FACE AS A MULTIMEDIA OBJECT

Ali Arya, Steve DiPaola

School of Interactive Arts and Technology, Simon Fraser University,
2400 Central City, 10153 King George Highway, Surrey, BC, Canada V3T2W1
E-mail: {aarya, sdipaola}@sfu.ca

ABSTRACT

This paper proposes the Face Multimedia Object (FMO), and *iFACE* as a framework for implementing the face object within multimedia systems. FMO encapsulates all the functionality and data required for face animation. *iFACE* implements FMO and provides necessary interfaces for a variety of applications in order to access FMO services.

1. INTRODUCTION

Psychological studies have shown that babies, only nine minutes after birth, are drawn to faces, although they can barely focus their eyes [6]. This fascination with faces continues throughout our lives. With its expressive capabilities, face is the window to our thoughts and emotions, and is perceived uniquely by a dedicated part of human brain [6]. In light of the important role of faces in visual communication, MPEG-4 standard has provided special Face Definition and Animation Parameters, in order to facilitate creating and transmitting facial views in multimedia presentations [4]. This helps describe and animate faces within a multimedia stream, independent of background and other media objects. Simple definition of geometry and desired animation movements, on the other hand, is far from a comprehensive framework for using a "face multimedia object".

Traditional multimedia objects are of audio and video types, usually considered as one concrete piece, i.e. not divided into smaller objects. Modern multimedia presentations and standards now allow media objects of higher complexity to be used, i.e. different audio and video objects forming a new "combined type". Familiar examples (or possible candidates) can be separate foreground and background objects in a video stream, speech as a special case of audio data that corresponds to a text to be spoken by a specific individual, and a "music video" as a special media object made of two parallel audio and video objects. Each one of these examples illustrates the need for (and convenience of) a higher level of abstraction on top of traditional media objects when used in a certain type of application. Such layer of abstraction encapsulates related data, isolates the user from unnecessary details, enforces certain relation between low-level components, and provides a unique access mechanism (similar to a class hierarchy in object-oriented programming languages).

Rapid of growth of visual communication systems, from video phones to virtual agents in games and web services, has a brought a new generation of multimedia systems that we refer to as face-centric. Such systems are mainly concerned with multimedia representation of facial activities. Examples can be video messaging on cell phones and online customer support agents. Needless to say, each one of these applications has its own domain-specific algorithms, data, and control mechanism, but they all share the same front end that has to support a well-defined set of face-related capabilities. The concept of a "face multimedia object" arises from the need to encapsulate all the common requirements of such applications into one single component.

In this paper, we introduce Interactive Face Animation – Comprehensive Environment (*iFACE*) as a framework for Face Multimedia Object (FMO). Just like any other object, FMO includes data objects of less complicated types (audio, video, etc), and needs to provide proper services and interfaces to access and process those objects. *iFACE* integrates:

- Hierarchical 3D head model for controlling facial actions from vertex to feature-group levels
- 2D image transformations for direct image-based animation
- Text-to-speech audio generation and lip-sync
- Structured content description language
- Streaming components
- Wrapper objects to simplify development
- Behavioural modeling
- Proper interfaces to access the underlying objects from a variety of client application

In Section 2, we briefly review some related work in the area of face animation. Basic concepts and the structure of *iFACE* are the topics of Sections 3 and 4.

Implementation notes and some concluding remarks are presented in Sections 5 and 6.

2. RELATED WORK

Facial Action Coding System (FACS) was one of the first comprehensive studies of facial actions [3]. MPEG-4 Face Animation Parameters (FAP) [4] later used a similar approach to define a set of codes to control a facial animation. 3D head models [4] and 2D image-based methods [1] have been used in order to implement these facial actions, usually in the form of simple components (suitable for a specific clients such as web pages), or stand-alone face players (see [1]).

Valentine [8] was one of the earliest researchers to introduce the concept of a face space. In both 3D and 2D methods, such a space is mainly a "flat" parameter space without dynamically accessible layers of abstraction and details [2,7]. Multi-resolution meshes [5] can reduce the number of parameters but do not change the way we look at the face space, e.g. feature view vs. vertex/pixel view. A true multi-layer space should allow working with face from higher levels like regions and their related high-level actions, to low-level points and their movements, depending on the application needs. MPEG-4 Group nodes for Face Definition Parameters (FDP) are introduced to support such head models.

Although some dedicated languages for facial animation have been developed in the last few years (see [1] for a quick review), most of them lack a comprehensive set of required features such as MPEG-4 compatibility, support for external events, and temporal relation between facial actions.

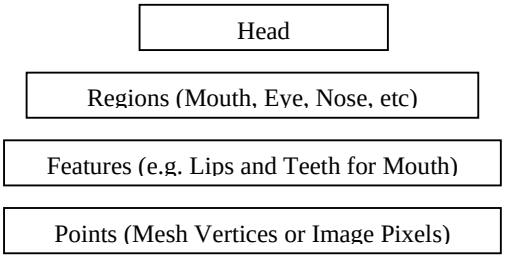

Figure 1. Hierarchical Head Model

3. FACE MULTIMEDIA OBJECT

Face Multimedia Object (FMO) encapsulates all the functionality and data related to facial actions. It forms a layer of abstraction on top of all the underlying details to simplify and streamline the creation of facial animations, including video, audio, timing and streaming-related data, control signals and events, and possibly descriptions (textual, metadata, etc).

