SPIE, Vol 4672, 2002 (86-94)

ANIMATION – a system for animation scene and contents creation, retrieval and display

Peter L. Stanchev*
Kettering University

ABSTRACT

There is an increasing interest in the computer animation. The most of the current animation software tools are described by Richard Parent in [5]. The driving ideas behind animation in general and computer animation in particular are presented in [6]. More and more animations are available in various places on the Internet. The increasing availability of potentially interesting materials makes the search extremely difficult especially when the search is by the animation contents. This work is devoted to analyzing the syntax contents of the animations. In the paper we present the ANIMATION system – a system for animation scene and contents creation, retrieval and display. The system is based on MPEG-4 standard [3, 4]. MPEG-4 is an ISO/IEC standard developed by MPEG (Moving Picture Experts Group). These standard makes interactive video on CD-ROM and Digital Television possible.

Keywords: Animation, Internet, Contents based retrieval, MPEG-4, MPEG-7.

1. Introduction

The MPEG-4 standard (that is already available in commercial systems) allows object-based encoding of video sequences. The new MPEG-7 standard, officially issued in the year 2001, will standardize content-based manipulation of objects in images and video sequences. MPEG–4 standard offers to developers a standardized way to combine different audio and visual objects in a complex, composite multimedia product. It specifies necessary methods to compose such objects together in an MPEG-4 terminal to form complete scenes. The standard also provides a standardized way to transport multimedia data from its storage to the end-user, which makes MPEG–4 suitable for network multimedia applications, including INTERNET applications. MPEG– 4 standard describes many aspects of multimedia applications. It describes 3D scenes definitions, binary coding of audio and visual objects, network transportation, etc. It allows implementation only of subset of standard's components and gives ability to easily extending features of an existing MPEG– 4 applications. In this paper we present the ANIMATION system, a system for animation scene and content creation, retrieval and display. The system is fully based on the MPEG -4 standard.

2. THE ANIMATION SYSTEM ARCHITECTURE

The ANIMATION system contains three separate tools: (1) the ANIMATION editor, (2) the ANIMATION searches by contents engine, and (3) the ANIMATION display tool (Fig. 1).

* pstanche@kettering.edu; phone: 1 810 762 7927; fax 1 810 762 9796; http://ketering.edu/~pstanche; Kettering University, Flint, Michigan, 48504 USA.

