Appear in Proceeding of the 2004 IEEE International Conference on Multimedia and Expo, 2004

An Extensible Scripting Language for Interactive Animation in a Speech-Enabled Virtual Environment

Tsai-Yen Li, Mao-Yung Liao, and Chun-Feng Liao
Computer Science Department, National Chengchi University
64, Sec.2, Chih-Nan Road, Taipei, Taiwan 11605, ROC
{li, g9105, g9104}@cs.nccu.edu.tw

Abstract

Character animations on most virtual environment systems are canned motions created off-line through motion capture techniques. The motions are then encoded and transmitted with a fixed format and played at the client side. In this paper, we have proposed an XML-based scripting language, called eXtensible Animation Markup Language (XAML), to describe interactive dialog-based animations. The language is designed to describe character animations at various command levels and to compose a new animation from existing animation clips. In addition, the language is extended to incorporate other dialog-based scripting language such as VoiceXML. We have implemented such a system in Java that can interpret the language and render 3D animations based on the user's interactive voice commands.

1. Introduction

3D contents are becoming prevalent in our daily life. However, despite the rich information that 3D contents can deliver, the adaptation of 3D applications is slower than we expected. The animations that we can see on the web now are mostly pieces of canned motions generated by motion capture techniques or keyframe-based software. The cost of designing effective 3D animations, which usually relies on the tedious work of skillful animators, is still too high. Therefore, how to make the production of interesting interactive animations more cost effective becomes an important issue.

A powerful animation scripting language is one of the key requirements for effective animation production. A general 3D animation scripting language should be flexible enough to accommodate animation specifications at different levels. In addition, it should be extensible to incorporate other specifications in a pluggable way. However, we are not aware of any general animation scripting language that can achieve these two goals. In this paper, we have proposed such a scripting language, called eXtensible Animation Markup Language (XAML), based on XML for general-purpose animation specification. We will show how animations can be specified and composed at various command levels. In addition, we will use a simplified version of VoiceXML as an example to show how a dialog-based language can be plugged into the markup language to make interactive animation possible.

2. Related Work

The design of an animation scripting language was a popular topic in the early days when animation software is not popular. Most animations today are produced by authoring software (such as 3DS MAX or MAYA) and motion capture software. However, the outputs of this type of software are mostly low-level animation specifications such as a list of keyframes.

Using high-level scripts to control animation characters is not a new idea. Since Improv [9] started to realize the concept by proposing a layered model, many researches have been devoted in this direction. Avatar Markup Language (AML) [7] and Scripting Technology for Embodied Persona (STEP) [6] are two examples focusing on describing the animations of humanoid characters. The design of AML focuses on synchronization of facial expression and voice as well as the richness of expressing an avatar's emotion. However, it lacks the function to extract or modify an existing animation. STEP is a scripting language for humanoid animations with its strength on integrating logic reasoning with computer animation.

Several researches have considered the problem of integrating voice dialog and graphical animation in order to develop anthropomorphic spoken dialog agents [2][3]. For example, [3] defines a language called MPML-VR to model interactive animation with dialogs using Javascript and VRML. The system proposed in [2] enhances the VoiceXML language, a W3C standard, by extending it to include special tags for animation scripting.

3. The XMAL Scripting Language

Traditionally, two design paradigms are used by computer animators: low-level specification and high-level control. Low-level specification can be used to flexibly express animation parameters and allows ani-

```
<AnimItem DEF ="WaveWalk" cycle="2000">
  <AnimImport src="Walk">
  <AnimItem DEF="SimpleWave" cycle="1000">
      <AnimNode target="r_shoulder">
         <OrientationInterpolator
             key ="…" keyValue="…" />
       </AnimNode>
  <AnimItem>
</AnimItem>
```

Figure 1. An example of XAML animation script

mators to have full control of details. However, the process of specifying the details may become too tedious and time consuming. In contrast, animators can also specify what they would like to see through high-level commands. The animation is stored as a canned motion, and the flexibility of modifying such motions is rather low. If the motion is generated on-line, the animators may not be able to predict its details. In addition, the computation complexity of implementing such high-level commands in real time remains a great challenge. Generally speaking, different levels of animation specification have their own pro and con. The best way would be to allow all different specifications to co-exist and be used according to their availability and characteristics.

