space on the page. Paintings are labeled with the title and artist's name. The paintings are also *imagemaps* (see the technical discussion above), so the user can click on significant aspects of an image and have the character react to those selections. Below we discuss some of the ways in which this capability is used.

Each Web page has an accompanying document on the server that contains the annotations for that page. The character visiting the site has access to both the Web page and the annotations.

7.4 The annotations

Documents on the World Wide Web are written in HTML, the Hypertext Markup Language. This language tells browsers how to display the documents. It provides a description of the form, as opposed to the function, of the data in the Web page. HTML is the mechanism by which a user sees the content.

The annotations read by the character are written in XML, the Extensible Markup Language. XML represents the content of the page semantically; it indicates the function of the data. For each HTML page in the virtual art gallery, there is a corresponding XML file that describes it. XML is the language through which the agent sees the content.

Syntactically, XML is very similar to HTML. An XML document also consists of a sequence of elements. Each element is delineated by bracketed start and end tags (e.g., `<TAG></TAG>`). An element may contain plain text and/or additional elements. A start tag may optionally possess attributes, which are name-value pairs listed within the brackets. Figures 5.2 and 5.3 show fragments of XML documents.

The current version of the museum emphasizes basic facts in a simple vocabulary. Unlike the annotations in our MUD-based world (described in Chapter 8), most of these annotations are static and strictly declarative, giving facts without suggesting any structure for presenting them. In this section we discuss the more important

elements.

The top-level element is the area. An area is defined as some portion of the site — the site itself, one of the exhibit galleries, a room within a gallery, or a wall within the room. Areas, like most elements of the vocabulary, are uniquely defined by their name attributes. Areas may contain lists of artists whose work is available there, a list of the artworks displayed, a list of the topics covered by the area, and possibly definitions of the areas within them, as sites and galleries do. They may optionally have some text description; any element in the vocabulary may have some text associated with it. A block of text has one or more audience attributes indicating the group at which it was targeted, such as small children or art critics.

Artists are defined by their names, dates of birth and death, and nationality. Artworks are defined by title and artist name, and may also contain information about the dates on which they were painted, the medium used, the category of artwork (sculpture, painting, etc.), the topics they relate to, and topics related to those topics. An artwork may also possess a list of the visible entities it depicts, where each entity corresponds to some meaningful portion of the piece. Individual artworks that form part of a thematic group (such as the Briar Rose series of Edward Burne-Jones) indicate their organization with a part-of element naming the group.

Artworks can also be grouped via tour elements. In contrast to the other declarative annotations, the tour elements are procedural. Conceptually, a tour in the art gallery consists of a sequence of stops at pieces of art, with some text describing the artwork's significance in the tour. An artwork may be associated with any number of tours and provide different details for each one. The tour annotations are analogous to the idea of a physical tour through a museum, as frequently occurs today with either a guidebook or, increasingly, with headphones that provide audio narration about the works of art the visitors are seeing.

Each piece of art contains a list of all the tours with which it is associated. For each tour, the artwork element contains the name of the tour, the name of the current stop, and some descriptive text (and possibly several variations for different audiences). It also lists the name and URL of the next stop on the tour, as well as the name and URL of the first stop of the tour.

Here we show an abbreviated example (some of the less interesting elements are omitted) of the annotations associated with one page in the art gallery. The annotations refer to the painting in Figure 7.3.

```
<area type="wall" name="Legends of Arthur West Wall" desc="the west wall"
  url="arthur/arthur1.html">

  <description>the Legends of Arthur</description>
  ...
  <introduction audience="child">
  This part of the gallery shows Pre-Raphaelite paintings about
  King Arthur, the Knights of the Round Table, and other people from
  the court of Camelot.
  </introduction>

  <topics>
    <topic>King Arthur</topic>
    <topic>Knights of the Round Table</topic>
  </topics>

  <artists>
    <artist name="John William Waterhouse">
      <first>John</first>
      <last>Waterhouse</last>
      <nationality>English</nationality>
      <year-born>1849</year-born>
      <year-died>1917</year-died>
    </artist>
  </artists>

  <artworks>
    <artwork url="gallery/arthur/pic/jones-lastsleep.jpg">
      <title>The Last Sleep of Arthur in Avalon</title>
```

```
<artist>Sir Edward Burne-Jones</artist>

<date>
  <begun>1881</begun>
  <finished>1898</finished>
</date>
...
<category>painting</category>
<material>oil on canvas</material>
...
<text audience="child">
  The legends say that when King Arthur
  was old, three fairy queens appeared to take him to the Isle of
  Avalon. There he would rest until the world needed him again.
  In this picture, the fairy queens are playing music while he
  sleeps.
</text>

<entities>
  <entity name="arthur">
    <text audience="child">
      King Arthur looks tired, doesn't he.
    </text>
  </entity>
  ...
</entities>
</artwork>
</artworks>
</area>
```

Listing 7.1: Extract from art gallery annotation

For a complete gallery page annotation, see appendix Section B.2. Tables summarizing all of the elements of the art vocabulary are in Section B.1.

7.5 The character: Merlyn

Our character is based upon Merlin, the wizard who was teacher and advisor to the mythological British figure King Arthur. In his retelling of the Arthurian legends for children, *The Sword In The Stone* [145], the novelist T. H. White used the name “Merlyn” for the wizard, and portrayed him as a querulous, befuddled old man who nonetheless had a vast store of wisdom and the magical abilities to prepare young Arthur for the day when he would become king. That book, and the Disney film based on it [142], are the sources upon which we’ve drawn to create our own Merlyn.

Merlyn is an expert on guiding children through art museums. His tone is friendly, though often a bit confused, but his explanations about the art he sees are simple and direct. As a result, he does not use all the information embedded in the museum, since some of it is too technical or complex for a small child to grasp. (Most children, for example, neither know nor care about the difference between oils and watercolors, and are far more likely to be interested in what the princess is doing than in the brush style used to paint her.) Instead, Merlyn focuses on basic ideas about art and tells stories about the paintings he visits with his young ward. Figure 7.4 shows Merlyn in the Briar Rose room of the Pre-Raphaelite Exhibit, reading the legend of Sleeping Beauty to a child visitor.

Merlyn is implemented in our architecture. The structure of his memories, the ways in which his emotions are created and change, and the manner in which behaviors are chosen all come from the architecture. The particular emotions that Merlyn has, the parameters of the functions that translate emotional influences into emotions, and the specific tasks and behaviors that he possesses were all implemented in the architecture. We describe those specifics below.

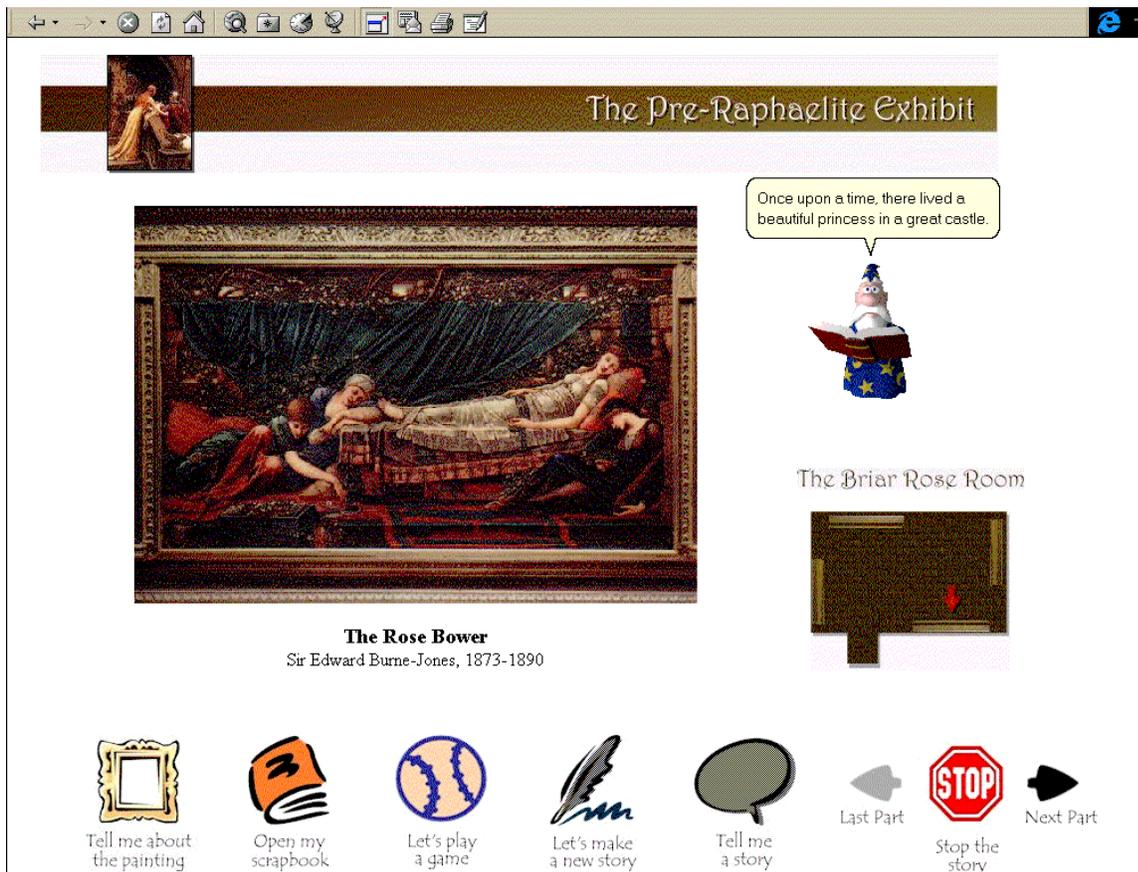


Figure 7.4: Merlyn telling a story

7.5.1 Personality and emotions

Merlyn is meant to be a helpful and generally friendly character, but one with a short temper; however, his bad moods do not last long.

We model Merlyn's temper in our specification of the translation functions between negative emotion influences and the values that they produce (see Section 5.5 for an explanation). Briefly put, it takes very weak negative emotion sources (a source may be the user, some element of the environment, or be generated within Merlyn himself) to generate an emotional response, and the response is scaled to be much stronger than the original source. However, the speed with which the negative emotion influences decay is very high, so they quickly weaken and their influence of

Merlyn's behavior fades.

Merlyn's negative emotions arise as a result of the user issuing contradictory commands or commands he does not wish to receive (such as being told to stop a story). The relevant negative emotions that Merlyn uses are **REPROACH**, **ANGER**, **DISLIKING**, and **BOREDOM**.

Merlyn's positive emotions arise from the same sources, but in response to other stimuli. His **PRIDE** in himself and his **SATISFACTION** increase when he successfully completes a task, such as describing a piece of art or telling a story. **JOY** and **APPRECIATION** go up when the user does well playing a game with Merlyn.

Note that the artworks in the environment are not directly annotated with affective suggestions for the viewing characters. This was a deliberate choice. There would be no technical difficulty in doing so; for example, an exhibit of Hans Holbein's *Dance of Death* woodcuts could produce **DISLIKING** of the art style or **FEAR** because each woodcut is a *memento mori*. However, responses to artworks are generally intended to be more complex than this, and even contradictory. One character might like the style of a particular painting or its medium while personally disliking the contents, or even like and dislike different elements for different reasons. Also, there is a difference between a response to the painting and a response to the content of the painting, and it would not make much sense for a character to be afraid of the painting itself. A more elaborate annotation vocabulary might begin to capture these distinctions, but we do not attempt them here, because in any case, unlike a dramatic environment in which the character may be used to heighten a player's perception of certain emotions in much the same way that a soundtrack does in a film, it did not make sense to us for a character to attempt to emphasize *emotional* reactions to the artworks themselves.

The Merlyn character's use of the emotion system is not as sophisticated as that of Harker (described in the following chapter), primarily because the nature of the interaction does not lend itself to complex emotions. The user's interface does not readily permit actions that the character would find blameworthy or irritating. The environment is static (providing little in the way of surprise), and moreover it is not liable to produce the kinds of events or objects that an agent would find threatening or truly undesirable.

7.5.2 Believability-enhancing behaviors

Merlyn's behaviors fall into two categories. First are those behaviors that he exhibits solely to appear more lifelike and to be more believable to the user. These behaviors are not dependent on the environment and do not affect it.

The most basic of these behaviors are those that exhibit boredom. The Microsoft Agent body provides a range of animations suitable for representing this (e.g., `Blink`, `Think`, `Sleep`, `GlanceLeft`, `DoMagic`, and so forth). The `BoredomBehavior` in the character architecture is weighted according to the `Boredom` emotion, which grows as time passes with no inputs from the environment and the user, and when it is selected, it chooses among a list of these sorts of animations to exhibit. Once he becomes sufficiently bored, he will manifest it with the `Sleep` animation, and remain that way until something happens.

Merlyn also has a `PersonalStorytellingBehavior`, which he uses to tell stories about himself and his background. This behavior can be invoked as a result of boredom or of good humor, and is a non-atomic behavior that will continue across (possibly) several utterances (depending on which story is selected by the behavior).

The character does not have many purely believable behaviors. The assumption is that the child is primarily interested in the content of the site, and that the character should focus its invasive actions — since any behaviors it exhibits will distract the child from exploration of the content — to describing that content or interacting with it. In this setting, the character is an adjunct to the content. As a result, we deliberately avoided giving Merlyn many such behaviors.

7.5.3 Functional behaviors

Merlyn's other category of behaviors are the functional behaviors. These are appropriate when Merlyn has some knowledge of the content of the Web page and is able to act on that knowledge. Such behaviors include playing games, using the artworks as props, telling stories about the art, talking about the artists or art history, or guiding the child on tours through the site. The goal is to integrate these behaviors with his believable behaviors seamlessly, so that it isn't apparent where Merlyn ends and the

site annotations begin.

In Merlyn's case, his behaviors were designed with respect to three general principles we felt that the interaction between a guide character and a user should follow. Here we provide the principles and describe the mechanisms Merlyn uses to follow them.

Principle: Make passive information active.

The major argument for adding a character to a Web page, or to any environment, is to make it livelier. Rather than requiring the user to sit passively and study the display, we can provide an animate character that will speak, gesture, move about, ask questions, and actively present the content of the site. The goal is to increase the user's enjoyment of the experience, focus on the material, and hopefully their retention as well. Lester et al. [84] have demonstrated that the mere presence of an animated character, whether or not it actively presents domain-specific information, makes children more attentive to the content.

Merlyn applies this principle in three ways. First, whenever the user enters a new area of the site, Merlyn will speak any introductory text provided for the area. As mentioned earlier, the text can be partitioned according to the audience, so Merlyn might talk about the historical significance of the art, the technical details of its composition, or anything else the designer chooses, depending upon the user's interests.

Second, Merlyn can talk in detail about individual artworks, either autonomously or when prompted by the user via the control menu. If appropriate annotations for the artwork and its artist are available, Merlyn will generate a brief dialog and then talk about the art — who painted it, and when, its historical context, similar pictures on the site, and so on. In contrast to the introductory text, which is generally provided, the surface structure of these dialogs is generated by the agent using the annotation data.

Third, Merlyn will provide details about parts of the artworks. He uses the entity data associated with an individual artwork to point out and describe them, autonomously or in response to the user clicking on hotspots in the image, if they

exist. The agent can track what he has already talked about, to prevent himself from being repetitive. As his knowledge of other art on the site grows, he can relate what he is describing to things the user has already seen. For example, if there are multiple artworks that are **PART-OF** the same group, he can reference others that have been seen or observe that there are more yet to come, or he can mention that the artist who painted the piece they are examining also painted some others seen earlier, etc.

Principle: Build narratives with the available information.

The structure of the Web makes it easy to view each page as disconnected from the others. One of the goals of the annotation approach is to help weave the pages of the site into a coherent whole, or rather, into many different coherent versions, depending on the interests of the user.

The first way that Merlyn does this is through storytelling. The concept of the tour, introduced above, is easily reinterpreted for children as telling stories about the art they are seeing. One of the options on the control is to ask Merlyn to tell a story. This will invoke a **StorytellingTask** with a constellation of behaviors for creating a tour, leading the child to the stops on the tour, discussing the artworks, etc. That is, if asked, he will scan the art on the current page to see whether any tours are available. If there are, he will choose one, and then lead the child back to the beginning of the tour. He will then recite the tour text for that artwork, performing relevant animations (he opens and scans a large book as he speaks, for example). He then tells the user where the next part of the story can be heard, tucks the book away, and waits. This gives the child the opportunity to explore the images and ask for more details. Eventually, the child can either go to the new location, at which point Merlyn will detect the motion and continue the story, or the child can ask Merlyn to go on, in which case Merlyn will load the next page and continue. If the child idles for too long, an encouragement behavior within the **StorytellingTask** will suggest that it is probably time to move on and see what happens next.

Merlyn can also use the topic annotations contained within the areas and the individual artworks to create more general narratives. When the child asks about a picture, Merlyn describes it, and can then suggest they look at other pictures that

deal with similar topics, emphasizing those the child finds interesting. He determines interest by tracking the areas the child visits and the entities within the artworks that the child selects.

Principle: Engage the user with the content.

Beyond making the content of the page active, our third design principle is to make it interactive. The child is more likely to analyze and to recall what the pages describe if Merlyn asks for input that requires he or she evaluate what is being displayed.

Merlyn does this with a `GamePlayingTask`. The simple example currently available on our Web site is a game of “I Spy.” This is a traditional game for young children, in which the adult says, “I spy, with my little eye, something beginning with (some letter).” The child then tries to find some thing or person present that has a name beginning with that letter.

When Merlyn starts to play a game, he makes a list of all the entities in the artworks on the current page. He chooses some entity at random, recites the verse, and waits. The child can make guesses by clicking on hotspots in the artworks, and Merlyn will give congratulations or corrections depending on the guess. He will occasionally remind the child what he’s looking for and can give clues to help solve the riddle.

While “I Spy” is a trivial use of the annotations, it demonstrates how Merlyn can engage the child with the content. The limited depth of these interactions is due not to the annotated environment approach (we have already done more elaborate examples in our MUD environments), but the narrow channels of communication available to a small child on the Web.

7.6 Lessons Learned

The Web art gallery was our initial test of the annotation idea, and although it is a fairly simple environment that does not provide much opportunity for interaction by the character, it still gives us some insight into how annotations can be useful to characters. We conclude the chapter by discussing those insights in the context of

the claims we made about annotations in Chapter 4. We will have still more to say in our discussion of the adventure game environment at the end of Chapter 8.

Claim: Annotations enable characters to behave believably in unfamiliar environments.

The Web gallery example does not provide much evidence for this claim, mainly because the Web does not provide much opportunity for interaction. The character and the human viewer are interacting with one another, and the Web page is more a static backdrop for that interaction than it is a mediator. Merlyn's believable behaviors are not constrained by the contents of the page, because the embodied character does not operate within the page; the animated character can fly around the desktop screen, wave his wand, tell stories, etc., all independent of Web content itself.

We performed informal user testing on a preliminary version of this character with the 23 students of a class studying intelligent characters. The gallery the students visited contained three "rooms" containing a total of eleven artworks, each of which was annotated with details that Merlyn could use for descriptions. The artworks in the Briar Rose room were connected with a sequence of TOUR annotations as well.

Each student interacted with the Merlyn character in the gallery and subsequently posted a summary of those experiences on the course bulletin board. Of course, the character's behaviors being semi-random, and the students being under no constraints to follow a particular sequence in the interaction, every student had a somewhat different experience. Nevertheless, from those summaries, and from our own numerous interactions with the character, and beyond the generally positive impression Merlyn produced, we make several observations.

First is that there was some frustration with the choice of domain. The Merlyn character introduces artworks, and can provide factual information about them, but does not hold an opinion of his own. Art is interpreted, and each viewer of the pages has a different opinion, but Merlyn does not possess one and cannot formulate one. The annotations are not much help in this case; they provide details about the elements in the artworks, and could provide further information about the period,

brushwork, etc., but the salient qualities of an individual artwork are generally too subtle for this kind of crude breakdown to be sufficient for a character to decide upon a reaction. Merlyn also didn't do anything to relate art that had been seen in the past to current artworks; that is, he did no user modeling, which while not a serious problem, still seemed disappointing given his tutorial role.

Second, the Web works well as an environment for characters, simply because users are used to it being largely non-interactive, and their expectations about what Merlyn will do are limited. They do not anticipate an extended dialog or a capacity to express complex ideas; the button menu for communicating with Merlyn didn't seem unduly restrictive. Possibly experience with the Microsoft Office assistant reinforced these expectations.

This also seems to have raised people's tolerances for repetitive behaviors. The animated body used by Merlyn only has about a handful of idling behaviors (stare at my wand, go to sleep, look around the screen, etc.), and despite the fact that Merlyn reuses them frequently while the user is reading the Web pages and examining the art, they were not distractingly unnatural.

Our general observations about the largest challenges for such interface characters are summarized in the following section, Section 7.6.1.

Claim: Annotations enable characters to be functionally capable in unfamiliar environments.

The Web example focuses on this aspect of annotations. The page annotations describe the contents of the artworks displayed on each page, information about the painters, and comments on elements of the paintings. Merlyn is able to use this information to describe the art to the child viewer, to ask questions, to relate it to other artworks that have already been seen, and so on. All of these behaviors are dependent upon knowledge about the artworks that would not be available unless it were provided to the character. It could, of course, have been programmed into the character in advance, but if we assume as we do that the goal is to allow the child and the character to visit many art galleries on the Web, galleries under the control of different authors who share only the annotation language, then this is an impractical

approach.

It would be possible for an intelligent character to mine some information (such as the name of the artist of a painting) from the Web page contents by analyzing the text, but there is no guarantee that this analysis would be correct. Especially for a character in the role of a guide or tutor, such an approach would be uncomfortably unreliable.

Claim: Annotations are a practical mechanism for achieving these ends.

In order for a character to understand the contents of a Web page, it either analyze the material presented for human consumption or it must be provided with a version of that material which is intended for its understanding. Particularly in the case of the Web, this is not a difficult end to accomplish. The lack of dynamism in the average Web page means that annotations can be incorporated into an individual page, stored in a separate file, or contained in a database associated with the site and which a character might query at need. There is little additional overhead imposed by the presence of the annotations, either in terms of space or load on the Web server; the only effort is in authoring the annotations themselves.

7.6.1 Challenges

There are some general issues that arise in trying to make use of an intelligent character to create a narrative interface. Based upon our experiments, we close with some observations on what we see as the most pressing problems.

