AFM File Format Reference

version 1.5 – 31.10.2007.

Visage Technologies AB • Tegskiftesgatan 291 • SE-58334 Linköping • Sweden
www.visagetechnologies.com

TABLE OF CONTENTS

1.Introduction

The AFM (Animatable Face and Body Model) file format is an open and extensible file format for 3D Virtual Character models with full definition of their animation capabilities based on morphing and skinning, with possible extensions to other animation methods. A Virtual Character in the AFM file format is ready for animation in any Visage Technologies product and in the 3rd party products built using visage|SDK.

The file consists of the model geometry in the standard VRML97 format, and the definition of animation behavior – the description of how the model will behave (deform) in response to each animation parameters. The animation parameters are international standard MPEG-4 face and Body Animation (FBA) parameters. This document often refers to this standard, and the reader is directed to the **Introduction to MPEG-4 Face and Body Animation** which is available as part of Visage Technologies documentation (the document "*MPEG-4 FBA Intro for visageSDK.pdf"*).

The AFM file format has the standard VRML 97 syntax. It uses PROTO statements to define a small number of new node types that are used to specify animation-specific information.

Typically, the first part of the file is a standard VRML scene graph containing the geometry of the virtual character model. This is followed by a number of animation-specific nodes of three types:

- **AFMVersion**: file format version number, used to manage changes in AFM format;
- **AnimationParameter**: used to define how each FBA parameter affects the 3D model;
- **IFSBoneWeights**: necessary for skinning;
- **NeutralPose**: if the initial position of the body and/or face is not the standard MPEG-4 FBA neutral position, then the neutral position is defined by this node;
- custom data for extensions.

There is one AnimationParameter node for each implemented animation parameter, and one IFSBoneWeights node for each mesh that uses skinning (bones). IFSBoneWeights nodes are not used if skinning is not used. NeutralPose node is optional and used only if the initial pose of the model is different from the MPEG-4 standard neutral pose. The following sections provide the PROTO definitions and full documentation for the AnimationParameter, IFSBoneWeights nodes and NeutralPose.

Further node types may be defined using PROTO statements in order to extend the file format with custom data. Any node types not recognized by the VT system when the file is parsed shall be passed to user-defined file-reading methods, so the user can access the data from these nodes.

2. Definitions

These definitions serve to establish a precise common understanding of the terms used in the description of the AFM file format. They assume that the reader has a good grasp of Computer Graphics concepts. The definitions are listed in a loosely logical order. Italics mean that the term has a definition in this list.

Scene graph. A hierarchical representation of a scene (in our case, the whole scene is the body of the virtual character). It is a hierarchy of nodes, with a root node on top of the hierarchy. Each node may have one or more children, and these relationships define the hierarchy. The main classes of nodes are geometry nodes and *transforms*. Geometry nodes contain visible geometry, typically a polygon mesh.

Transform (or transform node). A transform is a node in the *scene graph* that implements geometry transformation (translation, rotation, scale), which can be expressed by a transformation matrix. A transform thus changes the reference coordinate system, and all nodes below the transform are in the coordinate system of the transform. Transforms are compounded, i.e. a hierarchy of transforms forms a chain of transformation that successively changes the reference coordinate system. For each transform it is thus possible, by successively multiplying the transformation matrices of all transforms in the hierarchy starting from the root node, to compute the world-coordinate transformation matrix for the transform. This matrix expresses the transformation from the world coordinates to the transform, and it (or its inverse) can be used to convert between the world coordinate system and the local coordinate system of the transform and vice versa. Very importantly, any transform can also be a *bone*. We consider that a transform is a *bone* if it affects one or more vertices.

Bone-based animation. See *skinning*.

Skinning. Skinning is the process of applying one or more *transforms*, or *bones*, to the vertices of a polygon mesh. The purpose of skinning is to obtain a smooth deformation, typically when animating joints such as elbow or shoulder. Each *bone's* transformation is defined by its transformation matrix. Each bone has a weight that determines its influence on the vertex, and the final position of the vertex is the weighted sum of the results of all applied transformations. The sum of weights of all bones applied to any given vertex must be 1. If each vertex is affected by only one bone (meaning all weights are 1), this is rigid animation. Therefore it is interesting to notice that the classical scene graph, where each *transform* directly moves all the vertices directly under it in the *scene graph*, can be regarded as a special case of skinning – simply, every *transform* is a *bone* acting with weight 1 upon all vertices in the mesh(es) that are in the scene graph directly under the transform.

Bone. A bone is simply a *transform*. It is used in *skinning*. Any *transform* can be a bone, if it affects one or more vertices. For each vertex that a bone affects, a weight must be set in order to define the strength of influence of this bone on the vertex. The sum of weights of all bones affecting a vertex must always be 1. A weight is defined for every bone-vertex pair. A weight of 0 means that this bone does not affect this vertex, and these 0 weights are usually implicit (not written in a file). A weight of 1 means that only this bone, and no other, affects this vertex. This may also be set implicitely; for example, all vertices in a mesh directly under a *transform* are directly moved by that *transform*, so the *transform* is in fact a bone affecting all vertices with weight 1. Note that many systems, particularly modeling software (e.g. 3ds max), represent bones as segments, with a beginning and an end. This can be useful for visualizing the bone and for implicitely determining its influence. However, once the influence (weights) of the bone is determined, the bone is fully defined by the *transform* and it is not necessary to give it a length.

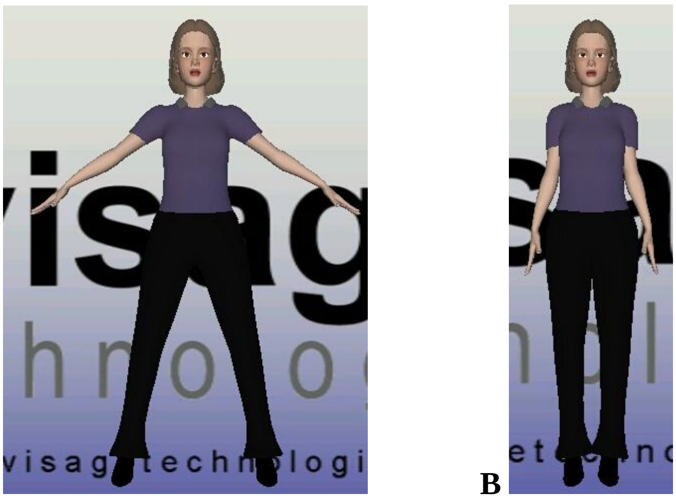
Morphing. Determining the final position of each vertex in a polygon mesh as a weighted sum of the positions of the corresponding vertices in one or more *morph targets*.