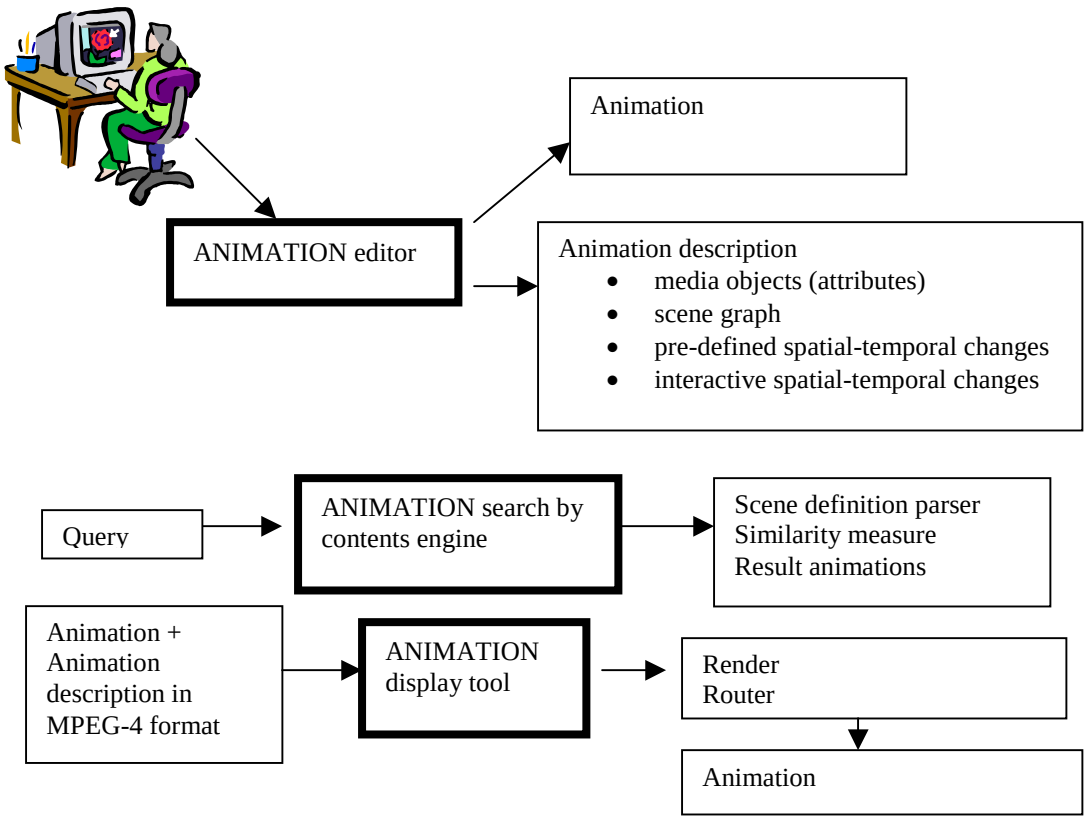

Figure 1. The ANIMATION system architecture

2.1. The ANIMATION editor

The first tool of the system is an editor for preparing an animation. During the animation creation scene description is created, based on MPEG-4 format [2]. Like in the Cambridge Animation Systems [1] the ANI MATION system supports Drawing window – provides multiple modes for viewing the scene in selected frames and Exposure Sheet window – comprehensive timing tools for setting key frames. Scene Graph provides a visual representation of the scene. Nodes represent elements and effects. The Replayer allows immediate viewing and checking of the composted scene. Effects Key frames can be rationed to allow camera or characters to track one another or to link effects together proportionally. Nodes represent all effects and elements. Users can make Macros of effects created by combining multiple nodes. Vector painting, drawing tools and color blending and blurred effects are also available.

The scene description is compiled to BIFS format (Binary Format for Scenes), which provide binary representation of scenes that reduce space needed to code them. BIFS contains the following four types of information:

1. The ***attributes of media objects,*** which define their audio-visual properties;

2. The **structure of the scene graph**, which contains these media objects. A scene description follows a hierarchical structure that can be represented as a tree. Nodes of an example graph form media objects, are illustrated in Figure 2. The structure of the graph is not necessarily static.

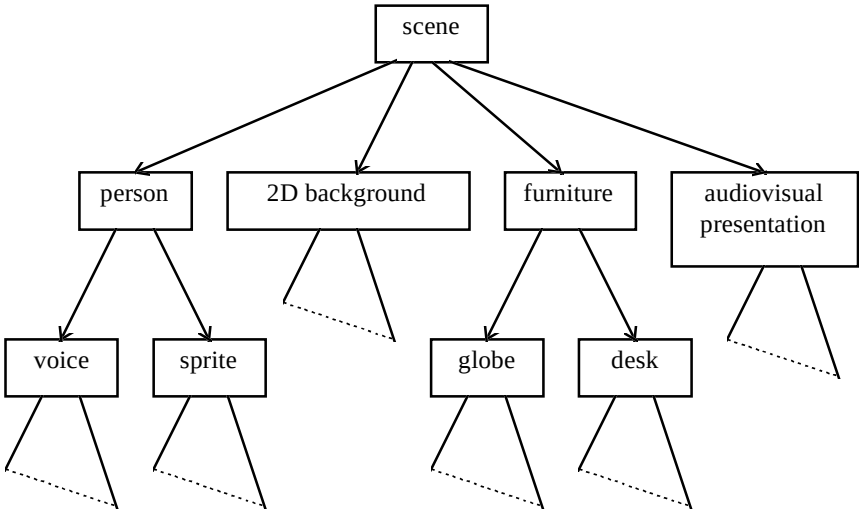

Figure 2. The logical structure of a scene

3. The **pre-defined spatial-temporal changes** (or "self-behaviors") of these objects. Media objects have both a spatial and a temporal extent. Complex objects are constructed by using appropriate scene description nodes that combine to form complex objects and thus build up the scene description tree. Objects may be located in 2-dimensional or 3-dimensional space. Every object has a local coordinate system. A local coordinate system is one in which the object has a fixed spatio-temporal location and scale (size and orientation). Objects are positioned in a scene by specifying a coordinate transformation from the object's local coordinate system into another coordinate system defined by a parent node in the tree.