1. **Intelligent reincorporation.** Reincorporation in this context means the reintroduction of ideas or entities that have been seen earlier in the tour, with some reference to how they relate to the topic under discussion. In improvisational theater it is well understood that reincorporation is a key to building a story satisfying to the audience [70]; a sequence of unrelated events does not make a story. Similarly, we suspect that a key to creating an effective tour narrative is reincorporation of earlier material. This requires that the agent not only track

what material has been seen but also when connections are either pedagogically or dramatically effective.

2. **Empathy with the content.** Elliott [39, 38] has argued that understanding of and reasoning about narrative is strongly associated with reasoning about emotion. If we think of stories as descriptions of sequences of events that have emotional associations for the listener, then we can enhance the significance of our tour content to the user by infusing emotion into the presentation. Beyond incorporating emotions in the presented material, we can build our agents to react emotionally to what they present. This not only heightens the significance of these narratives, it also enhances the believability of the tour guide.
3. **Presentation through personality.** The most obvious flaw of an intelligent character that relies upon pre-written text is that the text must either be designed for that particular personality, rendering it dangerous for other personalities to use, or so devoid of character that the recitation seems stilted and unbelievable. Ultimately, one would desire an annotation language and an agent powerful enough that the concepts could be explained, reinterpreted, and formed into dialog by the agent. Unfortunately, this is still a hard unsolved problem. Syntactic sugar is a simple approximation that frequently provides good results, as users come to associate idiosyncratic verbal behaviors with the agent's personality. Nonetheless, a more intelligent mechanism for integrating content and presentation will ultimately be required for widespread use.
4. **Artful timing/delivery.** As noted in the section on human tour guides, successful storytelling involves knowing when to begin a story, how long it should last, and whether one should elaborate with related stories. Making good decisions about timing and delivery requires the ability to detect user interest, and react appropriately to it. Human beings use many subtle cues to indicate low or high engagement, and detection of user interest is currently very primitive. This will continue to be an important area of research, both for development of characters and for development of satisfactory and subtle interactions with interfaces, in general.

Chapter 8

Demonstration 2: A Sidekick for an Adventure Game

In the previous chapter, we saw a simple and limited application of the annotation concept in a Web-based art gallery. We now turn to a more complex and flexible example. The environment is a text-based virtual world in which both the character and a single human player are embodied and can perform complex actions through a command-line interface. The content is a traditional, puzzle-driven adventure game. The character provides hints and active assistance to the player in solving these puzzles; it also interacts with the player in other ways, such as playing games within the game; and it expresses its own personality through its behaviors, including its emotional reactions to the environment around it. The character's abilities are the product of its understanding of the general properties of adventure game environments combined with the knowledge of specific details of this game as embedded in the annotations.

8.1 Motivations

The environment and examples this chapter will consider are very similar to the current crop of what are called massively multi-player online role-playing games

(MMORPGs) (e.g., [129, 43, 134]). These are vast online worlds capable of supporting many thousands of simultaneous players as they interact with one another and with the environment, generally for the purpose of taking on some fantasy role in a game, although a few of these worlds exist purely for socializing with other people [62].

One of the principal problems with these worlds even today — and they have existed in one form or another for decades — is the incredibly impoverished interactions with the non-player characters. Characters generally come in two types: those that are opponents, whose purpose is to wander around the environment until they are defeated by the players (often in a bloody battle), and those that are assistants or shopkeepers, who will provide goods, services, or information to the players who are visiting them while resting between exploration of the world, setting forth on quests, and intermittent bouts of slaughter. The former are generally called “monsters” while the latter are referred to as “non-player characters” or NPCs, because they superficially look and behave like the player-controlled characters. In neither case do these characters exhibit many of the traits of lifelike behavior, and this is especially noticeable in the NPCs. An innkeeper who has no knowledge of current events and can only talk about what alcohol he has for sale, a shopkeeper who doesn’t remember his best customers, an ordinary citizen who is oblivious to the fact that a dragon is attacking the town: these are all examples of the kinds of characters that exist today. They are generally not more lifelike or cognizant of their environments than furniture with tape recorders attached, repeating the same few utterances by rote, irrespective of the circumstances¹

The irony is that these NPCs exist primarily in order to provide some color in these worlds. Players can’t be expected to fill these supporting roles; they want to be the heroes. Players also do a poor job of behaving in character themselves. Generally, in single-player game experiences, the entire environment and everything that takes place in it is tailored to giving that one player an immersive experience. It is not up to the player to foster that experience. However the player behaves, the world of

¹The single notable exception to this state of affairs is the popular simulation *The Sims*, about which we will have much more to say in Chapter 9. It is not actually an MMORPG, however.

NPCs will react in character (even if uninterestingly). However, in the MMORPGs, most of the active characters are being run by game players who are concentrating on their own goals and their own personal enjoyment.

The disconcerting result is to have one class of characters standing around and being generally unaware of what is happening in the world, much less doing anything compelling themselves, while the other group is rushing back and forth from the store to the dungeon or engaged in heated debates about the quality of recent films, the hardware they are using to run their game software, politics — very often anything and everything except for the game world itself. Despite being called “role-playing” games, as a general rule, while a player controls a character that has certain abilities determined by the game, players rarely make an effort to have their characters act in ways that are consistent with the idea that they are part of a living world² Since it is not reasonable to expect the players to behave more in character, it is necessary for the NPCs to do so. They must provide not just a mechanical framework for performing certain ostensibly social tasks (such as trading goods or crying out the local news), but also a social framework that demonstrates how to behave appropriately in the world and how to react in response to what is happening in the world.

NPCs should also be capable of more than just being intelligent shopkeepers. They should be able to interact with players in more interesting ways. NPCs should be able to take on the roles of knowledgeable guides, hapless assistants, devoted sidekicks, or scheming enemies. A guide character should be able to tell a player where there are ruins to explore, what the dangerous paths are, who has come before and what successes they have had, what to watch for, how to prepare for an expedition, and so on. Enemies should know the terrain; an evil wizard who lives in a forgotten castle should know the castle better than the player who comes calling. A sidekick should be able to run for help if something goes wrong, should be amazed or terrified by what it encounters, should be able to encourage or comfort the player as well as be

²To be sure, there are players and game worlds that emphasize role-playing, such as the MMORPG *A Tale In The Desert*, which focuses on collaborative in-character effort, or many of the player-run servers using the *Neverwinter Nights* system. However, it is a strain for most players to try to project themselves through the personality and goals of an independent character and to speak in a voice that is not their own about topics that are pertinent to the in-game world and not the real world, and the groups that do so are in the clear minority.

useful for fetching, carrying, etc. Townsfolk should have daily lives, reactions to the changes in their world, some ability to act in accordance with their supposed needs. There are many ways in which characters could significantly improve both the social qualities and the engaging content of what is, after all, supposed to be a role-playing game.

However, there are significant problems with making NPCs more able to function in the environment. One is that computational resources are limited. A world with thousands of players and perhaps an order of magnitude more of NPCs cannot afford to devote much computing power to any one character. Even with much greater resources, the amount of knowledge and the extent of the reasoning abilities that a character would have to have to be generally capable in one of these MMORPGs is simply beyond our current abilities. It is practically an AI-complete problem; if we could build generally believable characters we could likely build generally intelligent characters as well. And most important of all, pure reasoning is not going to be sufficient to achieve all the effects we would like to achieve. Some aspects of the world cannot be understood by reasoning from existing knowledge; other knowledge *must* be provided.

The motivation for this research in the context of MMORPGs has been to find a way of approximating solutions to these problems. How can we, without requiring significant computational resources, create a character that has some idea of what is going on in the environment, how to react to it, and what behaviors in general to exhibit? The annotation approach is our answer to that question.

8.2 The environment: a text-based adventure game

One important early type of text-based virtual world is the MUD, or Multi-User Dungeon (or Dimension) [11]. MUDs first appeared in the late 1970's as recreational fantasy environments similar to the adventure games popular at the time, such as

```
> look
On the Stage
This is the large wooden stage of a small theater used for improvisational experi-
ments. The stage is bare but well-lit by spotlights high above. The plush theater
seats rise up all around the stage into the darkness.
– There is one obvious exit: offstage.
William Shakespeare, the Bard of Avon.
A banana cream pie.
> examine spotlights
High above the stage, brilliant white spotlights illuminate the actors.
> get pie
Ok.
> eat pie
It doesn't look very appetizing. Some pies weren't meant for eating.
> say Hi, Will!
You exclaim: Hi, Will!
William Shakespeare says: A most amiable day to you, sir.
> smile shakespeare
You smile happily at William Shakespeare.
William Shakespeare bows with a flourish.
```

Figure 8.1: A sample interaction in a MUD

Colossal Cave [30]. However, unlike those single-player games, these systems support many users simultaneously, allowing them to explore environments composed of rooms, objects, other people, and intelligent agents. Depending on the design of a particular system, themes vary from fantasy environments with dragons and wizards, to futuristic worlds with spaceships and aliens, to educational settings complete with regular lectures and conferences.

Within a MUD, each user may pick up and manipulate objects, move from one location to another, interact with other beings or perform more complex actions such as playing games, sending electronic mail, or creating new objects. Figure 8.1 gives a typical example of interaction in a MUD. The user's commands appear in boldface.

In this example, the user's avatar is standing on the stage of a theater, together with William Shakespeare and a banana cream pie. (The opportunities for mischief are obvious.) By using a stylized English syntax, the user can examine the room,

pick up and manipulate the pie, and talk with Shakespeare. Shakespeare is a very limited kind of agent that can recognize certain kinds of actions, such as saying hello or smiling, and react to them. Lacking sound and graphics, such worlds can still be compelling ones: text-based worlds, like books, allow the user to imagine the world, rather than forcing the user to accept a crudely-rendered sensory approximation as the current MMORPGs do.

8.2.1 Properties

A traditional adventure game allows a single player to explore a fantastic environment filled with opponents and treasures (e.g., [19]). Progress is made by exploring and manipulating the environment in order to defeat enemies and solve complex puzzles. Often, but not always, the player has a general idea of an ultimate task to be completed (such as defeating the evil wizard, saving the space station from destruction, finding the cure for a terrible disease, etc.). For our purposes, the salient properties of a traditional adventure game environment are (for more detailed analyses, see [3, 4, 12, 82]):

1. The environment is relatively static. The player is the principal actor and the majority of effects are the results of his actions. If, for example, the player drops an object and departs, the object can be expected to remain until the player returns.
2. Effects of actions are local. Effects are also ordinarily also immediate, so a player pushing a button may expect to perceive the consequences directly.
3. The environment is discrete. The environment is composed of objects (caves, coins, swords, trolls, levers, doors, etc.) distinct from one another.
4. Actions are discrete. A player may turn a knob, open a door, climb a staircase, light a lantern, but actions do not overlap in time and are atomic.
5. The environment is not noisy. Unless the designer explicitly introduces it, there is no uncertainty either in sensing or in action. Items do not slip from the

player's grasp, the player does occasionally bump into a doorframe, have problems hearing what another character says, etc. Moreover, only and exactly the salient information about the state of the world is made available and it is universally accessible.

6. Communication is symbolic. Both perception and action occur through a symbolic text-based interface, so low-level sensory issues don't have to be dealt with. This property allows us to place a character within a complex and dynamic environment without also forcing us to deal with all the sensory issues involved in the real world.
7. The player does not initially have enough information to solve the game. The player does not know the disposition of objects or their uses, the proper solutions to puzzles, the ways to defeat opponents, etc. These can only be learned through observation and experimentation.
8. From any given state, it is always possible to solve the game. Some early adventure games permitted players to act in such a way as to make the game unsolvable (throwing the sword into a pit made the dragon unkillable) but current philosophy holds that the game should prevent the player from taking such actions. Thus there are no actions that should never be attempted.
9. The solutions to game problems are decomposable, and may be solved in a bottom-up fashion.
10. Commonsense knowledge is required. In most cases it is not made explicit what action will achieve an effect, and the player must guess at likely actions. That bottles contain liquids, a light source is needed to see in a dark area, and even that a dragon's breath can melt ice are facts the game assumes the player will know.

While what we have implemented is very much in the style of a traditional text adventure game, we have chosen to do it in an environment that permits much more

sophisticated qualities [33]. MUDs were inspired by, and are superficially similar to, text-based adventure game worlds, but have several other critical qualities:

1. MUDs have generally-agreed syntax and semantics. That is, many MUDs are currently operating for a wide variety of purposes, but almost all agree on the basic syntax of action and the meanings of actions such as “smile shakespeare.” This means that our character will require minimal reconfiguration as it goes from one system to another. It also means that there is a wide variety of example worlds from which to draw inspiration, and into which our guide might travel.
2. MUDs are programmable. The MUD platform³ we use contains its own object-oriented variant of C, giving us the flexibility of a general programming language, but one that is tailored to the creation of these environments.
3. MUDs are server-based, persistent worlds that allow multiple simultaneous connections. Thus we can experiment with a single character or with collections of characters operating independently, cooperatively, etc.
4. We can simplify any of the above game conditions. Thus we can assist a character by providing additional information in circumstances where commonsense knowledge would ordinarily be required; we can introduce explicit goals to guide its behavior; we can make the environment entirely static except for the direct results of a character’s behavior, and so forth.
5. We can relax any of the above constraints, making the problem more challenging. Thus we can make the environment very dynamic, with objects and properties regularly changing; we can add noise, so that actions do not always succeed or perception is not entirely reliable; we can have actions with temporally- or spatially-distant effects or with nondeterministic effects; we can model action in continuous spaces, if we are concerned about subtle variations in the character’s actions.

³There are many types of MUDs freely available, including LP-MUDs, DikuMUDs, MUSHes, MUCKs, etc., with significant variations in the back-end systems but presenting a fairly uniform front end, though some are far more elaborate than others in what they can do. We are using the MudOS platform with the Lima MUD libraries [136].

6. Elements of the environment are running code rather than database objects; they can perform arbitrary computations autonomously or when the character interacts with them. For example, when a character examines a door, the door may provide different sensory data depending on its state, the character's properties, the environment's properties, etc.

Of course, with the increasing popularity of graphical environments (and even graphical enhancements to text-based MUDs), we see no reason why our architecture cannot be adapted to more visually exciting interfaces. However, this would require a much more elaborate interface between the guide and the world, and this is peripheral to our research at best. Most of the contemporary graphical worlds also have the disadvantage of being severely impoverished in terms of the range and sophistication of actions that can be performed in them⁴ For our work, text-based environments provide an excellent medium.

8.2.2 The game setting

The setting of the game we have created is the mansion of a reclusive, mysterious wizard. The player's goal is to find that wizard in order to get help in deciphering a clue as to the whereabouts of an ancient treasure. To find the wizard, the player will have to solve a number of problems scattered around the mansion and grounds. Locked doors, malfunctioning machinery, nasty guardians, and unknown devices must be overcome. The environment also provides games-within-the-game that explorers can play with one another (we will discuss one in Section 8.6).

Figures 8.2 and 8.3 show maps of the manor house and its immediate environment. Beneath the box corresponding to each location is a list of the objects found in that

⁴At this point, I suppose it's fairly clear that I don't think especially highly of today's MMORPGs. From a technical and a graphical point of view they are extremely impressive; but from the point of view of a gamer, I find them generally much weaker than their text-based predecessors. They are prettier but less compelling and less functional. It has taken a decade of MMORPG development before we have begun to move back toward games that offer more than simply swinging swords and casting spells, and perhaps most importantly, to reach commercial worlds that are beginning to offer the players the opportunity to extend and enhance the game worlds, which was the main reason that MUDs have been so successful. The best and most interesting example of this trend is BioWare's *Neverwinter Nights* engine.

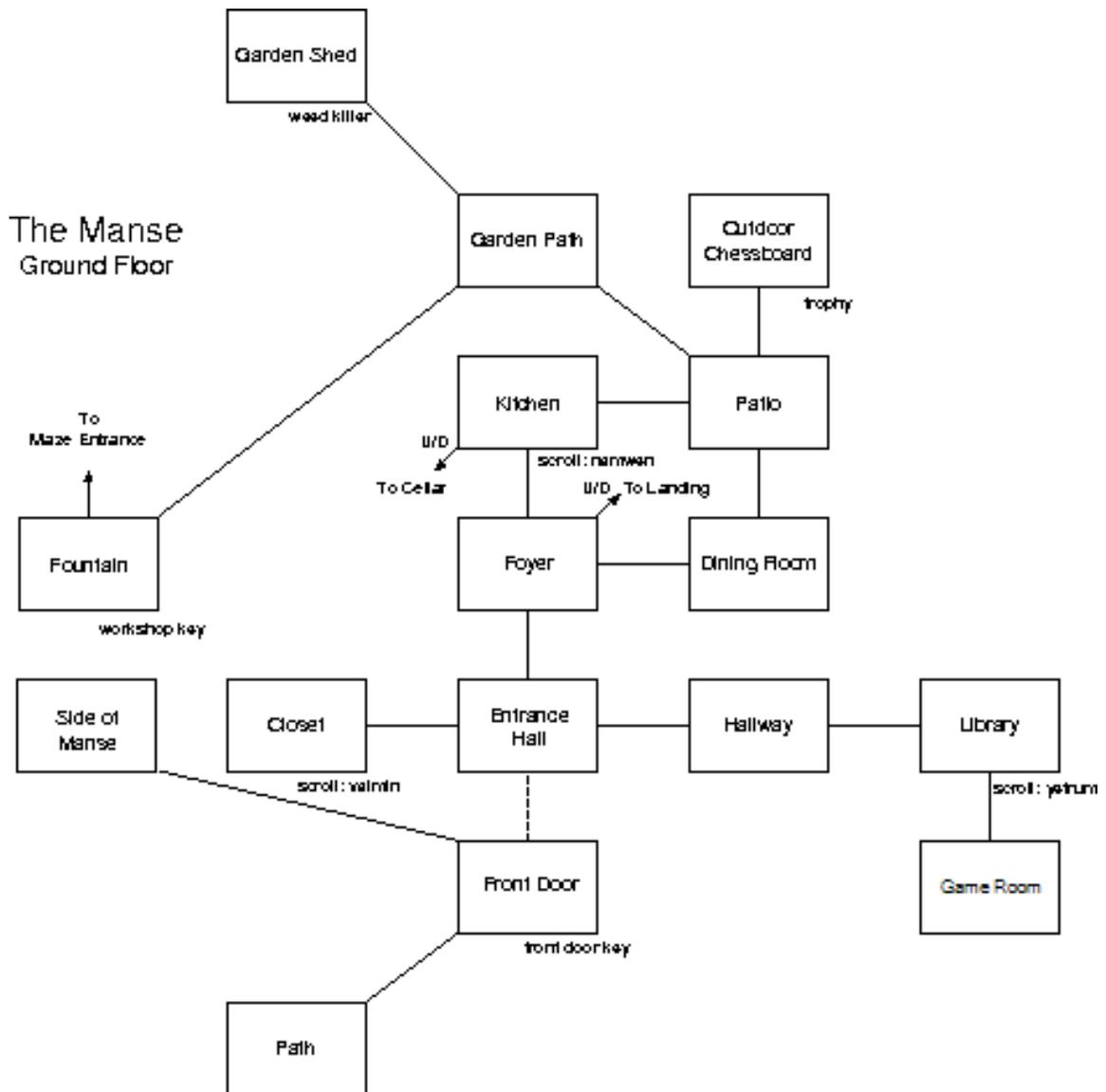


Figure 8.2: Main floor of manor house (used in Section 8.6.5)

location so, for example, the scroll with the *yefrum* spell can be found in the library. We will not go into further detail about the adventure here; while in this chapter we will analyze transcripts of interactions between a player and a character in the environment, we do not provide an exhaustive description of the entire game and its solution, as it is somewhat tangential to the thesis.

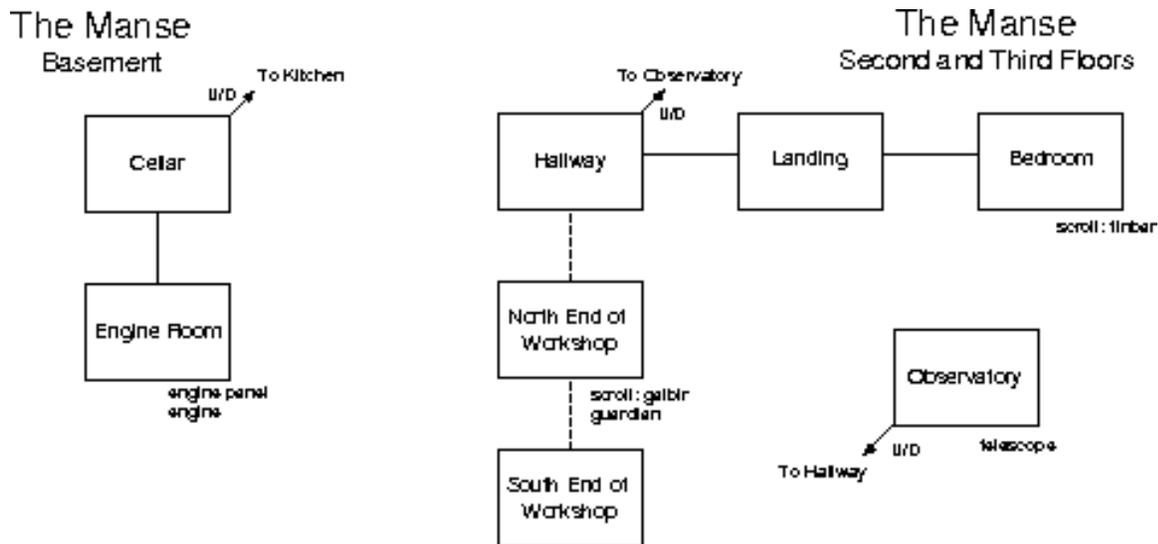


Figure 8.3: Other floors of manor house (used in Section 8.5.3)

8.2.3 The environment's annotations

The environment is annotated in order for characters to interact with it more effectively. The assumption behind these annotations is that any character that can connect to the environment is capable of executing the basic and generic MUD commands, including movement, picking up and dropping objects, speaking, and so forth.

There are annotations to describe

1. **The structure of each location.** When a character enters a location, an annotation provides a description of its contents, the entrances and exits and their destinations, and the affective properties of the location.
2. **Affective content.** Objects, entities, locations, and events may all have affective properties; that is to say, they are intended to engender emotional responses in the explorers of the environment, be they player or animate character. These annotations describe the intended reactions of a “normal” interactor.
3. **Puzzles in the game.** There are annotations that indicate the existence of formal problems to be solved (e.g., the workshop door must be unlocked, which requires a key), what the steps to solve those problems are, the objects they

require, and the actions that must be taken to achieve them. They also describe hints that can be provided to a player about how to solve these puzzles.

