NTNU
Innovation and Creativity

Composing distributed 3D scenes

Stein Olav Ness

Master of Science in Communication Technology
Submission date: June 2006
Supervisor: Leif Arne Rønningen, ITEM

Norwegian University of Science and Technology
Department of Telematics

Problem Description

The Distributed Multimedia Plays (DMP) System Architecture provides combined adaptive scene resolution and traffic control in packet networks, see http://www.item.ntnu.no/~leifarne.

This project focuses on adaptive scene composition declaration, specification and realisation, and comprises the following:

* Review of 3D multiview, autostereoscopic object oriented audiovisual scenes theory and practice
* Propose extensions to SMIL and SIP to handle adaptive composition of scenes consisting of distributed objects
* Propose and demonstrate extensions to SMIL enabling 3D, transparency and custom shapes

Assignment given: 16. January 2006
Supervisor: Leif Arne Rønningen, ITEM

Preface

The Master's Thesis concludes the 10th and final term of the Master of Science education at the Norwegian University of Science and Technology, NTNU. A Master's Thesis is an individual research work counting as 30 ECTS Credits with a normal duration of 20 weeks.

I'll use this opportunity to thank my supervisor, professor Leif Arne Rønningen at Department of Telematics for supervision and guidance throughout the work with this Master's Thesis.

Trondheim, June 12th, 2006

STEIN OLAV NESS
Student

Contents

List of Figures

List of Tables

Abbreviations

2D	Two-dimensional
3D	Three-dimensional
3GPP	Third Generation Partnership Project
API	Application Programming Interface
CAD	Computer-aided design
Call-ID	Call Identification
CSeq	Command Sequence
DHT	Distributed Hash Table
DMP	Distributed Multimedia Plays
DNS	Domain Name System
DOM	Document Object Model
DTD	Data Type Definition
DVD	Digital Versatile Disc
ETSI	European Telecommunications Standards Institute
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IETF	Internet Engineering Task Force
I/O	Input/output
IM	Instant Messaging
IMTC	Internet Multimedia Telecommunications Consortium
IP	Internet Protocol
ISO	International Organization for Standardization
LASeR	Lightweight Scene Representation
LCD	Liquid crystal display
Mbone	Internet Multicast Backbone
MMS	Multimedia Message Service
MMUSIC	Multiparty Multimedia Session Control
MPEG	Moving Picture Experts Group
MSRP	Message Session Relay Protocol

MTU	Message Transfer Unit
NVSG	Nvidia Scene Graph
OpenGL	Open Graphics Library
OSG	Open Scene Graph
P2P	Peer to Peer
PDU	Packet Data Unit
PNG	Portable Network Graphics
PSTN	Public Switched Telephone Network
RFC	Request For Comment
RGB	Red Green Blue
RGBA	Red Green Blue Alpha
RTCP	Real-time Transport Control Protocol
RTP	Real-time Transport Protocol
RTSP	Real-time Streaming Protocol
SAP	Session Announcement Protocol
SCTP	Stream Control Transmission Protocol
SDP	Session Description Protocol
SIMPLE	SIP for Instant Messaging and Presence Leveraging Extensions
SIP	Session Initiation Protocol
SIPS	SIP with Security
SMIL	Synchronized Multimedia Integration Language
SMTP	Simple Mail Transfer Protocol
SOSIMPLE	Self Organizing SIMPLE
SS7	Signaling System no. 7
STD	Standard
SVCD	Super Video Compact Disc
SVG	Scalable Vector Graphics
SYMM	Synchronized Multimedia
TIPHON	Telecommunications and Internet Protocol Harmonization Over Networks
UAC	User Agent Client
UAS	User Agent Server

UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VCD	Video Compact Disk
VoIP	Voice over Internet Protocol
VRML	Virtual Reality Modeling Language
W3C	World Wide Web Consortium
X3D	Extensible 3D
XCAP	XML Configuration Access Protocol
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language
xmlns	Extensible Markup Language Namespace
XPath	XML Path Language
ZIP	Compression File Format

Abstract

The frontpage description identifies the three main issues answered by this Master's Thesis. First a textual review of different 3D scene theories and practices, languages and standards (X3D, MPEG-4 Interactive, SVG, NVSG, OSG, SMIL and LASeR). Based on information obtained from documentations and other sources, their respective application areas, weaknesses and strengths and other properties are first addressed separately and then compared in a comparison chart. The second issue propose extensions to SMIL and SIP to handle adaptive composition of distributed scenes with distributed objects (objects may be i.e. video streams of human participants or furniture). Design goals of the system are established, requirements of distributed scene systems are identified and various models are presented. The third issue addresses proposals and demonstrates SMIL extensions allowing 3D, transparency and custom shapes in SMIL scenes. The goals of this part are to define methods for inclusion of SVG and X3D features in SMIL and to present methods used by X3D and SVG to allow 3D, transparency and custom shapes in their respective document environments.

To establish a knowledge base, a separate section describes background technologies, carries out comparisons between the competing application layer protocols XMPP and SIP, presents the session layer protocol RTP used to transport real-time media streams and the underlying transport layer protocols: TCP and UDP, as well as mechanisms, languages and theories used to send and represent data/documents.

The review section presents theories for representation and creation of 3D scenes, as well as theories that, despite no 3D options, may be teamed with other theories, allowing rich presentations of scenes. The conclusion deducted from the reviewed theories is a comparison chart presenting equalities and differences, application areas, etc.

SVG and X3D techniques may be used to represent 3D, transparent and custom shaped objects, these techniques are presented in a separate section and may be useful to team with SMIL's layout functionality when creating distributed 3D scenes. This section also demonstrates practical use of a number of SVG and X3D features. SVG and X3D may be embedded or referenced from SMIL documents to exploit their respective options when creating scenes with SMIL.

A proposal for exchange of SMIL scenes and object descriptions are presented, using MSRP sessions established with help from SDP/SIP over TCP. MSRP messages may

carry MIME content, i.e. SMIL, SVG and X3D documents, over P2P sessions between peers (the participants of distributed 3D scenes). TCP is a reliable protocol thus introducing potential problems due to real-time requirements. These problems may be solved by developing a set of rules addressing legal methods of updating scenes and exchanging scene descriptions (further work). The rtpmap element allows dynamic encoding of media streams (restricted to a set of predefined levels) included and described from SMIL documents, removing some burdens introduced if SDP used to describe media streams. Ambulant Player (a SMIL player) and Gaim (instant messages) are demonstrated to show how two existing applications play and exchange SMIL documents, despite these two applications' lack of support of hence SVG, X3D and real P2P sessions.

The discussion and conclusion parts of this Master's Thesis discusses the choice of protocols, binary compared to XML scene representations, and an alternative "hub-like" approach of scene description handling in contradict to the all-to-all/P2P approach described earlier. Problems caused by lack of present support, i.e. when SMIL documents are containing referenced or embedded SVG or X3D documents, are addressed, and at last proposals for future works.

1 Introduction

The first section of this Master's Thesis presents the background , limits of the work, other work, criticism of the most important sources used to obtain information, software tools and the structure of the thesis.

1.1 Background

The areas of this Master's Thesis have their foundation as potential parts of Distributed Multimedia Plays Virtual Dinner [1], a research project initiated by Leif Arne Rønningen at the Department of Telematics at the Norwegian University of Science and Technology. How to use SMIL and SIP to create and distribute scene descriptions rises many questions. This Master's Thesis will try to answer and sketch solutions to some of these questions and present ideas of how to realize them. "Distributed scenes" is an expression including the issues of creating scenes consisting of media streams from participants located at different (geographical) locations, and how to distribute a mutual scene description to the participants in a scene.

1.2 Limits of the Work

The Master's Thesis focus at scene description and protocols used to exchange scene descriptions. There is no focus on how to handle the storage of scenes at different locations or how participants interact through GUI functionality. To build a complete application implementing all the functionality of such a distributed system (including transparency, 3D and custom shaped objects) would be very labor-intensive and take far more time than available. No players or applications supports the combinations of document standards introduced in this Master's Thesis at the present point of time. Instead of creating a complete demonstrator application, most of the functionality will be demonstrated separately and in a smaller scale using the principle of divide and conquer. Security issues like authorization and privacy are not covered.

1.3 Other Works

During the work with this Master's Thesis no other work with precisely the same goals where discovered. Some of the aspects, like extending SIP with MSRP to get a fully P2P instant message session, are of course handled by others, but the application areas of these are not fully overlapping the main ideas of this Master's Thesis which seems to be unique.

1.4 Software Tools

The Master's Thesis are written in LaTeX. Editing and examining different types of documents and code are done using a simple text editor. Testing SMIL and SVG documents are carried out using several different applications. The full list of software and applications are listed in Appendix G.

1.5 Sources of Material

A large number of sources are used during the work with this Master's Thesis. The sources are (with no exceptions) listed in the Reference section, and the URL sources are also made available offline (as attached files, see Appendix N). Critics of the most often referenced sources are carried out below.

1.5.1 Wikipedia

Wikipedia is used to obtain more information about words or a topics than a regular dictionary or encyclopedia can tell. Wikipedia is only used to get information about "simple" topics where the political ideas and ideology of the article writer does not color the article in any way, and the accuracy of the article is undisputable.

1.5.2 Original Specifications and Documentations

Original specifications and documentations are first hand documents written by the creators and inventors of a specific subject. There are differences when it comes to

the level of detail in the specifications, but the majority of the specifications are well written and the content of the specifications and/or documentations reflect the subject described in a fair way as the writers most often want the subject to be described in a representative and objective manner.

1.5.3 IETF Standards

The most important sources of this Master's Thesis, IETF RFCs need their own section for explanation. It is important to know the difference between the IETF standards, how the IETF standardization process are carried out and the types of papers published by IETF. The general subjects of the IETF standards are protocols and mechanisms of the Internet [2].

> "At each stage of the standardization process, a specification is repeatedly discussed and its merits debated in open meetings and/or public electronic mailing lists, and it is made available for review via world-wide on-line directories."
>
> RFC2026 [2]

All the documents passing through the IETF standardization process are publicly available as "Request for Comments (RFCs)".

- **Proposed Standard:** First level, generally stable but still immature and may be changed. Implementation is not required at this level.
- **Draft Standard:** Second level, well understood and quite stable. At least two independent implementations exist, and these implementations should be inter operable.
- **Internet Standard:** Technically mature, well implemented. A standard also get a STD number in the Standards Track (still keeping the RFC number).
- **Experimental:** Not on the standards track, a part of some research.

- **Informational:** Not on the standards track, general information for the Internet community.
- **Historic:** Not on the standards track, a specification that has been superseded by a more recent specification.

RFC2026 [2]

The Master's Thesis incorporate protocols and protocol proposals having reached different levels of the IETF standardization process, or even not become a Proposed Standard yet.

1.6 Structure of the Thesis

After the formal sections of the thesis a number of important background technologies employed in the work are presented to give the reader a certain knowledge base. A review study of different 3D multiview, autostereoscopic object oriented audiovisual scenes theory and practice are then carried out.

The following sections address the questions from the description and the additional questions respectively. Theories and demonstrations of SMIL (with extensions allowing 3D, transparency and custom shapes) techniques enabling more advanced and exiting scenes than scenes composed using SMIL only are presented in Section 4. The next section (Composition of Distributed Scenes) address the requirement specifications of a system and how it is modeled.

The Discussion part discuss how different approaches towards solutions are taken, alternative approaches, if the proposed system conform with the requirement specifications presented in the previous section, as well as experience gained concerning issues worked with in the Master's Thesis.

2 Technology Background

This section describe the fundamentals of protocols, standards, languages and software used or considered used in this Master's Thesis. The task of exchanging SMIL documents apparently have many issues quite similar to instant messaging situations where people are writing text messages to each other, the main difference is that communicating applications shall exchange the SMIL documents and interpret the received documents using parsers. The overall delays and latencies introduced by the network have to be short, in fact as close to real-time as possible.

This section look deeper into, compare and discuss two IETF projects for instant messaging purposes: JABBER/XMPP and SIP/SIMPLE, and discuss which of them is the most suitable protocol carrying SMIL documents between scene participants. Protocols are discussed, two software clients: Gaim and Ambulant, the XML language and short introductions to scene graph theory and binary encoding.

2.1 SIP

To exchange SMIL documents between the participants an IM approach may give a well defined suite of protocols and methods. There are currently two protocols being developed by IETF supporting the services of instant messaging and presence awareness. The instant messaging version of SIP is shortened SIMPLE (Session Initiation Protocol for Instant Messaging and Presence Leveraging Extensions). There is also work on a SOSIMPLE protocol, a completely distributed variant of SIMPLE without any servers at all. The Jabber Software Foundation has developed the XMPP (eXtensible Messaging and Presence Protocol).

SIP is a signaling protocol used to establish communication sessions between two or more endpoints. These endpoints could be IP-telephones, IM clients or other applications [3]. Communication sessions that may include various types of data, multimedia, etc [4]. SIP is a text based protocol, similar to HTTP and SMTP.

> "SIP supports five facets of establishing and terminating multimedia communications:

- "**User location:** Determination of the end system to be used for communication.
- **User availability:** Determination of the willingness of the called party to engage in communications.
- **User capabilities:** Determination of the media and media parameters to be used.
- **Session setup:** "Ringing", establishment of session parameters at both called and calling party.
- **Session management:** Including transfer and termination of sessions, modifying session parameters, and invoking services."

Javvin network management and security [5]

SIP was created by IETF's MMUSIC (Multiparty Multimedia Session Control) working group in 1996 and is adopted and recognized as a de-facto standard by many companies and organizations, naming 3GPP, IMTC, ETSI TIPHON. SIP is a common protocol used in VoIP and IP telephony. SIP initiates, modifies and terminates a session and provides services present in PSTN and SS7. SIP behaves as a transporter for the Session Description Protocol, SDP. SDP describes the media content of the session, i.e. the kind of codecs and IP ports that are used.

2.1.1 SIP Architecture

In addition to the hardware or software endpoints SIP also requires "Proxies" and "Registrars" to work. The SIP architecture is described in Figure 1.

- **Proxy Server:** Routes requests to the user, authenticates and authorizes users for services, implements call-routing policies and provides some features to the users.
- **Registrar:** SIP provides a registration service that allows users to upload their current locations to a register for use by proxy servers.

- **Redirect Server:** Redirects the request back to the server in case the request does not reach the recipient. In case recipient have moved temporarily or permanently.
- **Location Server:** The Location Server handles the addresses registered to the Registrar.
- **User Agent Client:** The UAC represents the end system who sends requests.
- **User Agent Server:** The UAS represents the end system who responds to the requests from the UAC.

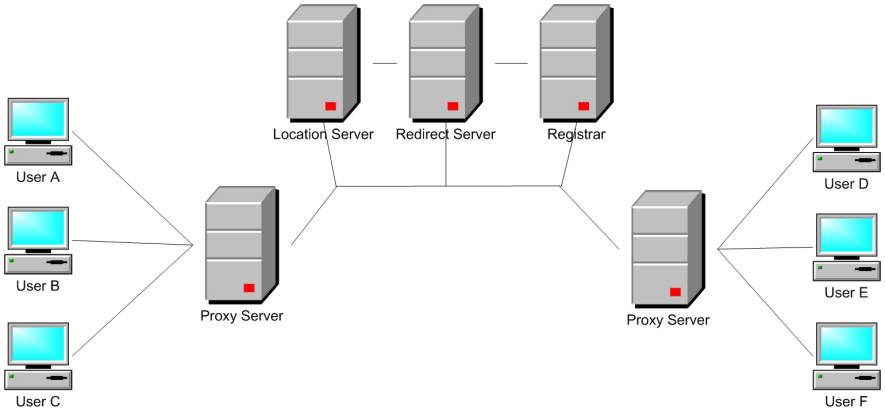


Figure 1: SIP architecture

The user agent clients acts and sends a request after a stimulus from a user (the user may for instance click a mouse button). The UAS then receives the request and responds to it. The message passes a number of proxies on the way from the UAC to the UAS. The request from the UAC must at least contain these header fields: `To`, `From`, `CSeq`, `Call-ID`, `Max-Forwards`, and `Via`. These header fields are mandatory in all SIP requests.

2.1.2 SIP Sessions

- "**INVITE:** Invites a user to a call.
- **ACK:** Acknowledgment is used to facilitate reliable message exchange for INVITEs.

- **BYE:** Terminates a connection between users.
- **CANCEL:** Terminates a request, or search, for a user. It is used if a client sends an INVITE and then changes its decision to call the recipient.
- **OPTIONS:** Solicits information about a server's capabilities.
- **REGISTER:** Registers a user's current location.
- **INFO:** Used for mid-session signaling."

e-Multimedia [6]

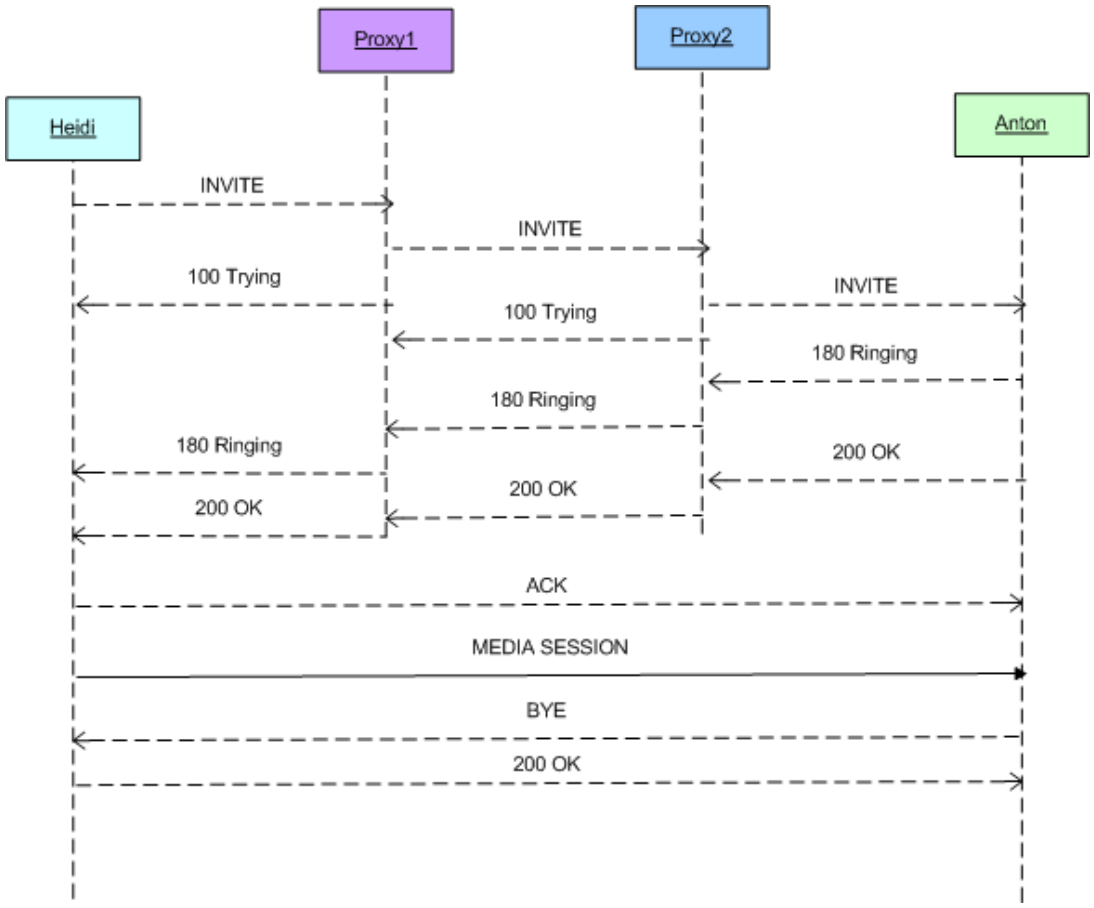


Figure 2: SIP session establishment

The "Media Session" only create a sense of a session in the case of i.e. VoIP or RTP transfer. Transfer of IM messages or SMIL documents are not convenient because

there is only a sense of an imaginative session where all the messages an fact stand completely alone. Each message require setup and tear down for itself, meaning all the messages stand alone. This is mostly suitable for paging services (sending one stand alone message) and similar. Each message represent a complete SIP transaction, and there is no sense of each message as a part of a whole/a session.

2.1.3 SIP Message Method

The Message Method is an extension to SIP allowing transfer of instant messages, it inherits SIP, thus SIP's request routing and security features [7]. The Message Method carry the content in form of MIME body parts. Exchanging messages does not create a dialog, each message still stands alone and there are no interconnection between messages. Each message has to pass one or more proxies following the SIP signaling path, this implicates a rather long time for passing the message. Desired performance of a real-time distributed system may be difficult to satisfy using the SIP method message, that is why SIMPLE is taken into consideration.