4. The **spatial-temporal changes** triggered by user interaction.

2.2. The ANIMATION searches by contents engine

While much work has already been done in the direction of matching point sets, two curves, or two regions, little attention so far has been paid to developing methods for matching a collection of curves and regions against another collection. For this purpose we use the attribute relation graphs. The ANIMATION searches by contents engine contains search procedures that can be added to standard browsers. It allows if the animation is presented as MPEG-4 format, created with our editor or any other tool supported this standard, sear by animation contents to be performed. It includes tools for specifying the animation media objects, which we search by their properties (attributes), mutual position of the search objects, and spatial-temporal changes of these objects. The scene definition parser reads the scene structure and the node attribute, and builds the scene tree. It also reads all ROUTE statements and prepares the routing table for the events. All scene creation is made thorough internal scene object interface.

The image query contains description of the desired animation. During the retrieval, the tolerance between the described animation and the result subset is given. The available retrieval methods are:

1. **Retrieval by object names.** The retrieval by object names is similar to the retrieval in conventional database systems.

2. **Retrieval by color similarity.** For retrieving an animation or an object with a given color. The following methods are presented:
   - pick a template from predefined images (such as a sunny sky, a clear sea, etc.) and query for the animation having images with similar color distribution;
   - make an own template by using parts of stored images and/or use drawing tools;

- use a color picker to specify from a color palette the percentage of the desired colors in the result subset (in this way a query can be formed as follows: “Show me an animation containing images with color distribution: 40% yellow and 60% blue”).

3. **Retrieval by shape similarity.** The departing point in this case is the shape similarity measure based on the correspondence of visual parts. For specifying an animation containing image objects with the desired shape the user can:
   - draw an approximation of the object shape;
   - copy the shape for the query from a “shape” gallery.

4. **Retrieval by animation example similarity.** For specifying an example animation the user can:
   - identify an animation from the animation database;
   - make their own animation.

5. **Retrieval by spatial and temporal constrains.** The spatial constraints are given using icons and text descriptions. There are also tools for scaling and rotating the icons and presenting a temporal change. The result subset contains the animations that contain objects arranged in a manner similar to the way shown in the query.

The retrieval value (RV) between Q and I is defined as: $RV_Q(I) = \Sigma_{i = 1, \ldots, n} (w_i * sim(q_i, x_i))$, where w_i ($i = 1, 2, \ldots, n$) is the weight specifying the importance of the i-th parameter in the animation description and $sim(q_i, x_i)$ is the similarity between the i-th parameter of the query animation $Q(q_1, q_2, \ldots, q_n)$ and analyzed animation $I(x_1, x_2, \ldots, x_n)$. $sim(q_i, x_i)$ is calculated in different way taking in account if q_i, x_i are symbol, numerical, linguistic values, histograms, attribute relational graphs, pictures, interval values, temporal or spatial representations.

2.3. The ANIMATION display tool

The ANIMATION display tool includes the Render and Event router.

1. **Render.** The render responsibility is to show a scene on the screen. It iterates recursively through scene graph applying relative transformation to each object. The render applies visual properties and renders it on the screen. It is also responsible to track visual properties from parent to its children. The rendering is made through GDI. For smooth animation a double buffering technique is implemented. In the rendering process each node is rendered in current graphics and transformation context. Each node can modify the context. There are two stacks that track these modifications and allow easy restoring of the context in backward iteration in the tree.

