1 Introduction and Overview

Introduction

Preface

I've long dreamt of being able to construct my own characters; characters that had a life of their own but were willing and able to let me share in their worlds and experience. In my mind's eye, I always saw them as fully three-dimensional, living in combinations of photorealistic and sketchy cartoon worlds. As a child I built characters constantly, acting out roles with toy soldiers and "action figures", building complex structures out of wood and mud; moving whole cities and armies around with my hand and my mind.

In high school and as an undergraduate, I spent an inordinate amount of time involved in theater; usually as an actor, but sometimes behind the scenes designing sets or managing props. As an actor, I was able to create characters and directly inhabit their bodies and minds. As a set designer, I was able to influence the construction of characters by enabling or constraining their activity, as well as realize the possibilities of a space itself as a kind of character. As a "prop master", I appreciated the power of the prop; how a certain prop could enable the perfect "bit of business" that, in the hands of the right actor, make a scene really work.

It's a perfectly logical progression then, as my attention and studies turned towards computers, that my dissertation would end up addressing issues vital to the problem of building computational characters that exist in the digital domain. My interest continues to be in the *process* of bringing a character to life; I am chiefly concerned with its construction, in what situations it might find itself in, and how it might handle itself in the context of some larger, scripted, narrative activity.

It should be noted that this approach is in contrast to the equally valid, but different, problem of freely interacting with a character in some emergent narrative form (**Laurel91, Bates93, Blumberg95**, **Galyean95**). I feel that until we can credibly talk about how the

character was constructed, what trade-offs were made in its design and implementation, and what ways we might improve on the current design — in short, until we address the **character construction process**, I would contend that we have little chance of making measurable progress in our desire to populate virtual environments with 3D semi-autonomous animated characters, at least in any way that we can generalize from our experience and reuse what we build.

From an actor's perspective, this means that I'm concerned with the rehearsal process, not the performance. It's not that the performance is not important; it's clearly the final goal. But until we've constructed the character over the course of many rehearsals, it makes little sense to invite an audience and raise the curtain.

What's so fascinating about the promise of doing this in the digital domain is in the fact that the experience, *the entire experience of what transpired*, can be recorded. Recorded for reuse, appropriation, reexperience in a different form or resolution — all of these become *possible* as we bring our characters to life in the digital domain. Our task, which I've taken on for this thesis, is to *decide how we can build them* so that this can happen.

Background

Researchers in Artificial Intelligence (**AI**) and Computer Graphics (**CG**) have long shared a dream that someday they would develop systems to allow the construction of three-dimensional animated characters. When given a particular task, such characters would be capable of independent activity in a virtual environment, and their activity would be purposeful and interesting. They would be able to be integrated into a given environment seamlessly, so that the artificial actor was indistinguishable from a human participant. Human users would be able to interact with such characters at the task level, asking them to perform activities or even join the human in performing some task.

Over the years, the AI community has made strides in many related areas, including reactive planning, constraint systems, and knowledge representation. Computational graphicists have made enormous advances in physically-based modeling, photorealistic scene description and rendering. Both fields have benefited enormously from fast workstations and advanced software development environments. Unfortunately, whenever researchers in either of these two domains try to address the larger character construction issue, they rarely use state-of-the-art techniques from the other. More

importantly, when researchers do try to integrate techniques from both fields into a solution, almost invariably they end up treating the other discipline's contribution to the problem as purely "an implementation issue. "

For example, when considering a scenario in which a character is given the task of arranging some items on a table, most AI researchers have concentrated on issues surrounding getting the character to decide to "pick up block A first, then block B." How the character goes about picking up block A (text output, robot arm, flat shaded rigid links, etc.) is usually considered an "implementation issue." What matters to the AI researcher is that the system decided to engage in the activity based on some internal and external motivations, and that the system has some mechanisms for determining, after the fact, if the activity was successful or not, and then reacting accordingly. How the actual activity is implemented is not usually considered an integral part of the problem.

Given the same domain, computational graphicists have instead concentrated on the issues of describing how the character and its environment's attributes (i.e. position, orientation, coloring, shading, geometric shape) change over the course of actually picking up the block, and how these changes occur in three-dimensional space and are eventually visualized onto an array of pixels. How the system decided to pick up a given block in the first place, or what it did next, were not considered part of the problem. Moreover, the task of modulating that instance of behavior during that particular span in time and space is almost invariably given over to a human (i.e. an animator).