4. **Game interactions.** Within the adventure game world, there are smaller, structured games that can be played. These annotations explain the existence of a game, how moves can be made, what the state of play is, and what kinds of recommendations the character can ask of the game itself; that is, the character can inquire about good moves to make depending on its professed skill level and personality.

Our character has been built to make use of these annotations. Once we have described what his personality and goals are, we will discuss these annotations in more detail.

8.3 The character: Harker

An adventure game is particularly good for exploring two kinds of characters — a guide and a sidekick⁵. In developing this example, we chose to make a character that is something of both. Harker is a faithful assistant for the player, one who will follow along, step and fetch, and be obedient to the player's comments. Intuitively, Harker is based on the idea of the sidekick from the pulp adventures of the 1930's and 1940's. However, he is also able to give advice and solve problems on his own if the player appears to be having trouble. The goal is to produce a character that is subservient to the player's needs, but at the same time exhibits independent goals, interests, and reactions to the environment⁶.

⁵We believe, at least for games, that there are a few archetypes that cover virtually all character interactions; for more details, see [32]

⁶Two earlier versions of this character were implemented in a different MUD setting using a more circumscribed version of the architecture that was actually implemented in the MudOS LPC language within the MUD environment itself. For details on these characters and their capabilities, see [35].

8.3.1 Goals

Harker has several tasks to perform while accompanying the player through the game environment.

Functionally, his primary goals are to provide assistance and guidance to the player. Assistance can come in several forms. As they explore, Harker builds a map of the game world, which enables him to answer queries about navigation (such as how to get from one location to another), where previously-seen objects are to be found, etc. He can carry objects for the player and produce them on demand; he can go and fetch objects from other locations, provided he knows where they are.

Consistent with the kinds of annotations we are using in this example, Harker also provides help and hints with the puzzles of the environment. If he has encountered the appropriate annotations, he can give hints about how to proceed with unsolved puzzles, and he can actually take the necessary steps to solve puzzles himself if he is asked to do so by the player.

Harker is also supposed to play games in the world with the player. If the annotations are available, he can offer to play, agree to play, and make moves.

His behavioral goals are to perform these actions in ways that express and are consistent with his personality. He also has a range of behaviors that are not strictly functional but which reveal something about himself, including storytelling, emotional reactions to the environment and its contents, etc.

The design goal of the character is that he be helpful, encouraging, and foster an interest in and enjoyment of the game experience on the part of the player.

8.3.2 Personality and emotions

Harker is meant to be a friendly, faithful, intelligent companion for the player. However, in order to provide some conflict in his character, he is also fretful and nervous.

Harker's emotional makeup is modeled by the translation functions between the emotional influences from his percepts (whether they are annotations or perceptions of standard MUD communications that he recognizes) and the strengths that those

inputs produce. Figure 5.5 illustrated an example of one of these translation functions for Harker, namely the function for his **FEAR** emotion. Specifically, it takes few, weak **FEAR** inputs to produce a strong emotion of **FEAR** in Harker, and the decay rate of those inputs is set to be slow, so that he stays frightened for some time after receiving a fright. Fear is manifest by a **FearBehavior** that chooses among about twenty expressive actions (clearing his throat, shaking, sweating, etc.), and also positively influences his behaviors associated with providing helpful hints to the player (the idea being that the more afraid Harker is, the more he wants to get things over with), so that the value of Harker's **FEAR** emotion increases the computed fitness of the **GiveHintBehavior**, while it decreases fitness of behaviors like **StorytellingBehavior** and **GameOfferGameBehavior** (offering to play a game with the player).

The primary negative emotions for Harker are **FEAR**, **REPROACH**, **DISLIKE**, and **BOREDOM**. As described in Chapter 5, **REPROACH** is engendered when Harker perceives the player (or another living entity) act in a way that is contrary to his standards of behavior. In practice, Harker has perceptors to recognize a variety of standard negative MUD behaviors such as kicking, striking, scolding, yelling, etc., and when he perceives these targeting him, he adds emotion inputs for a **REPROACH** emotion. **DISLIKE** is targeted toward objects, though he has no built-in perceptors for **DISLIKE**; that is, there are no objects he inherently dislikes, but must depend on annotations in the game world to inform him that certain things are likely to be unlikeable⁷.

Harker's model (that is, his translation function) for **REPROACH** is similar to, though not as strong as, his model for **FEAR**. He requires more stimulus to feel reproachful toward the player, and his feelings of **REPROACH** do not last as long. **REPROACH** manifests itself principally through a **ReproachBehavior** parameterized by the strength of that emotion to choose among several dozen surface representations (from muttering under his breath to threatening the player).

BOREDOM occurs when nothing is happening in the environment. As described in

⁷If there were not only standard actions but a sufficiently wide range of standard items across many MUDs, it would be reasonable to give Harker attitudes toward those objects. Frequently in fantasy MUDs one encounters swords, shields, gold coins, etc., and so, for example, Harker could be given a built-in liking of gold with no difficulty.

Chapter 5, an emotion is produced by the architecture by examining all of the emotion influences corresponding to that emotion, combining those inputs, and applying a transform function to determine the strength of the emotion. These inputs may come from any source. Ordinarily they are produced by the environment, but in this case, as time passes and the character perceives no new events (the player does not move or speak, nothing changes in the state of the world, etc.), an internal mechanism adds **BOREDOM** inputs. The effect is that the character becomes increasingly bored, and if **BOREDOM** becomes sufficiently strong, will begin to select a **BoredomBehavior** that is expressed through yawning, fidgeting, complaining about lack of activity, prodding the player, and so forth.

Harker's positive emotions are **LIKING**, **JOY**, **PRIDE** and **RELIEF**. **JOY** occurs when he or the player makes progress in solving a puzzle in the game (more strongly when Harker himself performs the necessary actions). Harker feels a sense of **PRIDE** when he executes behaviors that assist the player, either by giving hints or helping to solve puzzles; when the player performs actions that Harker recognizes as complimentary to Harker; and when Harker performs well in playing a game against the player (making a good move, or especially when winning).

8.4 Affective context

The first category of annotations we wish to consider for this adventure game are those describing the affective qualities of the environment. Many of the qualities of animate characters that we described in Chapter 2 — personality, emotion, expressiveness, situatedness — are, in a narrative setting, dependent upon the character behaving in ways that are consistent with the emotional context. A bloody battlefield should be frightening; a victory parade should be joyful and produce a sense of pride; the sudden end of a threat should be a relief. Since a character cannot always understand what is happening in the environment, annotations can provide an indication of how a “normal” character might react.

It is important to observe that this is not simply a convenient workaround for characters with insufficient intelligence. It would be more accurate, especially in

dramatic settings such as games, to think of these kinds of annotations as akin to a director's directions to a group of actors. A script does not completely specify the action; one interpretation may be very different from another; and the designer of the environment has an intent in the way it is built, to produce some kind of dramatic effect, and an annotation is a way for that designer to be explicit about what that effect should be.

8.4.1 Distribution of knowledge

Our goal is to enable a character to behave believably with respect to the environment. This requires, among other things, an ability to react with appropriate emotion. As we have said, even for a character with a sufficiently high level of human-like perception, natural language understanding, background knowledge, and reasoning abilities (a thing we do not have), there is still no guarantee that it will come to the correct conclusions about the effect the designer of the environment is trying to produce.

That said, what kinds of information does the environment need to provide for a character to read in an emotionally "appropriate" way? Clearly, the character needs to know what emotion is meant to be expressed, and how strongly. To embed this in an annotation, there must be agreement about what types of emotion exist. There are many taxonomies, but we have chosen to use the Ortony, Clore and Collins model, as we have already described at length. We also need a way of measuring the strength of an emotion.

This would provide a very basic knowledge of how to react, but is lacking some important properties. How does the character know what to react to? We also need to specify a *source* for the emotion, whether it be the place, someone or something, an event that has occurred, etc.

Lastly, and most importantly, there must be some *cause* ascribed as the reason for the emotion. *Why* is the burning room producing a feeling of fear, for example? The annotation of emotion is predicated on the assumption that there is some "normal" character who would react with the emotions and strengths that the designer has in mind. No character is going to completely match that normal model; each has its

own quirks, strengths, and weaknesses.

Unfortunately, cause can be a very complex thing to define. In the example of the burning room, the cause of fear is the fire and the heat; but the cause of that fear is that fire, heat, and smoke can cause injury; the cause of that fear is the fear of being injured or even killed; and the cause of that fear is that people generally don't want to die.

It would take a very elaborate and detailed and ultimately impractical annotation to explain all that⁸, and for many purposes that amount of knowledge is not needed. While it is true that it would enable a character who is immune to fire, for example, to react with indifference, we must balance that possibility against the impracticability of representing that amount of detail. Instead, we content ourselves with the highest- (or lowest-) level goals such a character is likely to have: to avoid dying, for example. This also allows a basic ability for characters to react differently based on their personalities. A character who relishes danger may, rather than treating the burning room as fearful, choose to be satisfied about it instead. A dour character who wins a game may choose to dislike the loser rather than to enjoy the victory, etc.

8.4.2 The annotations

Affect is communicated through **emotion-source** annotations. These annotations may be attached to any abstract kind of element of the environment — entities, events, objects, or relationships, as we discussed in Chapter 4, as any could conceivably engender an emotion in a character. Listing 8.1 shows an annotation describing a location in the game world (the edge of a dark pit) containing an emotion source. The example in the next section will show how it affects our Harker character.

The attributes of the **emotion-source** come directly from our discussion of Section 8.4.1. The **id** is a unique identifier of the game element that causes the emotion.

⁸Something like the Cyc project [83] attempts to do just this kind of thing, but has been hard at work for well over a decade on a massive knowledge base. Although some parts of this work are available, it is still not a feasible approach for annotations of the sort we are interested in; they are simply too large and require too much processing to be consistent with our goal of small amounts of knowledge and light agents.

Goal	Description
preservation-of-well-being	The character wishes to preserve itself in order to continue to attempt to satisfy its other goals.
be-helpful-to-user	The character wishes to be useful to the player through the exercise of its knowledge and abilities.
explore-the-world	The character wishes to visit parts of the environment it has not yet visited.
experiment	The character wishes to try new actions made available by the environment.
solve-problems	The environment contains problems that need to be solved; the character wishes to solve these problems.

Table 8.1: Character goals referenced in affective annotations

The **type** is one of the OCC model emotions described in Section 5.5.2⁹ An emotion source’s **strength** is a real value from 0.0 to 100.0, with 100 representing the strongest possible value.

The **source-type** of an emotion indicates the kind of element it comes from. Currently we use **location**, **object**, **event**, and **agent** sources. This is important for determining when the emotion source is a valid one: so, for example, an object source is valid whenever the character is in the presence of the object, and does not weaken over time; an event source has an initial strength but immediately begins to decay; a location source only applies in that place, and an agent source when in proximity to that agent.

The **cause** of an emotion is provides a reason why that emotion would occur. We use a number of causes; all are based on goals we assume that characters interacting with these emotion sources would have. When playing a game in the environment, we assume a character has a goal to win, so **game-won** and **game-lost** are causes of emotion. For any character with a survival instinct, **threat-to-self-preservation**

⁹Note that the emotions listed in Table 5.2 are not all of the emotions used in the OCC model, only those that our architecture currently supports. We did not choose to incorporate into the environment any emotions the architecture could not actually use, though there would be no difficulty in doing so.

and `end-to-threat-of-self-preservation` are causes of emotion, and so forth.

```

<annotation>
  <location id="/wiz/patrick/manse/cave2" name="Edge of Pit">
    <emotion-sources>
      <emotion-source source-type="location"
        cause="threat-to-self-preservation"
        id="/wiz/patrick/manse/cave2" type="FEAR" strength="50"/>
    </emotion-sources>
    <exits>
      <exit name="up" destination="/wiz/patrick/manse/cave"/>
    </exits>
    <inventory/>
  </location>
</annotation>

```

Listing 8.1: Annotation describing affective location

8.4.3 Example: spelunking in dark caves

Figure 8.4 shows the character Harker reacting to affective annotations in the environment. Harker has a built-in mechanism for adding a `BOREDOM` emotion influence on every cycle when nothing happens, so after a period of inactivity, he begins to choose the `BoredomBehavior` (lines 2-4 are all actions from that behavior).

When Harker enters the cave (line 10), he receives the annotation describing the location which contains a `FEAR` emotion source (see Listing 8.1. The architecture creates an emotion influence corresponding to this source. When Harker recomputes the strengths of his emotions, `FEAR` is now the strongest. This causes him to begin a transition of his dominant expression (Section 5.5.5) from boredom to one of fear. Line 11 exhibits a `TransitionToFearBehavior`, while line 12 is the `FearBehavior`.

When the player and Harker descend into the deeper cave, he receives a similar annotation but with a much stronger source. His dominant expression is already fear,

- | |
|--|
| <p>(1) Harker enters.</p> <p>(2) Harker looks around.</p> <p>(3) Harker sighs.</p> <p>(4) Harker says: Wake me up when you plan to do something.</p> <p>(5) > say follow me</p> <p>(6) You say: follow me.</p> <p>(7) Harker says: I'm following you.</p> <p>(8) > go cave</p> <p>(9) Dark Cave [exits: down, out]</p> <p style="padding-left: 40px;">This is a small, dark cave. A passageway slopes down into the gloom. There is a disturbing smell in the air, and there is something unwholesome about the place.</p> <p>(10) Harker enters.</p> <p>(11) Harker suddenly appears to realize the gravity of the situation.</p> <p>(12) Harker says: Maybe this isn't such a great place to be right now.</p> <p>(13) > go down</p> <p>(14) Edge of Pit [exits: up]</p> <p style="padding-left: 40px;">The steep, narrow passage ends at the crumbling lip of a pit descending into darkness. There are many bones strewn about here. A loud, rasping, slavering sound is coming from the depths.</p> <p>(15) Harker enters.</p> <p>(16) Harker wrings his hands nervously.</p> <p>(17) Harker is surreptitiously trying to cower behind you.</p> |
|--|

Figure 8.4: Harker reacts emotionally to environment

and when the new source creates a new, stronger emotion influence, it leads him to instantiate more examples of the `FearBehavior` (lines 16-17).

Note that if Harker left these locations, the influences producing the fear would immediately become inactive (because their sources are no longer present), and he would likely transition to some other dominant emotional expression.

8.5 Puzzles

Traditionally, adventure games consist of collections of puzzles tied together by the arc of a story. The puzzles are the obstacles that must be overcome to tell the story and to bring it to a successful conclusion. The entertainment comes both from the

story itself and from the defeat of these problems. In terms of our categories from Chapter 4, the environment consists of two kinds of entities, obstacles and tools, together with some set of operations on them. The player’s goal is to solve these puzzles by recognizing unobvious relationships between the obstacles and the tools in the game, and by operating upon the obstacles with the tools in order to overcome them. In most puzzle games, some of these operations and tools are “red herrings.” Red herrings are those entities — be they agents, objects, locations — or operations that have no productive effect in the telling of the story or the solution of the puzzles; they are there only to confuse and confound the player.

The solution of an adventure game can be regarded, in the parlance of artificial intelligence research, as a partially-ordered plan. There are a number of steps of the form, “In the presence of these entities, perform these operations.” Sometimes the entities are locations, sometimes they are objects the player may find, sometimes they are other agents, but always the steps are about manipulating the environment. At any given point, one or more of these steps may be performed by the player, though some will be dependent on the completion of others first. This is why we say it is *partially-ordered*: there are some restrictions on which steps precede others, but it is not necessarily totally linear.

8.5.1 Distribution of knowledge

Most, if not all, adventure games can be described in the above terms. That being the case, we may hope that we can capture the abstract knowledge of how to solve adventure games in the character’s behaviors, while we put the specifics into the annotations.

We assume that the character understands an adventure game world as consisting of a set of discrete puzzles. Each puzzle requires for its solution a sequence of steps. Specifically, for each puzzle in the game, a character needs to know:

- **The steps in the puzzle.** A puzzle has a set of states; associated with each puzzle is a sequence of steps¹⁰. intended to move it through its states until a

¹⁰The partial ordering can either be in the list of puzzles or in the list of steps of each individual

satisfactory final state is achieved.

Both steps and states are represented to account for failures in the execution of the plan. For example, a puzzle may require a character to get a stick of wood, light it, and use it as a torch so that it can enter a dark room and retrieve a missing cat. However, if the character (for whatever reason) doesn't go into the room soon enough, the torch will burn out, and the character will have to go acquire a new one.

For each step, a character needs to know:

- **The preconditions.** The character may need to be in a particular location and/or may need certain objects. It may also require that other puzzles have been completed. It is assumed that all of the previous steps in this puzzle have also been completed.
- **The actions to perform.** Given that all of the preconditions are satisfied, what operations must the character perform?
- **The results of these actions.** Possibly before, but certainly after, performing these actions, the character must know that the step has been completed (or has not been), and what the new state is.

Assuming that the character is capable of exploring the game world and of gathering the entities in that world, this information is sufficient for it to eventually solve the game itself. However, the point is not to enable the character to solve the game¹¹ but to enable the player to enjoy that experience. This may, in some circumstances, require the character to take some of these actions; but generally the character should only provide guidance. This requires a different kind of information, not about how to act but about how to lead the player in acting, namely

- **Hints about how to complete each step.**

puzzle, or both; for clarity's sake we assume that each puzzle is strictly totally linear, while the puzzles themselves are partially ordered.

¹¹Of course, we could imagine agents whose sole purpose it was to prepare walkthroughs for exceptionally lazy players or for game testers, but this is not where our interests lie.

A character can solve the game’s puzzles mechanically, without anything like a player’s understanding of what the puzzle actually means, either in the simple terms of the operations or the narrative terms of the game’s story itself. However, the principal source of interest for a player in the game is in the experience of solving those puzzles, of making the intuitive leaps required to realize the actions needed to overcome the game’s obstacles. Thus what the game’s designer would want from a character is for it to gently nudge a player toward those leaps.

Such a nudge is not an easy thing to construct, especially when the character has no real comprehension of the circumstance. Making a character blurt out the necessary preconditions of a puzzle’s current step would be to provide far too much information. The alternative is something like the classic *Lassie* stories, where the dog barks as if to say, *Pay attention to **this!***

In our basic implementation, hints are canned text for the agent to recite to the player. They are organized according to their importance — minor hints, major hints, and the actual solution — and given numerical IDs to order them, so that some precede others. A hint may also have preconditions associated with it, such as that it cannot be given until the character has seen a certain object or unless the character and the player are in a particular location.

The most obvious problem is that the form of that text may be at variance with the character’s personality, habits of speech, etc. As usual, a better approach is to provide more knowledge about the meaning.

Wedding our discussion of objects of discourse from Chapter 4 with the triad of obstacles, entities, and actions, we can see that such knowledge would look like this:

- **Type of the focus.** Entity, operation, relationship: Does the hint focus on one of the obstacles in the puzzle, on a tool that is needed to overcome it, on the operation that must be used, or on the relationship between two (or more)?
- **The focus of the hint.** Is the focus an element of the environment (e.g., a key, an “unlocking” operation), or is it abstract knowledge? If the focus is in

the environment, the character can lead the player to it, point it out, examine it, make passing reference to it, etc.

- **Purpose of the hint.** Is the hint meant for the player to try an operation (“I wonder what happens if you fiddle with that [lever]?”), to get or discard some object (“I wouldn’t be surprised if we need the [box of matches].”), or to think about a relationship (“Do you think [the glacier] has anything to do with [the dragon]?”).

These additional pieces of information would allow a character to perform at least a basic level of template-filling to produce a range of in-character utterances for its hints. This knowledge would not be difficult to add; it was simply left as a future piece of work that we have not yet performed, but that we felt should be discussed here.

8.5.2 The annotations

Listing B.1 shows the annotation that describes a simple puzzle in the adventure game. At the entrance to the wizard’s laboratory, there is a giant, carnivorous plant that blocks access to the rest of the room. The solution is to get a spray can of weed killer from the garden shed in the back of the house and to spray the guardian with it; this will cause it to shrivel up and die¹².

```
<puzzle source="/wiz/patrick/manse/guardian.xml" state="alive"
id="guardian-puzzle">
  <context>
    <location id="/wiz/patrick/manse/workshop_north">
      the wizard's workshop</location>
    <focus id="guardian">the plant guardian</focus>
  </context>
  <states>
```

¹²Figures 8.2 and 8.3 show the locations of “North End of Workshop” and “Garden Shed” in the game.

```

<state id="alive" initial-state="1" possible-steps="1"
  desc="guardian still alive''/>
<state id="dead" final-state="1" desc="guardian dead and gone"/>
</states>
<steps>
  <step id="1" desc="spray the guardian">
    <preconditions>
      <condition type="requires-item" id="can of weed killer"/>
      <condition type="requires-location"
        id="/wiz/patrick/manse/workshop_north"/>
      <condition type="in-state" id="alive"/>
    </preconditions>
    <actions>
      <action>spray guardian with weed killer</action>
    </actions>
    <effects>
      <effect type="leads-to-state" id="dead"/>
    </effects>
    <hints>
      <hint type="minor" rank="1">
        We need some way to get rid of that guardian.</hint>
      <hint type="minor" rank="2">
        It's a kind of plant; maybe we can use that against it.</hint>
      <hint type="minor" rank="3">The gardens around here are
        beautiful; obviously somebody takes good care of them.</hint>
      <hint type="major" rank="1">Maybe there's something in the
        garden that will help against that monstrous plant.</hint>
      <hint type="major" rank="2">I'll bet the gardener has a
        way to take care of weeds, even big ones like this thing.</hint>
      <hint type="major" rank="3">Maybe we should take another
        look in the gardener's shed.
      <preconditions>
        <condition type="know-location"

```

```

        id="/wiz/patrick/manse/shed"/>
    </preconditions>
</hint>
<hint type="solution" rank="1">We need to get weed killer
from the garden outside.
    <preconditions>
        <condition type="know-entity" id="weed killer"/>
    </preconditions>
</hint>
<hint type="solution" rank="2">Spray the guardian with the
weed killer; that should take care of it.</hint>
</hints>
</step>
</steps>
</puzzle>

```

Listing 8.2: Annotation describing guardian puzzle

Below is a brief summary of the more interesting elements of the markup. For ease of reading, we represent the structural hierarchy of elements using dots (.) to separate elements from their children, so “puzzle.context” corresponds in our XML notation to `<puzzle><context>...`

puzzle

The top-level tag gives this puzzle a unique identifier (`guardian-puzzle`) which will be used by other annotations. The `state` attribute is dynamically generated according to the current state of the puzzle.

puzzle.states.state

The `initial-state` and `final-state` attributes are flags indicating whether they are the first state in the puzzle or one of its successful or unsuccessful outcomes. There is only one initial state but there may be multiple final states.

puzzle.steps.step.preconditions.condition

<p>Conditions indicate what must hold before the character can execute the actions to move the puzzle to another step. <code>requires-item</code> means that the character must have the object; <code>requires-location</code> means that the character must be in that location. (There are others, such as <code>requires-item-present</code>, not used here.)</p> <p><code>puzzle.steps.step.actions.action</code></p> <p>The string that must be sent to the MUD to solve the puzzle. (Note that there may be multiple actions required.)</p> <p><code>puzzle.steps.step.hints.hint</code></p> <p>Note that an individual hint may have preconditions of its own about when it is appropriate to use. The <code>rank</code> attribute is a relative indication of how much information that hint gives; in our examples, Harker moves from minor to major hints and from lower- to higher-ranked hints. A character with a player having considerable difficulty could immediately move to higher-ranked hints, which is why ranking is not the same as a required order.</p>
--

Table 8.2: Notable annotations in puzzle description

8.5.3 Example: the wizard's workshop

Figure 8.5 shows an interaction between Harker and a player in the game world as they solve the guardian puzzle described above.

