

# The Specification and Design of a Prototype 2-D MPEG-4 Authoring Tool

**Deon Walter Viljoen**

Submitted in partial fulfilment of  
the requirements for the degree of  
**MAGISTER SCIENTIAE**  
in the Faculty of Science at the  
University of Port Elizabeth



February 2003

**Supervisor:** Prof. A.P. Calitz  
**Co-Supervisor:** Mr. N.L.O. Cowley

## **Acknowledgments**

I would like to thank my supervisors André Calitz and Lester Cowley for their valuable advice, encouragement and guidance for the duration of this dissertation. In particular, I wish to express my gratitude for the many hours that were spent in the reading of this document and the many valuable suggestions for improvements that were made.

I would also like to thank the Department of Computer Science and Information Systems for enabling me to conduct this research as well as my colleagues for their support and encouragement.

Finally to Martina Vankova, I would like to express my gratitude for her valuable assistance in the proofreading of this dissertation.

# Table of Contents

<b>Summary.....</b>	<b>vii</b>
<b>List of Figures.....</b>	<b>viii</b>
<b>List of Tables.....</b>	<b>x</b>
<b>List of Abbreviations.....</b>	<b>xi</b>
<b>Chapter 1: Introduction.....</b>	<b>1</b>
1.1 Background.....	1
1.2 Situation of Concern.....	2
1.3 Problem Statement.....	3
1.4 Objectives and Scope.....	4
1.5 Risks Associated with the Project.....	5
1.6 Dissertation Outline.....	6
1.7 Dissertation Roadmap.....	7
<b>Chapter 2: Overview of MPEG-4.....</b>	<b>9</b>
2.1 Introduction.....	9
2.2 Scope and Features of MPEG-4.....	10
2.2.1 Coding of Audio Visual Objects.....	11
2.2.2 Composition and Streaming of Media Objects.....	12
2.3 The Major Functionalities of MPEG-4.....	16
2.3.1 MPEG-4 Systems.....	16
2.3.2 MPEG-4 Visual.....	18
2.3.3 MPEG-4 Audio.....	20
2.3.4 MPEG-4 DMIF.....	21
2.4 MPEG-4 Profiles.....	22
2.5 The MPEG Research Process.....	23
2.6 Overview of Selected MPEG-4 Research Projects.....	24
2.6.1 Research by MPEG .....	25
2.6.2 Research by Other Research Groups.....	26
2.6.2.1 Research by SONG.....	26
2.6.2.2 Research by ENST.....	27
2.6.2.3 Research by IBM.....	27
2.6.2.4 The MPEG4IP Project.....	27
2.6.2.5 Research by MTrec .....	28

2.7 Application Areas of MPEG-4.....	28
2.8 Technologies Similar to MPEG-4.....	30
2.8.1 Extensible 3-D .....	30
2.8.2 Synchronized Multimedia Integration Language.....	30
2.8.3 Windows Media 9 Series.....	31
2.8.4 Macromedia Flash MX.....	31
2.9 Overview of MPEG Standards.....	32
2.9.1 The MPEG-7 Standard.....	33
2.9.2 The MPEG-21 Standard.....	35
2.10 Summary.....	36
<b>Chapter 3: Authoring Tools.....</b>	<b>39</b>
3.1 Introduction.....	39
3.2 Authoring Paradigms.....	40
3.2.1 The Scripting Language Paradigm.....	41
3.2.2 The Iconic/Flow Control Paradigm.....	41
3.2.3 The Frame Paradigm .....	42
3.2.4 The Card/Scripting Paradigm .....	43
3.2.5 The Cast/Score/Scripting Paradigm.....	43
3.2.6 The Hierarchical Object Paradigm.....	45
3.2.7 The Hypermedia Linkage Paradigm.....	45
3.3 Extant Systems Analysis.....	46
3.3.1 Object, Views and Interaction Design.....	46
3.3.2 Task Analysis.....	48
3.3.3 MPEG-4 ToolBox.....	51
3.3.3.1 Creating a New Scene.....	52
3.3.3.2 Adding Objects to the Scene.....	54
3.3.3.3 Adding Events to Objects.....	56
3.3.4 Internet Scene Assembler (ISA).....	57
3.3.4.1 Creating a New Scene.....	58
3.3.4.2 Adding Objects to the Scene.....	59
3.3.4.3 Adding Events to Objects .....	60
3.3.5 Macromedia Flash.....	63
3.3.5.1 Creating a New Scene.....	63

3.3.5.2 Adding Objects to the Scene.....	65
3.3.5.3 Adding Events to Objects .....	66
3.4 Conclusions of Study.....	66
3.4.1 Selecting an Appropriate Authoring Paradigm.....	66
3.4.2 Creating a New Scene.....	67
3.4.3 Adding Objects to the Scene.....	68
3.4.4 Adding Events to Objects.....	69
3.5 Summary.....	69
<b>Chapter 4: The MPEG-4 Framework.....</b>	<b>71</b>
4.1 Introduction.....	71
4.2 Binary Format for Scenes.....	72
4.2.1 Differences Between MPEG-4 and VRML.....	72
4.2.2 Brief Introduction to BIFS Programming.....	74
4.2.3 Extensible MPEG-4 Textual Format (XMT).....	82
4.3 MPEG-4 Video Compression.....	84
4.3.1 MPEG Video Compression.....	85
4.3.2 The MPEG-4 Video Algorithm .....	88
4.4 MPEG-4 Image Compression.....	90
4.5 MPEG-4 Audio Compression.....	91
4.5.1 MPEG Audio Compression.....	91
4.5.2 The MPEG-4 Audio Algorithm.....	92
4.6 The MPEG-4 File Format.....	95
4.7 Creating an MPEG-4 Scene.....	96
4.8 Summary.....	98
<b>Chapter 5: Specification and Design.....</b>	<b>100</b>
5.1 Introduction.....	100
5.2 System Specification.....	101
5.2.1 System Functionality.....	101
5.3 System Design.....	105
5.3.1 The Objects in SceneBuilder.....	105
5.3.2 The Views Associated with Objects.....	108
5.3.3 The Tasks Associated with Each View.....	110
5.3.4 The Interaction Style for SceneBuilder.....	111

5.3.5 A Draft Interface Design.....	112
5.3.6 Development Tools.....	113
5.4 Summary.....	114
<b>Chapter 6: Implementation.....</b>	<b>116</b>
6.1 Introduction.....	116
6.2 Brief Discussion of Class Implementations.....	117
6.2.1 The Object Class.....	117
6.2.2 The ObjectStore Class.....	118
6.2.3 The ObjectEditor Class.....	119
6.2.4 The EventsEditor Class.....	120
6.2.5 The SceneTree Class.....	123
6.2.6 The CodeGenerator Class.....	124
6.3 Scene Based Editing.....	126
6.4 Adding Authoring Templates to SceneBuilder.....	126
6.5 Implementation of the Interface.....	128
6.6 Problems Encountered.....	129
6.7 Summary.....	130
<b>Chapter 7: Testing and Evaluation.....</b>	<b>132</b>
7.1 Introduction .....	132
7.2 Methods Used for Testing and Evaluation .....	132
7.2.1 Case Study .....	133
7.2.2 User Evaluation.....	138
7.3 Conclusions of Testing and Evaluation.....	141
7.4 Related Research Done on MPEG-4 Authoring Tools.....	142
7.4.1 The GSRT Project.....	142
7.4.2 MTrec MPEG-4 Toolkit.....	143
7.4.3 The ENST MPEG-Pro Authoring System.....	145
7.4.4 ETRI.....	147
7.5 Comparison of MPEG-4 Authoring Systems.....	147
7.6 Summary.....	149
<b>Chapter 8: Conclusions and Future Research.....</b>	<b>150</b>
8.1 Introduction.....	150
8.2 Summary of Research and Implementation.....	150

8.3 Future Research.....	154
8.4 Conclusions .....	155
<b>Bibliography.....</b>	<b>157</b>
<b>Appendix A : Profiles in MPEG-4 .....</b>	<b>162</b>
<b>Appendix B : Installation Instructions .....</b>	<b>169</b>

## Summary

The purpose of this project was the specification, design and implementation of a prototype 2-D MPEG-4 authoring tool. A literature study was conducted of the MPEG-4 standard and multimedia authoring tools to determine the specification and design of a prototype 2-D MPEG-4 authoring tool. The specification and design was used as a basis for the implementation of a prototype 2-D MPEG-4 authoring tool that complies with the Complete 2-D Scene Graph Profile.

The need for research into MPEG-4 authoring tools arose from the reported lack of knowledge of the MPEG-4 standard and the limited implementations of MPEG-4 authoring tools available to content authors. In order for MPEG-4 to reach its full potential, it will require authoring tools and content players that satisfy the needs of its users. The theoretical component of this dissertation included a literature study of the MPEG-4 standard and an investigation of relevant multimedia authoring systems. MPEG-4 was introduced as a standard that allows for the creation and streaming of interactive multimedia content at variable bit rates over high and low bandwidth connections. The requirements for the prototype 2-D MPEG-4 authoring system were documented and a prototype system satisfying the requirements was designed, implemented and evaluated. The evaluation of the prototype system showed that the system successfully satisfied all its requirements and that it provides the user with an easy to use and intuitive authoring tool.

MPEG-4 has the potential to satisfy the increasing demand for innovative multimedia content on low bandwidth networks, including the Internet and mobile networks, as well as the need expressed by users to interact with multimedia content. This dissertation makes an important contribution to the understanding of the MPEG-4 standard, its functionality and the design of a 2-D MPEG-4 Authoring tool.

**Keywords:** MPEG-4; MPEG-4 authoring; Binary Format for Scenes (BIFS); authoring paradigm; MP4.



## List of Figures

Figure 1.1: Dissertation roadmap.....	8
Figure 2.1: Hierarchy tree for an MPEG-4 scene.....	13
Figure 2.2: A single MPEG-4 video object.....	13
Figure 2.3: The complete MPEG-4 scene .....	13
Figure 2.4: The MPEG-4 system layer model.....	15
Figure 3.1: Macromedia Authorware: an example of an iconic/flow paradigm.....	42
Figure 3.2: Quest: an example of a frame paradigm.....	43
Figure 3.3: Macromedia Director: an example of a cast/score/scripting paradigm.....	44
Figure 3.4: mTropolis: an example of a hierarchical object paradigm.....	45
Figure 3.5: The OVID methodology.....	48
Figure 3.6: MPEG-4 ToolBox.....	53
Figure 3.7: Changing the background colour of the scene.....	54
Figure 3.8: Changing the properties of an object.....	55
Figure 3.9: Adding a video to the object.....	55
Figure 3.10: Selecting an interpolator.....	56
Figure 3.11: Selecting a sensor.....	57
Figure 3.12: A new scene in ISA.....	59
Figure 3.13: Editing objects in a scene.....	60
Figure 3.14: Adding an event to an object.....	61
Figure 3.15: Selecting an event.....	62
Figure 3.16: Selecting a target for an event.....	62
Figure 3.17: Opening screen of Macromedia Flash.....	64
Figure 3.18: The Flash object library.....	65
Figure 4.1: General form of BIFS code.....	75
Figure 4.2: BIFS Code for a filled yellow circle.....	76
Figure 4.3: BIFS code for displaying text.....	77
Figure 4.4: BIFS code for displaying an image.....	78
Figure 4.5: The object descriptors for a MPEG-4 application.....	79
Figure 4.6: Object descriptor for an image object.....	80
Figure 4.7: A more complicated hierarchical scene tree.....	81
Figure 4.8: A more complicated example of BIFS code.....	82
Figure 4.9: Interoperability of XMT.....	84

Figure 4.10: The MPEG video algorithm.....	86
Figure 4.11: The MPEG-4 video encoder algorithm.....	89
Figure 4.12: Block diagram of the MPEG audio algorithm.....	92
Figure 4.13: MPEG-4 audio coding algorithms.....	93
Figure 4.14: Creating MPEG-4 content.....	97
Figure 5.1: Object class diagram for 2-D MPEG-4 authoring tool.....	107
Figure 5.2: Object class diagram with views.....	109
Figure 5.3: Draft interface design for a 2-D MPEG-4 authoring tool.....	113
Figure 6.1: The ObjectStore.....	118
Figure 6.2: The ObjectEditor.....	120
Figure 6.3: Adding and editing Object Events.....	121
Figure 6.4: The EventsEditor dialog box.....	122
Figure 6.5: Setting an event to move to another scene .....	122
Figure 6.6: The SceneTree.....	123
Figure 6.7: State transition diagram for the CodeGenerator class.....	125
Figure 6.8: Interface of SceneBuilder.....	129
Figure 7.1: The home page of the ICC Cricket World Cup web site.....	134
Figure 7.2: The venues of fixtures of the ICC Cricket World Cup .....	135
Figure 7.3: Creating the application with SceneBuilder .....	136
Figure 7.4: MPEG-4 Cricket World Cup application played by a MPEG-4 player.....	137
Figure 7.5: The Main Window of the MPEG-4 ToolBox.....	143
Figure 7.6: The Main Window of the MTREC Toolkit.....	144
Figure 7.7: The MPEG-Pro Authoring System.....	146

## List of Tables

Table 2.1: The parts of the MPEG-4 standard.....	16
Table 2.2: A subset of the MPEG-4 Profiles.....	23
Table 2.3: The organisational subgroups in MPEG.....	24
Table 4.1: The differences between BIFS and VRML.....	73
Table 4.2: Summary of MPEG audio layers.....	91
Table 5.1: Tasks related to the views of the object class diagram.....	111

## List of Abbreviations

<b>AAC</b>	Advanced Audio Coder
<b>ACE</b>	The Advanced Coding Efficiency
<b>API</b>	Application Programming Interface
<b>ARTS</b>	The Advanced Real-Time Simple Profile
<b>BIFS</b>	The BInary Format for Scenes
<b>BSAC</b>	Bit-Sliced Arithmetic Coding
<b>B-VOP</b>	Bi-directionally Predicted Video Object Plane
<b>CBT</b>	Computer-Based Training
<b>CD</b>	Compact Disk
<b>CELP</b>	Code Excited Linear Prediction
<b>CIF</b>	Common Intermediate Format
<b>DCT</b>	Discrete Cosine Transform
<b>DDL</b>	Description Definition Language
<b>DMIF</b>	Delivery Multimedia Integration Framework
<b>DVD</b>	Digital Versatile Disk
<b>ENST</b>	Ecole Nationale Supérieure des Télécommunications
<b>FAP</b>	Facial Animation Parameter
<b>FTP</b>	File Transport Protocol
<b>GOP</b>	Group of Pictures
<b>GSRT</b>	General Secretariat for Research and Technology
<b>HVXC</b>	Harmonic Vector eXcitation Coding
<b>IBM</b>	International Business Machines
<b>IEC</b>	International Electrotechnical Commission
<b>IP</b>	Internet Protocol
<b>IPI</b>	Intellectual Property Identification
<b>IPMP</b>	Intellectual Property Management and Protection
<b>ISA</b>	Internet Scene Assembler
<b>ISO</b>	International Organization for Standardization
<b>ITU-T</b>	International Telecommunication Union
<b>JVT</b>	Joint Video Team
<b>MAUI</b>	Mobile Audio Internetworking Profile
<b>MPEG</b>	Moving Pictures Experts Group
<b>MPEG-J</b>	Java APIs
<b>MTrec</b>	The Multimedia Technology Research Center
<b>NADIB</b>	Narrow band Audio DIgital Broadcasting

<b>OD</b>	Object Descriptor
<b>OVID</b>	Objects, Views, and Interaction Design
<b>QCIF</b>	Quarter Common Intermediate Format
<b>QoS</b>	Quality of Service
<b>RDF</b>	Resource Description Framework
<b>SAOL</b>	Structured Audio Orchestra Language
<b>SASL</b>	Structured Audio Score Language
<b>SGML</b>	Standard Generalised Markup Language
<b>SMIL</b>	Synchronized Multimedia Integration Language
<b>SoNG</b>	portalS Of Next Generation
<b>TTS</b>	Text To Speech
<b>Twin VQ</b>	Twin Vector Quantisation
<b>UCD</b>	User-Centred Design
<b>UML</b>	Unified Modeling Language
<b>VOP</b>	Video Object Plane
<b>VRML</b>	Virtual Reality Modeling Language
<b>W3C</b>	World Wide Web Consortium
<b>WYSIWYG</b>	What You See Is What You Get
<b>X3D</b>	Extensible 3-D
<b>XML</b>	Extended Markup Language
<b>XMT</b>	Extensible MPEG-4 Textual Format

# Chapter 1: Introduction

## 1.1 Background

In 1889 the first attempt was made to synchronise moving pictures with sound. Invented by Thomas Edison and William Dickson, the kinetophonograph reproduced brief flickering images with barely synchronised sound [Koe1999]. The kinetophonograph was however never commercially successful. Robust sound cinema would not appear for another 40 years.

In the last twenty-five years, the world of audio and video has undergone rapid changes [Chi2001]. The development of the Compact Disk (CD) and the Digital Versatile Disk (DVD) technologies provided the mass market with a delivery medium for high quality audio and video. In the early 1980s, the first videoconferencing system was developed and in the early 1990s the Internet Protocol (IP) was released for public use [Chi2001]. In addition to the Internet, other low bit-rate communication networks, specifically used for wireless communications, have been introduced.

In order to enable the use of the technologies mentioned in the previous paragraph, the analogue audio and video media had to be encoded and compressed into a digital format. To this end several standards were and are currently being developed. The MPEG-1 [Chi1996] standard was created to allow video and audio to be stored on compact disk. In order to convert analogue television signals to a compact digital form, MPEG-2 [Chi2000] was created which proved to be the enabling technology for satellite television and DVDs. Both standards were developed by the Moving Pictures Experts Group (MPEG).

Low bit-rate communication mediums, such as used by the Internet, can provide audio-visual services by streaming data at very low bit-rates and low quality, or by introducing a delay while the media is downloaded. The most important factor inhibiting the development of interactive audio visual services over low bit-rate mediums is the limited

bandwidth. The low bandwidth arena has tremendous potential, but in some ways it bears a resemblance to the kinetophonograph. Both technologies produce video and audio sequences that have poor synchronisation, poor quality and irregular playback of media.

In order to make interactive multimedia a reality on low bit-rate mediums, MPEG created a new standard known as MPEG-4. MPEG-4 addresses every aspect of the multimedia creation, delivery and representation processes over low and high bandwidths. For the MPEG-4 standard to reach its full potential, the prospective "users" of the standard must be provided with applications and systems that allow easy access to the functionality provided by MPEG-4.

MPEG-4 provides the functionality required to create multimedia applications that can be streamed over low and high bandwidths. The complexity of these applications may range from a single video application that offers no interactivity, to an interactive application that consists of multiple media types.

This chapter will discuss the situation of concern that identified the need for research into MPEG-4 authoring tools. A single sentence description of the research project will be formulated and the scope and objectives of the project will be summarised. Finally, an overview of the dissertation will be presented.

## **1.2 Situation of Concern**

At the start of this project, in January 2001, there was only one MPEG-4 authoring tool available to the public [DKR+2000]. This system was developed to create 3-D multimedia applications. Therefore there were no 2-D MPEG-4 authoring systems available to content authors.

In general there was a very poor understanding of the MPEG-4 standard and limited implementations making use of it. No matter how powerful a multimedia standard may be,

if the content authors and content users are not provided with easy access to implementations of MPEG-4, the standard will more than likely fail to reach its full potential.

For MPEG-4 to be adopted as a multimedia standard of choice, by content authors and users, it needs high quality media encoders and decoders, authoring tools and media players. Therefore, the aim of this dissertation is the specification and construction of a 2-D MPEG-4 authoring tool.

### 1.3 Problem Statement

A single sentence description of the project reads as follows: *The specification and construction of a prototype 2-D MPEG-4 compliant authoring tool which will allow computer literate users, having a working knowledge of the multimedia content of MPEG-4, to construct interactive MPEG-4 scenes.*

In the problem statement, the user is described as a content author who has working knowledge of the multimedia content of MPEG-4. This implies that the user can create and manipulate the media elements, (images, video, audio, etc.) that are used in MPEG-4 applications. An authoring system (discussed in Chapter 3) allows the user to create multimedia applications by assembling and manipulating media elements and defining their behaviour. The creation of media elements is not part of the functionality of an authoring system. Another important fact about the user is that he or she should be computer literate but *need not posses programming skills*. The problem statement further defines the functionality of the tool to produce basic MPEG-4 scenes. The next section will discuss the objective and scope of the project, which will give a more precise specification of the functionality of the tool.



## 1.4 Objectives and Scope

The following objectives were set for the project:

- " Conduct a literature study on the MPEG-4 standard and MPEG-4 authoring tools;
- " Conduct a survey of current research on MPEG-4;
- " Investigate the design of current authoring tools;
- " Conduct a study of authoring paradigms and discuss the selection of an appropriate paradigm for a MPEG-4 authoring tool;
- " Discuss the functionality of MPEG-4 and the process of creating MPEG-4 content;
- " Provide the specification and design for a 2-D MPEG-4 authoring tool;
- " Implement a prototype 2-D MPEG-4 authoring tool;
- " Evaluate the prototype system; and
- " Report on the findings of the research and derive conclusions.

The scope of the project was selected to balance the large scope of the MPEG-4 standard with the creation of an authoring tool that provides useful functionality for the user. As stated in Section 1.2, research on a 3-D MPEG-4 authoring tool was already completed. The creation of a 3-D system, depending on the scope, may be more complicated than creating a 2-D system. Therefore, it was decided to create a 2-D authoring tool. The scope of the authoring system has been defined as follows:

- " Support 2-D multimedia applications;
- " Support the use of synthetic and natural video;
- " Support the use of audio;
- " Support the use of images in JPEG format;
- " Support the use of geometrical shapes;
- " Support the use of text; and
- " Allow the content author to create applications that allow simple interaction with the end user or consumer.

## 1.5 Risks Associated with the Project

The primary motivations behind initiating the research project were the limited implementations of MPEG-4 applications and the apparent lack of knowledge of the MPEG-4 standard. These factors, that serve as motivation, also pose risks to the project. Implementing an end-to-end MPEG-4 solution, that encodes the media objects, creates the MPEG-4 scenes, compiles the MPEG-4 scripting code and plays interactive MPEG-4 scenes, would have made the scope of the project too large. Therefore, the project was dependent on other MPEG-4 tools. The quality, functionality and availability of those tools would directly influence the success of this project.

Furthermore, the information about MPEG-4 available to the general public was very limited. In January 2001, there were no books published that provided an in-depth technical description of the MPEG-4 standard. At that time, the only version of the MPEG-4 reference code and applications that was available from MPEG, was created in 1998. However, the latest version of MPEG-4 reference code and many technical documents were available to MPEG members only. Unfortunately, the author of this dissertation was unable to become a MPEG member as only representatives of an organisation were allowed to become MPEG members. MPEG-4, as a standard, was still growing and at the beginning of the project several extensions to the standard were in progress, thus creating a continuously changing problem domain.

There were many risks related to the project, however these risks are associated with cutting edge research projects that attempt to create a novel solution to an existing problem. The next section will provide a brief outline of the dissertation.

## 1.6 Dissertation Outline

This dissertation consists of eight chapters. Chapter 1 provides background information on the problem domain, a discussion of the situation of concern, followed by the problem statement. The objectives are then formulated in the context of the problem statement followed by the scope of the authoring system.

Chapter 2 provides an overview of MPEG-4 and selected research involving the MPEG-4 standard. The chapter will discuss the functionality offered by the standard. It will discuss some of the research projects that are relevant to this project. It will also provide an overview of MPEG-4 application areas, technologies similar to MPEG-4 and other MPEG standards.

Chapter 3 describes information that will allow the specification of a tool for the creation of 2-D MPEG-4 scenes. It will discuss various authoring paradigms and authoring tools that may provide information for designing a MPEG-4 authoring tool. The discussion of authoring tools will take on the form of an extant systems analysis. A suitable paradigm for a MPEG-4 authoring tool will be selected and the findings of the extant systems analysis will be discussed.

The aim of Chapter 4 is to provide a more in-depth and technical view of the standard by discussing how it provides the functionality discussed in earlier chapters. It will introduce the MPEG-4 framework by providing an overview of selected items in the MPEG-4 framework. A secondary objective of the chapter is to provide an introduction to the MPEG-4 framework, especially for researchers who wish to conduct further research on this project. Therefore the chapter will, with practical examples, provide an introduction into BIFS programming and the process of creating a MP4 file.

Chapter 5 consists of two sections. The first will discuss the specification of the authoring system. The second will discuss the system design of the authoring system. The chapter will follow, step-by-step, the phases of the software design methodology selected in Chapter 3.

Chapter 6 documents the implementation of the system by following the design specifications from Chapter 5. The implementation of the main system classes, the interface and other aspects of the functionality of the system will also be addressed.

The aim of Chapter 7 is to ascertain whether the authoring system satisfies its requirements and whether the system supports the tasks of its users. The latter part of the chapter will discuss MPEG-4 authoring systems developed by other research institutions during the lifetime of this project. Finally, the authoring system will be compared to these systems.

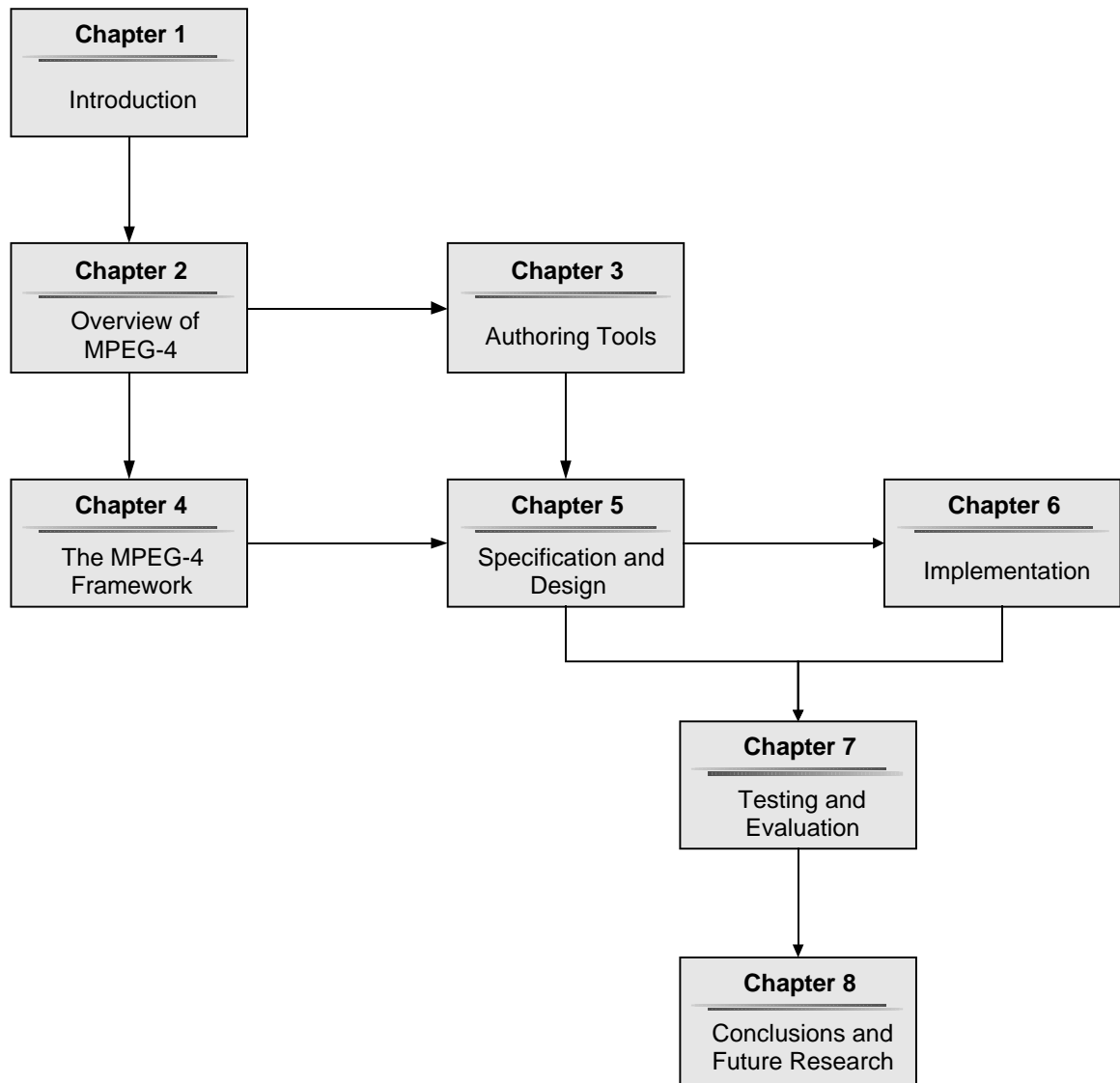
Chapter 8 concludes the dissertation by examining whether the objectives of this project have been achieved. The research conducted in the dissertation will be reviewed and future research projects will be discussed. This is followed by the bibliography and appendices.

## 1.7 Dissertation Roadmap

Figure 1.1 illustrates the flow of information within this dissertation. It provides a graphical roadmap listing the chapters from which the current chapter draws information. For example, Chapter 7 uses the information provided in Chapters 5 and 6. The information provided in Chapter 1 will be expanded in Chapter 2 from where it will be used in Chapter 3 and 4.

It is important to take note that from this stage onward, the dissertation will make the following distinction between users:

- " The user of an authoring system will be referred to as an *author*; and
- " The user of an MPEG-4 application developed by an author will be referred to as a *viewer*.



**Figure 1.1: Dissertation roadmap**

## Chapter 2: Overview of MPEG-4

### 2.1 Introduction

In January 1988, the Moving Pictures Experts Group (MPEG) was established with the sole mandate of developing standards for the coded representation of moving pictures, audio and combinations thereof. MPEG is a working group of ISO/IEC. The International Organization for Standardization (ISO) is a worldwide federation of national standards bodies from some 140 countries. The International Electrotechnical Commission (IEC) is an international standards and conformity assessment body for all fields of electrotechnology. MPEG is made up of approximately 350 experts from some 200 companies and organisations in about 20 countries. These experts form committees that participate in MPEG meetings and discussions. During its existence MPEG have developed several standards, of which MPEG-4 will form the focus of this project. The development of the MPEG-4 standard for the "Coding of audio-visual objects" formally known as ISO/IEC 14496, began in July 1993. Work on the standard started with the main objective of streaming media content, video and audio over low bandwidth networks, especially at bit rates below 64 kbps. The first set of standards, known as Version 1, was approved in October 1998 [Koe2001]. A major extension of the standard, Version 2, was approved early in 2000 [Koe2001].

During the development of the MPEG-4 standard, it became clear that there was a need for the standard to cater for the streaming of multimedia applications over low bandwidth networks and Internet connections. Such networks are characterised by a high degree of variability and flexibility in both the computational capability of terminals and the transmission characteristics of the network. This includes bandwidth and quality of service issues, such as variability in delay and loss of data [GBL+1998]. A terminal is a system that allows for the playback of the MPEG-4 stream to the viewer, for example a television with a MPEG-4 set-top box or a personal computer. MPEG-4 has since evolved into a framework allowing for the creation of multimedia applications with scalable content that may be transmitted over just about any network.

This chapter provides an overview of MPEG-4 and selected research involving the MPEG-4 standard. It will discuss the functionality offered by the standard while a discussion on how the functionality is provided will follow in Chapter 4.

Research on MPEG-4 is a very active area. Apart from the extensions being researched by MPEG, there are also many research projects run by industry and academic institutions. Many of these projects have not yet been completed, or published, or the researchers working in industry do not intend publishing the results of their work in an effort to preserve their competitiveness. This gives rise to the continuously changing landscape that is MPEG-4.

The scope of the MPEG-4 standard and the number of research projects involving MPEG-4 make it impossible to discuss all the research done on the standard. This chapter will therefore discuss some of the research projects that are relevant to this project. Work done on MPEG-4 affects similar technologies and vice versa. Therefore this chapter will also provide an overview of MPEG-4 application areas, technologies similar to MPEG-4 and other MPEG standards.

## **2.2 Scope and Features of MPEG-4**

The scope of the MPEG-4 standard grew considerably over the lifetime of the MPEG-4 project. Work on MPEG-4 started in 1993 during the early days of the World Wide Web. At that time, the focus of the standard was to create media that could be streamed over low bandwidth connections. During the five year development period of Version 1 of MPEG-4, it became clear that the standard should provide more than just another media encoding and decoding standard. The scope of the standard grew to incorporate the representation and delivery process of the encoded media. MPEG-4 then became a standard that allows for the creation of multimedia applications, which may be streamed over low to high bandwidth connections, irrespective of the underlying network transport protocol. The MPEG-4 standard provides the following features for content authors and users [Koe2001]:

1. For authors, MPEG-4 enables the production of reusable and flexible content, the protection of content owner rights and control over the playback process and the behaviour of the content. MPEG-4 also provides the author with a potentially very large market, as it may be used in technologies such as digital television, animated graphics, wireless and mobile applications and web pages.
2. For network service providers, MPEG-4 offers transparent information, which can be interpreted and translated into the appropriate native signaling messages of each network with the help of relevant standards. The foregoing, however, excludes Quality of Service (QoS) considerations, for which MPEG-4 provides a generic QoS descriptor for different MPEG-4 media. The exact translations from the QoS parameters set for each media to the network QoS are beyond the scope of MPEG-4 and are left to network service providers.
3. For end users, MPEG-4 brings the possibility of higher levels of interaction with content. It also brings multimedia to new networks, including those employing relatively low bit rates, for example, wireless and mobile networks.

The technology and concepts that enable MPEG-4 to achieve the above, have been termed the *features* of MPEG-4. These features will be discussed in the next section.

### **2.2.1 Coding of Audio Visual Objects**

The coding algorithm used in MPEG-4 has been extended from MPEG-1 [Chi1996] and MPEG-2 [Chi2000] to include more algorithmic flexibility and improved compression ratios for use in low bandwidth networks. One of the most important features of MPEG-4 is the ability to store media elements as objects. An object is a single media element that may be natural (recorded video) or synthetic (rendered video) and two or three dimensional. These objects are completely independent of each other and are stored and represented in a hierarchical tree with the media objects forming the leaves.



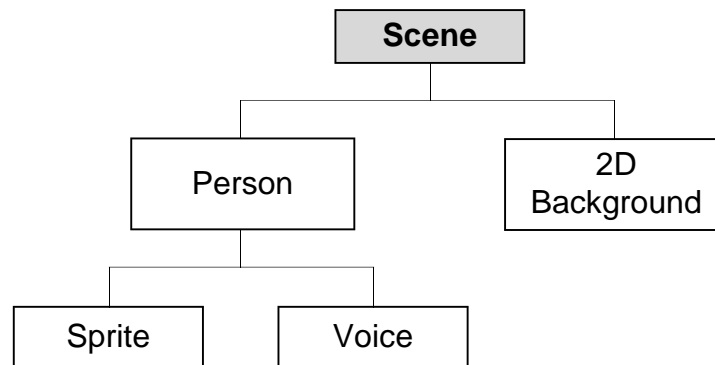
In the area of synthetic media objects, MPEG-4 has special features supporting synthetic human faces and bodies and synthetic audio. For synthetic audio MPEG-4 has standardised two useful technologies. The first is a Text To Speech (TTS) interface that can be synchronised with a synthetic face to produce a lifelike talking head. The second is called "Structured Audio" which provides powerful tools for the creation of synthetic audio.

Another important feature of the object model is that visual objects need not be stored as a rectangular array of pixels. The implication of this feature is that it is not necessary for authors of MPEG-4 multimedia content to have rectangular videos and images in their products. Thus MPEG-4 provides the author with the ability to select an item of irregular shape in a video and extract it from its surroundings as a video object which may then be used in MPEG-4 applications.

Furthermore MPEG-4 provides standard ways of placing objects in scenes and adding interaction or methods to describe the behaviour of the objects. This feature of MPEG-4 will be discussed in more detail in the next section.

### **2.2.2 Composition and Streaming of Media Objects**

As stated before, all MPEG-4 objects are stored in a hierarchical tree with media objects forming the leaves. For example, the scene tree depicted in Figure 2.1 contains a depiction of a person talking which consists of a sprite, a voice and a 2D background, each of which is a media object. The sprite consists of a video object and is separate from the audio track (voice object) associated with the video object. MPEG-4 also allows for the creation of compound media objects, as in the case of *Person* in Figure 2.1, where two media objects have been grouped together to form the compound media object *Person*.



**Figure 2.1: Hierarchy tree for an MPEG-4 scene**

While the hierarchy tree in Figure 2.1 shows how objects in a scene are stored, Figure 2.2 and Figure 2.3 give examples of what the actual on-screen representations of the elements in the tree will be. Figure 2.2 shows a video object at the leaf of the tree. This video object is video only and exists on its own and is separate from the audio track, allowing the author to manipulate the video and audio objects separately. This provides the author with far greater flexibility and reusability. Figure 2.3 depicts a complete scene where the video object from Figure 2.2 has been combined with an audio track and a background 2D image. The audio object is invisible and not shown in Figure 2.3.



**Figure 2.2: A single MPEG-4 video object**



**Figure 2.3: The complete MPEG-4 scene**

A scene as described in Figure 2.3, consists of multiple objects that are combined to form what the user will perceive as a single object. The code that allows the author to combine and manipulate objects into a scene is called The BInary FOrmat for Scenes (BIFS). BIFS describes the spatial and temporal arrangements of the objects in a scene and builds on and

extends several concepts from the Virtual Reality Modeling Language (VRML) [CBM1997]. By doing so, the user may interact with the objects in a scene or move around in a 3-D scene changing his/her viewing or listening points.

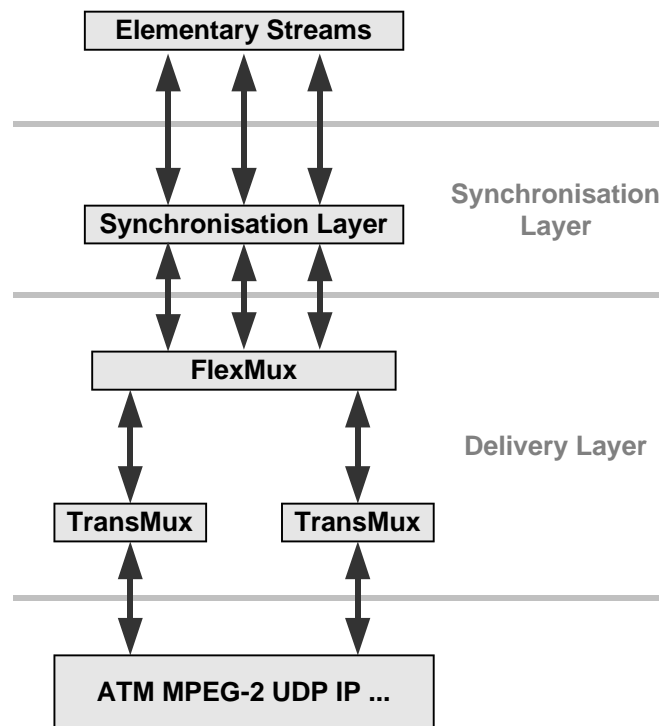
Once a scene has been constructed it would probably be streamed to the viewer. It is in low bandwidth streaming that media objects play a central role. Media objects can be streamed separately from each other, each with its own object descriptors and metadata which are conveyed in one or more elementary streams. The object descriptors carry information about the required resources and the precision of decoder timing information. Furthermore the descriptors may carry information as to the QoS required for transmission and the metadata links meta-information about the content and the intellectual property rights associated with the media objects.

Since every media object is streamed separately, it is possible to stream the objects using different QoS levels. Therefore more bandwidth can be allocated to some objects than to others. Figure 2.3 can be used as an example. If there is sufficient bandwidth, all objects will have the desired QoS, but if the bandwidth is a limiting factor, the important objects (the news presenter and her voice) would be allocated more bandwidth, while the unimportant 2-D background object could be degraded freeing up bandwidth.

Synchronisation of the elementary streams is handled by the synchronisation layer that allows timestamping and the identification of the timestamps of access units within each of the elementary streams. This layer functions independently of the media type and retrieves the object descriptors and metadata associated with the media object, allowing the interpretation of the required QoS and accurate synchronisation between objects. The syntax of this layer is configurable in many ways, allowing use in a broad spectrum of systems [Koe2001].