2.2 SDP

The Session Description Protocol (SDP) describes "real-time multimedia session description purposes" [8]. SDP or the Session Description Protocol is an Internet Standard developed by IETF's Network Working Group. The purpose of SDP is to convey advertisement of conference sessions and to pass communication setup information to participants of a communication session. SDP is a protocol which subject is to describe sessions, it is not a transport protocol. SDP is based on a textual description. The recipient of the SDP message decides to participate in the session after making a decision upon the containments of the SDP message. SDP may use different transport protocols, including SIP [8]. The format of a SDP message are shown in Figure 3.

```
m=<media> <port> <protocol> <format list>
```

Figure 3: SDP format
[9]

2.3 SIMPLE

SIMPLE solves the issue of the bad real-time performance of message delivery introduced by message paths following the signaling path. In the SIMPLE protocol, content (e.g. video streams) flow peer to peer and only the signaling travel through proxies. Compared to SIMPLE, SIP offer no special message session implicating that text messages also follow the signaling path. A SIMPLE message session is treated just like any other type of multimedia session. The sender and the receiver negotiates a session using the same SIP mechanisms as when a multimedia session is negotiated. When the session is initiated the messages with MIME content (MIME is described in Section 2.9) travel directly from the sender to the receiver without passing through a number of proxies. This will solve the issue of bad real-time performance using the original SIP Method Message (Section 2.1.3). A result achieved using SIMPLE is a message session treated just like any media stream. If the MIME type indicates for instance content with SMIL documents, this media stream can be considered as a SMIL stream (one independent media stream in each direction between two parts) [10].

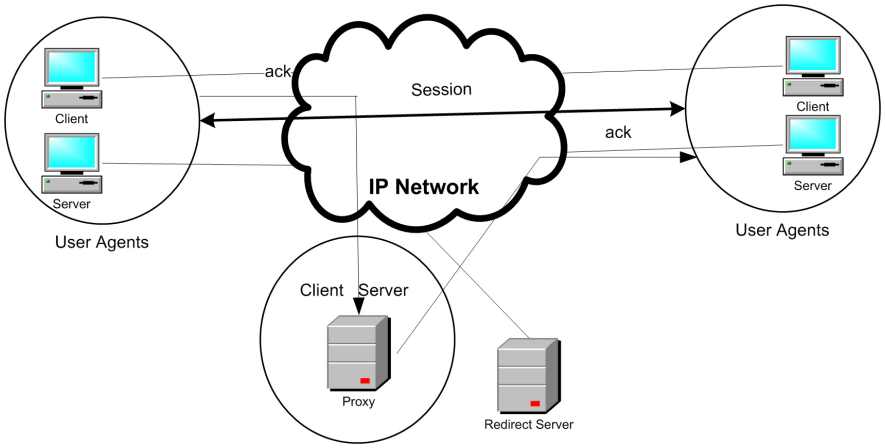


Figure 4: SIMPLE architecture

SIMPLE messages follow the same route as the RTP streams between the user agents and the user clients (peer to peer on Figure 4). The most popular implementation of SIMPLE today is perhaps Microsoft Windows Messenger.

A future alternative to SIMPLE may be SOSIMPLE. SOSIMPLE means Self Organizing SIMPLE, the whole acronym is written "Self Organizing Session Initiation Protocol for Instant Messaging and Presence Leveraging Extensions". SOSIMPLE is a fully

distributed 3D scene-system with no central entities, all the tasks are taken care of by the peers in the network. Having no central servers introduce the SOSIMPLE protocol version of SIP as an option and may become a realistic alternative in the future when the standard is ready. SOSIMPLE is described in Appendix H.

2.4 MSRP

> "MSRP (Message Sessions Relay Protocol) is a protocol for near-real-time, peer-to-peer exchange of binary content without intermediaries, which is designed to be signaled using a separate rendezvous protocol such as SIP."
>
> Relay Extensions for Message Sessions Relay Protocol (MSRP) [11]

MSRP is a session-oriented instant message transport protocol developed by the IETF Simple Working group [9]. The Message Method of SIP introduces large overhead and poor real-time performance, and XMPP (despite lower overhead) also introduce poor performance as message travel through intervening servers. MSRP solve some of these problems and works in a similar fashion to RTP or other media streams, delivering message streams between clients. MSRP is not a stand alone protocol, but works together with i.e. SIP.

> "MSRP sessions are managed using the Session Description Protocol (SDP) offer/answer model carried by a signaling protocol such as the Session Initiation Protocol (SIP)."
>
> The Message Session Relay Protocol [9]

The alternative to MSRP, using SIP to pass SMIL documents, makes all the messages traverse through intervening servers. In contradict, exchanging several SMIL documents over MSRP (the whole MSRP session) will need fewer SIP requests than a SMIL document portioned and send over a regular SIP network. The intervening servers are also removed the burden of all the signaling, client to client operations are faster and security are easier to apply.

2.5 XMPP: The Jabber Project

Figure 5: JABBER Software Foundation logo

The Extensible Messaging and Presence protocol (XMPP) is a protocol for streaming and exchanging XML data between network entities [12].

> "The Extensible Messaging and Presence Protocol (XMPP) is an open Extensible Markup Language [XML] protocol for near-real-time messaging, presence, and request-response services."
>
> RFC3920 [12]

XMPP has a client-server architecture (see Figure 6) where clients access servers over TCP connections. The server manages session(s) and handles the XML streams between clients. Information may be any kind of text messages or XML documents. All the messages in a session must traverse through intervening servers.

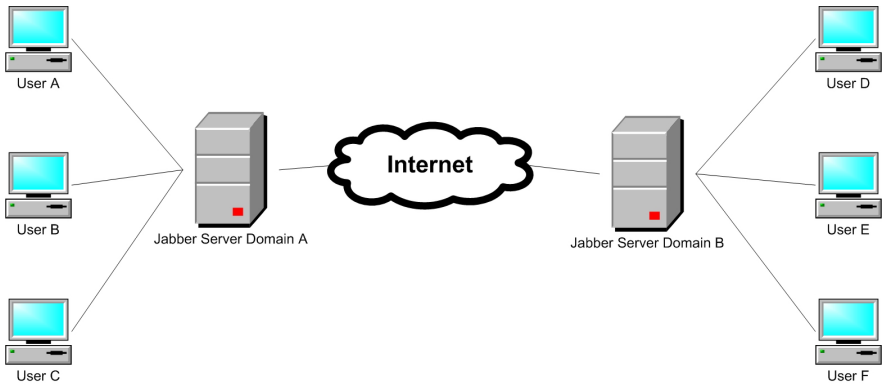


Figure 6: XMPP architecture

> "XMPP uses XML namespaces to extend the stanzas for the purpose of providing additional functionality."

RFC3921 [13]

IETF has approved XMPP, meaning it is a robust standard upon which developers can rely, and be confident that it won't change in the nearest future. There are also gateways to SIP/SIMPLE which enables transport between these two (different) IETF standards. XMPP may suit a number of distributed systems exchanging XML data well, but as Figure 6 shows, all messages have to pass through a number servers along the path from the sender to the receiver, introducing significant delay and results in poor real-time performance.

2.6 XMPP vs. SIMPLE

Basically, SIP negotiates, manages and terminates media sessions. These media sessions may be comprised by i.e. RTP streams. SIMPLE provides extensions to SIP giving the possibilities of IM and presence services. The initial message goes via a server, but later all messages are transported peer-to-peer. SIMPLE is not a particular protocol, but an extension to SIP. XMPP has been stable since 1999 and the IETF approval in 2004 makes it the first IETF-approved messaging and presence standard. The protocols operates by negotiating an XML stream between a client and a server. Once the XML stream is established and authenticated the user may send XML fragments over the XML stream to operate IM and presence data [14].

Name	Jabber/XMPP	SIP/SIMPLE
Source code	Open	Open
Transport	XML data transport	Signaling and media streams
Organization behind	Jabber Software Foundation	SIP for Instant Messaging and Presence Leveraging Extensions Working Group
Standardization organ	IETF	IETF
Focus of services	IM, presence	Mobile phone, PDA, IP telephone, IM, general purposes
Communication	Slower, via server	Real-time, P2P, Signaling via server(s)
Size of transported data	Unlimited	Undefined, portioned into chunks
Need of bandwidth	Low	Low when session is established
Extensible	Yes	Yes
IM functionality	Chat, contact list, group chat, file transfer, contact sharing	Chat session, MIME content
Underlying protocols	TCP	TCP or UDP
Application	May be extended across applications	SIP Clients
Message transport	Through server(s)	Through server(s), P2P with MSRP
Message size	No limit	No limit
Companies behind	Hewlett Packard, Intel, Sony, Hitachi	IBM, Microsoft, Novell, Sun Microsystems
Application developers	Mostly in the open source community	Big vendors
Status	Finished	Under development
Addresses	User@host	Flexible (sip:user@host.domain)

Table 1: XMPP vs. SIMPLE
[14], [15]

A conclusion deduced from comparing XMPP and SIP/SIMPLE is that real-time demands in a distributed multimedia system requires a peer-to-peer transport system for scene descriptions, excluding XMPP with all its messages traveling through intervening servers leaving behind SIP/SIMPLE/MSRP as the only option between these two protocols.

2.7 Transport Layer Protocols

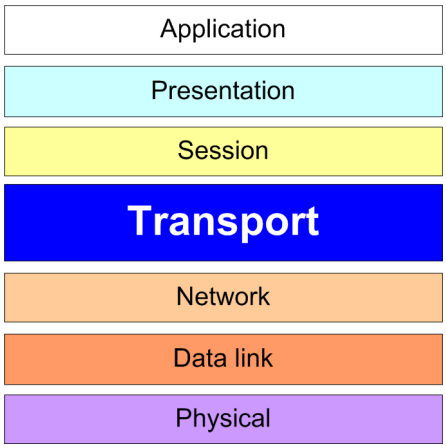


Figure 7: OSI model

Transport protocols are intermediate protocols between the application above and the underlying IP layer (network layer). The type of suitable transport protocol depends on the application/session type above the transport layer. Figure 7 shows the location of the transport layer in the ISO OSI protocol stack [16]. Both of the presented transport protocols work in cooperation with the underlying (at the network layer) IP protocol.

2.7.1 UDP

The User Datagram Protocol (UDP) is a transport layer protocol providing a fast and unreliable way to send packets called datagrams. Packets may arrive out of order or be lost, and the receiver does not acknowledge received datagrams. UDP is suitable as a transport protocol for media (i.e. as an underlying protocol for RTP) which does not suffer if a small number of packets are discarded or arrive out of order [17].

2.7.2 TCP

The Transmission Control Protocol (TCP) is a reliable stream based protocol that can guarantee that all packets arrive to the destination and that they arrive in order. TCP provides acknowledgement of received packets and retransmission of lost packets. [18] An alternative to TCP is the Stream Control Transmission Protocol offering some advantages if the network connection is unreliable or suffer from a large amount of loss (see Appendix M).

2.8 RTP

The Real-time Transport Protocol (RTP) is an IETF protocol standard used to transport audio and video over the internet. RTP only transports data, session control (call setup and tear-down) are usually controlled by SDP/SIP [19]. RTP are often used together with RTSP (Real-time Streaming Protocol) which is a protocol allowing users to remotely control the streaming of media from a server [20]. The RTCP (Real-time Transport Control Protocol) is used to provide feedback of the quality of the services provided by RTP [21].

2.8.1 RTPMAP

Rtpmap is an element available in the SMIL Streaming Media Object Module. The Synchronized Multimedia Integration Language (SMIL) is discussed in the review section.

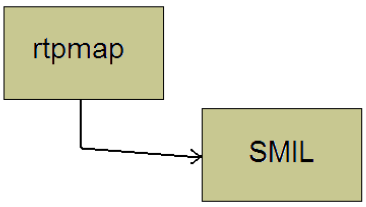


Figure 8: Rtpmap is a SMIL element

If dynamic video resolution or quality of audio is desired in a SMIL document the

SMIL streaming media object module requires the use of rtpmap elements. This attribute is contained in the media object element and encodes parameters needed to decode the dynamic payload type.

Rtpmap has two attributes, `payload` and `encoding`. `Payload` is listed in the parent's attribute and lists the optional encodings. The payload may be any kind of MIME type (Section 2.9). The following example gives the client the option to choose among three different quality levels of a rtsp stream:

```
<audio src="rtsp://www.example.org/foo.rtp" port="49170"
    transport="RTP/AVP" rtpformat="96,97,98">
    <rtpmap payload="96" encoding="L8/8000" />
    <rtpmap payload="97" encoding="L16/8000" />
    <rtpmap payload="98" encoding="L16/11025/2" />
</audio>
```

Figure 9: SMIL Streaming Media Object Module
[22]

2.9 MIME

> "MIME defines mechanisms for sending other kinds of information in e-mail, including text in languages other than English using character encodings other than ASCII as well as 8-bit binary content such as files containing images, sounds, movies, and computer programs."
>
> Wikipedia [23]

SIMPLE and MSRP messages may contain MIME (Multipurpose Internet Mail Extensions) data in their message bodies [7]. A message with the header: `MIME-Version: 1.0` indicates that the body of the message contains content of the type described as `Content-type: text/plain` [24]. The MIME type of a SMIL document is: `Content-type: application/smil`.

2.10 Ambulant Player

Figure 10: Ambulant Player

> "Ambulant is an extensible cross-platform multimedia playback engine in C++, aimed primarily at playback of SMIL documents."
>
> SourceForge [25]

Ambulant version 1.1 is an open source player with full support of SMIL 2.1, the player is designed with the ability to be extended when additional functionality is needed. There are currently two implementations of the Ambulant Player, one playing SMIL 2.1 and another only playing MMS documents, a strict subset of SMIL 1.0. Both RealNetworks' RealPlayer, Apple's Quicktime, Microsoft's Internet Explorer 6 and Mozilla Firefox (with a plugin) support some kind of SMIL, but at this point of time only Ambulant supports SMIL as it is specified in the SMIL 2.1 Specification [26]. A potential drawback of Ambulant compared to another application, X-Smiles, is that the latter one also support SVG, X3D, CSS [27] and XHTML [28] (but some problems due to version support are introduced when using X-Smiles).

> "The Ambulant player will use open-source media codecs and open-source network transfer protocols."

The Ambulant Team [29]

The player is basic and the idea is that developers can develop extensions to it when they need additional functionality instead of creating a new player from scratch. Developers can concentrate on their own extensions and integrate them with the baseline player. Ambulant Player is open source, not a commercial product [29].

2.10.1 Ambulant Design Overview

- "Global playback engine that controls the playback of a single SMIL document.
- The global playback engine object has all the others hanging off it.
- Other factories for creating renderers, file readers, parsers and windows.
- The application itself is responsible for the graphical user interface, GUI, opening and closing URLs and creating the playback engine when needed.
- There is an API between all the parts of the system.
- Machine dependency is handled by creating a machine-independent abstract base class and machine-dependent subclasses. A factory then creates a machine-dependent subclass, castes it and returns it to the machine-independent base class."

Ambulant Design Documentation [30]

2.10.2 Ambulant Player Interfaces

- **Datasource interface:** To get external data to media handlers and other modules there are several datasource interfaces. A general interface passes the URI or file I/O to several implementations of the datasource until a proper one able handle the datasource is found, returning a datasource object. The datasource interface also has a buffer that allow flow control over the net [30].
- **Playable interface:** The playable interface is implemented by media handlers and media renderers. The scheduler part of the player use this interface to make pictures, video, etc. appear on the screen.

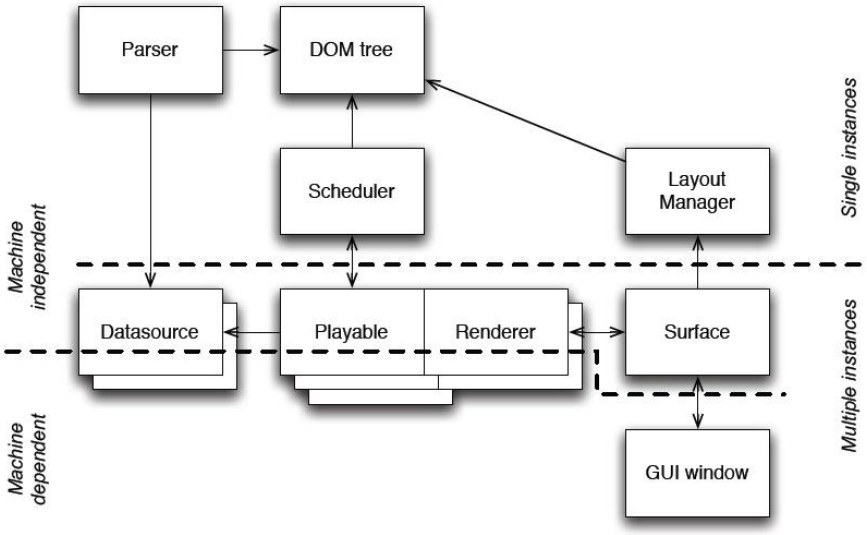


Figure 11: Ambulant overall structure diagram
[30]

Figure 11 shows the structure of the program, i.e. the parser creates a DOM tree or that the Playable Interface gets media data from the Datasource Interface.

2.11 Gaim

Gaim is an open source instant messaging client supporting SIP (Gaim 2.0.0 beta 3) in addition to other protocols like AIM, ICQ, MSN , Yahoo!, IRC, Jabber, etc [31]. Gaim supports the most common instant messaging functions (see list). There is also a free VoIP version of Gaim supporting SIP called PhoneGaim based on the instant messaging functions of Gaim [32].

Gaim functionality [31]:

- Instant messages.
- File transfer.
- Login to several different IM networks at the same time.
- Plugin functionality (the support of SIP/SIMPLE is a plugin).
- Status messages.

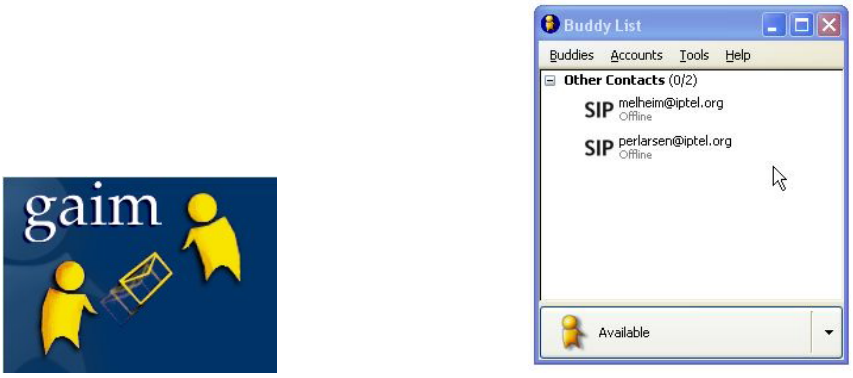


Figure 12: Gaim logo and client (version 2.0.0beta3)

- Message typing indicator.
- Operating systems supported: Linux, Windows.
- Encryption (available using plugins).

2.11.1 Gaim SIP/SIMPLE Protocol Plugin

Thomas Butter began the development of a SIP/SIMPLE protocol plugin for Gaim during "Google's Summer of Code Program 2005" [33]. A plugin is a loadable extension to Gaim. This particular plugin is free software under the GNU General Public License, just like Gaim [34]. The Gaim SIP/SIMPLE Protocol Plugin support the Message Method of SIP (see section 2.1.3 and Session Initiation Protocol Extension for Instant Messaging, RFC 3428 [7]), meaning there is no sense of a message session. All the messages follow the signaling route. The plugin has options for later support for MSRP removing a lot of burden to the intervening servers and giving better real-time delivery of messages. But as there is no other available open source implementation of SIMPLE suitable to present the ideas of this Master's Thesis, this SIP/SIMPLE Protocol Plugin is still used.

2.12 XML

The Extensible Markup Language (XML) is a human readable text format with original intentions to help publishing and exchanging data on the web. XML is the base for a

number of languages (later in this Master's Thesis SMIL, X3D, SVG are reviewed). Common for these languages based on XML is that each of them have their own XML Namespace (see Appendix C) [35].

> XML documents are made up of storage units called entities, which contain either parsed or unparsed data. Parsed data is made up of characters, some of which form character data, and some of which form markup. Markup encodes a description of the document's storage layout and logical structure. XML provides a mechanism to impose constraints on the storage layout and logical structure.
>
> Extensible Markup Language (XML) 1.1 [35]

2.13 XCAP

XCAP (XML Path Language) is a language providing the ability to address specific parts of remote XML documents and how to manipulate these specific parts . XCAP reference the URI in two parts. First part reference the document and the second part the node within the document [36].

- Remotely modify data stored in XML documents.
- XPath [37] is used for addressing the XML documents.
- Using HTTP: `GET, PUT, POST, HEAD, OPTIONS, DELETE`

> XCAP maps XML document sub-trees and element attributes to HTTP URIs, so that these components can be directly accessed by HTTP.
>
> J. Rosenberg [36]

XCAP client operations are presented in Appendix L.

2.13.1 XPath

XPath address parts of an XML document and may also be used on SMIL documents. This XPath expression address the boldface `<region ... />` tag of the SMIL document in Figure 13:

```
http://documenturl/smildocument.smil/smil/head/layout/
    region
```

In case there are several tags with the same name, a number in square brackets indicate the position of the referenced tag:

```
http://documenturl/smildocument.smil/smil/head/layout/
    region[1]
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 2.0//EN"
"http://www.w3.org/2001/SMIL20/SMIL20.dtd">
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
    <head>
        <layout>
            <region ...  />
            <region ...  />
        </layout>
    </head>
    <body>
        ...
    </body>
</smil>
```

Figure 13: XPath Referenced tag in a SMIL document

2.14 Scene Graph

The concept of a "scene graph" are underlying many of the languages and standards describing scenes. A scene graph is a hierarchical structure that organizes the scene in computer graphics. A scene graph is a object oriented structure used to represent

objects in a 3D scene in a logical way with parameters like size, shape, color, position, relationship to other objects, etc. The scene is organized in a hierarchical way with nodes that acts like parents and/or children. Sometimes both, but the node on top of the pyramid is a parent only. Theoretically spoken, a scene graph is a directed acyclic graph (see Appendix J).

When an operation affect a particular node, the effect of this operation is automatically propagated further down the hierarchy to the children of this node. This way of structuring means that similar or related objects can be grouped together by having the same parent node.