2. **Router.** The connection between the node generating the event and the node receiving the event is called a route. Routes are not nodes. The ROUTE statement is a construct for establishing event paths between nodes. ROUTE statements may either appear at the top or bottom level of a scene definition, or inside a node wherever fields may appear. Nodes referenced in a ROUTE statement are defined before the ROUTE statement. The event router is responsible to process all events generated by the scene nodes or invoked by user iteration. It is also responsible to avoid loops and fan-in and fan-out. The Render process scene self-behavior is based on time by notifying each node from scene graph. For time change the Render fires event that forces scene change**.**

Execution model. All events generated during a given event cascade are assigned the same timestamp as the initial event, since all are considered to happen simultaneously. Some sensors generate multiple events. In these cases, each event generated initiates a different event cascade with identical timestamps.

3. AN EXAMPLE FOR THE ANIMATION SYSTEM FUNCTIONING

Let us develop animation with the following animation frames (Fig. 3.)

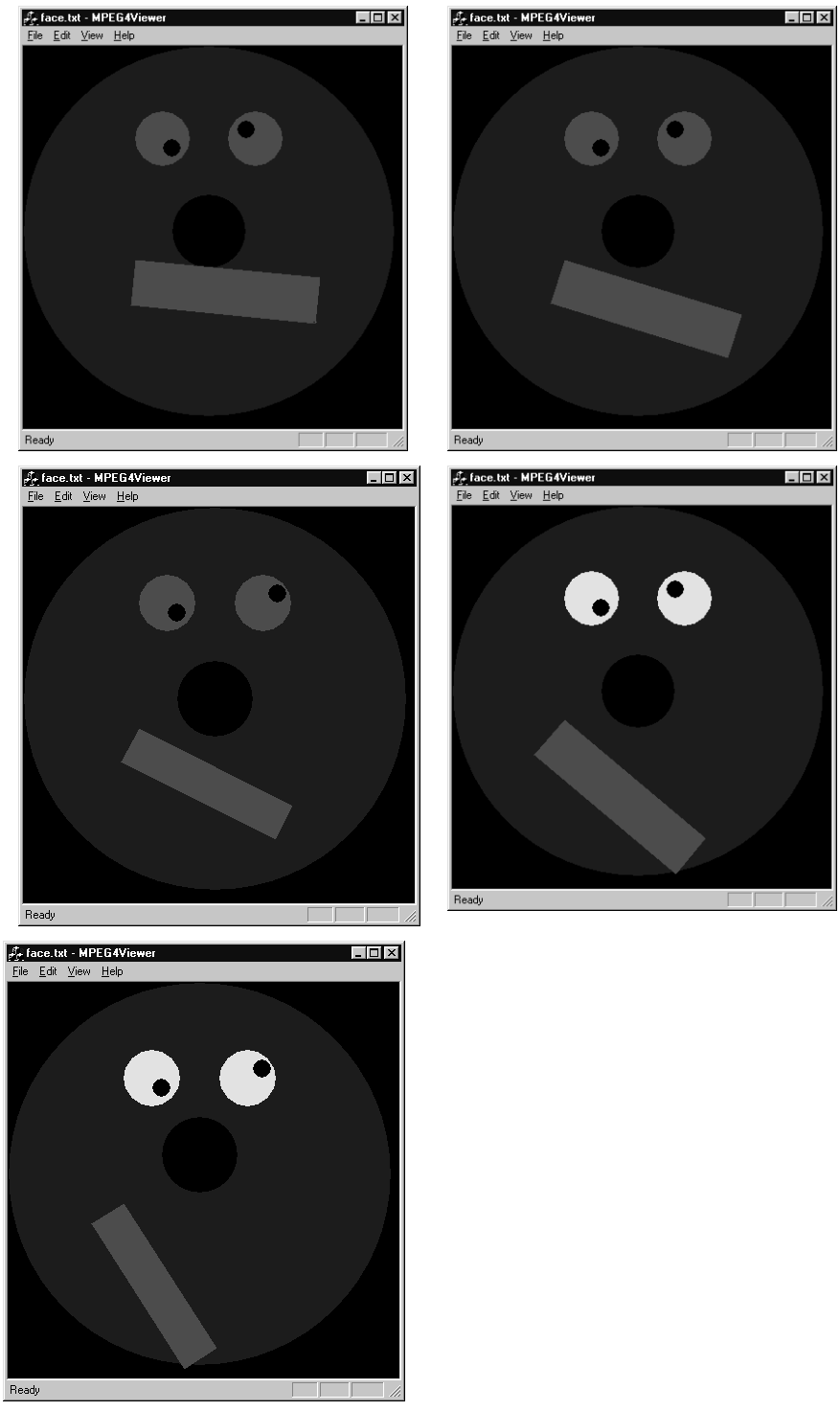

Figure 3. An example of animation frames

3.1.The ANIMATION editor

A portion of the developed animation description for these frames is given below.

```
Group2D {
	children [
		transform2D {
			translation 200 200
			rotationAngle 0
			children [
				Shape {
					appearance appearance {
						material Material2D {
							diffuseColor 0 0 1
							filled TRUE
						}
					}
					geometry Circle {
						radius 200
					}
				}
			]
		}
		transform2D {
			translation 150 100
			children [
				Shape {
					appearance appearance {
						material DEF MTREYE Material2D {
							diffuseColor 1 1 0
							filled TRUE
						}
					}
					geometry Circle {
						radius 30
					}
				}
			]
		}
		transform2D {
			translation 160 110
			children [
				Shape {
					appearance appearance {
						material Material2D {
							diffuseColor 0 0 0
							filled TRUE
						}
					}
					geometry Circle {
						radius 10
					}
				}
			]
		}
		transform2D {
			translation 250 100
			children [
				Shape {
					appearance appearance {
						material DEF MTLEYE Material2D {
							diffuseColor 1 1 0
							filled TRUE
						}
					}
					geometry Circle {
						radius 30
					}
```

```
                        }
                ]
        }
        DEF TRZEN1 transform2D {
                translation 240 90
                children [
                        Shape {
                                appearance appearance {
                                        material Material2D {
                                                diffuseColor 0 0 0
                                                filled TRUE