Morph target. An instance (or version) of a polygon mesh used in morphing. Each morph target is a version of the same polygon mesh deformed into some key position (e.g. lip shapes corresponding to visemes or phonemes). All morph targets for a polygon mesh have the same mesh topology, i.e. the only difference between them is in the positions of vertices. Thus, simple linear interpolation can be applied on the coordinates of each vertex in order to blend the position of each vertex between two or more morph targets. The result is a blending of the complete morph targets, e.g. a shape of the lips interpolated smoothly between closed mouth and an «o» phoneme.

Polygon mesh. A mesh of polygons (currently VT system supports only triangles). Usually the main building block for the model. A mesh is a node in the scene graph. It consists of vertices organized into polygons (triangles), and may also contain the normals, texture coordinates, material and other information related to geometry.

Mesh. See *polygon mesh*.

Indexed Face Set (IFS). See *polygon mesh*.



A B

Figure 1: (A) A typical initial/binding pose; (B) The neutral pose

Initial pose. The pose in which the character model geometry is written in the AFM file. In other words, if the AFM file is loaded into a standard VRML browser, the character model would appear in this pose.

Binding pose. The term is related to skinning. It is the starting pose of the body before any skinning is applied. Skinning is applied on the vertex positions from the binding pose. Usually the binding pose has arms and legs stretched out (see Figure 1A) in order to achieve good skinning results. In the AFM file, the *initial pose* is taken as the binding pose so **these two poses are the same**. In other words, the character model geometry written in the AFM file is taken as the binding pose.

Neutral pose (or neutral position). This is the neutral position of the face and body as defined by the MPEG-4 FBA standard. It is important for the animation system as the reference position from which the animation starts. When the AFM file is loaded into by Visage Technologies software, the character model will first be transformed into the neutral pose (see Figure 1B), then the animation will start from this pose. Note that the neutral pose is usually different from the *initial pose*/*binding pose*. The neutral pose is described in section 7.5 "MPEG-4 FBA neutral body and face position".

3.AFMVersion

This node is obligatory in the AFM file. It is used to manage changes in the AFM file format. The AFMVersion node is defined as follows (this PROTO definition must be in the AFM file):

```
PROTO AFMVersion [
    exposedField     SFFloat      versionNumber            0
]{Group{}}
```

When Visage Technologies software reads an AFM file it will **report a warning and exit with error code 199** in case the version number in the AFM file is not the same as the version that the software is designed to read.

NOTE: Because Visage Technologies libraries do not use MFC, the warning is printed to the stderr console and it may not be visible in an interactive application unless it is run in a debugger.

4. AnimationParameter

Each AnimationParameter node describes the interpretation of one animation parameter, i.e. defines how the geometry will be influenced by this parameter. The geometry may be influenced by morphing one or more IndexedFaceSet nodes and/or by moving one or more Transform nodes. Essentially, this provides a mapping from the animation parameters that appear in an MPEG-4 animation sequence to the geometry. For example, the *open_jaw* parameter may be mapped to morphing of the geometry in the jaw region; the *r_knee_flexion* parameter may be mapped to rotate the Transform node of the right knee. Thus each animation parameter is wired to specific actions on the geometry.

In a full implementation there would be one AnimationParameter node for each MPEG-4 FBA animation parameter. However, in a typical implementation only a subset of parameters is implemented – for example, a particular AFM file may implement only face animation, only body animation, or any subset of parameters deemed necessary for a specific application. The parameters that do not have an AnimationParameter node will be ignored when they appear in an animation sequence.

Section 10 provides the standard MPEG-4 parameter names. Their implementation should conform to the standard. i.e. the *r_knee_flexion* parameter must move the right knee, otherwise the model will not correctly interpret the standard MPEG-4 FBA animation files. There are 110 extension parameters that can be freely defined by the user to provide additional movements, e.g. moving the hair or some accessories, or providing other special effects.

The AnimationParameter node is defined as follows (this PROTO definition must be in the AFM file):

```
PROTO AnimationParameter [
    exposedField     SFString     name               ""
    exposedField     SFFloat      refValue           0
    exposedField     MFString     morphedIFSs        []
    exposedField     MFInt32      morphedVertices    []
    exposedField     MFVec3f      morphValues        []
    exposedField     MFString     tTransforms        []
    exposedField     MFVec3f      translations       []
    exposedField     MFString     rTransforms        []
    exposedField     MFRotation   rotations          []
    exposedField     MFString     sTransforms        []
    exposedField     MFVec3f      scales             []
]{Group{}}
```

4.1 name

This is the name of the animation parameter that is defined, e.g. “open_jaw”. The full list of animation parameter names is given in Section 10.

4.2 refValue

This is the reference value of the parameter, necessary because the animation mechanism is based on linear interpolation. The AnimationParameter node contains one static position of the 3D model, defined by vertex positions for morphing and/or the transform parameters. This is the reference position. This position corresponds to the reference value of the parameter. During animation, the positions for other values of the parameter are obtained by linear interpolation between the neutral position of the character and the reference position. The neutral position is the one defined by the MPEG-4 FBA standard.

This is best explained by example. Consider the body animation parameter *l_hip_abduct*. The MPEG-4 standard (see **Introduction to MPEG-4 Face and Body Animation**) defines this parameter as the rotation of the left leg from the hip outwards (to the left of the body). The neutral position is with the legs straight. Let us assume that the AnimationParameter for *l_hip_abduct* is implemented by rotating the transform node of the hip, and that the value for in the *rotations* field rotates the hip so that the leg goes out horizontaly to the left of the body. This is a rotation by 90 degrees from the neutral position. The unit for the body animation parameters is 10^{-5} radians, so 90 degrees is 157079 units. Therefore *refValue* field mujst be set to 157079.

Why does this work? Consider how the animation is done. Assume that at the current time in animation the value for *l_hip_abduct* is 78540. This is approximately half of the *refValue*. Therefore the linear interpolation will give the rotation which is half way between the neutral pose (leg straight down) and the reference pose (leg out at 90 degrees) – the leg will be at 45 degrees angle. This is exactly corresponding to the value for *l_hip_abduct* of 78540 (this is 45 degrees expressed in BAP units).