From a multimedia system point of view, in addition to obvious application-dependent features such as Realism, we consider the following requirements for FMO:

- Hierarchical Geometry: Face animation is dynamic manipulation of geometry, regardless of how we define it (e.g. pixels of a 2D image or vertices in a 3D model). The important attribute of this geometry is meaningful relations and grouping of its elements to form facial features and regions. Each one of these groups provides certain amount of detail and allow certain type of manipulation. For instance, a vertex can be moved to any new location by itself, but moving a feature (e.g. opening mouth) will move all related vertices. FMO has to expose proper layers of geometry for different types of access, mainly:
  - Vertex-level (or pixel-level)
  - Feature-level
  - Region-level
- Timeliness: Face is a time-based object. Facial actions need to be synchronized (parallel, sequential, etc) with each other and external events.
- Expressiveness: FMO should be able to express a variety of characters by personalizing the geometry and performing speech, movements, and emotions.
- Interactivity: Just like any other software object, FMO needs proper interfaces to expose its services to a different types of client, such as:
  - GUI applications
  - HTML-based web pages
  - Web service clients
- Behavioural Model: Actions of an agent (similar to people) is mainly based on stimulus-response model. In simplest case, behavioural rules, stored as individual "knowledge", determine the proper response to any stimulus. But in a more general context, "personality" and "mood" are also affecting factors on behaviour. An ideal behavioural modeling for FMO supports defining and dynamically changing all these three factors.

4. *iFACE* ARCHITECTURE

4.1. Basic Structure

Interactive Face Animation – Comprehensive Environment (*iFACE*) is our proposed architecture for implementing a face multimedia object. The basic structure of this framework is shown in Figure 2. Each layer exposes its own interfaces for access to encapsulated objects. Together, these interfaces form the *iFACE* Application Programming Interface (FaceAPI).

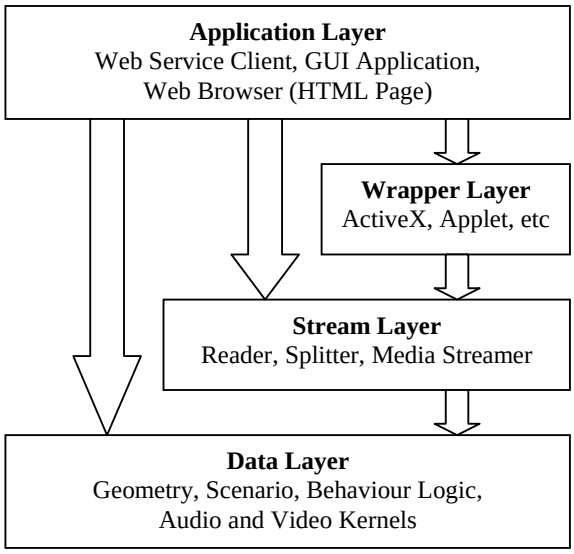

Figure 2. Basic Structure of *iFACE*

iFACE Data Layer is responsible for implementing main functional FMO components. This includes:

- Geometric Model: *iFACE* geometry can be defined by 3D models or 2D images, and includes similar region and feature layers in both cases. At the lowest level, features and regions are associated to vertices or image pixels.
- Scenario: Timeline of events and action in face animation is described using Face Modeling Language (FML) [1]. The language parser breaks the requested facial actions to smaller moves, depending on the number of frames.
- Video Kernel: *iFACE* uses texture-mapped 3D head models or 2D image transformations [1] in order to create video frames corresponding to desired facial actions.
- Audio Kernel: A text-to-speech (TTS) system is used to generate audio data based on the text to be spoken (embedded in FML input) and also provide lip-sync information for video kernel (number of video frames as a function of speech length). Also, Audio Kernel processes an input audio data to extract lip-sync info and suggest facial expressions. This component relies on existing software tools such as MBROLA and OnLive.
- Behavioural Logic: The behaviour of face agent depends primarily on the input scenario. In more dynamic situation, this input may serve merely as an external event, and FMO behavioural logic will decide on proper responses. This logic relies on:
  - Knowledge (global data and interaction rules)
  - Personality (long term characteristics)
  - Mood (short term personal characteristics)

  These can be defined in the modeling part of FML document and also dynamically change through interactions. This part of *iFACE* is still at conceptual/experimental phase.

Most of multimedia applications require audio and video data to be used as a multimedia stream with proper synchronization. Stream Layer has components for reading an input script, splitting audio and video paths, generating frames, and finally mixing them, all in a streaming mechanism (i.e. receiving and producing frames of data with proper timing). These components rely on the Data Layer components and isolate them from the higher-level applications that need streaming.

Using Data and Stream Layer components may be hard and complicated, especially for web-based applications where only simple scripts and HTML commands are available. In order to simplify application development in such cases, *iFACE* includes Wrapper components that perform all the tasks and provide simple API with commands such as LoadFMLFile and PlayMedia.