A scene graph is suitable to describe 3D worlds and 3D graphics. A node may represent an object in the modeled 3D world. A node called "bunch of keys" are composed by individual "keys" ("key" nodes) and a "key ring". The scene graph structure also keep control of the physical location of the "key" in the 3D world and the relationship between the "key ring" and the "keys".

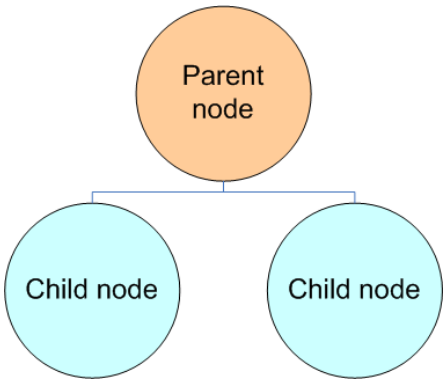


Figure 14: Scene graph tree

A large scene graph (which means most of the scene graphs) structure is often organized in a tree structure (Figure 14).

To reduce the memory requirements scene graphs use instancing. One single *bunch of keys* object are kept in the memory and all the *bunch of keys* appearing in the 3D world, with different physical representations (color, shape, etc) are instances of this object [38].

2.15 Binary Scene Descriptions

Binary encoding takes advantage of compression technologies. Binary scenes are often written using XML and scene graphs. Compared to uncompressed XML documents, binary encoded formats usually have a smaller size. Smaller sizes favors faster delivery over a network, and compression also makes parsing and reading the binary files faster. Binary coding techniques are out of scope of this Master's Thesis.

2.16 Section Summary

Summarizing the most important issues of this section, SIP comes forth as the most preferable protocol, teamed with SDP and MSRP, as a messenger carrying scene descriptions. SIP may (with MSRP) offer a P2P session for document exchange. XMPP, with it's intervening servers, introduce longer latencies and may not be as suitable as SIP in a high requirements system. The ease of sending MIME content (including SMIL documents) with MSRP also favors MSRP over SIP. The transport layer protocols UDP and TCP have been presented as well as the RTP protocol and the MIME mechanism used to transport and send media and information over networks. Ambulant Player's design overview and player interfaces have been presented to show the goals and the basic design of this SMIL player. The Gaim instant messaging client has been presented as well as XML, the base language for a number of other languages, with mechanisms allowing information exchange between XML documents. And at last the basics of scene graphs and binary encoding.

3 Review of 3D Multiview, Autostereoscopic Object Oriented Audiovisual Scenes Theory and Practice

This section carry out a review of 3D multiview, autostereoscopic object oriented audiovisual scenes theories and practice. Following languages and standards are reviewed: X3D, MPEG-4 Interactive, SVG, NVSG, OSG, LASER and SMIL. As a summary at the end of the section a comparison chart presents the most important characteristics of the reviewed languages and standards. The review of each language/standard are tried to be kept as short as possible but are still answering the key points needed in a review (Section 3.1).

3.1 Writing Reviews

When writing reviews it is important to realize what a review really is.

> "A review is a critical evaluation of a text, event, object, or phenomenon (...) a review makes an argument. The most important element of a review is that it is a commentary, not merely a summary. It allows you to enter into dialog and discussion."
>
> The Writing Center, University of North Carolina at Chapel Hill [39]

Transferring this into how to build a textual review may be carried out like this:

- First, a review gives the reader a concise summary of the content. This includes a relevant description of the topic as well as its overall perspective, argument, or purpose.
- Second, and more importantly, a review offers a critical assessment of the content. This involves your reactions to the work under review: What strikes you as noteworthy, whether or not it was effective or persuasive, and how it enhanced your understanding of the issues at hand.

- Finally, in addition to analyzing the work, a review often suggests whether or not the audience would appreciate it.

The Writing Center, University of North Carolina at Chapel Hill [39]

Using these tip-offs all the reviews in this Master's Thesis are carried out after the same pattern. Each review is divided into three subsections:

1. **Theory:** A summary of the theories behind.
2. **Practice:** How these theories are put into real life, and the application area.
3. **Summary:** A short concluding analysis.

The most important and comparative parts of the reviews are included in a comparison chart, stating properties of and differences between the reviewed standards and languages.

3.2 X3D

X3D (VRML in XML) is an ISO-Standard XML-enabled language for real-time 3D communication. The development of X3D is supervised by the Web3D Consortium.

3.2.1 X3D Theory

Figure 15: X3D logo
[40]

X3D enables 2D and 3D graphics and interactive media in physically simulated web based environments or in distributed networks on top of cross platform environments. X3D is an open standard providing real-time communication of graphics between local and network applications. X3D has a number of file formats available, including XML and it is a revision of the VRML97 ISO specification [41].

- **Componentized**: Integration with XML.
- **Extensible:** Allowing components to be added to extend functionality for vertical market applications and services.
- **Profiled:** Standardized sets of extensions to meet specific application needs.
- **Evolutionary:** Easy to update and preserve VRML97 content as X3D.
- **Broadcast/Embedded Application Ready:** From mobile phones to supercomputers.
- **Real-time:** Graphics are high quality, real-time, interactive, and include audio and video as well as 3D data.
- **Well-Specified:** Makes it easier to build conform, consistent and bug-free implementations.

Web 3D Consortium [40]

X3D also supports profiles, a profile is a subset of chosen modular blocks of functionality. A user may choose an appropriate profile according to his or her preferences. Profiles may also be extended to add more functionality (for instance network streaming). The profile is declared at the beginning of the execution, telling the player the set of components and the level of support needed to play.

The profiles shown in Figure 16 feature different services:

- **Interchange:** Communication between applications, geometry, texturing, basic lighting and animation.
- **Interactive:** 3D environment, sensor nodes for navigation and interaction, timing and extra lightning.

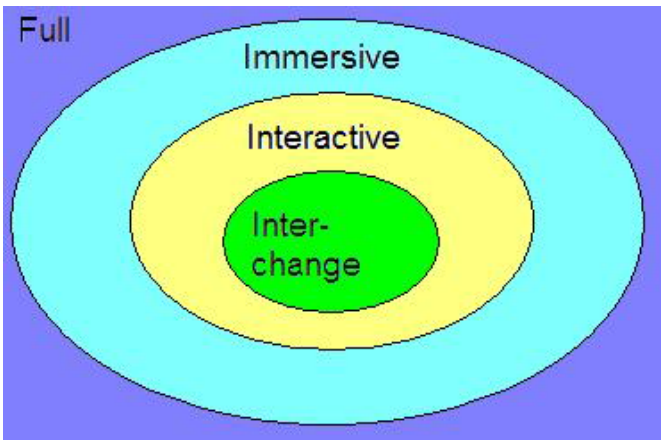


Figure 16: X3D baseline profiles

- **Immersive:** Full 3D graphics, interaction, audio, collision and fog.

- **Full:** All of the services mentioned above. The other profiles are subsets of this profile.

The base of the 3D world in X3D is a scene graph, and a single application programming interface (API) [40].

3.2.2 X3D Practice

The creators of X3D announces the advanced content and communication options available through different profiles according to needs by the user as an advantage of using X3D [40]. X3D has modular and reusable components which makes the work effort to get X3D up and running smaller, and the profiles gives an X3D creator the possibility to choose no more options than needed. The use of the XML-file format makes exchange over a network, between different applications, quite easy. X3D uses OpenGL. X3D has two extensions, "GeoVRML" allowing the design of outdoor worlds and "H-Anim" allowing human figures.

X3D is a specification under active development, the last DTD specifying X3D was published on January 10th, 2006 [42].

3.2.3 X3D Summary

X3D aims for preproduced scenes and scenarios built using advanced 3D programs like CAD or in medical imaging, visual simulations and cartography. X3D may be more focused at development and construction of advanced 3D scenes, than being a language suitable to distributed 3D scenes with a communication like Distributed Multimedia Plays Virtual Dinner [1]. But entertainment, gaming and conferencing are also optional application areas of X3D.

3.3 MPEG-4 Interactive

MPEG-4 Interactive is an additional X3D (see Section 3.2) profile designed for "broadcast, hand held devices and mobile phones" [40]. MPEG-4 Interactive is a scaled-down version of the X3D's Interactive profile. The development of MPEG-4 Interactive is also supervised by the Web3D Consortium.

> "MPEG-4 Interactive is a small footprint version of the Interactive profile designed for broadcast, handheld devices and mobile phones."
>
> Web3D Consortium [40]

3.3.1 MPEG-4 Interactive Theory

X3D is written in XML telling the media player where to find the included media (images, video, audio) and how to play in a textual format. On the other side MPEG-4 is a binary format wrapping both descriptions and media data in a binary format.

MPEG-4 Interactive is:

- Interoperable with the MPEG-4 standard.
- Supporting graphics and interactivity.
- Removing of complexity in other X3D profiles.

When it comes to networking MPEG-4 Interactive supports "file:" and "http:" protocols using relative URLs.

Following 3D geometry components are supported: Box, cone, cylinder, sphere and IndexedFaceSet which supports 10 vertices per face and 5000 faces. Each face is a polygon and faces should not intersect each other. (see Appendix I) [43].

3.3.2 MPEG-4 Interactive Practice

MPEG-4 Interactive is a binary format allowing simple networking and description of more or less simple 3D geometric shapes.

3.3.3 MPEG-4 Interactive Summary

MPEG-4 Interactive is basically an option to the more complex profiles in X3D intended for use on mobile devices with less computational power.

3.4 SVG

SVG means Scalable Vector Graphics and is an open standard describing two-dimensional graphics in XML. SVG is a W3C recommendation.

Figure 17: W3C logo

3.4.1 SVG Theory

The slogan at the SVG Home Page says: "XML Graphics for the Web" [44] and this first impression tells the reader about the main goal of SVG. As already mentioned SVG describes 2D graphics and applications in XML, compared to X3D and MPEG-4 Interactive enabling 3D representation. As a part of the name implies, "scalable"

means scalable to different resolutions and data sources, "vector" means geometric objects like circles or rectangles, and "graphics" means that both vector graphics and raster graphics (bitmap pictures) can be used in SVG.

SVG 1.1 is the latest official version. SVG also has SVG Mobile Profiles targeted at devices with low level resources, like mobile phones, SVG Basic and SVG Tiny. The different profiles lets graphics rendering perform optimally as well on powerful devices as on mobile devices. The application areas SVG targets are listed in Table 2.

- **Mobile:** SVG Tiny and SVG Basic Profiles. Primarily used making greeting cards, diagrams, animations on mobile devices.
- **Print:** Printing graphics and text, similar to Adobe's PDF-format.
- **Web Applications:** A platform to build graphics upon.
- **Design and Interchange:** High end graphical market.
- **GIS and Mapping:** Geographic Information Systems and maps.
- **Embedded Systems:** Very low resource devices.

Table 2: SVG application areas
[44]

SVG graphics are composed of shapes written using vector graphics, images and text. Many graphical objects may be grouped together, composed into one larger object, and transformation and styling may be applied to these objects [44].

3.4.2 SVG Practice

Since SVG includes the full XML Document Object Model it is allowed to be used within SMIL documents. SVG is supported in web browsers like Mozilla and Opera and Konqueror, Internet Explorer need a plugin an. SVG includes event handlers able to run scripts (Javascript) manipulating the graphics, like changing the shape of a rectangle, etc.

3.4.3 SVG Summary

SVG 1.1 does not support 3D graphics, but designing custom shapes and integrating them in SMIL or other XML based languages (like XHTML) are easy.

3.5 NVSG

As the worlds biggest graphics hardware producer (together with ATI) NVIDIA's NVSG (NVIDIA Scene Graph) deserve to be included in the review section of this Master's Thesis.

3.5.1 NVSG Theory

Figure 18: NVIDIA's and NVSG's logos

NVSG means NVIDIA Scene Graph and is an object oriented programming library (C++) which let developers exploit the latest features of NVIDIA's own graphic processing hardware to create advanced 3D applications.

> "NVIDIA's scene graph is intended to empower developers to quickly create sophisticated and thoroughly modern high-performance 3D graphics applications."
>
> Dan Vivoli, executive vice president of marketing at NVIDIA [45]

NVIDIA has chosen to focus NVSG against representation of 3D scenes and rendering. NVSG is intended to reduce the amount of OpenGL programming and take care of the rendering issues. In the future NVSG will become independent of which render

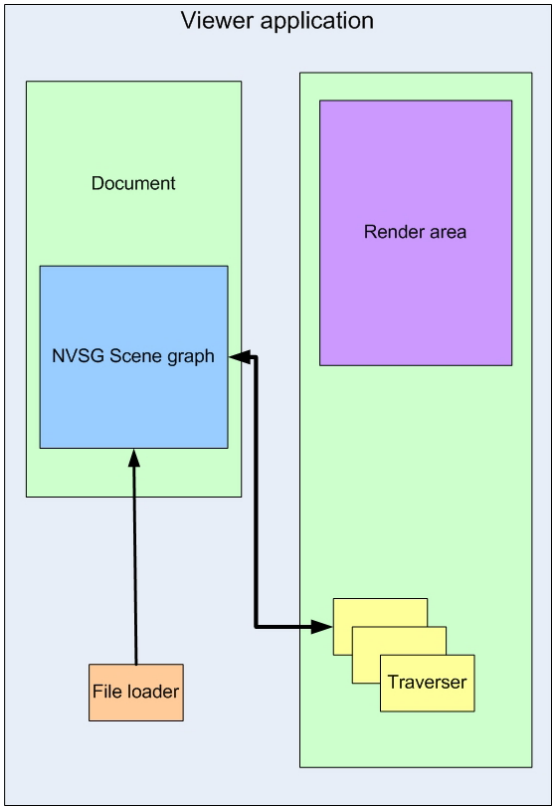


Figure 19: NVSG viewer application

device is underlying. There will be support for OpenGL (which is also present today), Microsoft DirectX (future) and Realtime Raytracer (future) [45].

As seen on Figure 19 there is a "traverser" class with different methods of traversing the document tree as a part of the rendering process, or to apply specific effects to a tree. A user developing a new effect may use the traverser class.

3.5.2 NVSG Practice

NVSG is a scene graph, organized like a scene graph (see Section 2.14). There are many objects which all inherits the "object" node, which then inherits "RCObject". Even though there are different (and more advanced) classes, the idea behind are basically similar to SVG. All nodes may have one or many parents, resulting in that it is easier to apply effects to different parts of the node tree.

There's one issue where SVG and NVSG differs a lot. NVSG features stereo camera,

or in other words 3D. The camera position is shifted to the right and the left from the center position and two different images is rendered to a "right frame buffer" and a "left frame buffer" (Figure 20).

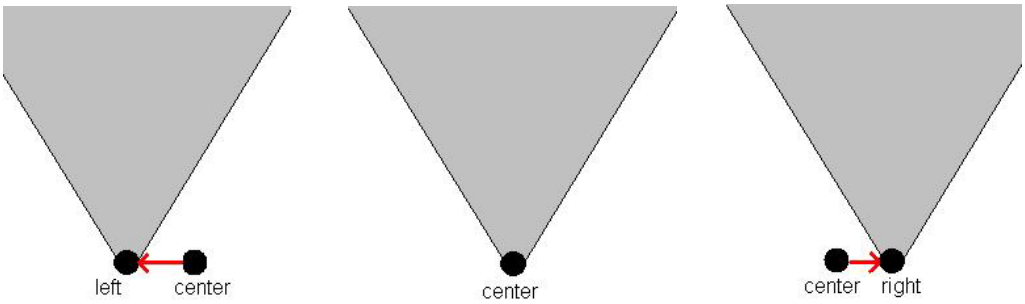


Figure 20: NVSG stereo camera

3.5.3 NVSG Summary

NVSG is a commercial product and favor the development of applications using NVIDIA's own hardware. The goal is probably to make NVIDIA's products sell better. Still, as one of the largest and most competitive developers of graphics hardware, using the tailored NVSG together with NVIDIA's hardware may give a developer an advantage compared to others not using NVSG.

3.6 OSG

Figure 21: OSG logo
[46]

Open Scene Graph is a standard using scene graphs to represent 3D worlds or a scenes. The idea of OSG is being a multi platform, open source graphics library (API) for C++ programmers on top of OpenGL. OSG is developed by the OpenSceneGraph community [47].

3.6.1 OSG Theory

The two basic goals of OSG is to take care of scene management and rendering optimization. The key factors of these two concerns are: Portability, extensibility, scalability and flexibility [46].

A node is represented by the `osg::Node` class. Four of its subclasses are:

- **Osg::Geode:** Geometry node.
- **Osg::Group:** Grouping of child nodes.
- **Osg::PositionAttitudeTransform:** Telling the location of the children of this node.
- **Osg::Drawables:** Renderable subclasses, instances physically seen in the scene.

3.6.2 OSG Practice

OSG is open source, being open source means that the value of the language is not the source code itself but the implementations using the source code. The application developer benefits from it and the open source code naturally evolves.

OSG does not interfere directly with the underlying graphics hardware, the OpenGL layer are in between OSG and the hardware, handling all the communication between these two layers.

OpenGL 2.0 supports 3D. If using a graphics hardware accelerator like Matrox Parhelia Precision SDT [48] with a Planar SD1710 3D LCD screen [49] and polarized glasses, the OSG scene can be viewed in 3D (Figure 22).

3.6.3 OSG Summary

OSG is a graphics library on top of OpenGL. It is suitable for heavy graphical applications like simulators, games and realistic 3D environments. OSG is not written in XML and needs a little more effort to learn than X3D and SVG, OSG is more complicated to integrate with SMIL.

Figure 22: Planar SD170, Matrox Parhelia Precision and polarized glasses
[48]

3.7 SMIL

Figure 23: WC3's SMIL logo
[50]

SMIL is an acronym for Synchronized Multimedia Integration Language and is pronounced *'smile'*. SMIL is built upon XML, it is not really a 3D auto stereoscopic model but its goal is to ease presentation of media. The tags of SMIL are much similar to HTML, but SMIL enables streaming of video, audio, images, text and other types of media. SMIL enables the creator to control timing and placement of the media, this goes for both layout and the timeline. The SMIL language is being developed by the W3C SYMM group [51].

3.7.1 SMIL Theory

SMIL is support the following implementations:

- "Internet or Intranet presentations.
- Slide show presentations.

- Presentations which link to other SMIL files.
- Presentations having control buttons (stop, start, next, ...).
- Defining sequences and duration of multimedia elements.
- Defining position and visibility of multimedia elements.
- Displaying multiple media types such as audio, video, text.
- Displaying multiple files at the same time.
- Displaying files from multiple web servers."