While this approach has been successful in coming up with solutions for many of the subproblems associated with the larger goal of constructing characters, no single system to date satisfactorily meets the goal of allowing the construction of arbitrary three-dimensional, semi-autonomous, animated characters engaged in purposeful and interesting behavior in a virtual environment, such that the virtual actors can be seamlessly integrated with real environments.

The goal of this work has been to address this problem by constructing a testbed for experimenting with the issues that arise in the construction of such characters, where the result is amenable to the application of state-of-the-art techniques from both the AI and CG domains.

The Thesis

The basic thesis that this dissertation discusses is that by considering computer graphics and artificial intelligence techniques as peers in a solution, rather than relegating one or the other to the status of an "implementation detail", we can achieve a synergy greater than the sum of the parts. This will allow us to overcome previous limitations and allow the development of a character construction system that will foster collaboration and foment discussion by practitioners in both domains.

It is my thesis that by using a single time-based representation for the *shape*, *shading* and *state* information that both the AI and the CG subsystems can **measure** and **manipulate** at an appropriate level of detail, I will be able to demonstrate an approach that allows and fosters the construction of arbitrary three-dimensional, semi-autonomous, animated characters that can engage in purposeful and interesting directed behavior in a virtual environment.

I claim that any such representation must have mechanisms to allow proper sampling of behavior, and mechanisms for intelligently reconstructing the results of those behaviors later. Ideally, the system should allow for **scalable** behavior to be built, where mechanisms can be put in place to behave differently in the presence of greater or lesser sampling of the behavior. The representation should encourage the construction of **reusable** behaviors that can be reused among a range of characters. These behaviors should be amenable to **composition**, so that sets of behaviors can be composed together to form more complex behaviors. Finally, the representation should facilitate the construction of **blendable** behavior, where the complementary result of behaviors blend together in their manipulation of the representation of the character.

Finally, we claim that due to the iterative nature of the character construction process, and the inherent demand for sophisticated debugging tools, it is imperative that the representation be able to be easily built up over time using state-of-the-art data manipulation and visualization techniques.

The Problem Posed as Questions

The problem that this thesis addresses can be best summarized as the following set of questions:

- How can we construct virtual actors (three-dimensional, animated semi-autonomous computer characters) in such a way that they are able to sense and act in the virtual environments they inhabit? (.i.e. "it is hot", "the ball is nearby", "I'm surrounded by enemies", etc.)

- How we can construct virtual actors such that both end users and character designers can interact with them at the task level? ("Open the door", "clean this place up", "fix the motor", "look at this", etc.)

- What is the computational infrastructure we need to compose such characters out of reusable parts authored by disparate groups of modelers/animators/artisans?

- How can we construct our characters in such a way that they have an understanding of their present resources in the computational infrastructure they inhabit and allow them to make use of this information in their activity? In other words, how can we reconcile the desire to naturally express behaviors of these characters as continuous functions over time with the discrete, sampled nature of animation? (i.e. are they running on a fast machine, running on a slow network, am I being undersampled with respect to the kind of signal (change in the model over time) I am trying to produce, etc.)

- How can we construct them such that they can be seamlessly integrated into virtual environments containing both real and computer generated imagery at arbitrarily high levels of quality (i.e. scalable high definition video, 35mm motion picture film, ShowScan, etc.)?

All of these questions are intertwined. The thesis will directly address the first three questions, while the latter two are important implementation issues that will be addressed in the high level design and implementation of the prototype testbed.

My Approach: An Overview

This section provides a brief overview of my answers to the questions posed in the previous section. Each of these issues is described and discussed in more detail in the rest of the dissertation. Before I begin, though, I feel a short glossary of terms I use frequently in WavesWorld (and will be using throughout this dissertation) would be useful to the reader unfamiliar with this work. Please note that all these terms will be explained, in context, in the following chapters, but this brief glossary of terms will hopefully stem any unnecessary initial confusion.

