

**ViSiCAST Deliverable D4-2:  
SiGML Notation-Avatar Software Driver**

<b>Project Number:</b>	IST-1999-10500
<b>Project Title:</b>	ViSiCAST Virtual Signing: Capture, Animation, Storage and Transmission
<b>Deliverable Type:</b>	Int
<b>Deliverable Number:</b>	D4-2
<b>Contractual Date of Delivery:</b>	December 2001
<b>Actual Date of Delivery:</b>	December 2001
<b>Title of Deliverable:</b>	SiGML Notation-Avatar Software Driver
<b>Work-Package contributing to the Deliverable:</b>	Workpackage 4 (Animation and Modelling)
<b>Nature of the Deliverable:</b>	PR (Prototype)
<b>Author(s):</b>	Ralph Elliott, John Glauert, Richard Kennaway, Kevin Parsons (UEA)

**Abstract:**

This document describes prototype synthetic deaf signing animation software. This software effectively forms the second half of the English-text-to-synthetic-signing pipeline described in the opening chapter of ViSiCAST deliverable D5-1. It takes as input a description of a sequence of signing gestures expressed in SiGML, from which it generates a corresponding stream of avatar animation parameters, which it then uses to drive a signing avatar on-screen.

# 1 Introduction

This deliverable consists of prototype software which generates synthetic deaf signing via a signing avatar, given a description of the required sequence of signing gestures expressed in the SiGML notation. The present covering document briefly describes this software and its role in the ViSiCAST project.

A major objective of ViSiCAST is the development of a prototype text-to-signing system, which takes (English) natural language text and translates this text into authentic sign language, represented in SiGML, which is then used to drive a computer-generated avatar, or virtual human, thus enabling the content of the original text to be presented to deaf users in their preferred language. The function and architecture of this text-to-signing system are described in some detail in Chapter 1 of deliverable D5-1, *Interface Definitions*, where the system architecture is presented as a pipeline of processing stages. Responsibility for the implementation of this pipeline is shared between the project's two core technology workpackages, namely WP5, *Language and Notation*, and WP4, *Animation and Modelling*. WP5 is responsible for the first half of the processing pipeline, and WP4 for the second half. The interface between the two halves is SiGML, the *Signing Gesture Markup Language* developed within the project, whose first version was defined in deliverable D5-2. SiGML is an XML application based on *HamNoSys*, the Hamburg Notation System, a well-established notation for representing sign languages at (approximately) the phonological level. More specifically, SiGML is based on HamNoSys version 4, which was developed in an earlier phase of the project, and which is described in deliverable D5-1. Although the text-to-signing pipeline provides the immediate context for the present deliverable, it should be emphasised that the SiGML-to-signing module for which the deliverable provides the initial prototype is intended to provide a general vehicle for applications requiring on-screen signing.

The next section (Section 2) describes the software of this deliverable in more detail. The last section (Section 3) describes the packaging and deployment of the software.

## 2 Overview of D4-2 Software

The software for this deliverable consists of two modules, the core animation engine and a collection of integration ("glue") components, which together result in a single package which displays the appropriate avatar and (repeatedly) allows the user to specify a SiGML input, from which a stream of animation parameters is then generated and fed to the avatar for presentation to the user. (The generated animation parameter stream is also saved in a file for future use.) In this section we describe these two modules in more detail.

### 2.1 *Animgen*

A prototype implementation of the first half of the text-to-signing pipeline was supplied as part of deliverable D5-3. Thus the function of the second half of the pipeline, as represented in the present deliverable, is to provide software to animate the set of example SiGML sequences generated by the D5-3 system. The core of the deliverable is a software module called *animgen*, a synthetic animation engine which converts a SiGML input stream into a stream of avatar animation parameters. These animation parameters may take either of two forms, corresponding to the two distinct target avatar technologies of interest. These are: the proprietary avatar technology supplied by Televirtual Ltd. (one of the ViSiCAST partners), and the VRML-based H-Anim avatar.

The first of these is ViSiCAST's primary signing avatar technology: it is more advanced than H-Anim, giving both high visual quality and high performance when supported by appropriate GPU (graphics processing unit) hardware. In this technology the avatar's appearance is defined by a complex seamless deformable mesh constructed from a large number of small textured polygons. The physical configuration of the mesh is dynamically controlled by the configuration of an underlying "skeleton", a set of bones (and muscles) organised, on anatomical lines, as a tree-structured hierarchy. Hence, a pose of the avatar is determined by specifying the spatial position and orientation of each bone/muscle (and, in the case of a muscle, also its length). So, a time-stamped sequence of animation parameter sets of this kind is sufficient, in principle, to drive the avatar.