Wikibooks, XML: Managing Data Exchange/SMIL [51]

SMIL is also used in mobile phones, generally MMS uses a subset of SMIL to define how multimedia content are put together in the MMS messages. SMIL is written using tags, a SMIL file (having the extension `*.smil`, `*.sml` or `*.smi`) has this template:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<smil xmlns="http://www.w3.org/SMIL21/Language">
    <head>
        <layout>
            <root-layout id="root_layout"/>
                ...
        </layout>
    </head>
    <body>
        ...
    </body>
</smil>
```

Figure 24: SMIL template

The first line is the XML declaration, this line defines the XML version and the character encoding used to write the XML document. The second line declares that the SMIL 2.0 namespace is used. Defining a namespace reduces confusion on how words are to be interpreted. A name cannot have more than one meaning and two different things cannot have the same name (Appendix C). The layout is described inside the `<layout>...</layout>`, and is divided into different regions using `<region id="..."/>` with parameters describing the placement and size of the current region. Inside the `<head>...</head>` transitions may be described using

<transition id"..."/>. Inside the <body>...</body> the media sources may be added using for instance <video src"..."/> or an audio or image source and assigned to a particular <region ... />. If the video are to be played in parallel with another media source, placing it inside a <par>...</par> or sequentially using <seq>...</seq>.

SMIL has a number of different functional areas, these areas are further divided into modules describing structure, content, actions and attributes associated within the SMIL 2.0 namespace [52].

> "A module is a collection of semantically-related elements. Attributes and their value range may be divided over different modules."
>
> SMIL Modules [52]

Where:

> "An element is a XML representation of a semantic feature. An element has one representation in any given namespace."
>
> SMIL Modules [52]

The SMIL Streaming Media Object Module extends SMIL Media Object Module adding elements and attributes making it possible to describe transport properties of streaming media. When RTP is used in SMIL the client need some initialization parameters to interpret the RTP data that is received. Usually these parameters are described using SDP. To optimize this the Streaming Media Object Module allow the SDP parameters to be merged into the SMIL document. Instead of first retrieving SMIL and then retrieving the SDP referenced in SMIL separately (which will need two round-trips on the network) this will need only one round-trip and offer better performance. The application area may for instance when a person looks at a particular object the client may ask the server to get a better visual representation of that particular object. Of course some other mechanism is needed to observe where the person looks. The Streaming Media Object Module has two primary usage scenarios [22]:

1. **Multicast:** The SDP packet describing the media stream is sent via another protocol.

2. **RTP:** Real-time delivery of media.

3.7.2 SMIL Practice

To play SMIL presentations the user need a SMIL player. The SMIL player may be incorporated into a web-browser like Microsoft Internet Explorer or Mozilla Firefox (only supported by Ambulant's plugin for Mozilla Firefox at the present time). Or the SMIL player may be stand-alone like X-Smiles or Ambulant.

3.7.3 SMIL Summary

SMIL's primary goal is to be a XML language for describing multimedia, the main issues are timing, layout, media sources and visual transitions. SMIL is written using tags quite similar to HTML. The SMIL source are human readable and easy to write using a simple text editor. SMIL is a simple language to understand, but a SMIL document in combination with media sources may be a powerful tool for creating presentations and scenes.

3.8 LASeR

LASeR, Lightweight Application Scene Representation is: *"A binary format for encoding 2D scenes, including vector graphics, and timed modifications of the scene"* [53].

3.8.1 LASeR Theory

LASeR is a part of MPEG-4 Part 20, it's goal is to deliver rich media services to devices like mobile phones which has a low level of computational resources.

- 2D vector graphics.

- Pictures.
- Text.
- Sound.
- Video.

LASeR is based on the Tiny Profile of the Scalable Vector Graphics format (reviewed in Section 3.4). In addition to the SVG Tiny Profile, LASeR is extended with frame accurate synchronization between audio and video and streaming and coding of SVG content.

A LASeR stream can be considered as an initial scene with modifications applied to it during the time of playing the scene. This stream can be downloaded from a streaming server.

3.8.2 LASeR Practice

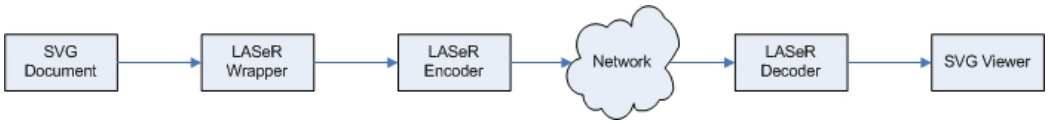


Figure 25: LASeR workflow

The operation of LASeR is described in Figure 25. The LASeR wrapper takes a SVG document and wraps it with SVG and LASeR extensions. The LASeR encoder then encodes a binary LASeR stream ready to be sent over a network. To receive and play a LASeR stream a user needs to decode the stream and play the document with a SVG player.

3.8.3 LASeR Summary

LASeR is not intended to be used on advanced 3D scenes with high bitrate/quality content. The application areas are low resource devices (like 3G mobile phones) needing simple 2D animation and television support.

In the future LASeR will support these features (among others):

1. "Allow an easy conversion from other graphics formats (e.g. BIFS, SMIL/SVG, PDF, Flash, ...).
2. Provide efficient coding suitable for the mobile environment.
3. Allow separate streams for 2D and 3D content.
4. Allow the representation of scalable scenes."

Call for Proposals for
Lightweight Scene Representation [54]

3.9 Section Summary

All the reviewed standards and languages have different ways of practice and different areas of application making it difficult to find any winners or losers among them. Some are suitable for heavy duty applications, others are appropriate for mobile or embedded devices, some are based on binary code which may be faster in some circumstances, others are based on XML which is easier for humans to read and understand and easier to extend with additional functionality. Perhaps the best way to make a conclusion is to present a comparison chart where the actual similarities and differences are easier to grasp.

3.9.1 Comparison Chart

Explanation of the column categories of the comparison chart (Table 3):

- **Name of standard:** Abbreviation of the name of the standard.
- **Develop. organ:** The organ developing the standard.
- **Open:** Access to the source of the standard.
- **Binary:** Not human readable, encoded file format.
- **Active dev.:** "Yes" if the standard is still under active development.
- **3D:** "Yes" if the standard support 3D.

- **Profiled:** If the standard supports modular blocks of functionality. If "Yes", the number of profiles are shown in the parenthesis.
- **Extensible:** Allowing components to be extended to add functionality.
- **Base:** The base theory of the standard.

Name of standard	Develop. organ	Open	Binary	Active dev.	3D	Profiled	Extensible	Base
X3D	Web3D Consortium	Yes	No (XML)	Yes	Yes	Yes (4)	Yes	Scene graph
MPEG-4 Interactive	Web3D Consortium	Yes	Yes	Yes	Yes	No	Yes	Scene graph
SVG	W3C	Yes	No (XML)	Yes	No	Yes	Yes	Vector graphics
NVSG	nVidia	Yes	Yes	Yes	Yes	No	No	Scene graph
OSG	Open Scene Graph Community	Yes	Yes	Yes	Yes	No	Yes	Scene graph
SMIL	W3C	Yes	No (XML)	Yes	No	Yes	Yes	XML
LASeR	MPEG	Yes	Yes	Yes	Yes	No	No	Scene graph

Table 3: Review comparison chart

4 3D, Transparency and Custom Shapes

The next section answers the problems outlined in the description.

> *Propose and demonstrate extensions to SMIL enabling 3D, transparency and custom shapes.*

SMIL does not support custom shapes, transparency or 3D in the current version (SMIL 2.1 of 13th of December 2005 [26]). Including these features in SMIL using SVG and/or X3D may be suitable alternative. Instead of creating entirely new extensions to SMIL, using SVG or X3D saves a lot of work effort. XML based languages, SVG and X3D are easily integrated with SMIL (discussed later).

4.1 Embedding and Referencing from SMIL

In order to build SMIL scenes (like the sketch in Figure 26) using features of SVG and X3D as well, the SVG and X3D documents have to be incorporated with SMIL. Two possible techniques may perform this task, embedding and referencing are presented and demonstrated in this subsection.

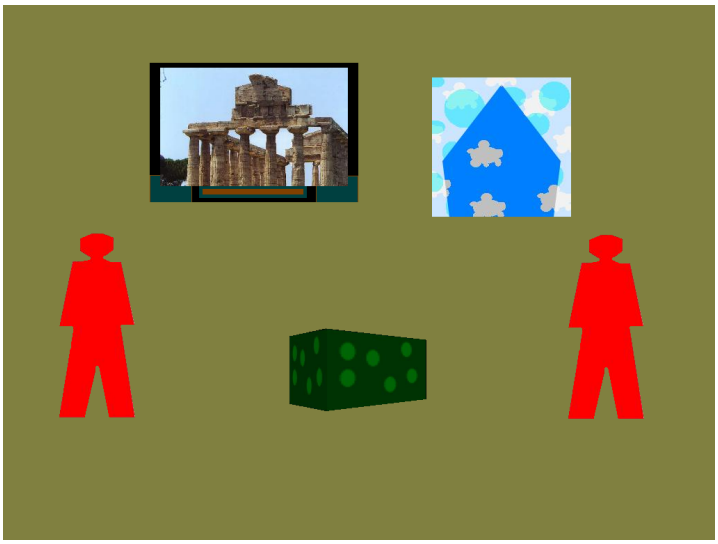


Figure 26: Sketch of the SMIL document with SVG and X3D

4.1.1 Embedding SVG and X3D Inline

Instead of creating new namespaces to give SMIL new features, it's possible to integrate SVG and/or X3D into SMIL. This way the features of SVG and X3D is integrated into SMIL.

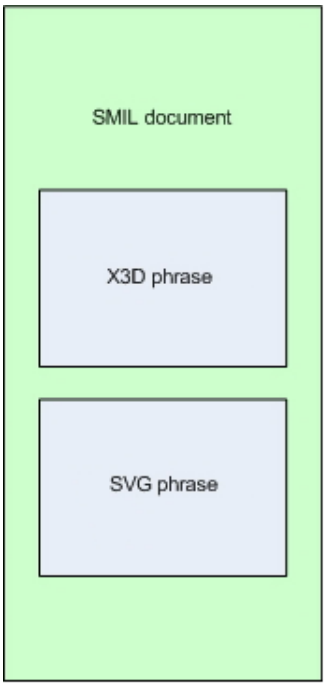


Figure 27: SVG and X3D embedded in SMIL

Embedding or embedding inline means that i.e. SVG or X3D documents are included inside the parent SMIL document. To play SMIL+SVG, SMIL+X3D or SMIL+X3D/SVG the player needs to support all three XML formats and supported media types. The currently available version of Ambulant Player only support SMIL, not SVG or X3D.

Documents embedded inline are located inside the `<body> ... </body>` tags of a parent SMIL document. Embedded documents are equal to other media types included in the SMIL document layout. An embedded SVG document must be encapsulated with the `<svg> ... </svg>` tags, shown on Figure 31. A larger example is demonstrated in Appendix A.4.1.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<smil xmlns="http://www.w3.org/SMIL20/Language">
    <head>
        <layout>
        <root-layout id="root_layout"/>
            ...
        </layout>
    </head>
    <body>
        <svg>
            ...
        </svg>
    </body>
</smil>
```

Figure 28: Embedded SVG document phrase

4.1.2 Demonstration of Embedded X3D and SVG

The possible scene introduced in Figure 26 may be created using embedded SVG and X3D documents. SVG is used to define polygon areas (the persons), and X3D takes care of 3D figures (the table). The flatscreen TV is included with regular SMIL tags (not shown). The complete SMIL document source is shown in Appendix A.4.1.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 2.0//EN"
    "http://www.w3.org/2001/SMIL20/SMIL20.dtd">
<smil ...  >
<!- SMIL, SVG and X3D namespaces -> <head>
<layout>
    ...
</layout>
</head>
<body>
<par>
<svg:svg region="svgpersona_region">
    ...
</svg:svg>
<svg:svg region="svgpersonb_region">
    ...
</svg:svg>
<xsd:X3D region="boxtable_region" version='3.1'
    profile='Immersive' >
    ...
</xsd:X3D>
...  <svg:svg region="window_region">
    ...
</svg:svg>
</par>
</body>
</smil>
```

Figure 29: Embedded documents

4.1.3 Referencing External SVG/X3D

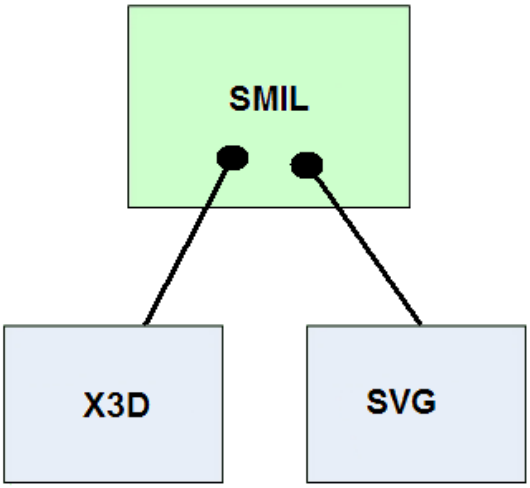


Figure 30: External SVG and X3D referenced from SMIL

An alternative to embed SVG or X3D documents is to reference external documents. Other SMIL documents may be referenced as well as relevant types of XML documents. In this case (Figure 31) the SVG part is an referenced external SVG document (`referencedsvg.svg`).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<smil xmlns="http://www.w3.org/SMIL20/Language">
    <head>
        <layout>
        <root-layout id="root_layout"/>
        <region id="referenceddocument" ...  />
        </layout>
    </head>
    <body>
        <switch>
            <ref type="image/svg+xml" src="referencedsvg.svg"
                 region="referenceddocument" />
            <img src="image.jpg" region="referenceddocument" />
        </switch>
    </body>
</smil>
```

Figure 31: Referenced SVG document

The `<switch> ... </switch>` tags is used to enable the player to display an

optional image if it does not have the ability to play SVG documents.

4.1.4 Demonstration of Externally Referenced Documents

The same scene may also be created with references to external SVG and X3D documents. A SVG document is referenced from inside the `<ref ... />` tag of the `<body>` of the SMIL document (Figure 32). Figure 32 is extracted from a complete SMIL document, referencing external SVG and X3D documents, the source of this SMIL document is shown in Appendix A.4.2.

```
<ref type="image/svg+xml" src="persona.svg"
    region="svgpersona_region"/>
...
<ref type="image/svg+xml" src="personb.svg"
    region="svgpersonb_region"/>
...
<ref type="model/x3d+xml" src="boxtable.x3d"
    region="boxtable_region"/>
...
<ref type="image/svg+xml" src="window.svg"
    region="window_region"/>
```

Figure 32: References to external documents

`Persona.svg`, `personb.svg`, `boxtable.x3d` and `window.svg` are shown in Appendix A.

4.2 SVG Custom Shapes

A "custom shape" in this context represent the situation where a region or an object are not only square or rectangular, but shaped like a polygon, circle, ellipse or whatever the vector based representation allow. SVG support the following shapes/graphical elements: Paths, Rectangles, Circles, Ellipses, Lines, Polylines and Polygons [55].

Among these shapes, the Polygon can be considered the most custom shape. It is defined by a closed shape put together by a number of straight lines. An example of a polygon is shown in Figure 33, its corresponding SVG-file is shown in Figure 34.

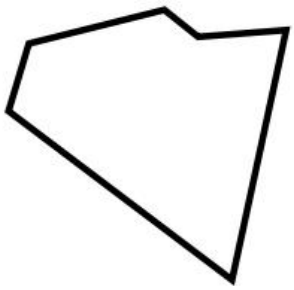


Figure 33: Simple polygon example

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
         "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="6cm" height="6cm" viewBox="0 0 500 500"
         xmlns="http://www.w3.org/2000/svg" version="1.1">
     <desc>
         Simple polygon example
     </desc>
     <polygon fill="white" stroke="black" stroke-width="10"
         points="100,100 70,200 400,450 480,80
         350,90 300,50"/>
</svg>
```

Figure 34: SVG file of the simple polygon example

The image of Figure 35 (`test.png`) may be clipped into the polygon of Figure 33, using the SVG container element *ClipPath* [55]:

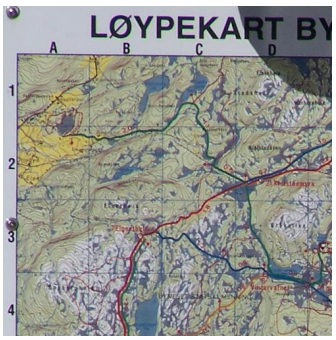


Figure 35: Square image

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
    "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="500" height="500" viewBox="0 0 500 500"
    xmlns="http://www.w3.org/2000/svg" version="1.1"
    xmlns:xlink="http://www.w3.org/1999/xlink">
    <desc>
        A square image (test.png) is clipped into the
        polygon (klippsti)
    </desc>
    <defs>
        <clipPath id="klippsti">
            <polygon points="100,100 70,200 400,450
            480,80 350,90 300,50"/>
        </clipPath>
    </defs>
    <g clip-path="url(#klippsti)" visibility="visible">
        <g id="klipp">
            <image x="0" y="0" width="500" height="500"
                xlink:href="test.png"/>
        </g>
    </g>
</svg>
```

Figure 36: SVG file with ClipPath

ClipPath's are regions where an image are applied inside a shape described by path coordinates, parts of the image appearing outside the clipping path are cut away. The alpha value of the image are zero outside the clipping path (fully transparent), and one (fully opaque) or between zero and one (semi transparent) on the inside. The result of the SVG file with ClipPath (Figure 36) is shown in Figure 37.

Figure 37: ClipPath result

4.2.1 Demonstration of Custom Shapes in SMIL+SVG

Arbitrary polygon shaped regions playing referenced video streams are not feasible using SMIL only. To implement this feature into a scene a combination of SMIL+SVG can be used. The idea described in the following example use a SMIL document as a base document with a reference to an external SVG document taking care of the polygon issue. The visual background of this scene is described using a second SMIL document and the media stream played inside the polygon is a `rtps://` referenced via an URI.

- **Smilbase.smil**: Structural Layout document with GUI buttons, etc (Figure 40).
- **Svgshapes.svg**: Polygon shapes with transparency and external media streams, an embedded SMIL document representing the background image, etc (Figure 41).

To be able to play this demonstration the following requirements have to be met:

- SVG streaming video enhancements. This feature is not available yet (under consideration by the SVG working group: *"The SVG working group is considering streaming enhancements to the SVG language"* [56]).
- A player application supporting both SMIL, SVG and streaming video.

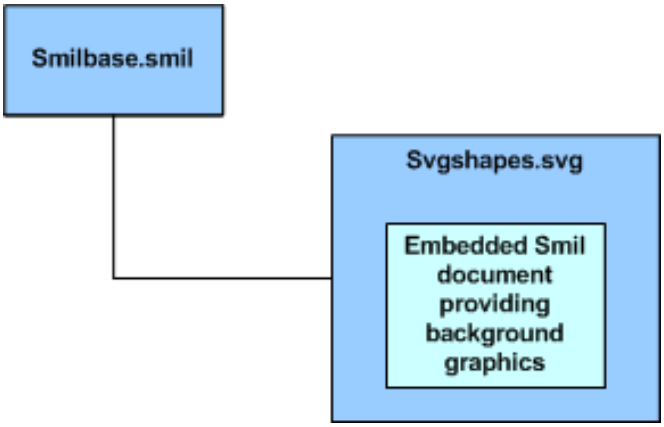


Figure 38: Document structure

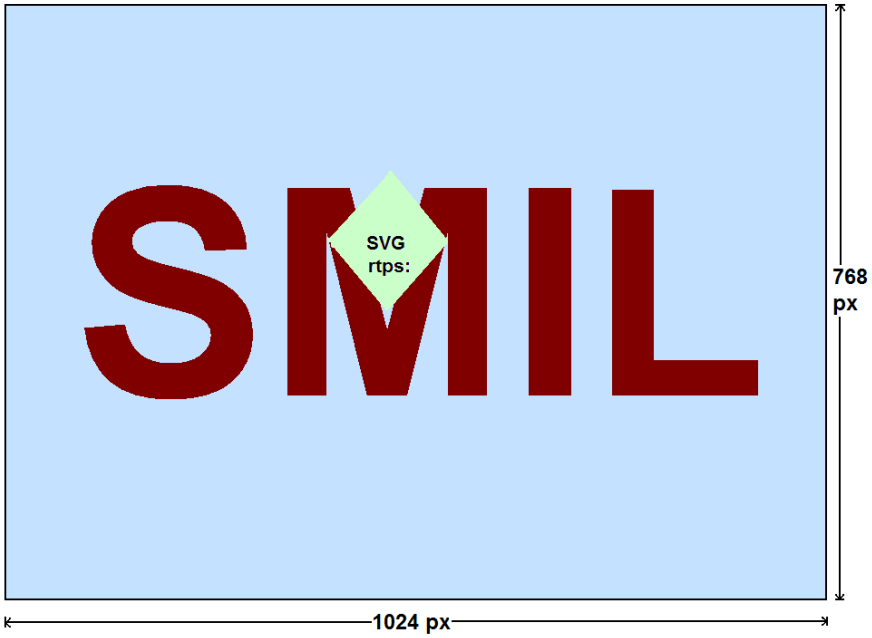


Figure 39: Result sketch

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 2.0//EN"
    "http://www.w3.org/2001/SMIL20/SMIL20.dtd">
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
<head>
    <layout>
        <root-layout width="1024" height="768" />
        <region id="svgshapes" left="40%" top="40%"
                fit="fill" />
    </layout>
</head>
    <body>
        <switch>
            <ref type="image/svg+xml"
                src="svgshapes.svg"
                region="svgshapes" />
            <img src="erroralternative.png"
                region="svgshapes" />
        </switch>
    </body>
</smil>
```

Figure 40: Smilbase.smil

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
    "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="1024" height="768" viewBox="0 0 500 500"
    xmlns="http://www.w3.org/2000/svg" version="1.1"
    xmlns:xlink="http://www.w3.org/1999/xlink">
    xmlns:smil="http://www.w3.org/2001/SMIL20/Language"
    <desc> SVG 1.2 video played in clippath with smil
        document background SVG </desc>
    <defs>
    <clipPath id="rute">
        <polygon points="0,200 200,0 400,200
            200,400"/>
    </clipPath>
    <g>
        <smil:smil>
            <smil:head>
            <smil:layout>
            <smil:root-layout width="1024"
                height="768" />
            <smil:region id="background"
                left="0%" top="0%"
                fit="fill" />
            </smil:layout>
            </smil:head>
            <smil:body>
            <smil:img src="room.jpg"
                region="background" />
            </smil:body>
        </smil:smil>
    </g>
    <g clip-path="url(#rute)" visibility="visible"
        opacity=".9" >
        <g id="klipp">
            <video xlink:href="rtps:/..." width="400"
                height="400" x="0" y="0"
                repeatCount="indefinite"/>
        </g>
    </g>
</svg>
```

Figure 41: Svgshapes.svg

4.3 Transparency

Transparency allow smoother overlapping of regions or partially see-through regions, a background sky or a city may be visible outside a window or through lucid curtains. An example from television is the partially see-through logo of a TV station logo displayed in one of the corners of the screen.

4.3.1 Transparency Theory

Video frames are opaque by default. A transparent image or video clip has the property that, at some degree, it is possible to see through it. Information about the transparency of a bitmap picture is stored in the picture's alpha channel. The alpha channel defines transparent areas. In fact, the alpha channel is a separate channel beside the more known RGB-channels (the visible colors) providing information about transparency without disturbing the picture's colors. The RGBA abbreviation (Red Green Blue Alpha) is often used if there is an alpha channel in addition to the color channels . Each pixel of the picture then has it's own RGBA values.

There are different methods and names of transparent or translucent images, some of them are listed below:

- **Alpha Channel:** A channel defining transparent areas of a clip.
- **Mask:** Another word sometimes is used for alpha channel.
- **Matte:** A file or a channel defining transparent areas of a clip. Often used when there is no alpha channel or when there is a channel defining transparency better than the alpha channel.
- **Keying:** Defining transparency by a particular color or luminance key.
- **Opacity:** Opaque is the opposite of transparency.
  Transparency = 1 - opacity.