                                        }
                                }
                                geometry Circle {
                                        radius 10
                                }
                        }
                ]
        }
        DEF TRNOSE transform2D {
                translation 200 180
                children [
                        Shape {
                                appearance appearance {
                                        material DEF NOSE Material2D {
                                                diffuseColor 0 0 0
                                                filled TRUE
                                        }
                                }
                                geometry Circle {
                                        radius 40
                                }
                        }
                ]
        }
        DEF TRMOUTH transform2D {
                translation 120 230
                rotationAngle 0.5
                children [
                        Shape {
                                appearance appearance {
                                        material Material2D {
                                                diffuseColor 1 0 0
                                                filled TRUE
                                        }
                                }
                                geometry def Rc rectangle {
                                        size 200 50
                                }
                        }
                ]
        }
    ]
}

DEF CI ColorInterpolator {
    key [.50,  1.1]
    keyValue [  1 0 0 , 1 1 0 ]
}

DEF CINOSE ColorInterpolator {
    key [.50,  1.1]
    keyValue [   0 0 0 , 1 1 1 ]
}

DEF TS timeSenSor {
    cycleInterval 3
}

DEF TSNOSE timeSenSor {
```

```
       cycleInterval 1.1
}

DEF TSZEN timeSenSor {
       cycleInterval 1.5
}

DEF P2DZEN1 Position2DInterpolator {
       key [.20, .40,.60,.80,1.]
       keyValue [  240 90 , 255 85, 265 90, 255 85,240 90]
}

DEF RECTSZ Position2DInterpolator {
       key [.20, .40,.60,.80,1.]
       keyValue [  200 50 , 190 45, 180 40, 190 45,200 50]
}

DEF PIn Position2DInterpolator {
       key [.50,  1.1]
       keyValue [  200 180 , 200 200 ]
}

ROUTE TS.fraction_changed TO CI.set_fraction

ROUTE TSZEN.fraction_changed TO P2dZen1.set_fraction
ROUTE P2dZen1.value_changed TO trzen1.translation

ROUTE TSZEN.fraction_changed TO rectsz.set_fraction
ROUTE rectsz.value_changed TO RC.size

ROUTE TS.fraction_changed TO TRMOUTH.rotationangle

ROUTE TSNOSE.fraction_changed TO CINOSE.set_fraction
ROUTE TSNOSE.fraction_changed TO PIN.set_fraction

ROUTE CI.value_changed TO MTREYE.diffuseColor
ROUTE CI.value_changed TO MTLEYE.diffuseColor
ROUTE CINOSE.value_changed TO NOSE.diffuseColorr

ROUTE PIN.value_changed TO TRNOSE.translation
```