A Quick WavesWorld Glossary

model
A model is an encapsulation of the **shape**, **shading**, and **state** information of a character. *Shape* refers to the geometry of objects. This includes what geometric primitives they're composed of (spheres, polygons, patch meshes, etc.), as well as what geometric transformations are acting on them (scale, skew, translate, rotate, etc.). *Shading* refers to the subtle surface qualities of an object due to the material properties of the objects in the scene, local illumination effects, and global illumination effects. *State* refers to other more ineffable, but still perceivable qualities, like elements of personality.

In WavesWorld, a model is an ordered list of **renderable** objects, some of which are **animatable**. *Renderable* objects are objects that respond to messages instructing them to render themselves over some span of time, and *animatable* objects are renderable objects whose properties/slots may change over time. The properties/slots of a model that can change over time are called **articulated variables**.

agent
In WavesWorld, an agent really just refers to a software module; an autonomous black box that has a well defined purpose, a separate namespace and its own thread of control. I use the term very carefully, and whenever it is used in this dissertation, it explicitly refers to such a black box that, when opened, is readily understandable by someone who might build such things (i.e. a programmer).

sensor agent
An agent which encapsulates some boolean perceptual mechanism. A sensor agent gathers its information by measuring some model(s) that correspond(s) to the character, its environment, or both.

goal agent
An agent which encapsulates a desire of the character, in terms of a sensor agent that it wants to be True.

skill agent
An agent which generates the behavior of a character. It does this by, *over the course of some amount of time*, **measuring** and **manipulating** the articulated variables of some model(s) that correspond(s) to the character, its environment, or both.

character
A character is the combination of a **model** and some set of **agents**, which **measure** and **manipulate** the model over time. The agents may be in a variety of configurations: chained together, competing, cooperating, etc.

My Approach

In order to construct virtual actors which can sense and act in the virtual environment they inhabit, we characterize the functional components of a character as an interconnected network of goal agents, skill agents, and sensor agents. A given set of agents can be connected together in a variety of configurations, with the most complex that we've explored is a reactive planner based on an implementation of the "spreading activation" planning algorithm described initially in (**Maes89**). We have significantly extended this algorithm to allow asynchronous action selection and parallel action execution. This work (**Johnson91**) was completed in 1991 for my SMVS thesis "Build-a-Dude: Action Selection Networks for Computational Autonomous Agents." We have since made several extensions to both the algorithm and our way of thinking about it (see **Chapter 4** and **Appendix A**).

Using Rasmussen's characterization of the *signs*, *signals*, and *symbols* to which a virtual actor attends (**Rasumussen83**), we dealt with the issues of dividing the perceptual mechanisms of a virtual actor into the discrete sampler of the continuous variable (a **receptor**) and the computational black box which computed the truth value of a given assertion based on the current sample set (the **sensor agent**). This split has powerful consequences with regard to graded recruitment of resources and apportionment of attention by the virtual actor to itself and its environment. This work is also complete, and has been integrated into the current implementation of the distributed, parallel planner and agent network, although it is not completely hooked up to the latest release of the testbed.

In order to build actors that we can interact with at the task level, it is necessary to facilitate the construction of sophisticated multi-modal user interfaces to measure and manipulate our characters. To this end, I have extended the NEXTSTEP® development environment to allow both the manipulation and visualization of character components (agents, shaders, shape, and behavior). This allows for levels of abstraction to be built up for a given character, allowing *guiding*, *animator*, and *task level* interaction with a given character (**Zeltzer85**). This work has been under development for the last few years, and is in use by several hundred users over the world. The design of this software and its current implementation will be discussed in **Chapter 3**.

In order to build the computational infrastructure necessary to build characters out of

reusable parts authored by disparate groups of creators, it was necessary to use a single discrete time-based representation for the shape, shading and state of information of the characters and their environment. I used the RenderMan® Interface (**Pixar89**) as a starting point for designing the **eve object framework** to address this need. While the core of the framework (a modeling language, compiler, and run-time system) has up and working for over two years, we have been successfully using this framework to construct character parts over the past year. Like any language, understanding and facility grow over time, and my understanding of what we've wrought has come forward in leaps and bounds in the last few months, but there is still room left in the learning curve. Examples of using this approach are explained in **Chapter 3**, and some future directions for this will be addressed in **Chapter 5**.