4.3 morphedIFSs

The list of Indexed Face Set nodes (IFS) that are morphed. Each string is the name of an Indexed Face Set in the VRML model, which must be defined in the file using a DEF statement, like this:

```
Geometry DEF aNiceName IndexedFaceSet {
```

The fields *morphedIFSs*, *morphedVertices* and *morphValues* together define a morph target – the target coordinates of all vertices in the model. The morph target corresponds to the reference value of the parameter (*refValue*), and other values of the parameter are implemented by linear interpolation between the neutral vertex position and the morph target. It is important to notice that, for each parameter, usually only a small number of vertices in the model are actually morphed (moved). The other vertices are not affected. Therefore, writing a morph target in the file simply as the list of target coordinates for all vertices would result in huge files containing many target coordinates which are the same as the original vertex coordinates, and therefore unnecessary. For this reason we adopt a slightly more complicated, but more efficient notation using these three fields. Basically, *morphedIFSs* tells which Indexed Face Sets of the model contain any vertices that are morphed; then *morphedVertices* tells which vertices within those IFSs are morphed. Finally, *morphValues* gives the target coordinates for each of the morphed vertices. This completely defines the morph target.

If morphing is not used for a particular parameter, all these fields are empty.

4.4 morphedVertices

The index to the vertices within the *morphedIFSs* that are morphed. This is a list of integers subdivided into sub-lists by the “-1” marker, like this:

```
i0 i1 i2 … in -1 j0 j1 j2 … jn -1 k0 k1 k2 … kn -1 …
```

The number of sub-lists must be the same as the number of Indexed Face Set (IFS) names in *morphedIFSs*, and each sub-list corresponds to one IFS. An IFS has a list of vertices: V_0, V_1, V_2 etc. Some of these vertices are morphed, some not. We regard them in groups of morphed and non-morphed vertices. i_0 is the number of morphed vertices in the first group; i_1 is the number of non-morphed vertices in the second group, and so on. For example:

```
3 2 4 -1 0 10 15 -1 …
```

In this example, in the first IFS vertices V_0, V_1, V_2 are morphed, vertices V_3, V_4 are not morphed, and vertices V_5, V_6, V_7, V_8 are morphed. In the second IFS, vertices $V_0 – V_9$ are not morphed, vertices $V_{10} – V_{24}$ are morphed, etc.

4.5 morphValues

The target coordinates for each morphed vertex. The ordered list of all morphed vertices in the model is established by the fields *morphedIFSs* and *morphedVertices*. For each of these vertices, there are target coordinates x, y and z. They are in the ***local*** coordinate system of each Indexed Face Set, just like the original coordinates written in the IndexedFaceSet node of the model.

4.6 tTransforms

The list of Transform nodes that are translated by this parameter. Each string is the name of a Transform in the VRML model, which must be defined in the file using a DEF statement, like this:

```
DEF aNiceName Transform {
```

4.7 Translations

A list of translations, one translation for each Transform node listed in the *tTransforms* field. Each Vec3f in this list is the absolute value of translation corresponding to the reference value (*refValue*) of the animation parameter. Thus, if the value of the animation parameter is exactly *refValue*, then the Transform will have its translation field set to the translation value from this list. For other values of the animation parameter, translation is linearly interpolated.

4.8 rTransforms

The list of Transform nodes that are rotated by this parameter. Each string is the name of a Transform in the VRML model, which must be defined in the file using a DEF statement.

4.9 rotations

A list of rotations, one rotation for each Transform node listed in the *tTransforms* field. Each rotation in this list is the absolute value, corresponding to the reference value (*refValue*) of the animation parameter. Thus, if the value of the animation parameter is exactly *refValue,* then the Transform will have its rotation field set to the rotation value from this list. For other values of the animation parameter, rotation is linearly interpolated.

4.10 sTransforms

The list of Transform nodes that are scaled by this parameter. Each string is the name of a Transform in the VRML model, which must be defined in the file using a DEF statement.

4.11 scales

A list of scales, one scale for each Transform node listed in the *tTransforms* field. Each Vec3f in this list is the absolute scale value, corresponding to the reference value (*refValue*) of the animation parameter. Thus, if the value of the animation parameter is exactly *refValue,* then the Transform will have its scale field set to the scale value from this list. For other values of the animation parameter, scale is linearly interpolated.

5.CubeMap

The CubeMap node allows to add a CubeMap texture to an Indexed Face Set. A CubeMap consists of 6 texture images, corresponding to 6 sides of a cube. The texture coordinates for a CubeMap consist of 3 coordinates per vertex, where one coordinate determines which side of the cube is used, and the other two work like classical texture coordinates within the chosen image. The VRML format does not support the CubeMap texture, so this is an extension to allow it. A CubeMap node is tied to a specific Indexed Face Set and specifies the CubeMap texture for it. This basically replaces the texture-related fields `url` (in the `ImageTexture` node) and `texCoord` (in the `IndexedFaceSet` node), because these fields can support only standard texture. Note that the `texCoordIndex` field in the `IndexedFaceSet` is still used in the standard way and must be filled correctly.

The CubeMap node is defined as follows:

```
PROTO Cubemap [
    exposedField SFString IFS ""
    exposedField MFString urls []
    exposedField MFVec3f texCoord []
]{Group{}}
```

5.1 IFS

The name of the Indexed Face Set for which we are defining the CubeMap, i.e. the string is the name of an Indexed Face Set in the VRML model, which must be defined in the file using a DEF statement, like this:

```
Geometry DEF aNiceName IndexedFaceSet {
```

5.2 urls

The list of the 6 texture images used in the CubeMap. The order of the image URLs needs to correspond to the following list of the CubeMap texture image targets:

1. GL_TEXTURE_CUBE_MAP_POSITIVE_X_EXT
2. GL_TEXTURE_CUBE_MAP_NEGATIVE_X_EXT
3. GL_TEXTURE_CUBE_MAP_POSITIVE_Y_EXT
4. GL_TEXTURE_CUBE_MAP_NEGATIVE_Y_EXT
5. GL_TEXTURE_CUBE_MAP_POSITIVE_Z_EXT
6. GL_TEXTURE_CUBE_MAP_NEGATIVE_Z_EXT

5.3 texCoord

Texture coordinates of the CubeMap. There are 3 coordinates per vertex.