3.2. The ANIMATION search by contents engine

A search query for the example animation could be: “Find animations which contain a rectangle moving inside a circle. There are also two circles inside the circle and their colors are changed from red to yellow in 5 seconds interval.” The user composes the query drawing the objects: three circles and one rectangle. Then s/he picks the colors for the objects. Subsequently the user places the drawing objects in window to discredit they mutual positions. Finally, the specifications for the timing constraints are given.

3.3.The ANIMATION display tool

After the scene is read, the time sensor starts to generate **fraction_changed** event. The value of the event is in interval [0.0 1.0]. This event is routed to the input event set_fraction of **ColorInterpolator**. Based on event’s value the interpolator raises **value_changed** event, which then is routed to the value of **diffuseColor** of the Shape material. The data type of event **value_changed** is SFColor. When **TimeSensor** fraction value is in interval [0.0 .5) (first 5 seconds of cycle interval) the interpolator’s event value is red color and in the next 5 seconds of the cycle the fraction is in interval [.5 1] and the value is yellow color. This color value is set by routing mechanism to the **Material2D** MT node, which causes color change of the **Shape**. This results is in change of the color every 5 seconds.

4. CONCLUSIONS

The realized system is an example of MPEG-4 terminal application. It is written in C++, (MS Visual C++ 5.0 compiler), using MFC framework library under Windows 95/NT platform. It is in state of testing.

The ANIMATION system allows:

1. Flexibility to existing application and makes it a good place to start development of a MPEG-4 programs. Due to the separation of the scene parser and render a data delivery mechanism can be easily replaced with layers, defined by standard stream without changes to scene representation routines.
2. The existing MPEG-4 parser can be used as a base for development of MPEG-4 scene definition compiler. Due to similarity of textual MPEG-4 definition and VRML the same parser can be used for VRML reader or even for a development of a converter from VRML to MPEG-4 applications. MPEG-4 standard includes all features defined by VRML.

The main advantages of the ANIMATION system are:

1. The creation of the animation content is done automatically during the animation creation.
2. The system allows retrieval by animation contents on Internet animations.
3. Each implemented feature is encapsulated and isolated of the other program parts, which gives possibilities for easy system enlargements.

The system can be used for different purposes including e-commerce [7].

REFERENCES

1. Cambridge Animation Systems - http://www.animo.com/casweb/naviga.htm.
2. Gibson J.D. (edt.), "Multimedia communications: directions and innovations", San Diego: Academic Press, 2001.
3. ISO/IEC FCD International Standard 14496-1 (Systems MPEG-4 Systems), 1998.
4. MPEG Home Page - http://mpeg.telecomitalialab.com/index.htm.
5. Parent R., "Computer Animation: Algorithms and Techniques", Morgan Kaufmann, 2001.
6. Pocock L., Rosebush J., "The Computer Animator's Technical Handbook", Morgan Kaufmann, San Francisco, 2001.
7. Puri R., Schmidt L., Huang Q., "On Feasibility of MPEG-4 for Multimedia Integration for E-Commerce", *IEEE International Conference on Multimedia and Expo*, 2000: 75-79.