This technology is realised as a collection of data files, defining the skeleton, mesh, and other details of a particular virtual human, together with the software modules needed to "drive" an animation of the model. The technology was first developed in the context of a system driven by data originating from motion-capture devices. Towards the end of the first year of the project (2000), Televirtual made this avatar technology available to partners in the form of an ActiveX Control, suitable for deployment in various software contexts, in particular in a WWW-browser, and including the data files defining the visual features of a project-specific avatar called *Visia*. Although still intended primarily to support applications driven by motion capture data, this software provides a rudimentary interface allowing the configuration of the *Visia* avatar to be set, as just described, in accordance with directly supplied skeleton configuration data, a *Bones Animation Format (BAF)* frame. The BAF animation data stream format was first used within ViSiCAST as the transmission format (with compression) for the first deliverable of the project's TV/Broadcast workpackage (WP1, deliverable D1-1, *Direct-Sign*). The *Visia* avatar, thus packaged as an ActiveX Control formed a major component of deliverable D2-1, *ViSiCAST Avatar Controls*.

For development purposes, however, it often proves more convenient to use *animgen* in conjunction with the alternative, H-Anim-based, target. In particular, as part of the *animgen* development effort a simple "stick-man" avatar has been defined. This stick-man is H-Anim compliant, although it also incorporates movable eyebrows (albeit in a crude form), in order to support the very restricted set of non-manual SiGML features used in the D5-3 examples. This avatar is thus similar to Televirtual's *Visia* in as much as each static pose is determined by the configuration of a hierarchically structured skeleton. Hence, animation of this avatar is defined in a manner very similar to that described above for *Visia*, that is, by a time-stamped sequence of skeleton configuration parameter sets — although for the H-Anim avatar these data streams take the form of VRML data-structures suitable for use by the VRML animation functions.

The main advantages of using the stick-man avatar as target in a development context are that animation data streams for this avatar are somewhat more compact, and quicker to generate, than for *Visia*, and the resulting animations, while obviously less visually appealing, are sometimes easier to inspect and to evaluate from a technical (as opposed to an aesthetic) perspective. A further significant factor is that the VRML animation model has well-defined timing properties, whereas the rudimentary interface mentioned above, which is the only currently available means of driving *Visia* from a BAF-stream, merely provides a method allowing a "frame" (i.e. a static pose in the sequence) to be passed to the avatar for rendering, but does not provide any means of controlling (or indeed of monitoring) the precise time at which the requested rendering is actually accomplished. Hence, while it is perfectly capable of giving an approximate indication of behaviour, the currently available version of *Visia* is not a suitable vehicle for proper evaluation of synthetically generated animation. However, the enhancement of the *Visia* ActiveX Control to include a BAF-frame rendering interface with well-defined timing properties is a priority for the project, and is expected to be available very shortly, allowing its incorporation in milestone M5-11 which marks the integration of the two halves of the text-to-signing pipeline (scheduled for achievement by the end of February 2002).

As stated at the start of this section, the development of *animgen* for the present deliverable has focussed on support for those SiGML features used in the set of examples generated by the D5-3 natural language processing software. This prototype version of *animgen* thus does not yet implement all features of SiGML, and in particular, its support for non-manual features of SiGML is restricted to the very small subset (eyebrow movements) required for the D5-3 examples. Also, a small range of manual features are not yet supported. For a full list of features not currently supported the reader is referred to Appendix B. Although the version of SiGML supported is essentially that delivered in D5-2, a small number of minor modifications have been made in the light of subsequent experience with features introduced with version 4 of HamNoSys. The current version of the SiGML DTD, incorporating these modifications, may be found in Appendix A.

The objective of synthesising animations for all the D5-3 sentences has almost but not quite been met. The set consists of a sequence of 21 sentences, containing a total of 81 signs. Of these 81 signs, *animgen* is unable to synthesise anything more useful than a “NULL” marker for the following:

```
glass green plate.
```

In addition, a further 5 signs out of the 81 contain some feature which *animgen* does not support, leading to partially correct synthesis for each of these:

```
big soup_bowl teaspoon large small.
```

*Animgen* is currently implemented in Perl. In consequence it is relatively inefficient: on a high-performance PC (with a processor speed in excess of 1.5GHz), it generates BAF frames at a rate of at least 15 fps, the rate generally taken within the project as being the minimum acceptable for adequate quality of signing animation). For the H-Anim stick-man target, the frame rate improves by a factor of at least 2. However, we intend to recode the current Perl implementation, which contains several obvious Perl-dependent inefficiencies, in a compiled language such as C++ or Java. We expect this to improve performance by a factor of at 3 or 4 at worst, and more probably by an order of magnitude. For technical details of *animgen*'s implementation of the mapping from SiGML documents to streams of animation parameters, the reader is referred to Appendix E, which contain copies of a paper presented at GW2001, the most recent Gesture Workshop (London, April 2001).

