A Study on Hierarchical Avatar Behavior Representation and Control Technique

Jae-kyung Kim, Yoon-Chul Choy
Dept. of Computer Science, Yonsei Univ., Seoul, Korea
{ki187cm, ycchoy} @rainbow.yonsei.ac.kr

Won-Sung Sohn
Computational Design Program, School of Architecture., Carnegie Mellon Univ., USA
sohnws @u.washington.edu

Beom-Joon Cho
Dept.of Computer Engineering, Chosun Univ., Kwangju, Korea
bjcho@chosun.ac.kr

Soon-Bum Lim
Dept. of Multimedia Science Sookmyung Women's Univ., Seoul, Korea
sblim @sookmyung.ac.kr

Abstract

Avatar techniques have rapidly progressed in recent years, and will be widely applied to various applications. However, current expression and control of avatar behavior lacks structured and standardized method which makes it difficult to express it. The paper proposes hierarchical approach for representation and control techniques for avatar behavior for simpler avatar control in various domains. We suggest three layered architecture: task-level behavior, high-level motion, and primitive motion. The task-level behavior represents task-oriented avatar behaviors used in various task domains such as cyber class, virtual shopping mall, etc., so that end user can control the avatar through easy and simple task-level user interface. High-level motion provides abstract and parameterized actions. It is independent from task domains and implementation environments such as avatar engines. The primitive motion describes avatar motions supported by each avatar engines or implementation tools. Thus, the user controls avatar at task-level layer and does not need to be concern about low-level animation data. The task-level behavior is translated to primitive motion thru high-level motion, so that the behavior can be applied to various tools. Our goal is to support flexible and extensible representation and control of avatar behavior by hierarchical approach separating application domains and implementation tools.

Key words: Avatar, Script Language, Behavior

1. Introduction

With computer becoming an important part of our lives, many of our activities are achieved in virtual environment. Now, computers are not just a machine doing simple jobs but one of the interfaces to our social activities. Thus, interactive user interface techniques between human and computer which can induce interests is becoming more and more important. The avatar is a representative example of the interface techniques.

This work was supported by Ministry of Commerce, Industry and Energy

Avatar techniques have rapidly progressed over the recent years. Gartner group[1] selected the avatar application as the major communication technique among ten branch information which were paid attention to in 21st century.

With these avatar application, study on avatar behavior representation and control is actively going on. Especially, since the establishment of XML, the web standard language, avatar behavior script can be based on XML and it is possible to standardize avatar behavior. Actually, many XML-based script languages which have various purposes are being developed such as CML[3], AML[2], VHML[4], XSTEP[5] and so on.

In the paper, we define the script languages for avatar behavior control by layered approach architecture to provide easy interface to users at various application domain and implementation environments. Suggested script consists of three layered architecture: task-level behavior, high-level motion, and primitive motion. We discuss them in the next chapter in detail.

2. Related Works

2.1 Task-level Avatar Behavior

Task is a kind of avatar behavior which has specific purpose or object[9]. For instance, avatar behaviors such as 'Walk to Table' or 'Jump Here to There' is task-oriented avatar behavior because it has the purpose of going to a specific location or an object, while avatar motions such as just 'Walk' or 'Jump' is just motion itself.

Each domain uses different task behavior. For example, 'lecture', 'answer', and 'query' tasks could be used in the learning domain, and 'sell' and 'buy' tasks should be used in the virtual shopping domain.

Some task-level behavior systems have been researched such as STEVE[7], PPP[8], and Wizlow[6]. The main advantage of these task-level avatar systems is that the user does not need to control complicated avatar motions. The user just gives avatar a certain task, and the avatar performs appropriate sequence of behaviors to achieve the task.

However, most systems convert the task-level behaviors into primitive motion or low-level animation data directly which is supported by avatar engines or motion library to control the avatar behavior. Therefore, task-level behavior might be dependent to low-level motion of the specific implementation tool or engine, and efficiency in extensibility or reusability is lowered.

2.2 High-level Motion Control

It is difficult for common users to control the motion of avatar by providing low-level motion such as rotation angle to every part of body. This kind of job is for professional animators and not for common users. Therefore, the common users should be abstracted from low-level animation data or physical representation of avatar motion.

High-level motion allows the users to control avatar at abstract motion level. Some preceded works such as AML, CML, VHML, STEP have been researched. These scripts are based on XML format and are aimed to control avatar motions independent from specific implementation environment.

The purpose and scope of the scripts are various. In case of AML, it calls static low-level animation data such MPEG4 and represents it by motion parameters. Another script, CML, defines base motion elements such as turn-to or move-to to bridge the gap between avatar tools and engines. STEP is based on dynamic logic, which provides semantics for complex behavioral patterns, and VHML is designed to accommodate the various aspects of human computer interaction with regards to facial animation, text to speech production, body animation, dialogue manager, emotional representation.

Recently, many researches about avatar motion are actively going on. These high-level motion scripts represent avatar motions well in detail but it is too complicated for users to control the avatar motion..