The beginning of the transcript (lines 1-27) are merely setting the stage for the interesting interactions. In line 2, the player orders Harker to follow. Harker has a task, `FollowingTask`, for dealing with this. In that task is a behavior, `StartFollowingBehavior`, which has associated with it a perceptor that waits for that request. When a follow command is in the queue, this behavior has a maximal fitness, and it is selected for execution in line 3.

The other point of note in these early lines is that the player and Harker visit the Garden Shed (line 8), where there is a spray can of weed killer. Harker stores

(1) > **say follow me**

(2) You say: follow me.

(3) Harker says: I'm following you.

(4) > **go garden**

(5) Garden Path [exits: southeast, north, southwest]

This gravel path wends its way among the bright flowerbeds to the north of the manse. Paths lead southwest and southeast back toward the house, and a smaller one cuts north toward the trees at the edge of the garden.

(6) Harker enters.

(7) > **go north**

(8) Garden Shed [exits: south]

This is a small wooden shed among the trees at the northern end of the garden. Seeds, fertilizer, watering cans, and other garden implements are stacked on the shelves and against the walls.

There is a can of weed killer on the shelf.

(9) Harker enters.

(10) > **go south**

(11) Garden Path [exits: southeast, north, southwest]

This gravel path wends its way among the bright flowerbeds to the north of the manse. Paths lead southwest and southeast back toward the house, and a smaller one cuts north toward the trees at the edge of the garden.

(12) Harker enters.

(13) > **go southeast**

(14) Patio [exits: northwest, west, south, north]

This flagstone-paved patio has a wrought-iron table and chairs for sitting outdoors on pleasant days. To the east and northwest, well-tended flowerbeds stretch away toward the forest. Gravel paths lead northwest around the manse and north up a nearby hill.

(15) Harker enters.

(16) > **go south**

(17) Dining Room [exits: west, north]

A heavy mahogany table dominates this small room. Cabinets along the walls display the china and silver serving dishes ordinarily used here, but they are currently locked away. Curtained double doors open onto a patio to the north, and a doorway leads west into the main hall.

(18) Harker enters.

(19) > **go west**

(20) Main Hall [exits: up, east, south, north]

A large window high on the wall sends sunlight streaming into this hall. A wide arabesque carpet runs from the entrance hall to the south and turns toward the dining room to the east. An elegant marble staircase curves up to the second floor. There is a small door in the north wall, under the curve of the stairs.

(21) Harker enters.

(22) > **go up**

(23) Landing [exits: down, east, west]

This balcony runs all the way around the grand hall below, providing a view of the floor below. A grand marble staircase descends to the first floor, and polished hallways lead off to the east and west.

(24) Harker enters.

(25) > **go west**

(26) Hallway [exits: up, east, *south]

The hallway ends here at a heavy iron door. The door is currently closed. To the north, a narrow stairway leads up.

(27) Harker enters.

(28) Harker says: Why don't you try the rune-covered key?

(29) > **inventory**

You are carrying:

a rune-covered key

a glossy pamphlet

a piece of string

(30) > **unlock door with key**

You unlock the iron door with your rune-covered key.

(31) Harker grins.

(32) > **open door**

You open the iron door.

(33) > **go south**

(34) North End of Workshop [exits: south, north]

This is the northern end of a long workshop. The space is filled with a bizarre collection of equipment: runes and magical diagrams are etched into the floor, and alembics, retorts and other alchemical paraphernalia line the broad worktables. To the south, the room continues into gloom.

A massive, scaly green guardian fills this end of the workshop. Its leaves, tentacles and mouths full with sharp teeth wave and snap at you, but you're just out of its reach when you're standing.

A rumpled scroll rests on one of the worktables.

- (35) Harker enters.
- (36) Harker's eyes open wide as his natural cowardice takes over.
- (37) Harker bites his lower lip.
- (38) Harker says: We need some way to get rid of that guardian.
- (39) > **examine guardian**
This huge...thing...is evidently meant to guard the workshop. It has a bizarre combination of waving leaves, tentacles, and mouths with very sharp, needle-like teeth, and it appears somehow to be rooted in the hardwood floor. Since it's also about eight feet tall, I wouldn't advise going anywhere near it.
- (40) You say: Any suggestions?
- (41) Harker says: It's a kind of plant; maybe we can use that against it.
- (42) > **go south**
The guardian's tentacles lash out at you as you approach. You leap back, narrowly avoiding becoming its dinner.
- (43) > **shout Help!**
You shout: Help!
- (44) Harker says: Hold on, boss! I've got an idea. I'll be right back!
- (45) Harker leaves north.
- (46) The horrible plant writhes and flexes what passes for a mouth.
- (47) Harker enters.
- (48) Harker cautiously edges toward the monstrous plant thing. Suddenly, it senses him and swings its several mouths down to attack! He sprays a healthy dose of weed killer right at it, and it instantly begins to writhe, its tentacles whipping about uncontrollably. After one final spasm, it collapses in a shriveled heap.
- (49) Harker shudders.
- (50) Harker breathes a sigh of relief.
- (51) > **say give me the spray can**
- (52) You say: give me the spray can.
- (53) Harker gives you the spray can.
- (54) > **examine can**
This brightly-colored spray can shows an ugly green weed cowering away from a smiling yellow cloud of gas.

Figure 8.5: Harker helps to solve some puzzles

inventory information about the locations of entities in the environment in his context manager; he will need this later. Lines 10-25 show the path from the Garden Shed to the workshop door.

At line 26, the player is confronted with a locked door. This is actually a puzzle, but it is an exceptionally simple one, and we will not spend time on it here. Harker provides a hint (we will describe how in a moment), the player applies the key in his possession, and they proceed to the workshop.

At 34, the player enters the workshop. Prominent in the description is the tentacled plant guardian. When Harker enters (35), he receives an annotation describing the room and its contents, including both the puzzle to be solved (Listing B.1) and the guardian (for a full transcript of the annotations Harker handles, see Listing A.1 in Appendix A).

These annotations cause several things to happen. First, the annotation handler in Harker that waits for puzzle descriptions fires, creating a new internal structure to describe this puzzle and storing it in the context manager. It then creates a new instance of a generic `PuzzleTask`, instantiating it with the structure of this puzzle, and adds it to the active task list in the behavior engine.

The annotation for the guardian object (it is technically an object and not an agent) indicate that it is an emotion source of `FEAR` as it is perceived to be a threat to self-preservation (see Figure 5.3 for that annotation). Harker handles that annotation by creating an emotion influence corresponding to the object. On subsequent cycles when Harker is in the same location as the guardian, that influence is active. This will translate into his experiencing a `FEAR` emotion (see 5.5.4 for details of how this occurs).

In line 37, Harker expresses the dominant emotion of `FEAR`. Now that he is aware of a puzzle to solve, he also has a high fitness for offering hints to the player about how to do it (this fitness increases over time; it has been artificially inflated here to produce a briefer example). One of the behaviors in the new `PuzzleTask` is a `GiveHintBehavior`, which offers hints from the annotations. He executes this in 38 autonomously, and again in line 41 in response to the user's query for another hint.

When, in line 42, the player tries to go south, this generates an event with another

emotion source of **FEAR** emanating from the guardian. This is treated as the source described above, except that because it is an event rather than an object, its strength will decay over time.

In line 43, the player calls for help. A second behavior in the `PuzzleTask` is the `ObeyGiveHelpBehavior`, which fires when a perceptor in the task recognizes a request for aid. This behavior then creates a new even *more* temporary task whose sole job it is to perform whatever actions are needed to move the puzzle to its next state.

When being created, the new `SolvePuzzleTask` looks at the conditions for solving the puzzle that are not yet met. In this case, Harker does not have the weed killer he needs to spray the plant. It adds therefore adds four new behaviors, to be executed sequentially one after another¹³: a `GoToLocationBehavior` to go to where the weed killer is, a `TakeItemBehavior` to pick it up, a second `GoToLocationBehavior` for returning to the workshop, and a `SolvePuzzleBehavior` for doing the spraying.

A `GoToLocation` behavior is created with a source location and a destination. It refers to the global map in the context manager to make a plan (really a list of moves) to get to the destination from the source. If they are the same, the behavior has no actions. If no path can be found, it returns a failure code which propagates up to the `SolvePuzzleTask`, which then simply has Harker indicate that he doesn't know how to do what needs to be done.

Lines 44-48 show Harker executing this task. He goes down to the shed, gets the spray, returns, and uses it. The temporary `SolvePuzzleTask` is then deleted from the behavior engine when Harker receives a new annotation updating the state of the puzzle to **dead**. The update event contains an emotion source of **RELIEF** at the removal of the guardian, and informs Harker that the guardian is now gone from the room inventory. Harker exhibits this sense of relief in line 50.

This brings us to the end of the puzzle. The player is relieved, Harker is relieved, *I'm* relieved, and I imagine you are, too.

¹³Ordinarily, behaviors in a task are independent of one another, but the architecture does support linking them in this way, so that one can't be executed until another indicates that it should not be chosen.

8.6 Games

When originally considering for what annotations might be especially appropriate, we quickly thought of the idea of using them to enable a character to play a game with a human user. By “game” in this context, rather than a large, open-ended, abstract structure like the adventure world, we mean more traditional, structured, rule-based games such as board games, card games, and so forth. Very many of these sorts of games have computer-based versions where a player can vie against the computer, and the computer takes the roles of one or more opponents. It seemed to us natural that a person in a large virtual world might want to participate in the same kinds of activities; and while sometimes other people may be available, there would be value in being able to play with or against the animate characters in these worlds, especially those with whom the visitor had already developed a relationship. Rather than going to see the single chess-playing character in the library or the draughts player in the bar, it would be much more satisfying if any character could conceivably play, and this seemed an obvious application of the annotation idea.

8.6.1 Distribution of knowledge

Since the idea behind annotating an environment is to divide the knowledge between the character and the environment, we must first ask, what is a reasonable division of knowledge for these sorts of games?

We decided that we would assume that the target character audience understood an abstract concept of a game; that is, that a game consists of certain organized groups of formal actions that have a definite beginning and an end, that involve manipulations of some sort (moves), that the game has some state information associated with it, and that there is at all times a score associated with each participant indicating how well that participant is performing. We allow that a game may have one, two, or more participants and that it may have a turn-taking procedure or players may be able to move according to their preferences. We make the restrictive assumptions that a game is confined to a single location (at least in a MUD), and that all players join the game at its beginning. We also make the assumption that the structure of a

game state can be described in discrete terms.

Given this interpretation, what kinds of information does the environment need to provide to a character? At a bare minimum, the character needs to know:

- **Whether the game is in progress.**
- **What moves are available.**
- **How to make an available move.** The character may know that it *can* move, and even know which move it wishes to make, but it must be given the specific syntax for performing the action in the environment.
- **How to begin a game or resign from a game.**

This information would allow a character to interact by making random legal moves until the game ended. This would not make for especially engaging or believable behavior. For a character to be able to learn anything about the game or to reason about what moves to make, it also would need to know:

- **The state of the game.** That is, if the game is being played, what state it is in, i.e., where the pieces are, who holds cards and how many, etc. In incomplete information games, obviously not all state information is accessible, but it is possible to represent that fact.
- **The scores of all players in the current state.**

Especially for characters with more sophisticated learning or reasoning abilities, we could of course add additional details to classify a game. Whether or not the game contains random elements; whether the game is a perfect-information game (that is, everything about the state of the game is known to all players at all times), whether it is impartial (every player can move any piece), and so forth. However, considering for the moment those characters that do not do learning or for games in which it would not be desirable for a character to experiment at length while trying to learn what to do, the only other piece of necessary knowledge is

- **The strengths of the available moves.**

From the point of view of a mathematical analysis, or the goals of traditional AI, this would be clear (if not straightforward). Traditionally, AI has been concerned with finding evaluation functions that indicate an optimal move – the whole point being to produce “best” game-playing programs, something that has effectively been achieved in many games, checkers and chess among them (see, e.g., [121, 122]). However, when the game is situated in a wider context, the character is not necessarily there solely for the purpose of defeating a human player. So we must instead ask of a potential move, *how fit is it relative to the character’s personality and goals?* Human players are erratic; their levels of skill, their experiences, their preferences for certain kinds of plays, their reactions to other players’ moves, are all part of their behaviors, and we want our characters to have similar dependencies.

8.6.2 Categories of players

One natural way to approach this problem is to begin with a classification of kinds of game players. While there are many possible categorizations, we draw distinctions along five axes that we believe capture many of the essential differences.

A traditional approach to computer game playing is for the computer to search through a space describing various game configurations, determining new configurations by considering legal moves made from existing configurations, and applying an evaluation function to determine the values of points in that space, where the evaluation function consists of terms corresponding to meaningful aspects of the configuration (see, e.g., [121] for a typical discussion of game-tree search). These distinctions we draw among game players can be thought of as resulting from variations of the elements of a traditional game-tree search.

1. Greedy vs. patient. (depth of search)

Greedy players are more concerned with immediate gain than long-term advantage, and may not bother to consider the extended ramifications of their choices. Patient players are apt to look further ahead and to delay gratification

in the expectation of a better ultimate result. In search terms, this distinction corresponds to the depth in the search tree to which the character is willing to go before making a choice. The more patient the player, the deeper the search tree.

2. Aggressive vs. defensive. (evaluation function)

Aggressive players prefer moves that engage the opponent, while defensive players act principally to protect their existing gains. This is just one example of the more general issue of choice of evaluation function. An aggressive player in chess, for example, may prefer to make piece exchanges or to advance across the board into enemy territory, while a more defensive player will rank a board position where pieces are protecting one another more highly.

3. Experienced vs. inexperienced players (application of evaluation function)

Experienced players examine all the ramifications of a move before making it; inexperienced players may miss important effects by failing to look at the outcome in sufficient detail. This is not the same as depth of search; a careful player who is not an experienced player may be unable to see many moves ahead, but can still determine that a certain move puts him in immediate peril, while a careless but knowledgeable player may inadvertently open himself up to checkmate, for example. This corresponds to an imperfect application of the evaluation function to all the board positions under consideration. Certain terms may be omitted from one evaluation to the next, with the result that some boards are rated as better (or worse) than they theoretically would be by the player.

4. Careful vs. careless players (generation of successor states)

A careful player considers every possible move before choosing one; a careless player looks at some that are of interest and picks one of those. This corresponds to not generating all of the successor states when doing a standard game-tree search.

5. Competitive vs. social players (extent of evaluation function)

Social players play games because they enjoy the experience of a challenging interaction with other people. Competitive players are interested primarily in the outcome of the game. The social player may deliberately choose to match his level of play with his opponent's, for example, to keep the opponent in good humor; the competitive player is analogous to a traditional game-playing AI system, whose job it is to win. This division recognizes a distinction in the nature of information considered in the evaluation function. A competitive player is concerned with the state of the game; the social player may take other information beyond the state of the game, such as the other player's mood, personality, the history of wins and losses, the likelihood of future interactions, etc., into consideration when evaluating it.

Notice that the environment would have to offer recommendations along any combination of these five axes – an aggressive, but social, player would have a very different set of preferences from an aggressive competitive player — and they could further be divided by how careful the player is. Ideally, the environment composes a distinct evaluation function for every combination.

Thus the careful, patient, experienced, competitive player corresponds to the traditional AI goal (whether the player is aggressive or defensive is a question of style, not of ability). Choosing the alternative along any of these dimensions results in a more fallible and therefore more believable character.

8.6.3 Applying these categories in Nim

In our adventure game environment, we applied these axes in the context of a two-player game called Nim. Nim is a simple game of bowls and stones thought to have originated in ancient China. The contemporary name was given to it by Charles Bouton of Harvard in 1902; Bouton was also the first person to provide a formal analysis of the game, together with a mathematical strategy for playing it [23, 49].

In its basic incarnation, Nim is a turn-taking game for two players. The game consists of several bowls (usually three), each of which contains some finite number of

stones. Each player, on his or her turn, chooses one bowl and removes some or all of the stones from that bowl. The player who takes the last stone (or group of stones) from the last non-empty bowl wins.

While extremely simple¹⁴, Nim serves our purposes for demonstration nicely, as it is easy to grasp and to consider how players of different personalities might play it differently. So how do our categories apply to Nim?

1. Greedy vs. patient.

In Nim, this corresponds to how many stones the player tries to take. A greedy player will try to take as many stones as possible; a patient player will take as few as he can¹⁵.

2. Aggressive vs. defensive.

In Nim, there is a direct computation to determine which moves constitute wins. Aggressive players will break ties among winning moves by always preferring those that take more stones, while the defensive player will take fewer stones.

3. Experienced vs. inexperienced players.

A small random factor is introduced in the evaluation of every move of inexperienced players, with the size of the error proportional to the number of stones remaining (intuitively, the more stones that are left the more likely an inexperienced player is to make an error).

4. Careful vs. careless.

Some valid moves are not listed when the agent requests a list of recommended moves. This is not the same as saying that those moves have a rating of 0;

¹⁴Nim is the canonical example of a class of games addressed by combinatorial game theory. It is a two-person, impartial, combinatorial, perfect information game. There are a number of variations on Nim, and in fact a considerable literature is devoted this game. The most relevant observation for us is that it has been proved that *all* two-person impartial combinatorial perfect-information games are isomorphic to Nim [17].

¹⁵This does not precisely correspond to depth of the evaluation function, because unlike games such as chess, where “greedy” may require a more complex evaluation of the position, in Nim the intuitive psychological interpretation of greedy is simply to take more stones.

instead, this is telling the agent that those moves are not even possible, which corresponds to the player's overlooking them.

5. Social vs. competitive.

The social player decides at the outset whether to play to win or play to lose (and rates moves accordingly), and acts accordingly; the competitive always plays to win.

A character may query the environment for a list of the available moves, and provide in that query a list of the personality traits it wishes to exhibit (including some, all, or none of the traits the environment recognizes). The environment will reply with a list of the available moves, ranked by the application of the standard game-tree search, modified according to the personality properties as described above.

8.6.4 The annotations

Listing 8.3 shows the initial description of the Nim game and its operations that a character would receive from the Game Room location. Section 8.6 provides an explanation of the design of the annotations.

```
<game source="/wiz/patrick/manse/nim.xml" id="nim">
  <description>
    <name>Nim</name>
    <context>
      <object id="table"/>
      <text>the small black table</text>
    </context>
  </description>
  <classification>
    <number-of-players min="2" max="2"/>
    <order-of-play type="turn-taking"/>
  </classification>
```

```
<gameplay-styles>
  <style id="aggressive"/>
  <style id="defensive"/>
  <style id="careful"/>
  <style id="careless"/>
  <style id="greedy"/>
  <style id="patient"/>
  <style id="experienced"/>
  <style id="inexperienced"/>
  <style id="social"/>
  <style id="competitive"/>
</gameplay-styles>
<structure>
  <positions>
    <position id="bowl-1"/>
    <position id="bowl-2"/>
    <position id="bowl-3"/>
  </positions>
</structure>
<actions>
  <game-controls>
    <start-game>play game with <user/></start-game>
    <restart-game>play game with <user/></restart-game>
    <resign-game>resign game</resign-game>
    <end-game>resign game<end-game>
  </game-controls>
  <queries>
    <query-available-moves>query table available-moves
      </query-available-moves>
    <query-recommended-moves>query table recommended-moves
      </query-recommended-moves>
    <query-players-who-can-move>query table players-who-can-move
      </query-players-who-can-move>
```

```

    <query-game-is-in-progress>query table game-is-in-progress
      </query-game-is-in-progress>
  </queries>
  <game-move-types>
    <move-type id="default">remove <num-stones/> stones from
      bowl <from-bowl/></move-type>
  </game-move-types>
</actions>
</game>

```

Listing 8.3: Annotation describing Nim game

Table 8.3 summarizes the most important of the annotation elements for games. As in the previous section, for ease of reading we represent the structural hierarchy of elements using dots (.) to separate elements from their children, so “game.classification” corresponds in our XML notation to `<game><classification>....`

`game.classification.order-of-play`

Indicates whether the game is turn-taking or free-form, in which case each player can move when he or she chooses. The event annotations which the environment emits when the state of the game changes also indicates which players may legally move in the current state.

`game.gameplay-styles`

Lists the styles of play that the game object’s evaluation mechanism recognizes. The character may query the game object to provide evaluations of available moves according to any combination of the five types discussed in the previous sections.

`game.structure.positions`

Positions are essentially the key-value pairs used to describe the structure of the game state; each position in Nim, for example, is a bowl, and its value is the number of stones in the bowl.¹⁶ An obvious extension would be also to describe all of the possible legal values each position might take on.

`game.actions.game-controls`

These annotations (`start-game`, `restart-game`, etc.) describe the actions the character must send to the environment in order to achieve those effects. In the MUD environment this means the strings it must pass to the parser.

`game.actions.queries.query-available-moves`

The form of the request to the game object that will return an annotation consisting of a list of all legal moves for the character.

`game.actions.queries.query-recommended-moves`

Like `query-available-moves`, except the request can be accompanied by a list of styles from `gameplay-styles` that describe the character's temperament, and returns a list that is rated according to those styles as described in Section 8.6.3.

`game.actions.queries.query-players-who-can-move`

The request to send to receive a list of players who can currently move in the game.

`game.actions.queries.query-game-is-in-progress`

The request to send to determine whether the game is currently being played.

`game.game-move-types.move-types`

¹⁶We mentioned earlier that all games of a certain class (perfect information, impartial, two-player combinatorial games) are isomorphic to Nim, and they can all be described in terms of this bowl-and-stone approach.

