

EFFICIENT MODELING OF VIRTUAL HUMANS IN MPEG-4

Tolga K. Capin¹, Eric Petajan², Joern Ostermann³

¹Computer Graphics Laboratory
Swiss Federal Institute of
Technology (EPFL)
1015 Lausanne, Switzerland
capin@lig.di.epfl.ch

²face2face animation, inc.
2 Kent Place Blvd.
Summit, NJ 07901
eric@f2f-inc.com

³ AT&T Labs - Research
Rm 3-231
100 Schultz Dr
Red Bank, NJ 07701
osterman@research.att.com

ABSTRACT

This and the accompanying paper present an overview of the face and body animation object in the MPEG-4 Version 2 standard. The MPEG-4 standard includes the representation, compression of virtual human models and their interface with other MPEG-4 objects, with bitrate requirements as low as 1 Kbit/second. In this paper we give an overview of geometrical modeling techniques for the FBA object, and in the accompanying paper we describe the animation techniques.

1. INTRODUCTION

The MPEG-4 version 2 standard, published by ISO in January 2000, aims to provide an integrated set of tools for compressing and streaming multimedia objects [4][5]. MPEG-4 is an object-based standard that contains media objects combined to form a scene [3]. MPEG-4 standardizes a number of such media objects, capable of representing both natural and synthetic content types, which can be either 2- or 3-dimensional. In addition to the elementary media objects such as natural audio and video, MPEG-4 defines the coded representation of objects such as: text and 2D/3D graphics, synthetic sound, and finally talking synthetic faces and moving bodies with associated text used to synthesize the speech and animate the face. In this paper, we review the Face and Body Animation (FBA) object in the MPEG-4 Version 2 standard.

FBA has a wide area of applications, such as e-commerce, games, virtual teleconferencing, virtual kiosks and call centers. FBA aims to provide tools for model-based coding of video sequences containing human faces and bodies. Instead of representing the faces and bodies as coded 2D images, the FBA object assumes a synthetic model, which can be defined and animated by specific FBA parameters. These parameters are typically extracted or synthetically generated, coded and transmitted. MPEG-4 specifies a rich set of FBA parameters to define the model, texture and animation of the face and body. There are two sets of parameters. The first set specifies the definition of the FBA model: FDPs (Face Definition Parameters) and BDPs (Body Definition Parameters). These parameters allow the decoder to create an FBA model with specified shape and texture. The second set defines the animation of the face and body: FAPs (Face Animation Parameters) and BAPs (Body Animation Parameters). Typically, FDPs and BDPs are

transmitted only once, while the FAPs and BAPs are transmitted each frame. MPEG-4 defines 68 FAPs and 186 BAPs to define the state of the FBA object in one frame during animation. Each FAP/BAP has a special meaning defining one degree of freedom.

2. FBA OBJECT IN MPEG-4

The goal for defining the FBA object is to specify sufficient parameters for animating both realistic and cartoon-like humanoid characters. There is no constraint on the complexity of the models: the models can be realistic representations of real persons, as well as very simple cartoon-like models. To achieve this goal, MPEG-4 defines a large set of parameters to animate the FBA object in real time, and the applications can select subsets of these parameters to animate less complex models.

The MPEG-4 FBA specification defines the syntax of the bitstream and the decoders' behaviour. The way of generating the models (e.g. 3D interactive modeling programs, vision based modeling, template-based modeling) is not specified by the standard, the encoders are free to choose the method depending on their application. The standard defines a scalable coding scheme, and the encoder can choose among coding parameters to achieve a selected bitrate and model quality.

The specification of the FBA object is distributed into two separate but related parts of the MPEG-4 specification: Systems (Part 1) and Visual (Part 2). The Systems part specifies the representation and coding of the geometry and the method to deform the surface of face and body, i.e. FDP/BDP parameters. The system part also defines the integration of FBA nodes with other AV objects in the same scene. The Visual part specifies the coding of animation parameters, i.e. FAP/BAP parameters. This paper discusses the systems part, the accompanying paper discusses the visual part [1].