5.4 Example usage

```
Cubemap {
                IFS "head"
```

```
          urls [
               "./cube_pics/right.bmp",
               "./cube_pics/left.bmp",
               "./cube_pics/bottom.bmp",
               "./cube_pics/top.bmp",
               "./cube_pics/front.bmp",
               "./cube_pics/rear.bmp"
               ]
          texCoord [
               0.15354 0.15354 0.15354
               0.148375 0.148375 0.148375
                              ...
               ]
}
```

6. IFSBoneWeights

The convention for setting the bone weights for skinning is the following:

- For each vertex in a mesh, the default bone is the transform directly above it in the scene graph hierarchy, and the default weight is 1.0. We will call this default transform the «native bone». This means that, if nothing more specific is defined (i.e. if no IFSBoneWeights nodes are used), we have the «classical» case of rigid-parts scene graph animation, each vertex only being affected by its native bone.
- For each vertex, one or more bones other than the native bone may be assigned. In this case a weight must be assigned for each of those other bones acting on the vertex. The weight for the native bone is computed as:

1.0 – [sum of all other bone's weights]

The weights for non-native bones are assigned using one IFSBoneWeights node for each mesh that is influenced by any non-native bones. The definition of the IFSBoneWeights node is:

```
PROTO IFSBoneWeights [
    exposedField     SFString     meshName            ""
    exposedField     MFString     bones               []
    exposedField     MFInt32      vertices            []
    exposedField     MFFloat      weights             []
]{ Group{}}
```

6.1 meshName

The name of the Indexed Face Set for which we are defining the weights, i.e. the string is the name of an Indexed Face Set in the VRML model, which must be defined in the file using a DEF statement, like this:

```
Geometry DEF aNiceName IndexedFaceSet {
```

6.2 bones

The list of bones that affect this mesh Indexed Face Set nodes (IFS) that are affected by this bone. Each string is the name of a Transform in the VRML model, which must be defined in the file using a DEF statement, like this:

```
DEF aNiceName Transform {
```

6.3 vertices

The list of vertices for which we are defining weights, i.e. the vertices that are affected by each bone. This is needed in order to reduce the file size. Typically, a relatively small number of vertices in a mesh is affected by bones, so it would be inefficient to store their weights in the file (they are all zero). This list therefore picks only those vertices for which it is necessary to define weights, and for other vertices weights are automatically set to 0.

This is a list of integers subdivided into sub-lists by the “-1” marker, like this:

```
i0 i1 i2 … in -1 j0 j1 j2 … jn -1 k0 k1 k2 … kn -1 …
```

The number of sub-lists must be the same as the number of Transform names in the *IFSs* field, and each sub-list corresponds to one bone. The mesh consists list of vertices: V_0, V_1, V_2 etc. Each of the sub-lists contains indices to those vertices. For example:

```
3 21 43 -1 0 10 15 -1 …
```

In this example, we indicate that weights for the first used bone will be set for vertices V_3, V_{21}, V_{43}; for vertices V_0, V_{10}, V_{15} for the second bone, etc.

6.4 weights

The list of weights for the vertices listed in the *vertices* field, following the convention for bone weights specified at the beginning of this paragraph. The number of weights in this list is the same as the number of vertex indices in the vertices field (the number of entries in the *vertices* field, not counting the -1 separators). For example, if the *vertices* field is like in the example given above (3 21 43 -1 0 10 15 -1 …), then the first three floating point values in the *weights* field are the weights for the first used bone for the vertices 3, 21 and 43; the next three values are the weights for the second bone for the vertices 0, 10 and 15, etc.

7. NeutralPose

The *NeutralPose* node is used to set the neutral pose of the character. For the discussion of the neutral pose and initial/binding pose see section 2 "Definitions". The neutral pose is described more precisely in the section 7.5 "MPEG-4 FBA neutral body and face position".

If the body is initially modelled in the MPEG-4 neutral position, *NeutralPose* node can be omitted.

The definition of the *NeutralPose* node is:

```
PROTO NeutralPose [
    exposedField     MFString     FAPs               []
    exposedField     MFInt32      FAPValues          []
    exposedField     MFString     NonNPTransforms    []
    exposedField     MFRotation   NPRotations        []
]{ Group{}}
```

7.1 FAPs

The list of FAPs that must be set to non-zero values to to put a neutral MPEG-4 face into the initial pose (see details in the next section). The FAPs are expressed by names, e.g. "open_jaw". The full list of animation parameter names is given in Section 10.

7.2 FAPValues

The values of the FAPs listed in the *FAPs* field. It contains such FAPs that would put a neutral MPEG-4 face into the initial pose. Only those FAPs and BAPs that are non-zero are written into the *NeutralPose* node.

The typical example of usage concerns the eyelids, which in the standard MPEG-4 neutral pose should be tangent to the iris, i.e. the upper eyelid should just touch the upper edge of the iris, and the lower one just touch the lower edge. The face is often not modelled in this way, and this may be corrected by setting the *close_t_r_eyelid*, *close_t_l_eyelid*, *close_b_r_eyelid* and *close_b_l_eyelid* parameters. For example, let us assume that the eyelids are slightly closed in the initial pose. The upper eyelid is lowered by 10% of the iris diameter, and the lower eyelid is raised by the same amount. The eyelid closing parameters in MPEG-4 are expressed in units equivalent to 1/1024 of the iris diameter (see details in the "Introduction to MPEG-4 Face and Body Animation" manual that is part of Visage Technologies documentation). Therefore, a 10% closure of the eyelid is expressed by setting the parameters to 102 (10% of 1024, rounded to nearest integer). So, the FAPs that put a neutral MPEG-4 face into this particular initial pose are *close_t_r_eyelid*, *close_t_l_eyelid*, *close_b_r_eyelid* and *close_b_l_eyelid* parameters all set to 102. The *NeutralPose* node would look like this:

```
NeutralPose {
    FAPs ["close_t_r_eyelid",
          "close_t_l_eyelid",
          "close_b_r_eyelid",
          "close_b_l_eyelid"]
    BAPValues [102,102,102,102]
}
```

7.3 NonNPTransforms

The list of Transform nodes that are not in the neutral pose, i.e. the Transform nodes that need to be rotated in order to put the character from the initial pose into the neutral pose.