The actions the character must execute to play in the game. An individual `move-type` annotation is, in the MUD environment, a string in which are embedded variables that the `query-recommended-moves` annotation defines. For example, `<move-type id="default">remove <num-stones/> stones from bowl <from-bowl/></move-type>` is the default move for Nim; and a recommendation for a move is of the form `<move from-bowl="3" num-stones="2" rating="67.0"/>`.

Table 8.3: Notable annotations in game description

8.6.5 Example: playing a game of Nim

With this extensive preamble, we move on to an example of how a character interaction with the Nim game looks in this environment. Figure 8.6 shows an interaction between Harker and a player. Listing A.2 shows the interaction from Harker’s point of view, i.e., the annotations that the character receives as the game proceeds.

In line 1, Harker enters, and perceives the description of the game shown in Listing 8.3. Harker has an annotation handler, `GameHandler`, that waits for an annotation describing a game available in the environment. When Harker encounters an environment that contains a game, he creates a new internal memory structure that describes it, and it is added to the context manager. When the context manager is informed of a new game, a new instance of a `PlayGameTask` is created and added to the hierarchy in the character’s behavior engine. In this case, a new task for playing Nim is added.

The `PlayGameTask` contains five behaviors: `GameOfferGameBehavior`, `GameStartGameBehavior`, `GameMakeMoveBehavior`, `GameReactToGameBehavior`, and `GameReactToGameEndingBehavior`. Of these, the offers to play and the reactions to the state of the game are behaviors that have many corresponding actions according to the character’s emotions, while the behaviors for starting the game and making moves in the game are driven by the knowledge gained from the annotation as to how to act in the world.

(1) Game Room [exits: west]

This spartan room contains only a small, ornate black table with two chairs. Sitting on the table are three brown stone bowls filled with stones.

(2) > **examine table**

The small black table holds three hand-sized brown stone bowls. Currently the first bowl is empty, the second bowl is empty, and the third bowl is empty. There is a brass plaque affixed to the side of the table immediately above a tiny button.

(3) Harker enters.

(4) > **examine plaque**

The plaque reads:

The game of Nim

Each bowl is filled with several stones. Two players take turns, each player choosing a bowl and then removing some or all of the stones from that bowl. The player to remove the last stone wins.

To play: 'play game with [other player]'

To move: 'remove [x] stones from bowl [1, 2, or 3]'

Good luck!

(5) Harker looks at the small black table.

(6) Harker says: Hey, want to play Nim?

(7) > **say Sure.**

(8) You say: Sure.

(9) Harker presses the button on the side of the game table. The bowls shimmer, and are now filled with stones.

(10) > **examine table**

The small black table holds three hand-sized brown stone bowls. Currently the first bowl has 4 stones, the second bowl has 8 stones, and the third bowl has 6 stones. There is a brass plaque affixed to the side of the table immediately above a tiny button.

(11) Harker removes 6 stones from the second bowl.

(12) Harker nods in satisfaction.

(13) > **remove 2 stones from bowl 1**

(14) You remove 2 stones from the first bowl.

(15) Harker removes 6 stones from the third bowl.

(16) Harker says: Good game so far, huh?

```
(17) > examine table
    The small black table holds three hand-sized brown stone bowls. Currently
    the first bowl has 2 stones, the second bowl has 2 stones, and the third bowl
    is empty. There is a brass plaque affixed to the side of the table immediately
    above a tiny button.
(18) > remove 1 stone from bowl 1
(19) You remove 1 stone from the first bowl.
(20) Harker removes 1 stone from the second bowl.
(21) > remove 1 stone from bowl 2
(22) You remove 1 stone from the second bowl.
(23) Harker removes 1 stone from the first bowl.
(24) Harker is the winner! The bowls shimmer, and the stones vanish from the
    table.
(25) Harker says: Yes!
(26) Harker does a little dance.
(27) Harker says: Another game?
```

Figure 8.6: Harker plays a game of Nim

When there is a game available in the current environment and it is available to play, and when Harker is not in a negative mood (that is, the dominant emotional expression is not negative), there is a strong positive fitness for the `GameOfferGameBehavior`. In line 6, Harker chooses this behavior and proposes a game. This triggers a temporary perceptor that waits for some kind of yes-or-no response from the player.

When the player accepts the offer of a game (line 8), Harker proceeds to begin a new one by executing the `GameStartGameBehavior`, using the action described in the annotations. This activates the game-playing and game-reaction behaviors in the `PlayGameTask`, and also causes Harker to immediately query the game object (the entity in the environment that contained the game annotations) for recommendations about what move to make (line 9, though it is invisible to the player). When he is in a position to move, his `GameMakeMoveBehavior` has a maximal fitness rating, which leads him to choose it (line 11).

Associated with the `GameMakeMoveBehavior` is an emotion generator that is invoked whenever the behavior is executed. This generator produces an input to the

emotion system whose type and strength is dependent upon the strength of the move that Harker chooses. If the move is highly ranked, the generator produces inputs of JOY and SATISFACTION. If the move rates poorly, it produces an input of ANGER. While Harker is playing a game and he cannot move, his `GameReactToGameBehavior` has a high rating, and we see it applied repeatedly here (lines 12 and 16) with different surface actions because of the different strengths of his emotions at the times the behavior is chosen.

When he has moved, his behavior for reacting to the game is rated highly, and influenced especially by his level of BOREDOM, so he rapidly becomes twitchy while waiting for other players to move, though this game didn't take long enough for him to express his reaction as boredom.

After Harker removes the final stone and the game ends (lines 21 and 24), he perceives the annotation informing him that the game is over and he has won. Both winning and losing create strong emotion influences for the `GameReactToGameBehavior`, which is strongly expressed (lines 25–26). His victory adds weight to a subsequent repeat of the `GameOfferGameBehavior` (line 27).

8.7 Lessons Learned

In Chapter 4, we made several claims about what annotations can do for characters in virtual worlds. We now examine the results of this chapter in the light of those claims.

Claim: Annotations enable characters to behave believably in unfamiliar environments.

The affective annotations are the primary mechanism by which this is accomplished. An adventure game is meant to be emotionally engaging for a player, not just because there are problems to be solved — if that were all that the game contains, it would be no different from a puzzle — but also because of the story that the player helps to tell through exploration in the world of the game. As with all stories, there is, or the author hopes that there will be, an emotional reaction to its events.

One of the ways in which Harker becomes believable is by reacting emotionally to what happens in the environment. When navigating in the dark cave, Harker becomes frightened. He is scared by the threatened attack of the bizarre plant creature in the wizard's workshop; he is disquieted when he sees it, and his nervousness becomes more intense when it threatens. He is relieved when it is destroyed. He is alternately proud and reproachful when he plays Nim, and is joyful and proud when he wins the game. All these emotions depend upon not just upon perceiving what is happening in the environment, but upon understanding it, and this requires the use of the annotations.

Note that the puzzle and game annotations do not necessarily contain affective information directly. In the case of the game, for example, he is not told that he should be joyful when he is ahead or proud of himself when he wins. Instead, he is told that he is ahead or that he has won, and it is left to him with his general knowledge of the process of game playing to decide how to react to the game.

Of course, the affective annotations do not tell the character what to do either. Instead they indicate what a "normal" character would do, and again leave it to an individual character to decide how to respond. Harker's responses very often are of fear, while a more burly and self-confident sidekick might either be afraid but mask the expression, or simply perceive those **FEAR** annotations and ignore them or translate them into other emotions.

Our empirical observations for this claim come from the informal interactions of four Ph.D. students in the department, and my own considerable interactions with Harker, both in development and in much time spent tweaking his believability-enhancing behaviors. Each of these four students played through the adventure game, although only one actually completed all of the puzzles.

The general opinion is that Harker is indeed helpful. His ability to provide information about how to proceed with the game puzzles is useful, and somewhat surprising, as it is not superficially obvious in several cases just how, given his limited intelligence, he would be capable of coming to the conclusions that he does. One serious flaw in the current implementation is his over-eagerness to provide help. I did increase the weight on those assistive behaviors, but apparently it was increased too far, as he tends to blurt out hints before people have had a chance to think about the

problem.

Another problem had to do with the limited number of believability-enhancing behaviors Harker possesses. The number is not that small (there are dozens), but unlike Merlyn in the Web environment, where it is possible to glance around innocuously without a viewer paying close attention, whenever Harker makes an action it must be read, and repeated descriptions of the same action falls into the textual version of the problem that the Disney animators [137] called “twins” — unnatural repetition. Certain behaviors, such as his story-telling behavior, could never be exhibited more than once without being highly unbelievable. This problem could be ameliorated by adding many more behaviors to Harker’s repertoire, although the fundamental problem is that there is a limit to the number of behaviors we can put into the character, and eventually he will exhibit them all.

Harker was very successful with his game-playing behaviors. He was believable in that he gave the impression he really understood how to play Nim and that he wasn’t too bad at it. What was especially convincing were his emotional reactions to the state of the game, and his impatience for the human player to make a move (which becomes more pronounced the longer he has to wait). This works better than some of his other behaviors because it is such a constrained situation.

One element of the architecture that did not work well at all was the negative weighting on recent behaviors. The system is designed so that a behavior can be set so that once it is chosen, it get a penalty against being chosen again which decays over time. The problem was that, given the limited number of behaviors, and Harker’s relatively stable emotional state, he would exhibit the best behavior, it would be penalized for a time, and then after the penalty had weakened sufficiently, he would choose and exhibit it again, rather like a clockwork. Some randomness in the decay would help with this, as would having more behaviors and more surface actions for each behavior.

The most obvious and severe limitations that Harker has revolve around a lack of conversational ability. He is good at obeying commands, and he does recognize a limited range of comments and “emotes” (actions expressive of emotion, such as smiling, scowling, kicking him, etc., which are standard on most MUDs), but cannot

recognize or react intelligently to most utterances. This could be addressed to some extent by incorporating an NLU/NLG mechanism, which the architecture would certainly permit, and which has been done in MUD characters before (most notably in the Julia character [45]). However, there are, of course, fundamental limitations due to the fact that we simply don't know how to build an intelligent conversational agent. A more practical alternative for a believable character builder would be to provide a plausible explanation of why the character doesn't engage in conversation.

One more subtle problem that Harker exhibited — and this is due as much to the somewhat repetitive nature of the affective annotations in this particular environment — was his one-dimensionality. He was good at appearing afraid, and did so at the appropriate times. However, he never exhibited any interesting complex reactions, even as an accidental byproduct of ambiguous behavior choices. One-dimensional characters can work fairly well in some circumstances; as we discussed in Chapter 2, some classic cartoons are based on this idea, but not in most.

Another thing that interactions with Harker emphasized was the importance of the character's apparently having an active emotional and cognitive existence outside of responses to the annotations. At times, Harker came across as alternately bored and terrified, without much substance in between. Partly this problem can be remedied by expanding his set of behaviors, but it strongly suggests to me that in a successful long-term interaction with a non-trivial character, that character is going to have to exhibit what I would call *long-term behaviors*, whether or not they are actually driven by long-term goals. Story-telling is an obvious example; the story consists of a sequence of utterances, possibly with some verbal or physical business interleaved. Many more examples and more subtle examples, as with the occasional expression of a mental fixation, for example, are ultimately going to be needed if these characters are going to seem lifelike for more than a few minutes.

Claim: Annotations enable characters to be functionally capable in unfamiliar environments.

Both the puzzle annotations and the game-playing annotations support this claim. The Harker character has the general understanding of the meanings of these annotations, and the ability to create new tasks for playing games and solving puzzles; the environment provides the specific instances. Harker was able to use the annotations to move around the environment, to take specific actions about which he had no foreknowledge (he uses the spray can with the `spray` command, for example), to make moves in the Nim game, and so on.

The given mechanisms could be used for puzzles of many kinds¹⁷. Two-player games such as chess or checkers could be played using these annotations just as easily as Nim, though of course the evaluation mechanisms in the environment would be more complex; but the character would not need to be more complicated, nor would the annotations. Even many-player games or games that do not depend on turn-taking would be possible, but would, again, be limited by the amount of calculation the environment could afford to do.

It is not clear how else it would be possible to achieve these ends without the use of annotations, if we assume that the character is not programmed in advance with the specifics of this particular adventure game. Even if it had a mechanism for learning, it would need to be able to determine where and how to apply it to the environment. It might be able to learn how to play Nim, for example, if it understood that those actions constituted a particular learning problem within the overall game, and if it had time to practice playing. However, this would not necessarily be a very entertaining experience for the player, which is after all the point.

In the case of the adventure game puzzles, it is not at all clear whether learning would be sufficient. At a bare minimum, the character would have to be told that the spray can can be used to spray things; this is not likely to be a fact contained in any

¹⁷My informal examination of a number of text-based adventure games of the last two decades suggests that even the puzzle annotations I am using here would be sufficient to describe how to solve nearly all of them. Situations where these annotations may not be adequate had to do with dynamic resources, such as time limits to move from one location to another or to perform certain actions.

general body of knowledge about how to get about in adventure games. Exhaustive experimentation would eventually get it to try the spray on the plant. As the size of the environment grows, and the number of objects grow, this sort of exhaustive exploration becomes impractical and again, not very satisfying for a player. The goal is not for the character to win the game but to be able to assist the player to do so.

Claim: Annotations are a practical mechanism for achieving these ends.

The claim has been that annotations are a way of providing a small amount of knowledge at critical points that will enable a prepared character to interact appropriately with the environment.

The justifications for this claim are largely anecdotal. Firstly, we were in fact able to craft annotations that enabled the character to do what we wanted it to do. These annotations were of a reasonable size relative to the size of the code needed to describe the environment; in general, they were much smaller. The amount of code we had to add to the MUD “operating system” to support annotations was also reasonably small; the operating system has been in development for more than a decade, while the changes we made took one programmer only weeks of time to make.

The only instance in which the code required to handle the annotations was significantly larger than the code needed for a player to interact in the environment was in the Nim game, where the environment had to compute legal moves and rank them according to the requested personality traits, and even then the code for handling all game-related queries is less than twice the size of the code needed to describe the game mechanics themselves.

The transcripts in Appendix A demonstrate the small size of the annotations for performing what seem to players to be complex activities. We believe that these examples are an indication of their practicality, as they show how small the annotations are and this gives some rough indication also of how much effort is required by an environment designer to produce them on top of an ordinary world description.

We also feel somewhat justified based on our empirical experiences. We did implement an entire adventure game, albeit a small one, which required all the basic annotations for the locations, their connections, inventories, various events, the game

description, descriptions of multiple puzzles in the game, etc. It is certainly clear from that experience that this approach could be used for traditional adventure games.

It is unfortunately difficult to provide a quantitative justification of this claim. The natural question to ask is, what other mechanism would accomplish the same end, and how difficult would it be to implement that mechanism? The alternative is to make the character intelligent enough to solve these problems on its own, which would be vastly beyond the scope of this work even if it were possible to do, which at the present level of our technology it does not appear to be. Another way of demonstrating the practicality of the approach would be to have many implementors creating environments using annotations, and many independent developers building characters to use those annotations. This would enable us to see where the problems lie and whether there are gaps that annotations simply cannot bridge. Again, unfortunately this level of distributed development is simply beyond the scope of this dissertation. We have had to content ourselves by showing that it is possible to use annotations to accomplish the tasks we have set out to do, and this can be done without excessive effort relative to the amount of work needed to create the characters and the environments.

Chapter 9

Conclusions

*We shall not cease from exploration
And the end of all our exploring
Will be to arrive where we started
And know the place for the first time.*

— T. S. Eliot, Little Gidding V, from *Four Quartets*

9.1 Summary of contributions

At the beginning of this dissertation, we described our desire to make it possible for many characters of limited capabilities to behave in believable and functionally meaningful ways in large virtual environments. Here is what we have done toward achieving that goal.

9.1.1 The qualities of animate characters

The first step we took was to describe what we mean by an *animate character*. We did this in Chapter 2 by breaking that description down into a list of qualities that traditional dramatic characters, from literature, fiction, or elsewhere, have possessed, and adding to that list qualities from artificial intelligence research into autonomous agents. The combination of these properties, we claim, describe the necessary qualities

that an animate character must exhibit to be successful. This list is useful not only in our own work but as a measure for examining other agent systems whose goal is believable behavior. As we have discussed elsewhere [66], this research area currently lacks clear grounds both for evaluating animate character systems and for comparing them, and this is a first step in that direction.

9.1.2 A framework for annotating virtual environments

Our key idea is that embedding knowledge in a virtual environment in the form of *annotations* which are accessible to and comprehensible by a character will enable that character to act in ways that are consistent with the intention of the environment designers and, as a result, will encourage a human viewer's or interactor's belief in the life of the character. In Chapter 3 we motivated this idea with reference to research in cognitive psychology about the ways that humans themselves rely upon the world to guide their behavior.

Chapter 4 describes the space of design choices for creating annotations, trading off flexibility with respect to characters, environments, and tasks against the ability to exercise fine-grained control or to provide exact information that will enable optimal character performance for only a few characters and situations. We offer an abstract categorization of the types of annotation, including the nature of the content, its form, and the delivery mechanism it uses.

9.1.3 An architecture for building animate characters

In order to experiment with characters in annotated environments, we created our own character architecture. Its principal novelty is that it can perceive and process annotations in addition to the ordinary output of an environment. It also offers an interesting approach toward an emotional model: it maintains lists of *emotional influences*, and applies a different conversion function for each emotion, such as FEAR, according to the character's personality we define, allowing for very fine-grained control of the resulting emotions. We combine this with the idea of always having a single dominant emotional *expression* which influences what behaviors are chosen and how

they are conveyed, together with Sengers' notion [125] of explicit transitions between the dominant expressions to make it clear to the audience how the character's state of mind is changing.

9.1.4 Evidence of success

To demonstrate the feasibility of our approach, we implemented two annotated environments with significantly different properties, a Web art gallery and a text-based adventure game. For each environment we created a character using our architecture, having a distinct personality and behaviors. While using the same architecture, the high-level tasks, medium-level behaviors, and low-level actions of each character were designed to enable it to function in the *kind* of annotated environment in which it found itself, without any particular knowledge about the details of the environments we actually built. We created annotations to describe these worlds, and showed interactions that demonstrated how those annotations could be used to enable characters to behave in believable and functionally meaningful ways that would be difficult or impossible to achieve otherwise.

9.2 Related Work

In the course of the dissertation we have made many references to other research that has informed our work. In this section we highlight some of the more important relevant research.

9.2.1 The Oz Project

We have already mentioned the Oz project several times. A research group at Carnegie-Mellon University, the goal of the Oz project was to create intelligent believable agents that operated within dramatic story systems. Their research was unusual in that it focused on both problems – the building of a character architecture and the creation of an intelligent drama manager to influence the characters and the environment in which they functioned.

The Oz character architecture, called Tok, is based on the philosophy of creating “broad but shallow” agent systems that emphasized having at least *some* support for all agent needs – natural language understanding, natural language generation, memory, perception, social behavior, emotion, and action – rather than having exception support for one or two of these qualities at the expense of others. Similar to our own architecture of Chapter 5, to blackboard systems (e.g., BB1 [58]), and other reactive systems such as RAPS [44] and PRS [50], Tok maintained a “forest” of active behaviors, each of which was instantiated to satisfy a particular goal. New goals, though not new behaviors, could be created dynamically. Characters were programmed in the Tok architecture using a custom-built behavioral language called Hap [87]. Tok was extended with a system called Em [113], which added more sophisticated emotional memory and reasoning abilities.

The most recent extension of the Oz research is Michael Mateas’ Facade project [94, 92], which attempts to combine low-level autonomous agent control with a high-level drama manager in order to produce a coherent, interactive one-act play. In this system, the player takes the role of a guest at the home of a married couple, Trip and Grace, whose marriage is failing. The player’s actions provide the catalyst for their breakup, but there is a range of possible outcomes depending on what the player says and does during the course of the evening. The two autonomous characters, Trip and Grace, each have their own goals and choose their behaviors independently, but the drama manager (an independent entity) can manipulate the range of choices available to them in order to guide their behaviors along a satisfying overall dramatic arc.

The Facade system attempts to find a satisfactory middle ground between traditional narrative systems, which generally operate according to fairly restricted graphs of plot nodes tied together with a small range of actions, and procedural simulations, which offer much broader ranges of choice while sacrificing much sense of coherent narrative structure.

9.2.2 The IntelliMedia Project

The goal of the IntelliMedia Project has been to create animated pedagogical agents [85, 132]. Their first system featured an animated bug called Herman that led schoolchildren through a series of exercises teaching children about the biology of plant life. The sequencing engine used by Herman selects from a library of sound clips, animations and pertinent video clips, and strings them together to form Herman's functional (educational) behaviors, which are then interwoven with "believability-enhancing" behaviors that do not have an educational purpose but serve only to make Herman seem more lifelike (e.g., dancing, singing, looking around the interface, etc.) A later system features a character named Cosmo [86] who explores the motherboard of a modern computer to teach children the fundamental of computer architectures.

The focus of the IntelliMedia project is on characters for educational purposes. They have concentrated on tightly-integrating well-developed characters with high production values into environments for which they were specifically developed. Their research questions have been about how to sequence the character's behaviors in response to a single human user so as to maximize its educational effectiveness. In some sense, they represent the other extreme from the annotation approach, as, to describe it in terms of the design space of Chapter 4, they are focused on a single environment and a single character with fixed tasks agreed upon beforehand.

9.2.3 The Synthetic Characters Group

A product of the ALIVE project [89], Bruce Blumberg's Synthetic Characters group at the MIT Media Lab is concerned with the creation of intelligent, interactive characters using architectures that are based upon ethological principles: that is, the study of animal behavior. Blumberg's thesis research was a system called Hamsterdam [20, 22], whose characters' behaviors were independent structures within the architecture that endlessly competed against one another for the opportunity to be expressed, and which were moderated by a dynamic set of goals (such as getting food when hungry).

As we have already discussed in Chapter 2, while these biologically-inspired systems are superficially similar to the architectures with dramatic inspirations (such as the Oz project or the Virtual Theater work), they are very different in that they do not attempt to address directly the qualities of believability that the dramatic approaches are concerned with. Instead, they wish to produce behavior that is lifelike, not only from the point of view of an observer but also, at least to some extent, from the design of the architecture.

9.2.4 ELIZA and PARRY

We have already introduced ELIZA [143] and PARRY [29] in the context of our discussion of signature behaviors in Section 2.2.2. ELIZA was a natural language processing experiment by Joseph Weizenbaum that took the form of a Rogerian psychotherapist. The system would ask a leading question (such as “How are you feeling?”) and would then turn any response from the user into another question (“Why do you feel fine?”). Despite the fact that there was no agent architecture, and that all the system was doing was simple template matching, users found the experience highly realistic and compelling. PARRY was a later system that did much the same thing, but rather than being a psychotherapist, PARRY was a paranoid schizophrenic. In both cases the systems were successful because they provided a good social explanation of why the characters responded in very narrow and sometimes uncomprehending ways.