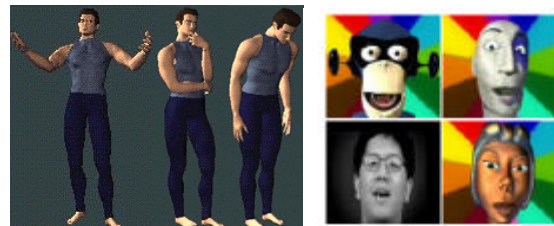


Figure 1: Face and Body Animation in MPEG-4

2.1 Default Posture and Expression

As the first step, MPEG-4 defines a generic face and body model in its neutral state. The default body has the following properties (see Figure 2):

- Standing posture, gaze is in direction of Z-axis.
- The feet point to the front direction.
- The two arms are placed on the side of the body with the palm of the hands facing inward.
- The hands point in direction of -Y-axis, except the thumb, which has 45 degrees inclination.

Similarly, the default face has the following properties:

- Gaze is in direction of Z-axis.
- All face muscles are relaxed.
- Eyelids are tangent to the iris.
- The pupil is one third of the diameter of the iris.
- Lips are in contact; the line of the lips is horizontal and at the same height of lip corners.
- The mouth is closed and the upper teeth touch the lower ones.
- The tongue is flat, horizontal with the tip of tongue touching the boundary between upper and lower teeth.

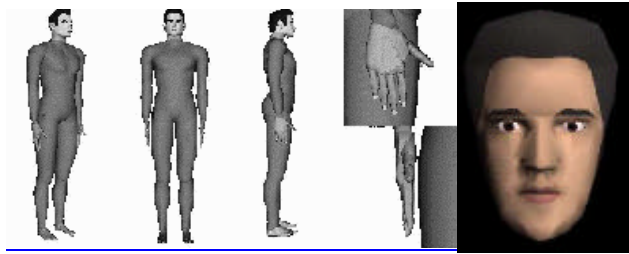


Figure 2: Default face and body posture for the FBA Object

2.2 FDPs and BDPs

FDP/BDP sets are used to define a new FBA model, using a scene graph in MPEG-4 BIFS. BIFS is a binary representation of scenes in MPEG-4. Based on the ISO VRML standard [7], BIFS defines coding techniques for each node and its fields. FBA defines new nodes, in addition to standard VRML nodes. These nodes are *Body*, *BDP*, *BAP*, *BodyDefTable*, *Face*, *FDP*, *FAP*, *FaceDefTable*, *FaceDefMesh*, *FaceDefTransform*.

The FBA scene graph has a *Body* node as its root. The *Body* node contains three children, *BDP*, *BAP* and *renderedBody*. The *BDP* node encapsulates the downloaded geometry of the body, together with the skeleton joint hierarchy, surface geometry and the deformation tables. The *BDP* child of the *Body* node is optional. If it is null, the FBA decoder has to use its default FBA model. This requires that each FBA decoder has its own default model. The *BAP* node contains the 296 BAPs as its fields, and the FBA decoder stores these values in these fields at each frame. The *renderedBody* is the root of the scenegraph which is animated at each frame and passed to the compositor for rendering. The *BDP* node has two children: *bodySceneGraph* and a *bodyDefTable* list. The *bodySceneGraph* contains a hierarchy of body joints, together with the attached surface and

texture nodes. The *bodySceneGraph* specification is based on the VRML 97 H-Anim 1.1 specification for a standard representation of body geometry [2].

The *Face* node is a descendent of the *bodySceneGraph* node. Similar to the *BDP* node, the *FDP* node is optional. The *faceSceneGraph* node contains the scene graph of the face geometry. The *FAP* field of the *Face* node contains the decoded FAPs.

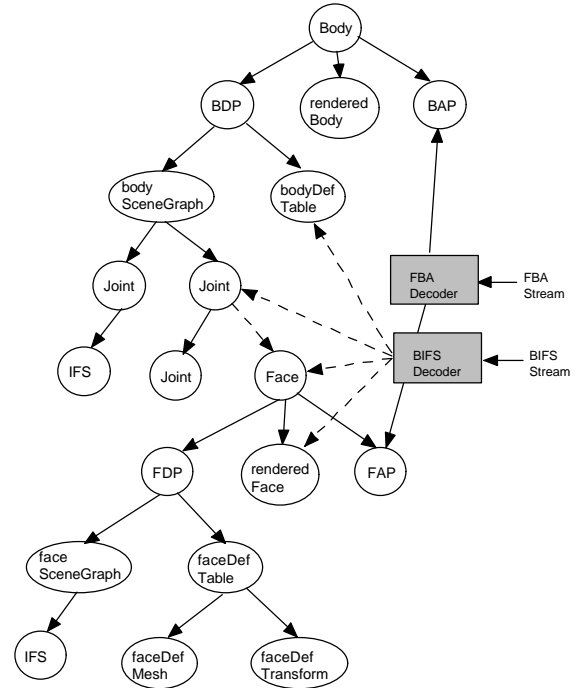


Figure 3: The FBA Scene Graph

2.2.1 Animation Rules using FaceDefTables

A *FaceDefTable* defines how a model is deformed as a function of the amplitude of the *FAP*. It specifies, for a *FAP*, which *Transform* nodes and which vertices of an *IndexedFaceSet* node are animated by it and how. *FaceDefTables* are considered to be part of the face model.