## 2.2 Integration (“Glue”) Components

The integration software consists of a collection of COM components which connect the relevant sub-stages of the pipeline and provide a simple graphical user interface to the signing avatar. These components support the construction of simple applications (e.g. in VB) or HTML pages (with JavaScript) which present animations of signing sequences generated by *animgen*, using either the VRML stick-man or Visia. The standard input format for *animgen* is SiGML, but using the HamNoSys-to-SiGML converter included with deliverable D5-3, these components also allow HNST (HamNoSys token sequences) as an acceptable alternative input format. In practice, as the standard output format for the English-to-BSL software developed at UEA for D5-3 is HNST, this latter is often the most convenient input format for the animation software described here.

The architecture and function of this integration software is described in more detail in a separate report, included here as Appendix D.

## 3 Packaging and Deployment of D4-2 Software

An installer is supplied for the delivered software (which runs on PCs under Windows XP/2000/98). This installer assumes (and checks for) prior installations of the following supporting software

components: Java JRE or JDK, v1.3 (needed by the HamNoSys-to-SiGML translation software), ActivePerl, v5.6 (needed by *animgen*), Cortona VRML Player/Plugin, v3.1 (needed to play VRML-format animations), ViSiCAST Avatar Controls (needed for the Visia avatar which plays BAF-format animations). The README text file supplied with the installer provides the necessary details about the installation process. The installer provides two ways for the end user to access the software, corresponding to the two supported animation formats described in the previous section:

- An HTML page which allows the user to generate and play VRML-format animations using the stick-man figure described in the previous section.
- A stand-alone application which provides analogous facilities for BAF-format animations using the project's Visia avatar.<sup>1</sup>

For the purposes of the present deliverable and its evaluation we regard the first of these two as the primary ("reference") implementation, due to the absence of BAF-frame synchronisation features in the current Visia avatar used by the second, as described in Section 2.1. Nevertheless, we include the Visia-based application in the deliverable distribution as well, in order to demonstrate the effectiveness of the integration software, and as a preliminary guide to what may be expected in the near future.

The installer inserts *Start* menu items allowing the user to invoke either of these configurations of the software. In either case the user may specify the folder containing data files, and may then chose any one of three input formats:

- HNST (HamNoSys tokens);
- SiGML;
- Avatar Animation Parameters (BAF or VRML).

In each case the *Files* panel displays a list of available files in the given format. If a file is specified using this panel, a click on the *Play* button will cause the animation for the specified file to be played by the avatar. For HNST or SiGML input, before playing, the animation data is first generated using the HamNoSys-to-SiGML converter, and *animgen*.<sup>2</sup> In both cases, the frame rate for the generated animation data may be specified using the combo box, and the resulting data stream is saved in a file which is fed as input to the avatar player software; this file is also available for subsequent use (as may be seen by switching to BAF/VRML input format).<sup>3</sup>

The VRML stick-man avatar comes with a simple set of GUI features providing information to the user, and allowing the user a useful measure of interactive control of the animation. The user documentation for these features may be found in Appendix C.

The installed implementation folder contains a handful of example files, giving animation sequences for: the full set of D5-3 examples, a much shorter sequence of three sentences from this set, the first sentence ever handled by the text-to-signing software ("I take the mug."), and a simple sequence of eyebrow manipulations. Obviously, the user is free to add further SiGML (or HNST) files to this collection. Note that when a sign contains features not currently supported by *animgen*, that sign is replaced either by a "NULL" place-holder, or by a partially correct sign, in the final animation.

---

<sup>1</sup> An HTML page for BAF-format animations using Visia, analogous the that for VRML-format animations, is also available, but for simplicity's sake is not included in this distribution.

<sup>2</sup> The unwary user is warned that the generation of the animation parameters, especially in BAF format, takes a noticeable amount of time. (See Section 2.1 for more details.)

<sup>3</sup> As indicated in Appendix D, the integration software is capable of streaming BAF animation data directly to the Visia avatar via an internal cache without using an intermediate file; this facility is used in the initial prototype of the fully integrated pipeline (Milestone 5-11, but not included in the present deliverable).

Further information about *animgen*'s treatment of unsupported SiGML features, and a list of signs in the D5-3 example set containing such features was given in Section 2.2 above. *Animgen* maintains a logfile, `PerlWrapperLog.txt` (in the implementation folder), which the interested user may consult for details of its progress.

## References to Other ViSiCAST Deliverables

D2-1 *Internet Browser Plugin* (March 2001)

D5-1 *Interface Definitions* (March 2001)

D5-2 *SiGML Definition* (May 2001)

D5-3 *Prototype Text to Sign Notation* (August 2001)

## Appendices

A SiGML DTD at 2001-12

B Status of AnimGen's Implementation of SiGML Features

C *Animgen* VRML GUI Widgets

D Report: *HamNoSys to Animation Components* (Kevin Parsons)

E Report: *Synthetic Animation of Deaf Signing Gestures* (Richard Kennaway)