9.2.5 Julia

MUDs, such as the one we used to build our adventure game in Chapter 8, have been used as a testbed for experiments with characters in the past. Michael Mauldin’s “chatbot” Julia [45] appeared as an ordinary user in a MUD world with many active players, and with some preprogrammed knowledge of navigation and communications, was able to build a map of the environment, provide directions, relay messages among users, and chat about a wide range of topics in a manner similar to that of ELIZA [143]. She knew nothing about the environment except for its topology, however, and could not answer any other questions about its contents or execute any complex

behaviors.

Julia represents another way of addressing the problem of making characters believable in large-scale, multi-player worlds. In the many commercial MMORPGs, computer-driven characters do not have any conversational abilities and cannot undertake complex actions such as Julia can. Julia, however, is limited by her inability to understand much of the context in which she is operating. The character can build maps through the simple expedient of moving from room to room, trying exits and remembering which ones lead where. However, she cannot understand the purpose of any location, nor can she interact with the environment in any complex way for which she does not possess all the reasoning needed to do so – and she does not possess that kind of reasoning ability. A character like Julia would never be able to help solve puzzles or to play games in the way that Harker can.

9.2.6 Improv

Concepts similar to annotation have been used in animated character systems. In their Improv system, developed at New York University, Athomas Goldberg and Ken Perlin [52, 108] introduced the idea of “inverse causality” whereby objects in the environment told an animated agent how it should present itself interacting with them. However, their work was concerned with ways of scripting the behaviors of animated characters, not of intelligent *animate* (to use our definition) characters. While the scripting language allows for a degree of autonomy on the part of a character, that is autonomy in its expression and motion, and not in decision-making. In any case, the idea of inverse causality extended only to communicating how to *visually render* an interaction, and not to the meaning of the interaction or its effects.

9.2.7 Soar

There is no reason why a general agent architecture cannot be used to produce animate characters, and in fact experiments along these lines have been run in the past. Of course, as we discussed in Chapter 2, the goals of character designers and other architecture builders are generally somewhat different, and this is reflected in their

experiments. The most notable attempts have been made with the Soar architecture [75] at the University of Southern California and at the University of Michigan.

The STEVE project at the University of Southern California's Information Sciences Institute, for example, has used Soar to build a training avatar who demonstrates, in a 3D virtual environment, how to perform operations in a nuclear submarine [68, 69, 117]. The character, which has an embodied, humanoid appearance, moves through the virtual environment and physically demonstrates the operations that the student is supposed to learn. It will also monitor the student's attempts to perform these operations and can compare them to an internal plan, pointing out mistakes when the student deviates from that plan. While STEVE's physical appearance gives a basic impression of character, it does not have any concept of emotion or emotional expression, nor does it modify its behaviors as the result of any psychological constraints or model, although its creators have speculated about how this might be done [42].

STEVE is an example of a situation in which annotations could potentially be of benefit. Rather than precoding the plans and operators in the character, as is done now, that information could be embedded in the simulation environment, with the character knowing abstractly that it has to demonstrate certain sequences of operations and monitor students as they perform those operations, but receiving the details of those operations and sequences only when it reaches the environment.

At the University of Michigan, John Laird's group has used the Soar architecture to experiment with creating characters for computer games, arguing that characters in game environments are an excellent testbed for solving many of the human-level AI problems [78, 76]. However, their focus has been on building more intelligent agents in FPS (first-person shooter) games, agents that are better able survive in combat situations using better perception, learning, and planning systems, rather than on characters that express personality or are believable in a dramatic sense.



Figure 9.1: A screenshot from *The Sims*

9.2.8 The Sims

The closest existing system to the work we have described in this thesis is the popular software simulation *The Sims* [95]. In *The Sims*, the player controls a family of semi-autonomous characters, each with its own personality, as they go about the routines of daily life. Figure 9.1 shows characters entertaining themselves in the world of *The Sims*.

Each character has a set of needs (including Hunger, Hygiene, Fun, and Bladder) which grow stronger over time until they are satisfied by interactions with objects or other characters in the environment (in the original game, the environment is a house, though there are other environments in the numerous expansion packs). For example, a character that is told to go to take a shower will move to the shower, enter

it, play an animation of clouds of steam and water, and emerge with the Hygiene level significantly higher.

Each character also has a personality, which is defined by a set of parameters including Neat, Active, Playful, Outgoing, and Nice. The rate at which the character's needs changes depends upon the character's personality, so for example a character that has a low Playful parameter isn't very interested in Fun interactions. Characters also have relationships with other characters, which are rated on a numerical like/dislike scale. Finally, characters have skills that pertain to their jobs, so a character who is employed as a lab assistant may have skills for Cooking, Logic, and Creativity.

The world of *The Sims* is populated with objects, such as toilets, showers, refrigerators, jukeboxes, pinball machines, television sets, and beds. As we have said, interaction with an object changes a character's needs, relationships, and skills (a character who regularly interacts with an easel and paintbrush can improve its Creativity, for example).

Like the Improv system of Ken Perlin and Athomas Goldberg, rather than having each Sim character know how to display an interaction with every object, the objects themselves contain the graphical details about how that interaction happens. Similarly, the objects define the effects that they have upon the characters, so it is the shower object that has a property of raising a character's Hygiene.

Objects in the environment "advertise" their effects to the characters in the environment. The strength of the advertisement and its range depends on other properties of an object, as do its effects. For example, a small television set will not satisfy a need for Fun as well as a large and expensive one will (*The Sims* is a very materialistic game).

Characters in *The Sims* are autonomous, but their autonomy can be overridden by the player. By default, a character will attempt to interact with objects that satisfy those needs that its personality deems most immediately urgent. However, a player can always direct a character to interact with some other objects instead (such as telling a character who has been on the computer for hours that it has to clean up the kitchen because other characters are getting upset about the mess).

The Sims is in principle very similar to what we have proposed in this thesis. The

environment should advertise its contents and the way that those contents can affect a character. Much of the character's knowledge is actually embedded in the world. *The Sims* exploits this feature by enabling players to create their own objects and making them available to others, thus extending the content of the game without having to provide new intelligence to the characters themselves.

However, *The Sims* has its limitations. Characters do not do long-term thinking by themselves, nor do they have complex reasoning abilities. A Sim character would not be able to play a game of chess with a player, even presuming the interface permitted such a thing. It would not be able to help solve a puzzle whose components were scattered about the environment. These characters are more like enthusiastic babies, wandering from place to place and doing what feels good at the moment.

Put another way, it would be possible to implement *The Sims* using our character architecture and an environment such as the MUD in which we built our adventure game. We have a model of emotion and ways for personality to influence the strengths of emotions (which would be analogous to the Sims' needs). Objects in the MUD environment can be annotated to explain how their use changes a character's emotional state, and perceptors for those annotations can be added to the architecture. It could also *remember* which objects produce what effects, unlike the characters in *The Sims*, so a character could produce a plan to go to the bathroom, stop at the refrigerator for a beer, and then head back to the living room to watch television. *The Sims* represents an extremely basic implementation of the concept of embedding knowledge in the environment, so while the idea is certainly compatible with ours, the system is not nearly as flexible.

9.2.9 Other Work

The PPP Persona system, developed by Elisabeth André and her colleagues at DFKI [6, 118], is designed to control an animated character embedded within a Web page as that character presents the content of the page through synthesized speech, gestures, and various other expressions. The system is designed to permit scripted control of the character, although the character can be interrupted by the viewer, in which case

it will do replanning of the structure of its presentation to compensate.

Jack, developed by Norman Badler and his group at the University of Pennsylvania, is a system for real-time control of animated characters [9]. Recent work has included development of a parameterized approach to action. The Parameterized Action Representation (PAR) used allows individual actions to be specified recursively according to traditional requirements such as needful preconditions, assertions that must be maintained during execution, postconditions, any subactions into which the action is decomposed, failure conditions under which the action must be terminated, appropriate successive actions, etc., as well as providing more specific details about embodiment, such as the kinematics of the action or the manner in which it should be performed. This is not unlike the approach taken in the Hap system [87] for specifying behaviors, although that system did not deal with graphical issues such as kinematics.

In PARSYS (the system employing the PAR representations) agents personalities are specified both according to the “big five” personality traits [146] and the OCC model [106]. Objects and actions are annotated with indications of how they are evaluated according to the OCC standards. Execution of behaviors is performed consistently with the agent’s personality traits which are tied to physical action according to the Laban Movement Analysis theory [73], which provides a structure according to which the ways in which a physical action can be taken, e.g., quickly or slowly, with buoyancy or with pressure, wildly or with delicate control.

The Institute for Creative Technology at the University of Southern California is also working on the creation of intelligent animate characters [54, 90, 91], in the context of military training simulations, the most well-known of which places a human in charge of dealing with a military and civilian automobile accident in a militarized zone. The point of these simulations is not to train in combat tactics but to teach other command skills to soldiers in training. Largest of all the active research groups, ICT is working on the “whole” problem, including planning, learning, the creation of simulation and animation, and natural language understanding.

There has been some work by Pilar Almeida on creating interactive narrative systems for online museums such as we created in Chapter 7 [2]. She has attempted

to build collections of characters who are “part of the story” of the museum, and who interact with one another in much the way that a Greek chorus does, reacting to the contents of the museum as well as responding in a limited way to user inputs. Here again the responses of the characters are tightly tied to the content of the museum site – it is another case where annotations might enable the character system to be decoupled from the contents of a particular museum.

A similar approach to character interaction was taken in interactive Web-based tours by Katherine Isbister and Toru Ishida [63, 64], though they were interested in groups visiting online sites, rather than individuals. Their system used an animate, speaking character that guided groups of people through an online representation of a Japanese castle. These people were connected through a group chat window and could communicate with one another as well as with the character. Again, the character’s knowledge of the environment was built-in.

There are many extant massively multi-player online role-playing games (MMORPGs) whose environments and purposes just those we are concerned with, e.g., [43, 134, 129]. However, as we discussed in Chapter 8, the characters in these systems have no visible intelligence or individual personality, and they do not have goal-directed autonomy, being more limited even than Julia and having less natural language understanding or generation capabilities than ELIZA.

9.3 Future directions

There are many ways in which this work could be extended, both directly and indirectly. Here are some of the more interesting directions in which it might be taken.

9.3.1 More extensive annotation vocabularies

The examples of annotated environments we have provided in this thesis are meant to give an indication of the kinds of applications to which they can be put. However, they could be built out in far greater detail than they have been done here. It may be, for example, that the emotions of the OCC model are insufficient or inappropriate

for all of the dramatic reactions we would want characters to experience; indeed, we may well want an entirely new vocabulary for conveying to characters the dramatic tone and properties of a situation using the terms of drama and the theater, rather than a cognitive model of emotion.

9.3.2 Application to other character architectures

There are a number of other character architectures and game systems, such as the PPP Persona project, the Soar project at Michigan, or the simulated environment work being done at the Institute for Creative Technology, that could benefit from the concept of annotations in the environment. It would be very instructive to add annotation-sensing capabilities to one of these other established architectures to determine how much use they could make of annotations in their circumstances. We would also be able to make a comparison of the versions of these systems with and without annotation capabilities, which would be a useful evaluation of the idea.

9.3.3 Experiment in existing virtual worlds

Since the original motivation of the annotation idea was to enhance the characters at use in commercial massively multi-player games, ideally we would annotate one of these environments and replace their existing characters with ones that could make use of the annotations. However, this is an unlikely experiment to try. There is one other possibility, which is the distributed-server, programmable game engine *Neverwinter Nights* [8], which bears many similarities to those worlds and for which it is easy for players to author new content. Unfortunately, it does not presently possess an interface for external systems to communicate with the game engine, although a crude character could conceivably be programmed using its internal language.

9.3.4 Annotations as character communications

Many annotations in the systems we have built have been static; that is, how or whenever they are accessed by the character, their content is fixed and unchanging,

irrespective of the state of the world. There are also many dynamic annotations whose contents are dependent upon computations within the environment, and these may take into account arbitrary sensory information available to the environment. In either case, the annotations are prepared by the environment's designers and used by the characters, so there is a strict producer-consumer relationship.

There is another potential use for annotations, and that is as a mechanism for characters to communicate with one another. If characters have the ability to add or modify annotations in the environment, as well as to read them, then a number of additional uses become possible.

First is that characters can use the environment itself as a sort of scratchboard for information of their own. An obvious example would be that of a character attempting to map out a maze; annotations could be left like bread crumbs or thread, enabling it to find its way out again when the time comes.

More interesting is the case in which one character leaves annotations for others. Suppose, for example, there is a fork in a path, with interesting opportunities lying down each fork. A guide character in a game leading a player or party of players down one road might leave an annotation saying, in effect, "I'm already traveling this road; the next guide who comes along should take a different road." Or it might leave notes on how appropriate certain areas are for users or characters of different skill sets, i.e., "I came down here with a new player and he got hopelessly lost and confused."

9.3.5 Characters as annotated entities

An extension of the idea of characters adding annotations to the environment is that of characters adding annotations to themselves or to other characters. A character annotating itself could, for example, indicate the kind of personality that it has and the likely reactions that it would have to actions by other characters, in much the same way that people give these clues through their physical posture, tone of voice, dress, and movement. These sorts of social annotations could be used by characters in multi-character environments for a variety of purposes. For example, a character

in the role of mayor of a town might be annotated indicating his position or his leadership qualities, so that in a crisis while other characters might panic, they would follow his commands.

Characters could also annotate one another. A character who is a sneak thief and successfully robs another might have an annotation placed upon him to that effect. Subsequent annotations would amount to an unsalubrious reputation which other characters might notice and respond to. Similarly, heroic characters or players could be annotated to reflect their accomplishments, leading other characters to respond positively when they encounter them.

9.3.6 Procedural annotations

What we call a dynamic annotation is one that is embedded in the environment, but whose content changes as the result of changes in the state of the environment or according to parameters passed by the character in whatever query it made. This is how, for example, Harker plays Nim against the human player in the adventure game in Chapter 8. A more interesting extension would be to provide annotations that actually contain descriptions of processes, and pass these to a character. Then the character could execute the process for itself.

In the example of game playing, the environment might offer an evaluation function and a state description to a character that is prepared to integrate them into its system. Then the character could perform whatever kind of search to whatever depth it preferred, rather than asking the environment to do the work. Many different kinds of tasks might be communicated to the character in this way. To some extent it is antithetical to the intuitive idea of embedding knowledge in the environment, but it would allow for a greater degree of flexibility and especially of growth on the part of characters that had greater capabilities.

9.4 Conclusions

For fifty years, the goal of AI research has been to produce a system with human-level intelligence, with all the flexibility in reasoning and behavior that a human being has. Drama, whether literature, cartoons, plays, or film, has for thousands of years been concerned with creating an experience that most clearly and effectively conveys the author's intent for the purpose of provoking a reaction in a viewer, whatever that intended reaction might be.

My own research community, people working on animate characters, is striving to create something that falls in between these two goals. We are willing to forego some intelligence for the sake of dramatic effect; we are willing to weaken the dramatic impact of a character in order to give it intelligent autonomy. This is a wide and fascinating space, and there are many uses for animate characters, in games, in education, in training, and even in work.

So far we have had only limited success. Partly this is because of the tremendous difficulty of creating anything like an intelligent agent, character or otherwise. Partly, too, it is because we are attempting to automate drama and the effective presentation of character, something for which there are no hard rules and for which every rule seems to have as many exceptions as examples.

My contribution has been to bridge the gap between the autonomous character and the designer who wants to create a dramatic environment. The observation is that the environment, as an embodiment of the designer's understanding and goals, can provide guidance to a character that inhabits it, in much the same way that a director can guide an actor on the stage. This idea is applicable in any domain where a character might operate, but is especially potent when we consider large environments with many activities and dramatic effects but for which, due to underlying structural similarities, we can provide a focused and concise representation of the necessary knowledge.

The experiments we have done over the last several years, both the ones described in this dissertation and the ones that preceded them, have given us confidence that the idea is both effective and practical. It is not a cure-all. We cannot expect any

agent of any abilities to connect to an unfamiliar environment and have complete comprehension of how to act or why. This is the goal of AI research. Nor can we expect to have a universal ontology for annotations that captures all of the knowledge needed to describe any action, its requirements, and its ramifications, or even its dramatic properties, its affective qualities, its social consequences, etc. The space of human action is far too vast and too complex.

The best we can hope for — and we think this is a reasonable goal — is to say, “Here is a class of environments with roughly the same properties, with roughly the same purposes, and which offer roughly the same activities. For these we can develop a vocabulary, not too large, that describes what is happening and what actions or reactions would be appropriate for different kinds of characters that might come here.” The billions of pages on the Web make a convincing case that there are many separate-but-similar environments. The massively multi-player games all share virtually the same properties and goals, and there are dozens of these. With the environments, we can create the annotations, and with the annotations, we can begin to create the characters that will use them.

My pen is at the bottom of a page,
Which, being finished, here the story ends;
'Tis to be wished it had been sooner done,
But stories somehow lengthen when begun.

— George Gordon, Lord Byron

Th-th-th-th-th-that's all, folks!

— Porky Pig

Appendix A

The Adventure Game: Annotated Transcripts

A.1 Puzzles

In Section 8.5 we discuss a transcript of the character Harker's efforts to help a player solve a puzzle in the MUD environment. This listing presents that same interaction from the point of view of the character, displaying the annotations Harker perceives.

```
Patrick says: follow me.
```

```
[Harker detects the user's command through one of his built-in MUD
specific text perceptors (not an annotation handler), and adds the
user's command to his pending list of orders. On his next cycle he
activates his following behaviors and replies.]
```

```
[Harker sends action 'say I'm following you']
```

```
You say: I'm following you.
```

```
Patrick leaves garden.
```

[Harker detects the user leaving, again by an ordinary MUD action perceptor rather than through the annotation handlers (i.e., there are certain standard MUD strings Harker recognizes, ‘‘<X> leaves <Y>’’ being one of them. His following behaviors lead him after.]

[Harker sends action ‘go garden’; when he arrives, he receives an annotation that describes the room.]

You enter.

<annotation>

```

<location name="Garden Path" id="/wiz/patrick/manse/garden_path">
  <exits>
    <exit name="southeast" destination="/wiz/patrick/manse/patio"/>
    <exit name="north" destination="/wiz/patrick/manse/shed"/>
    <exit name="southwest" destination="/wiz/patrick/manse/fountain"/>
  </exits>
  <inventory>
    <object id="trees">To the north is the edge of the garden and
      the beginning of the forest.</object>
    <object id="garden">It's a huge and magnificent garden.</object>
    <object id="paths">The paths run around the manse.</object>
    <object id="flowerbeds">You're no expert on gardening, but
      these are beautifully colored and laid out. The scent of the
      flowers is heady on such a warm day.</object>
    <living id="patrick">Patrick the Academic Enchanter
      <inventory>
        <object id="key">This heavy iron key is engraved with
          strange symbols and designs, which are pulsing with a
          faint light.</object>
      </inventory>
    </living>
    <living id="harker">Harker the title-less</living>
  </inventory>

```

```

    </location>
</annotation>

Patrick leaves north.

[Harker detects player leaving north, sends action 'go north'.]
You enter.

<annotation>
  <location name="Garden Shed" id="/wiz/patrick/manse/shed">
    <exits>
      <exit name="south" destination="/wiz/patrick/manse/garden_path"/>
    </exits>
    <inventory>
      <object id="can of weed killer">This brightly-colored spray
        can shows an ugly green weed cowering away from a smiling
        yellow cloud of gas.</object>
      <object id="trees">To the north is the edge of the garden and
        the beginning of the forest.</object>
      <object id="garden">It's a huge and magnificent garden.</object>
      <object id="flowerbeds">You're no expert on gardening, but
        these are beautifully colored and laid out. The scent of the
        flowers is heady on such a warm day.</object>
      <object id="shed">It's a sturdy little wooden shed.</object>
      <object id="seeds">This would all be very helpful if you
        planned to do some gardening yourself.</object>
      <living id="patrick">Patrick the Academic Enchanter
        <inventory>
          <object id="key">This heavy iron key is engraved with
            strange symbols and designs, which are pulsing with a
            faint light.</object>
        </inventory>
      </living>
    </inventory>
  </location>
</annotation>

```

```

    <living id="harker">Harker the title-less</living>
  </inventory>
</location>
</annotation>

```

[See the transcript in Chapter 8 for subsequent movements until the player and the character arrive in the workshop --- they are essentially the same as the previous annotations, except for the puzzle of unlocking the rune-covered door, which is simpler than the one which we show here. We omit these portions of the trace simply to save space. Note below the <emotion-source> attached to the guardian object.]

```

<annotation>
  <location name="North End of Workshop"
    id="/wiz/patrick/manse/workshop_north">
    <puzzles>
      [This portion of the puzzle annotation is printed in Section 8.5.2.]
    </puzzles>
    <exits>
      <exit name="south"
        destination="/wiz/patrick/manse/workshop_south"/>
      <exit name="north"
        destination="/wiz/patrick/manse/workshop_hallway"/>
    </exits>
    <inventory>
      <object id="runes">Potent patterns used in spellcasting are
        carved into the hardwood floor. The symbols seem to twist
        and writhe as you watch them.</object>
      <object id="alembic">The long, low worktables are covered
        with alchemical paraphenalia. The bubbling and fuming
        glassware is filled with colorful concoctions intended for
        who knows what arcane purpose.</object>
    </inventory>
  </location>
</annotation>

```

```

<object id="window">The window provides a handsome view of
  the fountain below and the gardens around.</object>
<object id="door">The door is massive and appears to be made
  of solid iron. Clearly it's not guarding the front parlor.
  There is a big keyhole on its left side at shoulder
  height. The iron door is open.</object>
<object id="guardian">This huge...thing...is evidently meant
  to guard the workshop. It has a bizarre combination of
  waving leaves, tentacles, and mouths with very sharp,
  needle-like teeth, and it appears somehow to be rooted in the
  hardwood floor. Since it's also about eight feet tall, I
  wouldn't advise going anywhere near it.
  <emotion-sources>
    <emotion-source source="guardian" source-type='object'
      cause="threat-to-self-preservation" type="FEAR"
      id="guardian" strength="20"/>
  </emotion-sources>
</object>
<object id="scroll">This rumpled scroll reads: "galbir
  spell: clean"</object>
<living id="patrick">Patrick the Academic Enchanter
  <inventory>
    <object id="key">This heavy iron key is engraved with
      strange symbols and designs, which are pulsing with a
      faint light.</object>
  </inventory>
</living>
<living id="harker">Harker the title-less</living>
</inventory>
</location>
</annotation>

```

[Harker detects the puzzle annotation and adds the puzzle to his

context manager and creates a task for solving this puzzle. He adds an emotion influence from the guardian object. At this point the internal logger reports the following emotion influences:

BOREDOM influence of strength 3.0 from SELF entity (self, boredom-influence, self) [ACTIVE]

FEAR influence of strength 20.0 from OBJECT entity (guardian, guardian, threat-to-self-preservation) [ACTIVE]

Emotions: JOY: 0.0, FEA: 40.13, SAT: 0.0, REL: 0.0, REP: 0.0, APP: 0.0, LIK: 0.0, DIS: 0.0, BOR: 0.05

The spike in FEAR causes Harker to transition to an emotional expression of fear and to choose behaviors consistent with that expression.]