Animation Definition for a Transform Node: If a *FAP* causes solely a transformation like rotation, translation or scale, a *Transform* node can describe this animation. By means of a *FaceDefTransform* node, the *FaceDefTable* specifies the type of transformation and a neutral factor for the chosen transformation. During animation, the received value for the *FAP* and the neutral factor determine the actual value.

Animation Definition for an IndexedFaceSet Node: If a *FAP* like smile causes flexible deformations of the face model, the animation results in updating vertex positions of the affected *IndexedFaceSet* nodes. The affected vertices move along piecewise linear trajectories that approximate flexible deformations of a face. A vertex moves along its trajectory as the amplitude of the *FAP* varies. By means of a *FaceDefMesh* node, the *FaceDefTable* defines for each affected vertex its own piece-

wise linear trajectory by specifying intervals of the FAP amplitude and 3D displacements for each interval.

If P_m is the position of the m^{th} vertex in the IndexedFaceSet in neutral state (FAP = 0), P'_m the position of the same vertex after animation with the given FAP and $D_{m,k}$ the 3D displacement in the k^{th} Interval, following algorithm must be applied to determine the new position P'_m :

1. Determine, in which of the intervals listed in the table, the received FAP is lying.
2. If the received FAP is lying in the j^{th} interval $[I_j, I_{j+1}]$ and $0=I_k \leq I_j$, the new vertex position P'_m of the m^{th} vertex of the IndexedFaceSet is given by:

$$P'_m = \text{FAP} * ((I_{k+1}-0) * D_{m,k} + (I_{k+2}-I_{k+1}) * D_{m, k+1} + \dots (I_j - I_{j-1}) * D_{m, j-1} + (\text{FAP}-I_j) * D_{m, j}) + P_m.$$
3. If $\text{FAP} > I_{\text{max}}$, then P'_m is calculated by using the equation given in 2 and setting the index $j = \text{max}-1$.
4. If the received FAP is lying in the j^{th} interval $[I_j, I_{j+1}]$ and $I_{j+1} \leq I_k=0$, the new vertex position P'_m is given by:

$$P'_m = \text{FAP} * ((I_{j+1} - \text{FAP}) * D_{m, j} + (I_{j+2} - I_{j+1}) * D_{m, j+1} + \dots (I_{k-1} - I_{k-2}) * D_{m, k-2} + (0 - I_{k-1}) * D_{m, k-1}) + P_m.$$
5. If $\text{FAP} < I_1$, then P'_m is calculated by using the equation 4 in step 4 and setting the index $j = 1$.
6. If for a given FAP and 'IndexedFaceSet' the table contains only one interval, the motion is strictly linear:

$$P'_m = \text{FAP} * \text{FAP} * D_{m1} + P_m.$$

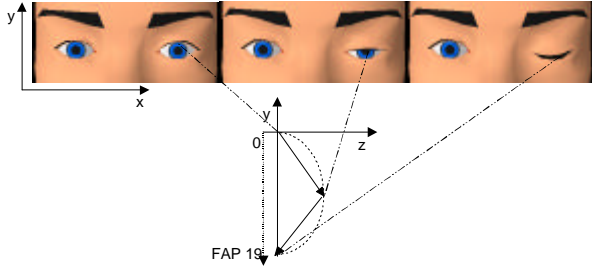


Figure 4: Neutral state of the left eye (left) and two deformed animation phases for the eye blink (FAP 19). The FAP definition defines the motion of the eyelid in negative y-direction; the FaceDefTable defines the motion of the vertices of the eyelid in x, y and z direction. For FAP 19, positive FAP values move the vertices downwards.