Note that the character must be modeled in such a way that it is possible to bring it into the neutral pose by rotations only (without translations or scaling).

7.4 NPRotations

A list of rotations, one rotation for each Transform node listed in the *NonNPTransforms* field. Each rotation in this list is the absolute value, corresponding to the neutral pose. In other words, if the rotation components of the Transform nodes listed in the *NonNPTransforms* field are replaced by rotations listed in this field, the body will be in the neutral pose.

7.5 MPEG-4 FBA neutral body and face position

The neutral position of the face (when all FAPs are 0) is defined as follows:

- the coordinate system is right-handed; head axes are parallel to the world axes
- gaze is in direction of Z axis
- all face muscles are relaxed
- eyelids are tangent to the iris
- the pupil is one third of IRISD0
- 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

The MPEG-4 standard neutral body position (when all BAPs are 0) is defined by standing posture, illustrated in Figure 2. This posture is defined as follows: the feet should point to the front direction, the two arms should be placed on the side of the body with the palm of the hands facing inward.

For more details concerning MPEG-4 Face and Body Animation, **consult the Introduction to MPEG-4 Face and Body Animation**, available as part of Visage Technologies documentation.

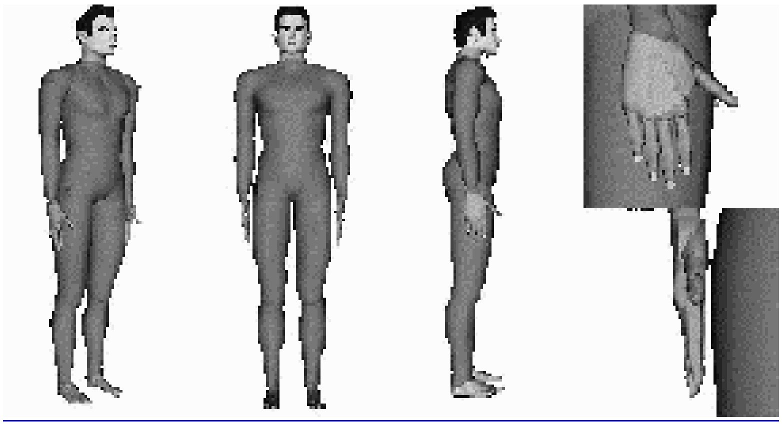


Figure 2: MPEG-4 standard neutral body pose

8. Sizing and positioning the character

The character should be modeled in a standing position, facing in the +Z direction with +Y up and +X to the humanoid's left. The origin (0, 0, 0) should be located at ground level, between the humanoid's feet.

The humanoid should be built with actual human size ranges in mind. All dimensions are in meters.

As a help to check whether the character is created with correct size and positioning, the following VRML model can be used. It contains a flat box of approximately the human size (1.8 meters). When this model is rendered together with the virtual character, feet of the character should be aligned with the bottom of the red box, and the size of the character should be similar to the box, se seen in Figure 3.

```
#VRML V2.0 utf8
Shape {
    appearance Appearance {
          material Material {
          diffuseColor 1 0 0
          }
    }
    geometry IndexedFaceSet{
          coord Coordinate { point [
          -0.3  0  0.01,
          -0.3  0  -0.01,
           0.3  0  -0.01,
           0.3  0  0.01,
          -0.3  1.8  0.01,
          -0.3  1.8  -0.01,
           0.3  1.8  -0.01,
           0.3  1.8  0.01
          ]}
          coordIndex [
          0 1 2 3 -1,
          0 3 7 4 -1,
          3 2 6 7 -1,
          1 5 6 2 -1,
          4 5 1 0 -1,
          4 7 6 5 -1
          ]
    }
}
```

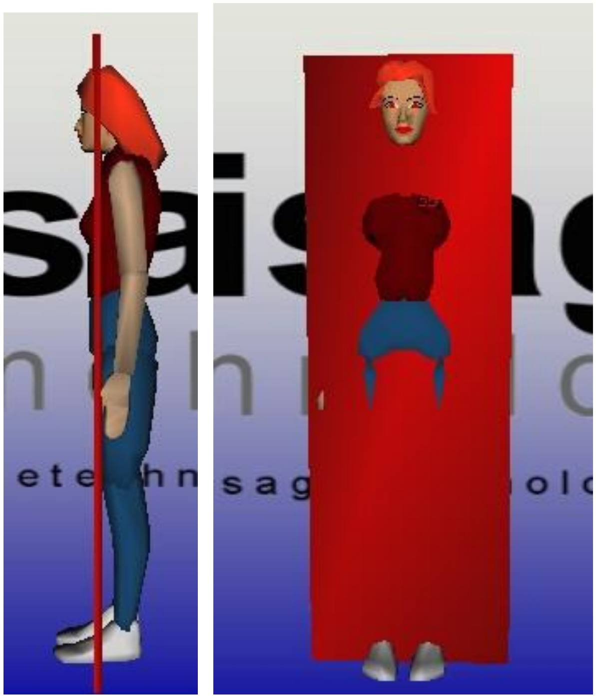


Figure 3: Virtual character rendered together with the helper box for positioning & sizing verification (side and front view)

9.Handling Non Uniform Scale

Non Uniform Scale (NUS) typically generates problems when used in a scene graph, because rotations in Trnasform nodes underneath the NUS turn into skew. In other words, if the top transform contains a NUS (e.g. it stretches in X direction more than in Y and Z), than in any transform in the hierarchy below the rotation will not look correct – when the rotation changes, the object will deform (skew) instead of just rotating, because of the influence of the NUS from the top transform node. It is therefore recommended not tu use NUS at all – each scale should be uniform, e.g. "scale 2 2 2", not "scale 2 4 2".

Unfortunatly, some modeling systems use NUS to create the body skeleton. This is the case of 3ds max Biped. The Visage Technologies software can handle NUS under certain conditions in order to support such systems. This is based specifically on the export of Bipeds from 3ds max, which has been tested. It is expected to function for other similar systems.

Visage Technologies will handle NUS under following conditions:

- NUS exists only in the nodes that do not contain visible geometry
- the names of such nodes (as defined by DEF) begin with "Bip"

This is the case when a Biped is exported from 3ds max (Bones must be hidden). All nodes that form the Biped skeleton hierarchy have names begining with "Bip" and the skeleton is invisible – it is used to move the skin using the skinning process.