Figure 2.4 depicts the MPEG-4 system layer model responsible for the synchronised delivery of data while exploiting the different QoS levels of the network. The streaming process can be described in terms of the synchronisation layer and delivery layer containing a two-layer multiplexer.

The first multiplexing layer conforms to the DMIF (Delivery Multimedia Integration Framework), which will be discussed in Section 2.3.4. This multiplex layer is known as the FlexMux and it allows for the grouping of elementary streams, making it possible to group streams with similar QoS needs, thereby reducing the number of connections required for streaming. The use of the FlexMux is optional if the second multiplexing layer provides the required underlying functionality.



**Figure 2.4: The MPEG-4 system layer model**

The second multiplexing layer is the TransMux and it is responsible for allocating elementary streams to transport services satisfying their QoS requirements. The TransMux is an instance of a suitable transport protocol stack such as ATM, IP or MPEG-2. MPEG-4 only specifies the interface to this layer and not the handling of the individual data packets. It is the functionality of the TransMux that makes the underlying network protocol transparent, making it possible to encode a MPEG-4 media file once and play it almost anywhere.

## 2.3 The Major Functionalities of MPEG-4

Section 2.2 provided a brief introduction to the scope and features of MPEG-4. The following section will take a more in-depth look at a selection of the different components, called the *parts* of the standard. MPEG-4 has in essence ten parts of which the first six parts of the standard correspond to those of MPEG-2. The titles of the first five are the same as in the MPEG-2 standard. Although the parts may have similar names, there are significant differences in the content of the parts between MPEG-4 and MPEG-2. The different parts of MPEG-4 and the year the latest version of these parts were or will be completed, are listed in Table 2.1. The parts that play a role in MPEG-4 authoring namely: systems, video, audio and DMIF will be discussed in Sections 2.3.1 – 2.3.4.

<b>Part #</b>	<b>Part Name</b>	<b>Year</b>
1	Systems (Version 2)	2000
2	Visual (Version 2)	2000
3	Audio (Version 2)	2000
4	Conformance Testing (Version 2)	2001
5	Reference Software	1999
6	DMIF (Version 2)	2000
7	Optimised Reference Software	2002
8	4onIP Framework	2002
9	Reference Hardware Description	t.b.a
10	Advanced Video Coding	2003

**Table 2.1: The parts of the MPEG-4 standard**

### 2.3.1 MPEG-4 Systems

The Systems part of the MPEG-4 standard concerns itself amongst other things with describing the relationship between the media objects that form a scene. The relationships are described by BIFS and Object Descriptors (OD's).

BIFS, as stated in Section 2.2.2, is modelled on VRML and is used to control the interaction between the objects and the scene or user. Object Descriptors work at a lower level and they define the relationship between interdependent elementary streams. An example of an interdependent elementary stream would be a video object of a person speaking or an audio object containing the voice of the speaker. Object Descriptors amongst others also return metadata such as the decoder requirements and intellectual property information.

MPEG-4 Systems provide an interface to various aspects of the terminal and network, in the form of Java APIs (MPEG-J) [Koe2000]. MPEG-J specifies an Application Programming Interface (API) allowing Java code to interact with MPEG-4 players, making it possible for content creators to have more control of the playback process. The Java code is sent as a separate elementary stream and runs in a MPEG-J runtime environment in the MPEG-4 terminal. MPEG have defined a set of APIs which may be used to create complex applications with enhanced interactivity. These APIs are:

- " The Scene Graph API which allows the editing of nodes in the scene graph;
- " The Resource Manager API which is used to regulate the performance levels of the playback session;
- " The Terminal Capability API which is used when program execution depends on the capabilities of the terminal;
- " The Media Decoders API which controls the available decoders;
- " The Network API which allows interaction with the network and the use of the Delivery Multimedia Integration Framework (DMIF) interface.

Furthermore MPEG-4 systems provide the following functionality:

- " An event model for triggering events allowing the viewer to interact with the scene;
- " Interleaving of multiple streams into a single stream, including timing information;
- " Storing MPEG-4 data in a file (MP4 format);
- " Transport layer independence. Mappings to relevant transport protocol stacks, like (RTP)/UDP/IP or MPEG-2 transport streams;
- " Text representation with international language support, font and font style selection, timing and synchronisation;

- " The initialisation and continuous management of the receiving terminal's buffers; and
- " Timing identification, synchronisation and recovery mechanisms.

MPEG-4 Systems also provides the industry with reference software. The IM1 software development team has a mandate to develop, integrate and demonstrate Systems software [IM12002]. This means in particular to support the creation of the MPEG-4 reference code and to provide software and test streams for demonstrating MPEG-4 capabilities.

### **2.3.2 MPEG-4 Visual**

The MPEG-4 Visual standard is responsible for the coding of natural and synthetic video or still images used to create MPEG-4 scenes. The compression algorithm caters for both 2D and 3D images and supports compression for bitrates between 5 kbps and 10 Gbps. This includes the compact coding of textures with a range of quality settings from the lower end of the bitrate scale to near lossless encoding.

The algorithm can produce progressive, as well as interlaced video with resolutions ranging from sub-QCIF to beyond HDTV. Progressive encoding allows the transmission of a low quality image which may then be progressively improved by subsequent transmissions. CIF (Common Intermediate Format) has 352x288 luminance pixels. QCIF (quarter CIF) has one-quarter of the CIF resolution and therefore luminance pixels are sent at a resolution of 174x144. CIF and Q-CIF form the H.261 video standard which was approved in 1990 and is widely used in video conferencing [CCI1990]. HDTV or high-definition television has an aspect ratio of 16:9 as compared with 4:3 used in current television standards and promises a near cinematic viewing experience.

The content-based coding of objects allows for the separate decoding and manipulation of video objects and random access to content in video sequences, supporting functions including pause, fast forward and rewind. Each object may be scaled to different levels of

quality required to satisfy the QoS. This is achieved by parsing the bitstream into bitstream layers with different bitrates. The combination of a subset of these layers will produce a meaningful signal with the quality improving with each additional layer used.

Since MPEG-4 is a standard, it only specifies what functionalities an encoder or decoder must have to comply with the MPEG-4 standard. Consequently developers need only implement a subset of the functionality of MPEG-4 to create encoders and decoders of varying complexity or strength. Encoders and decoders rely on complexity scalability for the creation of meaningful bitstreams, irrespective of the level of complexity of the algorithm used. This allows encoders and decoders of varying complexity to communicate. In the case of a less powerful decoder decoding a complex bitstream, only a part of the complex bitstream may be decoded. The complexity of an encoder or decoder is not determined by the developer at will. There are several conformance points or subsets of the standard that may be implemented. These subsets are called *Profiles* and will be discussed in Section 2.4. Apart from the complexity of the encoder or decoder, the standard also allows the scaling of spatial (resolution of object), temporal (frame rate) and quality (bitrate) aspects of MPEG-4 streams.

Error resilience plays an important role in error-prone environments at low bitrates (less than 64 Kbps). MPEG-4 Visual provides methods and tools which address both the bandwidth limited nature and error resiliency aspects of access over standard and wireless networks.

Furthermore MPEG-4 caters for the animation of human face and body objects, easing the creation and use of avatars. MPEG-4 does not specify or standardise the models; only the parameters are standardised and therefore it is possible to code the animation parameters independently of the model. Apart from supporting all the standard functionality for creating and using avatars, MPEG-4 specifies visemes, or visual lip configurations which are equivalent to speech phonemes, allowing the creation of very realistic facial movements during speech [CPO2000].



Finally MPEG-4 Visual supports the coding of 2-D and 3-D Meshes. For 2-D MPEG-4 uses triangular meshes and the 2-D mesh representation of video enables video object manipulation, compression and content-based video indexing. For 3-D Meshes, MPEG-4 extends 2-D meshes and support incremental rendering. Incremental rendering allows the creation of the faces in a mesh by using a subset of the bitstreams improving error resilience and providing a scalable level of detail [GPB+2001].

### 2.3.3 MPEG-4 Audio

MPEG-4 supports a large variety of audio objects, including both natural and synthesised sounds ranging from very low bitrates to high quality multichannel audio. Speech can be encoded using bitrates from 2 kbps up to 24 kbps. Lower bitrates averaging 1.2 kbps are also possible when using variable bitrate coding. General audio signals can be encoded at a bitrate starting at 6 kbps and a bandwidth below 4 kHz; it further includes broadcast quality audio from mono to multichannel.

MPEG-4 supports scalable text to speech encoders with bitrates ranging from 200 bps to 1.2 kbps. MPEG-4 audio provide a rich set of functionalities including following:

- " Lip synchronisation control with phoneme information;
- " Trick mode functionality: pause, resume, jump forward/backward;
- " International language and dialect support for text. (i.e. it can be signalled in the bitstream which language and dialect should be used);
- " International symbol support for phonemes;
- " Support for specifying age, gender, speech rate of the speaker; and
- " Support for conveying facial animation parameter (FAP) bookmarks.

For synthetic audio MPEG-4 provides a Structured Audio Decoder implementation that allows the application of score-based control information to musical instruments described in a special language named the Structured Audio Orchestra Language (SAOL). Additional functionality provided by MPEG-4 audio would be speed and pitch control and scalability of bitrate, bandwidth, error robustness, complexity, etc. as in the case of MPEG-4 Visual.

Bitrate scalability allows a bitstream to be parsed or sliced into bitstreams of lower bitrates such that the combination can still be decoded into a meaningful signal and the quality of the signal depends on the number of the parsed bitstreams combined. The bitstream parsing can occur either during transmission or in the decoder.

Bandwidth scalability occurs when part of a bitstream representing part of the frequency spectrum can be discarded during transmission or decoding, thereby freeing up valuable bandwidth.

Encoder complexity scalability, controlled by profiles, allows encoders of different complexity to generate valid and meaningful bitstreams. Decoder complexity scalability allows a given bitstream to be decoded by decoders of different levels of complexity. The audio quality, in general, is related to the complexity of the encoder and decoder used.

MPEG-4 Audio also implements audio effects that provide the ability to process decoded audio signals with complete timing accuracy to achieve functions for mixing, reverberation, etc.

### **2.3.4 MPEG-4 DMIF**

Delivery Multimedia Integration Framework (DMIF) is responsible for the creation of a transparent application and transportation interface irrespective of whether the peer is a remote interactive peer or broadcast or local storage media and irrespective of the network protocol used. In essence it is the same as File Transport Protocol (FTP). The only difference is that FTP returns data and DMIF returns pointers to data. The pointer returned will contain the location or URL of the file. DMIF also handles the monitoring of Quality of Service, delivered by a network by using three methods: continuous monitoring, specific queries, and QoS violation notification [HPH2001].

## 2.4 MPEG-4 Profiles

As stated before, the MPEG-4 standard has a very large scope. To expect developers to implement systems that conform to the entire standard is simply impossible. In order to allow effective implementations of the standard, MPEG-4 Systems, Visual and Audio have been broken up into subsets that allow complexity scalability in MPEG-4 applications.

These subsets are called Profiles. Through the use of Profiles, a developer can create a MPEG-4 compliant application by implementing a subset of the standard. For each of these Profiles, one or more Levels have been set, restricting the computational complexity even further. To identify the Profile of a system, the following naming convention is used: *Profile@Level*.

To adequately manage the complexity scalability of such a large standard, MPEG has created many profiles for MPEG-4. Table 2.2 lists the main Profiles for MPEG-4 [KSM1999]. A complete list of MPEG-4 Profiles is provided in Appendix A.

<b>Profiles</b>	<b>Features</b>	<b>Application Area</b>
<b>Natural video</b>		
Simple	Error-robust and efficient coding of rectangular video objects.	Mobile communications
Simple Scalable	Added capabilities for temporal and spatial scalability.	Networks with variable bit rates
Core	Support for coding arbitrarily shaped objects.	Internet multimedia applications
N-bit	Pixel depths ranging from 4 to 12 bits.	Surveillance
Main	Interlaced coding, and coding of semitransparent arbitrarily shaped objects (add-on to the core profile).	Broadcast
<b>Audio</b>		
Speech	All MPEG-4 speech coders and text-to-speech interface.	Mobile communications

<b>Profiles</b>	<b>Features</b>	<b>Application Area</b>
Scalable	Scalable coding of speech and music.	Broadcast, Internet
Synthesis	Bitstream-definable and wavetable synthesis.	Computers, games
Main	Superset of all the other profiles, containing all tools for natural and synthetic audio.	Complex virtual worlds
<b>Graphics</b>		
Simple 2D	Placement of one or more visual objects in a scene.	Mobile communications
Complete 2D	Two-dimensional graphics functionalities.	Broadcast, Internet
Complete	Advanced 3D graphical elements.	Complex virtual worlds
<b>Scene</b>		
<b>Graph</b>		
Audio	Scene graph elements for audio-only applications.	Audio broadcast
Simple 2D	Placement of one or more audiovisual objects in a scene, no interaction capabilities	Television broadcast
Complete 2D	Full 2D scene description and full interaction	Internet, interactive television broadcast
Complete	The complete set of scene graph elements for a 3D scene	Dynamic virtual 3D worlds and games

Table 2.2: A subset of the MPEG-4 Profiles

## 2.5 The MPEG Research Process

In order to manage the wide variety of technologies and large scope of MPEG projects, as well as the expertise of its members, requires careful organisation on the part of the MPEG group. To this end, MPEG makes use of the organisational subgroups listed in Table 2.3.

The bulk of the technical work done by the group takes place at MPEG meetings, with several hundred delegates attending each meeting. The members electronically submit contributions to a designated MPEG FTP site. This enables delegates to review their peers' contributions and prepare themselves prior to the meeting.

<b>Group</b>	<b>Description</b>
Requirements	Develops requirements for the standards under development currently, MPEG-4 and MPEG-7.
Systems	Develops standards for the coding of the combination of separately encoded audio, video and related information.
Video	Develops standards for coded representation of video of natural origin.
Audio	Develops standards for coded representation of audio of natural origin.
SNHC	The Synthetic- Natural Hybrid Coding group develops standards for the integrated coded representation of audio and video of natural and synthetic origin.
Multimedia Description	Develops structures for multimedia descriptions. This group only works on MPEG-7.
Test	Develops methods for the execution of subjective evaluation tests of the quality of technology produced by MPEG standards.
Implementation	Evaluates coding techniques so as to provide guidelines to other groups upon realistic boundaries of implementation parameters.
Liaison	Handles relations with bodies external to MPEG.
HoD	The group, consisting of the heads of all national delegations, acts in advisory capacity on matters of general nature.

**Table 2.3: The organisational subgroups in MPEG**

Equally important is the work that is done by ad-hoc groups in between MPEG meetings. These groups communicate via e-mail and rarely meet in person. The outputs produced by the ad-hoc groups are reported at the first plenary of the next MPEG meeting and function as valuable inputs for further deliberation.

## 2.6 Overview of Selected MPEG-4 Research Projects

The large scope of the MPEG-4 standard allows for a wide range of research projects that address many different technologies and disciplines. The research done on MPEG-4 will be broken up into two sections. The first section will focus on work done by MPEG to extend

the standard. The second will deal with research done by other parties. Chapter 7 will discuss, in detail, research on MPEG-4 authoring tools developed during the implementation of this project.

### **2.6.1 Research by MPEG**

At the time of writing, MPEG was in the process of adding extensions to the current functionality of MPEG-4. Research and extensions currently undertaken by MPEG are: AFX, Advanced Video Coding, Audio extensions, IPMP Extension and Multi-User Worlds [Koe2002].

AFX pronounced 'effects' is an acronym for the Animation Framework eXtension. AFX provides the functionality to create powerful synthetic MPEG-4 environments, and uses and builds on existing MPEG-4 tools. It offers the following:

- " Higher-level descriptions of animations (e.g. inverse kinematics);
- " Enhanced rendering (e.g. multi-texturing, procedural texturing);
- " Compact representations (e.g. piecewise curve interpolators, subdivision surfaces);
- " Low bitrate animations (e.g. using interpolator compression and dead-reckoning);
- " Scalability based on terminal capabilities (e.g. parametric surfaces tessellation);
- " Interactivity at user level, scene level, and client-server session level;
- " Compression of representations for static and dynamic tools; and
- " Compression of animated paths and animated models required for improving the transmission and storage efficiency of representations for dynamic and static tools.

Advanced Video Coding is a video codec that forms Part 10 of the MPEG-4 standard. The codec is a joint development between MPEG and the International Telecommunication Union (ITU-T) [ITU2002], known as the Joint Video Team (JVT). The project is based on earlier work by ITU-T named H.26L [MPEG2002], [TSJ+2001]. H.26L has not yet been completed as a standard. When it is completed, its name will change.

Currently, MPEG is working on two projects to extend audio coding efficiency even further. The two projects focus on bandwidth and parametric coding. The bandwidth extension aims to improve the quality of audio signals over current methods, while keeping the signal backwards-compatible. MPEG-4 currently provides parametric coding for general audio signals at low bitrates. The parametric coding extension aims to implement parametric coding for general audio signals using higher bitrates.

Other work currently in progress includes the Intellectual Property Management and Protection (IPMP) extension and Multi-User Worlds. Multi-User Worlds enables sharing of content between multiple users. This means that several users can interact with the content. All events generated by the users, in addition to other changes performed on the content, will be synchronised on all client terminals.

MPEG-4 provides mechanisms for IPMP by supplementing the coded media objects with an optional Intellectual Property Identification (IPI) data set. The data set, if present, is part of an elementary stream descriptor that describes the streaming data associated with a media object. The provision of the data set allows the implementation of mechanisms for audit trail, monitoring, billing, and copy protection.

## **2.6.2 Research by Other Research Groups**

The research done on MPEG-4 by other research groups is simply too diverse and voluminous to be accurately described in this section. Therefore this section will only discuss some of the work that influenced this project and/or that is of general interest.

### **2.6.2.1 Research by SONG**

SoNG (portalS Of Next Generation) intends to investigate, develop and standardise the building blocks for the next generation portals. These building blocks include existing Web technologies, but also 3D computer graphics elements, such as in computer games, intelligent agents embodied in realistic avatars, new user-friendly interfaces and real-time

audio-visual communications. The SoNG project demonstrates how these new technologies are integrated into simple e-commerce applications, allowing for easier and more natural access to resources and services [Song2002].

### **2.6.2.2 Research by ENST**

Ecole Nationale Supérieure des Télécommunications (ENST) is also known as the Higher National School of Telecommunications (a French research institution). The ENST MPEG-4 research team has participated actively in MPEG since 1995, and currently focuses on MPEG-4 Systems [Enst2002]. The unit is actively involved in BIFS research, IM1 reference software and conformance testing. Jean-Claude Dufourd, head of the unit, is the editor of the MPEG-4 Conformance document (ISO/IEC 14496-4) and of the MPEG-4 Reference Software document (ISO/IEC 14496-5) and co-editor of the third BIFS amendment to XMT (ISO/IEC 14496-1:2001 AMD2). The MPEG-Pro authoring system, developed by ENST will be discussed in Chapter 7.

### **2.6.2.3 Research by IBM**

International Business Machines (IBM) is a world leader in development and manufacture of the industry's most advanced information technologies. Multimedia research at IBM encompasses research in technology, tools, and usability along with active participation in the development of standards. IBM researchers have played leading roles in the definition of international standards for compression and storage of facsimile, DVD, MPEG-1 and MPEG-2 digital video. IBM also engineered the convergence of the MPEG-4/SMIL textual representation for object-oriented multimedia presentations. Currently, IBM are leading the development of almost half of the description schemes defined in the MPEG-7 standard [IBM2002].

### **2.6.2.4 The MPEG4IP Project**

MPEG4IP is an open source package designed to enable developers to create streaming servers and clients that are standards-based and free from proprietary



technology [MPE2002]. MPEG4IP, unlike the other projects listed in this section, is not a pure research group; however project members have taken part in several research conferences. The MPEG4IP project provides a standards-based system for encoding, streaming, and playing MPEG-4 encoded audio and video. This was achieved by integrating a number of existing open source packages.

### **2.6.2.5 Research by MTrec**

The Multimedia Technology Research Center (MTrec) based in Hong Kong is one of the leading MPEG-4 research centres in the world. Researchers at MTrec have been engaged in research in this area for several years. During this period the Centre worked on designing various compression algorithms for MPEG-4 and tools for the creation and display of MPEG-4 multimedia scenes.

Research conducted by the centre includes user-assisted video segmentation and MPEG-4 authoring tools. Video segmentation is the process of separating moving objects from video sequences [KJK+2001]. The segmented video may then be used in a MPEG-4 scene as an arbitrarily shaped video object. The research on MPEG-4 authoring tools will be discussed in Chapter 7.

## **2.7 Application Areas of MPEG-4**

The very large scope of MPEG-4 indicates that the standard allows for the creation of a wide variety of applications. In general, MPEG-4 applications can be used for the following application areas or any combination thereof:

- " Broadcast or streaming of multimedia content over at low and high bandwidths;
- " Interactive multimedia applications;
- " Playback of 2-D/3-D interactive multimedia applications from any source;
- " Low bandwidth voice communication; and
- " Creation of interactive 3-D wolds;

The most obvious use for MPEG-4 is streaming video from websites and local storage. Another possibility is the distribution of full length movies on a 700MB CD. A movie that would normally take up 9GB on a DVD can now be compressed to 700MB with little quality loss. MPEG-4 also has the ability, unlikely as it may be, to replace many of the competing proprietary formats used by Apple, Microsoft and Real Networks with one unified standard. The following paragraphs will discuss a few examples of MPEG-4 applications.

The low bandwidth streaming provided by MPEG-4 makes it possible to stream multimedia over wireless networks. As a direct result of the developments in low bandwidth streaming multimedia-capable cellular telephones are emerging. Samsung already has developed an operational phone on a third-generation network. The Samsung MPEG-4 Video Phone displays up to 16 million colours with a one million pixel resolution, and allows viewing of video at 15 frames per second. Video capability will still depend on when service providers can deliver the necessary infrastructure [Rub2001].

Currently MPEG-2 is the preferred technology for satellite television. By using the MPEG-4 Advanced Efficiency Encoding Profile, satellite broadcast streams, currently delivered at 2.5 to 3 Mbps, can be reduced to 1.6 Mbps [Yos2001].

MPEG-4 provides the author with new tools with which he or she can develop solutions for a wide variety of user needs. An innovative use for MPEG-4 is the implementation of a video phone for the deaf, called Lip Telephone [SS2001]. The system makes use of a synthetic head that with the use of Facial Expression Parameters (FAP) produces lifelike lip movements that allows a user to read the lips of the avatar.

## 2.8 Technologies Similar to MPEG-4

MPEG-4, as stated before, has a very large scope that brings it into competition with other standards and technologies ranging from streaming video to interactive multimedia applications. This section will give a brief overview of some similar and competing standards and technologies.

### 2.8.1 Extensible 3-D

Extensible 3-D (X3D) developed by Web3D is an open standard geared to enabling the deployment of visually rich 3-D graphics in applications ranging from lightweight web clients to high-performance 3-D broadcast solutions [X3D2002]. X3D is the latest step in the evolution of VRML toward more compact implementations with improved interoperability with other APIs. The standard was launched in August 2001 and remains fully backward compatible with VRML 97.

The MPEG-4 standard based some of its concepts on VRML. As stated previously, BIFS is an extension of VRML. Another similarity would be the hierarchical scene graph with its architecture heavily based on the scene graph representation used by VRML 97.

Currently the Web3D Consortium is working with the MPEG-4 group. The intention of the collaboration is for the X3D standard to form the core of MPEG-4's ongoing 3-D integration activities [BCL2000]. Work is also being done on tightly integrating X3D with the Extended Markup Language (XML).

### 2.8.2 Synchronized Multimedia Integration Language

Synchronized Multimedia Integration Language (SMIL) specifies a format for integrating independent multimedia objects into a single multimedia presentation, with coherent

temporal and spatial attributes. Developed by the World Wide Web Consortium (W3C) SMIL provides techniques for defining the temporal and spatial attributes of multimedia objects, as well as the use of hyperlinks [W3C2001].

The major differences between SMIL and MPEG-4 are that SMIL does not support 3-D and that the media objects are in separate URLs whereas a MPEG-4 file can contain all the media objects.

### **2.8.3 Windows Media 9 Series**

MPEG-4 has the potential to replace all the different media encoder and decoder standards with one standard, making MPEG-4 the universal standard. Working with one format will simplify the use of multimedia for content developers and viewers alike. MPEG-4 must overcome many stumbling blocks before this will become a reality. These range from its own licensing system to other companies that choose not to support the MPEG-4 standard.

Windows Media 9 series is the latest release in the Windows Media family and was released in January 2003 [Win2003]. The technology mainly targets the DVD and video streaming market, aiming to enable Web-based delivery of high quality video to broadband users. The licensing programme for Windows Media Audio and Video 9 Series provides developers with longer terms and lower prices than the licensing programme of MPEG-4 and MPEG-2.

### **2.8.4 Macromedia Flash MX**

Macromedia Flash MX provides content authors with powerful video, multimedia and application development features that allow the creation of rich Internet applications, and enterprise application front-ends [All2002].

Macromedia Flash MX can import any standard video file supported by QuickTime or Windows Media Player. Content authors can then manipulate, scale, rotate, skew, mask and animate any video object, as well as make them interactive, by means of scripting.

## 2.9 Overview of MPEG Standards

MPEG has produced MPEG-1 and MPEG-2, both of which have won Emmy Awards and are currently working on MPEG-4 along with MPEG-7 and MPEG-21. A brief description of each standard now follows:

1. MPEG-1: Is a standard used for the storage and retrieval of moving pictures and audio. The powerful compression and decompression algorithms used by MPEG-1 allows a compression ratio of 10:1. MPEG-1 Layer 3 is the standard behind the highly popular MP3 file and revolutionised the way music is stored and transported over the Internet.
2. MPEG-2: Is the standard used in digital television and it is largely responsible for the advances in satellite television and DVD technology.
3. MPEG-4: Known as the standard for multimedia applications, it allows the combination of media objects such as audio, video etc. into a scene. Once in a scene each individual object's behaviour can be programmed allowing interaction with the user or with the scene itself. One of the most impressive aspects of MPEG-4 is its ability to be streamed over very low bandwidth connections.
4. MPEG-7: Is known as the Multimedia Content Description Interface and Version 1 was approved in September 2001. MPEG-7 provides a language allowing a set of description schemes and descriptors to be defined for the content of a multimedia product. These descriptors can then be searched which will allow the user to filter, manage and process the multimedia product. Finally a compression scheme for the descriptors will be provided, allowing the multimedia product to be packaged with its descriptors.

5. MPEG-21: This standard is an attempt to create a framework that will describe the relation between the elements that make up multimedia content. Currently, multimedia technology provides the different players in the multimedia value and delivery chain (from content creators to end-users) with an excess of information and services. The vision for MPEG-21 is to define a multimedia framework to enable transparent and augmented use of multimedia resources across a wide range of networks and devices used by different communities [BH2001].

A brief description of each MPEG standard was given in the numbered list above. The following sections will discuss the MPEG-7 and MPEG-21 standards in more detail.

### **2.9.1 The MPEG-7 Standard**

The concept of MPEG-7 was born in 1993 while MPEG was finalising MPEG-2 and defining the scope of MPEG-4. A proposal had been made to investigate the need to develop a standard that would allow users to identify the content that could be found in 500 TV channels. Work on the standard, named Multimedia Content Description Interface, officially started in 1997.

MPEG-7 has the objective of specifying a standard way of describing various types of multimedia data. This includes complete works or parts thereof and repositories. The description must be formulated independently of the representation format and storage medium. The goal is the efficient identification of desired information and the management of that information [Mar2002].

This process of describing and searching multimedia content is already in use by various digital asset management systems. These systems however, in general do not allow a search across different repositories and do not facilitate the exchange of data across different databases using different description schemes. These are interoperability issues, and creating a standard, is an appropriate way to solve the problem.

The guiding principles, which set the foundations of the MPEG-7 standard are as follows [Mar2002]:

- " MPEG-7 will be applicable to content associated with any application domain, be it real-time or not;
- " MPEG-7 descriptions may be used stand-alone, multiplexed with content or linked via hyperlinks to content;
- " MPEG-7 will consider most audiovisual data, no new descriptions for text will be used. The Standard Generalised Markup Language (SGML), Extensible Markup Language (XML) and Resource Description Framework (RDF) provide adequate tools;
- " MPEG-7 will remain independent of the media that carries the content;
- " MPEG-7 will be object based and allow the identification of primitive objects in the content;
- " MPEG-7 will be independent of the content representation format, whether digital or analogue, compressed or uncompressed;
- " MPEG-7 descriptions will allow several levels of abstraction from low-level, often statistical features to high-level descriptions conveying the semantic meaning of content; and
- " MPEG-7 will be extensible by allowing the standard to grow while keeping interoperability intact.

Several tools were developed to allow MPEG-7 to achieve its goals. These included descriptors (the elements), description schemes (the structure), the Description Definition Language (DDL) and a number of systems tools. A description of these tools follows.

The Descriptor is a representation of a feature, which is a distinctive characteristic of the data. A descriptor defines the syntax and semantics of the feature and allows the evaluation of the feature through the descriptor value. Description Schemes specify the structure and semantics of the relationships between components. The Description Scheme corresponds to an entity description diagram. The Description Definition Language allows the creation

of new description schemes and descriptors. The Systems tools manage the binary encoding process, synchronisation, transport and storage of descriptors, as well as the management of intellectual property rights.

MPEG-7, through the tools discussed above, provides the functionality for users to search for information by means other than typing a query. For example, speaking, drawing images and by humming a tune could be used as an alternative method of searching.

### **2.9.2 The MPEG-21 Standard**

Today there is a wealth of multimedia content available for content creators and end-users with many forms of delivery and consumption. Access to information and services from almost anywhere at anytime can be provided with ubiquitous terminals and networks. However, no complete solutions exist that allow different communities, each with their own models, rules, procedures, interests and content formats, to interact efficiently using this complex infrastructure [BH2001]. This causes an excess of information and services available to content users.

Developing a common multimedia framework will facilitate co-operation between the different communities. It will result in a more efficient implementation of the different models, rules, procedures, interests and content formats allowing improved integration of resources between communities.

To support multimedia content delivery and consumption, the content has to be identified, described, managed and protected. A multimedia framework is required to support this new type of multimedia usage. The MPEG-21 multimedia framework identifies and defines the key elements needed to support multimedia delivery and consumption as described above.

The seven key elements defined in MPEG-21 are:

- " Digital Item Declaration which is a uniform and flexible abstraction and interoperable schema for declaring Digital Items;



- " Digital Item Identification and Description provides a framework for identification and description of any entity regardless of its nature, type or granularity;
- " Content Handling and Usage provides interfaces and protocols that enable creation, manipulation, search, access, storage, delivery, and (re)use of content across the content distribution and consumption value chain;
- " Intellectual Property Management and Protection provides the means to enable content to be persistently and reliably managed and protected across a wide range of networks and devices;
- " Terminals and Networks enables the ability to provide interoperable and transparent access to content across networks and terminals;
- " Content Representation entails how the media resources are represented; and
- " Event Reporting provides the metrics and interfaces that enable users to understand precisely the performance of all reportable events within the framework.

The scope of the standard entails the integration of technologies that enable the use of multimedia across a wide range of networks and devices. The use of multimedia can be classified as: content creation, content distribution, content consumption, content packaging, intellectual property management and protection, content identification and description, financial management, user privacy, terminals and network resource abstraction, content representation and event reporting.

## **2.10 Summary**

The MPEG-4 standard, developed by MPEG, allows for the creation of multimedia applications, which may be streamed over low to high bandwidth connections irrespective of the underlying network transport protocol. For example, MPEG-4 Visual supports bitrates between 5 kbps to more than 1 Gbps. The standard incorporates the entire representation and delivery process of MPEG-4 encoded media.

MPEG-4 stores media elements as objects. An object is a single media element that may be natural, synthetic, two dimensional or three dimensional. Objects are completely independent of each other and are stored in a hierarchical tree with the the media objects forming the leaves.

Objects are combined into a scene and controlled by BIFS code. BIFS describes the spatial and temporal arrangements of the objects in a scene and builds on and extends several concepts from VRML. For the streaming process MPEG-4 specifies an interface to a suitable transport protocol stack such as ATM, IP and not the handling of the individual data packets. This makes it possible to encode a MPEG-4 media file once and play it almost anywhere using a MPEG-4 player.

In order to allow developers to adequately manage the large scope of MPEG-4, the functionality of the standard is devised into subsets, called Profiles. Through the use of Profiles, a developer can create a MPEG-4 compliant application by implementing a subset of the standard.

Research on MPEG-4 started in 1993 and still continues as several extensions of the standard, currently researched by MPEG, have not yet been completed. Research on MPEG-4 is not only done by MPEG but includes many industry and research institutions across the world.

This chapter discussed the research done on MPEG-4 by MPEG and other research groups and highlights topics that are of interest to the project. Not only does the large scope of MPEG-4 make it a powerful and more marketable standard, it also brings it into competition with many other technologies as discussed in this chapter.

The aim of this chapter was to serve as an introduction to MPEG-4, and to establish a basic understanding of the scope and features of the standard. In order to prepare for subsequent chapters, this chapter presented the fundamental ideas behind MPEG-4 and outlined the

functionality of MPEG-4. It has become clear that the MPEG-4 standard has a scope that covers a very large area. It is therefore important that the scope of the project is carefully defined as it must not only be manageable, but should also deliver a meaningful result.

The following chapter will discuss authoring tools in general and will provide an analysis of MPEG-4 authoring tools and other authoring tools that provide similar functionality to that of MPEG-4. At the time of writing there has been a very limited amount of research done on MPEG-4 authoring systems. Therefore acquiring a 2-D MPEG-4 authoring tool for evaluation purposes was not possible and thus an alternative solution had to be found. It was decided that an evaluation of authoring tools that provide similar functionality to that of a 2-D MPEG-4 authoring system would be done.

## Chapter 3: Authoring Tools

### 3.1 Introduction

MPEG-4, as discussed in the previous chapter, provides the functionality required to produce multimedia content that may be streamed over low bandwidth networks. The multimedia content or MPEG-4 scene, consists of audiovisual objects that interact with each other and possibly with the user as well. The behaviour and interaction process of the objects, together with their physical and timing attributes, are controlled by BIFS code.

To create a MPEG-4 scene, the author would either write the BIFS code and assemble the scene by hand, or use an application that automates the process. In order to create a scene by hand the author is required to write the BIFS code using a text editor. The code is then compiled and multiplexed with the media objects into a MP4 file. This process requires the author to have programming skills and an understanding of the BIFS language.

The other option is to use a program that automates or simplifies parts of or the entire process of creating the MPEG-4 scene. Programs that allow an author to manipulate different media elements spatially and temporally, and add interactive behaviour to them, are called authoring systems [CC2000] and assemble them into an integrated whole as a multimedia application. The spatial and temporal properties of objects refer to their on-screen position and time-based properties respectively.

In general, authoring can be seen as a faster form of programming. Rather than typing out all the code by hand, the authoring tool allows the author to generate code through the authoring process. Providing the content author with a multimedia authoring tool would reduce the prerequisite knowledge required of the author and increase authoring productivity. This is accomplished by the high-level programming support built into the interface of the authoring tool [RME1996].

By using an authoring tool, the project time to market will be reduced as well as the cost of programmer time spent on writing code. It is however important to keep in mind that content creation (images, video, audio, etc.) is not affected by the use of an authoring tool and there will be no production gains other than the fast generation of BIFS code.

The aim of this chapter is to describe information that will allow the specification of a tool for the creation of 2-D MPEG-4 scenes. The chapter will discuss various authoring paradigms and authoring tools that may provide information for designing a MPEG-4 authoring tool. The discussion of authoring tools will take on the form of an extant systems analysis. An extant systems analysis is the study of an existing system that may provide information for the design of the new system. At the time the analysis was conducted, in the second semester of 2001, there were no 2-D MPEG-4 authoring tools available for analysis. It was decided that authoring tools with similar functionality to that of a 2-D MPEG-4 authoring tool will be analysed. A detailed discussion of research on MPEG-4 authoring tools, completed during the implementation of this project, will be provided in Chapter 7. A suitable paradigm for a MPEG-4 authoring tool will be selected and the findings of the extant systems analysis will be discussed.

## **3.2 Authoring Paradigms**

Studies of authoring systems have revealed paradigms into which authoring systems can be classified. An authoring paradigm, also known as an authoring metaphor, is the metaphor that an authoring tool uses to accomplish its task. It is the way a particular authoring tool structures multimedia data and the interactions with the author. For example, the authoring tool can use a cast, score and scripting paradigm as explained in Section 3.2.5. In this example, the authoring tool uses a movie as a metaphor. The media objects are called actors and are stored in the cast. The display area is known as a stage and a score, similar to that used in music productions, is used to model the temporal behaviour of the objects.

The main paradigms used by authoring tools are [Sig1999]:

- " Scripting Language;
- " Iconic/Flow Control;
- " Frame;
- " Card/Scripting;
- " Cast/Score/Scripting;
- " Hierarchical Object; and
- " Hypermedia Linkage Paradigm.

Each of the above mentioned paradigms will be briefly discussed in the following sections.

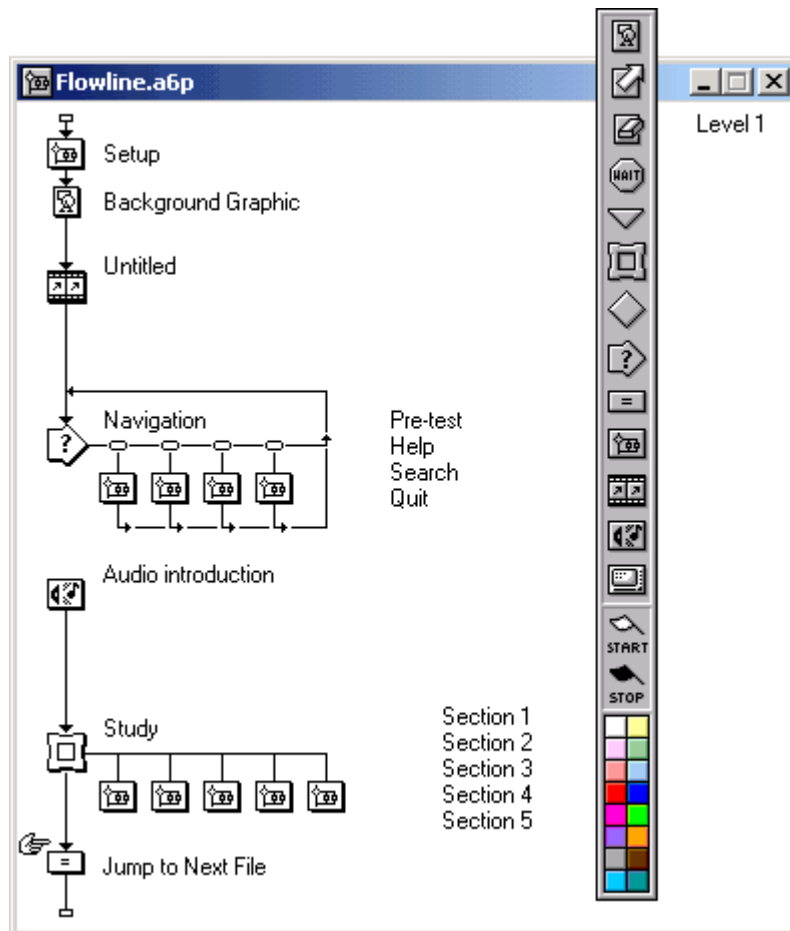
### **3.2.1 The Scripting Language Paradigm**

The scripting language paradigm is the closest to traditional programming, with a powerful object orientated programming language forming the core of the authoring tool. The main difference between a traditional programming language and a scripting language is that the traditional programming language is compiled, while a scripting language like JavaScript is interpreted. The scripting paradigm has little or no high-level programming support built into the interface. Therefore the scripting paradigm tends to have a longer development time, but it generally allows for more control over the media elements in the production and therefore supports more powerful interactions with the user.

### **3.2.2 The Iconic/Flow Control Paradigm**

The iconic/flow paradigm is widely used for creating computer-based training (CBT) solutions. It is best suited for rapid prototyping and short development time projects, as it is one of the fastest development paradigms. An example of a system using this paradigm is Macromedia Authorware [Mac2003]. Figure 3.1 shows the development screen of Authorware. The system uses icons to represent the individual components of the system

and the structure of the system is represented by a flow line. Authorware is not based on a scripting language and it is suitable for non-programmers. However, the structure used by the flow line is similar to that of a traditional programming language.