2.2.2 Animation Rules using BodyDefTables

The BodyDefTable node specification is an extension of the FaceDefTable node. Note that each FaceDefTable is indexed by a single FAP. This means, that either each vertex is controlled by only one FAP (i.e. each vertex appears in only one FaceDefTable), or each vertex may be controlled by more than one FAP (i.e. a vertex may appear in more than one FaceDefTable). In the latter case, the final displacement of the vertex is calculated by adding the displacements by different tables, with equal coefficients. In body animation, however, multiple BAPs typically affect the same vertex without equal coefficients (for example, the upper arm vertices may be

deformed, based elbow and shoulder BAPs). That is, it is not sufficient to add the displacements from different tables. Therefore, the BodyDefTables use multiple BAPs to control vertices. A BodyDefTable contains “key deformations”, each key consisting of a combination of selected BAPs. A BodyDefTable contains the following data:

Table 1: BodyDefTable contents

BAPs				Vertices			
BAP ₁	BAP ₂	...	BAP _k	Vertex ₁	Vertex ₂	...	vertex _N
0	0	.	0	0	0	.	0
0	0	.	100	D_{11}	D_{21}	.	D_{N1}
0	100	.	0	D_{21}	D_{22}	.	D_{N2}
...							

Note that each entry in the BodyDefTable corresponds to a combination of BAPs. If the current body posture is not one of these BAP combinations, the vertex deformations have to be interpolated based on these BAP combinations. A linear interpolation technique solves this problem with low computational overhead.

Represent BAP keys (rows in Table 1) as k -dimensional points, where k is the number of BAP columns. Given several BAP keys $P_1, P_2, P_3, \dots, P_n$ computing linear interpolation at BAP point P . n represents the number of key positions to be used for interpolation, specified in a field of the BodyDefTable node.

Let $d_1, d_2, d_3, \dots, d_n$ be respective distances from P to keys. Let $v_1, v_2, v_3, \dots, v_n$ be tabular displacement values of a vertex at the keys.

For any key P_i , the deformation contributed by P_i should be inversely proportional to distance d_i from point P . Let this proportionality factor be f_i . Thus

$$DEF_i \text{ (deformation due to } P_i) = f_i * v_i$$

$$DEF \text{ (Total deformation at } P) = f_1 * v_1 + f_2 * v_2 + \dots + f_n * v_n$$

With the condition that $f_1 + f_2 + \dots + f_n = 1.0$ computing f_i requires simple arithmetic: Let total distance:

$$D = d_1 + d_2 + \dots + d_n$$

$$f_i = (1 - d_i/D)/(n-1)$$

Calculation of the proportionality factors is below. First compute direct proportionality factors t_i

$$t_1 = d_1/D, t_2 = d_2/D, \dots, t_n = d_n/D$$

Now $t_1 + t_2 + \dots + t_n = 1.0$. If we take deformation contribution by P_i as $DEF_i = t_i v_i$ then keys closest to point p contribute least. To have the opposite effect we get inverse proportionality by replacing t_i s:

$$t_i <-- (1 - t_i) <-- (1 - d_i/D)$$

$$t_1 <-- 1 - d_1/D, t_2 <-- 1 - d_2/D, \dots, t_n <-- 1 - d_n/D$$

But now

$$t_1 + t_2 + \dots + t_n = n - 1$$

To make right hand side 1.0, we divide by $n-1$. Thus the final factor $f_i = t_i/(n-1) = (1-d_i/D)/(n-1)$.

2.3 FDP/BDP Compression

The FBA BDP/FDP set is transmitted as a BIFS scene graph to the decoder. BIFS is a tool developed in MPEG-4 for coding MPEG-4 scene graphs [5]. It is a superset of VRML standard, with extensions of 2D nodes and control for streaming. The FBA nodes, defined in Section 2.3, are transmitted as BIFS nodes and coded using BIFS tools. The fields of the FBA nodes are quantized using a QuantizationParameter node, which specifies the quantization category and coding parameters such as min-max values. BIFS defines different quantization categories for different types of fields: *Position3D*, *Position2D*, *Color*, *TextureCoordinate*, *Angle*, *Scale*, *InterpolatorKeys*, *Normals*, *Rotations*, *ObjectSize3D*, *ObjectSize2D*, *Linear Quantization*, *Coordinate Quantization*.

