Classifying malware into families using N-grams

PATRICK RAND and REYNIER ORTIZ
School of Computing and Information Sciences
Florida International University
Miami, Florida, USA

Abstract–We present HapGL (ver. *beta*), a javascript library to generate expressions on 3D virtual agents based on the Facial Action Coding System, a Text-To-Speech synthesizer web-service, and a lip-synchronization algorithm to generate audiovisual speech streams. HapGL was implemented using a client-server architecture. The client side uses webGL, ThreeJS to render the virtual characters, and makes requests to the text-to-speech synthesizer service, the server, to generate the audio stream and provide timing information of the *visemes* to be displayed. The lip-synchronization algorithm then starts the audio and synchronously displays the sequence of visemes. A smooth viseme transition was implemented to provide a more realistic virtual human.

Categories and Subject Descriptors: I.3.5 [**Artificial Intelligence**]: Natural Language Processing—*Speech recognition and synthesis*; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Animation*

Additional Key Words and Phrases: malware, classification, n-grams

1. INTRODUCTION

Some introduction *blahg blah* etc.

2. PROBLEM DEFINITION AND METHODS

2.1 Task Definition

Blah blah [WebGL 2014].

2.2 Algorithms and Methods

Blah blah [WebGL 2014].

3. EXPERIMENTAL (AND/OR THEORETICAL) EVALUATION

3.1 Methodology

Blah blah [WebGL 2014].

3.2 Results

Blah blah [WebGL 2014].

3.3 Discussion

Blah blah [WebGL 2014].

Phonemes and visemes are also important concepts when developing a lip-synchronization algorithm. A phoneme is a basic unit of a language's phonology, which is combined with other phonemes to form meaningful units such as words or morphemes. The phoneme can be described as "The smallest contrastive linguistic unit which may bring about a change of meaning" [Cruttenden 2008]. In this way the difference in meaning between the English words kill and kiss is a result of the exchange of the phoneme /l/ for the phoneme /s/. The English language uses a rather large set of 13 to 21 vowel phonemes, including diphthongs, although its 22 to 26 consonants are close to average.

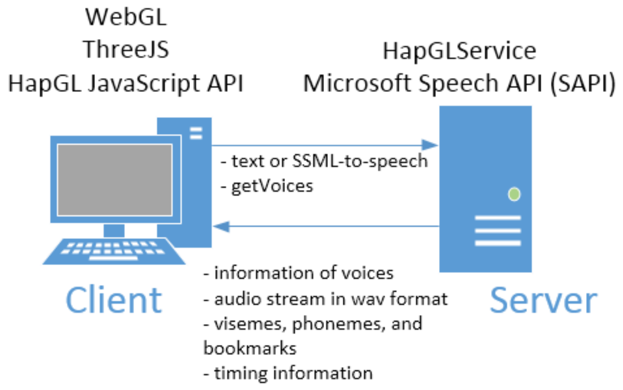


Fig. 1. HapGL architecture.

A viseme is any of several speech sounds which looks the same, for example when lip reading [Fisher 1968]. The mapping between phonemes and visemes is not one-to-one as many phonemes have the same visual appearance when speaking, therefore several phonemes may share a common viseme.

4. RELATED WORK

Blah blah[WebGL 2014].

The current HapGLService exposes three operations:

```
string GetVoices();

string SpeakText(string text, string voice,
                 string audioFormat);

string SpeakSSML(string ssml, string voice,
                 string audioFormat);
```

The GetVoices() operation returns a list of the SAPI voices installed in the server, the information of each voice contains the name, gender, age and culture. Both SpeakText() and SpeakSSML() operations returns a structure with:

—The audio stream in wav or mp3 format in a base64 string
—The sequence of visemes, where each viseme contains the viseme number, the audio position in milliseconds, and the duration in milliseconds

The difference between SpeakText() and SpeakSSML() is that the first one only accepts a plain input string and synthesizes using the default options, whereas SpeakSSML() accepts a string in SSML. Voice manipulation is specified in SSML by using the `<prosody>` elements and specifying parameters such as: volume,

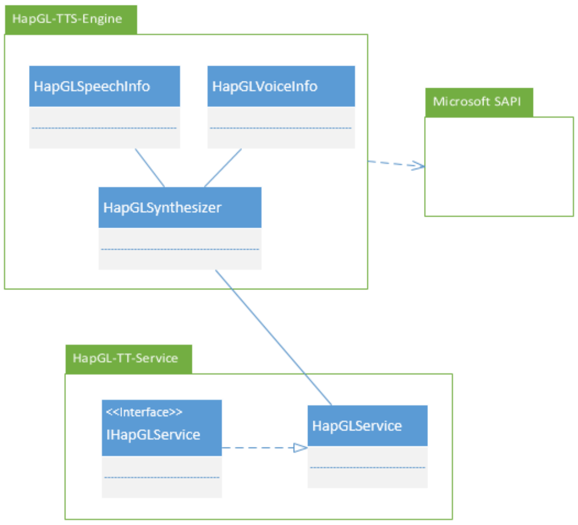


Fig. 2. HapGLService package diagram.

rate and pitch [SSML 2014]. The information returned by these operations are sufficient for the lip-synchronization algorithm to generate the sequence of viseme transitions aligned with the audio stream. Fig 2 shows the package diagram for the HapGLService subsystem.