**Figure 3.1: Macromedia Authorware: an example of an iconic/flow paradigm**

### 3.2.3 The Frame Paradigm

The frame paradigm is similar to the iconic/flow control paradigm as it usually incorporates an icon palette and enables rapid application design. The difference between the two paradigms is the links between the icons which are conceptual and do not necessarily represent the flow of the program. Another difference is the possible inclusion of a scripting language as found in Quest [All2003], shown in Figure 3.2, which uses the C programming language for scripting purposes.

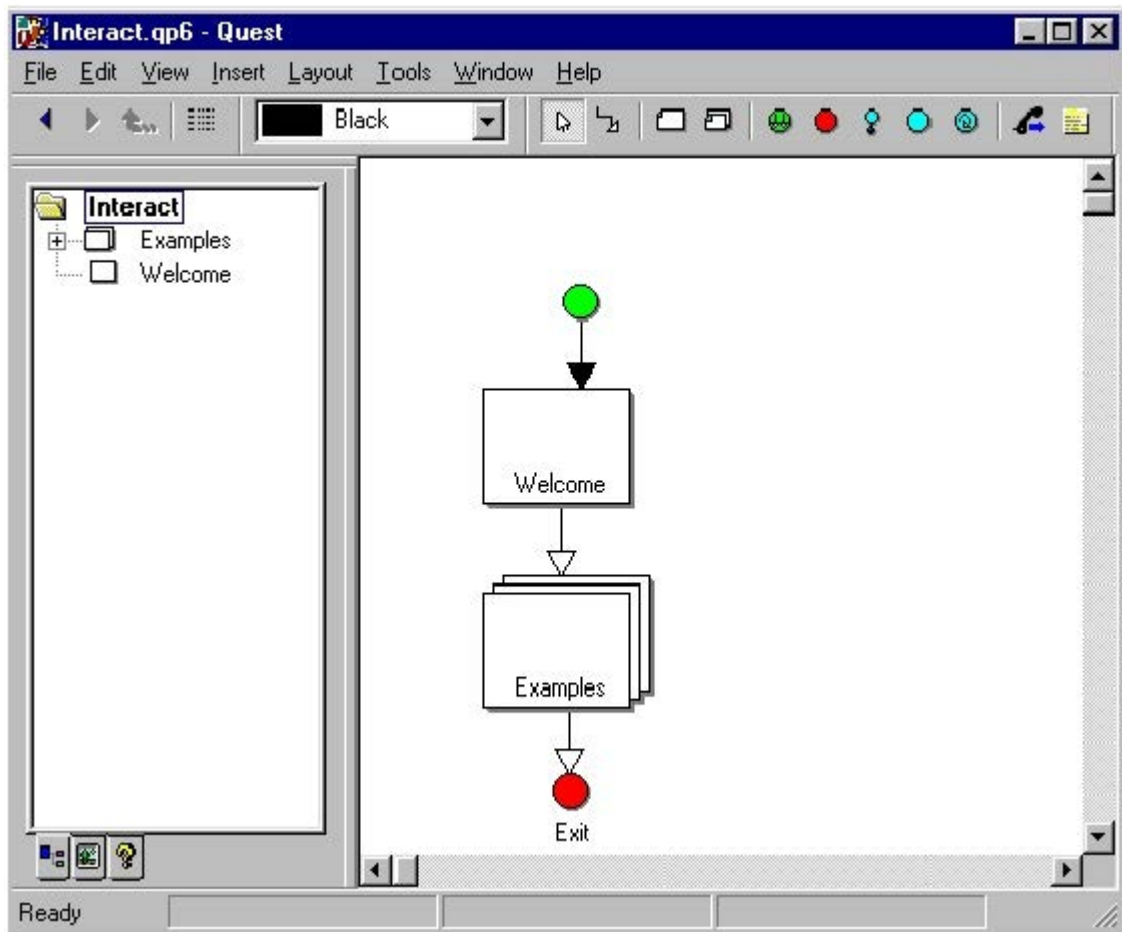


Figure 3.2: Quest: an example of a frame paradigm

### 3.2.4 The Card/Scripting Paradigm

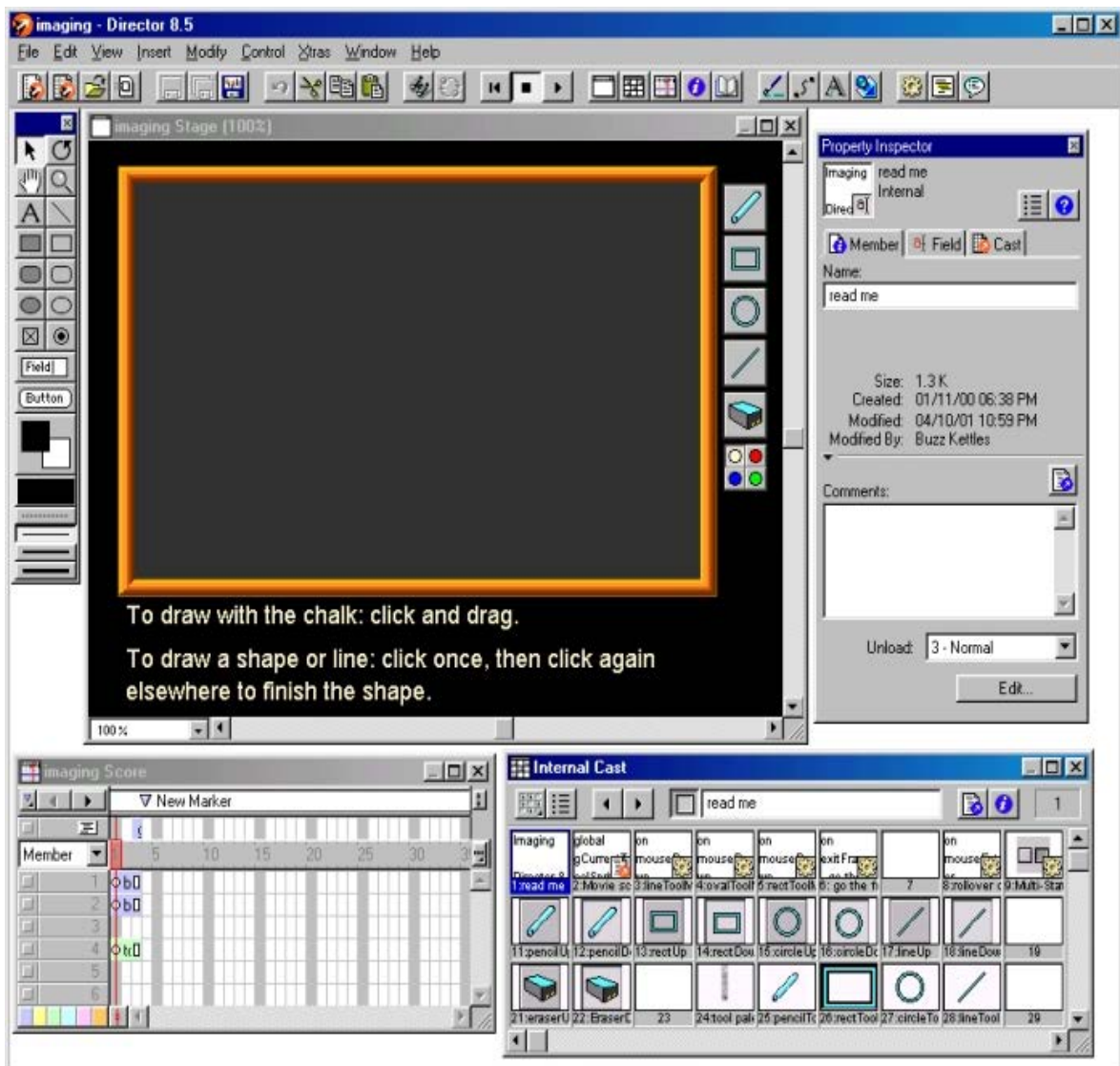
The card/scripting paradigm is very powerful due to its scripting capabilities and is ideally suited for hypertext and navigation intensive applications. Many entertainment applications are prototyped by using a card/scripting system before it is implemented in another programming language.

### 3.2.5 The Cast/Score/Scripting Paradigm

The cast/score/scripting paradigm uses a music score as its primary authoring metaphor. Figure 3.3 shows the development interface of Macromedia Director [Mac2002] which is



one of the most popular authoring tools using this paradigm. The media elements are imported into a cast and can then be placed on the stage. The strength of this metaphor lies in its ability to add scripts to the behaviour of all the objects in the cast. All elements or cast members on the stage are shown in the score in horizontal lines depicting their temporal attributes. When objects lie beneath each other in the score, it indicates that they appear simultaneously on the screen at that point in time. Authoring systems using this paradigm are ideally suited for animation-intensive or synchronised media applications.



**Figure 3.3: Macromedia Director: an example of a cast/score/scripting paradigm**

### 3.2.6 The Hierarchical Object Paradigm

The hierarchical object paradigm uses an object orientated metaphor which is visually represented by embedded objects and icons. Using a tool such as this one, allows for the re-use of code through multiple inheritance and therefore saving development time. Unfortunately the flow of the program is not visually represented in this paradigm. Therefore, the paradigm does not indicate the flow of the program from one screen to another or the temporal behaviour of the objects used on screen. This, especially when the program is large and complicated, may cause confusion for the author. An example of an authoring tool using a hierarchical object paradigm is mTropolis [mTr2002] as shown in Figure 3.4.

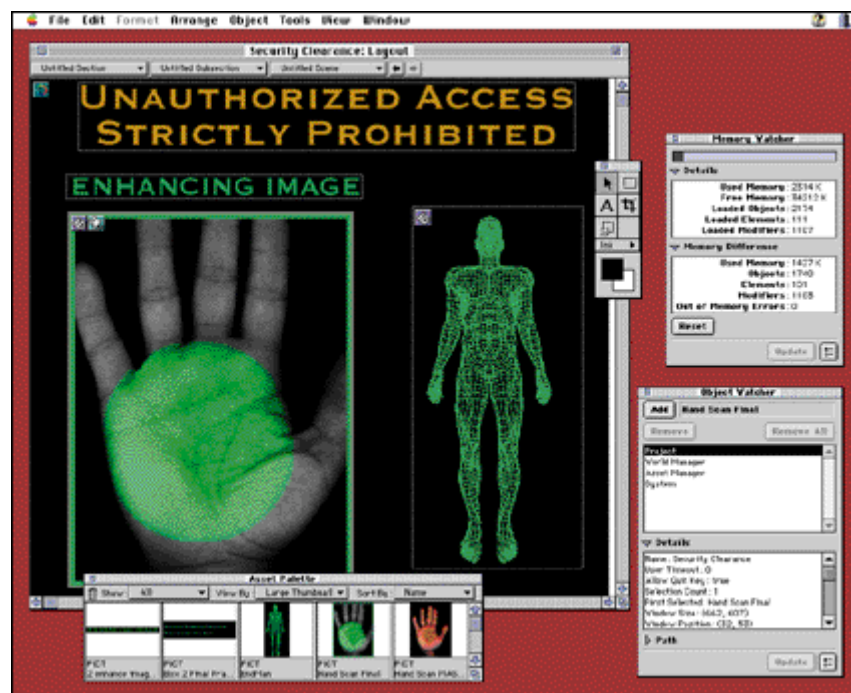


Figure 3.4: mTropolis: an example of a hierarchical object paradigm

### 3.2.7 The Hypermedia Linkage Paradigm

Similar to the frame paradigm, the hypermedia linkage paradigm uses a visual element and menu-driven linker interface. This allows all elements to be accessed and arranged in a

frame view, showing the conceptual links between the elements. This paradigm is well suited for CBT content development and prototyping of applications that act as a guide or that tells a story.

The paradigms were discussed in order to provide an overview of how authoring tools can be classified. In the next section existing authoring systems, with similar functionality to that of a 2-D MPEG-4, will be discussed.

### **3.3 Extant Systems Analysis**

Before creating a specification document for a MPEG-4 authoring tool, it is important to take cognisance of work already done in the area. At the time the extant systems analysis was conducted in 2001, there were no 2-D MPEG-4 authoring systems available for analysis. Therefore this section will contain a analysis of authoring tools that have similar goals to that of this research project.

To ensure that the final system is usable, a user-centred approach will be adopted by applying a formal methodology. By using a user-centred approach the systems requirements will be generated to cater for the needs of the author. The methodology selected for this purpose, explained in the next section is Objects, Views, and Interaction Design (OVID). OVID is a formal methodology for designing user friendly programs based on the analysis of a user's goals and tasks [BIM1997].

#### **3.3.1 Object, Views and Interaction Design**

There are several methodologies available for the design of an object-orientated program. To select an appropriate methodology the following requirements of the system with regard to a methodology was considered:

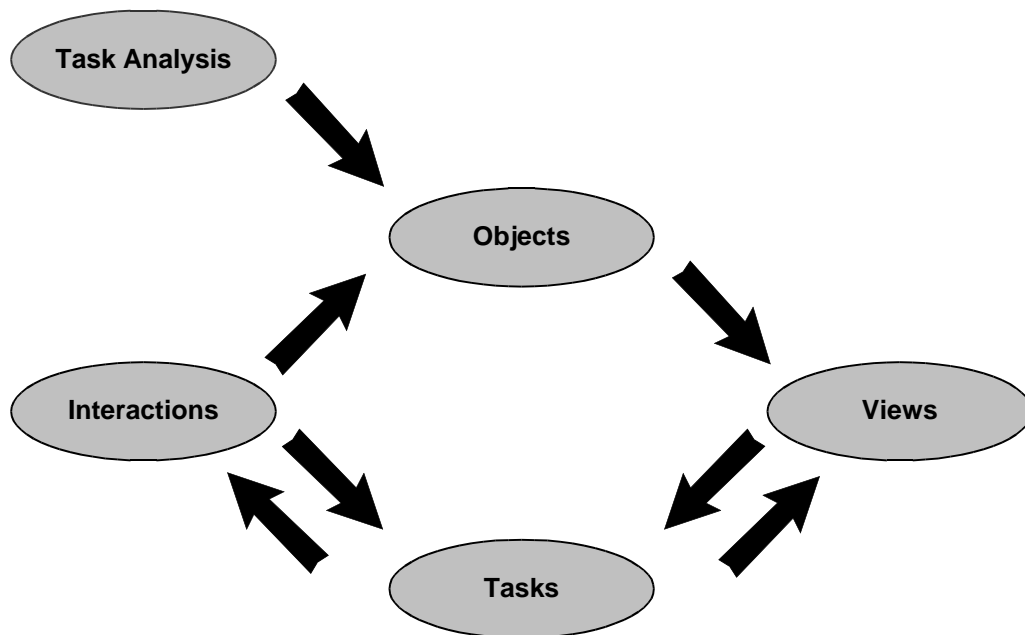
- " A user-centred approach must be used;
- " Take advantage of the user's mental model;

- " Provide support for iterative prototyping;
- " Cater for the design of an object-orientated system; and
- " Produce an output that specifies the design of the user interface and underlying data model.

The requirements listed above were used to select an appropriate methodology. The OVID methodology, developed by IBM, was selected as it satisfies all the requirements of the project. OVID produces a complete, correct model that can be used as basis for implementation. OVID is well suited for the design and implementation of object-orientated programs and allows an iterative prototyping approach. Object-oriented interfaces allow users to interact with the system by using objects that correspond to the user mental model.

From the name, it is clear that the methodology focuses on the objects that the author can interact with, the views provided by the objects and the interactions that the author has with the objects. OVID follows an iterative cycle and starts with a task analysis as described in Figure 3.5. The task analysis is an abstract description of the authoring process used by the author to create an MPEG-4 application. The task analysis is used to create an initial set of objects followed by defining views for the objects. The tasks of the author are then described in terms of the objects, views and the interactions of the author with the objects.

The first step in the OVID methodology is to complete a task analysis which will be described in the following section. Thereafter an evaluation of current authoring tools with similar goals to that of the project will follow. The reason for doing an extant systems analysis before proceeding to the object step in the OVID methodology, is that a great deal can be learned from evaluating current tools. The evaluation will provide guidelines for what the objects, views and interactions could be.



**Figure 3.5: The OVID methodology**

### 3.3.2 Task Analysis

The specification of the task analysis posed an interesting problem. In Chapter 1, it was indicated that the author using the proposed system is expected to be a computer literate user, having a working knowledge of the multimedia content of MPEG-4. As stated before, there were no 2-D MPEG-4 authoring tools available for evaluation. Therefore this project, in a certain sense, aimed to create something that did not exist. This meant that there was no user of MPEG-4 authoring systems available for the project. With no user and no extant system to provide a task model it was necessary to construct the task analysis by other methods.

In order to generate a task analysis, the functionality provided by MPEG-4 together with the scope of the project was utilised. Since the goal of the system is to produce BIFS code, the functionality provided by MPEG-4 could be determined by studying the BIFS language. Therefore it was possible to determine what the author could and could not do. Once a task

analysis was constructed, it was used to evaluate existing authoring tools with similar goals to that of the project. An extant systems analysis then provided information on how the systems allow the author to complete his or her task.

In the task analysis, the author must firstly create an empty scene and then populate it with video, image, audio, text and geometry objects. After the objects have been created the user must define the behaviour of an object, by using events, on one or more of the objects. The constructed task model is a normative task model and it contains the steps that are thought to be required to complete a typical given task. The normative task model below makes use of the functionality provided by BIFS and will list the general steps required to complete the task as stated above.

Three authoring systems have been selected for evaluation to assist with the task analysis. The first is MPEG-4 ToolBox [DKR+2000] and it is the only MPEG-4 authoring tool to be evaluated. The remaining two systems are Internet Scene Assembler [Par2002], a VRML authoring tool, and Macromedia Flash [All2002], an authoring tool widely used for creating interactive multimedia presentations for the Internet.

Once the analysis has been completed, the actual steps required to accomplish the given tasks would have been observed through the use of the three systems. By creating a list of the actual steps required to complete a task, a descriptive task model can be constructed. A descriptive task model lists the actual steps required to accomplish a given task. A normative task model for a 2-D MPEG-4 authoring tool is given below:

- 1 Select New Scene from the file menu
  - 1.1 Add a background object
  - 1.2 Set the background colour
  - 1.3 Set the Scene size in pixels
  
- 2 Select a Video Object and insert it onto the Scene
  - 2.1 Select the video file associated with the object
  - 2.2 Set the starting time of the video

- 2.3 Set the playback size
- 2.4 Set the loop option to true or false
- 2.5 Set the transparency level
- 2.6 Set the position of the video in pixels
  
- 3 Select an Image Object and insert it onto the Scene
  - 3.1 Select the image file associated with the object
  - 3.2 Set the size of the image
  - 3.3 Set the transparency level
  - 3.4 Set the position of the image in pixels
  
- 4 Select a Audio Object and insert it onto the Scene
  - 4.1 Select a audio file associated with the object
  - 4.2 Set the playback rate of the object
  - 4.3 Set the starting time of the audio
  
- 5 Select a Text Object and insert it onto the Scene
  - 5.1 Set the text in the object
  - 5.2 Select the font type and size
  - 5.3 Set the colour of the text
  - 5.4 Set the transparency level
  - 5.5 Set the position of the text in pixels
  
- 6 Select the Geometry Shape object and insert it onto the Scene
  - 6.1 Select the type of object
  - 6.2 Set the colour of the object
  - 6.3 Set filled true or false
  - 6.4 Set the transparency level
  - 6.5 Set the position of the shape in pixels
  
- 7 Add an event to a object
  - 7.1 Select the desired object

- 7.2 Select add event
- 7.3 Select the type of action that will trigger the event
- 7.4 Select the object(s) which will be affected
- 7.5 Select the property to be modified
- 7.6 Set the new value of the property

The normative task model, as constructed above, will now be compared to the actual steps needed to complete the same tasks by using each of the three authoring systems. The actual steps required to complete the tasks will be divided into three sections. The first section will discuss the process of creating a new scene, the second will discuss adding objects to a scene and finally, the steps required to add events to an object will be discussed.

### **3.3.3 MPEG-4 ToolBox**

As stated in Chapter 1, there were very few implementations of the MPEG-4 standard and in particular MPEG-4 Authoring tools available at the time project was started in 2001. What further complicates matters is that developers and researchers are not willing to share their work. This is understandable as they wish to protect their investment and competitive advantage in the market. MPEG-4 ToolBox however is an exception to the rule and was the only MPEG-4 Authoring tool available for evaluation. This Authoring tool was the first software application that was evaluated. Although MPEG-4 ToolBox is a 3-D authoring tool, it can still provide useful information for the design of a 2-D Authoring tool. The process of adding objects to a scene and modifying their attributes will be similar in a 3-D and 2-D tool.

MPEG-4 Toolbox was produced through a collaborative effort by the Informatics and Telematics Institute and the Information Processing Laboratory at the University of Thessaloniki, both in Greece [DKR+2000]. It is a MPEG-4 Authoring tool that allows the user to create 3D MPEG-4 scenes. The system consists of a graphical user interface that allows the user to place and modify objects in a scene. The system supports the creation of spheres, cones, rectangles, cylinders, text, audio, geometrical shapes and a human face or avatar. Once the objects have been placed in the scene the user can modify them by



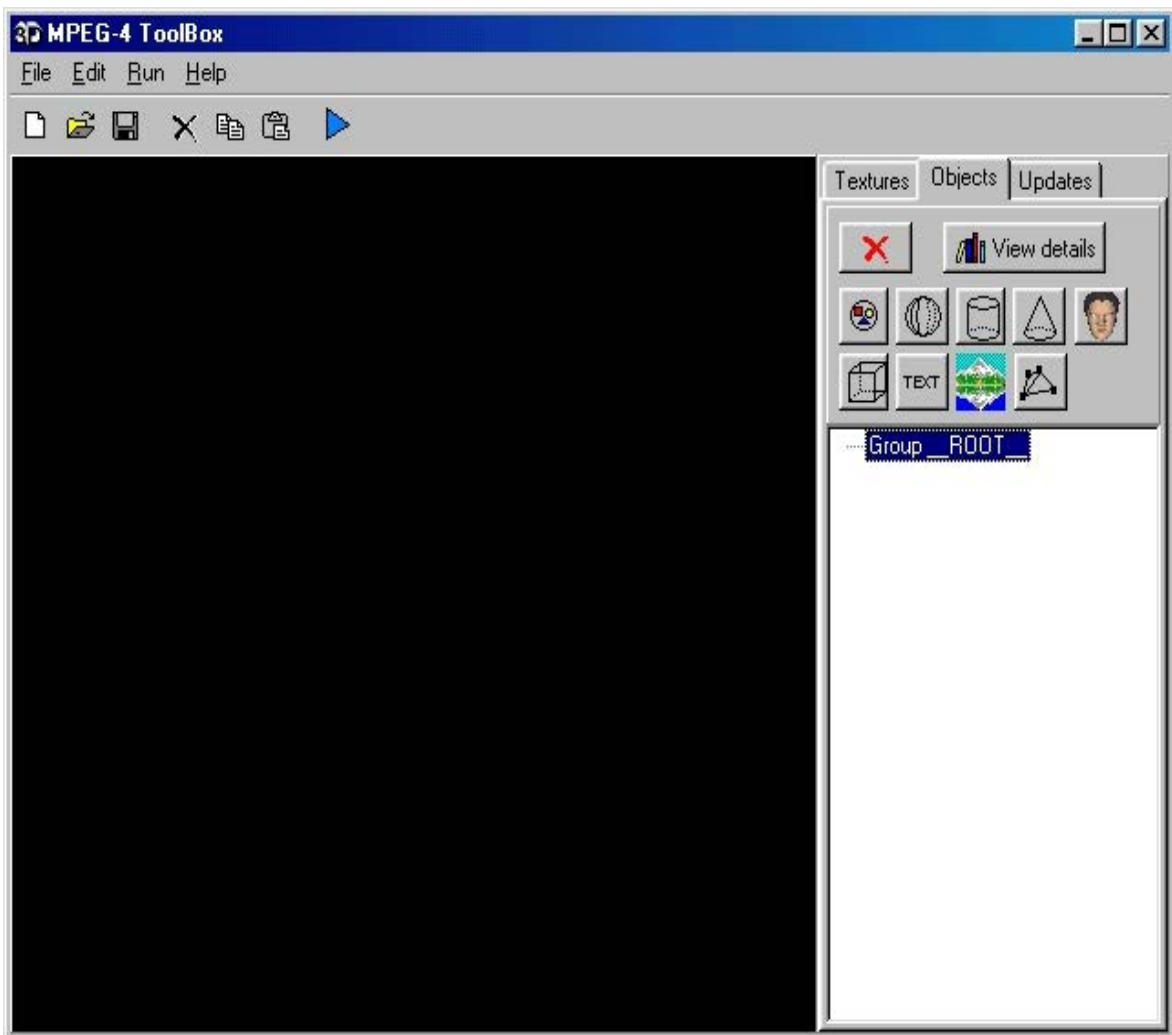
changing their physical and spatial attributes by altering the position, colour and textures of the objects. The user also has the option to add various methods of interaction to the objects.

After the author has created a scene, it may be encoded to a MP4 file and viewed by using a MPEG-4 compliant 3D player. The development of the encoding and playback process however, was not part of the scope of the project. MPEG-4 ToolBox makes use of software developed by the IM1 MPEG-4 development team. It makes use of the BIFS encoder and MP4 file multiplexer to create the file and the IM1 3-D player to play the file. In the next section, MPEG-4 ToolBox will be evaluated to determine how it supports the tasks of the user.

### **3.3.3.1 Creating a New Scene**

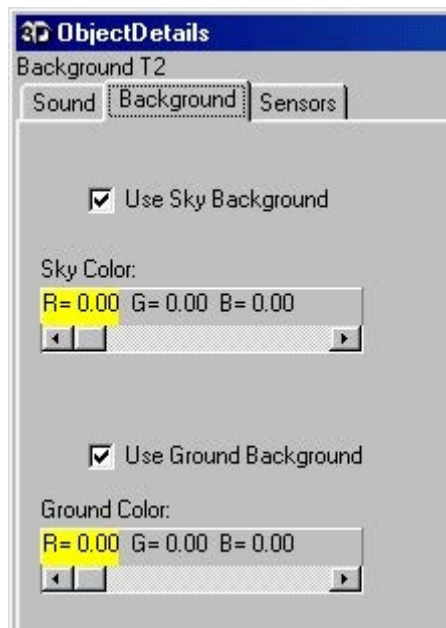
The initial startup screen for MPEG-4 ToolBox is shown in Figure 3.6. It has a standard Windows look and feel with a large editing area in the centre of the screen. The editing area provides the author with a preview of the scene. The objects that can be placed in the scene are on a tool palette on the top right of the screen. An object tree similar to the those found in VRML authoring tools is below the tool palette. The object tree provides a visualisation of the hierarchical tree in which all MPEG-4 objects are placed and it is a very useful tool for visualising the layout of the scene. Since the authoring tool does not support all the tasks as listed above by the task analysis, only a subset will be performed and evaluated.

In order to create a new scene the author must click on the standard blank document icon or click on "File" and then "New". When changing the background the user must first add a background to the scene and only then can its colour be changed. The author must click on the background icon to add a background object on the screen and in the object tree. When changing the colour of the background the author must click on the object in the tree and then on the "View Details" button.



**Figure 3.6: MPEG-4 Toolbox**

The only way the author can select an object for modification is to select it in the object tree, the author cannot click on the object in the scene. As shown in Figure 3.7 the author now has the opportunity to change the background colour for the top and bottom half of the screen. The interaction style used here is very cumbersome. The author only has the opportunity to change the RGB values one at a time by clicking on the value, highlighting it with a bright yellow colour and then dragging the scroll bar. The size of the scene cannot be set and it depends on the viewing area of the viewer's MPEG-4 player.



**Figure 3.7: Changing the background colour of the scene**

### 3.3.3.2 Adding Objects to the Scene

In order to add an object to the scene, the author must click on the corresponding object on the tool palette. The object is then added to the middle of the viewing area that represents the scene. To modify the object the author must select the object in the tree and then click on "View Details". As shown in Figure 3.8 the author has the opportunity to modify the properties of the object. The different properties are grouped together into tab panes. On the "General" tab, the author has the opportunity to change the spatial attributes of the object.

The "Material" tab allows the author to access the colour properties of the object. The "Video" tab as shown in Figure 3.9 gives the author an option to browse for a video object by clicking on the "Load Video" button. After loading the video, the system displays the path to the video. Here the author may specify the start and stop times and whether the movie should loop or not. The "Sound" tab has very much the same design as the "Video" tab. The "Sound" tab allows the author to browse for a sound file and set the start and stop times for the audio file.

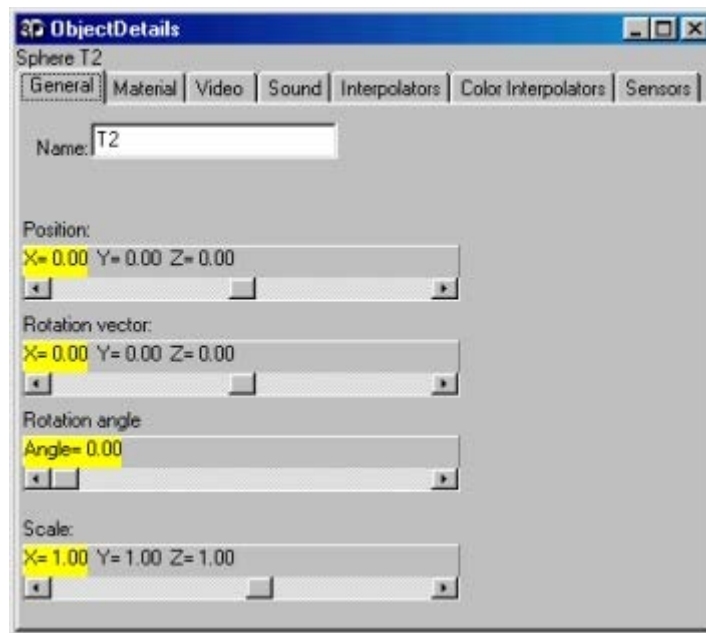


Figure 3.8: Changing the properties of an object

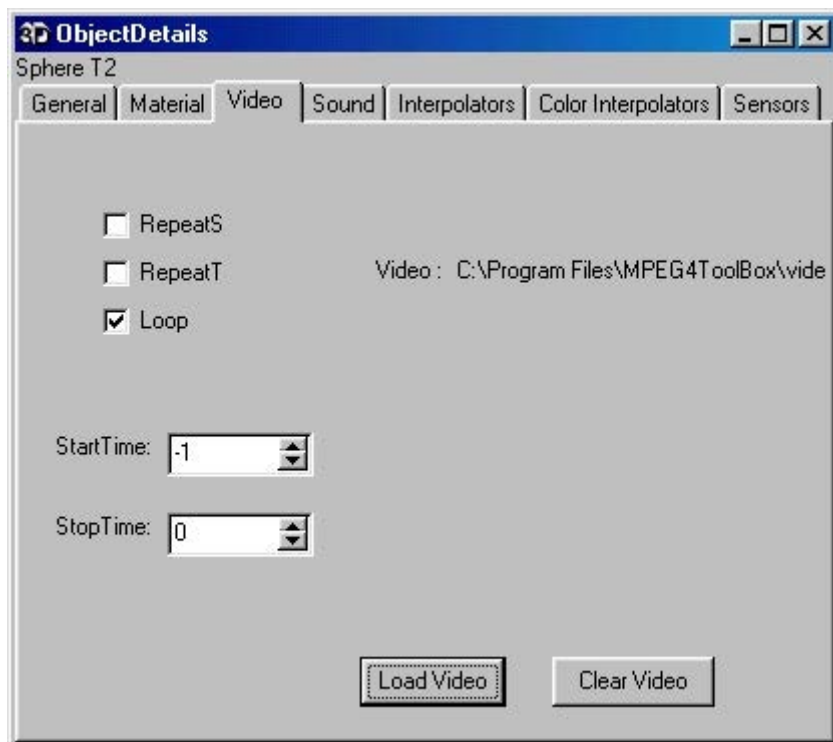
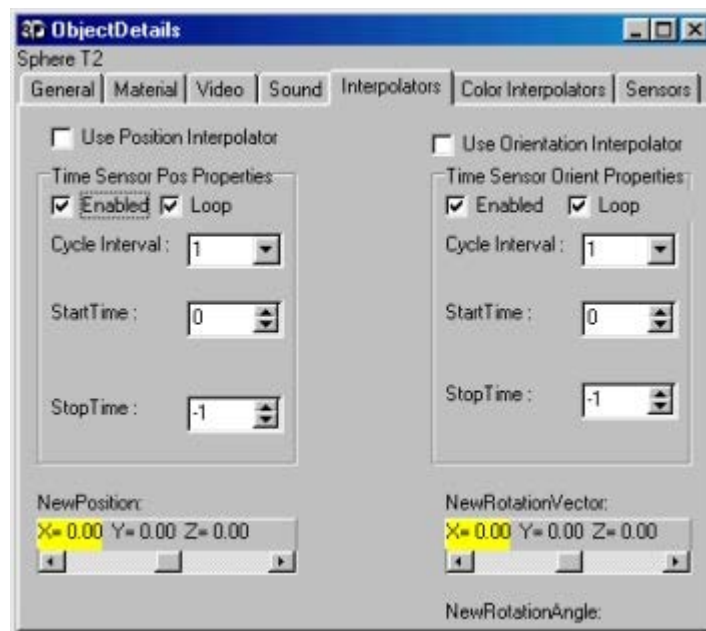


Figure 3.9: Adding a video to the object

### 3.3.3.3 Adding Events to Objects

The interaction options are on the "Interpolators", "Colour Interpolators" and "Sensors" tabs. The terms interpolators and sensors come directly from VRML and BIFS. An interpolator is a function that accepts a value and interpolates it to give a suitable output (for example, accepting the elapsed time as a input and giving the position for an object at the given time). A sensor is used to sense the occurrence of a specific event. For example, a TouchSensor allows an object to sense when a mouse pointer is over it or when a viewer clicks on it.

The "Interpolators" tab is shown in Figure 3.10. The author has the opportunity to modify the position or orientation of the object. The author can choose between changing the position and orientation of the object by clicking the corresponding check box. Once the selection has been made the author may associate a timer to the event by changing the time sensor properties. Finally the author can select the new position or orientation of the object by using the scroll bars at the bottom of Figure 3.10. The "Colour Interpolator" tab has the same design allowing the user to select the change in colour and if required the author may select the use of a timer.



**Figure 3.10: Selecting an interpolator**

After setting the interpolators, the author must select an event that will trigger them. The user can select the sensors that trigger events on the "Sensors" panel in Figure 3.11. Here the author can select the type of sensor and which interpolator to activate when the event takes place. In Figure 3.11 below the author selected the "TouchSensor" event that will activate when the author clicks on the object and it will activate the "Position Interpolator". MPEG-4 ToolBox does not allow an event on one object to affect any other object(s) other than the object with the event associated to it.

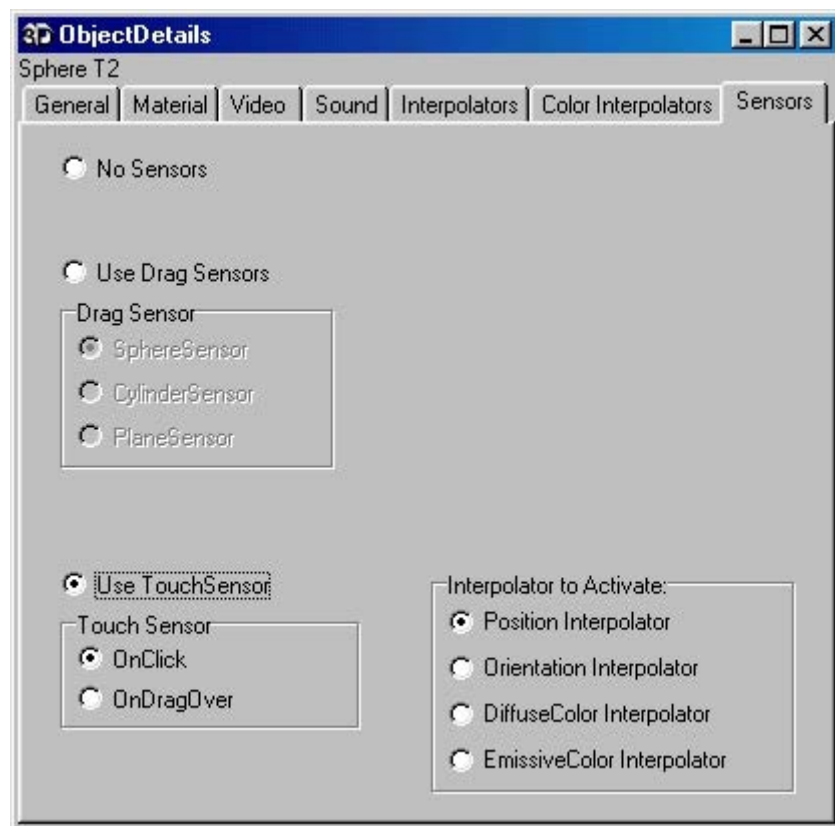


Figure 3.11: Selecting a sensor

### 3.3.4 Internet Scene Assembler (ISA)

In general, multimedia productions consist of audiovisual objects which are controlled by some form of underlying code. MPEG-4 is no different and uses BIFS code to govern the

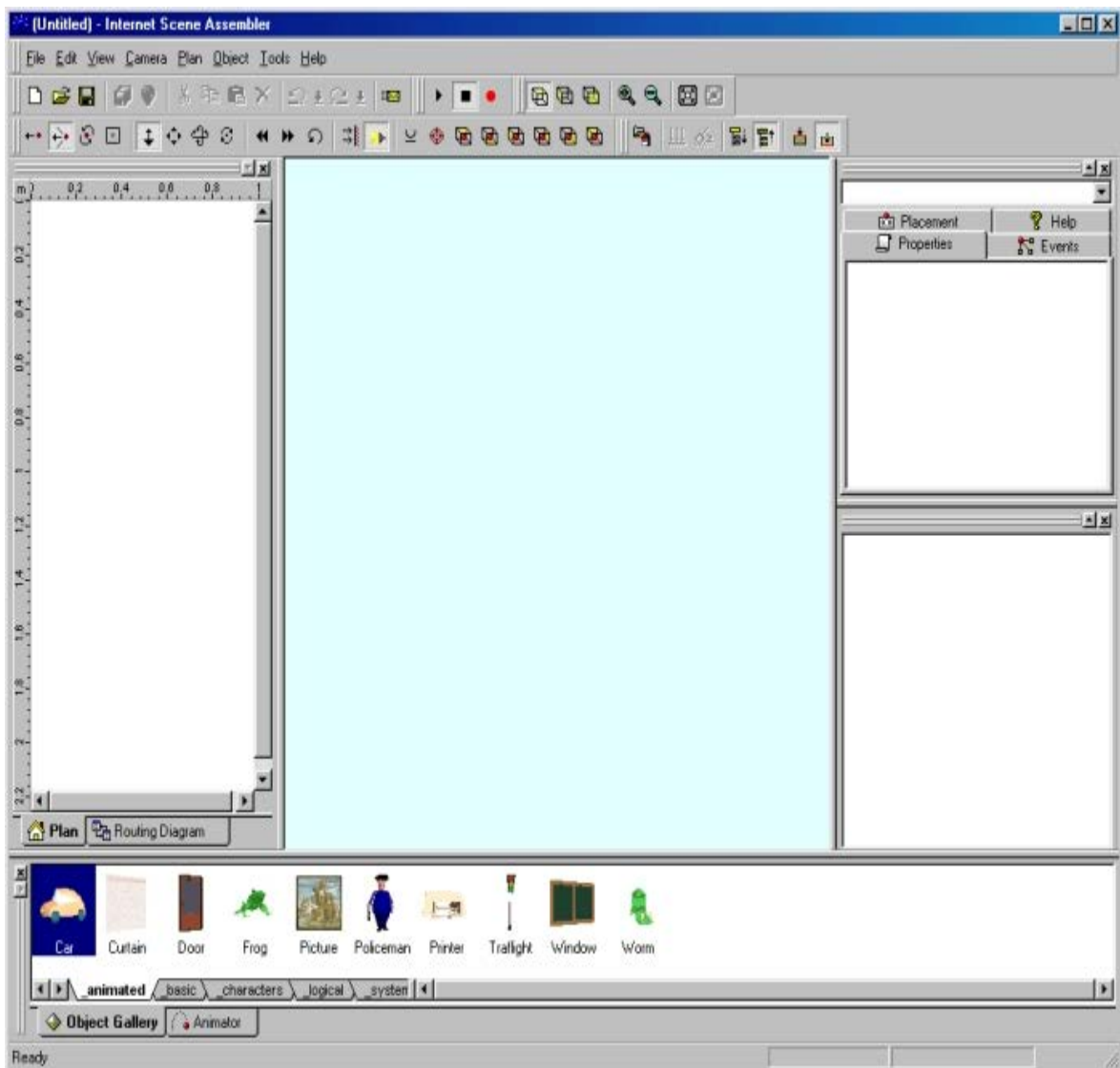
behaviour of its objects. BIFS as previously explained, is based on VRML and extends it to support the MPEG-4 standard. As BIFS is the underlying code that controls the objects in a scene it will, to a large extent, determine the interface of a MPEG-4 authoring tool.