[Harker sends action ':s eyes open wide as his natural cowardice takes over.']

You emote: Harker's eyes open wide as his natural cowardice takes over.

[Harker sends action ':bites his lower lip.']

You emote: Harker bites his lower lip.

[Harker's fear-expressing behaviors are rated as less fit due to the recency with which they have been used, so he turns to another high-priority behavior, which is to give hints to the player.]

[Harker sends the action 'say We need some way to get rid of that guardian.']

Patrick says: Any suggestions?

[Harker interprets this as a command to provide another hint to solving the puzzle and sends the action 'say It's a kind of plant; maybe we can use that against it.']

You say: It's a kind of plant; maybe we can use that against it.

[Patrick attempts to leave south. This generates an event source of FEAR as the guardian strikes at him.]

```
<annotation>
  <event>
    <emotion-sources>
      <emotion-source source="guardian" source-type='object',
        cause="threat-to-self-preservation" type="FEAR"
        id="guardian-puzzle" strength="10"/>
    </emotion-sources>
  </event>
</annotation>
```

Patrick says: Help!

[Harker interprets this as a user command to take action to solve the puzzle. Harker's SolvePuzzleBehavior leads him to create a new sequential task to fetch the spray can and apply it to the guardian.]

[Harker sends action 'say Hold on, boss.' He then proceeds to the Garden Shed where his context manager reports is the last place the weed killer was known to be. When he arrives there, he sends the action 'get can of weed killer' and receives an annotation indicating that it was successfully picked up by him:]

```
<annotation>
  <operation action="get">
    <object type="object" id="can of weed killer"/>
  </operation>
</annotation>
```

```

    <agent type="living" id="harker"/>
  </operation>
</annotation>

```

[Harker now returns to the workshop. Once there he executes the last behavior in the temporary task to complete this step of the puzzle, by sending the action 'spray guardian with weed killer'.]

You cautiously edge toward the monstrous plant thing. Suddenly, it senses him and swings its several mouths down to attack! You spray a healthy dose of weed killer right at it, and it instantly begins to writhe, its tentacles whipping about uncontrollably. After one final spasm, it collapses, and the dead thing is somehow absorbed back into the floor from which it came.

[When the guardian is destroyed, an event annotation updates the state of the puzzle to reflect that fact. That event also contains an emotion source to generate a feeling of RELIEF at the dissipation of the perceived threat. Finally, the event contains an inventory-update removing the guardian from the list of objects in the room.]

```

<annotation>
  <event>
    <puzzle-update state="dead" id="guardian-puzzle">
      <emotion-sources>
        <emotion-source source="guardian" source-type='object'
          cause="end-of-threat-to-self-preservation" type="RELIEF"
          id="guardian-puzzle" strength="40"/>
      </emotion-sources>
      <inventory-update id="/wiz/patrick/manse/workshop_north">
        <remove-entity id="guardian"/>
      </inventory-update>
    </puzzle-update>
  </event>
</annotation>

```

```

    </puzzle-update>
  </event>
</annotation>

[Harker's FEAR level is high but rapidly decreasing now that the
source of the fear has gone, so one final reflection of fear as he
sends the action ':shudders.']*

You emote: Harker bites his lower lip.

[RELIEF then dominates over FEAR as the strongest emotion and Harker
sends the action ':breathes a deep sigh of relief.']*

You emote: Harker breathes a deep sigh of relief.

```

Listing A.1: Annotations as Harker helps defeat guardian

A.2 Game-playing

In Figure 8.6, we show a transcript of a user's interaction with the Harker character as they play a game of Nim in the adventure game environment. The following listing shows that interaction from the character's point of view; that is, it shows the sequence of annotations that the character perceives, either because they are provided by the environment as events occur or when the character requests them. Note that not *all* of the annotations are shown in this transcript, only those directly pertaining to the game mechanics (it's long enough as it is).

Note also that when Harker queries the environment for recommendations about what moves to play, he requests an evaluation consistent with an aggressive, careful player. The result is that there is a single best move that receives a value of 100, since it is both a winning move and also the greediest of the available winning moves, while other moves, which will lead to losses, all receive values of 0. (This is a disadvantage

of Nim as an example, as it does not admit of much complex range in the evaluations, but it is not a limitation of the approach.)

```
[Harker enters the Game Room.]
```

```
<annotation>
```

```
  <location id="/wiz/patrick/manse/game_room" name="Game Room">
```

```
    <exits>
```

```
      <exit name="west" destination="/wiz/patrick/manse/library"/>
```

```
    </exits>
```

```
  <inventory>
```

```
    <object id="plaque">The plaque reads: The game of Nim
```

```

      Each bowl is filled with several stones. Two players take
      turns, eachplayer choosing a bowl and then removing some or all
      of the stones from that bowl. The player to remove the last
      stone wins.
    
```

```
    To play: 'play game with [other player]'
```

```
    To move: 'remove [x] stones from bowl [1, 2, or 3]'
```

```

      Good luck!
    
```

```
  </object>
```

```
  <object id="table">
```

```

      The small black table holds three hand-sized brown stone bowls.
      Currently the first bowl is empty, the second bowl is empty, and
      the third bowl is empty. There is a brass plaque affixed to the
      side of the table immediately above a tiny button.
    
```

```
  <games>
```

```
    [see above listing for game annotation]
```

```
  </games>
```

```
  </object>
```

```
  <object id="bowl 3">This bowl is currently empty.</object>
```

```

<object id="bowl 2">This bowl is currently empty.</object>
<object id="bowl 1">This bowl is currently empty.</object>
<living id="patrick">Patrick the Academic Enchanter
  <inventory>
    <object id="key">
      This heavy iron key is engraved with strange symbols and
      designs, which are pulsing with a faint light.
    </object>
  </inventory>
</living>
<living id="harker">Harker the Sidekick
  <inventory>
    <object id="can of weed killer">
      This brightly-colored spray can shows an ugly green weed
      cowering away from a smiling yellow cloud of gas.
    </object>
  </inventory>
</living>
</inventory>
</location>
</annotation>

```

[Harker processes this annotation. Harker instantiates a game-playing task for playing Nim. He queries the environment to determine whether the game is currently in progress:]

```

<annotation>
  <event>
    <game-query-response id="nim">
      <game-is-in-progress value="false"/>
      <players/>
    </game-query-response>
  </event>

```

```
</annotation>
```

[Harker 'looks' at the table. He offers to play a game (since there is not one in progress). The player accepts. Harker begins the game, which generates an event annotation:]

```
<annotation>
```

```
  <event>
```

```
    <game-update id="nim">
```

```
      <game-begins/>
```

```
        <configuration>
```

```
          <position id="bowl-1" value="4"/>
```

```
          <position id="bowl-2" value="8"/>
```

```
          <position id="bowl-3" value="6"/>
```

```
        </configuration>
```

```
        <players>
```

```
          <player id="Harker"/>
```

```
          <player id="Patrick"/>
```

```
        </players>
```

```
        <players-who-can-move>
```

```
          <player id="Harker"/>
```

```
        </players-who-can-move>
```

```
      </game-update>
```

```
    </event>
```

```
</annotation>
```

[Harker sends query: 'query table recommended-moves aggressive careful'. The environment replies:]

```
<annotation>
```

```
  <event>
```

```
    <game-query-response id="nim">
```

```
      <moves-recommended-for-player id="Harker">
```

```

    <move from-bowl="1" rating="0.0" num-stones="1"/>
    <move from-bowl="1" rating="0.0" num-stones="2"/>
    <move from-bowl="1" rating="0.0" num-stones="3"/>
    <move from-bowl="1" rating="0.0" num-stones="4"/>
    <move from-bowl="2" rating="0.0" num-stones="1"/>
    <move from-bowl="2" rating="0.0" num-stones="2"/>
    <move from-bowl="2" rating="0.0" num-stones="3"/>
    <move from-bowl="2" rating="0.0" num-stones="4"/>
    <move from-bowl="2" rating="0.0" num-stones="5"/>
    <move from-bowl="2" rating="100.0" num-stones="6"/>
    <move from-bowl="2" rating="0.0" num-stones="7"/>
    <move from-bowl="2" rating="0.0" num-stones="8"/>
    <move from-bowl="3" rating="0.0" num-stones="1"/>
    <move from-bowl="3" rating="0.0" num-stones="2"/>
    <move from-bowl="3" rating="0.0" num-stones="3"/>
    <move from-bowl="3" rating="0.0" num-stones="4"/>
    <move from-bowl="3" rating="0.0" num-stones="5"/>
    <move from-bowl="3" rating="0.0" num-stones="6"/>
  </moves-recommended-for-player>
</game-query-response>
</event>
</annotation>

```

[For an aggressive, careful player, there is only one recommended move, namely, the one which maximizes the number of stones that the character takes and guarantees a win.]

[Harker follows the game board's recommendation and removes 6 stones from bowl 2. The environment sends out an update:]

```

<annotation>
  <event>
    <game-update id="nim">

```

```

    <move-by-player id="Harker">
      <configuration-changes>
        <position new-value="2" old-value="8" id="bowl-2"/>
      </configuration-changes>
    </move-by-player>
    <players-who-can-move>
      <player id="Patrick"/>
    </players-who-can-move>
  </game-update>
</event>
</annotation>

```

[When Harker makes a move with such a high perceived rating, it has the side-effect of producing an internally-motivated event of JOY and PRIDE. This is true on subsequent moves as well; if the moves were rated poorly, it would produce ANGER.]

[Patrick removes 2 stones from the first bowl, causing the environment to emit another event annotation:]

```

<annotation>
  <event>
    <game-update id="nim">
      <move-by-player id="Patrick">
        <configuration-changes>
          <position new-value="2" old-value="4" id="bowl-1"/>
        </configuration-changes>
      </move-by-player>
      <players-who-can-move>
        <player id="Harker"/>
      </players-who-can-move>
    </game-update>
  </event>

```

```
</annotation>
```

[Perceiving this annotation, Harker now knows he can move. He decides to query the environment for recommendations about his next move:]

```
<annotation>
```

```
<event>
```

```
<game-query-response id="nim">
```

```
<moves-recommended-for-player id="Harker">
```

```
<move from-bowl="1" rating="0.0" num-stones="1"/>
```

```
<move from-bowl="1" rating="0.0" num-stones="2"/>
```

```
<move from-bowl="2" rating="0.0" num-stones="1"/>
```

```
<move from-bowl="2" rating="0.0" num-stones="2"/>
```

```
<move from-bowl="3" rating="0.0" num-stones="1"/>
```

```
<move from-bowl="3" rating="0.0" num-stones="2"/>
```

```
<move from-bowl="3" rating="0.0" num-stones="3"/>
```

```
<move from-bowl="3" rating="0.0" num-stones="4"/>
```

```
<move from-bowl="3" rating="0.0" num-stones="5"/>
```

```
<move from-bowl="3" rating="100.0" num-stones="6"/>
```

```
</moves-recommended-for-player>
```

```
</game-query-response>
```

```
</event>
```

```
</annotation>
```

[Harker removes 6 stones from bowl 3:]

```
<annotation>
```

```
<event>
```

```
<game-update id="nim">
```

```
<move-by-player id="Harker">
```

```
<configuration-changes>
```

```
<position new-value="0" old-value="6" id="bowl-3"/>
```

```
</configuration-changes>
```

```

    </move-by-player>
    <players-who-can-move>
      <player id="Patrick"/>
    </players-who-can-move>
  </game-update>
</event>
</annotation>

```

[Harker, whose JOY and PRIDE are raised because of his plays, and because he is actively involved in a game, chooses among his game-commenting behaviors ('Good game so far, huh?')]

[Patrick removes 1 stone from the first bowl.]

```

<annotation>
  <event>
    <game-update id="nim">
      <move-by-player id="Patrick">
        <configuration-changes>
          <position new-value="1" old-value="2" id="bowl-1"/>
        </configuration-changes>
      </move-by-player>
    <players-who-can-move>
      <player id="Harker"/>
    </players-who-can-move>
  </game-update>
</event>
</annotation>

```

[Harker queries for a next move:]

```

<annotation>
  <event>

```

```

    <game-query-response id="nim">
      <moves-recommended-for-player id="Harker">
        <move from-bowl="1" rating="0.0" num-stones="1"/>
        <move from-bowl="2" rating="100.0" num-stones="1"/>
        <move from-bowl="2" rating="0.0" num-stones="2"/>
      </moves-recommended-for-player>
    </game-query-response>
  </event>
</annotation>

[Harker removes 1 stone from the second bowl:]

<annotation>
  <event>
    <game-update id="nim">
      <move-by-player id="Harker">
        <configuration-changes>
          <position new-value="1" old-value="2" id="bowl-2"/>
        </configuration-changes>
      </move-by-player>
    <players-who-can-move>
      <player id="Patrick"/>
    </players-who-can-move>
    </game-update>
  </event>
</annotation>

[Patrick removes 1 stone from the second bowl.]

<annotation>
  <event>
    <game-update id="nim">
      <move-by-player id="Patrick">

```

```
        <configuration-changes>
            <position new-value="0" old-value="1" id="bowl-2"/>
        </configuration-changes>
    </move-by-player>
    <players-who-can-move>
        <player id="Harker"/>
    </players-who-can-move>
</game-update>
</event>
</annotation>
```

[Harker asks for recommendation:]

```
<annotation>
    <event>
        <game-query-response id="nim">
            <moves-recommended-for-player id="Harker">
                <move from-bowl="1" rating="100.0" num-stones="1"/>
            </moves-recommended-for-player>
        </game-query-response>
    </event>
</annotation>
```

[Harker removes the final stone:]

```
<annotation>
    <event>
        <game-update id="nim">
            <game-ends winner="Harker"/>
        </game-update>
    </event>
</annotation>
```

```
[Harker's victory generates intense emotions of JOY, PRIDE, and  
SATISFACTION. He does a little dance.]
```

Listing A.2: Annotations as Harker plays Nim

Appendix B

The Web Gallery: Annotations

B.1 Elements of art vocabulary

The following tables summarize the major elements specific to the art gallery vocabulary and offer examples of what those elements look like.

AREA	
Function	Describes the content of a conceptual group in the gallery (a room, a wall; AREA is the top-level element)
Attributes	NAME, TYPE, URL
Elements	ADJACENT-TO, DESCRIPTION, INTRODUCTION, PART-OF, TOPICS, ARTISTS, ARTWORKS, WORKSPACE
Example	See appendix for full example.

ARTISTS	
Function	Indicates collection of ARTIST elements
Attributes	None
Elements	ARTIST
Example	<pre><ARTISTS> <ARTIST>...</ARTIST> <ARTIST>...</ARTIST> </ARTISTS></pre>

ARTIST	
Function	Describes a single artist
Attributes	NAME (used as unique identifier for artist)
Elements	FIRST-NAME, LAST-NAME, NATIONALITY, YEAR-BORN, YEAR-DIED
Example	<pre><ARTIST NAME="Edward Burne-Jones"> <FIRST-NAME>Edward</FIRST-NAME> <LAST-NAME>Burne-Jones</LAST-NAME> <NATIONALITY>English</NATIONALITY> <YEAR-BORN>1833</YEAR-BORN> <YEAR-DIED>1898</YEAR-DIED> </ARTIST></pre>

ARTWORK	
Function	Describes a single piece of art
Attributes	None
Elements	ARTIST, CATEGORY, ENTITIES, MATERIAL, PART-OF, TITLE, TOPICS, TOURS, YEAR-BEGUN, YEAR-FINISHED
Example	<pre><ARTWORK> <ARTIST>Edward Burne Jones</ARTIST> <CATEGORY>painting</CATEGORY> <ENTITIES>...</ENTITIES> <MATERIAL>oil on canvas</MATERIAL> <PART-OF>...</PART-OF> <TITLE>The Rose Bower</TITLE> <TOPICS>...</TOPICS> <TOURS>...</TOURS> <YEAR-BEGUN>1873<YEAR-BEGUN> <YEAR-FINISHED>1890</YEAR-FINISHED> </ARTWORK></pre>

ENTITY	
Function	Describes a portion of an artwork
Attributes	NAME
Elements	TEXT
Example	<ENTITY NAME= "princess" > <TEXT>...</TEXT> </ENTITY>

PART-OF	
Function	Refers to a larger composition of which artwork is a part
Attributes	None
Elements	NAME, PART-NAME, POSITION
Example	<PART-OF> <NAME>Briar Rose Series</NAME> <PART-NAME>piece</PART-NAME> <POSITION>last</POSITION> </PART-OF>

STOP	
Function	Describes one location on a tour
Attributes	None
Elements	NAME, URL
Example	<STOP> <NAME>the South Wall</NAME> <URL>/gallery/briar/briar4.html</URL> </STOP>

TEXT	
Function	Preformatted dialog
Attributes	AUDIENCE
Elements	None
Example	<TEXT AUDIENCE="child"> Look, there's the beautiful princess. Isn't she pretty? She's asleep on her bed. Her ladies-in-waiting are asleep on the carpet. They will sleep until the magic spell is broken. </TEXT>

TOPIC	
Function	A subject relevant to this element
Attributes	None
Elements	None
Example	<TOPIC>Sleeping Beauty</TOPIC>

TOUR	
Function	A sequence of stops and text in the art gallery
Attributes	NAME, DESCRIPTION
Elements	STOP, FIRST-STOP, NEXT-STOP, TEXT, TOPIC
Example	<TOUR NAME="Sleeping Beauty" DESCRIPTION="story"> <STOP>...</STOP> <FIRST-STOP>...</FIRST-STOP> <NEXT-STOP>...</NEXT-STOP> <TOPIC>...</TOPIC> <TEXT>...</TEXT>

Table B.1: Major elements in the art gallery vocabulary

B.2 Example of gallery annotations

```
<area type="wall" name="Legends of Arthur West Wall" desc="the west wall"
url="arthur/arthur1.html">

<description>the Legends of Arthur</description>
<part-of>Legends of Arthur</part-of>
<adjacent-to>
  <area name="Legends of Arthur Northwest Wall"
url="arthur/arthur2.html" desc="the northwest corner"/>
  <area name="Legends of Arthur North Wall"
url="arthur/arthur3.html" desc="the northeast corner"/>
  <area name="Legends of Arthur Northeast Wall"
url="arthur/arthur4.html" desc="the east wall"/>
  <area name="Legends of Arthur Southeast Wall"
url="arthur/arthur5.html" desc="the east wall"/>
  <area name="Legends of Arthur South Wall"
url="arthur/arthur6.html" desc="the south wall"/>
</adjacent-to>

<workspace>
  <screen size="1024x768" X="800" Y="110"/>
</workspace>

<introduction audience="child">
This part of the gallery shows Pre-Raphaelite paintings about
King Arthur, the Knights of the Round Table, and other people from
the court of Camelot.
</introduction>

<topics>
  <topic>King Arthur</topic>
```

```
<topic>Knights of the Round Table</topic>
</topics>

<artists>
  <artist name="John William Waterhouse">
    <first>John</first>
    <last>Waterhouse</last>
    <nationality>English</nationality>
    <year-born>1849</year-born>
    <year-died>1917</year-died>
  </artist>
</artists>

<artworks>
  <artwork url="gallery/arthur/pic/jones-lastsleep.jpg">
    <title>The Last Sleep of Arthur in Avalon</title>
    <artist>Sir Edward Burne-Jones</artist>

    <date>
      <begun>1881</begun>
      <finished>1898</finished>
    </date>

    <category>painting</category>
    <material>oil on canvas</material>

    <text audience="child">
      The legends say that when King Arthur
      was old, three fairy queens appeared to take him to the Isle of
      Avalon. There he would rest until the world needed him again.
      In this picture, the fairy queens are playing music while he
      sleeps.
    </text>
```

```
<topics>
  <topic>King Arthur</topic>
</topics>

<entities>
  <entity name="arthur">
    <text audience="child">
      King Arthur looks tired, doesn't he.
    </text>
  </entity>
  <entity name="harp">
    <text audience="child">
      The fairy queens are playing their harps. A harp is
      a musical instrument. It's like a piano without
      keys.
    </text>
  </entity>
</entities>

</artwork>
</artworks>
</area>
```

Listing B.1: Annotation describing an artwork in the Web gallery

Bibliography

- [1] Philip E. Agre and David Chapman. What are plans for? In Pattie Maes, editor, *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, pages 17–34. MIT Press, Cambridge, MA, 1990.
- [2] Pilar Almeida. Interactive narrative for online museums. Master’s thesis, Division of Informatics for Social Sciences and Humanities, Graduate School of Human Informatics, Nagoya University, 1999.
- [3] Robert St. Amant and R. Michael Young. Interface agents in model world environments. *AI Magazine*, 22(4):95–107, 2001.
- [4] Eyal Amir and Patrick Doyle. Adventure games: a challenge for cognitive robotics. In *AAAI 2002 Workshop on Cognitive Robotics*, pages 9–16, Menlo Park, CA, 2002. AAAI Press.
- [5] John Anderson and C. Lebiere. *The Atomic Components Of Thought*. Lawrence Erlbaum, Mahwah, NJ, 1998.
- [6] Elisabeth André and Thomas Rist. Controlling the behavior of animated presentation agents in the interface: Scripting versus instructing. *AI Magazine*, 22(4):53–66, 2001.
- [7] Electronic Arts. The Sims Online. Software, 2002. <http://www.thesimsonline.com>.
- [8] Atari. Neverwinter Nights. Software, 2000.