Adobe Premiere Pro, User Guide for Windows [57]

Transparency is also called alpha blending. The value of the resulting color when color `Value1` with an alpha value of `Alpha` is drawn over a background of color `Value0` is given by:

$$Value = Value0(1.0 - Alpha) + Value1(Alpha)[58]$$

4.3.2 Transparency in SMIL

SMIL 2.0 is not really supporting transparency, the only kind of support is transparency indicated by the *z-index*, controlling the stacking of one region upon other regions. In other words, a region is either 0 or 100 percent transparent.

```
<layout>
    <root-layout id="RootRegion" title="RootRegion"
        width="600" height="400"/>
    <region id="region_back" title="Back" top="0"
        left="0" width="600" height="400" z-index="1"/>
    <region id="region_front" title="Front" top="50"
        left="50" width="100" height="100" z-index="2"/>
</layout>
```

Figure 42: Transparency using the z-index

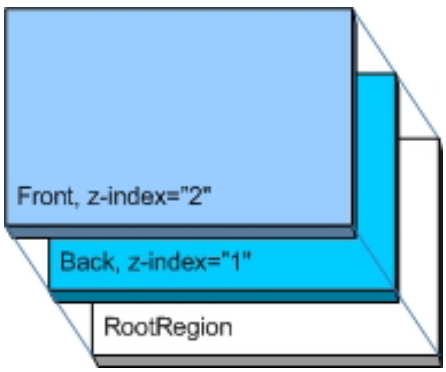


Figure 43: Z-index

A high `z-index` indicates that the object with this attribute will appear on top of others with a lower `z-index`. In this example the "Front" region are on top of the "Back" region. Generally spoken, the `z-index` control the stacking order of overlapping regions

4.3.3 Transparency in RealPlayer

Figure 44: RealPlayer

RealPlayer's SMIL extension namespaces supports transparency, transparency is easily applied for GIF, JPEG, or PNG images and Flash animations, but more difficult for streaming video. A negative thing is that the transparency attributes are only recognizable by the RealOnePlayer and other SMIL players may not recognize these attributes [59].

RealPlayer support a number of attributes used to apply transparency. These are listed in Table 4 and shortly describes the transparency features RealPlayer adds to SMIL [59].

Attribute	Value	Function
`rn:backgroundOpacity`	percentage	Adjusts background transparency.
`bgcolor`	nnnnnn	Substitutes color for transparency.
`rn:chromaKey`	color_value	Turns selected color transparent.
`rn:chromaKeyOpacity`	percentage	Adds opacity to chromaKey.
`rn:chromaKeyTolerance`	color_value	Widens range of chromaKey.
`rn:mediaOpacity`	percentage	Makes opaque colors transparent.

Table 4: RealPlayer's transparency features
[59]

As previously mentioned these attributes are only applicable to GIF, JPEG, or PNG and Flash graphics played with Real's own player.

4.3.4 Transparency in X3D

X3D is based on the concept of a scene graph, meaning everything represented in a scene is some kind of node in a tree (in a directed acyclic graph). There are a number of

nodes in X3D having textures: `Background`, `ImageTexture`, `MovieTexture`, `MultiTexture`, and `PixelTexture`.

Transparency may be applied to these nodes with texture maps or 2D images containing an array of colors describing the texture (i.e. RGB and RGBA), including video. To describe X3D transparency, knowledge about the base types of abstract nodes are needed (see Appendix K for more details).

- **X3DTextureCoordinateNode:** Base type node specifying geometric properties.
- **X3DTextureNode:** Base type node for all nodes specifying textures.
- **X3DTexture2DNode:** Base type node for all nodes specifying 2D textures.

A supported video format is MPEG1-Video with video and audio from a source specified by an `url`. The MPEG1-Video are in fact a time dependent texture map, with the same transparency properties as a RGBA image texture map [60].

```
MovieTexture :  X3DTexture2DNode, X3DSoundSourceNode,
    X3DUrlObject{
    ...
}
```

Figure 45: A MovieTexture is constructed from abstract nodes
Web3D Consortium [60]

The MovieTexture node (see Figure 45) defines a texture map contained in a movie file describing how to control the movie and how to apply mapping of the textures. The `MovieTexture` or `ImageTexture` can be used to "dress" the sides of a 3D object with one or many images and/or video streams.

4.3.5 Transparency With SVG

SVG incorporates an "Opacity Attribute Module" [61] which apply opacity to objects and groups. The object/group is first rendered into an RGBA image (not displayed during the rendering process), and thereafter the opacity settings specify how this rendered RGBA image is blended upon a background [55]. Opacity can be applied to any container element, including ClipPaths (described in Section 4.2).

The opacity settings apply to the whole object. The level of transparency may obtain any value between 0.0 and 1.0 (Figure 46) [55].

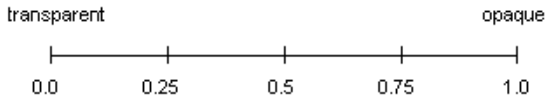


Figure 46: Transparency/opacity

To describe use of transparency in SVG the example SVG file in Figure 36 is continued in Figure 47.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
    "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="500" height="500" viewBox="0 0 500 500"
    xmlns="http://www.w3.org/2000/svg" version="1.1"
    xmlns:xlink="http://www.w3.org/1999/xlink">
    <desc>
        A square image (test.png) is clipped into the
        polygon (klippsti)
    </desc>
    <!- Background blue circle ->
    <circle cx="290" cy="220" r="100" fill="blue"/>
    <defs>
        <clipPath id="klippsti">
            <polygon points="100,100 70,200 400,450
                480,80 350,90 300,50"/>
        </clipPath>
    </defs>
    <g clip-path="url(#klippsti)" visibility="visible"
        opacity=".3">
        <g id="klipp">
            <image x="0" y="0" width="500" height="500"
                xlink:href="test.png"/>
        </g>
    </g>
</svg>
```

Figure 47: SVG file with ClipPath and opacity

All that needs to be added is **`opacity=".3"`** inside the properties of the <g> tag. "*The 'g' element is a container element for grouping together related graphics elements* [61]". The SVG file in Figure 47 shows opacity of 30% (transparency of 70%)

and the white background shines through the filled blue circle and the clipped image (`test.png`) The result are presented in Figure 48.

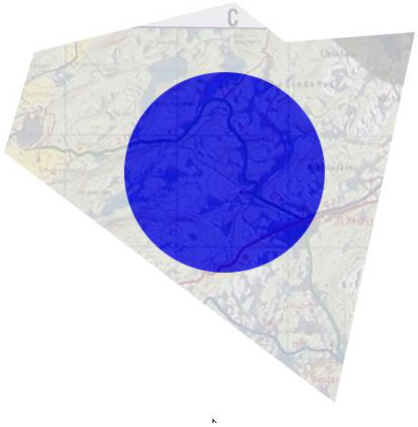


Figure 48: A 70% transparent image of a map upon a blue filled circle

4.4 3D with X3D

As the name implies X3D aims at representing 3D graphics. X3D may create synthetic textures or use referenced images or video streams from local or internet sources.

4.4.1 X3D Shapes and Textures

X3D allow a number of different shapes. Figure 51 demonstrate the use of a `<Sphere>` using the image `test.png` as texture.

If `<Sphere/>` in the X3D file (Figure 51) is replaced with `<Box/>` the shape of the 3D object is changed from a globe to a cube. The result may be seen in Figure 50.

```
<X3D version='3.1' profile='Immersive'
    xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance'
    xsd:noNamespaceSchemaLocation='http://www.web3d.org/
        specifications/x3d-3.1.xsd'>
    <Scene>
        <Transform>
        <NavigationInfo headlight='false'
            avatarSize='0.80 2.6 0.99'
            type='ANY'/>
        <SpotLight/>
        <Transform translation='0.0 0.0 0.0'
            rotation='0.5 0.5 0.5 0.5'>
            <Shape>
                <Sphere/>
                <Appearance>
                    <ImageTexture url='test.png'/>
                </Appearance>
            </Shape>
        </Transform>
        </Transform>
    </Scene>
</X3D>
```

Figure 49: X3D sphere with ImageTexture

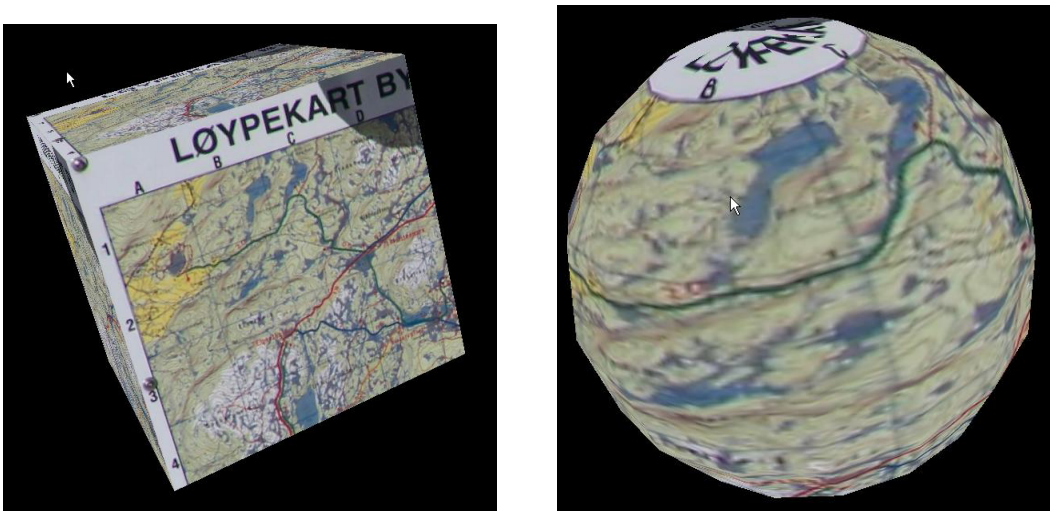


Figure 50: X3D <Box/> and <Sphere/> with test.png as texture

4.4.2 Video Stream Played on 3D Object

To have video streams played as textures on 3D objects, X3D uses the <MovieTexture> node (see Appendix K.2.2).