HapGL requires to instantiate a `HapGL()` which expects the URL of the TTS, the mesh of the 3D character and the mesh of the hair, for example:

```
// Using ThreeJS, load character and
// hair into mesh1 and mesh2 respectively

var hapgl = HapGL.init({
        ttsUrl: 'http://localhost:88/',
        character: mesh1,
        hair: mesh2
});

// Example to activate an Action Unit
hapgl.setAU('AU1', 100);
```

4.0.1 *Activating EmFACS emotions with HapGL.* Generating emotions in HapGL was done also in the same manner as in HapFACS. Emotion FACS (EmFACS) introduces a mapping of subsets of action units to universal emotion identified by Ekman [P. Ekman and Freisen 1983] namely fear, anger, surprise, disgust, sadness, and happines. The emotions implemented in HapGL were:

—Happines, combining AU6, AU12, and AU25
—Sadness, combining AU1, AU4, and AU15
—Surprise, combining AU1, AU2, AU26, and AU5
—Anger, combining AU4, AU5, AU7, AU23, and AU24
—Disgust, combining AU9, AU15, and AU16

The current version of HapGL provides three methods related to the speaking portion:

—`getVoices(function getVoicesCallback)`
—`speak(String text, String voice)`
—`speakssml(String ssml, String voice)`

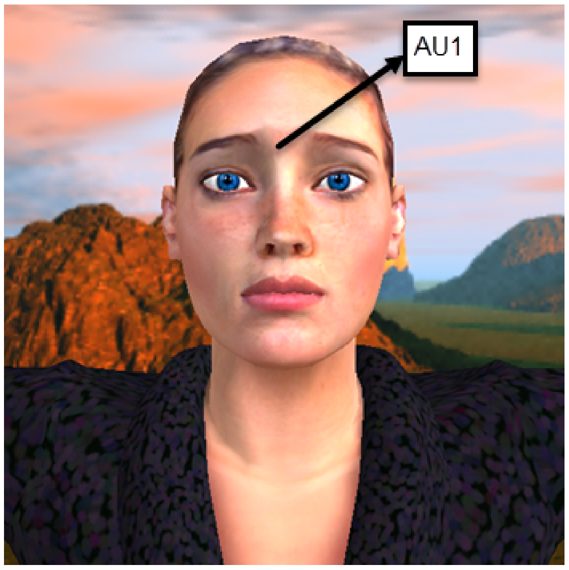


Fig. 3. Result of setting AU1 to 100 intensity.

`getVoices(function getVoicesCallback)` returns a JSON object with an array of voices and will immediately call the `getVoicesCallback` function with the result. Since the HapGLService uses Microsoft Speech API (SAPI), the voices are SAPI-compatible installed on the web-server. More sophisticated voices can be purchased and installed separately. The following is an example of the result of calling `getVoices`:

```
hapgl.getVoices(function(output) {...});

// Example of voices returned
{
    voices : [{
        id : "MS-Anna-1033-20-DSK",
        age : "Adult",
        name : "Microsoft Anna",
        gender : "Female",
        culture : "en-US"
    }]
}
```

`speak` and `speakssml` are similar, the only difference is that `speak` will only accept a plain string, whereas `speakssml` can accept a string in SSML format as input [SSML 2014]. The `voice` parameter corresponds to the *name* attribute of the voices returned by `getVoices`, if no `voice` is passed in then the HapGLService will synthesize using the default voice in the TTS Server. The *ssml* string argument for the `speakssml()` function should be a well-formed SSML Version 1.0 [SSML 2014]. SSML allows to manipulate the voices by modifying parameters such as: *volume*, *rate*, and *pitch* in the `<prosody>` elements. Several `<prosody>` elements can be combined to produce the desired pronunciation of sentences.