Since VRML and BIFS are so similar it would follow that the authoring tools that generate the code should have some characteristics in common. Therefore the second tool that was evaluated is a VRML authoring system named Internet Scene Assembler (ISA). Once again the system that was evaluated is a 3D editing tool and as with MPEG-4 ToolBox, only relevant areas of the system will be evaluated.

#### **3.3.4.1 Creating a New Scene**

The startup screen for ISA is shown in Figure 3.12. The large screen in the center is the editing or the "Perspective View" window in which the 3-D scenes are viewed and navigated. The "Properties/Events" window is located at the top right and it allows the author to view and change the properties and events of the selected object. Below this window is the "Scene Tree" window which is used for viewing and selecting objects within the hierarchical structure of the current scene. The "Object Gallery" at the bottom of the screen contains objects that can be added to the scene. Objects are grouped by category and are displayed by clicking a category tab at the bottom of the gallery window. On the left hand side the author can use the "Routing Diagram" to route events between objects.

In order to create a new scene the author must click on the standard blank document icon or click on "File" and then "New". To change the background the author must first add a background to the scene. To do this the author must click on the system tab in the "Object Gallery" and drag the the background icon onto the editing area. Once the author has created a background object it may be selected in the "Scene Tree" and customised as required. As in the previous tool the scene size cannot be set and it depends on the browser application.



**Figure 3.12: A new scene in ISA**

### 3.3.4.2 Adding Objects to the Scene

In order to add an object to a scene, the author can select the relevant category from the "Object Gallery" and then drag the desired object onto the scene. To add an object to the "Object Gallery", the user must click on "Object" and then "Add Object" from the menu bar at the top of the screen.

Once an object is in the scene, the author can select it by clicking on it in the "Scene Tree" window as shown in Figure 3.13 below. To move an object the author can click on and



select the object and then drag it to the desired position. Once an object has been selected the author can change its properties in the "Properties/Events" window which is also shown in the Figure 3.13 below.

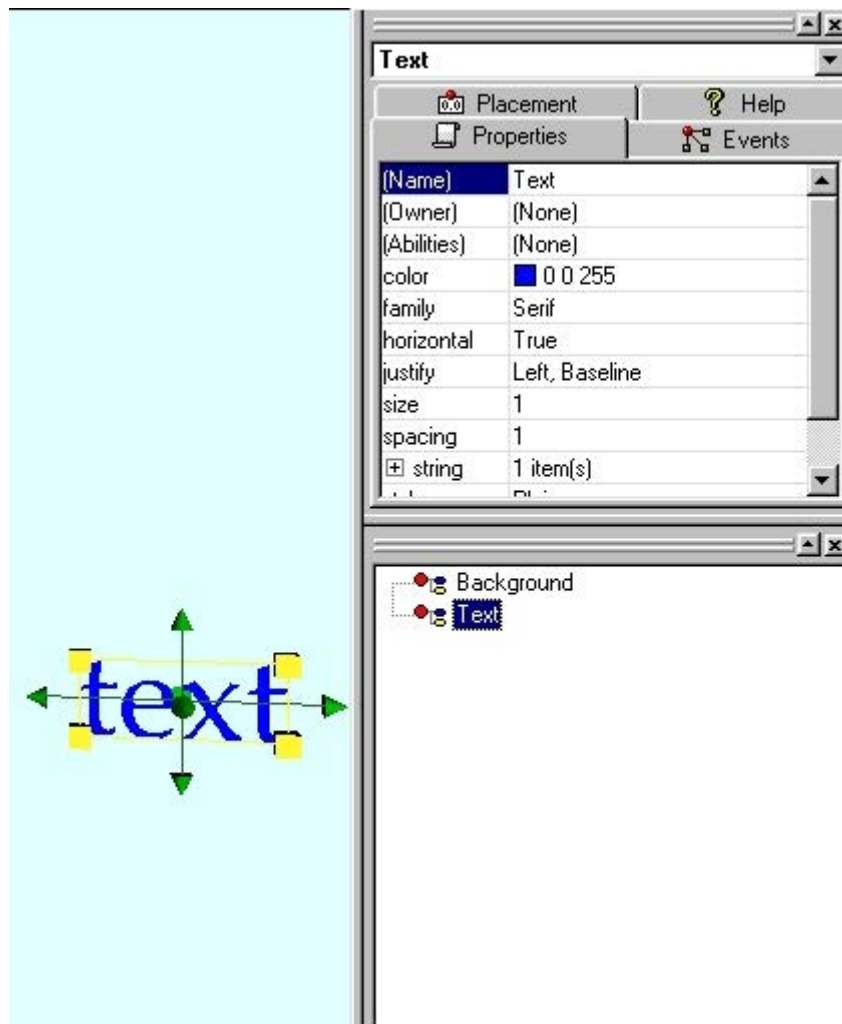
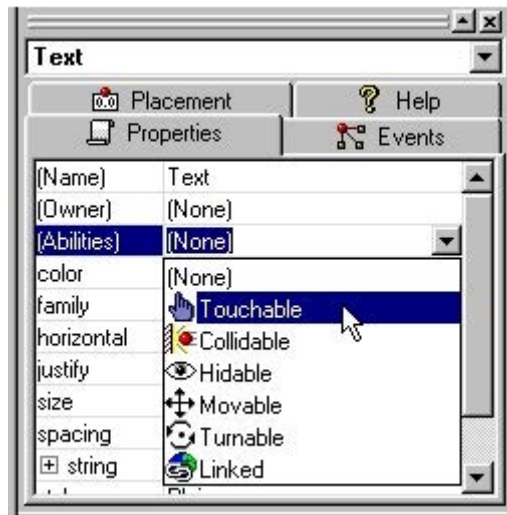


Figure 3.13: Editing objects in a scene

### 3.3.4.3 Adding Events to Objects

In order to add an event to a selected object, the author must change its "Abilities" option in the "Properties/Events" window under the "Properties" tab as shown in Figure 3.14. Once the object has an event associated with it, the author may route the events by clicking

on the "Events" tab, where the author will find a list of all events associated with the object. Each event has a black arrow indicating whether the event can send (out-event) or receive (in-event) a message.



**Figure 3.14: Adding an event to an object**

In order to route an event from one object to another the author must select an out arrow from an event on the object that will be the source of the event. Once the out-event has been selected all event arrows that have an out arrow or events that cannot receive an event message from the source object become invisible. This leaves only in-events that can receive a message. The author must then select the desired object and then select the in-event that will receive the message.

Figure 3.15(a) and (b) depicts the sequence of steps required to make a light switch on when an object is clicked. The objects that will be used are a "Triggered Button" and a "PointLight". In this example the author has to select the object, a "Triggered Button", click on "Properties" and make it "Touchable" and then click on the "Events" tab. Here the author selects an event, in this case the "toggle" event which is an out-event. Once the author clicks on the out-event all out-events disappear leaving only events that will accept the message.

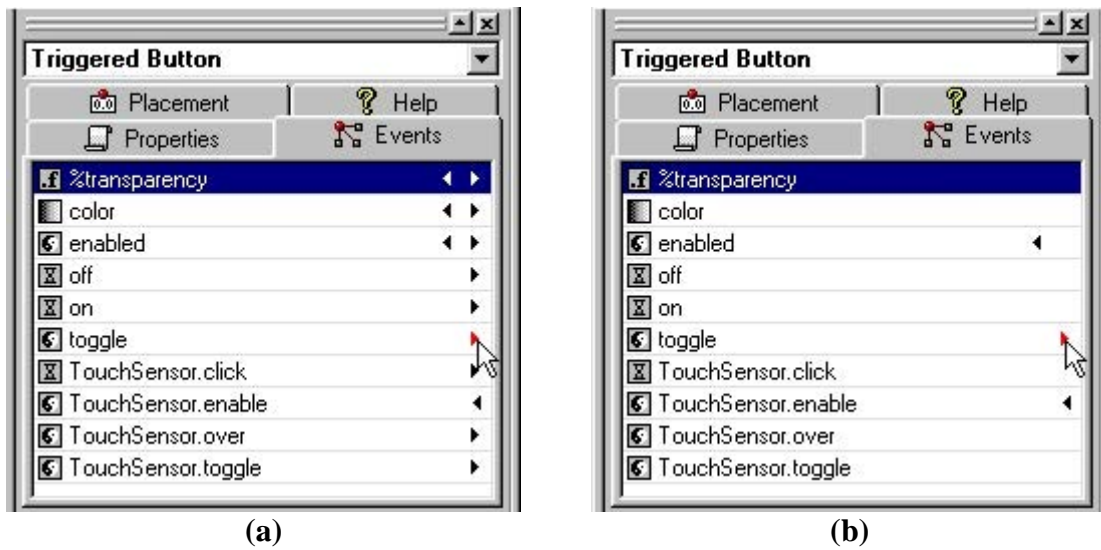


Figure 3.15: Selecting an event

In the next step described in Figure 3.16(a) the author must select the object that will receive the event which in this case is a "PointLight". After selecting the object the author can select the in-event and the routing of the event will be complete. Figure 3.16(b) shows the result of the route with an arrow between the trigger button event and the "on" for the "PointLight" object. The "Events" tab of the trigger button will be updated in the same way but it will have a arrow from the "toggle" event to the "PointLight" object.

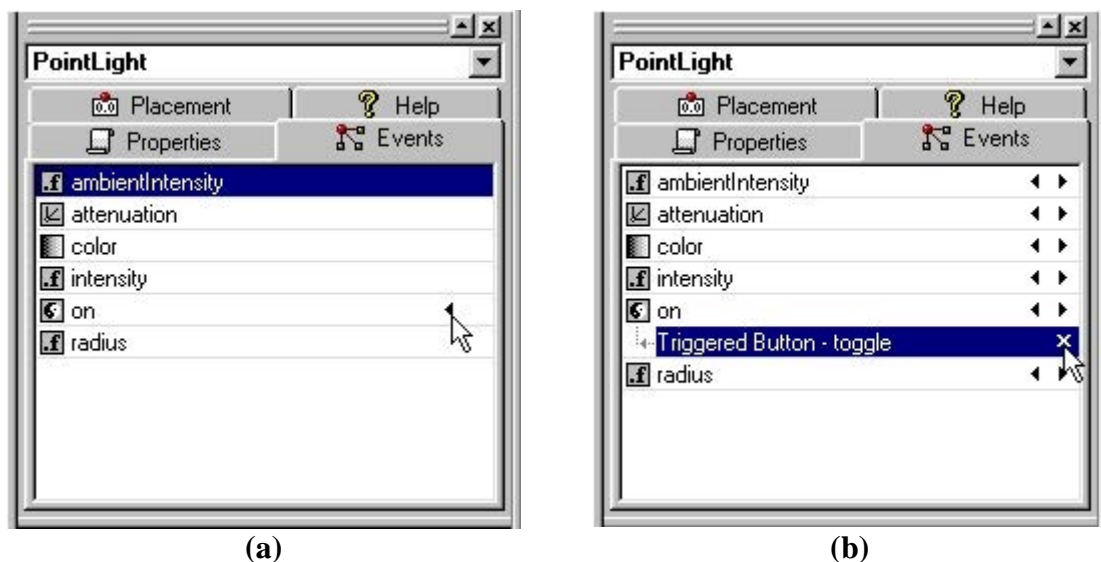


Figure 3.16: Selecting a target for an event

### **3.3.5 Macromedia Flash**

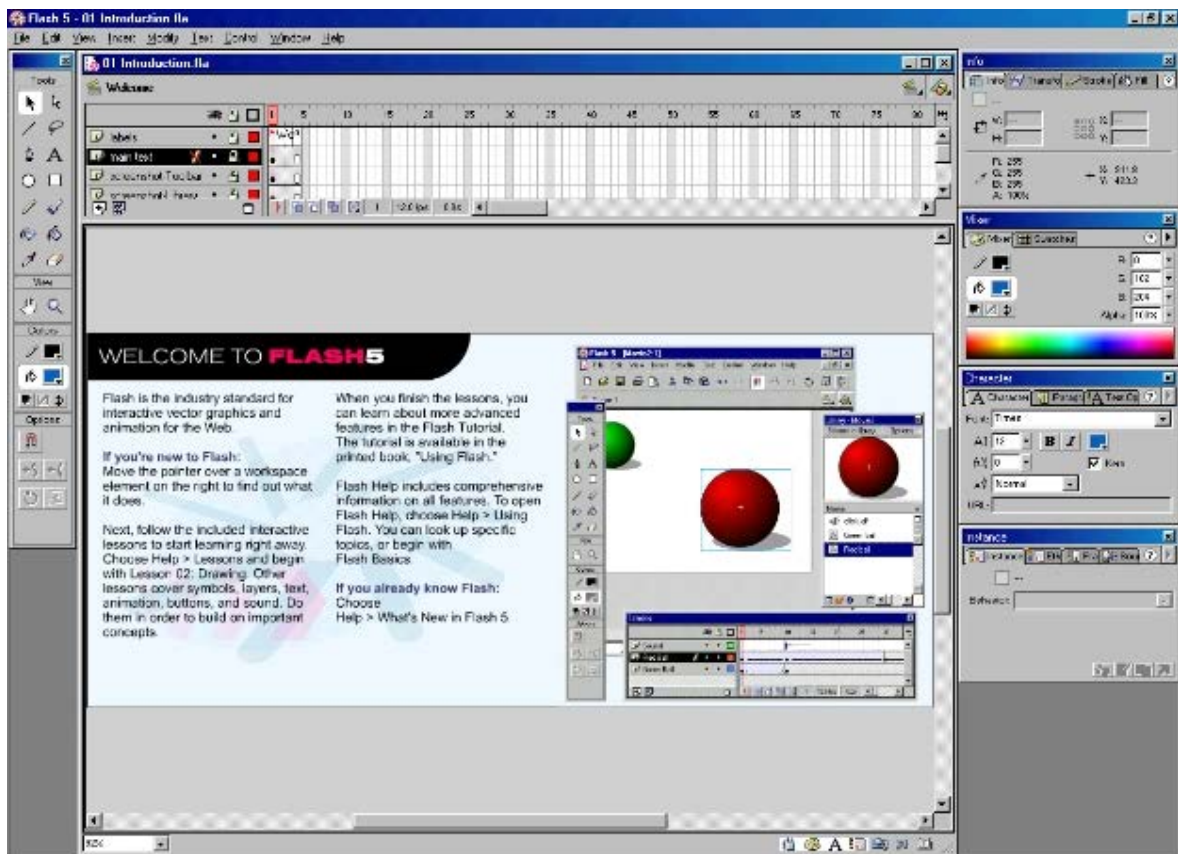
Macromedia Flash is a 2-D authoring tool that uses the Cast/Score/Scripting paradigm. Flash provides the author with powerful tools for animation and creating interactive multimedia applications that are widely used on the Internet today.

The reason for selecting Macromedia Flash is because it has very similar goals and application areas to that of MPEG-4. Flash allows the author to create interactive videos which are mainly animation based. Furthermore, Flash movies can be streamed over the Internet and are widely used on web pages.

Although animation support is not a primary goal of the project, there may be a lot that can be learned from evaluating Macromedia Flash. Unlike Flash, none of the previously evaluated systems made use of a timeline or score, which may be an useful tool in a MPEG-4 authoring system.

#### **3.3.5.1 Creating a New Scene**

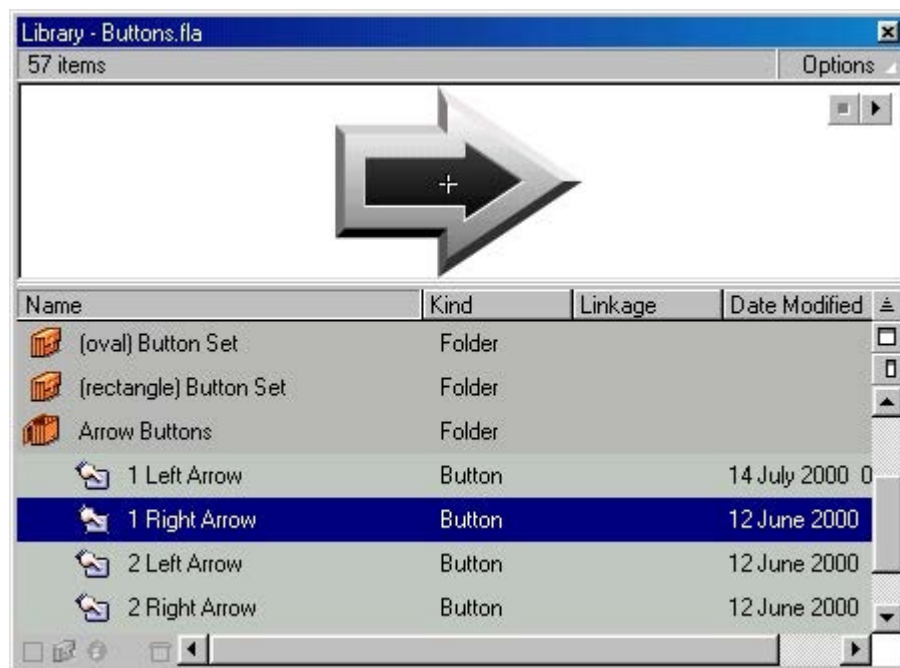
Figure 3.17 shows the opening screen of Flash. The development area is divided up into a tool palette on the left, an editing area and score in the center of the screen. The right-hand side of the screen contains property windows that display information on the currently selected object or scene. Flash movies consist of a group of scenes. When editing an author creates each scene separately from the other. Therefore each scene will have its own score associated with it.



**Figure 3.17: Opening screen of Macromedia Flash**

The tool palette provides the author with a standard set of tools for constructing a scene. Flash also provides the author with a set of pre-constructed objects such as buttons and animations contained in the "Library". The "Library" is similar to the "Object Gallery" used in ISA. Objects can be imported into the "Library" and the author can then insert the objects into a scene.

In order to create a new movie the author must click on "File" and then "New". When a new movie is created, it consists of a single scene and the movie size and background colour has a default setting. To change the size or background colour of a movie, the author must right-click the scene and use "Movie Properties" or click "Modify" on the main menu and then "Movie". To add an additional scene to the movie, the author may click on "Insert" on the main menu and then "Scene".



**Figure 3.18: The Flash object library**

### 3.3.5.2 Adding Objects to the Scene

In order to add objects to a scene, the author can select an object from the tool palette and then draw the object on the scene. For example, to add a rectangle to a scene the author would click on the rectangle object on the tool palette and then click on the scene and draw the selected object on the scene. Another method to add objects to a scene is to select an object in the "Library" and drag it onto the scene.

Once objects are in a scene, the author can select an object using the "Arrow Tool". Once an object is selected, the author may edit its properties through the property windows. Flash provides the author with two arrow tools. The first, which has already been discussed, is the "Arrow Tool" and the second is the "Subset Tool". The "Subset Tool" is used to select subparts or subsets of an object.

### **3.3.5.3 Adding Events to Objects**

Macromedia Flash allows the author to add "Actions" to its objects. Actions determine the behaviour of an object. To add an action to an object the author must set the behaviour of the object. The behaviour of an object in Flash terms means what kind of behaviour the object will have. For example, the behaviour of an object may be set as a button or movie clip etc.

Once an object has been assigned the behaviour of a button, it will respond to the viewer clicking on it or moving their mouse pointer over it. To set the response of the object to events the author must write a script.

## **3.4 Conclusions of Study**

The previous sections provided an overview of authoring paradigms and discussed the extant systems analysis of the selected authoring tools. This section will discuss the selection of a paradigm that will be suitable for an MPEG-4 authoring tool as well as the results of the extant systems analysis. The results of the extant systems analysis will be discussed under the same headings used in the extant systems analysis itself (Section 3.3).

### **3.4.1 Selecting an Appropriate Authoring Paradigm**

There are several different paradigms available (as discussed in Section 3.2) and it is possible that there is more than one paradigm that may be applicable to MPEG-4 authoring. The underlying data and functionality of a system are not the only factors that determine the appropriate paradigm for an authoring tool. The goal or purpose of the authoring system is another factor that plays a large role. For example, the underlying data for MPEG-4 suggests a hierarchical object paradigm. However, if the goal of the system is to produce slides for a presentation, then the hypermedia linkage paradigm would be a better choice.

As stated in Chapter 2, all MPEG-4 objects are stored in a hierarchical tree with the media objects forming the leaves. Therefore, according to the data structure of MPEG-4, the hierarchical object paradigm would be an appropriate paradigm for a MPEG-4 authoring tool.

The next consideration would be the goals of the system. As stated in Chapter 1, the system will allow an author to construct an interactive scene containing video, audio, image and geometry objects. Therefore, the system will be a general purpose authoring tool and not focused on a single goal. For example, the tool will not be aimed at creating presentations or CBT programs. Another goal of the system is to minimise the amount of code the author must write. Ideally the author should not have to write any code.

According to the requirements above, all paradigms that use scripting or that cause the authoring to specialise in creating a certain kind of application can be discarded. Another useful source of information for selecting a paradigm can be found in the evaluation of the ISA and MPEG-4 ToolBox. It is clear that both MPEG-4 ToolBox and ISA use the hierarchical object paradigm. Therefore, the authoring paradigm best suited for a general purpose 2-D MPEG-4 authoring tool, with goals as discussed in this section, is probably the hierarchical object paradigm.

The following sections will discuss the findings of the extant systems analysis with regard to the process of creating a scene, populating it with objects and adding interaction to the objects.

### **3.4.2 Creating a New Scene**

The process of creating a new scene follows the standard windows interaction style by allowing the author to click on the new file icon or on "File" and then "New" on the menu bar. Two of the evaluated systems were aimed at creating 3D scenes and therefore the author did not have to set the initial size of the scene. For a 2-D system however, this



information will be required. When creating a new scene, the author may be prompted and the scene size would be requested. Alternatively a new scene will always be created with a default size which may be altered by the author at any stage as it is done in Flash.

Adding a background colour to the scene will differ between 3-D and 2-D systems. In the 3-D systems a background colour or object is optional where in the case of a 2-D system it is not. Another difference is that the 3-D systems use a background that is divided into sections for sky and ground colour which would not be applicable in a 2-D tool. Therefore, in 2-D a background object will be added automatically. Setting the colour for the background will be included in the process of setting the size of the scene.

### **3.4.3 Adding Objects to the Scene**

In all evaluated systems, the author may select an object from a tool palette and place it in the scene. In the 3-D systems, when an object is placed in a scene it is automatically inserted into the scene tree. A object in a scene may be selected by either clicking on it or by using the scene tree if available. When an object is selected, the author may change its physical attributes and move it to any position in the scene.

The object palettes used by the systems differ in complexity. The standard palette consists of a set of icons representing the default objects that may be used by the author. The default objects would be video, audio, image and geometrical objects like rectangles and lines. The more complex tool palette contains the same default set of icons but also has a variable set of objects. Here the author may import objects onto the tool palette from where they can be inserted into the scene. Therefore, instead of having an icon that represents an image or any other media object the author can import a media object onto the palette. In the case that the imported media object is an image or video object it may be represented by a thumbnail view on the tool palette.

The evaluation of the 3-D systems showed that there is an object missing in the scope of the project. Both systems make use of timers which have not been included in the initial scope of the project. Timer objects will allow the author to have more control over the

playback of the scene and for that reason it will be included in the scope of the system. The timer object can be dragged onto a scene and its start and end time may then be set. Events can then be added to the timer object which may be triggered at selected points in time.

### **3.4.4 Adding Events to Objects**

The addition of events to objects, in all the evaluated systems, are completed by a set of similar steps. These are: Select the object that will be the source of the event; Select the kind of event; Select the target object for the event and finally specify the action of the event.

The process of adding events to objects will play an important role in the success of this project. Adding events to objects in both 3-D systems proved to be a challenge and was very confusing. An author without knowledge of VRML and BIFS would not be able to use the 3-D systems without firstly studying the concepts of the language. In Flash, the author was required to write a script to determine the behaviour of the object and had no choice but to learn how to write Flash scripts.

Both the evaluated 3-D authoring tools allow an author to construct a scene without writing code. Yet both 3-D systems, and specifically MPEG-4 ToolBox, do not abstract the authoring process from the coding process. MPEG-4 ToolBox uses BIFS terminology and the authoring tool mimics the coding process. If the author is not required to write code then why expect him/her to know how the code works? The authoring process should be abstracted further from, and should not model, the coding process. For example, the author should not be exposed to terms like "Position Interpolators" and "RepeatS".

## **3.5 Summary**

Authoring tools are programs that allow an author to manipulate different media elements spatially and temporally, and add interactive behaviour to them. Authoring tools can be

used to compliment or replace the coding process. Providing the content author with a multimedia authoring tool would reduce the prerequisite knowledge required of the author and increase the number of potential users of MPEG-4 applications.

Authoring tools come in a wide variety of shapes and forms which can be grouped into authoring paradigms. An authoring paradigm is the metaphor or methodology that the tool uses to help an author accomplish his or her task. This chapter provided a summary and examples of the major paradigms used in the field today and the hierarchical object paradigm was chosen as a suitable paradigm for this project.

The latter part of the chapter contained an extant systems analysis of a 3D MPEG-4 authoring tool, a VRML authoring tool and Macromedia Flash. This analysis formed part of the OVID methodology which was selected for the project. OVID (Objects, Views, and Interaction Design) is a formal methodology for designing user friendly programs based on the analysis of user's goals and tasks.

The results of the analysis was discussed and it was shown that MPEG-4 and VRML authoring tools have similar characteristics. It was found that the initial scope of the project as stated in Chapter 1 had omitted the inclusion of a timer object that could prove to be very useful in the authoring process. Therefore the scope of the project was enlarged to include a timer object.

Two important conclusions were made from the analysis. Firstly, the authoring process should be abstracted from, and should not model, the coding process. Secondly the process of adding events to objects must be kept as simple and intuitive as possible as it will play a direct role in the success of the system.

In the next chapter the MPEG-4 framework will be discussed. The aim of the chapter is to expand on the knowledge gained in Chapter 2, which was an introduction to the standard and its functionality. The chapter will provide a more technical view of the standard and will discuss how the standard provides the functionality as discussed in Chapter 2.

## Chapter 4: The MPEG-4 Framework

### 4.1 Introduction

Chapter 2 provided an overview of MPEG-4 by discussing the scope of and functionality provided by the MPEG-4 standard. MPEG-4 addresses every aspect of the creation and streaming of interactive multimedia content over low and high bandwidth connections, irrespective of the transfer protocol.

Chapter 3 discussed authoring systems and the paradigms used by different systems. Chapter 3 also included an analysis of three authoring systems. The results of the analysis will provide useful information for the design of a 2-D MPEG-4 authoring tool.

The aim of this chapter is to provide a more in-depth and technical view of the standard by discussing how it provides the functionality discussed in earlier chapters. MPEG-4 makes use of many different technologies and existing standards. By combining these technologies and standards, MPEG-4 produces a framework that enables the standard to achieve its goals. A framework is a re-usable, semi-complete application that can be used to produce complete applications [FS1997].

This chapter will introduce the MPEG-4 framework by providing an overview of selected items in the MPEG-4 framework. The size and scope of the standard makes it impractical to discuss the framework for the entire standard. Therefore only subsections of the standard that will play a role in the creation of a MPEG-4 authoring tool will be discussed. These sections are BIFS, MPEG-4 encoding algorithms and the MPEG-4 file format. The chapter will also contain a discussion on the process of creating a MP4 file.

A secondary objective of the chapter is to provide an introduction to the MPEG-4 framework, especially for researchers who wish to do further work on this project. Before future work on the project can start, the prospective researcher, after studying the theory of

the MPEG-4 standard, must gain an understanding of the practical process of creating MPEG-4 content. Therefore this chapter will, with practical examples, provide an introduction into BIFS programming and the process of creating a MP4 file.

## 4.2 Binary Format for Scenes

In order to facilitate the coding of individual objects into a scene, MPEG have developed a binary language known as Binary Format for Scenes (BIFS). Through the use of BIFS the content author is allowed to control the playback process of a scene and add interaction which will enrich the viewer's experience. BIFS builds on and extends many concepts from the Virtual Reality Modeling Language (VRML). The spatial and temporal arrangements as well as the properties of the objects in a scene are controlled by BIFS. The author can therefore add interaction to a scene by linking the properties of the object to an event which may be triggered by the viewer or another object.

BIFS code is streamed in its own elementary stream and kept separate from the media objects. By separating the BIFS code from the media objects, the content author is able to extract and edit the BIFS code without decoding the media objects. Separating BIFS code from the media objects has many advantages. When streaming a MPEG-4 application the BIFS code can be downloaded and executed before the media objects are completely downloaded. Furthermore if an author wishes to modify the BIFS code in a MPEG-4 application, he or she can do so without having to decode the media objects.

### 4.2.1 Differences Between MPEG-4 and VRML

BIFS as stated before, is based on VRML and extends VRML to provide a programming language that satisfies the goals of the MPEG-4 standard. There are however, several differences between the two standards. Table 4.1 contains a summary of the differences between MPEG-4 and VRML.

<b>Binary Format for Scenes</b>	<b>Virtual Reality Markup Language</b>
Multiple profiles	Single Profile
Timing and synchronization nodes	No provision of synchronization
Synthetic audio effect nodes	No audio and synthetic effects nodes
Human facial animation nodes	Supports avatars but no animation
Supports streaming	No support for streaming
Supports MPEG-J	MPEG-J not supported
Specification of rendering performance	No control over quality
BIFS code is compressed	VRML code is uncompressed

**Table 4.1: The differences between BIFS and VRML**

MPEG-4 specifies several profiles where VRML has only one general profile. The advantage of having multiple profiles is that it allows encoder and decoder scalability as explained in Section 2.2.4. By using profiles to specify different levels of complexity it is possible to create an MPEG-4 encoder or decoder that does not support complete functionality of the standard. VRML has a single profile and therefore all encoders and decoders must support the functionality of the entire VRML97 standard. The formal designation of the VRML standard is ISO/IEC 14772.

The hierarchical scene tree that represents a MPEG-4 scene, is constructed using two general types of nodes. These are the media object nodes which form the leaf nodes of the tree and the scene description nodes which contain data describing the scene. The MPEG-4 standard specifies additional nodes to those found in VRML. Timing and synchronisation nodes do not exist in VRML nor does audio and synthetic effects nodes.

Each media object is separately placed in the hierarchical scene tree, thus separating video and audio from each other. The timing and synchronisation nodes play important roles in the synchronization and coherent playback of the two separate media objects. The audio and synthetic effect nodes provide the functionality needed to create synthetic or synthesized speech and music controlled by a structured language.

Both standards support the use of avatars. However MPEG-4 adds additional functionality by animating the facial expressions of the avatar using facial expression files or FAP files. With the use of these files, the lip movements of the avatar can be synchronised with the audio file, making the facial expressions of the avatar more realistic.

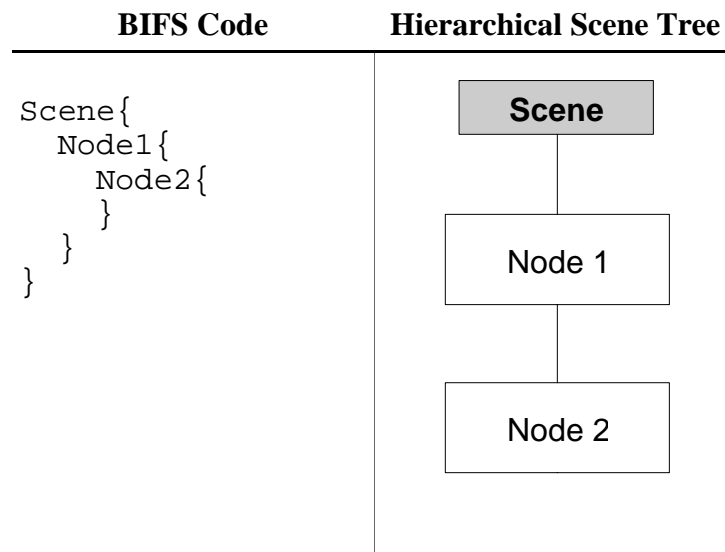
VRML scenes do not support streaming and the entire scene must be downloaded before it can be viewed. In MPEG-4 the playback of scenes can occur during the download process. MPEG-4 allows the content author to control the playback process using MPEG-J. MPEG-J, as explained in Chapter 2, specifies an API allowing Java code to interact with MPEG-4 players, making it possible for content creators to have more control over the playback process. Further more, MPEG-J and BIFS allow the content creator to specify the rendering performance and therefore priority of individual media objects during the playback process. Finally VRML uses uncompressed text to describe the scene. BIFS uses binary code, which is 10 to 15 times smaller.

## **4.2.2 Brief Introduction to BIFS Programming**

In the MPEG-4 model, a scene consists of a set of objects. These objects may be media objects or scene descriptor objects. Each object forms a node in the hierarchical scene tree, with the media objects forming the leaf nodes.

The hierarchical scene tree is a graphical representation of a MPEG-4 scene that is constructed and controlled by BIFS code. In general the BIFS code takes on the form shown in the BIFS Code segment as depicted in Figure 4.1. The hierarchical Scene tree segment of Figure 4.1 depicts the hierarchical scene tree representing the BIFS code on its left. In BIFS code each scene consists of a set of nodes and each node, depending on the type of node, can contain any number of child nodes.

The different nodes that can be placed in a scene include the image, video, audio, text and geometry objects. As an introduction to BIFS, the code required to insert geometry, text and image objects, will be discussed. The image, video and audio nodes are similar and therefore only the image node will be mentioned.



**Figure 4.1: General form of BIFS code**

The BIFS code in Figure 4.2 represents a single node in the hierarchical scene tree which in this case is a 2-D yellow circle. The actual MPEG-4 scene is displayed using the EnvivioTV plug-in [Env2002] with the QuickTime player from Apple Computing [App2003].

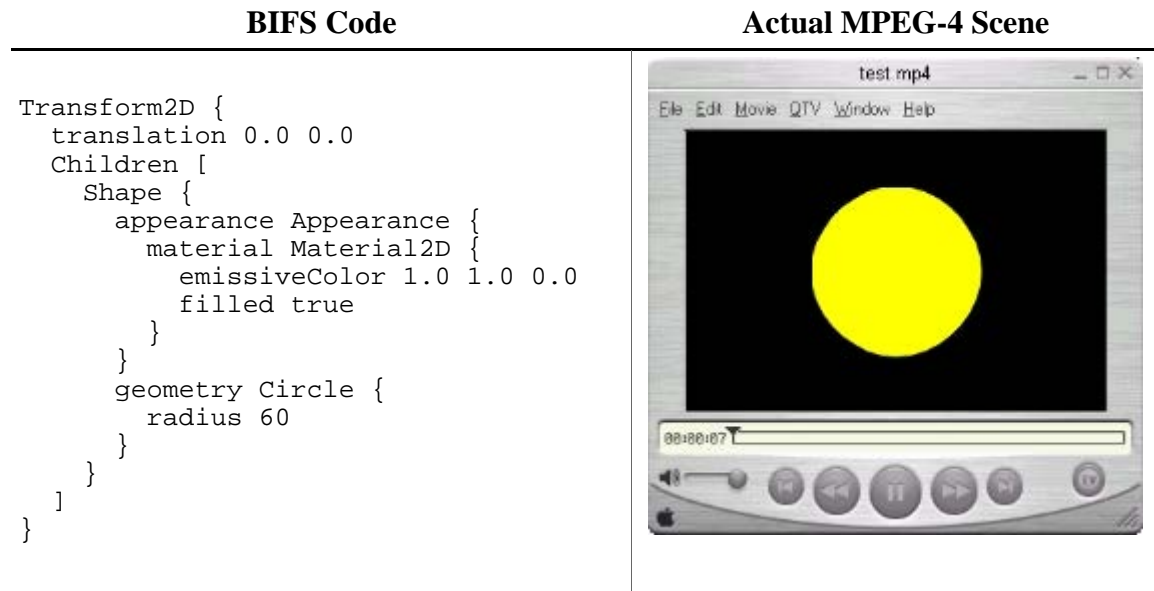
An explanation of the code is as follows: `Transform2D` specifies that the following node will be two-dimensional. The keyword `translation` is used to specify the position of the node relative to the point of origin in the scene. The BIFS coordinate system uses the middle of the scene as a point of origin. For example, if the `x` and `y` coordinates of the circle are equal to zero, the circle will be centered in the middle of the scene.

The next step is to add children to the node by using the `children` keyword. At this stage any number and kind of nodes can be added to the parent node. In this case a single shape node has been added to the parent node. Once a shape has been added the appearance of that shape can be set as well as the geometry of the shape.

The colour of the circle is set by using `emissiveColor` and it is then filled by setting `filled` to true. The colour is set using RGB values as a percentage out of 255. Thus a




value of 1.0 is equal to 255 and a value of 0.7843 is equal to 200. The key word `geometry` is used to set the geometry of the object to a circle and `radius` is used to set the size of the circle.



**Figure 4.2: BIFS Code for a filled yellow circle**

The BIFS code in Figure 4.3 for text is similar to that of the circle explained above. The scene contains a `Transform2D` node with a `shape` node as a child. The main differences are that the `geometry` of the object is set to `Text` and that there are more settings for text objects than circle objects. To set the text, the `string` value must be set to the desired text value. In this case the text is "Hello World". To display multiple lines of text the input string must be subdivided with double quotes. For example, if you wish to have "Hello" on one line and "World" on the next your BIFS command would be: `string [ "Hello" "World" ]`.


BIFS Code	Actual MPEG-4 Scene
<pre> Transform2D {   translation 0.0 0.0   children [     Shape {       appearance Appearance {         material Material2D {           emissiveColor 1.0 1.0 1.0         }       }       geometry Text {         string [ "Hello World" ]         fontStyle FontStyle {           family [ "Times" ]           justify [ "MIDDLE" "MIDDLE" ]           size 40.0         }       }     }   ] } </pre>	 <p>The screenshot shows a video player window with the title 'test mp4'. The main display area is black with the text 'Hello World' centered in a white serif font. Below the display area is a playback control bar with a progress indicator and several control buttons (play, stop, previous, next, full screen).</p>

**Figure 4.3: BIFS code for displaying text**

The BIFS code in Figure 4.4 contains a Transform2D node with a shape node as a child similar to the previous examples. The code in Figure 4.4 will display an image in the scene. For the shape node, a texture has been specified and the geometry is a Bitmap.

In this case the shape node has no colour associated with it but instead a texture and an url. The url is an identifier that links to a node in the object descriptors, which contains the file path to the image file. The scale of the bitmap is set in the geometry section and is given as a percentage of height and width in relation to the original size of the bitmap.

In this example a new keyword is listed under material. All visual objects that have a material associated to it can be set to different levels of transparency by using the transparency keyword. The transparency of the object is set by using a percentage value where 1.0 is completely transparent and 0.5 is semitransparent.

BIFS Code	Actual MPEG-4 Scene
<pre> Transform2D {   translation 0.0 0.0   children [     Shape {       appearance Appearance {         texture ImageTexture {           url [ "1" ]         }         material Material2D {           transparency 0         }       }       geometry Bitmap {         scale 1.0 1.0       }     }   ] } </pre>	 <p>The screenshot shows a video player window titled "test.mp4". The window has a menu bar with "File", "Edit", "Movie", "QTV", "Window", and "Help". The main display area shows a wolf standing in a snowy, mountainous landscape. Below the video frame is a progress bar showing "00:00:16" and a set of playback controls including play/pause, stop, previous, and next buttons.</p>

**Figure 4.4: BIFS code for displaying an image**

Once the BIFS code has been written for a MPEG-4 application the author must write the object or scene descriptors. Object descriptors store configuration information and metadata about objects or elementary streams. In other words, object descriptors specify the configuration of the scene and objects used in the BIFS code. Object descriptors are responsible for setting properties like the screen size of the application, the synchronisation, timing and priority of objects, the version of BIFS code and the url of files to be included into the MPEG-4 application.