3.1. XAML language grammar and structure

XAML is an XML-based scripting language for extensible animations. An example of this language is shown in Figure 1. <AnimItem> is the most basic grouping tag for an animation component. Each animation component uses attributes and contents of the XML tag to specify the details of an animation component. Attributes are designed to manage controls such as play time, number of loops, animation speed, and etc. Content part of the tag contains the body of the animation, which might recursively include other animation components with the <AnimImport> tag or low-level parameters with the <AnimNode> tag.

The syntax of the XAML language is shown in Figure 2. We have classified the specifications of animations into three levels: *high*, *medium*, and *low*. In the low level specification, the target field is assigned the object of the transformation. For example, it could be a Node name (e.g. joint name) defined in the H-Anim specification [5]. In addition, we use the keyframe mapping mechanism that includes the <OrientationInterpolation> and <PositionInterpolation> tags with key and key value pairs, similar to the animation specification of X3D [4]. In high-level specification, the system is able to accept high-level commands with certain language pattern such as "Walk to café". The verb

```
<AnimItem> := (<AnimItem> | <AnimHigh> | <AnimImport> |
   <AnimNode> | <AnimTransition> | <AnimPlugin>)+
<AnimHigh> := text
<AnimTransition> := (<AnimItem> | <AnimImport> | <Anim-
   Transition>)+
<AnimNode> := (<OrientationInterpolator> |
   <PositionInterpolator>)+
<AnimPlugin> := <xaml-v> | other-plugins
```

Figure 2. The grammar of the XAML language

"Walk" and the object "café" must be known terms in the database in order to obtain their properties.

In mid-level specification, the system currently supports two types of tags: <AnimImport> and <AnimTransition>. The <AnimImport> tag, a type of <AnimItem>, is used to import an existing animation component from an external source. In the attributes of this tag, one can specify only a portion of the animation tree for composing the final animation. The <AnimTransition> tag, which is also a type of <AnimItem>, is designed to provide smooth transition between two consecutive <AnimItem> components by automatically patching appropriate interpolation.

3.2. Time Control Model of the XAML language

We use the method of constraint propagation, a common method for temporal reasoning, to design the time control model in XAML [1]. In XAML, an <AnimItem> tag contains a mode attribute of temporal relation, sequential or parallel, for the children nodes. Motion components can be played sequentially (seq) or in parallel (par). Two parallel components will have the same start time while the start time of next component will succeed the end time of the previous one.

For each animation component, we can use time-related attributes, such as start, end, and cycle, to specify the time interval for the animation. An example is shown in Figure 3. The start and end attributes define the interval of the animation while the cycle attribute defines the loop time. The start time of a component, default to 0, is a value relative to the start time of its parent node. It can also be absolute time if specified explicitly. However, all children nodes must satisfy the constraint set by the ancestor nodes. For example, if the end time of a child node is later than its parent, its animation will end with its parent. Under this time control model, the thirteen relations between two intervals, summarized in [8], can then be specified.

3.3. The XAML-V Plug-in Extension

In addition to be able to accommodate different levels of animation components, XAML can be extended to consider plug-in components which are unknown to

```
<AnimItem mode="seq">
   <AnimItem DEF="A" start="5"
       end="1000"> …
   </AnimItem>
   <AnimItem DEF="B" start="50"> …
   </AnimItem>
</AnimItem>
```

Figure 3. An example of time-related attributes.

the designer of an XAML system. The tag for this purpose is called <AnimPlugin>. The contents inside this tag can be customized functions that are not part of this language. Instead, the system will pass the subtree below this tag to its plug-in handler. For example, in this paper, we will describe such an extension, called XAML-V (XAML-Voice Extension), to allow interactive animation and voice dialogs based on the VoiceXML standard. This extension starts with the <xaml-v> tags, and its syntax is very similar to the VoiceXML language. In fact, this extension is a subset of the VoiceXML language focusing on voice dialog management and some customized tags for graphical animation. The system currently support form-based dialog management and allows a user to communicate with the system via its Text-To-Speech (TTS) and Automatic Speech Recognition (ASR) functions. During a dialog, the system can embed animation scripts in the XAML format inside an <animation> tag. These scripts, at form or field levels, are sent back to the animation management system for display.