```
<X3D version='3.1' profile='Immersive'
    xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance'
    xsd:noNamespaceSchemaLocation='http://www.web3d.org/
        specifications/x3d-3.1.xsd'>
<Scene>
<Group>
    <Transform>
    <Shape>
        <Box/>
        <Appearance>
            <MovieTexture loop='true' url='
                "trikse1.mpg" '/>
        </Appearance>
    </Shape>
    </Transform>
</Group>
</Scene>
</X3D>
```

Figure 51: X3D box with as MovieTexture

The Xj3D player used to capture the images of Figure 50 does not fully support the MovieTexture node [62]. Figure 52 is captured using another player, Octaga [63].

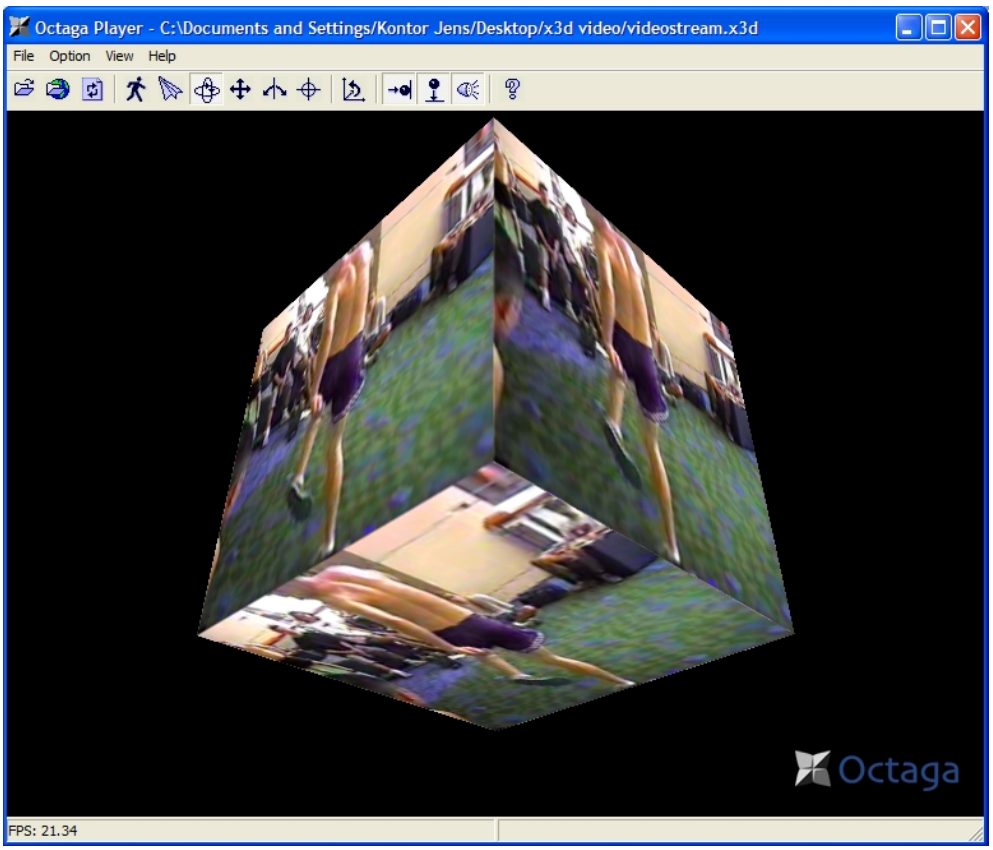


Figure 52: X3D box played in Octaga Player

4.5 Section Summary

This section has presented techniques used to represent 3D, transparency and custom shape features that may be interesting and useful to employ in a distributed 3D scene. SVG and X3D are, because of the XML base, both suitable use teamed with SMIL. The transparency and custom shape (`ClipPath`) features of SVG and the 3D options given by X3D are possibly the most interesting features to use while creating distributed scenes. How to embed or reference SVG and X3D documents from SMIL documents are discussed in a later section.

5 Composition of Distributed Scenes

To be considered distributed, scene descriptions (SMIL documents) have to be exchanged between users which are taking part in the same scene. The scene descriptions must allow different levels of quality of objects in the scene, meaning the bitrate of audio and video streams must be adaptive according to preferences given by users or available bandwidth of the network. 3D, transparency and custom shapes presented in Section 4 may also be included.

This section continue the discussion of which of the technologies and protocols presented in Technology Background (Section 2) are preferable as fundamental and underlying technologies of a distributed system. The end-user applications are not of the greatest importance in this section, but using Ambulant Player and Gaim helps describing the system. Shortly written this section presents a problem study with proposed solutions due to scene composition and exchange of scene descriptions. This Section of the Master's Thesis will answer the following questions:

1. Designing what?
2. Defining a requirement specification.
3. How may distributed scenes be composed?
4. Does the proposed system conform to the requirement specification

5.1 Designing What?

A possible scene, related to Distributed Multimedia Plays [1] are presented as an idea of how a scenario may look like. In this possible scene one of the participants keep track of the master SMIL document and passes this document to the other participant(s) when the scene session is started. This way of organizing a scene may not be the most fail safe and effective way, but how the scenes are composed and how data is exchanged between participants do no differ very much if this organization is rearranged. The participants submit changes relating to their status of participation or changes of the object role they play to all the other participants during the time of participation. Each user update their own version of the master SMIL document with this new information.

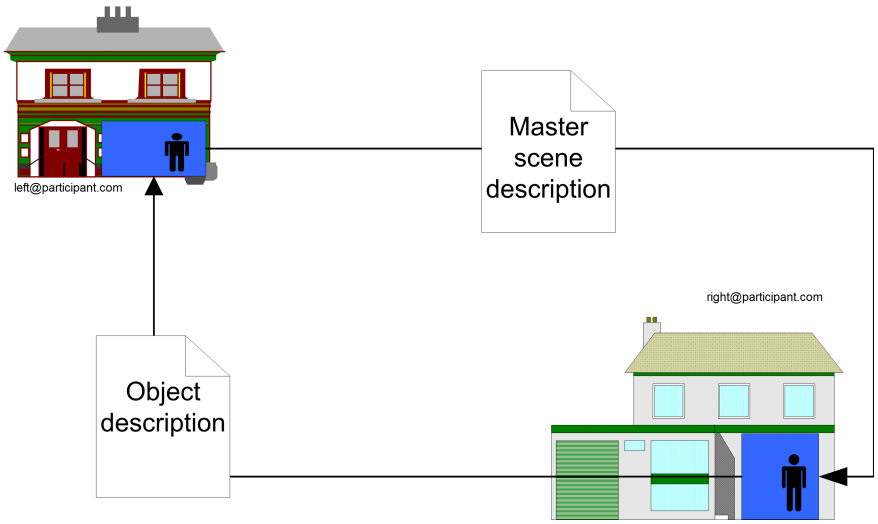


Figure 53: Initial SMIL document exchange

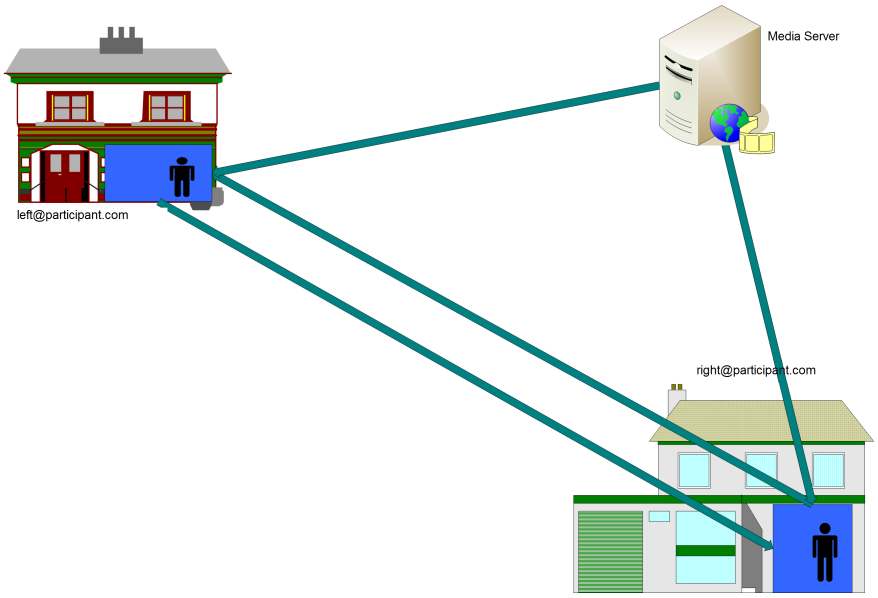


Figure 54: Media streams

Media streams originates from each of the participants (and may also originate from a third party server), the media streams correspond to descriptions in the master SMIL document.

Object roles are played by participants in real-time. The blue rooms at Figure 53 and Figure 54 are so-called "Multimedia Home Spaces" [64] (some kind of home studios). The blue walls makes it easier to recognize the shape of the person(s) in the room, this

is convenient in the process of creating polygon shaped objects to be embedded into the SMIL scene description. The polygon will be shaped so it fits the shape of the person's (or object's) video stream that is extracted from the blue background walls. The network is a high-capacity fiber network connecting different "Multimedia Home Spaces" and additional servers. The quality of the scenes approaches "near natural", implicating high spatial and temporal resolution of objects in the scene.

Contents of a scene may be synthetic and live media streams may comprise 2D or 3D video, images, text and sound. The SMIL document (Master scene description) describes the layout of a scene, the timing of objects and sound. Whether objects appear in parallel or serial are described in the SMIL document. The object shapes are getting more realistic using other embedded or referenced XML-based languages with the SMIL document (discussed in Section 4.1.1).

5.1.1 User Interaction Use Case Diagrams

Use case diagrams describe the interaction between the user and the system to give a concise idea what the user expect from the system. The level of detail of the use cases are rather low, the purpose is to illustrate the basic idea, not the whole system. The use cases depicts an external view to the system [65].

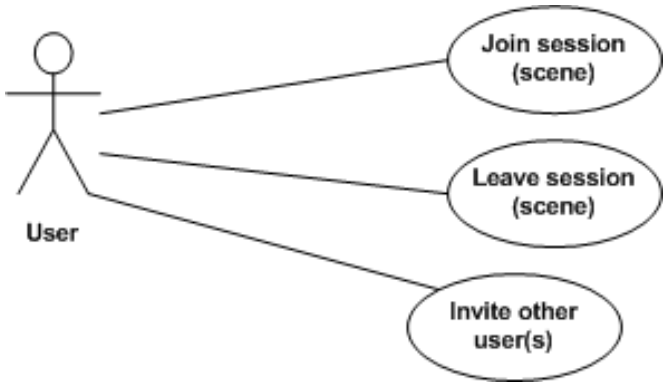


Figure 55: Use case: Join, leave, invite

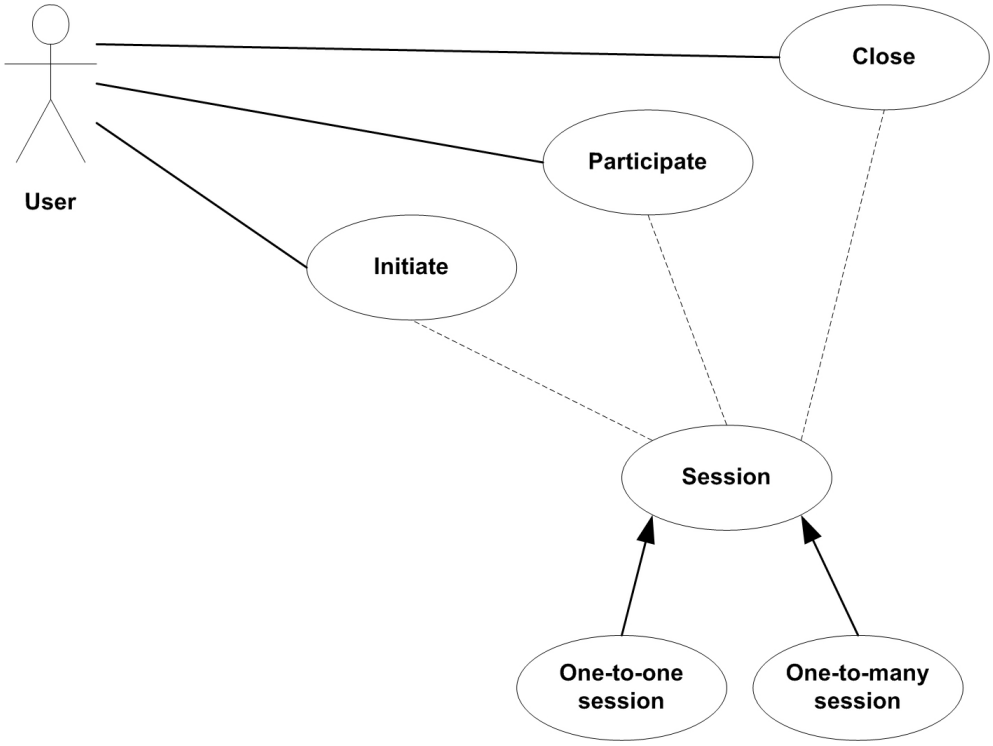


Figure 56: Use case: Initiate, participate and close a session

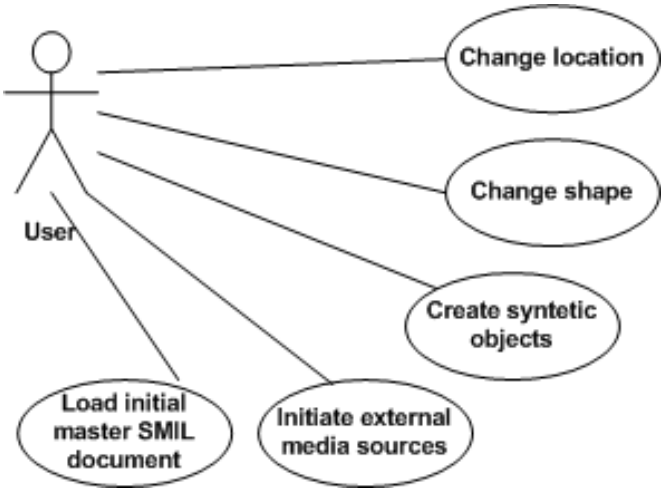


Figure 57: Use case: Change location, change shape, create objects

5.1.2 System Parts Interaction Use Case Diagrams

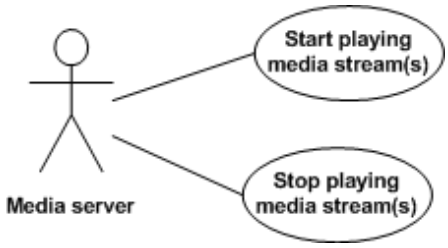


Figure 58: Use case: Media server

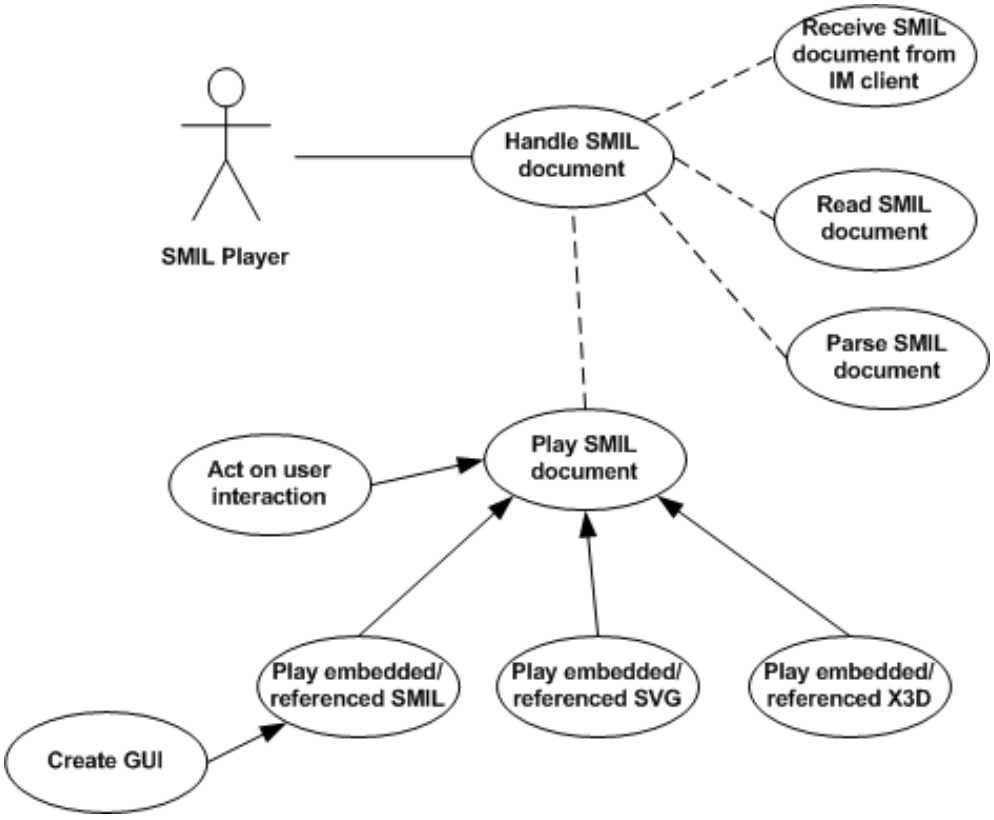


Figure 59: Use case: SMIL player

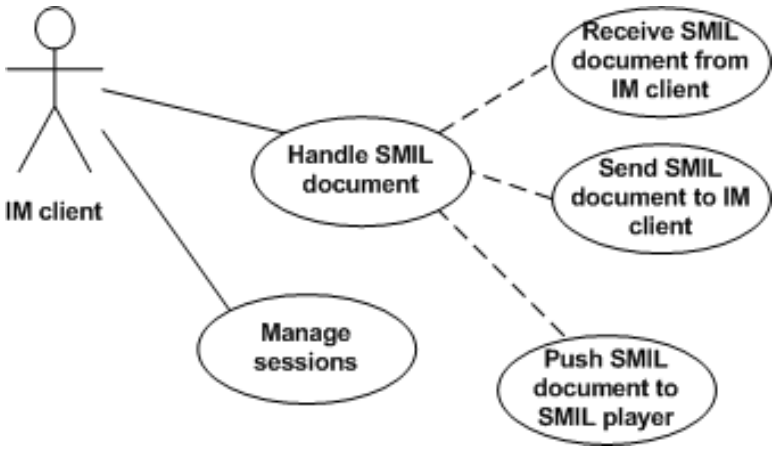


Figure 60: Use case: IM client

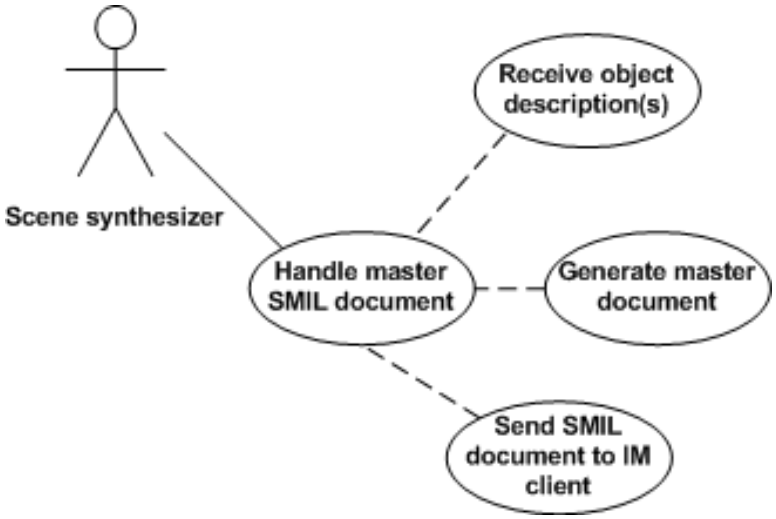


Figure 61: Use case: Scene synthesizer

5.1.3 Deployment Diagram

The deployment diagram show the hardware and the software of the system and the structure of how the parts are connected.

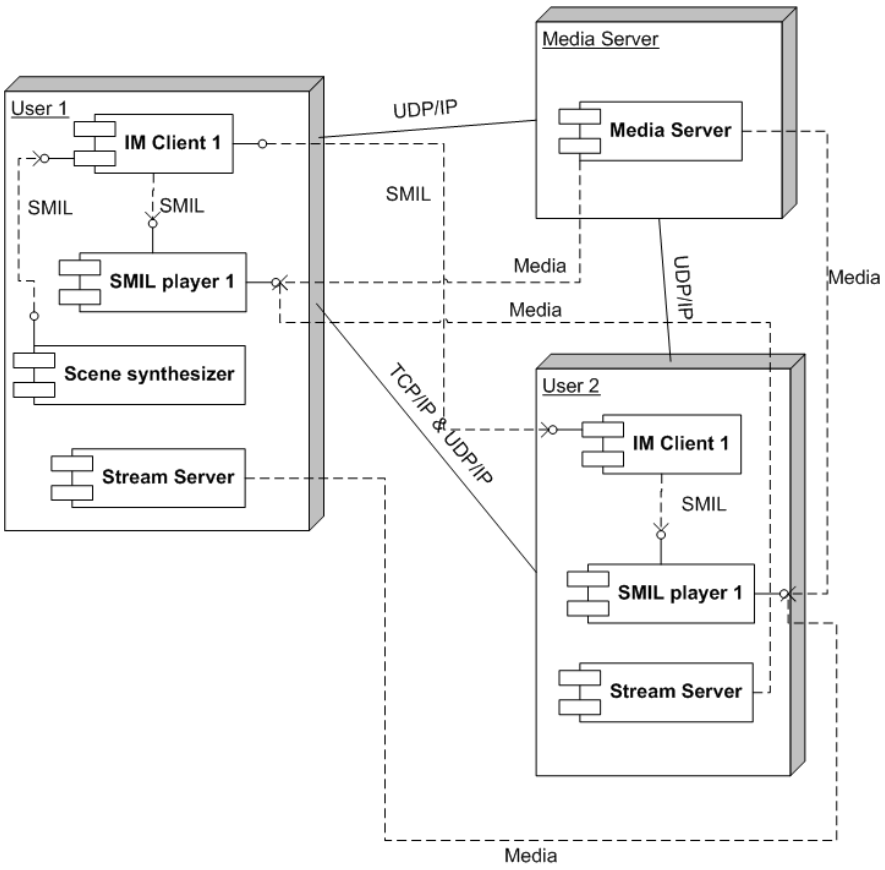


Figure 62: Deployment diagram

5.1.4 Sequence Diagram

Generally spoken UML sequence diagrams show how sequences of messages are passed between objects in the use cases. Figure 63 identify the messages sent when "User 1" loads an initial master SMIL document and invites "User 2" to take part in the scene. The number "1" means that the physical location is at the same place as "User 1" and "2" means that the physical location at with "User 2's" location.

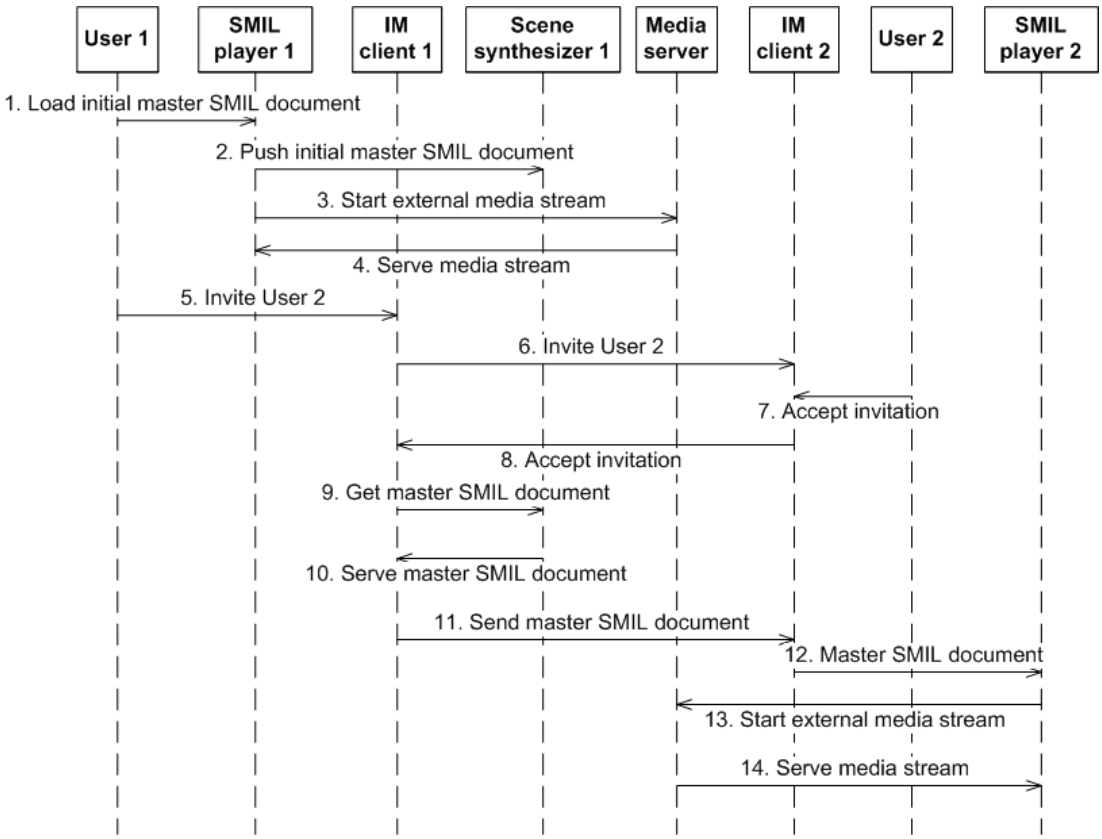


Figure 63: Sequence diagram: User loads scene and invites another participant

5.2 Requirement Specification

The requirement specification presented are somewhat hypothetical, but looking at the Possible Scene (5.1) in combination with High-QoS constraints and specifications mentioned by Leif Arne Rønningen in "Distributed Multimedia Plays Virtual Dinner" [1] may give some ideas of a potential requirement specification.

End-to-end delays have to be guaranteed and limited only by delay introduced by the time of propagation. Scene resolution have to be adaptable and scaled in real-time to avoid traffic overload and package dropping [64].

1. Fiber optic, packet switched network. The end-to end delay in packet switched network can be guaranteed but the resolution of objects in the SMIL scene must be adaptive.

2. Natural, real-life/3D auto-stereoscopic. Requirements due to the perception of

the participants of a scene. With temporal resolution at least 120 Hz [1].

3. Properties of stream communication: End-to-end time delay and latency of media streams less than 10-20 *ms* because of human perception [1], meaning creation/update and exchange of SMIL documents must be even faster.
4. Use RTP/UDP/IP to exchange media data.
5. Use extensions to SIP to enable reliable exchange of scene descriptions.
6. Multiparty sessions with more than two users should be allowed.
7. SMIL documents should be passed in a P2P manner between participants of a session.

5.3 How to Compose Distributed Scenes?

How to compose a distributed scene with video streams originating from different locations? The following sections present how to compose an entire scene using SMIL, how participants may report behavior and parameters to master SMIL document and how to allow adaptive datarate of objects in a distributed scene using rtpmap.

5.3.1 SMIL Over MSRP

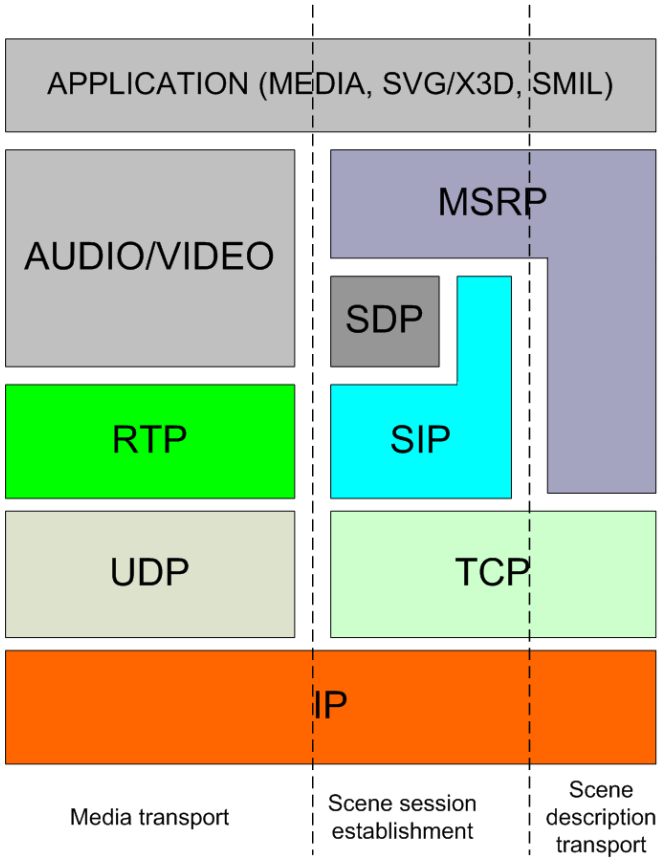


Figure 64: Protocol Layer Stack

Figure 64 shows the protocol stack when the transport of media is done using RTP/UDP/IP, negotiation of MSRP sessions are done using SDP/SIP/TCP/IP and the exchange of scenes descriptions are done using MSRP/TCP/IP. The choice of MSRP exclude UDP as a transport protocol for scene descriptions (MSRP requires a reliable transport protocol).

MSRP gives a number of advantages compared to the SIP message method (listed in Section 2.4). Shortly written MSRP sessions are negotiated via SDP "offers" and corresponding "answers". The exchanges of "offer"/"answer" are done using SIP messages. MSRP may transfer instant messages and other MIME content over a transport protocol (TCP is the only transport protocol supported by MSRP at the present time), including SMIL documents. Set up of a MSRP session using SIP is quite similar to establishing a "normal" media session (see Figure 2). A SIP message with the following

SDP media content line are sent as an offer. The port number is ignored, and replaced with the MSRP address.

```
m=message 6666 MSRP *
```

Figure 65: SDP content
[9]

As Section 2.11.1 described, Gaim only support the SIP Message Method, not SIMPLE IM sessions where messages are transferred P2P between participants. Exchange of SMIL documents using the Message Session Relay Protocol (MSRP) enables P2P transfer of documents.

If the *left@participant.com* in Figure 53 wants to exchange a SMIL document to *right@participant.com* he must first negotiate a MSRP session.

1. *Left@participant.com* sends an "offer" (Figure 66).
2. *Right@participant.com* "answers" (Figure 67).
3. *Left@participant.com* sends ACK to *right@participant.com* (Figure 68).

```
INVITE sip:right@participant.com SIP/2.0
To:  <sip:right@participant.com>
From:  <sip:left@participant.com>;tag=001
Call-ID: 1000ok1000
Content-Type:  application/sdp
c=IN IP4 www.participant.com
m=message 6666 TCP/MSRP *
a=accept-types:text/plain
a=path:msrp://www.participant.com:6666/aghtR12gbh;tcp
```

Figure 66: *Left@participant.com* sending an "offer"

Now, the MSRP session is established and *left@participant.com* can send the SMIL document to *right@participant.com* by telling that Content-type of the MRSP message is `application/smil` (Figure 69).

Right@participant.com sends an OK message back to *left@participant.com* (Figure: 70) when he has received the SEND message. If the SEND message are containing a very large SMIL document, it may be partitioned into smaller messages called "chunks" [9].