As the bodySceneGraph scene graph contains geometry of the body, it is coded using BIFS tools for coding geometry. However, the other FBA nodes are special nodes and are coded based on specific syntax. The BodyDefTable node syntax and the coding scheme for its fields are as follows:

```
BodyDefTable [
  exposedField SFString bodySceneGraphNodeName // String
  exposedField MFInt32 bapIds // Integer, range [1,296]
  exposedField MFInt32 vertexIds // No quantization, range [0,+1]
  exposedField MFInt32 bapCombinations
                                     // No quantization, range [0,+1]
  exposedField MFVec3f displacements //MFVec3f
  exposedField SFInt32 numInterpolateKeys
                                     //No quantization, range [2,+1]
]
```

Figure 5: BodyDefTable node syntax

The similar coding scheme is used for FaceDefTable coding. After the node fields are quantized, the nodes are coded using arithmetic coding.

The BAP and FAP nodes are handled in a different way than the other FBA nodes, as they are used for streaming of animation in the form of FAPs/BAPs, rather than definition of geometry. When FBA is used in a profile together with BIFS, the FBA elementary stream is encapsulated within a BIFS-Anim stream. BIFS-Anim is a functionality provided by BIFS, to allow a single stream to update fields in multiple nodes in the transmitted BIFS scene graph. A BIFS-Anim frame with an FBA frame is similar to FBA stream, with a slight difference. The BIFS frame consists of a BIFS-Anim frame header and BIFS-Anim frame data. The data field has the same content as the FBA frame data: Face data followed by Body data. The BIFS-Anim frame header is also similar to FBA elementary stream frame header, however, it contains the BIFS-Anim mask instead of FBA object mask. This is an n -length Boolean array, where n is the number of nodes affected by the FBA stream.

Experiments on FaceDefTable and BodyDefTable coding have shown that it is possible to create high-quality deformable face and body models. The model shown in Figure 6 requires only 24 Kbytes for each node to contain the deformation tables in

addition to the static body geometry. Thus, it is possible to code body deformations with low overhead.

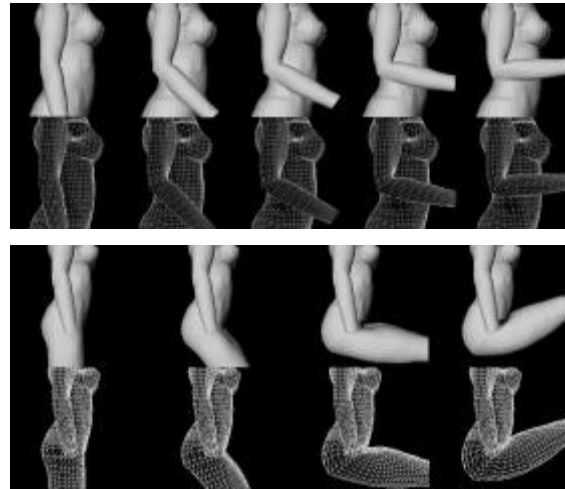


Figure 6: Result on Use of Body Deformation Tables

3. CONCLUSIONS

In this paper, we have presented the geometrical representation of the FBA Object in MPEG-4, for very low bitrate compression of human-like characters. The methods presented give efficient compression and are suitable for real-time communication applications such as videoconferencing, e-commerce, games, multiuser chat worlds. Further research will concentrate on the content creation side: creation of detailed cloth models as well as hair is still an area of research, and we expect the virtual character models to appear with high-quality models of cloth and hair in the near future. Additionally, using FBA models as avatars in multiuser worlds might require techniques to lower required bitrate per avatar, using high-level representation techniques.

4. REFERENCES

- [1] Capin Tolga K., Eric Petajan, Joern Ostermann, Very Low Bitrate Coding of Virtual Human Animation in MPEG-4, *Proceedings of ICME2000*, New York, NY, July 2000.
- [2] Web3D Working Group on Humanoid Animation, Specification for a Standard Humanoid, Version 1.1, August 1999.
- [3] MPEG-4 Overview, ISO/IEC JTC1/SC29 N2995, available at <http://drogo.cselt.it/mpeg/standards/mpeg-4/mpeg-4.htm>, October 1999.
- [4] ISO/IEC 14496-1:1999, Coding of Audio-Visual Objects: Systems, Amendment 1, December 1999.
- [5] ISO/IEC 14496-2:1999, Coding of Audio-Visual Objects, Visual, Amendment 1, December 1999.
- [6] Signes Julien, Yuval Fischer, Alexandros Eleftheriadis, MPEG-4's Binary Format for Scene Description, *Signal Processing: Image Communication*, Vol. 15, No.4-5, pp 321-345, January 2000.
- [7] VRML97: ISO/IEC 14772-1:1997, The Virtual Reality Modeling Language, 1997.