When this kind of AFM model is read by Visage Technologies software, the nodes of the skeleton hierarchy are processed in such a way that all scales are set to [1 1 1], i.e. there is no scaling. To compensate, translations are set in such a way that the joints of the skeleton remein in their correct places. The angles of the skeleton do not change in this process, so the animation is correct.

Non Uniform Scale should not be applied to other nodes in the model, otherwise it is very likely that animation will not be correct.

10. Animation Parameter Names

This is the list of the standard names allowed in the name field of the AnimationParameter node. The more detailed definitions are available in the "MPEG-4 Face and Body Animation Introduction" document. The parameters are divided into four groups: visemes, expressions, low-level face animation parameters, body animation parameters and extension (user-defined) parameters.

10.1 Visemes

viseme_sil	none, neutral face
viseme_PP	p, b, m, as in words Put, bed, mill
viseme_FF	f, v, as in words Far, voice
viseme_TH	T, D, as in words Think, that.
viseme_DD	t, d, as in words Tip, doll
viseme_kk	k, g, as in words Call, gas
viseme_CH	tS, dZ, S, as in words Chair, join, she
viseme_SS	s, z, as in words Sir, zeal
viseme_nn	n, l, as in words Lot, not
viseme_RR	r, as in words Red
viseme_aa	A:, as in words Car.
viseme_E	e, as in words Bed
viseme_ih	I, as in words Tip.
viseme_oh	Q, as in words Top.
viseme_ou	U, as in words book.

10.2 Expressions

expression_neutral	Neutral face, no expression.
expression_joy	Joy expression.
expression_sadness	Sadness expression.
expression_anger	Anger expression.
expression_fear	Fear expression.
expression_disgust	Disgust expression.
expression_surprise	Surprise expression.

10.3 Low-level face animation parameters

open_jaw	Vertical jaw displacement (does not affect mouth opening).
lower_t_midlip	Vertical top middle inner lip displacement.
raise_b_midlip	Vertical bottom middle inner lip displacement.
stretch_l_cornerlip	Horizontal displacement of left inner lip corner.
stretch_r_cornerlip	Horizontal displacement of right inner lip corner.
lower_t_lip_lm	Vertical displacement of midpoint between left corner and middle of top inner lip.
lower_t_lip_rm	Vertical displacement of midpoint between right corner and middle of top inner lip.
raise_b_lip_lm	Vertical displacement of midpoint between left corner and middle of bottom inner lip.
raise_b_lip_rm	Vertical displacement of midpoint between right corner and middle of bottom inner lip.
raise_l_cornerlip	Vertical displacement of left inner lip corner.
raise_r_cornerlip	Vertical displacement of right inner lip corner.
thrust_jaw	Depth displacement of jaw.
shift_jaw	Side to side displacement of jaw.
push_b_lip	Depth displacement of bottom middle lip.
push_t_lip	Depth displacement of top middle lip.
depress_chin	Upward and compressing movement of the chin (like in sadness).
close_t_l_eyelid	Vertical displacement of top left eyelid.
close_t_r_eyelid	Vertical displacement of top right eyelid.
close_b_l_eyelid	Vertical displacement of bottom left eyelid.
close_b_r_eyelid	Vertical displacement of bottom right eyelid.
yaw_l_eyeball	Horizontal orientation of left eyeball; NOTE: the unit for rotation in 1e-5 rad.
yaw_r_eyeball	Horizontal orientation of right eyeball; NOTE: the unit for rotation in 1e-5 rad.
pitch_l_eyeball	Vertical orientation of left eyeball; NOTE: the unit for rotation in 1e-5 rad.
pitch_r_eyeball	Vertical orientation of right eyeball; NOTE: the unit for

	rotation in 1e-5 rad.
thrust_l_eyeball	Depth displacement of left eyeball.
thrust_r_eyebal	Depth displacement of right eyeball.
dilate_l_pupil	Dilation of left pupil.
dilate_r_pupil	Dilation of right pupil.
raise_l_i_eyebrow	Vertical displacement of left inner eyebrow.
raise_r_i_eyebrow	Vertical displacement of right inner eyebrow.
raise_l_m_eyebrow	Vertical displacement of left middle eyebrow.
raise_r_m_eyebrow	Vertical displacement of right middle eyebrow.
raise_l_o_eyebrow	Vertical displacement of left outer eyebrow.
raise_r_o_eyebrow	Vertical displacement of right outer eyebrow.
squeeze_l_eyebrow	Horizontal displacement of left eyebrow.
squeeze_r_eyebrow	Horizontal displacement of right eyebrow.
puff_l_cheek	Horizontal displacement of left cheeck.
puff_r_cheek	Horizontal displacement of right cheeck.
lift_l_cheek	Vertical displacement of left cheek.
lift_r_cheek	Vertical displacement of right cheek.
shift_tongue_tip	Horizontal displacement of tongue tip.
raise_tongue_tip	Vertical displacement of tongue tip.
thrust_tongue_tip	Depth displacement of tongue tip.
raise_tongue	Vertical displacement of tongue.
tongue_roll	Rolling of the tongue into U shape; NOTE: the unit for rotation in 1e-5 rad.
head_pitch	Head pitch angle from top of spine; NOTE: the unit for rotation in 1e-5 rad.
head_yaw	Head yaw angle from top of spine; NOTE: the unit for rotation in 1e-5 rad.
head_roll	Head roll angle from top of spine; NOTE: the unit for rotation in 1e-5 rad.
lower_t_midlip_o	Vertical top middle outer lip displacement.
raise_b_midlip_o	Vertical bottom middle outer lip displacement.
stretch_l_cornerlip_o	Horizontal displacement of left outer lip corner.
stretch_r_cornerlip_o	Horizontal displacement of right outer lip corner.

lower_t_lip_lm_o	Vertical displacement of midpoint between left corner and middle of top outer lip.
lower_t_lip_rm_o	Vertical displacement of midpoint between right corner and middle of top outer lip.
raise_b_lip_lm_o	Vertical displacement of midpoint between left corner and middle of bottom outer lip.
raise_b_lip_rm_o	Vertical displacement of midpoint between right corner and middle of bottom outer lip.
raise_l_cornerlip_o	Vertical displacement of left outer lip corner.
raise_r_cornerlip_o	Vertical displacement of right outer lip corner.
stretch_l_nose	Horizontal displacement of left side of nose.
stretch_r_nose	Horizontal displacement of right side of nose.
raise_nose	Vertical displacement of nose tip.
bend_nose	Horizontal displacement of nose tip.
raise_l_ear	Vertical displacement of left ear.
raise_r_ear	Vertical displacement of right ear.
pull_l_ear	Horizontal displacement of left ear.
pull_r_ear	Horizontal displacement of right ear.