3. Layered Representation and Control of Avatar Behavior

Proposed method seeks high extensibility, convertibility and reusability as its goals by defining behavior expression and control language for layered animated character behavior control. Behavior expression and control language is composed of animated character task-level behavior language which users create in specific domain context, primitive motion script for expression of basic motions supported by animated character animation engine or library, and high-level language which expresses animated character behavior independently from such domain or software program. The table 1 will show characteristics of the proposed languages.

Table 1. Characteristics of the proposed languages

Script Level	Characteristics
Task-Level Behavior	・ Consists of task behaviors that are required for specific domain such as cyber education and shopping mall ・ Dependent on domain but independent from animation engine
High-Level Motion	・ Express general character motion through various parameter ・ Abstract expression and control language ・ Independent from both domain and animation engine
Primitive Motion	・ Primitive motions supported by animation engine or motion library of the specific software program ・ Physical representation of character motion parameter ・ Independent from domain, but dependent on animation engine

To interpret script language on each layer, the proposed technique uses task-level behavior translator and context-based high-level motion translator. Since we focused on script languages, the translators are briefly described here. The task-level behavior translator analyzes behavior task-level behavior script using information of domain environment and produce parameterized high-level motion sequence. The high-level motion translator converts high-level script language into primitive motion script based on the context, that is, information on the physical structure in the virtual world. In other words, the script language on each layer exists independently from each other and is produced by the proposed translator. The following section will discuss each of these script languages in detail.

3.1 Task-level Behavior

As we mentioned in previous chapter, task-level avatar behavior is to accomplish a certain task in a specific domain environment and how the actual behavior is performed differs depending on the attribute of domains.

In this paper, we define the XML DTD according to the following concepts that represent script language of task-level avatar behavior which can be used in various domain environments.

Design Concept

▪ Simplicity: Easy to write by human scripter.

▪ Abstraction: Completely abstracted from low-level concepts

▪ Readability: Human-readable and also machine readable

▪ Parameterization: Variable behaviors derived from same behavior with different parameters

▪ Synchronized: Sequential and parallel control of behaviors

Table 2. Elements of task-level behavior script

Element	Description
Behavior Name	Task-level behavior name
Target Object	Object or location of task
Purpose	Purpose or result of task
Adverb	Adverb such as speed, intensity, etc
Synchronization	Parallel or sequential execution of behaviors

According to the concepts, we developed necessary elements for the task-level script which is described in table 1 and the brief overview of grammar for the script is the following:

Task-level behavior = <behavior name>[<target object>|<target location>] [<purpose>] [<adverb>]* [<synchronization>]+

3.2 High-level Motion

In the paper, high-level avatar motion is independent from both the specific application domain and implementation environment, and has abstract description for common avatar motions. Namely, the high-level motion represents and controls the avatar motion by using parameters such as speed, intensity, direction, and so on.

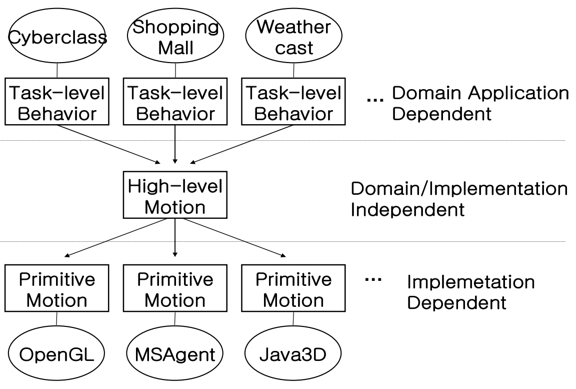


Fig. 1 Overall Architecture of Proposed System

We analyzed the parameters which are used in the existing high-level motion scripts, and defined high-level motion script. In the proposed method, for the definition of high-level motion, we analyzed the existing parameters and property in the existing high-level motion script to draw fac-tors for animated character motion expression. The high-level motion contains more than one of the motions lists element. The motion list is composed of motion elements such as motion name and parameters like space, time, intensity, and verbal elements.

First, spatial element is divided into destination property appointing target with hand and feet gestures and target property expressing character's direction. Time element is character's speed, continuance duration of motion, repetition of motion, and sequential and parallel motion. Intensity element is element that stresses or changes the intensity of the animated character motion property. Finally, verbal ele-ment is to express speech information for output of character's voice and sound effect.

These attributes represent avatar motion independent from both domain and implementation. The script takes a role which is similar to mediator between task-level behavior and primitive motion shown in fig. 1

These parameters are converted to lower hierarchy, which is dependent to implementation and express physical geometric information of virtual world. Because the high-level motion script does not express physical geometric information, context-based translator extract needed information from the rendering engine of implementation and convert the abstract parameters into physical. To extract the context, the translator traverses all nodes of cyberspace structure consisted of scenegraph, XML/HTML or etc. Detailed processing of the translator will not be discussed here.

In the proposed script shown in fig. 2, the parameters were expressed after the defined XML based W3C schema for high-level motion script language to bridge gap between task-level and primitive motion in a standardized way.