Figure 4.5 shows an example of the object descriptors for a MPEG-4 application that does not contain any media objects. The descriptors listed are always required when creating a MPEG-4 application and represent the bare minimum of the possible settings. The object descriptors are divided into two sections. The first is the `InitialObjectDescriptor` which sets the initial properties of the MPEG-4 application. The second is the updates made to the `InitialObjectDescriptor` which set the properties of the media objects.

Each object descriptor has its own unique `objectDescriptorID` and `es_id` (elementary stream identifier). For the purposes of this introduction the only important part

of the `InitialObjectDescriptor` is the `decSpecificInfo`. Here the version of the BIFS language is set and the screen size of the application is specified. If the `decSpecificInfo` is set to `BIFSConfig` then the application will use BIFS Version 1 code if it is set to `BIFSV2Config` then it will make use of BIFS Version 2 code. To set the screen size of the MPEG-4 application (in pixels), the author must use `pixelWidth` and `pixelHeight` keywords.

### BIFS Code

```
InitialObjectDescriptor {
  objectDescriptorID 1
  esdescr [
    ES_Descriptor {
      es_id 1
      streamPriority 0
      decConfigDescr DecoderConfigDescriptor {
        objectTypeIndication 1
        streamType 3
        decSpecificInfo BIFSConfig {
          pixelWidth 300
          pixelHeight 200
        }
      }
      slConfigDescr SLConfigDescriptor {
      }
    }
    ES_Descriptor {
      es_id 2
      streamPriority 0
      decConfigDescr DecoderConfigDescriptor {
        streamType 1
      }
      slConfigDescr SLConfigDescriptor {
      }
    }
  ]
}
```

**Figure 4.5: The object descriptors for a MPEG-4 application**

After the `InitialObjectDescriptor` has been completed, the object descriptors for the individual media objects can be added. This forms the second part of the object descriptors. Figure 4.6 shows the addition or update of one object descriptor of an image object. This code will be added on directly below the `InitialObjectDescriptor`.

The `AT 0` keyword specifies that all actions below it are to be executed when the time of the application is equal to zero. When a MPEG-4 application starts, a timer within the application starts with it. Therefore all the commands within the `AT 0` keyword will be executed the moment the application starts. The `AT` keyword can be set to any time and need not only be confined to adding object descriptors. The `AT` keyword (amongst other things) can also be used to add and remove nodes and routes from the application. For example, the author can specify the addition of an image object to a scene at a given time using the `AT` keyword.

### BIFS Code

---

```

AT 0 {
UPDATE OD [
  ObjectDescriptor {
    objectDescriptorID 1
    esdescr [
      ES_Descriptor {
        es_id 3
        streamPriority 0
        decConfigDescr DecoderConfigDescriptor {
          objectTypeIndication 108
          streamType 4
        }
        slConfigDescr SLConfigDescriptor {
        }
        muxInfo muxInfo {
          fileName "Image\wolf.jpg"
        }
      }
    ]
  }
]
}
]

```

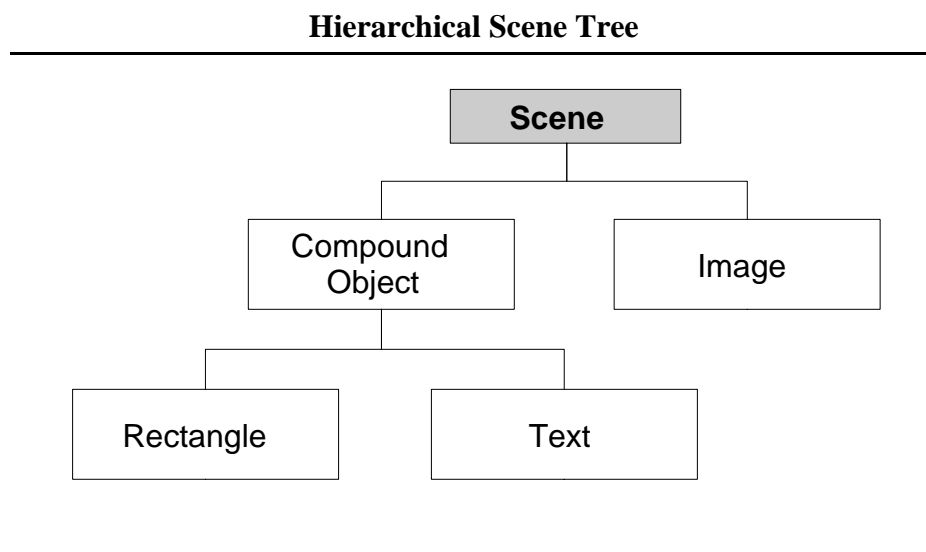
---

**Figure 4.6: Object descriptor for an image object**

The `UPDATE OD` keyword in Figure 4.6 means that an object descriptor will be added to the application. There can be any number of `UPDATE OD` commands within a single `AT` statement. For the purposes of this introduction the most important parts of the `UPDATE OD` is the `objectDescriptorID` and the `muxInfo`. The `objectDescriptorID` is the identifier that is used in the `url` of a media object. For Figure 4.4 the `url` for the media object is 1 and therefore the `objectDescriptorID` for the the object descriptor

of the media element will also be 1. The `muxInfo` contains the path to the media object that must be included in the MPEG-4 file. The path will be used during the multiplexing process of the MP4 file.

Figure 4.7 below shows a hierarchical scene tree for the BIFS code in Figure 4.8. The BIFS code in this example will generate a scene containing text, a rectangle and an image. The text and rectangle object have been grouped together to form a compound object.



**Figure 4.7: A more complicated hierarchical scene tree**

The benefit of using compound objects is that all the objects in a compound object can be treated as one object. For example, an event triggered by a mouse click on the compound object will monitor all the objects in the scene that form part of the compound object.

Another use for a compound object is to group all objects in a scene together. As a MPEG-4 application consists of multiple scenes, each scene will contain its own objects. If all the objects for each scene are grouped together, it will make the updates required in the transition from one scene to another a much simpler task.

---

**BIFS Code**


---

```

Group {
  children [
    Transform2D {
      translation 0 0
      children [
        Transform2D {
          ...
          geometry Text {
            string [ "Hello World" ]
          }
          ...
        }
        Transform2D {
          translation 0.0 220.0
          ...
          geometry Rectangle {
            size 630.0 120.0
          }
          ...
        }
      ]
    }
    Transform2D {
      translation 0.0 0.0
      children [
        Shape {
          appearance Appearance {
            texture ImageTexture {
              url [ "1" ]
            }
            ...
          }
          geometry Bitmap {
            scale 1.0 1.0
          }
          ...
        }
      ]
    }
  ]
}

```

---

**Figure 4.8: A more complicated example of BIFS code**

### 4.2.3 Extensible MPEG-4 Textual Format (XMT)

Once BIFS code has been compiled into binary form, it is not possible to decompile it to give an exact copy of the original code. This is the case not only for BIFS but for most programming languages that are compiled into binary form including: C++, Pascal, etc.

One reason for this is that a compiler, in many cases, goes through an optimisation process when compiling the code to binary form. In this optimisation step the compiler restructures and changes the code in order to optimise the performance of the compiled application. Another reason would be that there are different decompilers available, which are

implemented at different profiles and levels. Therefore, not all of them implement the full functionality of MPEG-4. As a consequence, only a subset of the BIFS code will be decompiled and the original intentions of the author may be lost.

As it is not possible to decompile binary BIFS code to its original form, there is no easy way for content authors to exchange code unless the uncompiled code is shared. XMT has been designed to provide an exchangeable format between content authors whilst preserving the author's intentions in a high-level textual format [KWC2000]. XMT is an author friendly coding format that is designed to cater for authors who know X3D [X3D2002], SMIL [W3C2001] and HTML. Therefore XMT does not only cater for MPEG-4 authors but also for X3D and SMIL authors. The XMT framework consists of two levels of textual syntax and semantics: the XMT-A format and the XMT- $\mathfrak{D}$  format.

XMT-A is based on XML and closely mirrors the binary representation of MPEG-4 content. The goal of the XMT-A format is to provide a deterministic one-to-one mapping to ISO/IEC 14496:1999 binary representations and to be interoperable with Extensible 3-D (X3D) [KWC2000]. X3D developed by Web3D is an open standard enabling the deployment of visually rich 3-D graphics applications. X3D is the latest step in the evolution of VRML and remains fully backward compatible with VRML97.

XMT-A contains a subset of X3D, and specifies its own representations of MPEG-4 specific features using the X3D format. By doing so X3D and MPEG-4 XMT code become interoperable and it is possible to translate a X3D application to MPEG-4 and vice versa.

The XMT- $\mathfrak{D}$  is based on SMIL and offers a high-level abstraction of MPEG-4 content. Synchronized Multimedia Integration Language (SMIL) was developed by W3C and is an XML-based language. SMIL provides techniques for defining the temporal and spatial attributes of multimedia objects as well as the use of hyperlinks.

For every XMT- $\mathfrak{D}$  element, there is a non-deterministic mapping to a sequence of XMT-A elements. Therefore one element can be represented by different sequences of XMT-A. XMT-A however, has only one binary representation. Therefore the binary code can be



decompiled to give the original XMT-A code that was used to create it. The XMT-A code can then be translated to equivalent XMT-D code. The XMT-D code however, will not be exactly the same as the original.

The XMT format is interchangeable between an SMIL player, VRML player, and an MPEG-4 player. Figure 4.9 presents a graphical description of the interoperability of the XMT. The XMT-D format can be parsed and played directly by a SMIL player. It could then be parsed to the corresponding X3D nodes and played back by a VRML player, or compiled to create an mp4 file, which can then be played by an MPEG-4 player.

In addition to interoperability with X3D and SMIL, XMT also integrates with MPEG-7. MPEG-7 is an emerging standard for describing multimedia content and was discussed in Chapter 2. The integration of MPEG-7 with XMT will enable the content-based retrieval of MPEG-4 objects.

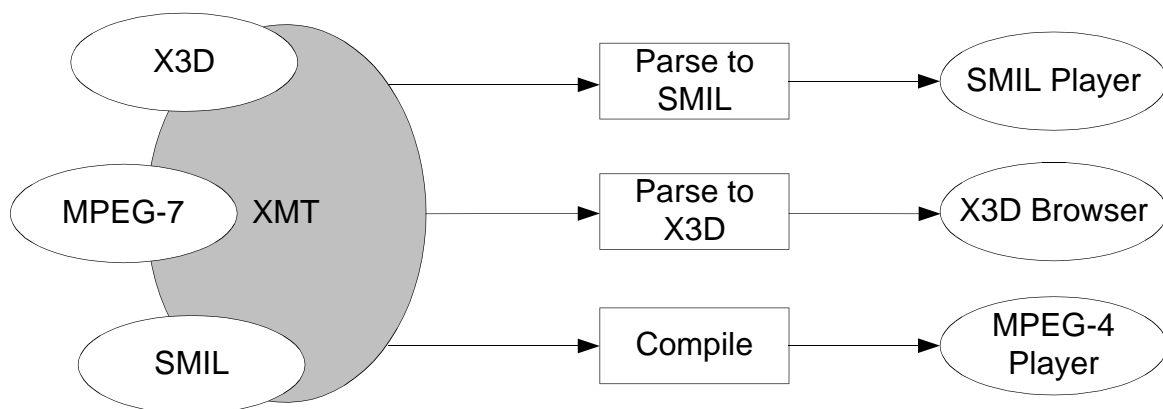


Figure 4.9: Interoperability of XMT

### 4.3 MPEG-4 Video Compression

The MPEG video standards in general are a composition of several other international standards for the compression of digital video. The MPEG video algorithm is a highly

refined version of a class of video algorithms called motion-compensated discrete cosine transform or MC-DCT algorithms [GBL+1998]. MC-DCT algorithms use a variety of compression techniques, which can be applied to a variety of signals. These include:

- " Temporal prediction exploits redundancy between video frames;
- " Frequency domain decomposition decomposes spatial blocks of image data in order to exploit statistical and perceptual spatial redundancy;
- " Quantisation reduces data rates while minimising the loss of perceptual quality; and
- " Variable-length coding exploits redundancy in the symbol sequence resulting from quantisation.

The MPEG-4 video algorithm builds on and extends the MPEG-2 algorithm, which in turn is based on MPEG-1. Therefore, the general MPEG video algorithm will be discussed before discussing the MPEG-4 video algorithm.

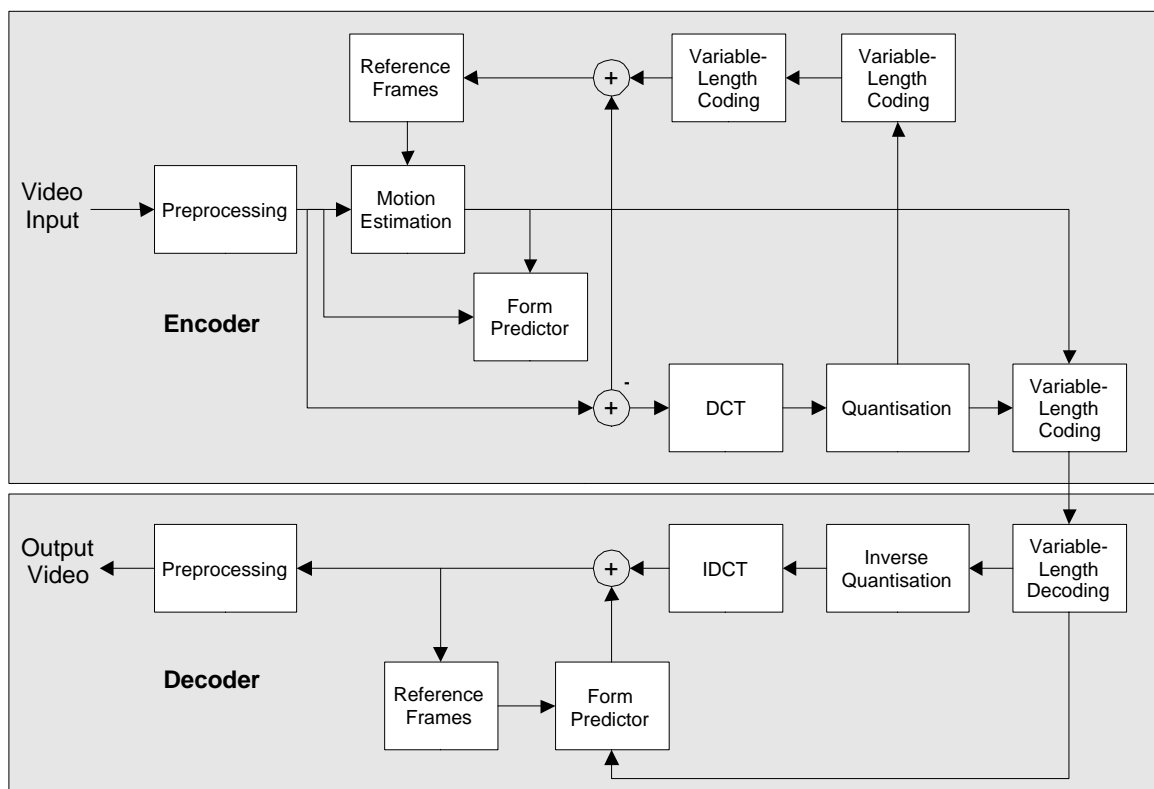
### **4.3.1 MPEG Video Compression**

The MPEG-1 standard in reality does not define a compression algorithm but defines a data stream syntax and a decompressor. Therefore, manufacturers can create their own compressors, thereby allowing a competitive marketplace [CC2000]. In practice the compressor makes use of a standard set of MC-DCT compression techniques as described above and the algorithm will now be discussed, making reference to these techniques. Figure 4.10 provides a diagram of the MPEG video compression algorithm.

In most cases sequential video frames are very similar to one another except when there are transitions between scenes. For instance, the background of a video scene may remain the same over a sequence of frames while only the foreground changes. Temporal prediction takes advantage of the similarity of the two frames by only recording the changes from one frame to another. A simple implementation of temporal prediction would be subtracting the value of each pixel in a frame from the corresponding pixel in the previous frame resulting in a difference frame.

The areas in the resulting difference frame that have a zero value represent the areas that have not changed since the previous frame. Difference frames only record the difference between the previous frame and the current frame. Therefore, less space is required to store the resulting frame. This process is called picture differencing.

The results from picture differencing can be enhanced by noting that changes across a sequence of frames are caused by the motion of objects. Motion compensation concerns itself with the tracking of areas of change across frames. Video sequences usually consists of objects that move as a whole. The object could be a car driving down the street or a football being kicked. Motion compensation attempts to identify the area of a moving object making it possible to record the movement of the object together with the changed pixels.



**Figure 4.10: The MPEG video algorithm**

MPEG compressors do not attempt to identify objects in a scene. Each scene is divided into blocks of 16x16 pixels known as macroblocks. The algorithm then tries to ascertain the position of the macroblock in the next frame. The difference frame is then constructed by subtracting each macroblock from its predicted counterpart resulting in more zero pixels.

The next step of the algorithm is temporal compression which makes use of I-, B-, and P-frames, also called I-, B-, and P-pictures. I-frames or intra frames in MPEG compression are the same as key frames. Key frames are frames that are left uncompressed or are only spatially compressed. Difference frames that use the I-frames are called P-frames or predictive frames. P-frames are always based on earlier I-frames, B-frames on the other hand are based on earlier and later frames. Bi-directionally predictive frames can use motion compensation from the next/previous I- or P-frame or both if needed.

A video clip can be encoded as a sequence of I-, P- and B-frames. The sequence used does not have to be fixed throughout the video but in general compressors use a repeating sequence known as a Group of Pictures (GOP). The GOP sequence will always start with a I-frame and a typical GOP sequence would be IBBPBB. Here the P-frame would depend on the initial I-frame. The B-frames will depend on the I- or P-frames that precede and follow them. Thus the first B-frame could depend on the I-frame and the following P-frame. Published measurements show that P-frames compress three times as much as I-frames and B-frames compress one and a half times as much as P-frames [Tud1995].

B-frames are computationally more demanding than the other frames but provide better compression, forcing a trade-off between speed and compression when a GOP sequence is selected. Another factor that will affect performance is the random access to frames in a video. Random access in a video allows the viewer to start viewing the video at any point within the video. Therefore the viewer is not limited to watching a video from start to end, but able to start and move to any point by specifying the time of that point. Providing random access to P- and B-frames is complicated and is avoided by including a sufficient number of I-frames to allow random access to several I-frames per second.

Not only are B-frames more complex to reconstruct they also need information from the I- and P-frames that follow them in the GOP sequence. The actual sequence in which the frames appear in the GOP is called the display order. The solution to the problem is to re-order the frames in a suitable order called the bitstream order. The bitstream order is the order in which the decoder requires the frames to be able to reconstruct the video. Therefore the I- and P-frames will be placed in the GOP before the B-frames that depend on them. As an example take the case of a MPEG frame sequence with a display order such as IBBPBBIBBPBBI. For this sequence the corresponding bitstream order will be IPBBIBBPBBIBB and the decoder will receive the I- and P-frames before receiving the B-frames that depend on them.

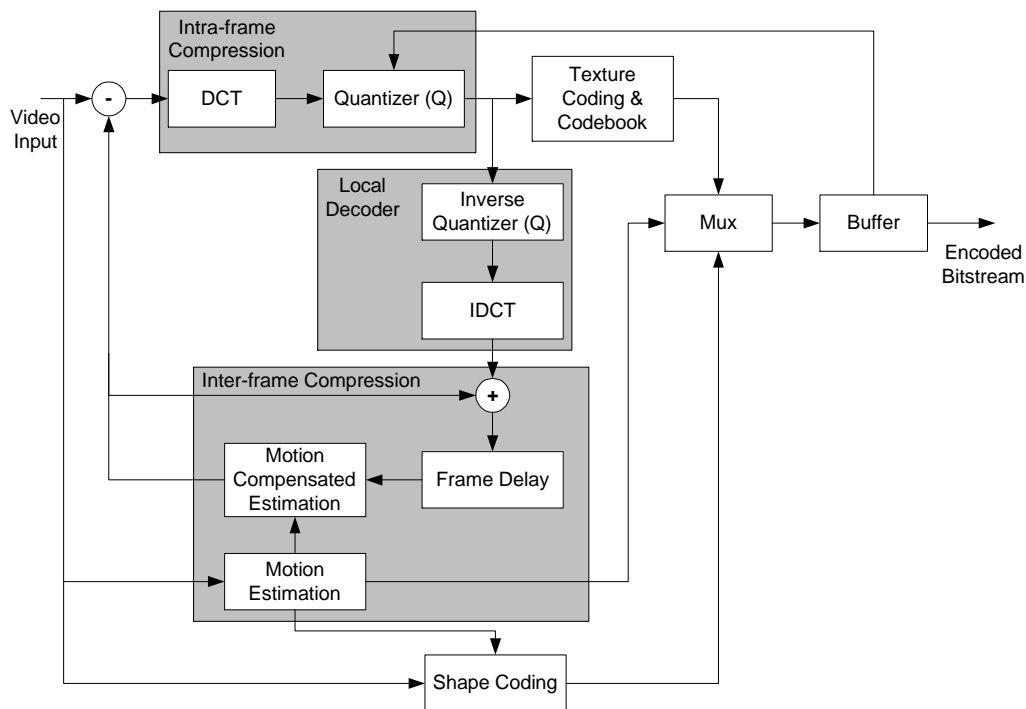
After motion compensation, the resulting information, be it an I-, P- or B-frame, is transformed using a discrete cosine transform (DCT). After transformation the DCT coefficients typically have 12 bits or more of precision. Quantisation is then applied and the number of bits needed is reduced while the perceptual quality of the frames remains the same. Both the quantised and other miscellaneous data exhibit statistical concentration so that variable length encoding can be employed to reduce the overall average bitrate. When data exhibits statistical concentration it contains a large amount of repetition in the data. MPEG makes use of modified Huffman codes, a type of lossless variable length coding, to exploit the redundancy in repeated data.

If the frame size is restricted to 352x240 and the frame rate is 30 fps, the video can be compressed to a data rate of 1.86 Mbits per second, which is the data rate specified for compact disk video. MPEG-1 is unable to compress interlaced or HDTV formats, hence the need for MPEG-2 video compression used for satellite TV and DVDs.

### **4.3.2 The MPEG-4 Video Algorithm**

MPEG-4 is the first MPEG standard that includes streaming support as a part of the standard [Aus2002]. The standard provides a flexible multimedia encoding format designed to support a wide range of bit rates suitable to cater for low bitrate wireless streaming to HDTV applications. Figure 4.11 shows the MPEG-4 video algorithm.

In addition to streaming support, the standard provides for Content Based Coding which allows the separate encoding and decoding of selected video data [Sik2002]. Through Content Based Coding, it is possible to identify and selectively decode and reconstruct video content of interest, referred to as Content-Based Scalability. For this purpose, MPEG-4 introduces Video Object Planes (VOPs). Each frame can be segmented into a number of arbitrarily shaped VOPs, which in turn may be encoded and decoded separately. In contrast to the video source format used for the MPEG-1 and MPEG-2 standards, the video input encoded by MPEG-4 thus no longer needs to be a rectangular region.



**Figure 4.11: The MPEG-4 video encoder algorithm**

Successive VOPs belonging to the same physical object in a scene are referred to as Video Objects (VOs). The shape, motion and texture information of the VOPs belonging to the same VO is encoded into a separate Video Object Layer or VOL. Decoding of all VOP-layers reconstructs the original image sequence. In order to separately reconstruct content only a single VOL or a set of Video Object Layers needs to be decoded [Ebr2001].

The MPEG-4 compression algorithm is based on the techniques already employed in the previous MPEG coding standards. The MPEG-4 coding algorithm encodes the first VOP similarly to an I-frame called an I-VOP. Each subsequent frame is coded using Inter-frame VOP prediction P-VOPs or Bi-directionally predicted VOPs (B-VOPs). Similar to the general MPEG encoding algorithm the encoder creates macroblocks and applies motion compensation techniques followed by DCT, quantisation and variable length encoding processes.

Using Content-Based Scalability, MPEG-4 provides the functionality to cater for different bandwidths and display capabilities as well as prioritised streaming of objects in video content. Spatial scalability allows the decoder to reconstruct a video object at different quality levels and resolutions depending on the capabilities of the playback device and medium. Temporal scalability has the same aim as spatial scalability. By using temporal scalability it is possible to provide different frame rates for different VOLs within the same video sequence. This makes it possible to assign a higher frame rate to an more important foreground VOs than a VOs in the background.

## **4.4 MPEG-4 Image Compression**

The MPEG-4 image and texture compression algorithm is based on the zero-tree wavelet coding scheme. Wavelet compression is an alternative to DCT encoding and does not suffer from the often visible blocking artefacts associated with DCT [Aus2002].

Wavelet compression is well suited for use in the MPEG-4 standard as it provides spatial and quality scalability as well as being able to encode images of arbitrary shape. In the case of MPEG-4, it provides eleven levels of spatial scalability and continuous quality scalability [Koe2001]. The wavelet formulation provides for scalable bitstream coding in the form of an image resolution pyramid for progressive transmission and temporal enhancement of still images.

## 4.5 MPEG-4 Audio Compression

MPEG-4 audio provides the functionality for the representation of natural audio and the generation of synthetic sound from a structured language. As with MPEG-4 video the audio component of MPEG-4 has a lot in common with the general MPEG audio coding scheme as it uses MPEG-2 AAC for encoding natural audio.

### 4.5.1 MPEG Audio Compression

The MPEG audio compression standard defines a family of algorithms that efficiently encodes a wide range of audio material ranging from speech and music to synthetic sound effects. As most MPEG standards do, the MPEG audio standard defines several conformance points known as layers. The layers are ordered from 1 to 3 in increasing complexity. Each layer is backwards compatible with the previous ones thus allowing a decoder of a certain level to decode all lesser layers.

Table 4.2 below provides a summary of the different MPEG layers. For typical music and soundtrack material, stereo audio can be coded well at rates of 64 kbps per channel with Layer 3, 128 kbps per channel for Layer 2 and 192 kbps per channel for Layer 1.

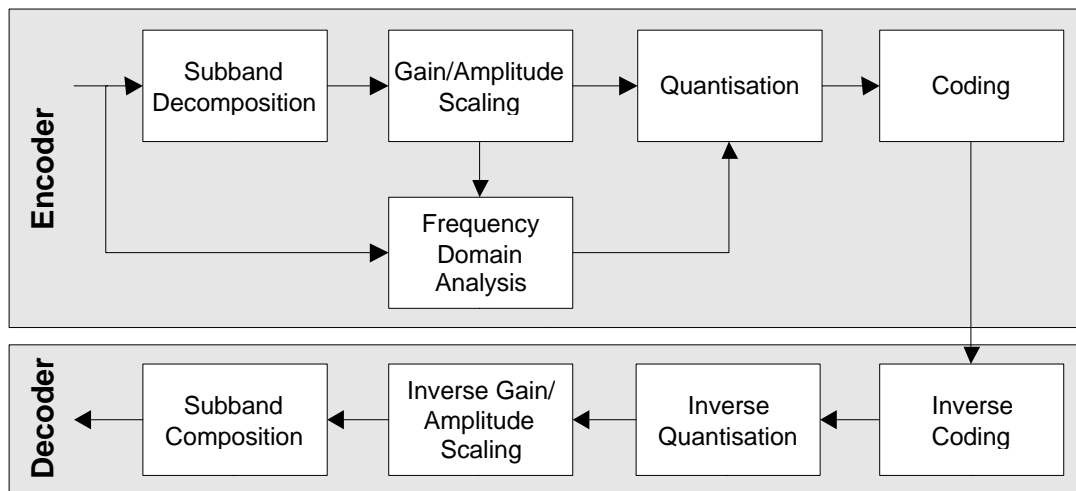
	<b>Bit Rate Range (Kbit/s)</b>	<b>Target Bit Rate (Kbit/s)</b>	<b>Compression Ratio</b>
Layer 1	32-448	192	1:4
Layer 2	32-384	128	1:6 to 1:8
Layer 3 (MP3)	32-320	64	1:10 to 1:12

**Table 4.2: Summary of MPEG audio layers**

Figure 4.12 shows the basic MPEG audio algorithm. Sub-band decomposition separates the incoming audio signal into 32 frequency bands, which are subsequently scaled and quantised. The 32 frequency bands are an approximation to the 24 critical bands of the



human ear. Additional analysis for selection of the quantiser step size in each sub-band is provided by the frequency domain analysis function. The audio is then coded and formatted with additional information for transmission.

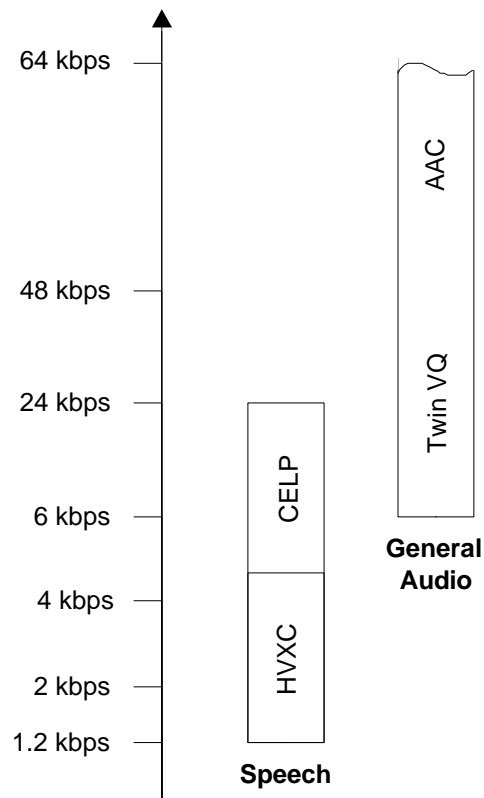


**Figure 4.12: Block diagram of the MPEG audio algorithm**

The decoder unpacks and decodes coefficients and the additional information and performs inverse quantisation. The result is multiplied by the appropriate factor in each band and sub-band composition is applied to recover the original audio signal. For a detailed description of the MPEG algorithm the reader may consult [GBL+1998].

### 4.5.2 The MPEG-4 Audio Algorithm

The MPEG-4 standard was designed for use in a much wider range of applications and therefore uses a more complex structure than MPEG-1 and -2. MPEG-4 uses two groups of algorithms, one for speech coding and the other for general audio coding, as shown in Figure 4.13. This shows the different MPEG-4 encoding algorithms relative to the bandwidth targeted by the individual algorithms. The voice coding is based on algorithms developed for military secure communication. The general audio algorithms are based on work done on MPEG-2.



**Figure 4.13: MPEG-4 audio coding algorithms**

The voice coding algorithms are Harmonic Vector eXcitation Coding (HVXC) and Code Excited Linear Prediction (CELP). HVXC is a parametric vocoder, operating at 2 and 4 kbps, which offers good speech quality at low data rates. Vocoders are a group of speech encoders that digitise speech by explicitly modelling the vocal tract. The CELP algorithm offers encoding from 6 kbps up to 18 kbps.

The general audio coding algorithms are Twin Vector Quantisation (Twin VQ) and Advanced Audio Coder (AAC). AAC is based on the MPEG-2 AAC low complexity audio encoding algorithm. MPEG-4 AAC is fully backward compatible with MPEG-2 AAC. Twin VQ is a general audio coder that is optimised for encoding music at very low bitrates, typically 8 kbps.

The MPEG-4 audio algorithm for natural sound defines the following set of audio coding tools:

- " The error robustness tools can be divided into codec specific error resilience tools and a common error protection tool. The set of error resilience tools offer improved error resilience for AAC by reducing the perceivable deterioration of the signal. The error protection tool or EP tool provides error protection for all MPEG-4 Audio version 2 audio objects;
- " The general MPRG-4 audio algorithm has an algorithmic delay of up to several hundred milliseconds at low bitrates. This delay is undesired in real-time bi-directional communication and therefore MPEG-4 Version 2 specifies a Low-Delay Audio Coder tool which is derived from AAC;
- " MPEG-4 Version 2 provides for fine granularity scalability with the Bit-Sliced Arithmetic Coding (BSAC) tool. This tool is used in combination with the AAC coding tools and provides scalability in steps of 1 kbps per audio channel. One base layer bit stream and many small enhancement layer bit streams are used. The base layer is combined with some or all enhancement layers to obtain a better audio quality;
- " Parametric Audio Coding tools combine very low bit rate coding of general audio signals with the possibility of modifying the playback speed or pitch during decoding without the need for an effects processing unit;
- " The silence compression tool provided CELP silence compression and reduces the average bit rate producing a lower bit rate compression for silence;
- " The Environmental Spatialisation tools enable composition of an audio scene with more natural sound sources and sound environment modelling than is possible in MPEG-4 Version 1. Although the Environmental Spatialisation tools are related to audio, they are part of the BInary Format for Scene description and are referred to as Advanced AudioBIFS; and
- " The MPEG-4 Audio Transport Stream defines a mechanism to transport MPEG-4 Audio streams without using MPEG-4 Systems and is dedicated for audio-only applications.

For synthetic sound, MPEG-4 specifies decoders that can synthesise sound from several kinds of structured inputs. Text input is converted to speech in the Text-To-Speech, or TTS decoder, while music can be generated by a special synthesis language called Structured Audio Orchestra Language (SAOL).

TTS coders bitrates range from 200 bps to 1.2 Kbps and provide commands that can be used to allow synchronisation to associated face animation, international languages for text and international symbols for phonemes.

SAOL is used to define an “orchestra” made up of “instruments”. The instruments are downloaded in the bitstream and not fixed in the terminal which create and process control data. An instrument is a small network of signal processing primitives that might emulate some specific sounds such as those of a natural acoustic instrument [Koe2001]. The orchestra is controlled by “scores” or “scripts” which are downloaded in the bitstream. A score is a time-sequenced set of commands that control various instruments to produce music and/or sound effects. The score is contained in a language called Structured Audio Score Language (SASL).

## 4.6 The MPEG-4 File Format

The final product of the MPEG-4 authoring process is the MP4 file. The design of the file format is based on the QuickTime format from Apple Computer Inc. The MP4 file provides a flexible, extensible format which facilitates interchange, management, editing, and presentation of MPEG-4 media [Koe2001].

The MPEG-4 standard allows the actual media objects to be stored within the MP4 file or referenced externally via an URL. The MP4 file is composed of object-oriented structures called 'atoms'. Each atom can be uniquely identified. Most atoms form part of the movie atom and contain metadata that give information such as synchronisation data and pointers to media objects.

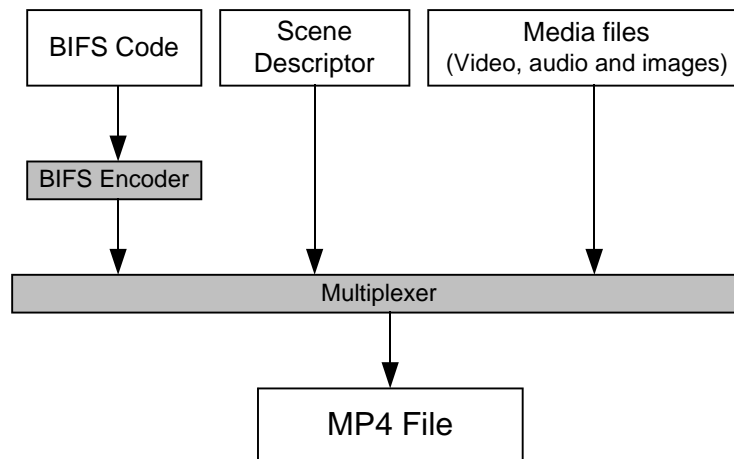
MP4 files may be played from local storage on the host system or may be streamed over a network. The file format is designed to be independent of any particular transport protocol while providing efficient support for streaming in general. The entire file itself is actually never streamed over a transmission medium. To achieve this, the MP4 file uses hint tracks. Hint tracks contain metadata that provide instructions to a server application on how to deliver the media data over a particular transport protocol. A single MP4 file may contain multiple hint tracks, each providing streaming instructions for a specific transmission protocol. Thus a MP4 file facilitates streaming without ever being streamed directly.

## 4.7 Creating an MPEG-4 Scene

This chapter has discussed BIFS code and the encoding of video, audio and images, all of which form the building blocks of a MPEG-4 multimedia application. The process of combining the different building blocks to form a MP4 file will now be discussed.

The final product of a MPEG-4 authoring tool is a MP4 file that consists of object or scene descriptors, BIFS code and the media elements used, as shown in Figure 4.14. There is more than one method to create a MP4 file. In this section, two methods will be discussed. Each method uses its own utility(ies) to create the file. The first method will discuss the procedure followed by the IM1 reference software [IM12002]. The second method will make use of the MP4Tool from ENST [Enst2002].

Both procedures make use of object descriptors, BIFS code and the raw media files as described in Figure 4.14. There are however, small differences between the two procedures. The object descriptors describes the scene, by providing information regarding the scene size, the raw media objects and the synchronisation of objects in the scene. Furthermore, it provides details about how these elements are encoded and the Quality of Service (QoS) of the elements. The BIFS file contains the code that controls the temporal and spatial attributes of the objects. BIFS code also defines the interaction between the different objects and the viewer.



**Figure 4.14: Creating MPEG-4 content**

The procedure using the IM1 reference software will be discussed first. This procedure requires the author to create the following: the BIFS code in a text file, the scene descriptors in a separate text file and the raw media elements in MPEG-4 format. The text file with the BIFS code should have "scene" as an extension. The file containing the descriptors should have an extension of "script".

For the next step BifsEnc, (the BIFS file encoder) and MP4Enc, (the MP4 file multiplexer) are required. Both tools were developed by the IM1 team. The BIFS file is encoded by using the BifsEnc utility and will result in the encoded BIFS code in a .bif file. Once the BIFS file has been encoded, the author can multiplex the MPEG-4 media files together with the .bif and .scene file by using the MP4Enc utility.

The procedure for using the ENST utility is as follows. The author must create a text file that contains the BIFS code and the object descriptors. The BIFS code must be placed first in the file followed by the object descriptors. The MP4Tool utility can then be used to encode the BIFS and multiplex the encoded BIFS and the media files in one operation.

## 4.8 Summary

For developers and engineers building MPEG-4 compliant systems a solid understanding of the MPEG-4 framework is an invaluable tool. The overview of MPEG-4 in Chapter 2 and the introduction to the MPEG-4 framework in this chapter, provides the developer with a useful starting point to developing MPEG-4 compliant systems.

MPEG-4 builds on and extends on many different standards and technologies which are superbly blended into one powerful standard for the representation of multimedia content over almost any data communication medium.

The Binary Format for Scenes or BIFS code provides the author with the means of controlling the playback of a MPEG-4 scene. This is achieved by writing code that defines the behaviour of objects within a scene. BIFS builds on and extends many concepts from the Virtual Reality Modeling Language.

The encoding of video and audio in MPEG-4 is based on methods and algorithms used in MPEG-1 and -2. For video compression, MPEG-4 introduces the use of arbitrarily shaped video segments with much improved compression ratios over its predecessors. The MPEG-4 audio algorithm uses several audio codecs for different audio streams. Different codecs cater for natural sound, synthetic sound and speech. MPEG-4 audio also introduces structured languages for the generation of speech and music.

The BIFS code and media objects created by the authoring process are packaged into a MP4 file. The MP4 file provides a flexible, extensible format which facilitates interchange, management, editing, and presentation of MPEG-4 media. The process of creating an MPEG-4 file consists of multiplexing the BIFS and object descriptors and the media objects encoded in MPEG-4 format.

The next chapter will discuss the specification and design of the 2-D MPEG-4 authoring tool. The chapter will continue with the OVID methodology from the Object phase through

to the final Interactions phase. The objectives of the chapter include providing a class model and draft screen design that can be used as basis for implementation in Chapter 6. The chapter will also discuss the selection of a suitable implementation tool.