```
SIP/2.0 200 OK
To:  <sip:left@participant.com>;tag=101aa
From:  <sip:right@participant.com>;tag=001
Call-ID: 1000ok1000
Content-Type:  application/sdp
c=IN IP4 www.participant.com
m=message 12345 TCP/MSRP *
a=accept-types:text/plain
a=path:msrp://www.participant.com:12345/fgh53gds23;tcp
```

Figure 67: *Right@participant.com* sending an "answer" back to *left@participant.com*

```
ACK sip:right@participant.com SIP/2.0
To:  <sip:right@participant.com>;tag=101aa
From:  <sip:left@participant.com>;tag=001
Call-ID: 1000ok1000
```

Figure 68: SIP/SDP ACK

```
MSRP x1234567 SEND
To-Path:  msrp://www.participant.com:12345/fgh53gds23;tcp
From-Path:  msrp://www.participant.com:6666/aghtR12gbh;tcp
Message-ID: 77777
Byte-Range:  y/x
Content-Type:  application/smil

The content of this message is the first
"y" bytes of a total of "x" bytes of the
SMIL document
-----x1234567$
```

Figure 69: MSRP SEND

```
MSRP x1234567 200 OK
To-Path:  msrp://www.participant.com:6666/aghtR12gbh;tcp
From-Path:  msrp://www.participant.com:12345/fgh53gds23;tcp
Byte-Range:  y/x
-----x1234567$
```

Figure 70: MSRP OK

Right@participant.com uses the same MSRP session to send "object descriptions" to *left@participant.com* who is keeping track of the master SMIL document and updating this when new information arrives.

5.3.2 SMIL Over Gaim

When addressing the operation of the SIP/SIMPLE plugin some plugin specific functions (see `simple.c` in the source code of Gaim in Appendix N) are referred to.

- **Simple_im_send:** Sending an instant message runs the next function
- **Simple_send_message:** Sending IM or "typing indicator" messages
- **Send_sip_request:** Translates the IM into a SIP request of the "MESSAGE" type
- **Sendout_pkt:** Sends packets containing the "MESSAGE"
- **Simple_process_incoming_message:** Parsing incoming messaged. Finds out what type of packet is incoming and who sent the packet.
- **Send_sip_response:** If the message contains content of the MIME type `"text/plain"`, "OK" responses are sent back to the "From".
- **Simple_keep_alive:** Keeps alive the connection when no packets are sent by sending a UDP packet with 0 bytes.

To set up a SIP session, registering must take place using regular SIP messages: `send_sip_requests` and `send_sip_responses` are interchanged by two Gaim clients and the intervening servers. Transactions contain `Via`, `From`, `To`, `Max-Forwards`, `Cseq`, `User-Agent: Gaim SIP/SIMPLE Plugin`, `Call-ID`, `Content-Length` and `Method` type. Packets are sent over the signaling route of the existing SIP session to another Gaim client (with address variable `GaimConnection *gc`) using the `sendout_pkt` function. Figure 71 illustrates the transfer of a single message over an established session. The `sendout_pkt` function sends packets over a TCP or UDP connection.

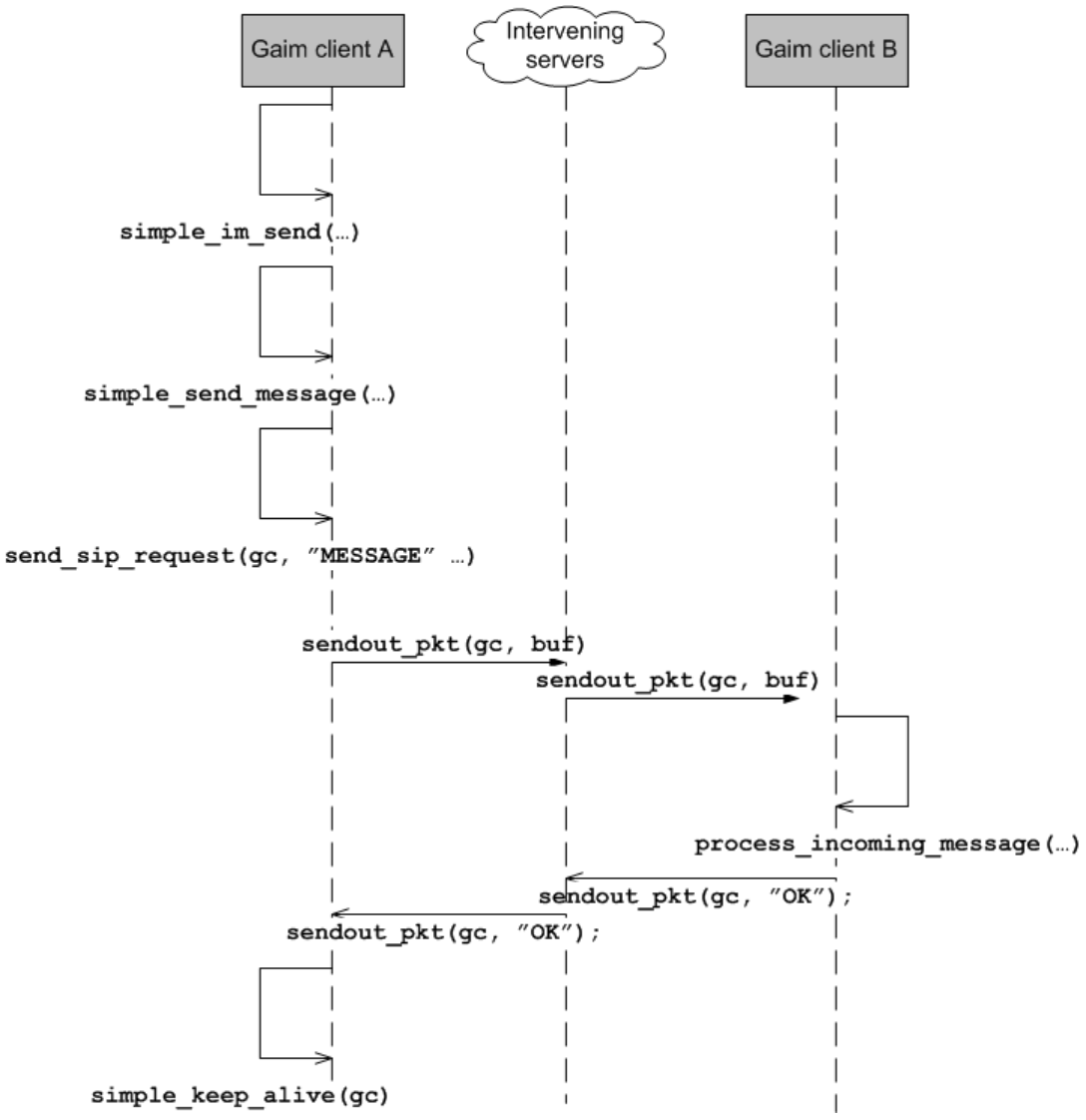


Figure 71: Gaim: Sending a message

5.3.3 Object Descriptions

Object description in this context means description of several aspects concerning the participation in a distributed scene. This description can tell for instance if the person wants to move (change location) within the scene, etc. The master SMIL document has to be updated with this new information and the participant must provide the necessary information.

- Location of a participant inside a scene.
- Shape of a region.
- Media quality parameters, etc.

One way to handle this is to send small XML documents (content type: `application/xml`) over the existing MSRP session with this information. This XML document are then used to update the remote master SMIL document with new preferences concerning the participant.

The following example uses XCAP (XML Configuration Access Protocol) to modify a remote SMIL document using PUT, in particular the `<region ... />` which is referenced with an XPath URI. The `<region ... />` keeps track of the location in which the participant's video stream are displayed.

```
PUT http://documenturl/smildocument.smil/smil/head/layout/
    region HTTP/1.1
Content-Type:  application/xml-fragment-body
<region id="participanta_region" left="80%" top="40%"
    width="200" height="300" />
```

Figure 72: Region element modification

Synchronization problems may occur if several participants are allowed to write to the same document. This problem may be solved if the participants may only modify a referenced document describing their own participation and the master SMIL document is not writable by the participants.

Instead of using HTTP, the IM client can handle the SMIL document modification over MSRP. The UML deployment diagram of Figure 73 describes this option.

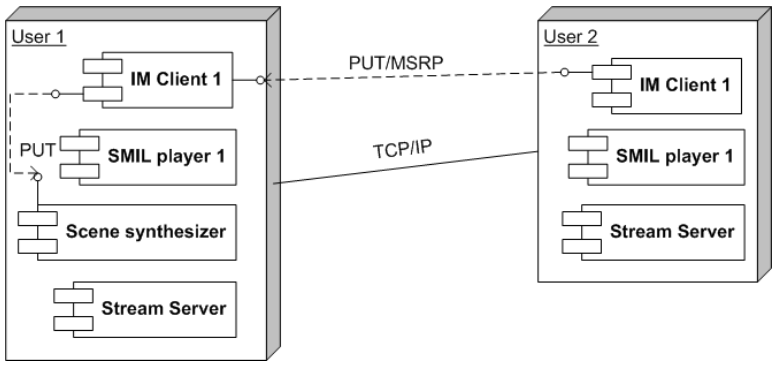


Figure 73: IM client handles the document modification

A proposal for a new address scheme, using ideas from XCAP and XPath (Section 2.13), may look like the example in Figure 74. The first part of the URI are changed to the MSRP URL pointing at one and only one possible scene description. The second part is pointing at nodes within the document. If several documents may be remotely accessed, one possibility may be to include the filename of the document after the MSRP URL.

```
"msrp://hostaddress:port/protection;transportprotocol"/smil
    head/layout/region
```

Figure 74: New address

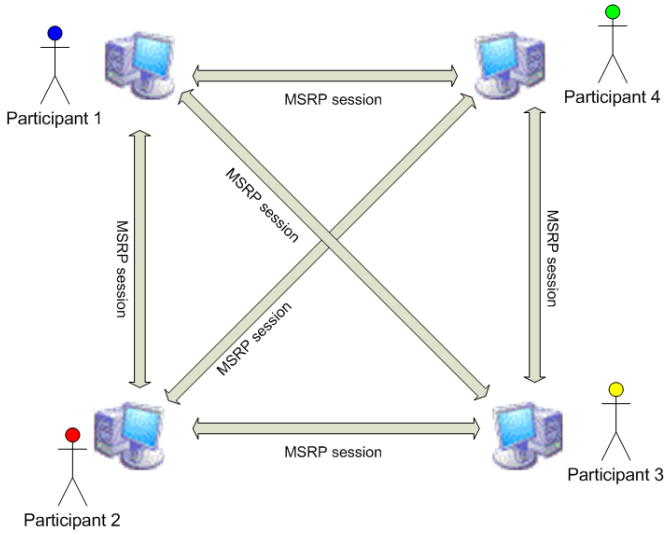
5.3.4 Multiparty Sessions

MSRP is a P2P protocol. To enable multiparty sessions all the participants needs to establish MSRP sessions between each others (Figure 5.3.4). One of the users initiate a conference room and this user thus becomes the "conference focus". This conference room has its own SIP URI. Other participants may join this conference room (sending a SIP INVITE request to the conference room). In order to exchange scene descriptions between each others the participants need to [66]:

1. Obtain each others identities from a received "conference event package" from the "conference focus".

2. Establish P2P sessions to each of the other participants.

3. Send private messages containing scene descriptions to all the other participants.



In the following example (Figure 75) Participant 1 wants to update his region and sends a Object Description to the other participants (to each of their respective MSRP URLs over the existing MSRP sessions).

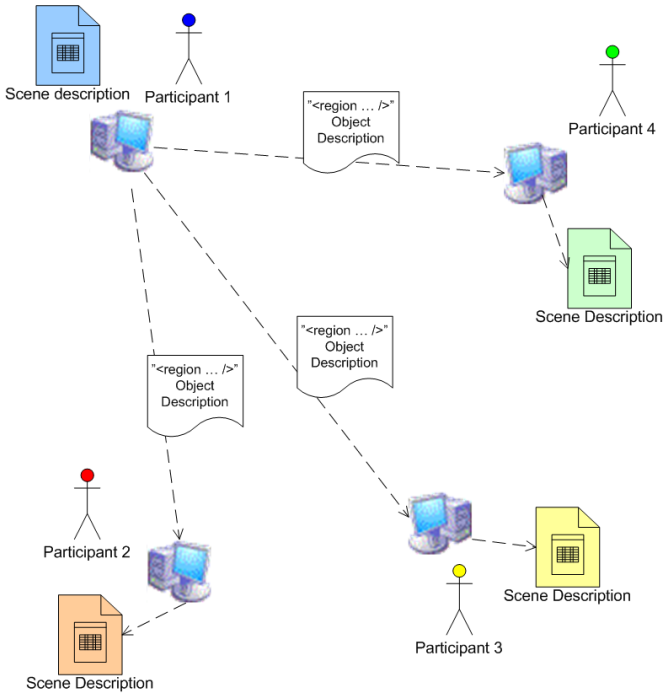


Figure 75: Participant 1 sending Object Descriptions

5.3.5 Auto-stereoscopic 3D View

Using X3D and OpenGL. It's possible to write auto-stereoscopic scenes in X3D as the 3D graphics are rendered by OpenGL. When auto-stereoscopic X3D files are used within SMIL only the part with X3D really looks three dimensional. A pair of LCD glasses or similar is needed to watch the scene. Stereo cameras are rendered the same way as with NVSG (Section 3.5):

1. "Set the geometry for the view from left human eye.
2. Set the left eye rendering buffers.
3. Render the left eye image.
4. Clear Z-buffer (if the same Z-buffer for left and right image is used).
5. Set the geometry for the view from right human eye.
6. Set the right eye rendering buffers.
7. Render the right eye image.
8. Swap buffers."

Stereoscopic OpenGL Tutorial [67]

5.3.6 Adaptive Datarate of Objects in a Scene

As mentioned in Section 3.7, the SMIL Streaming Media Object Module introduce the possibility of describing properties of a RTP stream appearing in a SMIL document. A media player need some parameters to understand the RTP data, these SDP parameters may be merged into the actual SMIL document (details about this can be found in "Integrating SDP Functionality Into SMIL" by Philipp Hoschka at W3C [68]).

Rtpmap allow dynamic video resolution or quality of audio according to constraints on available bandwidth. To illustrate the use of rtpmap the part of the SMIL document in Figure 76 is extended. The part of the document where a video stream is included is extended with the ability to choose among a number of video streams with alternative bitrates (Figure 77) [22].

```
<video src="videosource.mpg" region="playedvideo" fit="fill"
    repeatCount="indefinite" type="mpg" />
```

Figure 76: Without rtpmap: One quality level

The video stream "videosource.mpg" in this example may be changed to "videosource.rtsp" imagined to be of type MPEG-2 (MIME type video/mpeg) with available bitrates of payload="1", payload="2", payload="3" at (for instance) 1150 kbps, 2000 kbps and 5000 kbps (VCD, SVCD and DVD quality).