3.3 Primitive Motion

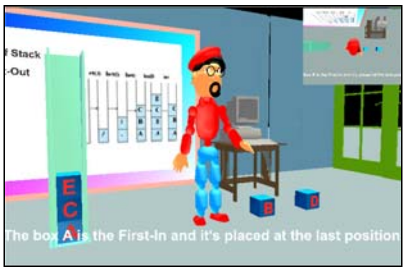
A primitive motion expresses an avatar motion which is supported by avatar engines or motion libraries. While high-level motion is ordinary and abstracts avatar motion, primitive motion is dependent to avatar control engines. Since each engine or library supports different kinds of motions, primitive motion would be in different and expressed differently depending on its environments.

Also it contains physical information for specific implementation tool compared to high-level motion. For example, a high-level motion script, "<go to="table">" should be expressed like "<walk_to x="100" y="12" z="-2">" at the primitive motion script.

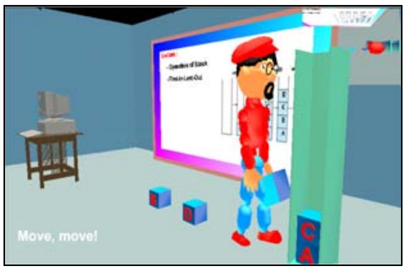
In the paper, we define primitive motion scripts that support 3D avatar motion engine based on OpenGL and MS Agent motion library. A user writes down the task-level behavior script and it is translated into the high-level motion script. It is converted to each primitive motion which belongs to corresponding avatar engine or library.

4 Implementation Results

Our world represents an ordinary classroom with some components: a lecture, a blackboard, a computer, a table, a door, walls and several lecture-related objects. In task-level script language, author command the lecturer by combination of these components and behavior. After the example script is translated to a high-level script language by the suggested formal translator, the high-level script will be loaded to the system and converted to primitive motions and the animated character performs its tasks in fig. 2(a). As we mentioned before, the primitive motion contains the physical information of the system, the same task-level script can be properly applied even though physical structure of the virtual world is reorganized as shown in fig. 2(b).


Stack
-Out
The box A is the First-In and it's placed at the last position

(a) before repositioning objects



(b) after repositioning objects

Fig. 2. The animated character is teaching an algorithm in cyber classroom

Moreover, the script language can be applied to completely different implementation tool or environments. For instance, the same task-level script is loaded to two applications as shown in fig. 3. Because our architecture takes layered approach, application tools and script languages are explicitly decoupled. This makes easier to apply script language to various implementations environments

4. Conclusion

We suggested the three layered architecture consisted of task-level behavior, high-level motion and primitive motion to provide simple interface for avatar control at various application domains to users. Also each layer interacts independently. Task-level behavior is explicitly separated from implementations so that it could be reusable at different tools.

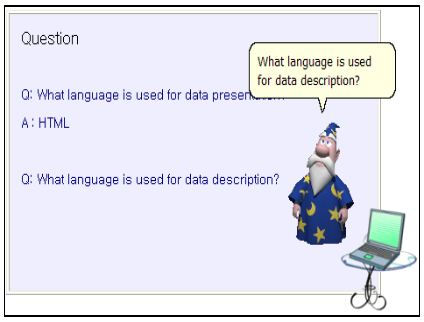

Question
What language is used for data description?
Q: What language is used for data presen
A : HTML
Q: What language is used for data description?

Fig. 3. The animated character performs his tasks in the web environments

Using this approach, avatar behavior can be controlled more easily in task-level and in high level and primitive motion, it is possible to control and express avatar motion with great reusability and extensibility which does not depend on implementation environment through abstract and physical expression.

In future works, an intuitive graphical user interface for the input of avatar tasks, and avatar motion controls based on avatar-object interaction technique are required for providing more efficient interface to users.

References

1. Gartner Group, http://www.gartner.com.
2. Kshirsagar S. et al, “Avatar Markup Language,” Proc. Eurographics, EGVE 2002, pp.169-177, 2002
3. Yasmine Arafa, Abe Mamdani, “Scripting embodied agents behaviour with CML: character markup language,” Proc. IUI, pp.313-316, 2003
4. Marriott, A. & Stallo, J., “VHML- Uncertainties and Problems... A discussion." Proc AAMAS 2002, Bologna, Italy., 2002
5. Zhisheng Huang, et al, “Implementation of a scripting language for VRML/X3D-based embodied agents,” Proc web technology, pp.91-100, 2003
6. James C.Lester, et al, “Explanatory Lifelike Avatars,” Autonomous Agents., 1999
7. Jeff Ricket, et al, “Task-Oriented Collaboration with Embodied Agents in Virtual Worlds,” Embodied Conversational Agents, MIT Press., 2002
8. Andre Elisabeth, et al, “WebPersona : A Life-like Presentation Agent for the WWW,” International Multimedia Conference, 1998
9. Thalmann D., “Autonomy and Task-level Control for Virtual Actors,” Programming and Computer Software, No4., 1995