10.4 Body animation parameters

sacroiliac_tilt	Forward-backward motion of the pelvis in the sagittal plane.
sacroiliac_torsion	Rotation of the pelvis along the body vertical axis (defined by skeleton root).
sacroiliac_roll	Side to side swinging of the pelvis in the coronal plane.
l_hip_flexion	Forward-backward rotation in the sagittal plane.
r_hip_flexion	Forward-backward rotation in the sagittal plane.
l_hip_abduct	Sideward opening in the coronal plane.
r_hip_abduct	Sideward opening in the coronal plane.
l_hip_twisting	Rotation along the thigh axis.
r_hip_twisting	Rotation along the thigh axis.
l_knee_flexion	Flexion-extension of the leg in the sagittal plane.
r_knee_flexion	Flexion-extension of the leg in the sagittal plane.

l_knee_twisting	Rotation along the shank axis.
r_knee_twisting	Rotation along the shank axis.
l_ankle_flexion	Flexion-extension of the foot in the sagittal plane.
r_ankle_flexion	Flexion-extension of the foot in the sagittal plane.
l_ankle_twisting	Rotation along the knee axis.
r_ankle_twisting	Rotation along the knee axis.
l_subtalar_flexion	Sideward orientation of the foot.
r_subtalar_flexion	Sideward orientation of the foot.
l_midtarsal_twisting	Internal twisting of the foot (also called navicular joint in anatomy).
r_midtarsal_twisting	Internal twisting of the foot (also called navicular joint in anatomy).
l_metatarsal_flexion	Up and down rotation of the toe in the sagittal plane.
r_metatarsal_flexion	Up and down rotation of the toe in the sagittal plane.
l_sternoclavicular_abduct	Up and down motion in the coronal plane.
r_sternoclavicular_abduct	Up and down motion in the coronal plane.
l_sternoclavicular_rotate	Rotation in the transverse plane.
r_sternoclavicular_rotate	Rotation in the transverse plane.
l_acromioclavicular_abduct	Up and down motion in the coronal plane.
r_acromioclavicular_abduct	Up and down motion in the coronal plane.
l_acromioclavicular_rotate	Rotation in the transverse plane.
r_acromioclavicular_rotate	Rotation in the transverse plane.
l_shoulder_flexion	Forward-backward motion in the sagittal plane.
r_shoulder_flexion	Forward-backward motion in the sagittal plane.
l_shoulder_abduct	Sideward motion in the coronal plane.
r_shoulder_abduct	Sideward motion in the coronal plane.
l_shoulder_twisting	Rotation along the scapular axis.
r_shoulder_twisting	Rotation along the scapular axis.
l_elbow_flexion	Flexion-extension of the arm in the sagittal plane.
r_elbow_flexion	Flexion-extension of the arm in the sagittal plane.
l_elbow_twisting	Rotation of the forearm along the upper arm axis.
r_elbow_twisting	Rotation of the forearm along the upper arm axis.

l_wrist_flexion	Rotation of the hand in the coronal plane.
r_wrist_flexion	Rotation of the hand in the coronal plane.
l_wrist_pivot	Rotation of the hand in the sagittal planes.
r_wrist_pivot	Rotation of the hand in the sagittal planes.
l_wrist_twisting	Rotation of the hand along the forearm axis.
r_wrist_twisting	Rotation of the hand along the forearm axis.
skullbase_roll	Sideward motion of the skull along the frontal axis.
skullbase_torsion	Twisting of the skull along the vertical axis.
skullbase_tilt	Forward-backward motion in the sagittal plane along a lateral axis.
vc1_roll	Sideward motion of vertebra C1.
vc1_torsion	Twisting of vertebra C1.
vc1_tilt	Forward-backward motion of vertebra C1 in the sagittal plane.
vc2_roll	Sideward motion of vertebra C2.
vc2_torsion	Twisting of vertebra C2.
vc2_tilt	Forward-backward motion of vertebra C2 in the sagittal plane.
vc3_roll	Sideward motion of vertebra C3.
vc3_torsion	Twisting of vertebra C3.
vc3_tilt	Forward-backward motion of vertebra C3 in the sagittal plane.
vc4_roll	Sideward motion of vertebra C4.
vc4_torsion	Twisting of vertebra C4.
vc4_tilt	Forward-backward motion of vertebra C4 in the sagittal plane.
vc5_roll	Sideward motion of vertebra C5.
vc5_torsion	Twisting of vertebra C5.
vc5_tilt	Forward-backward motion of vertebra C5 in the sagittal plane.
vc6_roll	Sideward motion of vertebra C6.
vc6_torsion	Twisting of vertebra C6.
vc6_tilt	Forward-backward motion of vertebra C6 in the sagittal plane.

vc7_roll	Sideward motion of vertebra C7.
vc7_torsion	Twisting of vertebra C7.
vc7_tilt	Forward-backward motion of vertebra C7 in the sagittal plane.
vt1_roll	Sideward motion of vertebra T1.
vt1_torsion	Twisting of vertebra T1.
vt1_tilt	Forward-backward motion of vertebra T1 in the sagittal plane.
vt2_roll	Sideward motion of vertebra T2.
vt2_torsion	Twisting of vertebra T2.
vt2_tilt	Forward-backward motion of vertebra T2 in the sagittal plane.
vt3_roll	Sideward motion of vertebra T3.
vt3_torsion	Twisting of vertebra T3.
vt3_tilt	Forward-backward motion of vertebra T3 in the sagittal plane.
vt4_roll	Sideward motion of vertebra T4.
vt4_torsion	Twisting of vertebra T4.
vt4_tilt	Forward-backward motion of vertebra T4 in the sagittal plane.
vt5_roll	Sideward motion of vertebra T5.
vt5_torsion	Twisting of vertebra T5.
vt5_tilt	Forward-backward motion of vertebra T5 in the sagittal plane.
vt6_roll	Sideward motion of vertebra T6.
vt6_torsion	Twisting of vertebra T6.
vt6_tilt	Forward-backward motion of vertebra T6 in the sagittal plane.
vt7_roll	Sideward motion of vertebra T7.
vt7_torsion	Twisting of vertebra T7.
vt7_tilt	Forward-backward motion of vertebra T7 in the sagittal plane.
vt8_roll	Sideward motion of vertebra T8.
vt8_torsion	Twisting of vertebra T8.