## Chapter 5: Specification and Design

### 5.1 Introduction

Early in 2001, when this research project was initiated, limited research had been done on MPEG-4 authoring tools. As stated in the situation of concern in Chapter 1, there was a lack of research on MPEG-4 authoring tools and a very limited understanding of the MPEG-4 standard in general. These formed the primary motivations for starting this research project.

In order to solve the situation of concern, the goals of this project are to provide more information on the standard and its uses, and to develop a prototype 2-D MPEG-4 authoring tool, called SceneBuilder. In the preceding chapters the MPEG-4 standard was introduced and the functionality of the standard was discussed using practical examples. The previous chapters also include a discussion on authoring tools in general and an analysis of authoring tools that provide functionality similar to that of an MPEG-4 authoring tool. The results of the analysis provided the tasks that should be supported by a MPEG-4 authoring system. The creation of the task model completed the first step of the OVID methodology.

This chapter consists of two sections. The first will discuss the specification of the SceneBuilder system. The second will discuss the system design of SceneBuilder. The system specification will outline what the system needs to do without discussing how it will be done. The system design will discuss the internal working of the program providing more information on how the system accomplishes its goals.

The chapter will follow the remaining steps of the OVID methodology and will discuss the creation of an Unified Modeling Language (UML) diagram [BJR1999] that specifies the classes and interface of the system. Thereafter a draft version of the interface will be constructed and the selection of possible implementation tools will be discussed.

## 5.2 System Specification

The specification of the system defines, in detail, the required functionality of SceneBuilder. The functionality that the system must provide was determined by studying the functionality of the MPEG-4 standard, (Chapters 2 and 4) and conducting a task analysis of selected authoring tools (Chapter 3). Another important factor in determining the functionality of the system is the scope of the project as defined in Chapter 1. The size of the MPEG-4 standard will become a serious risk to a MPEG-4 related project, if the scope for that project is not set to achieve realistic goals. The following sections will discuss the OVID methodology and the functionality required for a 2-D MPEG-4 authoring tool.

The OVID methodology uses an user-centred design approach by using the tasks and mental model of the user as a basis for system design. The main goal for any user-friendly system is to support the tasks of the user in such a way that the user can reach his or her goal in a easy and efficient manner. Therefore, the functionality and requirements of the system can be discussed in terms of the tasks of the author.

### 5.2.1 System Functionality

As stated before, the functionality of the SceneBuilder system will be determined by the scope of the project, the task model of a 2-D MPEG-4 authoring tool and the functionality of the MPEG-4 standard itself. The knowledge gained by an in-depth investigation into the MPEG-4 standard and general authoring tools showed that the initial scope of the project as stated in Chapter 1 needed to be changed. The final scope for the SceneBuilder system is as follows:

- " The system will support 2-D scenes and objects;
- " The objects that will be supported are:
  - < Video objects;
  - < Audio objects;

- < Image objects;
  - < Geometry objects;
  - < Text objects;
  - < Timer objects; and
- " The system will allow the MPEG-4 author to add simple interaction to objects.

The task model constructed in Chapter 3, Section 3.2.1 was a normative model and it listed the steps that were thought to be required to complete the given tasks. Once the normative task model was constructed, three authoring systems were evaluated and these evaluations allowed a descriptive task model to be derived. A descriptive task model is one that lists the actual tasks required to achieve a goal. A generalised task model is a combination of a normative and descriptive task model. The most important differences between the generalised and normative task models are that media objects are loaded into an object store before they are placed in a scene, and the addition of a timer object. The generalised task model for a 2-D MPEG-4 authoring tool is listed below:

**Tasks:**

- 1 Create a new scene
- 2 Add video, audio, image, text and geometrical objects to the scene
- 3 Add a timer to the scene
- 4 Add events to objects
- 5 View the MPEG-4 scene in an MPEG-4 player

**Generalised task model:**

- 1 Select New Scene
  - 1.1 Set the background colour
  - 1.2 Set the Scene size in pixels
- 2 Import a Video Object into the object store
  - 2.1 Select a Video Object and insert it into the Scene
  - 2.2 Set the start and stop time of the video
  - 2.3 Set the playback size

- 2.4 Set the loop option to true or false
- 2.5 Set the transparency level
- 2.6 Set the position of the video in pixels
  
- 3 Import an Image Object into the object store
  - 3.1 Select an Image Object and insert it into the Scene
  - 3.2 Set the size of the image
  - 3.3 Set the transparency level
  - 3.4 Set the position of the image in pixels
  
- 4 Import an Audio Object into the object store
  - 4.1 Select a Audio Object and insert it into the Scene
  - 4.2 Set the start and stop time of the audio
  
- 5 Select a Text Object and insert it into the Scene
  - 5.1 Set the text of the object
  - 5.2 Select the font type and size
  - 5.3 Set the colour of the text
  - 5.4 Set the transparency level
  - 5.5 Set the position of the text in pixels
  
- 6 Select the Geometry Shape object and insert it onto the Scene
  - 6.1 Set the colour of the object
  - 6.2 Set filled true or false
  - 6.3 Set the transparency level
  - 6.4 Set the position of the shape in pixels
  
- 7 Drag a timer onto the scene
  - 7.1 Set the start and end time for the timer
  - 7.2 Set the loop option to true or false

- 8 Add an event to an object
  - 8.1 Select the desired object
  - 8.2 Add the desired event to the object
  - 8.3 Select the type of action or the case that will trigger the event
  - 8.4 Select the object(s) that will be affected
  - 8.5 Select the attribute to be modified
  - 8.6 Set the new value of the attribute
  
- 9 Click the run button to view the scene in an MPEG-4 player.

From the task analysis listed above, it becomes clear that SceneBuilder should have the following functional and non-functional requirements:

**Functional Requirements:**

1. The system must allow the author to create a new scene. Once a scene has been created the author may set the background colour and size of the scene.
2. The author may populate a scene with available and supported objects. The system will support video, audio, images, text, timer and geometry objects.
3. The author may edit the properties of objects. Objects may be copied and then pasted in a scene or deleted from a scene. The changes made to objects should be reflected, in real-time, by the on-screen appearance of objects.
4. The author may add events to objects and specify the behaviour of the objects when the events are triggered. The system will include events that are triggered by user interaction with the mouse or by elapsed time. Events may be modified and removed from objects.
5. The author must be able to create a MP4 file and view the content in a MPEG-4 player.
6. The author must be able to change the order of objects in a scene.

**Non-Functional Requirements:**

1. The author may create more than one scene. Each scene may then have its own objects and the viewer may navigate through the scenes.

2. The system must allow an author without programming knowledge to create a MPEG-4 scene. The author only needs working a knowledge of the raw media objects used in a scene; no knowledge of the BIFS programming language is required.
3. The system must be reliable and responsive to the actions of the author.

## 5.3 System Design

With the task model, extant systems analysis and requirements of the system completed, the Objects phase of the OVID methodology can commence. As indicated in Figure 3.5, after the task analysis has been completed, an object model for the system can be designed. The object model is produced by identifying all objects in the system and their properties and interrelationships by making use of the task model.

### 5.3.1 The Objects in SceneBuilder

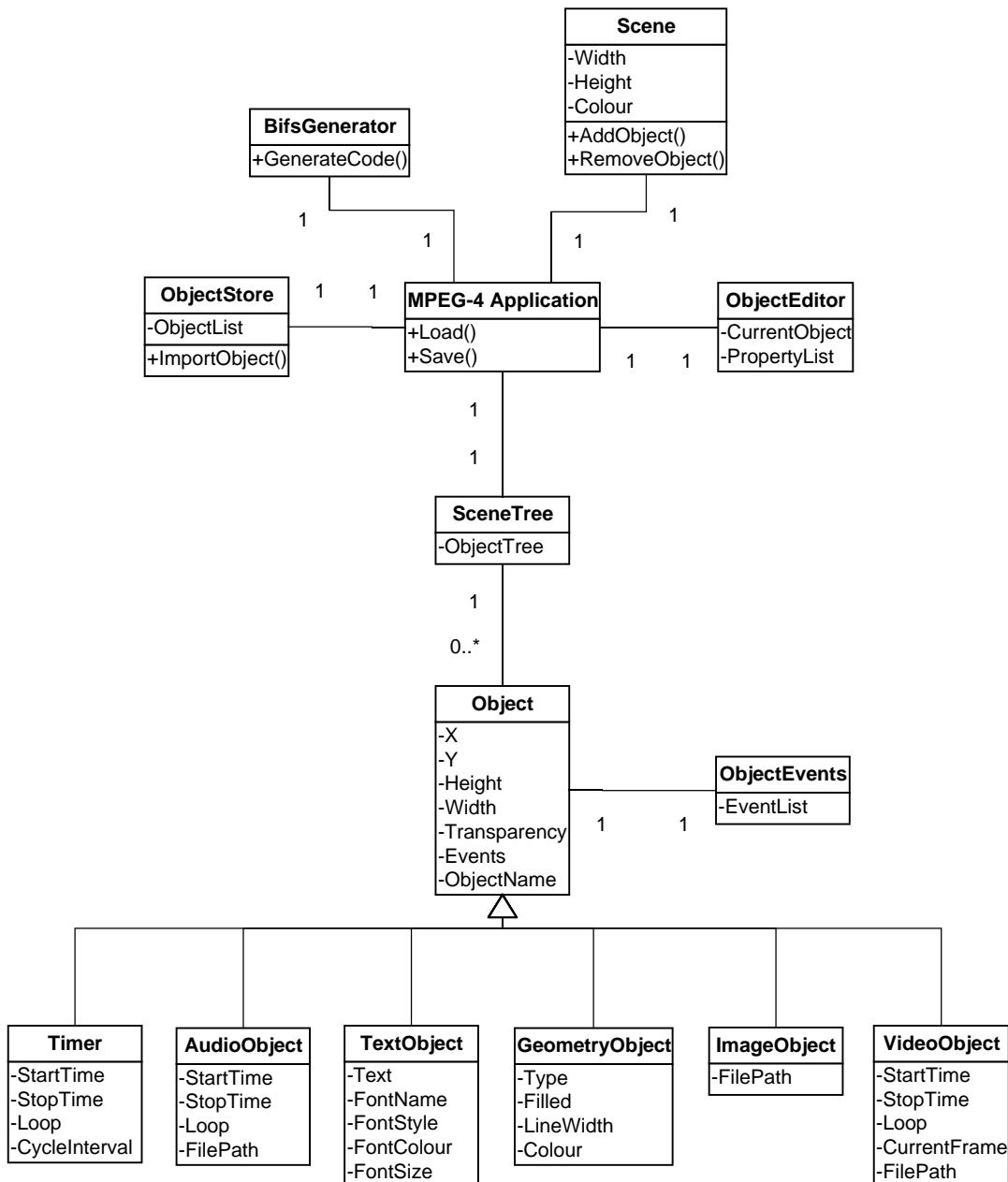
Figure 5.1, shows the object class diagram for the system. The main class for SceneBuilder is the MPEG-4 Application class. The MPEG-4 Application has a CodeGenerator, an ObjectStore, an ObjectEditor, a SceneTree, and a Scene associated to it. Each Object has associated with it an single ObjectEvents class. Each object can be inserted into the SceneTree as an instance of a VideoObject, ImageObject, AudioObject, TextObject, GeometryObject or TimerObject class.

The rationale behind creating a base class for all the objects is that all the objects have some functionality in common. Therefore, a single parent class can be created from which they will inherit the basic functionality they have in common with one another. The Object class provides functionality for representing objects on screen (size, position) and setting the transparency of objects.

SceneBuilder should allow the author to create more than one scene. Each scene can then have a number of objects inserted into it. It should therefore follow that the MPEG-4

Application class should have a one to many relationship with the Scene class. However, the relationship between the two classes is one to one, as depicted by Figure 5.1. The reason for this is that the association between objects and the scene they belong to is already stored in the SceneTree class. The SceneTree class provides the author with a representation of the hierarchical tree in which all objects and scenes are stored. This includes nodes that represent the scene and the objects in each scene. The SceneTree class will therefore store the scenes and all of the objects related to each scene, making multiple instances of the Scene class unnecessary. The nodes of the hierarchical tree constructed by the SceneTree class will contain pointers to an instance of each class represented by the node. The only exception is that the node representing a scene will not contain a pointer to an instance of a Scene class but instead only stores the name of the scene. The order of the scenes and the objects in each scene is then implicitly recorded by the structure of the tree.

Every instance of an Object will have its own events associated with it as depicted by Figure 5.1. Therefore events are stored by object and not in a single instance of the ObjectEvents class. By doing so the implementation process is greatly simplified. This implies that all the events for Objects need not be stored in a general list with unique identifiers associating objects and events. For example, when objects are deleted their events will implicitly be deleted with them without having to remove the events from a general list of all events.



**Figure 5.1: Object class diagram for 2-D MPEG-4 authoring tool**

The ObjectEvents class had to be carefully designed to cater for the needs of an author who might not have any programming skills. In general, programming languages have a static mapping between variables and events. This mapping, usually, is the name of the variable and the source of the event. For example, consider the case of code which affects a variable when executed after an user clicks on a button. If the name of the button or variable changes or if one of the two is deleted, the code will not compile and the compiler will



generate an error. It is then left up to the programmer to solve the problem. In the case of SceneBuilder, the author can select an object, select the type of event and then select the target object that will be affected by the event. Therefore, if a static mapping, such as the names of the objects, exists between the source and target of the event the author will have to keep track of all the events. Once an event has been set, the names of the objects involved may not change and the target object may not be deleted. As the potential author may not have programming skills it cannot be expected that he/she keep track of all the events. For this reason SceneBuilder implements dynamic links between the source object and target object in an event.

Dynamic links between the source and the target object of an event will allow the author to change the names of the objects after events for those objects have been set. The author will also be able to delete the target objects of events and the system will remove the affected events on his/her behalf. The implementation of the dynamic links will be discussed in Chapter 6. The next section will discuss the Views step of the OVID methodology in which views are associated with objects.

### **5.3.2 The Views Associated with Objects**

After identifying the object classes that play a role in the authoring tool, the Views step of the OVID methodology can be started. A view is the representation of an class object in the user interface. The task analysis is used as a guide to selecting views, as it is the view that will allow the user to complete his or her task.

Starting with the most frequent task, a view is created for each object used in that task. The resulting object class diagram with views is shown in Figure 5.2. The most frequent tasks of the author are inserting objects into a scene and then changing their attributes. Therefore views are required of the ObjectStore, the Objects themselves and an ObjectEditor.

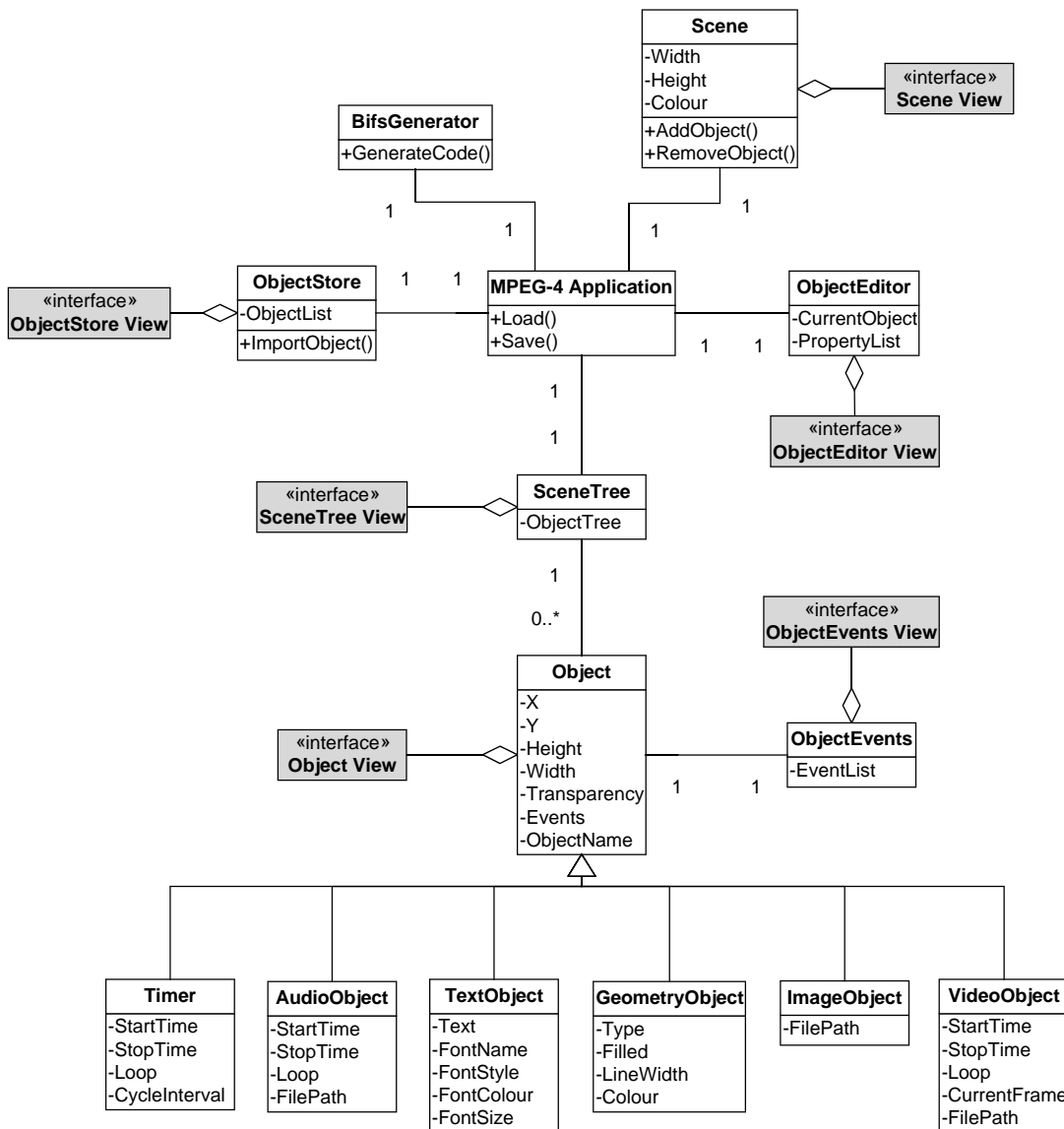


Figure 5.2: Object class diagram with views

The **ObjectStore** can contain zero or many objects and therefore it would provide the user with a list of objects to select from. The **ObjectEditor** will list the attributes of a single object making it possible for the author to edit the attributes of the selected object. Each object will have a different view depending on the type of object and the actual media object it represents. An image object, for instance, will display the actual image associated to the object when inserted into a scene.

Other views include the Scene View, the ObjectsEvents View and the SceneTree View. The SceneView will be a display area onto which the author can drag and insert objects. The SceneView will provide the author with a static preview of the scene. The EventsEditor View will allow the author to add, edit and delete events from objects. The SceneTree View will provide the author with a view of the hierarchical tree in which all objects are stored.

### 5.3.3 The Tasks Associated with Each View

The next step of OVID is to determine the tasks associated with the views of the different objects. Views and tasks are tightly coupled and it is the view that must support the author in accomplishing his or her task. If tasks were grouped together in the task analysis the grouping should be incorporated into the view. It is important that a view should not support too many tasks as it will make the view overly complex [BIM1997].

Table 5.1 lists the tasks of the author that are associated to each view. The ObjectView has been omitted from the list as it does not allow the author to complete a task other than just to view the object. The processes of displaying and editing the events and the attributes of objects will have similar user interfaces. Therefore it might be possible to incorporate the ObjectEditor and ObjectEvents views into one control.

<b>Views</b>	<b>Tasks</b>
Scene View	Drag and drop an object from the ObjectStore Select an object in the scene Select the scene itself Move an object to a desired position
ObjectStore View	Import an object into the ObjectStore Select an object from the ObjectStore Insert an object into the scene by dragging it onto the scene
ObjectEditor View	Select the attribute to be changed Edit the attribute and change it to the desired value
SceneTree View	Select the object in the scene tree

<b>Views</b>	<b>Tasks</b>
ObjectEvents View	Add the desired event to the object Select the response to the event Select the target object that will be effected by the response Select the attribute of the object that will be affected Enter the new value for the attribute

**Table 5.1: Tasks related to the views of the object class diagram**

### 5.3.4 The Interaction Style for SceneBuilder

The objects have been specified and their views have been defined in terms of the tasks they must support. This process was completed without defining the actual user interface of the system. This is useful as it allows the object views and tasks to be defined with a high level of completeness without having to design an interface.

Before the interaction techniques can be defined, it is important to take cognisance of the interaction techniques that will be used by the author. In the design of any system it is critical to take into account the environment, equipment and special needs of the author. For this system the interaction techniques will include a pointing device such as a mouse and the keyboard. The environment of the author will be the general home or office workstation.

Given an enumerated task list such as the generalised task model discussed earlier together with the interaction techniques, a table can be generated to explicitly list the interactions of each view. Interaction techniques may further be defined as user interface controls. These controls should be defined abstractly to allow for some degree of flexibility during the development phase. For example, a drop-down list could be called a list, or a slider could be called a finite range control [BIM1997]. For the purposes of this project, however the explicit interactions for each view will not be created in table form. Section 5.3.5 will discuss the design of a draft interface for SceneBuilder and will provide possible interaction styles for the different views.

### 5.3.5 A Draft Interface Design

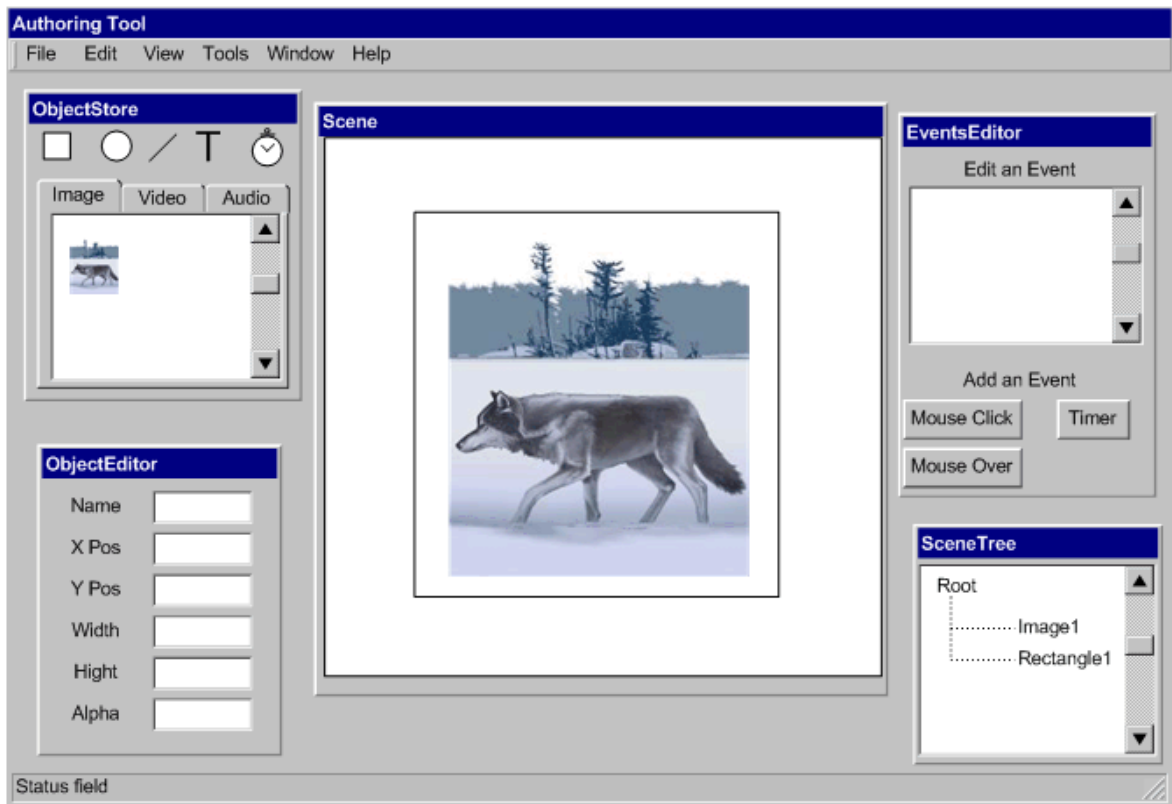
Based on the results of the extant systems analysis and the objects defined by OVID, a draft interface design for SceneBuilder is shown in Figure 5.3. The interface was created by using Microsoft Visio [Mic2002] as a prototyping tool. The interface is a draft version and can be used as a starting point for the implementation phase.

The interface has an edit/preview screen which provides the author with an editing area representing the scene. The author can add objects to the scene and edit their properties. Applying changes in real time will allow the author to view the result in a WYSIWYG (What You See Is What You Get) manner. This view however will not show the results of interaction or behaviour added to objects. In order to accomplish this, the author will have to encode the scene and view the result in a MPEG-4 player.

The ObjectStore window allows the author to add objects to the scene. The window can be divided into two sections. The first will contain objects that are generated by BIFS. These objects are geometrical shapes, text and timer objects. The second set will be external raw media types such as video, audio and images. These objects can be imported onto the tool palette. Objects may be dragged onto and inserted into a scene, more than once if required. Examples of similar tool palettes can be found in Macromedia Director [Mac2002] and Adobe Premiere [Ado2003].

The ObjectEditor window is used to display and edit the properties of the currently selected object. The EventsEditor window will allow the author to add and edit events for objects. These two windows may be incorporated into one as both are used to edit an attribute of an object. A possible way to combine the two is to use tabs similar to those on the ObjectStore window.

A SceneTree is provided in order to gain a structured view of the scene as a tree. This view gives a graphical view of how objects are stored in the scene hierarchy tree and will provide a valuable tool for the selection and ordering of objects in a scene.



**Figure 5.3: Draft interface design for a 2-D MPEG-4 authoring tool**

### 5.3.6 Development Tools

This section will discuss the selection of tools which was used to develop SceneBuilder. Preceding the selection of the tool, a list of requirements for a suitable development tool was compiled.

The first consideration was the target platform that the authoring tool will run on. There are no available MPEG-4 players that play interactive MPEG-4 scenes in an operating system other than Microsoft Windows. Therefore the authoring tool would be designed for the Microsoft Windows family of operating systems. Since platform independence is not a functional requirement, it will not play a role in selecting a development tool.

The execution speed and memory management of the development tool, are the next considerations. The authoring system will be used to manipulate a variety of objects and the visual objects will require the most processing power. These objects will be displayed in the scene where the author may move and resize the objects in real time. These objects could be semitransparent and very large image or video files. Furthermore the tool must be able to handle multiple visual objects in the scene at the same time. Thus the development tool must allow for fast and powerful graphics operations and fast code execution.

During the MPEG-4 scene creation process, the author will create and delete objects, add and remove events for those objects. These objects can occupy a large amount of memory and therefore the selected tool must have efficient memory management capabilities. Other considerations include the ease with which new interface components can be created and the current components can be extended.

Finally the selected tool must support rapid prototyping. With the limited knowledge on MPEG-4 authoring tools it was expected that the development process would be largely experimental and that several versions of the prototype would be created before an acceptable design was found.

The tools that were considered were C++, Java, Delphi and Visual Basic. The development tool that satisfied the requirements as discussed above is Borland Delphi 6. Delphi supports rapid prototyping and allows for the creation and modification of interface components making it a very flexible tool and preferred above C++. Other languages that were considered were Visual Basic and Java but their execution speed and memory management were not adequate for this project.

## **5.4 Summary**

This chapter discussed the system specification and design of the SceneBuilder system. The system specification listed the revised scope of the project, the generalised task model and functional and non-functional requirements for the SceneBuilder system. The scope for

the implementation of SceneBuilder was changed from what was listed in Chapter 1. The extant systems analysis in Chapter 3 showed that a timer object should be added to the list of objects supported by SceneBuilder. The generalised task model was listed and there were two differences to that of the normalised model discussed in Chapter 3. The differences are the inclusion of a timer object and that objects must be imported into the ObjectStore before they are placed in a scene.

The system design section discussed the completion of the remaining steps of the OVID methodology and the selection of an implementation tool. The remaining steps were the definition of Objects, Views, Tasks and Interactions. Defining the objects is the process of identifying the properties and interrelationships of all objects in the system by making use of the task model. The objects and their relationships were then used to create a UML diagram containing all the object classes. Once the objects were defined, additional view classes were added to the UML diagram. A view is the representation of an class object in the user interface that allows the author to complete a task. During the Tasks phase of OVID, all the tasks were listed against the view that supported them. The purpose of this phase is to detect task overload in which a view supports too many tasks. Thereafter the interactions of the author with the system was defined.

The last section of the chapter discussed the selection of an implementation tool. Before the tool was selected, a list of requirements for a suitable development tool was compiled. These included execution speed, memory management and customisability of components. The tool that was selected was Borland Delphi 6.

The next chapter will discuss the implementation of the SceneBuilder system. This chapter will discuss the process of using the specification and design from Chapter 5 to create the SceneBuilder system.



## Chapter 6: Implementation

### 6.1 Introduction

At the start of the research project in 2001, two major risks were identified. The first was gaining an understanding of the MPEG-4 standard and the creation of MPEG-4 content. Gaining an in-depth understanding of such a large standard was a very difficult task. This was complicated by the fact that MPEG-4 was still under development and therefore changing and growing during the two year period of this project. Another complication and one of the reasons for starting this project was the lack of public knowledge and the unwillingness of researchers to share information of MPEG-4 system developers.

The second major risk was implementing a prototype system. At the start of the project very little was known of the BIFS programming language and the process involved in creating interactive MPEG-4 applications. Furthermore, the project was directly dependent on the available MPEG-4 technologies. The aim of the project was to specify and implement a 2-D MPEG-4 authoring tool and not tools for creating primitive MPEG-4 media types or MPEG-4 players. Therefore the available technology would determine the type of media objects used by SceneBuilder as well as the functionality of the MPEG-4 scenes produced by the tool.

Gaining an understanding of the standard formed the bulk of the research work done in the first year of the project. The first successfully created interactive MPEG-4 scene satisfying the requirements of the project, and containing self-made content was completed at the start of the second year of the project. As a result the implementation phase of the project only started in April 2002.

Chapter 5 discussed the specification and design of SceneBuilder. The chapter followed the OVID methodology and produced the objects, views and interactions required for the

implementation phase of the project. An UML diagram was constructed showing the objects, their interrelationships and views. The chapter ended with draft design for the interface for SceneBuilder and the selection of Borland Delphi 6 as implementation tool.

This chapter documents the implementation of the system by following the design specifications from the previous chapter. The chapter will discuss the implementation of the main system classes, the interface and other aspects of the functionality of the system.

## **6.2 Brief Discussion of Class Implementations**

This section will discuss the implementation of the main classes of the system. These classes are: the Object, ObjectStore, ObjectEditor, EventsEditor, SceneTree and CodeGenerator classes. The excluded classes provide very little of the core functionality of the system and will therefore not be discussed.

### **6.2.1 The Object Class**

The Object class forms the base class for all the objects that the author can insert into a scene. The rationale behind creating a base class is two-fold. Firstly all the objects have common requirements. Each object needs to be displayed on screen, has height, width, position and transparency attributes. Therefore all the objects can inherit this functionality from a single parent class.

The second reason is to make use of polymorphism. Polymorphism allows an instance of a certain class to be stored as an instance of any of its parent classes. Therefore if there is a variable of type Object all variables of its child classes can be stored in it. The child classes of Object are the video, image, text, geometry, audio and timer objects. This allows all the before mentioned child classes to be stored and manipulated by using a variable of type Object making the system more flexible.

## 6.2.2 The ObjectStore Class

The ObjectStore class is responsible for importing and displaying the media objects. The author can then select and insert any of these objects into a scene. The ObjectStore class provides the author with two sets of objects as shown in Figure 6.1. The first set consists of objects that can be created using BIFS code, for example, rectangles and timer objects. The second set contains raw media objects that can be imported and stored in a list from which the author can select the object and insert it into the scene.



**Figure 6.1: The ObjectStore**

The ObjectStore can import image, video and audio media types. For images the system will accept any JPG file. For video and audio object types the system is unfortunately not capable of accepting MPEG-4 media types, for reasons explained in the next paragraph. Instead the system accepts AVI files for video and WAV files for audio objects. The MPEG-4 encoded version of the media object must be placed in the same directory as the imported WAV or AVI file. The system will then use the WAV and AVI file during the authoring process and only use the MPEG-4 media during the MP4 file creation process.

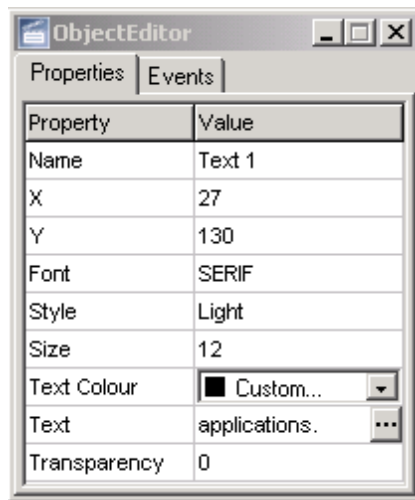
There are several reasons for this approach. At the time of implementation there were two methods available to create a MPEG-4 file. The first was to use MP4Tool to create the file and the Envivio plug-in for playback of the file. The second was to use the IM1 reference

software to create and play back the MP4 file. These two methods, however did not use the same MPEG-4 media format and the system could not cater for all cases. The media types were not the only difference at the time. The players of the two methods provided different levels of functionality and support for the MPEG-4 standard. Therefore when the implementation of the project was initiated, it was decided that the system would support both methods by using a common file type in the authoring process (WAV and AVI). Since then the Envivio plug-in was improved and currently provides all the needed functionality and as such the system only creates MP4 files for that player. Unfortunately due to time constraints, the system could not be changed to work with the MPEG-4 media types.

### **6.2.3 The ObjectEditor Class**

The ObjectEditor class allows the author to view and edit the attributes of objects. The ObjectEditor class queries an object for its attributes and then displays the attributes to the author in a grid format as shown in Figure 6.2.

The two columns in the grid represent the name of the object property and the current value of that property. Depending on the type of property being edited, the author may change the property by typing in a new value, selecting a value from a list or by setting the property by using a dialog box. The author can edit any property by clicking on it. If the property is represented by a text value, the author may type in the new value. If the property can be selected from a list, the author will be presented with a drop-down list. The dialog box is used for properties that are not easily edited by the previous two methods. For example, if the author wishes to change the colour of an object, the author will be presented with a colour selection dialog box. Once a change has been made to a property, the affected object is updated and the author can view the change in real-time.

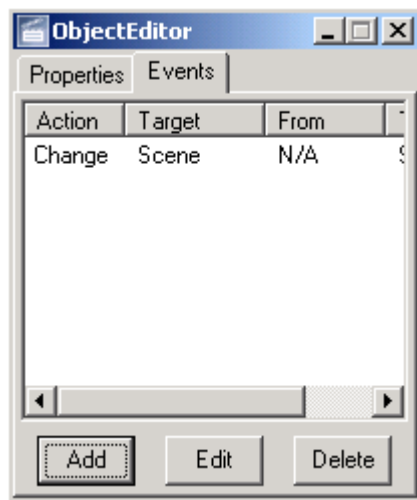


**Figure 6.2: The ObjectEditor**

## 6.2.4 The EventsEditor Class

The EventsEditor class stores and processes the events for each object. To add and edit events, the author must use the "Events" tab on the ObjectEditor as shown in Figure 6.3. Here the author has the opportunity to add and edit events that will respond to a mouse click, the mouse moving over an object or a timer event. The author may add, remove and edit events for the selected object. The EventsEditor also provides a list of the events of the selected object. The list contains the action of the event, the target object, the current value of the attribute to be affected and the value it will change to. The list provided by the EventsEditor summarises the events associated to the selected object, allowing the author to view, select and change any event.

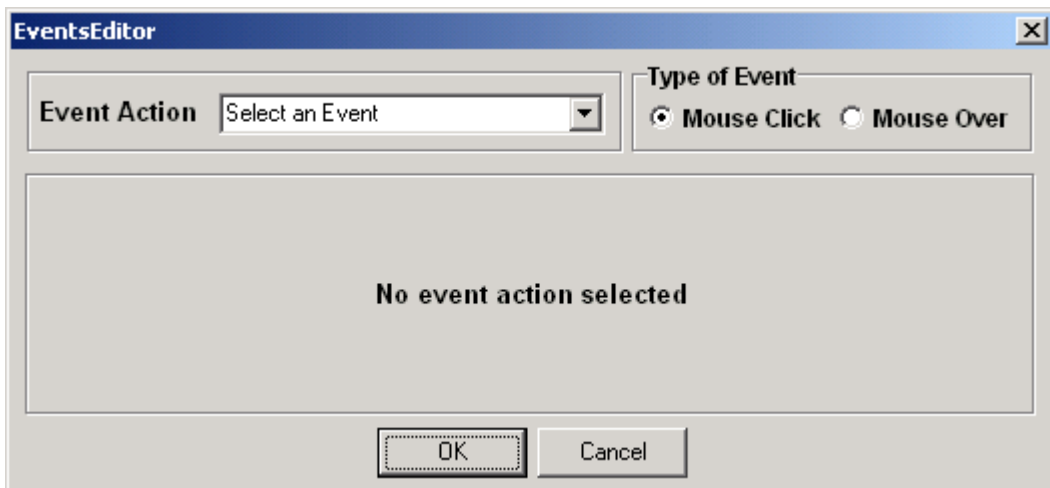
Since the potential author need not have any programming skills, it cannot be expected that he or she has a full understanding of how events function. As stated in Section 5.3.1, traditionally systems require the developer to keep track of events and the objects that are effected by the events. SceneBuilder, however does not require the author to keep track of the events and their target objects, by making use of dynamic links.



**Figure 6.3: Adding and editing Object Events**

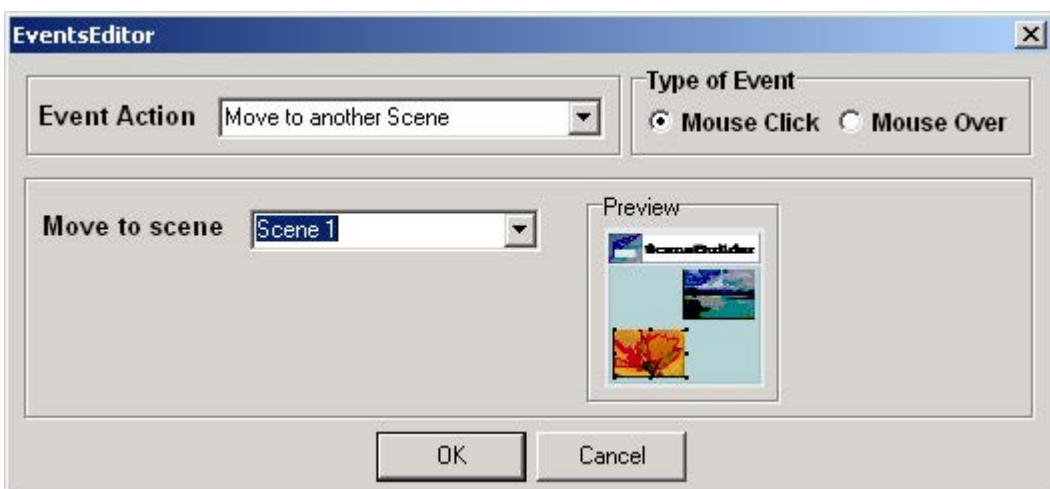
Dynamic links are created by making use of pointers to identify objects and assign events. A pointer is a reference to the memory address where the object has been stored. Therefore the target object of an event can be deleted or its name may be changed subsequent to an event targeting it. The pointer to an object will remain the same while all other attributes of the object are changed. When the properties of an object change, the values in the memory address will change but not the address itself. Therefore the link to the target object will not be lost when the name of the object is changed. When an object is deleted the events of all other objects searched and the events that target the deleted object are removed.

When adding an event to an object, the author is presented with the EventsEditor dialog box as shown in Figure 6.4. The dialog box allows the author to create or edit an event in a simple intuitive way that provides flexibility while protecting the author from making errors or having to write code.