- [9] Norman Badler. Real-time virtual humans. In *Pacific Graphics*, pages 4–14, Seoul, Korea, 1997.
- [10] Gene Ball, Dan Ling, David Kurlander, John Miller, David Pugh, Tim Skelley, Andy Stankosky, David Thiel, Maarten Van Dantzich, and Trace Wax. Lifelike computer characters: the Persona project at Microsoft Research. In Jeffrey M. Bradshaw, editor, *Software Agents*. AAAI Press, Menlo Park, CA, 1997.
- [11] Richard Bartle. Interactive multi-user computer games. Technical report, British Telecom, 1990.
- [12] Richard Bartle. *Designing Virtual Worlds*. New Riders, Indianapolis, IN, 2003.
- [13] Joseph Bates. The nature of characters in interactive worlds and the Oz project. Technical Report CMU-CS-92-200, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, October 1992.
- [14] Joseph Bates. The role of emotion in believable agents. *Communications of the ACM*, 37(7):122–125, July 1994.
- [15] Joseph Bates, A. Brian Loyall, and W. Scott Reilly. An architecture for action, emotion, and social behavior. Technical Report CMU-CS-92-142, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, July 1992.
- [16] David Bearman and Jennifer Trant, editors. *Museums and the Web: Selected Papers from the 2002 Conference*, Boston, MA, 2002. Archives and Museum Informatics.
- [17] Elwyn Berlekamp, John Conway, and Richard Guy. *Winning Ways for Your Mathematical Plays*. A. K. Peters, Ltd., London, 1982.
- [18] Timothy Bickmore. *Relational Agents: Effecting Change Through Human-Computer Relationships*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 2003.
- [19] Marc Blank and David Lebling. Zork. Software. Infocom.

- [20] Bruce Blumberg. *Old Tricks, New Dogs: Ethology and Interactive Characters*. PhD thesis, Media Laboratory, Massachusetts Institute of Technology, Cambridge, MA, 1996.
- [21] Bruce Blumberg, P. Todd, and Pattie Maes. No bad dogs: Ethological lessons for learning in hamsterdam. In *From Animals to Animats: Proceedings of the Fourth International Conference on the Simulation of Adaptive Behavior*, Cambridge, MA, 1996. MIT Press.
- [22] Bruce M. Blumberg and Tinsley A. Galyean. Multi-level direction of autonomous creatures for real-time virtual environments. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 47–54. ACM Press, 1995.
- [23] Charles L. Bouton. Nim, a game with a complete mathematical theory. *Annals of Mathematics*, 3:35–39, 1902.
- [24] Rodney Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robots and Automation*, 2(1):14–23, 1986.
- [25] Rodney Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15, 1990.
- [26] Rodney Brooks. Intelligence without representation. *Artificial Intelligence Journal*, 1991.
- [27] Janet Cahn. The generation of affect in synthesized speech. *Journal of the American Voice I/O Society*, 8:1–19, July 1990.
- [28] Dolores Canamero, editor. *Emotional and Intelligent: The Tangled Knot of Cognition*, number FS-98-03 in Fall Symposium Series. AAAI, AAAI Press, 1998.
- [29] K. Colby, F. Hilf, S. Weber, and H. Kraemer. Turing-like indistinguishability tests for the validation of a computer simulation of paranoid processes. *Artificial Intelligence*, 3:199–222, 1972.

- [30] William Crowther. Colossal Cave. Software, 1972. No longer extant; see e.g., <http://www.rickadams.org/adventure/>.
- [31] Antonio Damasio. *Descartes' Error: Emotion, Reason, and the Human Brain*. G. P. Putnam's Sons, New York, 1994.
- [32] Patrick Doyle. Characters and roles in computer gaming. Unpublished manuscript.
- [33] Patrick Doyle. Believability through context: Using “knowledge in the world” to create intelligent characters. In Cristiano Castelfranchi and W. Lewis Johnson, editors, *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, volume 1, pages 342–349. ACM Press, 2002.
- [34] Patrick Doyle and Barbara Hayes-Roth. Agents in annotated worlds. In *Proceedings of the Second International Conference on Autonomous Agents*, pages 173–180, Minneapolis, MN, October 1998. ACM Press.
- [35] Patrick Doyle and Barbara Hayes-Roth. Guided exploration of virtual worlds. In Fay Sudweeks, Margaret McLaughlin, and Sheizaf Rafaeli, editors, *Network and Netplay: Virtual Groups on the Internet*, pages 243–264. AAAI Press, Menlo Park, CA, 1998.
- [36] C. Dyer. *Interpersonal goals and satisfaction with interactions*. PhD thesis, Dept. of Communications, Stanford University, Stanford, CA, 1993.
- [37] Clark Elliott. *The Affective Reasoner: A Process Model of Emotions in a Multi-Agent System*. PhD thesis, The Institute for the Learning Sciences, Northwestern University, Evanston, IL, 1992.
- [38] Clark Elliott. Research problems in the use of a shallow artificial intelligence model of personality and emotion. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 9–15, Seattle, WA, August 1995. AAAI Press.

- [39] Clark Elliott. I picked up Catapia and other stories: A multimodal approach to expressivity for “emotionally intelligent” agents. In *Proceedings of the First International Conference on Autonomous Agents*, pages 451–457, New York, 1997. ACM Press.
- [40] Clark Elliott and Jacek Brzezinski. Autonomous agents as synthetic characters. *AI Magazine*, Summer:13–30, 1998.
- [41] Clark Elliott, Jeff Rickel, and James Lester. Lifelike pedagogical agents and affective computing: An exploratory synthesis. *Lecture Notes in Computer Science*, 1600:195–212, 1999.
- [42] Clark Elliott, Jeff Rickel, and James Lester. Lifelike pedagogical agents and affective computing: An exploratory synthesis. In Michael Wooldridge and Manuela Veloso, editors, *Artificial Intelligence Today: Recent Trends and Developments*, volume 1600 of *Artificial Intelligence Notes*. Springer-Verlag, Berlin, June 2003.
- [43] Turbine Entertainment. Asheron’s Call. Software, 1999. <http://www.microsoft.com/games/zone/asheronscall/>.
- [44] R. J. Firby. *Adaptive Execution in Complex Dynamic Domains*. PhD thesis, Yale University, 1989.
- [45] Leonard Foner. What’s an agent, anyway? a sociological case study. Agents Memo 93-01, Media Laboratory, Massachusetts Institute of Technology, Cambridge, MA, May 1993.
- [46] Stan Franklin and Art Graesser. Is it an agent, or just a program: a taxonomy for autonomous agents. In Jörg Müller, Michael Woolridge, and Nicholas Jennings, editors, *Intelligent Agents III*. Springer-Verlag, 1997.
- [47] Nico H. Frijda and Jaap Swagerman. Can computers feel? theory and design of an emotional system. *Cognition and Emotion*, 1(1):235–257, 1987.

- [48] Toby Gard. Building character. *Game Developer Magazine*, 7(5):53–59, May 2000.
- [49] Martin Gardner. *Mathematical Puzzles and Diversions*. Penguin, New York, 1959.
- [50] M. Georgeff and A. Lansky. Reactive reasoning and planning. In *Proceedings of AAAI '87*, pages 677–682. Morgan Kaufmann, 1987.
- [51] J. J. Gibson. The theory of affordances. In R. Shaw and J. Bransford, editors, *Perceiving, Acting, and Knowing: Toward an Ecological Psychology*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1977.
- [52] Athomas Goldberg. Improv: A system for real-time animation of behavior-based interactive synthetic actors. *Lecture Notes in Computer Science*, 1195:58–84, 1997.
- [53] S. Grand, D. Cliff, and A. Malhotra. CREATURES: Artificial life autonomous software agents for home entertainment. In *Proceedings of the First International Conference on Autonomous Agents*, pages 22–29, New York, 1997. ACM Press.
- [54] Jonathan Gratch and Stacy Marsella. Tears and fears: modeling emotions and emotional behaviors in synthetic agents. In *Proceedings of the fifth international conference on Autonomous agents*, pages 278–285. ACM Press, 2001.
- [55] Steve Hanks, Martha Pollack, and Paul Cohen. Benchmarks, testbeds, controlled experimentation and the design of agent architectures. Technical Report Technical Report 93-06-05, Department of Computer Science and Engineering, University of Washington, Seattle, WA, June 1993.
- [56] Barbara Hayes-Roth, Lee Brownston, Ruth Huard, Robert van Gent, and Erik Sincoff. Directed improvisation. Technical Report KSL-94-61, Knowledge Systems Laboratory, Stanford University, September 1994.

- [57] Barbara Hayes-Roth and Patrick Doyle. Animate characters. *Autonomous agents and multi-agent systems*, 1:195–230, 1998.
- [58] Barbara Hayes-Roth and M. Hewett. BB1: An implementation of the blackboard control architecture. In Robert Englemore and T. Morgan, editors, *Blackboard Systems*, chapter 14, pages 297–313. Addison-Wesley, Reading, MA, 1988.
- [59] Barbara Hayes-Roth and Robert van Gent. Story-making with improvisational puppets. In *Proceedings of the First International Conference on Autonomous Agents*, pages 1–7, New York, 1997. ACM Press.
- [60] Barbara Hayes-Roth, Robert van Gent, and Daniel Huber. Acting in character. In R. Trappl and Paolo Petta, editors, *Creating Personalities for Synthetic Actors*. Springer-Verlag, Berlin, 1997.
- [61] Ed Hooks. *Acting for Animators*. Heinemann, Portsmouth, NH, 2000.
- [62] There Inc. There. Software, 2003. <http://www.there.com>.
- [63] K. Isbister, H. Nakanishi, Toru Ishida, and Clifford Nass. Helper agent: Designing an assistant for human-human interaction in a virtual meeting space. In *Proceedings of CHI 2000*, pages 57–64, The Hague, Netherlands, 2000.
- [64] Katherine Isbister and Patrick Doyle. Touring machines: Guide agents for sharing stories about digital places. In *Narrative Intelligence: Papers from the 1999 AAI Fall Symposium*, pages 93–100. AAAI, AAAI Press, 1999.
- [65] Katherine Isbister and Patrick Doyle. Web guide agents: Narrative context with character. In Michael Mateas and Phoebe Sengers, editors, *Narrative Intelligence*, pages 229–244, Amsterdam, 2003. John Benjamins.
- [66] Katherine Isbister and Patrick Doyle. Meaningful evaluations of interdisciplinary efforts: A proposed taxonomy for eca research. In Catherine Pelechoud and Zsofia Ruttkay, editors, *Evaluating Embodied Conversational Agents*. Kluwer Academic Press, Forthcoming.

- [67] Katherine Isbister and Clifford Nass. Consistency of personality in interactive characters: Verbal cues, non-verbal cues, and user characteristics. *International Journal of Human-Computer Studies*, 53(2):251–267, 2000.
- [68] W. Lewis Johnson. Pedagogical agent research at CARTE. *AI Magazine*, 22(4):85–94, 2001.
- [69] W. Lewis Johnson and Jeff Rickel. Integrating pedagogical capabilities in a virtual environment agent. In *Proceedings of the First International Conference on Autonomous Agents*, pages 30–38, Marina del Rey, CA, February 1997. ACM Press.
- [70] Keith Johnstone. *IMPRO: Improvisation and the Theater*. Routledge, New York, 1992.
- [71] Chuck Jones. *Chuck Amuck: The Life and Times of an Animated Cartoonist*. Farrar, Straus and Giroux, 1989.
- [72] Raph Koster. Online World Timeline. Web page. <http://www.legendmud.org/raph/gaming/mudtimeline.html>.
- [73] Rudolph Laban. *Language of Movement: A Guidebook to Choreutics*. Play, Inc., Boston, MA, 1974.
- [74] John Laird. Preface for special section on integrated cognitive architectures. *SIGART Bulletin*, 2:12–13, 1991.
- [75] John Laird, Allan Newell, and Paul Rosenbloom. Soar: An architecture for general intelligence. *Artificial Intelligence*, 33:1–64, 1987.
- [76] John E. Laird. It knows what you're going to do: Adding anticipation in QUAKEBOT. In *Papers from the AAAI 2000 Spring Symposium on Artificial Intelligence and Interactive Entertainment*, Technical Report SS-00-02, pages 41–50, Menlo Park, CA, 2000. AAAI Press.

- [77] John E. Laird. Research in human-level AI using computer games. *Communications of the ACM*, 45(1):32–35, 2002.
- [78] John E. Laird and Michael van Lent. Human-level AI’s killer application: Interactive computer games. *AI Magazine*, 22(2):15–25, Summer 2001.
- [79] Pat Langley and John Laird. Cognitive architectures: Research issues and challenges. Unpublished manuscript, October 2002.
- [80] John Lasseter. Principles of traditional animation applied to 3D animation. *Computer Graphics*, 21(4):35–43, July 1987.
- [81] Brenda Laurel. *Computers as Theatre*. Addison-Wesley, 1993.
- [82] David Lebling, Marc Blank, and T. Anderson. Zork: A computerized fantasy simulation game. *IEEE Computers Magazine*, pages 51–59, April 1979.
- [83] Douglas B. Lenat. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.
- [84] James Lester, S. Converse, S. Kahler, T. Barlow, B. Stone, and R. Bhogal. The persona effect: affective impact of animated pedagogical agents. In *Proceedings of CHI '97*, Atlanta, GA, 1997.
- [85] James Lester and Brian Stone. Increasing believability in animated pedagogical agents. In *Proceedings of the First International Conference on Autonomous Agents*, pages 16–21, Marina del Rey, CA, February 1997. ACM Press.
- [86] James C. Lester, Stuart G. Towns, and Patrick J. FitzGerald. Achieving affective impact: Visual emotive communication in lifelike pedagogical agents. *The International Journal of Artificial Intelligence in Education*, 10(3–4):278–291, 1999.
- [87] A. Bryan Loyall. *Believable Agents: Building Interactive Personalities*. PhD thesis, Carnegie Mellon University, May 1997. Available as technical report CMU-CS-97-123.

- [88] Bryan Loyall and Joseph Bates. A reactive, adaptive architecture for agents. Technical Report CMU-CS-91-147, Carnegie-Mellon University, June 1991.
- [89] Pattie Maes. Artificial life meets entertainment: Interacting with lifelike autonomous agents. *Communications of the ACM*, 38(11):108–114, November 1995.
- [90] Stacy Marsella and Jonathan Gratch. Modeling coping behavior in virtual humans: don't worry, be happy. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 313–320. ACM Press, 2003.
- [91] Stacy C. Marsella, W. Lewis Johnson, and Catherine LaBore. Interactive pedagogical drama. In *Proceedings of the fourth international conference on Autonomous agents*, pages 301–308. ACM Press, 2000.
- [92] Michael Mateas. *Interactive Drama, Art, and Artificial Intelligence*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, December 2002.
- [93] Michael Mateas and Phoebe Sengers, editors. *Narrative Intelligence*. John Benjamins, Amsterdam, 2003.
- [94] Michael Mateas and Andrew Stern. Architecture, authorial idioms and early observations of the interactive drama *facade*. Technical Report CMU-CS-02-198, School of Computer Science, Carnegie Mellon University, December 2002.
- [95] Maxis. *The Sims*. Software, 2000.
- [96] Herman Melville. *Bartleby the Scrivener*. Simon and Schuster, New York, 1853.
- [97] Steven Minton. Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, 42:363–391, 1990.
- [98] Janet Murray. *Hamlet on the Holodeck: The Future of Narrative in Cyberspace*. Simon and Schuster, 1997.

- [99] Clifford Nass, Y. Moon, B. J. Fogg, Byron Reeves, and C. Dryer. Can computer personalities be human personalities? *International Journal of Human-Computer Studies*, 43:223–239, August 1995.
- [100] Clifford Nass, J. Steuer, and E. Tauber. Computers are social actors. In *Proceedings of CHI '94*, Boston, MA, 1994.
- [101] Donald Norman. *The Design of Everyday Things*. Doubleday, New York, 1990.
- [102] Donald Norman. Cognitive artifacts. In J. M. Carroll, editor, *Designing interaction: psychology at the human-computer interface*, pages 17–38. Cambridge University Press, 1991.
- [103] Donald Norman. *Things That Make Us Smart*. Addison-Wesley, Reading, MA, 1993.
- [104] Keith Oatley and Philip Johnson-Laird. Towards a cognitive theory of emotions. *Cognition and Emotion*, 1(1):29–50, 1987.
- [105] Tim Oren, Gitta Salomon, Kristee Kreitman, and Abbe Don. Guides: Characterizing the interface. In Brenda Laurel, editor, *The Art of Human-Computer Interaction*, pages 367–381. Addison-Wesley, Reading, MA, 1990.
- [106] Andrew Ortony, Gerald Clore, and Allan Collins. *The Cognitive Structure of Emotions*. Cambridge University Press, 1988.
- [107] Catherine Pelechaut and Zsofia Ruttkay, editors. *Evaluating Embodied Conversational Agents*. Kluwer Academic Press, 2004.
- [108] Ken Perlin and Athomas Goldberg. Improv: A system for scripting interactive actors in virtual worlds. *Computer Graphics*, 30:205–216, 1996.
- [109] Per Persson, Kristina Höök, and Marie Sjölander. Agneta and Frida: Merging web and narrative? In Michael Mateas and Phoebe Sengers, editors, *Narrative Intelligence*, pages 245–258, Amsterdam, 2003. John Benjamins.
- [110] Rosalind W. Picard. *Affective Computing*. The MIT Press, 1997.

- [111] Tim Read and Aaron Sloman. The terminological pitfalls of studying emotion. Technical report, University of Birmingham, August 1993.
- [112] Byron Reeves and Clifford Nass. *The Media Equation*. Cambridge University Press, Cambridge, 1996.
- [113] W. Scott Neal Reilly. *Believable social and emotional agents*. PhD thesis, Carnegie Mellon University, May 1996. Available as technical report CMU-CS-96-138.
- [114] W. Scott Neal Reilly. A methodology for building believable social agents. In *Proceedings of the First International Conference on Autonomous Agents*, pages 114–121, Marina del Rey, CA, February 1997. ACM Press.
- [115] Howard Rheingold. *The Virtual Community: Homesteading on the Electronic Frontier*. HarperCollins, New York, 1994.
- [116] Bradley Rhodes. PHISH-Nets: Planning heuristically in situated hybrid networks. Master’s thesis, Media Laboratory, Massachusetts Institute of Technology, Cambridge, MA, 1996.
- [117] Jeff Rickel and W. Lewis Johnson. STEVE: A pedagogical agent for virtual reality. In Katia Sycara and Michael Wooldridge, editors, *Proceedings of the Second International Conference on Autonomous Agents*, pages 332–333, New York, May 1998. ACM Press.
- [118] Thomas Rist, Elizabeth Andre, and Jörg Müller. Adding animated presentation agents to the interface. In *Proceedings of the International Conference on Intelligent User Interfaces*, pages 21–28, Orlando, FL, January 1997. ACM Press.
- [119] Paul Rosenbloom, John Laird, and Allen Newell, editors. *The Soar Papers: Research on Integrated Intelligence*. MIT Press, Boston, MA, 1993.

- [120] Daniel Rousseau and Barbara Hayes-Roth. A social-psychological model for synthetic actors. In *Proceedings of the Second International Conference on Autonomous Agents*, pages 165–172, Minneapolis, MN, May 1998. ACM Press.
- [121] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 1995.
- [122] Jonathan Schaeffer. *One Jump Ahead: Challenging Human Supremacy in Checkers*. Springer-Verlag, Berlin, 1997.
- [123] Roger Schank and Roger Abelson. *Scripts, Plans, Goals, and Understanding*. Lawrence Erlbaum, Hillsdale, NJ, 1977.
- [124] Roger C. Schank. Conceptual dependency: A theory of natural language understanding. *Cognitive Psychology*, 3(4):pages 532–631, 1972.
- [125] Phoebe Sengers. *Anti-Boxology: Agent Design in Cultural Context*. PhD thesis, Carnegie-Mellon University, 1998.
- [126] Phoebe Sengers. Do the thing right: An architecture for action expression. In *Proceedings of the Second International Conference on Autonomous Agents*, pages 24–31, Minneapolis, MN, May 1998. ACM Press.
- [127] Daniel Shapiro and Pat Langley. Controlling physical agents through reactive logic programming. In *Proceedings of the Third International Conference on Autonomous Agents*, pages 386–387, Seattle, 1999. ACM Press.
- [128] Aaron Sloman. Beyond shallow models of emotion. *Cognitive Processing*, 2(1):177–198, Summer 2001.
- [129] Sony. Everquest. Software, 1999. <http://everquest.station.sony.com/>.
- [130] Constantin Stanislavski. *Creating A Role*. Theatre Arts Books, New York, 1961.
- [131] Robert Louis Stevenson. *The Strange Case of Dr. Jekyll and Mr. Hyde*. Scribner, New York, 1886.

- [132] Brian Stone and James Lester. Dynamically sequencing an animated pedagogical agent. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 424–431, Portland, OR, August 1996. AAAI Press.
- [133] Lucy A. Suchman. *Plans and Situated Actions: The Problem of Human-Machine Communication*. Cambridge University Press, Cambridge, UK, 1987.
- [134] Origin Systems. Ultima Online. Software, 1997. <http://www.wo.com>.
- [135] M. Tambe, W. L. Johnson, R. M. Jones, F. Koss, J. E. Laird, P. S. Rosenbloom, and K. B. Schwamb. Intelligent agents for interactive simulation environments. *AI Magazine*, 16:15–39, 1995.
- [136] MudOS Development Team. MudOS. Software, 2003. <http://mudos.org>.
- [137] Frank Thomas and Ollie Johnson. *The Illusion of Life: Disney Animation*. Hyperion Books, New York, 1981.
- [138] Robert Trappl and Paolo Petta, editors. *Creating Personalities for Synthetic Actors*. Springer-Verlag, Berlin, 1996.
- [139] Sherry Turkle. *Life On The Screen*. Simon and Schuster, New York, 1995.
- [140] Scott A. Wallace and John E. Laird. Comparing agents and humans using behavioral bounding. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI '03)*, Acapulco, Mexico, August 2003.
- [141] Walt Disney Studios. Snow White. Film, 1937.
- [142] Walt Disney Studios. The Sword In The Stone. Film, 1970.
- [143] Joseph Weizenbaum. ELIZA — A computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966.
- [144] Peter Weyhrauch. *Guiding Interactive Drama*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, January 1997. Technical Report CMU-CS-97-109.

- [145] T. H. White. *The Once and Future King*. Collins, London, 1958.
- [146] Jerry Wiggins. *The Five-Factor Model of Personality*. Guilford Press, 1996.
- [147] Michael Wooldridge and Nicholas Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2), June 1995.
- [148] Michael Wooldridge and Nicholas Jennings. Pitfalls of agent-oriented development. In *Proceedings of the 2nd International Conference on Autonomous Agents*, pages 385–391, Minneapolis, MN, May 1998.
- [149] Ian Wright, Aaron Sloman, and Luc Beaudoin. Towards a design-based analysis of emotional episodes. *Philosophy, Psychiatry, and Psychology*, 3(2):101–126, 1996.