The output of `speak(..)` and `speakssml(..)` contains the necessary information to render the sequence of viseme transitions synchronized with the audio stream to produce a realistic talking virtual human. The following example shows the output when speaking the word "Hello".

```
hapgl.speak(''Hello'');
```

```
// The output of speaking ‘‘Hello”
{
    audioFormat: ‘‘data:audio\/wav;base64”,
    audioStream: ‘‘...”,
    visemes: [{
       number : 0,    // silence
       audioPosition : 0.0,
       duration : 3.0,
       emphasis : 0
    }
}
```

The sequence of visemes is already sorted in the correct order to be animated. All this information is sufficient for the lip-synchronization algorithm to render each viseme at the correct time. The viseme transitions are done smoothly, otherwise, just displaying the viseme in its maximum intensity would create an undesired illusion. Each viseme transition takes a pair of visemes V_0 and V_1, where V_0 is the starting viseme and V_1 is the ending viseme. To do the transformation $V_0 \rightarrow V_1$ we consider the duration d_0 of V_0. In d_0 time, V_0 "fades out" and V_1 "fades in". By "fade out" we mean interpolating from V_0 by setting the value of the corresponding morph gradually from 1 to 0 in d_0 time. Conversely, "fade in" means interpolating to V_1 by setting the value from 0 to 1. Each viseme maps to a corresponding phoneme, we use the mapping provided by the Microsoft Speech API as seen in Table I.

Table I. Viseme to phonemes mapping in Microsoft Speech API

Viseme	Phoneme(s)	Viseme	Phoneme(s)
0	silence	11	ay
1	ae, ax, ah	12	h
2	aa	13	r
3	ao	14	l
4	ey, eh, uh	15	s, z
5	er	16	sh, ch, jh, zh
6	y, iy, ih, ix	17	th, dh
7	w, uw	18	f, v
8	ow	19	d, t, n
9	aw	20	k, g, ng
10	oy	21	p, b, m

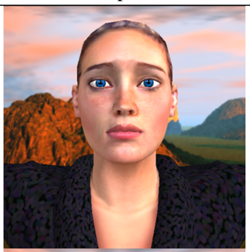
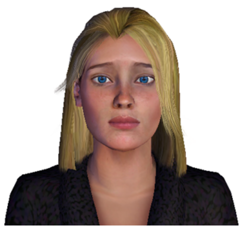
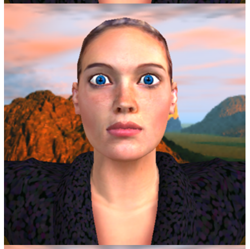
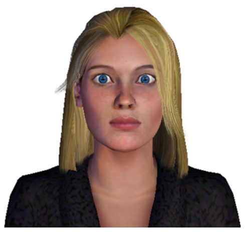
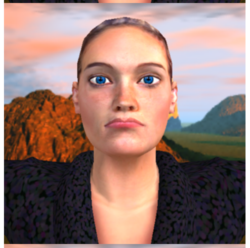
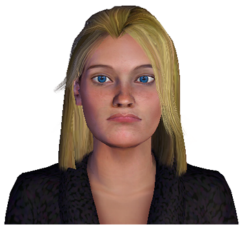
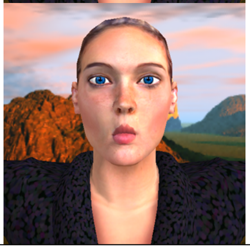
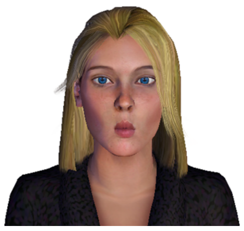
Since not all the phonemes have its corresponding Haptek morph register, we choose the most similar morph. We used the following viseme to morph mapping in HapGL:

```
var visemeMorphMapping = {
   ‘0’: {name: ‘neutral’}, ‘11’: {name: ‘aa’},
   ‘1’: { name: ‘aa’ }, ‘12’: { name: ‘ih’ },
   ‘2’: { name: ‘aa’ }, ‘13’: { name: ‘n’ },
   ‘3’: { name: ‘aa’ }, ‘14’: { name: ‘n’ },
   ‘4’: { name: ‘ey’ }, ‘15’: { name: ‘s’ },
   ‘5’: { name: ‘er’ }, ‘16’: { name: ‘ch’ },
   ‘6’: { name: ‘ih’ }, ‘17’: { name: ‘th’ },
   ‘7’: { name: ‘uw’ }, ‘18’: { name: ‘f’ },
   ‘8’: { name: ‘ow’ }, ‘19’: { name: ‘d’ },
   ‘9’: { name: ‘aa’ }, ‘20’: { name: ‘g’ },
  ‘10’: { name: ‘ow’ }, ‘21’: { name: ‘m’ }
};
```

Table II. AUs with recognition rate of less than 100%. Taken from HapFACS [Amini and Lisetti 2013].

AU	Recognition Rate	AU	Recognition Rate
10	66.67%	16	33.33%
11	66.67%	20	66.67%
12	66.67%	23	33.33%
13	66.67%	25	33.33%
14	66.67%	28	33.33%

Table III. AUs comparisson between HapGL and HapFACS.