**Figure 6.4: The EventsEditor dialog box**

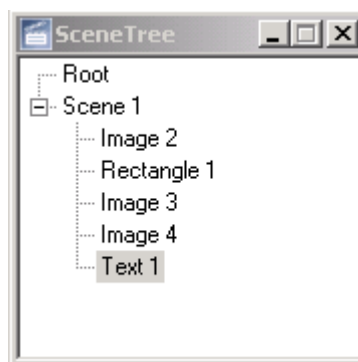
The author can choose from a wide selection of actions including: changing the current scene, changing the position or transparency of an object and controlling the playback of a video. Each action selected by the author has its own display, allowing the author to enter the required information for the event. For example, if the action is to change the current scene, the author is presented with a drop-down list from which he/she can select the desired scene shown in Figure 6.5. The preview area in the figure provides the author with a preview of the scene the author has selected from the drop-down list.



**Figure 6.5: Setting an event to move to another scene**

## 6.2.5 The SceneTree Class

The SceneTree class plays two roles in the system. The first is that it forms a data structure into which all instances of the Object class are placed. The second is that it provides a visual representation of the hierarchical tree in which all objects in the scene is stored shown in Figure 6.6. The scene tree allows the author to select objects by clicking on them in the tree and provides a visual representation of the order of objects in a scene.



**Figure 6.6: The SceneTree**

The SceneTree is not an exact representation of the hierarchical tree used by MPEG-4. The SceneTree provides an abstraction of the actual MPEG-4 hierarchical tree. The MPEG-4 hierarchical tree models the nodes as they are declared in the BIFS code. Therefore, it will contain nodes representing amongst others the Transform2D node and object descriptor nodes. Therefore, an exact representation of the hierarchical tree used by MPEG-4 will expose the author to terms used in BIFS code. As stated in Section 3.4.4, the authoring process should be abstracted from the coding process. The SceneTree therefore only contain nodes that represent the scene structure and the objects that belong to each scene. The hierarchical scene tree used by MPEG-4 will not necessarily have a structure consisting of scenes with objects associated to them. The SceneTree presents the author with a tree that is based on the mental model of the author, making it an usable and author friendly tool.



## 6.2.6 The CodeGenerator Class

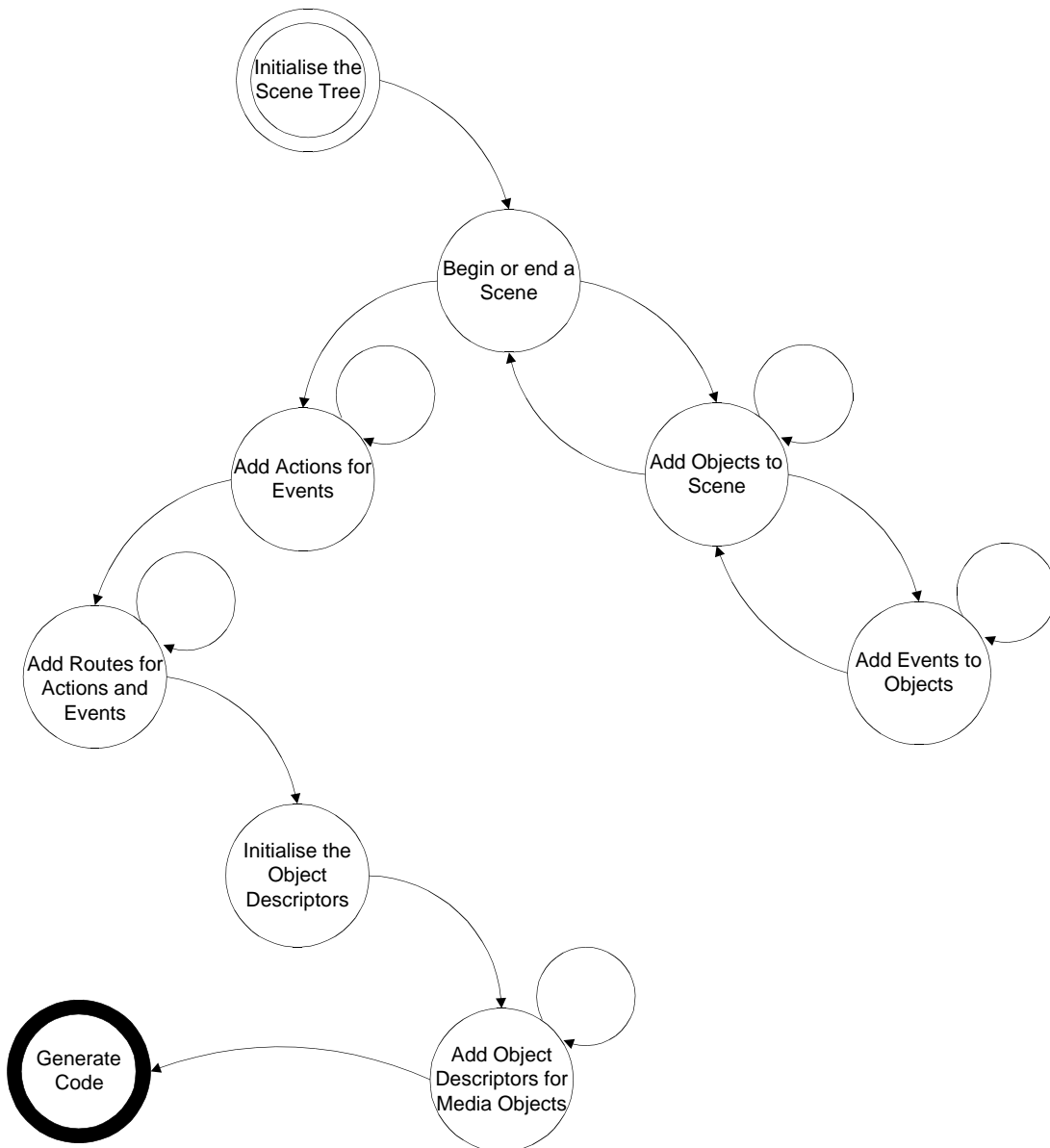
The CodeGenerator class has no display and is not visually represented in the system. The main function of the class is to take the SceneTree as input and produce the BIFS code to represent the scenes. The CodeGenerator algorithm has several states as shown in the state transition diagram of Figure 6.7.

During the initialisation process the code required to initialise the MPEG-4 application is generated and a background object is added to the tree. In the SceneBuilder system the author does not have a choice whether to add a background or not. The system will always have a background object. The author may change only the colour of the background object. The next state involves the addition of new scenes to the hierarchical tree. The process of adding a new scene can be described in four steps. The first step is the declaration of a new scene node. In BIFS code there is no actual scene node. SceneBuilder uses a standard Transform2D node to represent a scene and then adds the objects associated to the scene, each a single Transform2D node, to the children section of the scene node. All nodes that are added to the tree after the creation of a scene node, will be children of the current scene node. The next two steps are to add all the object nodes and their respective event nodes to the scene. Thereafter, the scene node is finalised and a new scene or nodes that describe the actions for events may be added to the tree.

Once all the scenes have been added to the tree, the nodes that describe the actions for events may be added to the tree. Not all events require nodes that describe their actions. It is possible to directly route a value from the object that fired the event to a target object without using a node to describe the action. For example, it is possible to directly route the position of one object, using an appropriate event, to another when the object is moved. Nodes that describe the actions for events are used for more complex actions for instance, when object A is clicked stop the playback of movie B and make object C transparent.

In the state that routed the events, the occurrence of an event will be routed to a node that describes the action or directly the target object itself. A Route takes on the following

form: ROUTE nodeA.event to nodeB.variable. Here the node can be an object in a scene or an action description node. After this state has been completed the object descriptors can be generated.



**Figure 6.7: State transition diagram for the CodeGenerator class**

The object descriptors are nodes that describe the MPEG-4 application and selected objects in the scene. The object descriptors are started by describing the MPEG-4 application by specifying properties including the screen size of the application and the version of BIFS code used. Once an object descriptor node for the application has been added, further nodes

are added to describe nodes that contain media elements. These nodes contain information about the intellectual property rights of the media, the type of media, streaming priority, synchronisation and path to the raw media file, etc. Once all the object descriptor nodes have been generated, the BIFS code is complete and may be compiled and multiplexed with the media files to produce a MP4 file.

### **6.3 Scene Based Editing**

As shown by the task analysis done in Chapter 3, it became clear that the author would want to create MPEG-4 applications that consist of multiple scenes. Therefore the concept of scene based editing was built into the interface of the system to support the mental model of the author. The author may add or remove scenes and then add objects to the scenes. When the MPEG-4 application runs it will by default show the first scene. The movement to another scene will depend on the events specified by the author. For instance, the author can allow the viewer to progress to another scene by interacting with an object or on a time dependent event. The author can create a single scene or a multiple scene application.

From the outset of the project the goal of the system was to allow an author with little or no programming skills to develop MPEG-4 applications. Therefore the interface of the authoring tool was completely abstracted from the coding process. The authoring tool is therefore more than just a visual interface to the coding process. It hides several of the coding procedures and creates several new concepts by combining and manipulating the functionality offered by the code. This differentiates the SceneBuilder system from any other MPEG-4 authoring system as it is more than a programmer's aid. This system allows any author to effectively write a program without writing a single line of code.

### **6.4 Adding Authoring Templates to SceneBuilder**

Once the implementation was completed and all the requirements of the system were satisfied, it was decided that to add authoring templates or components to the SceneBuilder

system. The templates are not part of the requirements or scope of the project. The addition of templates however, will increase the productivity of the author and the usability of SceneBuilder as a whole.

The authoring templates are semi-complete scenes similar to document or design templates used by Microsoft Word and PowerPoint respectively. Authoring components will allow the author to insert interface components, such as buttons and video playback controls. Before the addition of templates, the process required to create a button entailed the author having to create an object representing the button. In order to do so the author had to create the standard view of the button consisting of an image or geometry object with a text object over it. In addition to a standard view the author had to create views to represent the state of the button when the viewer clicks on it or positions the mouse cursor over it. The addition of templates would allow the author to reuse previously created components therefore saving a lot of time.

Adding templates and components to MPEG-4 authoring tools is not a new idea. The system developed by ENST named MPEG-Pro, discussed in Chapter 7, also makes use of templates [BCB+2000]. Their system makes use of a combination of Java code, the development tool for the application, and BIFS code. This allows the developer to create powerful templates and components. A drawback to their system and that is that the author, unless he or she knows Java code, can not create a template. Therefore the templates and components used in SceneBuilder makes use of BIFS code alone. Any author, using the system, can create templates. To make the templates and components more usable the author will have to add a short description of the template or component when it is created. The description will contain information specifying the goal of the template or component and how it should be used.

The templates and components will be displayed in a list on the ObjectStore under a Templates tab. As with other objects, the author can import the templates and components. The system will also import all templates and components located in the templates directory in the root of the system's install directory. Once a template or component has been imported into the ObejctStore the author can insert it into any scene. The templates

and components will be displayed in the same way as a video file in the ObjectStore. All templates and components will have a default icon with the file name next to it. This method, if the templates and components do not have descriptive names, can cause confusion in selecting a desired template or component. Unfortunately due to time constraints it was not possible to implement a more visual way of identifying templates and components.

## 6.5 Implementation of the Interface

The design of the interface as discussed in the previous chapter was implemented as shown in Figure 6.8. The interface remained similar to the proposed draft design. The ObjectEditor is a combination of the ObjectEditor and EventsEditor, allowing the author to edit the properties of the objects as well as their events. The properties of the objects and their events are on separate tab panes.

All the functionality as specified in the previous chapter was implemented as well as the standard functionality associated with such a system. This includes the ability to save, load, copy, paste, delete, move an object to the front or back of the scene, etc. Apart from the mentioned additions in functionality the system also has additional functionality that supports the authoring process. The author may select multiple objects and move, copy, paste or delete them as a group. Furthermore, the system provides the author with a grid as a guide to align objects during the authoring process. The author has the choice of using a grid or not. If the grid is used the author may set the size of the grid and enable the snap to grid option. This option causes objects to automatically align themselves to the grid, making the alignment of objects a simple task for the author.

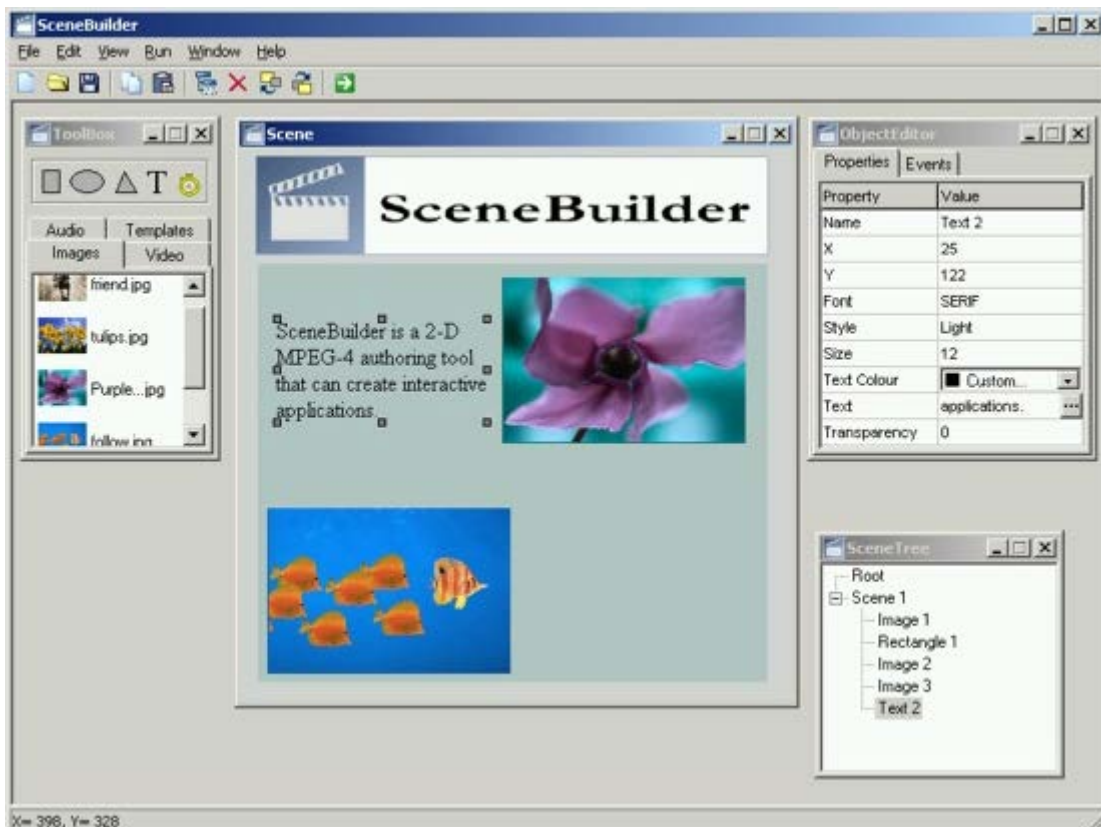


Figure 6.8: Interface of SceneBuilder

## 6.6 Problems Encountered

Section 1.5 gave a brief discussion of the risks associated with the project. These risks almost caused the project to change direction and goals by the end of 2001. The goal of the project was to design and implement a prototype 2-D MPEG-4 authoring tool. The goals did not include the creation of tools that encoded existing media objects to MPEG-4 format and that played an interactive MPEG-4 scene. Finding and creating MPEG-4 media content proved to be a significant challenge. A MPEG-4 media encoder application named mpegable X4 was only acquired in the latter stages of the project [mpe2003].

Gaining an understanding of the BIFS programming language provided another considerable challenge. There was no detailed documentation available on BIFS programming. The only sources of information available, came from studying VRML and from decompiling existing MPEG-4 applications and studying the BIFS code.

It took a substantial amount of time before the first interactive MPEG-4 scene was successfully created. However, there was one major problem with the playback of the scenes. The playback of the video and audio media content was tightly coupled with the playback of the entire scene. This caused the playback of scene to restart and stop when the media objects were restarted or stopped. For example, when a video was stopped the interactive scene would stop as well and become inactive. Similarly, if the video restarted, the scene would restart as well. It took several months to solve this problem and it had a negative impact on the progress of the implementation of SceneBuilder.

The assumption was made that the problem was with the Envivo plug-in [Duf2002]. The problem was in fact with the synchronisation identifiers of the scene and the objects. The video objects, unless otherwise specified, were synchronised with the scene by default. To resolve the problem, the video objects had to be assigned an individual synchronisation identifier by making use of the `OCR_ES_ID` property in the object descriptors.

Although the risks associated with the project posed challenging obstacles, SceneBuilder was implemented according to its specification as set out in Chapter 5. The level of success of the implementation of SceneBuilder will be determined in Chapter 7.

## 6.7 Summary

This chapter discussed the implementation of the SceneBuilder system. The discussion started with a brief description of the implementation of selected classes identified in Chapter 5. The `ObjectEditor` and `EventsEditor` views defined in Chapter 5 were combined into a single view, allowing the author to edit the events and properties of an object in the same window. It was shown that by making use of pointers, dynamic links could be created between the source and target objects of events. The tree displayed by the `SceneTree` was discussed and it was shown that it supports the mental model of the author rather than modelling the nodes in the BIFS code.

The latter half of the chapter showed how SceneBuilder supports scene-based editing and discussed the addition of templates and components to the functionality of the system. The chapter ended with a discussion of the interface of SceneBuilder. The next chapter will discuss the testing and evaluation of SceneBuilder. The testing and evaluation process consists of two phases. The first phase consists of implementing a MPEG-4 application that will make use of all the functionality offered by the system. Thereafter the performance of the system will be measured against its requirements as listed in Chapter 5. The second phase of the evaluation consists of observing potential MPEG-4 content authors using the system.



## Chapter 7: Testing and Evaluation

### 7.1 Introduction

In Chapter 3, the OVID methodology was selected with the intention of creating a user-friendly authoring system. The OVID methodology makes use of an user-centred design (UCD) approach, which supports the creation of user-friendly software that matches the mental model of the user. UCD describes not only the techniques used to create usable software but also the philosophy that places the requirements of the user in the centre of the software design process [Rub1994].

The aim of this chapter is to ascertain whether the SceneBuilder system satisfies its requirements and whether the system supports the tasks of the author. The testing and evaluation will consist of two phases: the first will take on the form of a case study and the second will involve observing potential MPEG-4 content authors using the system. Once the evaluation process has been completed, the results will be discussed.

The latter part of the chapter will discuss MPEG-4 authoring systems developed by other research institutions during the lifetime of this project. Finally the SceneBuilder system will be compared to these other systems.

### 7.2 Methods Used for Testing and Evaluation

The testing and evaluation of the SceneBuilder system had two goals. The first was to compare the functionality of the system to its requirements and the second was to test the usability of the system. Usability as defined by ISO 9241, entitled *Ergonomic requirements for office work with visual display terminals* [ISO1997] is as follows:

"*Usability* is the effectiveness, efficiency and satisfaction with which specified users achieve specified goals in particular environments; where *Effectiveness* is the accuracy and

completeness specified users achieve specified goals in particular environments; *Efficiency* is the resources expended in relation to accuracy and completeness of the goals achieved; and *Satisfaction* is the comfort and acceptability of the work system to its users and other people affected by its use".

Usability testing can be conducted in two ways. The first is an analytical test, which is performed without the participation of users and takes on the form of walkthrough methods. An example of a walkthrough method is a heuristic evaluation. Heuristic evaluations make use of a list of design heuristics or principles which is used to identify design problems. The second evaluation method is an empirical method, which involves real users. This method will involve observing a sample of content authors using SceneBuilder.

The testing and evaluation of SceneBuilder will consist of a case study and user evaluation. The case study will involve the creation of a MPEG-4 application. The application to be created will test all the functionality provided by the system. The case study will also include a comparison of the functionality of the system with its requirements.

The user evaluation will take on the form of an empirical observation test. The content author will use the system and follow a set of given tasks while he or she is observed. The content author will also be requested to talk out aloud, raising his or her opinion, while completing the given tasks. The results of the evaluation process will then be summarised and the conclusions will be discussed.

### **7.2.1 Case Study**

In order to test and evaluate the SceneBuilder system, it was decided to conduct a case study. This case study involved creating a MPEG-4 application that would be similar to an existing application. The rationale behind this decision was that a real-world scenario would provide a better test than a hypothetical test case. With the ICC World Cup being

hosted in South Africa, it was decided that the case study would be to create an enhanced version of the official world cup web site ([www.cricketworldcup.com](http://www.cricketworldcup.com)). Figure 7.1 provides a screenshot of the home page of the web site.



**Figure 7.1: The home page of the ICC Cricket World Cup web site**

The web page in Figure 7.1 consists of images, rectangles and text. The user can click on any of the links on the left-hand side to get more information on topics such as the mission statement or fixtures and venues of the world cup. The internal pages of the site have a different design as depicted by Figure 7.2. Figure 7.2 provides a screenshot of the page displayed to the user when he or she clicks on the "Fixtures and Venues" link.

The page displaying information on the fixtures and venues has a different design from the home page and it makes use of images and text. The SceneBuilder system will be able to create a MPEG-4 application similar to the web site. However, the system is not able to

create a scene that allows the viewer to use a scrollbar to scroll an object that is larger than the scene. For example, the web page displayed in Figure 7.2 has a list of the fixtures and venues that is larger than the page resulting in the need for a scrollbar.



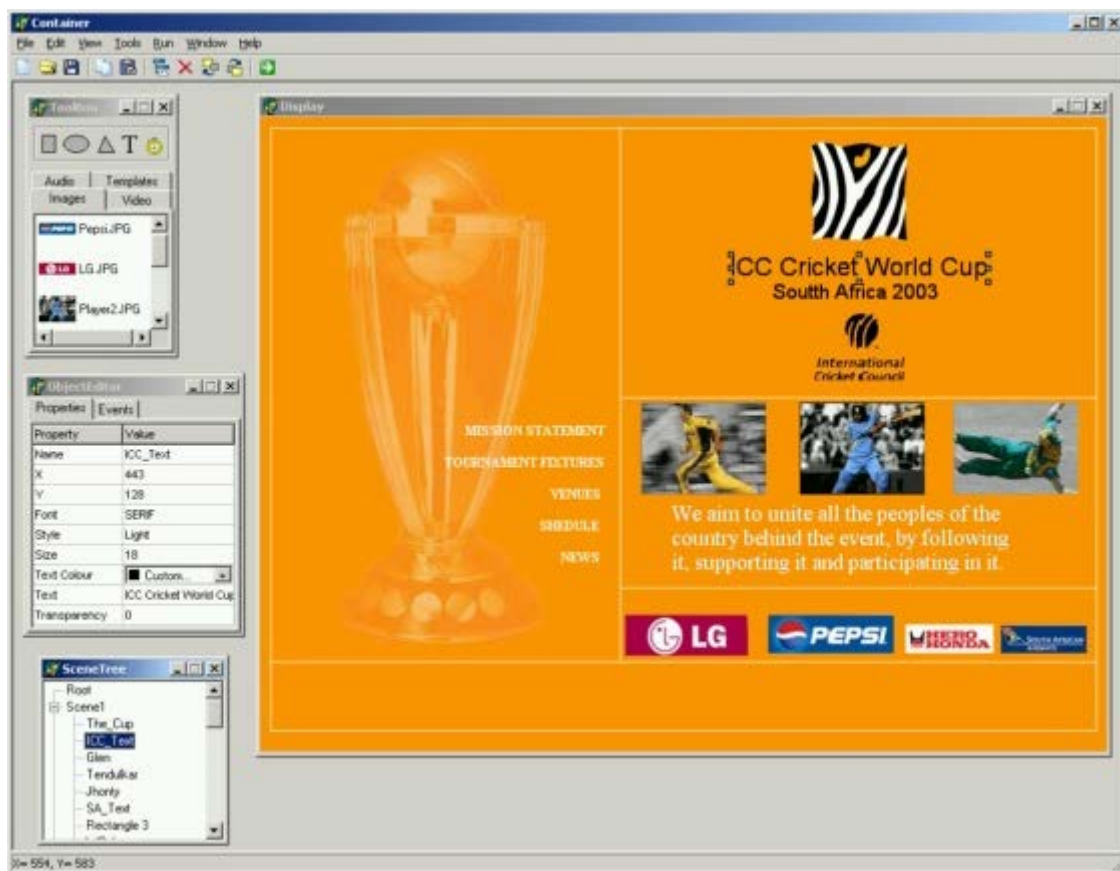
The screenshot shows the ICC Cricket World Cup 2003 website. The header includes the ICC Cricket World Cup logo, the International Cricket Council logo, and logos for sponsors LG, PEPSI, HERO HONDA, and SOUTH AFRICAN AIRWAYS. The navigation menu includes HOME, HOW THE TOURNAMENT WORKS, FIXTURES & VENUES, FAQ, TICKETING, SECURITY, NEWS, and SOUTH AFRICA A-2. The main content area is titled 'FIXTURES & VENUES' and displays a table for the 'ICC CRICKET WORLD CUP 2003 : PRELIMINARY ROUND'. The table lists matches from September 8th to September 15th, including the opening ceremony and various matches between teams from South Africa, Sri Lanka, Zimbabwe, Australia, Bangladesh, Canada, Kenya, India, West Indies, New Zealand, England, and South Africa. The venues listed are Newlands, Cape Town; Goodyear Park, Bloemfontein; Harare Sports Club, Harare; The Wanderers, Johannesburg; Kingsmead, Durban; North West Stadium, Potchefstroom; Boland Park, Paarl; St. George's Park, Port Elizabeth; Harare Sports Club, Harare; Pietermaritzburg Oval, Pietermaritzburg; and Newlands, Cape Town.

DATE	TYPE	Match	Bye	TEAM	No.	vs	No.	TEAM	POOL	VENUE
Sat. 08				OPENING CEREMONY UNDER LIGHTS						Newlands, Cape Town
Sun. 09	D/N	1		South Africa	B-1	vs	B-3	West Indies	B	Newlands, Cape Town
Mon. 10	Day	2		Sri Lanka	B-2	vs	B-4	New Zealand	B	Goodyear Park, Bloemfontein
Mon. 10	Day	3		Zimbabwe	A-5	vs	A-6	Namibia	A	Harare Sports Club, Harare
Tues. 11	Day	4		Australia	A-1	vs	A-3	Pakistan	A	The Wanderers, Johannesburg
Tues. 11	D/N	5		Bangladesh	B-6	vs	B-7	Canada	B	Kingsmead, Durban
Wed. 12	Day	6		South Africa	B-1	vs	B-5	Kenya	B	North West Stadium, Potchefstroom
Wed. 12	Day	7		India	A-4	vs	A-7	Holland	A	Boland Park, Paarl
Thur. 13	Day	8		West Indies	B-3	vs	B-4	New Zealand	B	St. George's Park, Port Elizabeth
Thur. 13	Day	9		Zimbabwe	A-5	vs	A-2	England	A	Harare Sports Club, Harare
Fri. 14	Day	10		Sri Lanka	B-2	vs	B-6	Bangladesh	B	Pietermaritzburg Oval, Pietermaritzburg
Sat. 15	D/N	11		Kenya	B-6	vs	B-7	Canada	B	Newlands, Cape Town

Figure 7.2: The venues of fixtures of the ICC Cricket World Cup

SceneBuilder was used to implement a scaled down version of the web page with enhanced features including video and audio content. It was necessary to enhance the MPEG-4 version of the web page as it only tested a part of the functionality of SceneBuilder. Therefore, it was decided to include video and audio objects. Figure 7.3 depicts SceneBuilder during the construction of the application. The scenes created were very similar to the individual web pages of the original site. In most cases the scenes consisted of image, text and geometry objects.

The first step in creating the MPEG-4 application was to acquire the required media objects. As stated in Section 3.2 it is not the task of the authoring tool to create the media objects. This was accomplished by downloading the images from the web page. The images were then converted to JPG format and imported into the authoring system. The images, text and geometry objects were then added to the scene to create a replica of the web page. During this phase the grid and snap-to-grid functionality of SceneBuilder proved to be very useful in positioning and aligning the objects.



**Figure 7.3: Creating the application with SceneBuilder**

Once the first scene was created, the objects in the scene could be copied and pasted in a new scene making the creation of further scenes a trivial task. After all the scenes were constructed, events were added to the different objects. To allow the viewer to navigate through the scenes in the application, events were added to text objects that represent the links of the web page. Figure 7.4 depicts the MPEG-4 content created by SceneBuilder being played in a QuickTime player making use of the Envivio plug-in.



**Figure 7.4: MPEG-4 Cricket World Cup application played by a MPEG-4 player**

During the creation of the MPEG-4 application, several bugs in the SceneBuilder system were identified and corrected. In most cases the problems that were experienced were only cosmetic and were quite simple to rectify. There is however, one remaining problem with the system. This problem stems from the fact that the MPEG-4 players only support a small set of fonts. The font families, discussed in Section 4.2.2, that are supported are "SERIF" (the default) for a serif font such as Times Roman; "SANS" for a sans-serif font such as Sans Serif; and "TYPEWRITER" for a fixed-pitch font such as Courier. These fonts in turn have a small discrepancy with the fonts used in SceneBuilder. The net result is that the fonts displayed in SceneBuilder, during the scene design phase, in some cases differ from the fonts displayed by the Envivio plug-in.

After the case study was constructed, the functionality provided by the SceneBuilder system was compared to the functionality required from a MPEG-4 authoring system as discussed in Section 5.2.1. From the case study it was clear that the SceneBuilder system met all the functionality required and provided additional functionality to enhance the productivity and overall authoring experience of the author.

## 7.2.2 User Evaluation

There are several empirical usability tests that could be used to evaluate SceneBuilder [Jor1998]. The method that was selected to evaluate SceneBuilder is known as the *think aloud protocol*. This method involves a test participant speaking about what they are doing and thinking, when using the interface. The participant could be given a fixed set of tasks to perform or be allowed to freely explore the system.

The reason for selecting this method is that SceneBuilder is a prototype system and thus not fully implemented. The think aloud protocol is well suited for testing partially completed projects. Furthermore, this method does not need to be conducted in a laboratory environment and requires between three to five test subjects or participants [Nie1998]. This method however, does have its advantages and disadvantages [Jor1998].

The advantages of the think aloud protocol are:

1. From the verbalisations of the participants, it is possible to find out not only what problems they have but also why these problems arise.
2. It is possible to gather performance data, such as task success and number of errors made.
3. The protocol allows for the efficient gathering of a large volume of data from a few participants.

The disadvantages of the think aloud protocol are:

1. Since participants have to explain what they are doing they may feel they must provide motivations for their action, which may cause them to interact with the system differently from what they normally would.

2. There could be an interference between talking out aloud and completing the task at hand.
3. The investigator could, by prompting the participant, cause an error or prevent an error from occurring.

The main goal of this test was to establish whether the system follows the mental model of the user and if the system is easy to use. The test focuses on the interaction between the system and the author. The test did not focus on cosmetic usability issues, for example, the names of interface components. However, many cosmetic usability problems were identified. Five participants were selected for the test. The participants were expert users with no knowledge of MPEG-4, but with some experience of working with authoring tools. The time for the evaluations ranged from 20 to 45 minutes and were conducted in an office environment.

The test consisted of giving the participant a brief introduction to MPEG-4 and the interface of SceneBuilder. The participants were then shown examples of MPEG-4 applications created with SceneBuilder. They were allowed to freely explore the interface of SceneBuilder before they were given a task to accomplish. The tasks that had to be performed by the participants were not handed to them as a structured task list, which listed each task step-by-step, but instead they were presented with a scenario. In this scenario the participant views a MPEG-4 application that demonstrates the functionality of the MPEG-4 standard and downloads a prototype MPEG-4 authoring tool from the Internet. The prototype authoring tool was then used to create an interactive application similar to the application that was viewed by the participant. Each of the participants were observed and the following usability problems were identified:

- " The ObjectEditor displays information of each event, listing the type, the target, the affected variable of the event and what the value in the variable will change to. The ObjectEditor however, does not display whether the event will occur when the viewer clicks on an object or positions the mouse over an object;



- " When editing an event the current values of the property to be changed are displayed and not the previous values that were entered for the event. For example, a object is 0% transparent and an event was set to make it 50% transparent. When that event was subsequently edited it had a value 0% and not 50%;
- " When selecting the object to be affected by the current event, the author must select the target object from a drop-down list. The drop-down list contains all objects that have been inserted into a scene. The participants requested to have an option to display all objects or only the objects in the current scene;
- " When editing or deleting events from the event list, the participants expected that the first event will always be selected when the list is viewed. For example, when the participant wanted to delete an event from the list and it was the only event listed, he or she would press the delete event button without selecting the event. Since the event is not automatically selected, the system does nothing when the delete event button is pressed;
- " There are no buttons or interface components on the ObjectStore that allows the author to import objects. The author must right-click the ObjectStore to import objects or make use of the main menu. The participants requested that a import button be added to the ObjectStore;
- " The participants requested more information on what can be entered in the fields of the ObjectEditor. For example, the label of the transparency field should include some indication that the required input value should be between 0 and 100;
- " To create a text object, the author must drag it from the ObjectStore onto the scene. Once the object is in the scene the author may change the text property in the ObjectEditor to set the text displayed by the text object. This is the only way to change the text displayed by the text object. Several of the test participants double-clicked the text object expecting to be presented with a text edit dialog box; and
- " The SceneTree displays a tree consisting of objects, scenes and a root node. If the author selects an object and then clicks on one of the scene nodes in the tree the object remains selected in the scene. The object should be de-selected as the scene is now the selected node in the tree. If the user attempts to add an event to the object, the system does not respond since the scene node is selected and an event cannot be added to a scene, yet the object appears to be selected.

The user evaluation uncovered a number of bugs in the functionality of SceneBuilder, but the system proved to be stable. These bugs were only in the user interface and were corrected. The feedback from the user evaluation was not only negative, there were several positive comments made. The participants described SceneBuilder as an user friendly and intuitive system that allows for creation of interactive multimedia scenes in a short amount of time. Other comments included that it took a short period of time to learn how to use the system and once the author was familiar with the interface, he or she would be able to quickly and easily create interactive multimedia scenes. A further observation was that the participants rarely interacted with the SceneTree. Further research is required to ascertain why this is the case and whether the author will make more use of it as he or she becomes more accustomed with the system. The SceneTree however, is an essential interface component as it allows the author to select items that may not be visible and to identify the visual order of objects in a scene. For example, the SceneTree indicates if one object is in front of another. The next section will discuss the conclusions of the testing and evaluation of SceneBuilder.

### **7.3 Conclusions of Testing and Evaluation**

The case study showed that SceneBuilder satisfies its functional and non-functional requirements. This test showed that SceneBuilder can create large and complex interactive MPEG-4 applications that contain several objects and events. The case study provided the first thorough test of SceneBuilder and it revealed several bugs in the system. All the bugs found in the case study were corrected before the user evaluation took place.

The user evaluation focused on testing the interaction between the author and SceneBuilder. From the problems experienced by the participants it is clear that there are very few discrepancies between the way SceneBuilder works and the way the participant thinks it should. Most of the issues raised by the user evaluation are concerned with the functionality of the system and not the interaction. Therefore it can be said that SceneBuilder has successfully satisfied the requirements as set out in this dissertation.

## 7.4 Related Research Done on MPEG-4 Authoring Tools

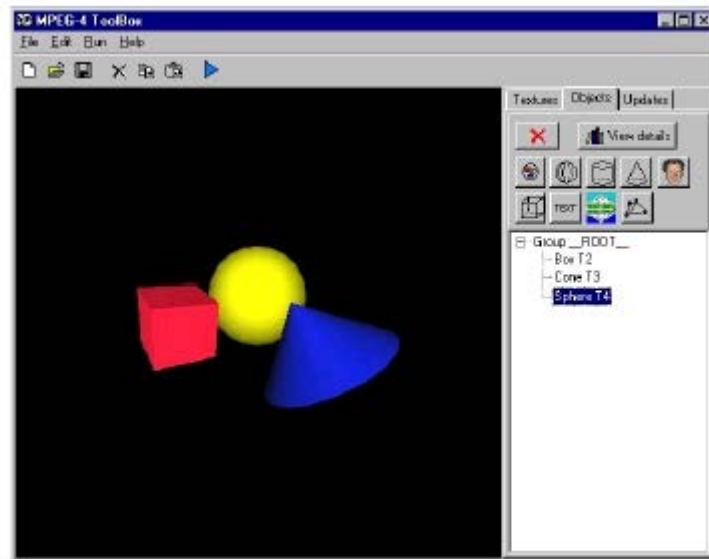
At the start of this research project in January 2001, there was only one MPEG-4 authoring tool available on the Internet. Since then several projects focussing on MPEG-4 authoring have been initiated and completed. This section will highlight the research work done on MPEG-4 authoring tools during the lifetime of this project.

It is important to note that there are several utilities that claim to be MPEG-4 authoring tools. The definition of an authoring tool as discussed earlier is: An application that allows the author to manipulate media objects in space and time and define the behaviour of selected objects. Systems that convert video files from one format to MPEG-4 or that multiplex and encode BIFS files with their respective media objects are not considered as authoring tools.

### 7.4.1 The GSRT Project

The GSRT (General Secretariat for Research and Technology) project was a collaborative effort by the Informatics and Telematics Institute and the Information Processing Laboratory of the University of Thessaloniki, both in Greece [DKR+2000].

The project produced a 3-D MPEG-4 authoring tool, named MPEG-4 ToolBox, that allows the author to create interactive three-dimensional MPEG-4 scenes. The system provided the author with an edit/preview area implemented in OpenGL. Once the author completed the construction of the scene, the tool would generate the BIFS code. The BIFS code is then encoded and multiplexed into one self-contained MP4 file by using the IM1 reference software. The resulting MP4 could then be played back with the IM1 3-D player from the IM1 reference software. Figure 7.5 shows the main screen of MPEG-4 ToolBox.

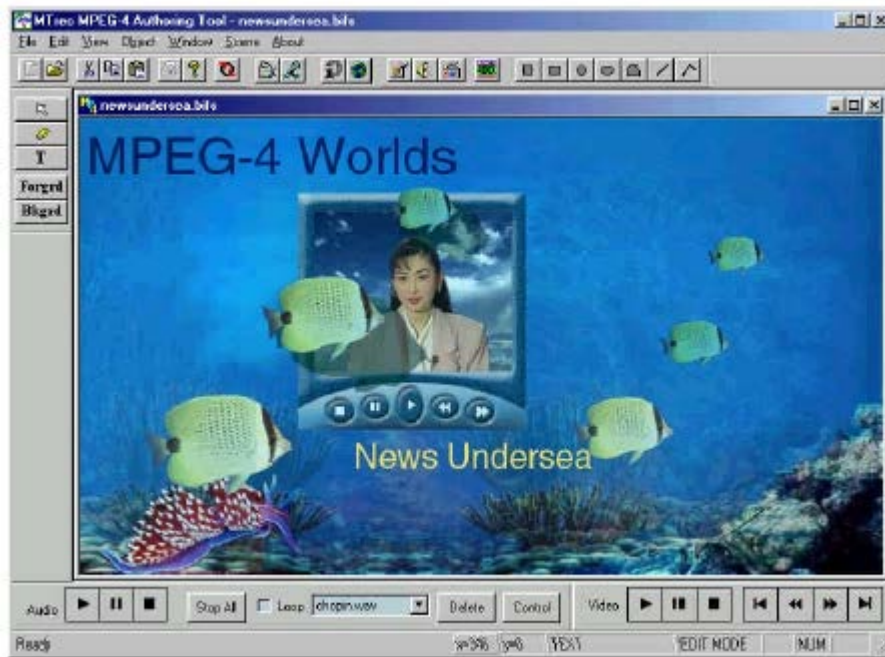


**Figure 7.5: The Main Window of the MPEG-4 Toolbox**

### 7.4.2 MTrec MPEG-4 Toolkit

The MTrec MPEG-4 Toolkit provides an environment for generating MPEG-4 based multimedia sequences. Taking advantage of the object-based structure of MPEG-4, the authoring tool provides a wide range of functionalities to a novice author. The multimedia objects supported by the system are audio and video clips or images, text, graphics, and animation, as shown in Figure 7.6.

The Toolkit also allows the author to specify the behaviour of objects by adding mouse- or time-triggered events. Simple animation is also possible allowing the author to define a path for an object to follow.