In order to construct characters that can be seamlessly integrated with both real and computer imagery, we must be able to resolve the discrete nature of computational simulation/animation with the desire to record a scene continuously (i.e. with no sampling artifacts). This means that the underlying representation must maintain some continuity over time in order to interpolate parts for moments when it does not already have a sample. This problem is also intimately tied to the issue of giving a character's components a valid sense of their dynamic computation and communication resources while executing. In other words, it's vitally important to allow agents to have an understanding of how finely they are being sampled, so that they can adjust the signals they are generating accordingly. WavesWorld facilitates this by decentralizing sampling frequency to the signal generators (the **agents**), and by using a framework that allows object-based reconstruction and interpolation of signals in the scene (**eve**). Also, since I based my framework atop the RenderMan® Interface, and took full advantage of the capabilities provided by it, my system can easily take advantage of the state-of-the-art in commercial and research rendering systems (i.e. Pixar's Academy Award® winning PhotoRealistic RenderMan® renderer and Larry Gritz's hybrid raytracer/radiosity renderer from the Blue Moon Rendering Tools).

Contributions of this Work

The central contribution of this dissertation is a testbed for experimenting with the issues surrounding designing, developing, debugging, and delivering 3D, semi-autonomous animated characters.

This testbed sits atop an object-oriented framework for constructing and animating models. This framework facilitates iterative construction of the parts and props that comprise a character. It also provides facilities for writing and wiring together agents, which are processes that measure and manipulate models over time to produce a character's behavior. This framework encourages and facilitates encapsulation and reuse at many levels, which benefits collaborative character construction.

This testbed can be used to compose three-dimensional, photorealistic animatable characters, where characters are composed of variously interconnected agents and a model, where a model is a set of objects encapsulating shape, shading and state information over time. It's possible to quickly build reusable, composable, blendable behaviors, where a behavior is the result of some set of processes operating on a model over time.

One especially useful result of using this framework to develop media is its facility in acting as a very rich and compact storage medium for photorealistic scenes. This storage representation builds directly atop the RenderMan® Interface, an industry standard for describing photorealistic scenes. In our object-oriented representation, though, since we maintain some level of continuity in our scenes over time, such scenes can have 3D models that change over time, where different parts of the models in the scene can be changing at different rates. Especially interesting is that these scenes need only a very modest run-time system for playback at arbitrary frame rates with respect to the scene time (in addition to a graphics system for display).

Assuming the underlying components of the scene were sampled appropriately, the scene can be played back at arbitrary spatial and temporal frequency. In other words, the scene can be treated as continuous media. With appropriate sampling, the representation is not lossy. For a large class of scenes, this allows orders of magnitude of compression of the amount of data which need be stored or transmitted. This framework is extensible, so compound components of a scene can be encapsulated and efficiently stored and transmitted.

By using this framework, I will describe and demonstrate how it is straightforward to integrate a variety of animation control mechanisms such as kinematics, motion control, and procedural animation, and have high level controls to blend the results together. It also facilitates experimentation with a variety of AI techniques, some ethologically based, for doing behavior-based animation. This behavior based approach to animation is much more conducive to collaboration, as one collaborator usually need only concern themselves with the behaviors they are designing and how it manipulates the character, and do not necessarily have to coordinate with other behavior designers, especially when the behaviors are complementary.

Map of the Dissertation

Chapter 2 begins by discussing a variety of ideas about constructing characters and related work in the Media Arts. I begin with examples of characters and character construction in the analog domain, and then try to summarize what insights we can carry over into the digital domain. I then discuss more technical work related to this dissertation, concentrating on the AI and CG communities, with special emphasis on hybrid systems which incorporate elements of both.

Chapter 3 discusses in detail an object-oriented framework for 3D modeling over time that facilitates building characters out of composable parts.

Chapter 4 discussed my parallel, distributed agent architecture, which addresses the issues of representing the attitudes and abilities of a character. I discuss the idea that animation is really a set of processes measuring and manipulating a model over time. I then show how by combining our object oriented modeling approach in Chapter 3 with this agent-based approach to generating behavior, we can build three-dimensional, semi-autonomous animated characters.

Chapter 5 summarizes what I've presented, and presents several course of direction for future work to both extend this work and to use it as a framework for building characters today.

Appendix A discusses in more detail an action selection network algorithm, based on work done by Maes (**Maes89**), and extensions we have made to it. This work is relevant to the work presented in Chapter 4, but since it was only one of several action selection algorithms (all the rest were much simpler), it was described here instead of there.