4.2. Head and Face Geometry

iFACE head model uses an abstracted hierarchy consisting of Head as the top object, Regions, Feature Lines and Points, and Physical Points, as shown in Table 1. The features include all the feature points defined by MPEG-4 FDPs in addition to extra points and also line objects.

Region	Feature Types
Face	Cheek, Forehead, Chin (points)
Eye	Brow, Lid, Iris, Pupil (lines and points)
Nose	Corner, Tip, etc (lines and points)
Mouth	Lip, Tooth, Tongue (lines and points)
Ear	Top, Bottom, etc (points)
Hair	Top, Center, Corner, etc (points)

Table 1. Head Regions and Features

Head, Regions, and Features form a higher layer of abstraction on top of the physical model which can be a 3D mesh or one or more 2D images. The model-making tool assigns Regions and Features to certain vertices or pixels. Abstract objects allow model-independent control of animation by exposing proper interfaces for different animation movements and also model modification (resizing, reshaping, etc).

Each abstracted object also belongs to one or more "transform group" that defines interdependencies of facial areas (e.g. in movements). It is also possible to define a HeadBase region that includes all regions except Eye, Nose, and Mouth to simplify some animations.

4.3. Face Modeling Language

Face Modeling Language (FML) [1] is a Structured Content Description mechanism based on eXtensible Markup Language (XML). The main ideas behind FML are:

- Hierarchical representation of face animation (from frames to simple moves, to meaningful actions, and finally stories)
- Timeline definition of the relation between facial actions and external events (parallel and sequential actions, and also choice of one action from a set based on an external event)
- Defining capabilities, behavioural templates, and models (FML is independent of the type of model but provides means of defining it.)
- Compatibility with MPEG-4 (MPEG-4 FAPs are supported explicitly, and FDPs implicitly by general-purpose model definition mechanisms.)
- Compatibility with XML and related web technologies

Following is a sample FML document that defines an external event in the **<model>** part and uses that to choose fromtwo possible actions, in **<story>.**

```
<model><event id="user" val="-1" /></model>
<story><ser>
       <talk>Hello </talk>
       <excl ev_id="user">
              <talk ev_val="0">Ali</talk>
              <talk ev_val="1">Steve</talk>
       </excl></ser></story>
```

Figure 3. Sample Output Images

5. IMPLEMENTATION

iFACE Data Layer components use Microsoft Direct3D, DirectSound, and .NET framework to allow interfacing through web services and other distributed components. Stream Layer components are built based on DirectShow technology in order to use the built-in streaming functionality. They are DirectShow media filters that encapsulate Data Layer objects in streaming situations.

The default Wrapper component is an ActiveX called *iFACEPlayer* that wraps all DirectShow filters required for typical streaming scenarios, i.e. reading and parsing input stream, activating audio and video frame generators and mixing and transmitting final stream. .NET web controls and Java applets can also be used for compatibility purposes.

A simple JavaScript code to use *iFACEPlayer* on a web page is shown below. Typical visual outputs (using 3D and 2D methods) are shown in Figure 3.

```
function onPageLoad() {
   iFPlayer.InputFile = "SampleFml.xml";
   iFPlayer.CreateMedia();
   iFPlayer.Play();}
function onUserChange() {
   SFPlayer.SetEvent(USER,User.selected);}
```

6. CONCLUSION

The growing demand for "face-centric" multimedia applications makes it necessary to define and develop a Face Multimedia Object (FMO) that encapsulates all the media types and required services for facial animation. This paper proposes *iFACE* framework for a face multimedia object. *iFACE* integrates proper objects to implement all FMO functionality, and also interfaces for a variety of application types.*iFACE* objects are developed in three main layers: Data (for core functionality), Stream (for streaming purposes), and Wrapper (for ease of use).

7. REFERENCES

[1] Arya, A., and Hamidzadeh, B., "ShowFace: A Framework for Personalized Face Animation," *IEEE/EURASIP Intl Workshop RichMedia*, 2003.

[2] DiPaola, S., "FaceSpace: A Facial Spatial-Domain Toolkit", Proceedings of Sixth International Conference on Information Visualisation, 2002.

[3] Ekman, P., and W.V. Friesen, *Facial Action Coding System*, Consulting Psychologists Press Inc., 1978.

[4] Lee, W. S., et al., "MPEG-4 Compatible Faces from Orthogonal Photos," *IEEE Conf Computer Animation*, 1999.

[5] Luebke, D.P., "A Developer's Survey of Polygonal Simplification Algorithms," *IEEE Computer Graphics & Applications*, May, 2001.

[6] McNeill, D., *The Face*, Little-Brown, 1998.

[7] Parke, F.I., "Parameterized models for facial animation revisited," *ACM SIGGRAPH Facial Animation Tutorial Notes*, 1989.

[8] Valentine, T., "A Unified Account of the Effects of Distinctiveness, Inversion and Race in Face Recognition," *Quarterly Journal of Experimental Psychology*, 43A, 161-204.