AU	HapGL	HapFACS
AU1		
AU5		
AU17		
AU18		

To evaluate the lip-synchronization mechanism, we used the Microsoft Speech API voices installed by default in **Windows 8.1 x64**: "David" (Male, en-US), "Hazel" (Female, en-GB), and "Zira" (Female, en-US), however, the results were identical with the three voices. We first evaluated HapGL with a set of 231 words, and for each viseme we recorded the time position in milliseconds where it was displayed and compared to the audio position returned by the HapGLService and the average difference was $\approx 10ms$, which is acceptable because "appropriate A/V sync limits have been established and the range that is considered acceptable for film is $+/-22ms$. The range for video, according to the ATSC, is up to $15ms$ lead time and about $45ms$ lag time" [Kudrle 2011]. This is a complete list of the randomly-generated words used int this test:

Adult, Aeroplane

A similar test was done but with sentences instead. We tested 12 sentences and also obtained the same result of $\approx 10ms$. Below is the complete list of sentences:

—*"The quick, brown fox, jumps over the lazy dog."*

Then, we chose 4 paragraphs and also achieved the same result of $\approx 10ms$ difference between the audio position where the viseme was displayed and the actual time. Although this is also a satisfactory result, for long texts the HapGLService takes several seconds to synthesize the entire input, and more than 2 to 3 *seconds* might be unacceptable on most real-time dialog systems. The following example is of one the paragraphs used in this part of the evaluation:

"I have this fear. It causes my legs to shake.." [YourDictionary 2014]

Similar tests were performed using the default SAPI voice "Microsoft Anna" installed in **Windows 7 x64** and **Windows Server 2008**, and there were noticeable differences between the audio and visemes, for example, in words like *well* and *cry*. Therefore, for production systems we would recommend to install the HapGLService in **Windows Server 2012** as the default SAPI voices are more natural and the timing information is more accurate, or to purchase more sophisticated third-party voices compatible with **Windows 7** or **Windows Server 2008**.

5. CONCLUSION

Nevertheless, HapGL is still far from being used in production systems as it lacks of other necessary functionalities which could be part of future works. To name some, and not intended to be a comprehensive list, we suggest the following:

—*Detailed Evaluation*, due to time constraints, a thorough evaluation could not be performed. We suggest to measure the *believability* of the system by surveying a random sample of users, preferably greater than 20. Although this is a subjective measure, a user study is still a good indication about the quality of the system.

REFERENCES

Reza Amini and Christine Lisetti. 2013. HapFACS: an Open Source API/-Software to Generate FACS-Based Expressions for ECAs Animation and for Corpus Generation. (2013). http://ascl.cis.fiu.edu/hapfacs.html

Robert Anderson, Bjorn Stenger, Vincent Wan, and Roberto Cipolla. 2013. Expressive Visual Text-to-Speech Using Active Appearance Models. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '13)*. IEEE Computer Society, Washington, DC, USA, 3382–3389. DOI:http://dx.doi.org/10.1109/CVPR.2013.434

Jonas Beskow and Mikael Nordenberg. 2005. Data-driven synthesis of expressive visual speech using an MPEG-4 talking head. In *Proc. Interspeech 2005*. 793–796.

Jonas Beskow and Kalin Stefanov. 2013. Web-enabled 3D talking avatars based on WebGL and HTML5. (2013).

Alan Cruttenden. 2008. *Gimson's pronunciation of English* (7 ed.).

Mr. Doob. 2013. Three.js. (2013). https://github.com/mrdoob/three.js,2013

Tonny Ezzat and Tomaso Poggio. 1998. Visual Speech Synthesis by Morphing Visemes. (1998).

C. G. Fisher. 1968. Confusions among visually perceived consonants. (1968).

T. Di Giacomo, S. Garchery, and N. Magnenat-Thalmann. 2007. *Expressive Visual Speech Generation*. Springer, Ulrich Neumann, Zhigang Deng (eds.).

Sara Kudrle. 2011. Fingerprinting for Solving A/V Synchronization Issues within Broadcast Environments. (Jul 2011).

N. Magnenat-Thalmann. 2007. *Chapter 2: Expressive Visual Speech Generation*. Vol. 978-1-84628-906-4. in Zhigang Deng and Ulrich Neumann eds., Springer Press, 29–59.

R. W. Levenson P. Ekman and W. V. Freisen. 1983. Autonomic Nervous System Activity Distinguishes among Emotions. 221 (1983).

C. Shaw. 2002. Haptek. (2002). http://www.haptek.com

SSML. 2014. Speech Synthesis Markup Language (SSML) Version 1.0. (Nov. 2014). http://http://www.w3.org/TR/speech-synthesis/

WebGL. 2014. Getting Started - WebGL Public Wiki. (2014). https://www.khronos.org/webgl/wiki/Getting_Started

YourDictionary. 2014. Narrative Essay Examples. (2014). http://examples.yourdictionary.com/narrative-essay-examples.html