**Figure 7.6: The Main Window of the MTREC Toolkit**

The MPEG-4 Toolkit is [MTr2002] composed of three components: MPEG-4 Scene Editor/Player, Video Encoder and Decoder, and MPEG-4 System. Each component is a separate stand-alone sub-system and is described as follows:

- “ The MPEG-4 Scene Editor/Player module allows the author to create and edit a MPEG-4 scene by manipulating audio, video, graphics and text objects in space and time. The system then generates the BIFS code that will reconstruct the scene and the behaviour of the objects in a MPEG-4 player. The system also encodes the media objects to MPEG-4 format and packages the encoded objects and BIFS code into an MP4 file. The Editor/Player allows the author to save and load BIFS as well as MP4 compressed files. The player supports interaction and can render scenes with multiple synthetic or natural audio, video clips combined with images, text and graphics;
- “ The Video Encoder/Decoder forms a major component of the MPEG-4 Toolkit. This encoder/decoder software component is integrated with the Authoring Tool to compress and decompress all the video objects present in the scene into MPEG-4 compliant bitstreams; and

- " The System Layer Module is responsible for generating/interpreting scene composition information and the compression/decompression of media objects. The scene composition information is supplemented with synchronisation information and is passed to the delivery layer that multiplexes it into coded binary streams that can be transmitted or stored in an MP4 file.

### 7.4.3 The ENST MPEG-Pro Authoring System

The MPEG-Pro authoring system, developed by ENST and implemented in Java, is shown in Figure 7.7. The system allows an author to create MPEG-4 content from the end-user interface construction phase to the creation of a cross-platform MP4 file [BCD+2000].

The authoring system goes further than just supporting MPEG-4 functionality. It also provides the author with tools for temporal constraints and templates for improved productivity.

MPEG-4 provides the means to implement spatial dependancies between objects. For example, when one object is moved by an event the other spatially dependent object/s will move as well. For temporal dependancies however, there is no support provided by MPEG-4. During the authoring process the temporal values for temporally dependent objects must be set by hand. If one object changes later in the authoring process all other dependent objects must be updated by hand.

For example, imagine two objects named A and B entering a scene, with the second object entering five seconds after the first. If Object A appears in the scene when the scene time is equal to 10 then Object B will appear when the time is 15. If the author decides to change the time when object A enters the scene then the author must also change the time for Object B accordingly. MPEG-Pro addresses this problem by introducing the concept of temporal constraints management.

To express and compute temporal dependancies between objects the system uses the following inequality [BCD+2000]:

$$\text{Time}(o) - \text{Time}(o_i) \leq R \hat{O}$$

where:  $\text{Time}(o)$  is the start or end time of object  $o$ .

$\hat{O}$  is a time difference value.

The inequality above is used by the underlying system while the author will use a set of high-level expressions such as, Object A Co-begin Object B. The temporal constraints management system uses a constraints resolution algorithm that manages all temporal constraints. This system will then manage all the specified constraints on behalf of the author. For example, let the author specify that Object B must start an action at the same time that Object A does. Therefore the system is given a constraint that the start time of an action for Object A minus the start time of Object B must be less than or equal to zero. If the author then changes the start time of Object A the system will then update the start time of Object B to satisfy the constraint. The temporal constraints information will only be available during the authoring process as MPEG-4 does not provide for temporal constraints in BIFS code.

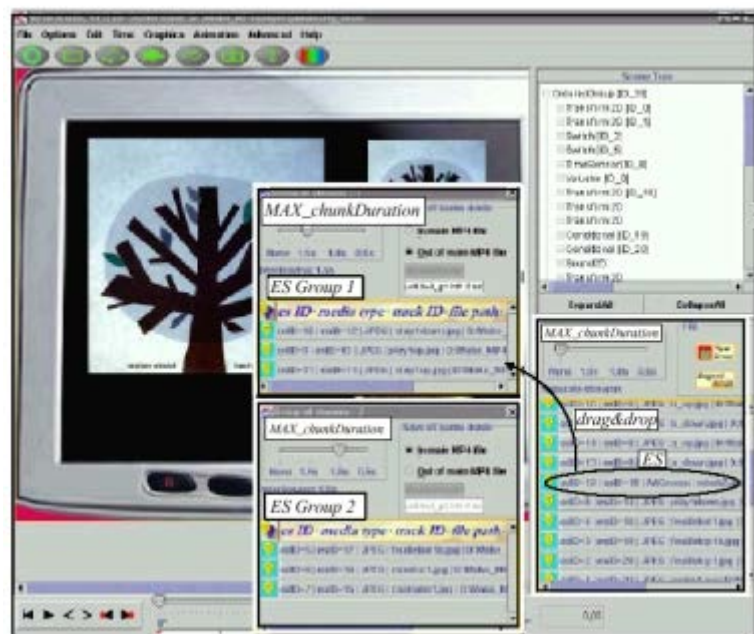


Figure 7.7: The MPEG-Pro Authoring System

The templates supported by MPEG-Pro are reusable BIFS and Java authoring elements. The templates are of two types. The first is a complete MPEG-4 scene which may then be adapted to suit the author's needs. The second is a building block, a button for example.

Further research done by ENST on the project resulted in adding delivery support to the tool [BCB+2000]. The addition of delivery support to the functionality of the tool allows the author to manually group and interleave the elementary streams of similar QoS. This allows the author to have more control over the generation of object descriptors.

#### **7.4.4 ETRI**

The Electronics and Telecommunications Research Institute based in Korea was established in 1976, and is a non-profit government-funded research organization that has been at the forefront of technological excellence for more than 25 years.

The authoring system developed by ETRI allows the author to construct 2-D MPEG-4 scenes and supports all the standard objects, and events [KKC+2001]. The system produces an MP4 file that can be played back using the IM1 2D player. This system is also the only system developed by a research group that makes use of a score. A score allows the author to see the temporal attributes of multiple objects at the same time. Scores are used in Macromedia Director and most video editing suites.

### **7.5 Comparison of MPEG-4 Authoring Systems**

This section will discuss the differences and similarities between SceneBuilder and the other MPEG-4 authoring systems described in Section 7.4. It is important to note that the only authoring system that the author of this dissertation was able to obtain a copy of was MPEG-4 ToolBox. Efforts to acquire copies of the other systems were in vain. Therefore the comparison between SceneBuilder and these other authoring systems will be based on what could be learned from the literature on these tools.



The discussion will for each of the major authoring components of a MPEG-4 authoring tool and discuss the similarities and differences between the systems as follows:

- " All the authoring tools have a display area that allows the author to create scenes in a WYSIWYG style. None of the tools, however allow the author to preview the final interactive scene in the authoring system itself. In all cases the authoring system must first create a MP4 file and play it back with a MPEG-4 player;
- " The tool palettes for the other tools are all similar to each other but differ from SceneBuilder. SceneBuilder provides the author with a tool palette similar to the other systems but also provides the author with an object store. The author can load media objects from file into the objects store from where they can be inserted into the scene;
- " Two of the other authoring systems implement controls that allow the author to manipulate the temporal properties of the scene. MPEG-Pro makes use of a simple timeline while the ETRI system makes use of a score similar to that found in Macromedia Director. In the case of the ETRI system the score plays an important role as the main function of the system is to broadcast MPEG-4 multimedia scenes. In SceneBuilder, a timeline would offer very little extra functionality to the system;
- " The object and event editing processes across all the systems are very similar;
- " Only two of the other systems implement a scene tree. In both cases the trees are a representation of the MPEG-4 hierarchical tree created by the BIFS code. SceneBuilder makes use a tree that abstracts from an MPEG-4 hierarchical tree to fit the mental model of the user;
- " From what can be learnt from the literature, MPEG-Pro is the only system, other than SceneBuilder, which makes use of templates;
- " SceneBuilder is the only system that follows a scene-based authoring approach. None of the other systems have the support for multiple scenes built into the authoring process. Most of the other authoring systems only support the creation of a single scene; and
- " SceneBuilder is the only system that abstracts the authoring process from the coding process and therefore it is the only system that will be user-friendly for authors that have no programming skills.

## 7.6 Summary

This chapter discussed the testing and evaluation of the SceneBuilder system. The aim of the chapter was to ascertain whether SceneBuilder satisfies its requirements and whether the system supports the tasks of the author. To this end, the testing and evaluation of SceneBuilder consisted of two phases. The first, took on the form of a case study and the second involved observing potential MPEG-4 content authors using the system.

In the case study, SceneBuilder was used to create an enhanced version of the official 2003 cricket world cup web site. The MPEG-4 application, created by SceneBuilder, closely resembled the original web site, which was enhanced by adding interactive videos.

The user evaluation took on the form of a talk aloud protocol. The participant was given a brief overview of the MPEG-4 standard and then introduced to the SceneBuilder system. The participant did not have to follow a rigid task list, but was shown a simple MPEG-4 application which they used as a guide. A total of five expert users, with previous experience of working with authoring tools, were tested. The observations made during the evaluation was documented and discussed. From the testing and evaluation it was shown that SceneBuilder successfully satisfies its requirements and that the system does support the tasks of the user.

The latter part of the chapter highlighted the research work done on MPEG-4 authoring tools during the lifetime of this project. Thereafter SceneBuilder was compared to the other systems and the findings were discussed. It was shown that SceneBuilder is unique in its approach to follow the mental model of the user and in abstracting the authoring from the coding process.

The next chapter will summarise the research done in this dissertation, recommend directions for further research and discuss the final conclusions of this dissertation.

## **Chapter 8: Conclusions and Future Research**

### **8.1 Introduction**

The goal of this dissertation was the specification and design of a prototype 2-D MPEG-4 authoring tool. A further goal of the dissertation was the implementation and evaluation of the prototype system. The primary motivations behind initiating this research project was the apparent lack of knowledge of the MPEG-4 standard and the limited implementations of MPEG-4 authoring tools. As far as the author of this dissertation could ascertain, there were, at the start of the project in January 2001, no 2-D MPEG-4 authoring systems available in the public domain. In order for MPEG-4 to reach its full potential, it will require authoring tools and content players that satisfy the needs of its users. This dissertation and further research in this area could facilitate the acceptance of MPEG-4 by content authors, software developers and end-users.

This chapter concludes the dissertation by reviewing the research conducted in this dissertation and discussing areas for future research.

### **8.2 Summary of Research and Implementation**

MPEG-4 is an ISO/IEC standard developed by the Moving Pictures Experts Group (MPEG). Work on the standard started with the main objective of streaming media content, video and audio, over low bandwidth networks, especially at bit rates below 64 kbps. During the development of the MPEG-4 standard, it became clear that there was a need for a standard that enabled the streaming of multimedia applications over low bandwidth networks and Internet connections. MPEG-4 has since evolved into a framework allowing for the creation of multimedia applications with scalable content that may be transmitted over most networks.

Chapter 2 provided an overview of the MPEG-4 standard and discussed some of the research that was relevant to this project. MPEG-4 was introduced as a standard that offered a rich set of features to content authors, service providers and users [Koe2001]. For content authors, MPEG-4 enables the production of reusable and flexible content, the protection of content owner rights and control over the playback process and the behaviour of the content. For network service providers, MPEG-4 offers transparent information, which can be interpreted and translated into the appropriate native signalling messages of each network with the help of relevant standards bodies. For end users, MPEG-4 brings higher levels of interaction with content. It also brings multimedia to new networks, including those employing relatively low bit rates, for example, wireless and mobile networks.

One of the most important features of MPEG-4 is its ability to store media elements as objects. An object is a single media element that may be natural (recorded video) or synthetic (rendered video) and two- or three-dimensional. These objects are completely independent of each other and are stored in a hierarchical tree with the media objects forming the leaves. Furthermore MPEG-4 provides a coding language that controls the placing of objects in scenes and adding interaction or methods to describe the behaviour of the objects. The code that allows the author to combine and manipulate objects into a scene is called The BInary FFormat for Scen<sub>e</sub>s (BIFS). BIFS describes the spatial and temporal arrangements of the objects in a scene and builds on and extends several concepts from the Virtual Reality Modeling Language (VRML).

The large scope of MPEG-4 makes it impossible for developers to implement systems that conform to the entire standard. In order to allow effective implementations of MPEG-4, the standard was broken up into subsets, called Profiles, that allow complexity scalability in MPEG-4 applications. Through the use of Profiles, a developer can create an MPEG-4 compliant application by implementing a subset of the standard. A complete list of MPEG-4 Profiles are provided in Appendix A.

Research on MPEG-4 started in 1993 and still continues as several extensions of the standard, currently researched by MPEG, has not yet been completed. Chapter 2 discussed

the research done on MPEG-4 by MPEG and other research groups, highlighting topics that are of interest to the project. Not only does the large scope of MPEG-4 make it a powerful and more marketable standard, it also brings it into competition with many other technologies as discussed in Section 2.8.

Having established a basic understanding of the MPEG-4 standard, the next step was to conduct a study of authoring tools as discussed in Chapter 3. Authoring tools are programs that allow an author to manipulate different media elements spatially and temporally, and add interactive behaviour to them [CC2000]. Studies have revealed several paradigms into which authoring tools can be classified [Sig1999] as discussed in Section 3.2. An authoring paradigm is the metaphor or methodology that the tool uses to help an author accomplish his\her task.

An extant systems analysis of a 3D MPEG-4 authoring tool, a VRML authoring tool and Macromedia Flash was conducted. This analysis formed part of the OVID methodology which was selected for the project. OVID (Objects, Views, and Interaction Design) is a formal methodology for designing user friendly programs based on the analysis of user's goals and tasks [BIM1997]. The aim of the analysis was to take cognisance of work already done in the area as well as to identify the tasks of the user.

The results of the analysis showed that MPEG-4 and VRML authoring tools have similar characteristics [VCC2002]. It was found that the hierarchical object paradigm was probably the best suited paradigm for a MPEG-4 authoring tool. It was also found that the initial scope of the project as stated in Chapter 1, did not specify the inclusion of an object that could be proven to be very useful in the authoring process. Therefore the scope of the project was extended to include a timer object. Two important conclusions were made from the analysis. Firstly, the authoring process should be abstracted from, and should not model, the coding process. Secondly the process of adding events to objects must be kept as simple and intuitive as possible as it will play a direct role in the success of the system.

Before commencing with the design of the authoring tool, it was necessary to conduct an in-depth study of the MPEG-4 framework as discussed in Chapter 4. The main aim of the

study was to gain an understanding of the processes that are required to produce an MPEG-4 application. A secondary objective of the chapter was to provide an introduction to the MPEG-4 framework for researchers who will do further work on this project. To this end the Binary Format for Scenes was discussed, in detail, by making use of examples as well as the process of creating an MPEG-4 file. The MPEG-4 file, named an MP4 file, is created by multiplexing the compiled BIFS code and object descriptors together with the media objects. Included in the discussion of the MPEG-4 framework was an overview of the MPEG-4 encoding algorithms for the encoding of video, audio and images.

Having established the theoretical basis of the project, the next step was the specification and design of a 2-D MPEG-4 authoring system named SceneBuilder. A final version of the scope of the project was discussed in Section 5.2 as well as the generalised task model. The functional and non-functional requirements for the SceneBuilder system were discussed in Section 5.2.1. The next step was the completion of the remaining steps of the OVID methodology which included:

- " The identification of the properties and interrelationships of all objects in the system by making use of the task model. The objects and their relationships were then used to create a UML diagram containing all the object classes;
- " Creating views for objects. A view is the representation of an class object in the user interface that allows the author to complete a task;
- " Listing the tasks of the author against the views in order to detect a task overload. A task overload occurs when a view supports too many tasks; and
- " Finally the interactions of the author with the system was defined.

As discussed in Section 5.3.6 a list of requirements for a suitable development tool was compiled before selecting an appropriate tool. These included execution speed, memory management and customisability of components. The tool that was selected was Borland Delphi 6.

Chapter 6 documents the implementation of SceneBuilder and the implementation of selected classes identified in Chapter 5 were discussed. It was shown in Section 6.2.4 that

by making use of pointers, dynamic links could be created between the source and target objects of events. The hierarchical tree of objects displayed by SceneBuilder was discussed Section 6.2.5, and it was shown that it supports the mental model of the author rather than modelling the nodes in the BIFS code. It was shown that SceneBuilder further supports the mental model of the author by supporting scene-based editing as discussed in Section 6.3. Additional functionality, not in the scope of the project, was added to SceneBuilder as templates and components were added to the functionality of the system. Finally the interface of SceneBuilder was presented and discussed in Section 6.5.

The testing and evaluation of SceneBuilder was discussed in Chapter 7. The chapter provided a brief overview of usability and usability testing. The testing and evaluation of SceneBuilder was conducted in two phases. This first took on the form of a case study and the second of a user evaluation. The results of the testing and evaluation was discussed in Section 7.3 and it was concluded that SceneBuilder had successfully satisfied its requirements and that the system supports the tasks of the author. The latter part of the chapter highlighted the research work done on MPEG-4 authoring tools during the lifetime of this project. SceneBuilder was compared to the other systems and it was shown that SceneBuilder is unique in its approach to follow the mental model of the user and in abstracting the authoring from the coding process.

### **8.3 Future Research**

This dissertation has created many possibilities for future research projects. These can be separated into projects that extend SceneBuilder and other projects that study the use of the authoring tool and MPEG-4 standard.

Projects that extend the functionality of SceneBuilder can include the following:

- " Adding a view that allows the author to edit and change the object descriptors of objects. SceneBuilder does not currently allow the author to set the properties of the object descriptors;

- " Adding the functionality to perform object segmentation in videos and images and to use the segmented objects in scenes. Thus the system should allow the author to extract an object from a video or image which can then be imported and used in a scene. Unfortunately the Envivio plug-in does not provide support for segmented objects presently;
- " Incorporate the functionality of the MPEG-7 standard by allowing the author to add descriptions to objects;
- " Extend SceneBuilder to 3-D. Many classes from the SceneBuilder system can be used as is or with small changes in a 3-D authoring system; and
- " Provide the author with a view that displays the BIFS code as the scene is constructed. The author must then be allowed to edit the code and view the changes in the scene as the code is changed.

Projects that study SceneBuilder may include the following:

- " Investigate how end users view and interact with interactive videos. Useful outcomes from this investigation could be guidelines for scene and interaction design. How do users identify an object? How do users read text and interact with a scene when watching a video?; and
- " Conducting a study comparing different MPEG-4 authoring tools and performing a formal usability study on the different systems.

## 8.4 Conclusions

MPEG-4 has the potential to satisfy the increasing demand for innovative multimedia content on low bandwidth networks, including the Internet and mobile networks, as well as the need expressed by users to interact with multimedia content. However, there is an apparent lack of knowledge of the standard and limited implementations of MPEG-4 applications, particularly in the area of MPEG-4 authoring.

The dissertation provides an overview of the MPEG-4 standard and discusses in detail the functionality provided by MPEG-4. The dissertation specified the design of a 2-D MPEG-4



authoring tool that follows the mental model of the author and abstracted the authoring process from the coding process. A prototype version of such a tool was successfully implemented, evaluated and compared to other MPEG-4 authoring tools. It was shown that SceneBuilder is unique in its approach to follow the mental model of the user and in abstracting the authoring from the coding process. Therefore the dissertation makes an important contribution to the understanding of the MPEG-4 standard, its functionality and the design of a 2-D MPEG-4 Authoring tool.

---

## Bibliography

- [Ado2003] Adobe Systems Inc., *Adobe Premiere*,  
<http://www.adobe.com/products/premiere/main.html>, Retrieved: 2003.
- [All2002] ALLAIRE, J., *Macromedia Flash MX - A Next-Generation Rich Client*,  
Macromedia, Inc, 2002.
- [All2003] Allen Communication, <http://www.allencomm.com/>,  
<http://www.allencomm.com/>, Retrieved: 2003.
- [App2003] Apple Computer, Inc., *QuickTime*, <http://www.apple.com/quicktime/>,  
Retrieved: 2003.
- [Aus2002] Austerberry, D., *The Technology of Video & Audio Streaming*, Focal Press,  
2002.
- [BCB+2000] Bouilhaguet, F., Concolato, C., Boughoufalah, S., and Dufourd, J., *Adding  
Delivery Support to MPEG-Pro, An Authoring System for MPEG-4*, ENST,  
2000.
- [BCD+2000] Boughoufalah, S., Dufourd, J., and Bouilhaguet, F., *MPEG-Pro, an  
Authoring System for MPEG-4*, IEEE, 2000.
- [BCL2000] Battista, S., Casalino, F., and Lande, C., *MPEG-4: A MULTIMEDIA  
STANDARD for the Third Millennium, Part 2*, IEEE, 2000.
- [BH2001] Bormans, J. and Hill, K., *MPEG-21 Overview*, ISO/IEC JTC1/CS29/WG11,  
2001.
- [BIM1997] Berry, R., Isensee, S., Mullaly, J., and Roberts, D., *OVID: An Overview*,  
IBM Corporation, 1997.
- [BJR1999] Booch, G., Rumbaugh, J., and Jacobson, I., *The Unified Modelling  
Language User Guide*, Addison Wesley, 1999.
- [CBM1997] Carey, R., Bell, B., and Marrin, C., *Virtual Reality Modeling Language,  
(VRML97)*, ISO/IEC 14772-1:1997, 1997.
- [CC2000] Chapman, N. and Chapman, J., *Digital Multimedia*, Jhon Wiley and sons,  
Ltd, 2000.
- [CCI1990] CCITT SG XV, *Recommendation H.261 - Video Codec for Audiovisual  
Services at px64 kbit/s*, International Telecommunication Union, 1990.

- 
- [Chi1996] Chiariglione, L., *Short MPEG-1 Description*, ISO/IEC JTC1/SC29/WG11, 1996.
- [Chi2000] Chiariglione, L., *Short MPEG-2 Description*, ISO/IEC JTC1/SC29/WG11, 2000.
- [Chi2001] Chiariglione, L., *The MPEG-4 Standard*, <http://leonardo.telecomitalialab.com/paper/china98/china98.html>, Retrieved: 2001.
- [CPO2000] Capin, T., Petajan, E., and Ostermann, J., *Efficient Modeling of Virtual Humans in MPEG-4*, IEEE, 2000.
- [DKR+2000] Daras, P., Kompatsiaris, I., Raptis, T., and Strintzis, M., *MPEG-4 Authoring Tool for the Composition of 3D Audiovisual Scenes*, Informatics and Telematics Institute, 2000.
- [Duf2002] Dufourd, J., Personal Communication, Higher National School of Telecommunications, Paris, France, 2002.
- [Ebr2001] Ebrahimi, T., *MPEG-4 Video Verification Model*, Swiss Federal Institute of Technology, 2001.
- [Enst2002] Dufourd, J., *MPEG-4 @ ENST*, <http://www.comelec.enst.fr/~dufourd/mpeg-4/>, Retrieved: 2002.
- [Env2002] Envivio, *EnvivioTV plug-in*, [www.envivio.com](http://www.envivio.com), Retrieved: 2002.
- [FS1997] Fayad, M., E. and Schmidt, D., *Object-Oriented Application Frameworks*, Communications of the ACM, Vol. 40, No. 10, 1997.
- [GBL+1998] Gibson, J., Berger, T., Lookabaugh, T., Lindbergh, D., and Baker, R., *Digital Compression for Multimedia: Principles & Standards*, Morgan Kaufmann Publishers, 1998.
- [GPB+2001] Garaventa, P., Pockaj, R., Bonamico, C., and Lavagetto, F., *Mesh Simplification Algorithm Applied To MPEG-4 Compliant 3D Facial Models*, University of Genova, 2001.
- [HPH2001] Haghghi, K., Pourmohammadi, Y., and Alnuweiri, H., *Realizing MPEG-4 Streaming Over the Internet*, IEEE, 2001.
- [IBM2002] IBM, *IBM Research*, <http://www.research.ibm.com/compsci/multimedia/>, Retrieved: 2002.

- [IM12002] Telecomitalialab, *MPEG-4 Systems Reference Implementation (Im1)*, <http://mpeg.telecomitalialab.com/faq/mp4-sys/>, Retrieved: 2002.
- [ISO1997] ISO 9241-1, *Ergonomic requirements for office work with visual display terminals*, ISO, 1997.
- [ITU2002] ITU, *ITU Telecommunication Standardisation Sector*, <http://www.itu.int/ITU-T/>, Retrieved: 2002.
- [Jor1998] Jordan, P., *An Introduction to Usability*, Taylor & Francis, 1998.
- [KJK+2001] Kim, M., Jeon, J., Kwak, J., Lee, M., and Ahn, C., *Moving object segmentation in video sequences by user interaction*, Image and Vision Computing, 2001.
- [KKC+2001] Kim, K., Kim, H., Cheong, W., and Kim, K., *Design and Implementation of MPEG-4 Authoring Tool*, IEEE, 2001.
- [Koe1999] Koenen, R., *MPEG-4 Overview*, IEEE Spectrum, Vol. 36, No. 2, 1999.
- [Koe2000] Koenen, R., *Overview of the MPEG-J Standard*, ISO/IEC JTC 1/SC29/WG 11, 2000.
- [Koe2001] Koenen, R., *Overview of the MPEG-4 Standard*, ISO/IEC JTC1/CS29/WG11, 2000.
- [Koe2002] Koenen, R., *Overview of the MPEG-4 Standard*, ISO/IEC JTC1/SC29/WG11, 2002.
- [KSM1999] Kneip, J., Schmale, B., and Möller, H., *Applying and Implementing the MPEG-4 Multimedia Standard*, IEEE, 1999.
- [KWC2000] Kim, M., Wood, S., and Cheok, L., *Extensible MPEG-4 Textual Format (XMT)*, ACM, 2000.
- [Mac2002] Macromedia Inc., *Macromedia Director*, <http://www.macromedia.com/software/director/>, Retrieved: 2002.
- [Mac2003] Macromedia Inc., *Macromedia Authorware*, <http://www.macromedia.com/software/authorware/>, Retrieved: 2003.
- [Mar2002] Martínez, J., *MPEG-7 Overview*, ISO/IEC JTC1/SC29/WG11, 2002.
- [Mic2002] Microsoft Corporation, *Microsoft Visio 2000*, <http://www.microsoft.com/office/visio/>, Retrieved: 2002.

- 
- [MPE2002] MPEG4IP, *MPEG4IP: Open Source, Open Standards, Open Streaming*, <http://mpeg4ip.sourceforge.net/index.php>, Retrieved: 2002.
- [mpe2003] mpegable, *mpegable X4*, <http://www.mpegable.com>, Retrieved: 2003.
- [MPEG2002] Convenor of MPEG, *JVT completes the technical work leading to AVC availability!*, ISO/IEC JTC 1/SC 29/WG 11 N5320, 2002.
- [mTr2002] Hunter, G., *mTropolis, the Greatest CBD Tool that Never Was*, <http://www.mtropolis.com/>, Retrieved: 2002.
- [MTr2002] MTrec, *MTrec Multimedia Technology Research Center*, [http://www.mtrec.ust.hk/projects/mpeg\\_4.html](http://www.mtrec.ust.hk/projects/mpeg_4.html), Retrieved: 2002.
- [Nie1998] Nielsen, J., *Guerrilla HCI: Using Discount Usability Engineering to Penetrate the Intimidation Barrier*, Academic Press, 1998.
- [Par2002] Parallel Graphics Limited, *Internet Scene Assembler*, <http://www.parallelgraphics.com/products/isa/>, Retrieved: 2002.
- [RME1996] Reddy, B., Markowitz, S., and Epelman-Wang, H., *User Interfaces for Authoring Systems with Object Stores*, Proceedings of COMPCON '96, IEEE, 1996.
- [Rub1994] Rubin, J., *Handbook for Usability Testing*, John Wiley & Sons Inc., 1994.
- [Rub2001] Ruber, P., *Minimultimedia*, Internet World, 2001.
- [Sig1999] Siglar, J., *Multimedia Authoring Systems FAQ Version 2.234*, <http://www.tiac.net/users/jasiglar/MMASFAQ.HTML>, Retrieved: 1999.
- [Sik2002] Sikora, T., *The Structure of the MPEG-4 Video Coding Algorithm*, <http://wwwam.hhi.de/mpeg-video/papers/sikora/>, Retrieved: 2002.
- [Song2002] Avaro, O., *SoNG (portalS Of Next Generation)*, <http://www.octaga.com/SoNG-Web/>, Retrieved: 2002.
- [SS2001] Sarris, N. and Strintzis, M., *Constructing a Videophone for the Hearing Impaired Using MPEG-4 Tools*, IEEE, 2001.
- [TSJ+2001] Topiwala, P., Sullivan, G., Joch, A., and Kossentini, F., *Performance Evaluation of H.26L, TML 8 vs. H.263++ and MPEG-4*, VCEG-N18, 2001.
- [Tud1995] Tudor, P. N., *MPEG-2 video compression*, Electronics and Communication Engineering Journal, 1995.

- [VCC2002] Viljoen, D., Calitz, A., and Cowley, N., *A 2-D MPEG-4 Multimedia Authoring Tool*, ACM SIGGRAPH, 2002.
- [W3C2001] World Wide Web Consortium, *Synchronized Multimedia Integration Language (SMIL 2.0)*, <http://www.w3.org/TR/smil20/>, Retrieved: 2001.
- [Win2003] Microsoft Corporation, *Windows Media 9 Series*, <http://www.microsoft.com/windows/windowsmedia/>, Retrieved: 2003.
- [X3D2002] Web3D CONSORTIUM, *Extensible 3D (X3D) Graphics*, <http://www.web3d.org/x3d.html>, Retrieved: 2002.
- [Yos2001] Yoshida, J., *MPEG-4 Tapped for Set-Tops, Home Networks*, Electronic Engineering Times, 2001.

---

## Appendix A : Profiles in MPEG-4

In order to allow effective implementations of the standard, MPEG-4 Systems, Visual, and Audio have been broken up into subsets that allow complexity scalability in MPEG-4 applications. The subsets known as Profiles are discussed below:

### 1. Visual Profiles

The MPEG-4 visual Profiles provides Profiles for the coding of natural, synthetic, and synthetic/natural hybrid visual content.

#### 1.2 Natural Video

There are five Profiles for natural video content in MPEG-4 Version 1:

- " **The Simple Visual Profile** provides efficient, error resilient coding of rectangular video objects, suitable for applications on mobile networks;
- " **The Simple Scalable Visual Profile** adds support for coding of temporal and spatial scalable objects to the Simple Visual Profile. It is useful for applications which provide services at more than one level of quality due to bitrate or decoder resource limitations, such as Internet use and software decoding;
- " **The Core Visual Profile** adds support for coding of arbitrary shaped and temporally scalable objects to the Simple Visual Profile. It is useful for applications such as those providing relatively simple content interactivity;
- " **The Main Visual Profile** adds support for coding of interlaced, semi-transparent, and sprite objects to the Core Visual Profile. It is useful for interactive and entertainment quality broadcast and DVD applications; and
- " **The N-Bit Visual Profile** adds support for coding video objects having pixel depths ranging from 4 to 12 bits to the Core Visual Profile. It is suitable for use in surveillance applications.

---

MPEG-4 Version 2 added the following Profiles:

**The Advanced Real-Time Simple Profile (ARTS)** provides advanced error resilient coding techniques for rectangular video objects and improved temporal resolution stability with the low buffering delay. It is suitable for real time coding applications; such as the videophone, videoconferencing and the remote observation;

- " **The Core Scalable Profile** adds support for coding of temporal and spatial scalable arbitrarily shaped objects to the Core Profile. It is useful for applications such as the Internet and mobile phones; and
- " **The Advanced Coding Efficiency (ACE) Profile** improves the coding efficiency for both rectangular and arbitrary shaped objects. It is suitable for applications such as mobile broadcast reception and other applications where high coding efficiency is required.

### 1.3 Synthetic and Synthetic/Natural Hybrid Video

MPEG-4 Version 1 Profiles:

- " **The Simple Facial Animation Visual Profile** provides a simple means to animate a face model, suitable for applications such as audio/video presentation for the hearing impaired;
- " **The Scalable Texture Visual Profile** provides spatial scalable coding of still image (texture) objects useful for applications needing multiple scalability levels, such as mapping texture onto objects in games, and high resolution digital still cameras;
- " **The Basic Animated 2-D Texture Visual Profile** provides spatial scalability, and mesh based animation for still image (textures) objects and also simple face object animation; and
- " **The Hybrid Visual Profile** combines the ability to decode arbitrary shaped and temporally scalable natural video objects (as in the Core Visual Profile) with the ability to decode several synthetic and hybrid objects, including simple face and animated still image objects. It is suitable for various content-rich multimedia applications.

The Version 2 Profiles for synthetic and synthetic/natural hybrid visual content are:

- " **The Advanced Scaleable Texture Profile** supports decoding of arbitrary shaped texture and still images. It is useful for applications that require fast random access as



---

well as multiple scalability levels and arbitrary-shaped coding of still objects. Examples are fast content based still image browsing on the Internet, multimedia enabled PDA's, and Internet ready high resolution digital still cameras;

- " **The Advanced Core Profile** combines the ability to decode arbitrary shaped video objects (as in the Core Visual Profile) with the ability to decode arbitrary shaped scalable still image objects (as in the Advanced Scaleable Texture Profile.) It is suitable for various content-rich multimedia applications such as interactive multimedia streaming over Internet; and
- " **The Simple Face and Body Animation Profile** is a superset of the Simple Face Animation Profile, adding body animation.

In subsequent Versions, the following Profiles were added:

- " **The Advanced Simple Profile** has a lot in common with the Simple profile in that it has only rectangular objects. It has however, a few extra tools that make it more efficient: B-frames, extra quantization tables and global motion compensation;
- " **The Fine Granularity Scalability Profile** allows truncation of the enhancement layer bitstream at any bit position so that delivery quality can easily adapt to transmission and decoding circumstances. It can be used with Simple or Advanced Simple as a base layer;
- " **The Simple Studio Profile** is a profile with very high quality for usage in studio editing applications. It only has I frames, but it does support arbitrary shape and multiple alpha channels and bitrates of up to 2 Gigabit per second are supported; and
- " **The Core Studio Profile** adds P frames to Simple Studio, making it more efficient but also requiring more complex implementations.

## 2. Audio Profiles

Four Audio Profiles were defined in MPEG-4 Version 1:

- " **The Speech Profile** provides HVXC, which is a very-low bit-rate parametric speech coder, a CELP narrowband/wideband speech coder, and a Text-To-Speech interface;
- " **The Synthesis Profile** provides score driven synthesis using SAOL and wavetables and a Text-to-Speech Interface to generate sound and speech at very low bitrates;

- " **The Scalable Profile**, a superset of the Speech Profile, is suitable for scalable coding of speech and music for networks, such as Internet and Narrow band Audio Digital Broadcasting (NADIB). The bitrates range from 6 kbit/s and 24 kbit/s, with bandwidths between 3.5 and 9 kHz; and
- " **The Main Profile** is a rich superset of all the other audio profiles, containing tools for natural and synthetic Audio.

MPEG-4 Version 2 added the Following four Profiles:

- " **The High Quality Audio Profile** contains the CELP speech coder and the Low Complexity AAC coder including Long Term Prediction. Scalable coding can be performed by the AAC Scalable object type. Optionally, the new error resilient (ER) bitstream syntax may be used;
- " **The Low Delay Audio Profile** contains the HVXC and CELP speech coders (optionally using the ER bitstream syntax), the low-delay AAC coder and the Text-to-Speech interface TTSI;
- " **The Natural Audio Profile** contains all natural audio coding tools available in MPEG-4, but not the synthetic ones; and
- " **The Mobile Audio Internetworking Profile (MAUI)** contains the low-delay and scalable AAC object types including TwinVQ and BSAC. This profile is intended to extend communication applications using non-MPEG speech coding algorithms with high quality audio coding capabilities.

### 3. Graphics Profiles

Graphics Profiles define which graphical and textual elements can be used in a scene.

These profiles are defined in the Systems part of the standard:

- " **Simple 2-D Graphics Profile** provides for only those graphics elements of the BIFS tool that are necessary to place one or more visual objects in a scene;
- " **Complete 2-D Graphics Profile** provides two-dimensional graphics functionalities and supports features such as arbitrary two-dimensional graphics and text, possibly in conjunction with visual objects;

- " **Complete Graphics Profile** provides advanced graphical elements such as elevation grids and extrusions and allows creating content with sophisticated lighting. The Complete Graphics profile enables applications such as complex virtual worlds that exhibit a high degree of realism; and
- " **The 3D Audio Graphics Profile** sounds like a contradictory in terms, but really isn't. This profile does not propose visual rendering, but graphics tools are provided to define the acoustical properties of the scene (geometry, acoustics absorption, diffusion, transparency of the material). This profile is used for applications that do environmental spatialization of audio signals.

Profiles under Definition or Consideration. The following profiles were under development at the time of writing this Overview; their inclusion in the standard was highly likely, but not guaranteed:

- " **The Simple 2D+Text profile** has a lot in common with the simple 2D, adding the BIFS nodes to display text which can be colored or transparent. Like simple 2D, this is a useful profile for low-complexity audiovisual devices;
- " **The Core 2D Profile** supports fairly simple 2D graphics and text. Meant for set top boxes and similar devices, it can do such things as picture-in-picture, video warping for animated advertisements, logos, and so on;
- " **The Advanced 2D profile** contains tools for advanced 2D graphics. Using it, one can implement cartoons, games, advanced graphical user interfaces, and complex, streamed graphics animations; and
- " **The X3D Core profile** is the only 3D profile that is likely to be added to MPEG-4. It is compatible with Web3D's X3D core profile under development and it gives a rich environment for games, virtual worlds and other 3D applications.

## 4. Scene Graph Profiles

Scene Graph Profiles (or Scene Description Profiles), defined in the Systems part of the standard, allow audio-visual scenes with audio-only, 2-dimensional, 3-dimensional or mixed 2-D/3-D content.

- " **The Audio Scene Graph Profile** provides for a set of BIFS scene graph elements for usage in audio only applications. The Audio Scene Graph profile supports applications like broadcast radio;
- " **The Simple 2-D Scene Graph Profile** provides for only those BIFS scene graph elements necessary to place one or more audio-visual objects in a scene. The Simple 2-D Scene Graph profile allows presentation of audio-visual content with potential update of the complete scene but no interaction capabilities. The Simple 2-D Scene Graph profile supports applications like broadcast television;
- " **The Complete 2-D Scene Graph Profile** provides for all the 2-D scene description elements of the BIFS tool. It supports features such as 2-D transformations and alpha blending. The Complete 2-D Scene Graph profile enables 2-D applications that require extensive and customised interactivity;
- " **The Complete Scene Graph Profile** provides the complete set of scene graph elements of the BIFS tool. The Complete Scene Graph profile enables applications like dynamic virtual 3-D worlds and games; and
- " **The 3D Audio Scene Graph Profile** provides three-dimensional sound positioning in relation either with acoustic parameters of the scene or its perceptual attributes. The user can interact with the scene by changing the position of the sound source, by changing the room effect or moving the listening point. This Profile is intended for usage in audio-only applications.

## **5. MPEG-J Profiles**

### **5.1 Personal - a lightweight package for personal devices.**

The personal profile addresses a range of constrained devices including mobile and portable devices. Examples of such devices are cell video phones, PDAs, personal gaming devices. This profile includes the following packages of MPEG-J APIs:

- " Network API;
- " Scene API; and
- " Resource API.

## **5.2 Main - includes all the MPEG-J API's.**

The Main profile addresses a range of consumer devices including entertainment devices. Examples of such devices are set top boxes, computer based multimedia systems etc. It is a superset of the Personal profile. Apart from the packages in the Personal profile, this profile includes the following packages of the MPEG-J APIs:

- " Decoder API
- " Decoder Functionality API
- " Section Filter and Service Information API

## **6. Object Descriptor Profiles**

The Object Descriptor Profile includes the following tools:

- " Object Descriptor (OD) tool;
- " Sync Layer (SL) tool;
- " Object Content Information (OCI) tool; and
- " Intellectual Property Management and Protection (IPMP) tool.

Currently, only one profile is defined that includes all these tools. The main reason for defining this profile is not subsetting the tools, but rather defining levels for them. This applies especially to the Sync Layer tool, as MPEG-4 allows multiple time bases to exist. In the context of Levels for this Profile, restrictions can be defined, e.g. to allow only a single time base.

## **Appendix B : Installation Instructions**

### **1. Contents of the SceneBuilder CD**

The SceneBuilder CD contains the following:

- " The SceneBuilder application with help files;
- " The EnvivioTV plug-in;
- " MP4 Tool;
- " QuickTime;
- " Sample MPEG-4 media; and
- " Sample MPEG-4 applications.

### **2. Installation Instructions**

Insert the SceneBuilder CD in the CD-ROM drive and run setup.exe. SceneBuilder and all the required applications will be installed. SceneBuilder has been tested with EnvivioTV Version 1 and QuickTime Version 5.0.2. Using SceneBuilder with any other version of EnvivioTV and QuickTime is not recommended.