vt8_tilt	Forward-backward motion of vertebra T8 in the sagittal plane.
vt9_roll	Sideward motion of vertebra T9.
vt9_torsion	Twisting of vertebra T9.
vt9_tilt	Forward-backward motion of vertebra T9 in the sagittal plane.
vt_10_roll	Sideward motion of vertebra T10.
vt10_torsion	Twisting of vertebra T10.
vt10_tilt	Forward-backward motion of vertebra T10 in sagittal plane.
vt11_roll	Sideward motion of vertebra T11.
vt11_torsion	Twisting of vertebra T11.
vt11_tilt	Forward-backward motion of vertebra T11 in sagittal plane.
vt12_roll	Sideward motion of vertebra T12.
vt12_torsion	Twisting of vertebra T12.
vt12_tilt	Forward-backward motion of vertebra T12 in sagittal plane.
vl1_roll	Sideward motion of vertebra L1.
vl1_torsion	Twisting of vertebra L1.
vl1_tilt	Forward-backward motion of vertebra L1 in sagittal plane.
vl2_roll	Sideward motion of vertebra L2.
vl2_torsion	Twisting of vertebra L2.
vl2_tilt	Forward-backward motion of vertebra L2 in sagittal plane.
vl3_roll	Sideward motion of vertebra L3.
vl3_torsion	Twisting of vertebra L3.
vl3_tilt	Forward-backward motion of vertebra L3 in sagittal plane.
vl4_roll	Sideward motion of vertebra L4.
vl4_torsion	Twisting of vertebra L4.
vl4_tilt	Forward-backward motion of vertebra L4 in sagittal plane.
vl5_roll	Sideward motion of vertebra L5.

vl5_torsion	Twisting of vertebra L5.
vl5_tilt	Forward-backward motion of vertebra L5 in sagittal plane.
l_pinky0_flexion	Metacarpal flexing mobility of the pinky finger.
r_pinky0_flexion	Metacarpal flexing mobility of the pinky finger.
l_pinky1_flexion	First knukle of the pinky finger.
r_pinky1_flexion	First knukle of the pinky finger.
l_pinky1_pivot	Lateral mobility of the pinky finger.
r_pinky1_pivot	Lateral mobility of the pinky finger.
l_pinky1_twisting	Along the pinky finger axis.
r_pinky1_twisting	Along the pinky finger axis.
l_pinky2_flexion	Second knuckle of the pinky number.
r_pinky2_flexion	Second knuckle of the pinky number.
l_pinky3_flexion	Third knuckle of the pinky finger.
r_pinky3_flexion	Third knuckle of the pinky finger.
l_ring0_flexion	Metacarpal flexing mobility of the ring finger.
r_ring0_flexion	Metacarpal flexing mobility of the ring finger.
l_ring1_flexion	First knukle of the ring finger.
r_ring1_flexion	First knukle of the ring finger.
l_ring1_pivot	Lateral mobility of the ring finger.
r_ring1_pivot	Lateral mobility of the ring finger.
l_ring1_twisting	Along the ring finger axis.
r_ring1_twisting	Along the ring finger axis.
l_ring2_flexion	Second knuckle of the ring number.
r_ring2_flexion	Second knuckle of the ring number.
l_ring3_flexion	Third knuckle of the ring finger.
r_ring3_flexion	Third knuckle of the ring finger.
l_middle0_flexion	Metacarpal flexing mobility of the middle finger.
r_middle0_flexion	Metacarpal flexing mobility of the middle finger.
l_middle1_flexion	First knukle of the middle finger.
r_middle1_flexion	First knukle of the middle finger.
l_middle1_pivot	Lateral mobility of the middle finger.

r_middle1_pivot	Lateral mobility of the middle finger.
l_middle1_twisting	Along the middle finger axis.
r_middle1_twisting	Along the middle finger axis.
l_middle2_flexion	Second knuckle of the middle number.
r_middle2_flexion	Second knuckle of the middle number.
l_middle3_flexion	Third knuckle of the middle finger.
r_middle3_flexion	Third knuckle of the middle finger.
l_index0_flexion	Metacarpal flexing mobility of the index finger.
r_index0_flexion	Metacarpal flexing mobility of the index finger.
l_index1_flexion	First knukle of the index finger.
r_index1_flexion	First knukle of the index finger.
l_index1_pivot	Lateral mobility of the index finger.
r_index1_pivot	Lateral mobility of the index finger.
l_index1_twisting	Along the index finger axis.
r_index1_twisting	Along the index finger axis.
l_index2_flexion	Second knuckle of the index number.
r_index2_flexion	Second knuckle of the index number.
l_index3_flexion	Third knuckle of the index finger.
r_index3_flexion	Third knuckle of the index finger.
l_thumb1_flexion	First knukle of the thumb finger.
r_thumb1_flexion	First knukle of the thumb finger.
l_thumb1_pivot	Lateral mobility of the thumb finger.
r_thumb1_pivot	Lateral mobility of the thumb finger.
l_thumb1_twisting	Along the thumb finger axis.
r_thumb1_twisting	Along the thumb finger axis.
l_thumb2_flexion	Second knuckle of the thumb number.
r_thumb2_flexion	Second knuckle of the thumb number.
l_thumb3_flexion	Third knuckle of the thumb finger.
r_thumb3_flexion	Third knuckle of the thumb finger.
HumanoidRoot_tr_vertical	Body origin translation in vertical direction.
HumanoidRoot_tr_lateral	Body origin translation in lateral direction.
HumanoidRoot_tr_frontal	Body origin translation in frontal direction.

HumanoidRoot_rt_body_turn	Rotation of the skeleton root along the body coordinate system's vertical axis.
HumanoidRoot_rt_body_roll	Rotation of the skeleton root along the body coordinate system's frontal axis.
HumanoidRoot_rt_body_tilt	Rotation of the skeleton root along the body coordinate system's lateral axis.