4. System Design and Implementation

We have design an authoring and display system to play the animation scripts in the XAML language. The system uses Java3D as the underlining rendering library and VRML as the geometric format. The overall system architecture is shown in Figure 4. The system consists of four parts: *XAML parser*, *Animation Manager*, *Resource Store*, and *3D Renderer*, as describe in more details below.

4.1. XAML Parser

The parser translates all animation scripts in XAML into low-level tree structures consisting of *AnimUnit* and *AnimData* nodes. The AnimUnit nodes, which are the leaf nodes, contain the transformation for objects while the AnimData nodes, which are internal nodes, contain other data such as time. An AnimUnit node must be managed by an AnimData node, and an AnimData node may contain other AnimData nodes.

When a XAML node is extracted from an XML tree, the geometric model of the animation subject is retrieved by the *Animation Factory*. The animation factory then dispatches the node to appropriate handling

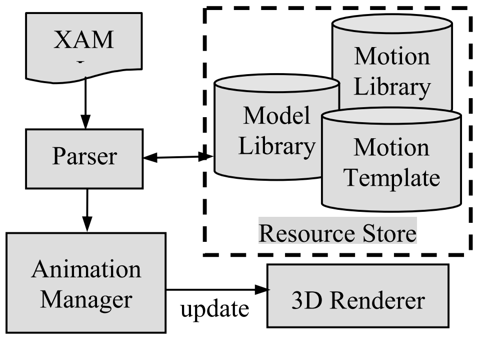

Figure 4. Overview of system architecture

routines to create their corresponding internal representation. For example, AnimNode and AnimItem nodes can be directly translated into AnimUnit and AnimData. When an AnimHigh node is encountered, the system will retrieve a low-level structure to realize the high-level command from a motion template database. The AnimImport node needs an import processor to retrieve the XAML tree for the animation component from an external source such as a motion library. The AnimTransition node, on the other hand, needs to generate necessary data for transition between two nodes. Finally, when an AnimPlugin node is encountered, the type of plug-in is determined first and then the subtree under the node is passed to the handling routine at a later time when the node is triggered.

4.2. Animation Manager

The output of the parser is the animation components represented as internal tree structures. The root node of such a tree represents an independent animation. The animation manager keeps a list of root nodes for all independent animations and contains a timer triggered periodically. When an update starts, each root AnimData node will be examined to see if the animation is valid for the current time. If yes, the fraction for the current time will be set to its associated AnimUnit node. Subsequently, all AnimData nodes below the current node will be processed recursively.

As described in previous sections, the display time of an animation component will be affected by the constraints set by its ancestors. For example, whenever a parent node starts a new loop, the relative timing of all its successor nodes need to be reset accordingly. All the constraints from the ancestor nodes are propagated into their successor nodes at run time.

5. Experimental Results

In Figure 5, we show an example of interactive animation with voice dialog written in XAML with XAML-V extension. The scenario is that a virtual

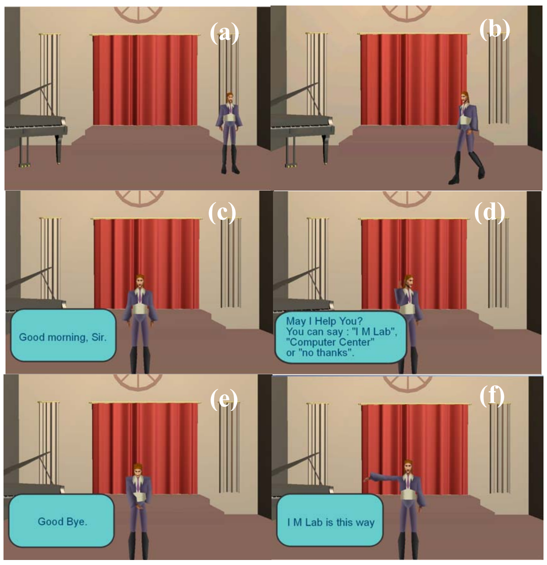

Figure 5. An example of animation script with interactive voice dialog

```
1 <!-- initial script -->
2 <?xml version="1.0"?>
3 <AnimItem DEF="Service" playMode="seq">
4   <AnimHigh>Walk to user.location</AnimHigh>
5    <AnimPlugin>
6     <xaml-v version="1.0">
7      <form id="helloForm">
8       <block>Good morning, Sir.</block>
9       <field name="helpType">
10         <prompt>May I Help You? You can say :
"I M Lab", "Computer Center" or "no
thanks".</prompt>
11         <animation><!--Field Level Anim-->
12            <AnimImport src="listen" />
13         </animation>
14       </field>
15       <block>
16         <submit next="helpFormResponse.jsp"
namelist="helpType"/>
17       </block>
18      </form>
19    </xaml-v>
20   </AnimPlugin>
21 </AnimItem>
```

Figure 6. Animation script for the example in Figure 5.

character is assigned the job of greeting other users via voice dialog when they enter the virtual environment. When the user enters the space, the receptionist, originally standing aside (Figure 5(a)), starts an animation script with dialog. The receptionist will walk to the front of the user (5(b)) and greet the user by saying "Good morning, Sir" and the destinations that he knows (5(c)). Then he will listen to the user's input for the destination that he/she is interested in (5(d)). If the user does not need any assistance, the receptionist will end the dialog by saying "Good-Bye" (5(e)). If the user designates one of the destinations, he will guide the user to the destination (5(f)).

The XAML scripts for this script are organized in several files and some of them are generated dynamically. The main script, shown in Figure 6, is composed of several animation components such as high-level command of "WalkTo user.location" and a XAML-V dialog enclosed in the <AnimPlugin> tag (lines 5-20). The dialog consists of a form with a prompt and the corresponding animation. Note that the animation enclosed in an <animation> tag (lines 11-13) in XAML-V will be send to the animation manager. The response from the user is collected through a voice recognition module and submitted to a server-side program. When the server receives a response, it can generate the successive dialog from a file or on-the-fly according to the answer.

6. Conclusions

We have proposed an extensible animation scripting language and used voice dialogs as an example to illustrate its extensibility. The format of XML is inherently extensible as long as the customized tags can be understood by both sides of the communication. The language can accommodate animation specifications at various levels through multiple means. By adding the plugin function, we can further extend the language to consider other sets of XML-based languages such as VoiceXML. We have also implemented an interpreter and an authoring program to demonstrate the power of the language.

7. References

[1] J.F. Allen, "Time and time again: The many ways to represent time," *Int'l J. of Intelligent Systems*, 6, 4, pp. 341-356, July 1991.

[2] B. Carpenter et al, A portable, server-side dialog framework for VoiceXML. *Pro. of Int'l Conf on Spoken Language Processing (ICSLP) 2002.*

[3] S. Descamps, H. Prendinger, and M. Ishizuka, "A multimodal presentation mark-up language for enhanced affective presentation," *Proc. of the Int'l Conf. on Intelligent Multimedia and Distant Education (ICIMADE-01)*, pp. 9–16, 2001.

[4] Extensible 3D(X3D), http://www.web3d.org

[5] H-Anim, http://www.h-anim.org

[6] Z. Huang, Anton Eliens, and Cees Visser, "STEP: A Scripting Language for Embodied Agents," *Pro. of the Workshop on Lifelike Animated Agents*, 2002.

[7] S. Kshirsagar, A.Guye-Vuilleme, and K. Kamyab, "Avatar Markup Language," *Proc. of 8th Eurographics Workshop on Virtual Environments*, pp.169-177., 2002.

[8] P. Ladkin, and R. Maddux, "Representation with Convex Time Intervals," TR KES.U.88.2 Kestrel Institute, Palo Alto, CA, 1988.

[9] K. Perlin, and A. Goldberg, "Improv: A System for Scripting Interactive Characters in Virtual Worlds," *Proc. of SIGGRAPH 96, ACM Press*, pp. 205-216,