```
<video src="rtp://www.videosource.com/videosource.rtsp"
    port="1024-1026" transport="RTP/AVP" rtpformat="1,2,3"
    region="playedvideo" fit="fill"
    repeatCount="indefinite" >
<rtpmap payload="1" codec="mpv2"/>
<rtpmap payload="2" codec="mpv2"/>
<rtpmap payload="3" codec="mpv2"/>
</video>
```

Figure 77: With rtpmap: I.e. three quality levels

If the available bandwidth in network suddenly is reduced, the client can be forced to reduce the rtpmap payload="3" to rtpmap payload="2". This is a way of down scaling the content resolution when there is a lack of available network resources. The transport type in this example is UDP over IPv4 (with port range from 1024 to 65535). Also worth mentioning is the transport="RTP/AVP" attribute telling that the transport protocol is RTP/AVP, IETF's Real-time Transport Protocol for the Audio/Video profile over UDP [69].

5.3.7 Ambulant Playing SMIL Documents

To describe how a SMIL Client opens a document, Ambulant Player us used, despite the current lack if support of other document standards like SVG and X3D. When interacting through the GUI, a user may select to open an URI or a local document. The application then need to get the document being opened, parse the document into a Document Object Model (DOM) tree and create a player to play the DOM tree. The player needs to know how to fetch the media data (video, audio streams, etc.) and how to create windows and buttons. Ambulant 1.6.1 has two possibilities for opening

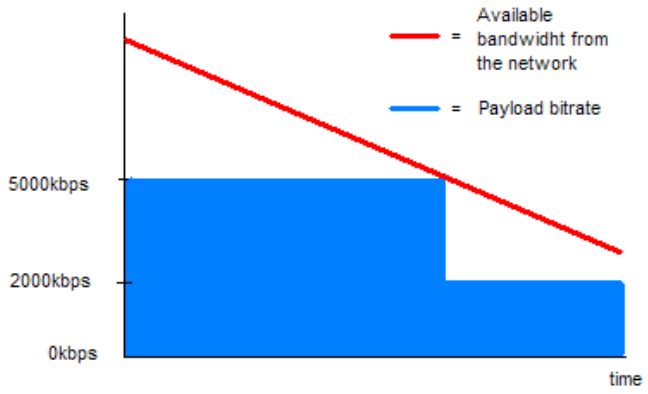


Figure 78: Payload vs. available bandwidth

a document, opening an existing document with (`MmDoc::OnOpenDocument`) and opening an URL with `MmDoc::SetPathName`. SMIL documents may be loaded by Ambulant from several different sources:

- **Create_from_file:** Opens from SMIL document files.
- **Create_from_string:** The function reading from a SMIL source.
- **Create_from_tree:** Creates from a DOM tree.
- **Create_from_url:** Creates from a URL.

When users opens XML documents (with any kind of XML player), the player needs to fetch the document and parse the document into a DOM tree. The player application must be able to to play the DOM tree and possible media data referenced from the XML, know how to create GUI buttons, windows, etc.

SMIL documents are opened by `read_data_from_url` and DOM trees are created from the SMIL documents by `create_from_string`. A document object containing each DOM tree. A player object is then created by running `create_smil2_player` together with the DOM, different factories and an embedder (embedding elements into the document played on the screen). The player object is then implemented by the main program. After the smil_player is implemented it needs to create its internal structures to make playback enabled. To play SVG or X3D specific factories has to be developed to handle rendering of these two kind of standards.

There are a number of factories put together in a `factories` struct:

- **Window_factory:** The factory the player calls when it needs a window.
- **Global_playable_factory:** The object use to create media renderers.
- **Datasource_factory:** The functions that create datasources for media. Functions

Playback is initiated by calling the `start` method of the `player` object. This procedure runs `start` in the scheduler which starts playing the root node of the DOM tree. The scheduler sometimes needs things to be rendered. The scheduler then calls a method `new_playable` from the global_playable_factory which passes the DOM node to a bunch of factories to see if any of them can handle it and create a playable for the object. Most of the times `start` is executed the renderer needs to get data from a URL or a local source. The scheduler creates a datasource through the datasource_factory object. The renderer asks the datasource when it needs something to be drawn. The user interact by invoking a method called `playable_notification` when he pushes a button (like `pause`) or by some other way interact with the player.

5.4 Conformance to Requirement Specification

It is difficult to state if the demands of the requirements specification met by the proposal (using MSRP) without testing or simulating. Humans find 10-20*ms* to be an acceptable delay when expecting visual results of a response. That's why end-to-end delay caused by TCP's retransmission of lost packets may become a problem in real-time systems. The described proposal and Distributed Multimedia Plays Virtual Dinner [1] uses RTP over UDP as the transport mechanism for streaming media, if a packet is lost does not introduce any problems with UDP. On the other hand, MSRP needs a reliable protocol in the underlying transport layer. Delays are critical for real-time scenes, introducing some mechanisms is necessary avoid any problems and keeping the delays low, or at least to keep the possibility of perceiving delays at a minimal level. One approach may be to define larger transparent areas (in i.e. SMIL+SVG) surrounding objects which are likely to movie. If a person object moves the transparent area will handle the delay introduced by TCP in a neat fashion. Other "rules" are possible to define for alternative situations.

5.5 Section Summary

A requirement specification of a distributed system, and proposed models of a system exchanging SMIL scene descriptions over MSRP sessions established with help from SIP and SDP has been presented. Multiparty sessions are taking place over P2P MSRP sessions between each participants (all-to-all). Rtpmap have been proposed as a potential solution of handling adaptive bitrates of media content, enabling different preconfigured levels of bitrates. How to exchange SMIL documents over existing applications (demonstrated with Gaim) and how SMIL documents are loaded and played with Ambulant are also demonstrated, but it is vital to realize that Gaim does not support MSRP or any kind of P2P message sessions, and Ambulant does not support playback of either SVG or X3D documents.

6 Discussion

When proposing answers to the questions in the description, a number of presuppositions have to be made. These early stages of the process of proposing solutions, with a number uncertainties, may come up with suggestions that may not be feasible or implementable. Some of this uncertainties are discussed in this section of the Master's Thesis.

6.1 Protocols

SMIL document transport in a distributed scene system are preferred to be P2P because of the real-time nature of the distributed scene. To sum up the comparison of SIP and XMPP (Section 2.6) it is obvious that XMPP excludes itself as a messenger carrying entire SMIL scenes or small XML fragments with update information (object descriptions). XMPP messages travel through a number of the intervening servers introducing latencies. SIP allows MSRP sessions to be created between communicating entities (called peers or the participants of a distributed scene).MSRP allow i.e. MIME content (SMIL or other kinds of XML documents included) to be included in the message body. MSRP works over TCP as the transport layer protocol. A problem due to real-time demands may be that TCP requires ACK messages to acknowledge the receipt of packages. This problem may be possible to circumvent by making rules for how often users can send new object descriptions or how the degree of importance of the content of each MSRP packet containing an object description.

6.2 XML or Binary Scene Descriptions

One of the more important questions when establishing a real-time system is to exchange data in the most effective (fastest way), one of the simplest way to speed up this factor is to use a lesser number of bytes (smaller size). Binary scene descriptions are compressed in a number of ways, making it more effective than a regular ZIP compression. Object descriptions, which is the most regular type of messages sent between peers of a distributed 3D scene, are short messages. The upper limit may often be not more than 1000 bytes (using a short XML file as example). Creating a ZIP

file (best compression) of such a file lowers the size to about 70%. Binary files add some other compression schemes making the size even a bit smaller. But having high capacity networks makes the time difference between passing such small files more or less neglectable.

6.3 Scene Handling

Another way of handling scenes, compared to the P2P multiparty scheme in Section 5.3.4, may be to arrange a hub-like system with a master scene description kept and updated by a single participant or a server (but still having P2P media streams between participants). This approach makes it necessary to download complete scene descriptions quite more often in order to obtain the most recently updated scene description, and the documents passed between the hub center and the participants are also larger. Another negative way with this approach is the vulnerability if the network connection, hardware or software at the hub becomes inaccessible or damaged. When using MSRP multiparty messages it is also possible to send messages from each MSRP Client through a relay called MSRP Switch [66]. This MSRP Switch forwards the message (containing i.e. an object description) to all the participants (the other MSRP clients) which updates their own version of the scene descriptions.

6.4 Different Ways of Writing Scenes

Using SMIL as the base document is an effective way to handle regions with corresponding media types (using SMIL or other kinds of referenced/embedded XML based documents) and any time varying content. It is also possible to use SVG as the base document (and reference SMIL when SMIL's functions are needed) but this may be a more bothersome approach. Which approach is taken does not result in any better performance. The question if SVG or X3D documents should be referenced or embedded may be answered in different ways. The performance when playing embedded compared to referenced document are equal, but updating, keeping track of and manage the scene in a structural way may seem better when the documents are referenced and stored as separate files.

6.5 Lack of Present Support

At the present point of time, no player support the latest versions of both SMIL, SVG and X3D at the same time. X-Smiles [70] is tested, it supports a combination of SMIL, SVG and X3D but uses an outdated SVG player (from August 2006, called CSIRO SVG Toolkit). X3D is supported by an embedded version of Xj3D [71] and the SMIL 2.0 Basic Profile is also supported. Even though Xj3D is supposed to support full X3D, another player (the Octaga Player [63]) was needed when testing X3D streaming video played upon a 3D object. The ideal player which is not yet available should support the latest versions of both SMIL, SVG and X3D.

As previously mentioned, SVG is not supporting any kind of streaming media. This subject is under consideration by W3C [56]. Summarizing the lack of present support concludes that the full integration of SVG and X3D with SMIL has been impossible to test.

7 Conclusion

The conclusion part answers the problems addressed in description in a short and concise way as well as other important discoveries and achievements. The first part of the Master's Thesis carries out a review of 3D multiview, autostereoscopic object oriented audiovisual scenes theory and practice. Seven theories (languages or standards) are examined and the theory and practice of each of these theories are identified and presented. Many of these theories have different targeted application areas, but a concluding comparison chart presents the most important issues concerning these seven theories.

The next part presents possible solutions of how to incorporate the currently lacking possibilities of 3D, transparency and custom shapes into SMIL. Embedding inline (the SMIL document) or referencing external SVG or X3D exploits these two standards' features and makes them available through a SMIL document. All of these three standards are easily integrated with each other by referencing their respective XML namespaces. SVG and X3D have many features that may be interesting in distributed scenes, referencing these features avoids "reinvention of the wheel" by creating brand new SMIL namespaces with similar features. To be able to play the most recent SMIL+SVG+X3D documents a (currently not supported) player application is needed.

The last of the main parts of this Master's Thesis is looking into composition of distributed scenes. Four questions are presented in the section introduction asking for proposed solutions to issues like: What is to be designed? The requirement specification. How will distributed scenes be composed? And if the proposals conform with the requirement specification. Using SDP over SIP to establish MSRP sessions between participants able to carry MIME contents (i.e. SMIL and other XML based documents) are one of the possibilities of managing exchange of scene descriptions and scene updates. A consequence of this approach is that any problems introduced by using MSRP over TCP have to be taken care of, i.e. by defining special rules concerning scene management.

7.1 Future Works

- Develop an application able to generate scene descriptions, partially from an initial SMIL document and partially from incoming XML/SMIL object descriptions. This application (may be called a scene synthesizer) should also be able to feed scene data to a SMIL player when updated versions of the scene are available. (A Scene Synthesizer).
- Choose a SVG/X3D/SMIL player and develop a plugin making this player automatically load incoming documents from a chosen instant message client. Also create plugins to the instant message client to push documents (received over an instant message session) to the SVG/X3D/SMIL player application.
- Create a MSRP over SIP plugin for an instant message client (Gaim [31]), and carry out teletraffic measurements of the time needed for a SMIL/XML document to traverse a network using this approach.
- Create factories or embed SVG or X3D players in Ambulant Player to make it able to handle playback of SVG and/or X3D documents.
- Develop a set of special "rules" for scene exchange and object updating, making exchange using MSRP over TCP a more effective method in relation to real-time requirements and the contraints introduced by the reliable TCP protocol.

References

[1] Leif Arne Rønningen. Distributed multimedia plays virtual dinner.
URL: http://www.item.ntnu.no/~leifarne/VirtualDinner4.pdf
URL-date: March 27th, 2006.

[2] IETF. Rfc 2026, the internet standards process – revision 3 (best current practice).
URL: http://www.ietf.org/rfc/rfc2026.txt
URL-date: April 7th, 2006.

[3] SIP Forum. Sip forum.
URL: http://www.sipforum.org
URL-date: February 3rd, 2006.

[4] IETF. Session initiation protocol (sip) working group.
URL: http://www.ietf.org/html.charters/sip-charter.html
URL-date: February 3rd, 2006.

[5] Javvin network management and security. Sip: Session initiation protocol.
URL: http://www.javvin.com/protocolSIP.html
URL-date: March 24th, 2006.

[6] e Multimedia. Sip (session initiation protocol).
URL: http://geocities.com/intro_to_multimedia/SIP/index.html
URL-date: February 6th, 2006.

[7] IETF. Rfc 3428, session initiation protocol (sip) extension for instant messaging (internet standards track protocol).
URL: http://www.ietf.org/rfc/rfc3428.txt
URL-date: March 24th, 2006.

[8] IETF. Rfc 2327, sdp: Session description protocol (internet standards track protocol).

URL: http://www.ietf.org/rfc/rfc2327.txt
URL-date: February 10th, 2006.

[9] IETF. The message session relay protocol (internet-draft).
URL: http://www.ietf.org/internet-drafts/draft-ietf-simple-message-sessions-14.txt
URL-date: April 5th, 2006.

[10] Internet Engineering Task Force. Internet-draft: Sip instant message sessions.
URL: http://www3.ietf.org/proceedings/01dec/I-D/draft-ietf-simple-im-session-00.txt
URL-date: March 22nd, 2006.

[11] C. Jennings. Relay extensions for message sessions relay protocol (msrp).
URL: http://www.sipfoundry.org/msrp/draft-ietf-simple-msrp-relays-07.txt
URL-date: May 30th, 2006.

[12] IETF. Rfc 3920, extensible messaging and presence protocol (xmpp): Core (internet standards track protocol).
URL: http://www.ietf.org/rfc/rfc3920.txt
URL-date: March 10th, 2006.

[13] Internet Engineering Task Force. Extensible messaging and presence protocol (xmpp): Instant messaging and presence.
URL: http://www.ietf.org/rfc/rfc3921.txt
URL-date: May 5th, 2006.

[14] Jabber Inc. Architectural considerations for presence and instant messaging infrastructure - comparing xmpp and sip/simple.
URL: http://www.jabber.com/index.cgi?CONTENT_ID=55&VMX_TRACKED=YES
URL-date: March 17th.

[15] Cathleen Moore. Xmpp vs simple: The race for messaging standards.
URL: http://www.infoworld.com/article/03/05/23/21FExmpp_2.html
URL-date: March 17th, 2006.

[16] International Organization for Standardization. Open system interconnection - basic reference model.
URL: http://standards.iso.org/ittf/PubliclyAvailableStandards/s020269_ISO_IEC_7498-1_1994(E).zip
URL-date: June 4th, 2006.

[17] Wikipedia. User datagram protocol.
URL: http://en.wikipedia.org/wiki/User_Datagram_Protocol
URL-date: June 4th, 2006.

[18] Wikipedia. Transmission control protocol.
URL: http://en.wikipedia.org/wiki/Transmission_Control_Protocol
URL-date: June 4th, 2006.

[19] Wikipedia. Real-time transport protocol.
URL: http://en.wikipedia.org/wiki/Real-time_Transport_Protocol
URL-date: March 27th, 2006.

[20] Wikipedia. Real-time streaming protocol.
URL: http://en.wikipedia.org/wiki/Real_Time_Streaming_Protocol
URL-date: March 27th, 2006.

[21] Wikipedia. Real-time transport control protocol.
URL: http://en.wikipedia.org/wiki/Real_time_control_protocol
URL-date: March 27th, 2006.

[22] Philipp Hoschka and Rob Lanphier. The smil streaming media object module.
URL: http://www.w3.org/TR/2000/WD-smil-boston-20000622/streaming-media-object.html
URL-date: March 24th, 2006.

[23] Wikipedia. Multipurpose internet mail extensions (mime).
URL: http://en.wikipedia.org/wiki/MIME
URL-date: March 24th, 2006.

[24] IETF. Rfc 2045, multipurpose internet mail extensions (mime) part one: Format of internet message bodies (internet standards track protocol).
URL: http://www.ietf.org/rfc/rfc2045.txt
URL-date: March 24th, 2006.

[25] SourceForge. Ambulant.
URL: http://sourceforge.net/search/?words=ambulant&type_of_search=soft
URL-date: April 21st, 2006.

[26] W3C. Synchronized multimedia integration language (smil 2.1).
URL: http://www.w3.org/TR/SMIL2/
URL-date: May 30th, 2006.

[27] Wikipedia. Cascading style sheets.
URL: http://en.wikipedia.org/wiki/Cascading_style_sheets
URL-date: May 4th, 2006.

[28] Wikipedia. Xhtml.
URL: http://en.wikipedia.org/wiki/Xhtml
URL-date: May 30th, 2006.

[29] The Ambulant team. Why a new smil player.
URL: http://www.cwi.nl/projects/Ambulant/Why.html
URL-date: February 1st, 2006.

[30] The Ambulan team. Ambulant design documentation.
URL: http://www.cwi.nl/projects/Ambulant/Docs/ambulantdesign.pdf
URL-date: February 2nd, 2006.

[31] Gaim. Gaim.
URL: http://www.w3.org/TR/xlink/
URL-date: May 27th, 2006.

[32] Inc Linspire. Phonegaim: The all-in-one instant messaging and internet calling solution.
URL: http://www.phonegaim.com
URL-date: May 17th, 2006.

[33] Google. Summer of code 2005.
URL: http://code.google.com/summerofcode05.html
URL-date: May 16th, 2006.

[34] Inc Free Software Foundation. Gnu general public license.
URL: http://www.gnu.org/licenses/gpl.html
URL-date: May 16th, 2006.

[35] W3C. Extensible markup language (xml) 1.1.
URL: http://www.w3.org/TR/2004/REC-xml11-20040204/
URL-date: May 22nd, 2006.

[36] J. Rosenberg. The extensible markup language (xml) configuration access protocol (xcap).
URL: http://www.jdrosen.net/papers/draft-ietf-simple-xcap-10.txt, May 30th, 2006.

[37] W3C. Xml path language (xpath).
URL: http://www.w3.org/TR/xpath
URL-date: May 30th, 2006.

[38] Wikipedia. Scene graph.
URL: http://en.wikipedia.org/wiki/Scene_graph
URL-date: April 25th, 2006.

[39] University of North Carolina at Chapel Hill The Writing Center. Reviews.
URL: http://www.unc.edu/depts/wcweb/handouts/review.html
URL-date: May 28th, 2006.

[40] Web3D Consortium. X3d.
URL: http://www.web3d.org
URL-date: April 23rd, 2006.

[41] ISO/IEC. The virtual reality modeling language - international standard iso/iec 14772-1:1997.
URL: http://tecfa.unige.ch/guides/vrml/vrml97/spec
URL-date: April 24th, 2006.

[42] Don Brutzman. Extensible 3d (x3d) specification document type definition (dtd) x3d-3.0.dtd.
URL: http://www.web3d.org/specifications/x3d-3.0.dtd
URL-date: April 24th, 2006.

[43] Web3D Consortium. Mpeg-4 interactive profile.
URL: http://www.web3d.org/x3d/specifications/ISO-IEC-19775-X3DAbstractSpecification/Part01/MPEG-4interactive.html
URL-date: April 24th, 2006.

[44] W3C. Scalable vector graphics (svg).
URL: http://www.w3.org/Graphics/SVG
URL-date: April 25th, 2006.

[45] nVIDIA. developer.nvidia.com.
URL: http://developer.nvidia.com/object/nvsg_home.html
URL-date: April 25th, 2006.

[46] Leandro Motta Barros. A short introduction to the basic principles of the open scene graph.
URL: http://www.cscience.org/~lmb/OSG/ASIttBPoOSG--02005-10-23.pdf
URL-date: April 27th, 2006.

[47] OSG Community. Open scene graph.
URL: http://www.openscenegraph.org
URL-date: April 27th, 2006.

[48] Matrox. Matrox parhelia precision sdt.
URL: http://www.matrox.com/mga/workstation/3dws/products/special/sdt_technology.cfm
URL-date: April 27th, 2006.

[49] Planar. Sd1710.
URL: http://www.planar.com/Products/flatpanel_monitors/stereoscopic/stereoscopic.cfm
URL-date: April 27th, 2006.

[50] W3C. Synchronized multimedia.
URL: http://www.w3.org/AudioVideo/
URL-date: June 10th, 2006.

[51] Wikibooks. Xml: Managing data exchange/smil.
URL: http://en.wikibooks.org/wiki/SMIL
URL-date: February 6th, 2006.

[52] Patrick Schmitz Warner ten Kate, Ted Wugofski. Synchronized multimedia integration language (smil) modules.
URL: http://www.w3.org/TR/2000/WD-smil-boston-20000622/smil-modules.html
URL-date: March 27th, 2006.

[53] Cyril Concolato. Mpeg-4 laser white paper.
URL: http://www.chiariglione.org/mpeg/technologies/mp04-lsr
URL-date: April 27th, 2006.

[54] ISO/IEC. Call for proposals for lightweight scene representation.
URL: http://www.comelec.enst.fr/~dufourd/laser/laser_saf_call.htm#_Toc58987152
URL-date: May 4th, 2006.

[55] W3C. Scalable vector graphics specification: Clipping, masking and compositing.
URL: http://www.w3.org/TR/SVG/masking.html#ObjectAndGroupOpacityProperties
URL-date: May 8th, 2006.

[56] W3C. Streaming (svg 1.2).
URL: http://www.w3.org/TR/2004/WD-SVG12-20041027/

streaming.html
URL-date: May 27th, 2006.

[57] Adobe Systems Incorporated. *Adobe Premiere Pro User Guide for Windows*. Adobe, 2003.

[58] Wikipedia. Alpha blending.
URL: http://en.wikipedia.org/wiki/Alpha_blending
URL-date: February 1st, 2006.

[59] Real Networks. Realnetworks production guide.
URL: http://service.real.com/help/library/guides/realone/ProductionGuide/HTML/htmfiles/cliptags.htm#178848
URL-date: March 29th, 2006.

[60] Web3D Consortium. Extensible 3d (x3d), part 1: Architecture and base components, 18 texturing component.
URL: http://www.web3d.org/x3d/specifications/ISO-IEC-19775-X3DAbstractSpecification
URL-date: May 5th, 2006.

[61] W3C. Scalable vector graphics specification.
URL: http://www.w3.org/TR/SVG/index.html
URL-date: May 4th, 2006.

[62] Web3D Consortium. Xj3d implementation status.
URL: http://www.xj3d.org/status.html
URL-date: June 5th, 2006.

[63] Octaga AS. Octaga player.
URL: http://www.octaga.com/
URL-date: June 5th, 2006.

[64] Leif Arne Rønningen. Multimedia home space.
Graceful Handover between Mobile and Fixed Networks – Scenario 2016.

[65] Marting Fowler with Kendall Scott. *UML Distilled Second Edition, A Brief Guide to the Standard Object Modelling Language*. Addison-Wesley, 2000.

[66] IETF. Multi-party instant message (im) sessions using the message session relay protocol (msrp).
URL: http://www1.ietf.org/internet-drafts/draft-niemi-simple-chat-04.txt
URL-date: June 7th, 2006.

[67] GALI-3D. Stereoscopic opengl tutorial.
URL: http://www.gali-3d.com/
URL-date: June 7th, 2006.

[68] W3C Philipp Hoschka. Integrating sdp functionality into smil.
URL: http://www.w3.org/AudioVideo/1998/08/draft-hoschka-smilsdp-01
URL-date: February 10th, 2006.

[69] IETF. Framing rtp and rtcp packets over connection-oriented transport.
URL: http://www.ietf.org/internet-drafts/draft-ietf-avt-rtp-framing-contrans-06.txt
URL-date: May 23rd, 2006.

[70] X-Smiles.org et.al. X-smiles, an open xml-browser for exotic devices.
URL: http://www.xsmiles.org
URL-date: May 4th, 2006.

[71] Web3D Consortium. Xj3d.
URL: http://www.xj3d.org
URL-date: May 8th, 2006.

[72] Wikipedia. Namespace.
URL: http://en.wikipedia.org/wiki/Namespace
URL-date: February 7th, 2006.

[73] Softpedia. Gaim 2.0.0 beta 3.
URL: http://www.softpedia.com/get/Internet/Chat/Instant-Messaging/Gaim-for-Windows.shtml
URL-date: April 26th, 2006.

[74] IETF. Rfc3261, sip: Session initiation protocol (internet standards track protocol).

URL: http://www.ietf.org/rfc/rfc3261.txt
URL-date: February 6th, 2006.

[75] The Ambulant team. Ambulant open smil player.
URL: http://www.cwi.nl/projects/Ambulant
URL-date: May 18th, 2006.

[76] Microsoft. Microsoft windows xp professional sp2.
URL: http://www.microsoft.com/windowsxp/pro/default.mspx
URL-date: May 18th, 2006.

[77] Christian Schenk. Miktex.
URL: http://www.miktex.org
URL-date: May 18th, 2006.

[78] LaTeX project team. Latex.
URL: http://www.latex-project.org
URL-date: May 18th, 2006.

[79] ToolsCenter.org. Texniccenter.
URL: http://www.toolscenter.org
URL-date: May 18th, 2006.

[80] Microsoft. Microsoft office visio professional 2003 sp2.
URL: http://office.microsoft.com/en-us/FX010857981033.aspx
URL-date: May 18th, 2006.

[81] Bjarke Viksoe. Gmail drive shell extension.
URL: http://www.viksoe.dk/code/gmail.htm
URL-date: May 18th, 2006.

[82] Corel Corporation. Corel paint shop pro x.
URL: http://www.corel.com/servlet/Satellite?pagename=Corel3/Products/Display&pid=1047025487586
URL-date: May 18th, 2006.

[83] Bruce B. Lowekamp David A. Bryan and Cullen Jennings. Sosimple: A serverless, standards-based, p2p sip communication system.
URL: http://www.cs.wm.edu/~bryan/pubs/bryan-AAA-IDEA2005.pdf
URL-date: April 4th, 2006.

[84] Wikipedia. Distributed hash table.
URL: http://en.wikipedia.org/wiki/Distributed_Hash_Table
URL-date: April 4th, 2006.

[85] Wikipedia. Directed acyclic graph.
URL: http://en.wikipedia.org/wiki/Directed_acyclic_graph
URL-date: April 27th, 2006.

[86] Jonathan Rosenberg. Xcap tutorial.
URL: www.jdrosen.net/papers/xcap-tutorial.ppt
URL-date: May 30th, 2006.

[87] IETF. Sctp as a transport for sip.
URL: http://www3.ietf.org/proceedings/02mar/I-D/draft-ietf-sip-sctp-01.txt
URL-date: June 4th, 2006.

A Appendix: SMIL, SVG and X3D Documents

Referenced SMIL, SVG and X3D documents with pictures referenced inside from inside the respective documents are presented in this Appendix.

A.1 SVG Documents

A.1.1 Persona.svg

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
    "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg width="500" height="500" viewBox="0 0 500 500"
    xmlns="http://www.w3.org/2000/svg" version="1.1"
    xmlns:xlink="http://www.w3.org/1999/xlink">

    <desc>Person A(persona)</desc>
    <defs>
        <clipPath id="persona">
            <polygon points="50,40 66,35 84,35 100,40
                100,60 84,70 100,75 110,75 130,170
                110,170 130,300 90,300 75,225
                60,300 20,300 40,170 20,170
                40,75 50,75 66,70 50,60"/>
        </clipPath>
    </defs>
    <g clip-path="url(#persona)" visibility="visible"
        opacity="1.0" >
        <g>
            <image x="0" y="0" width="500" height="500"
                xlink:href="personb.png"/>
        </g>
    </g>
</svg>
```

Figure 79: Persona.svg

A.1.2 Personb.svg

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
    "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg width="500" height="500" viewBox="0 0 500 500"
    xmlns="http://www.w3.org/2000/svg" version="1.1"
    xmlns:xlink="http://www.w3.org/1999/xlink">

    <desc>Person B(personb)</desc>
    <defs>
        <clipPath id="personb">
            <polygon points="50,40 66,35 84,35 100,40
                100,60 84,70 100,75 110,75 130,170
                110,170 130,300 90,300 75,225
                60,300 20,300 40,170 20,170
                40,75 50,75 66,70 50,60"/>
        </clipPath>
    </defs>
    <g clip-path="url(#personb)" visibility="visible"
        opacity="1.0" >
        <g>
            <image x="0" y="0" width="500" height="500"
                xlink:href="personb.png"/>
        </g>
    </g>
</svg>
```

Figure 80: Personb.svg

A.1.3 Window.svg

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
    "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="500" height="500" viewBox="0 0 500 500"
    xmlns="http://www.w3.org/2000/svg" version="1.1"
    xmlns:xlink="http://www.w3.org/1999/xlink">

    <desc>A cloudy and blue sky (sky.png) is seen
        outside thin white curtains with green dots
        on them (curtaincoordinates)
    </desc>
    <!- Background sky->
    <image x="0" y="0" width="200" height="200"
        xlink:href="sky.png"/>
    <defs>
        <clipPath id="curtaincoordinates">
            <polygon stroke="black" stroke-width="1"
                points="0,0 200,0 200,200 175,200
                185,120 100,10 15,120 25,200 0,200" />
        </clipPath>
        <clipPath id="skycoordinates">
            <polygon stroke="black" stroke-width="0"
                points="175,200 185,120 100,10
                15,120 25,200" />
        </clipPath>
    </defs>
    <g clip-path="url(#curtaincoordinates)"
        visibility="visible" opacity=".8" >
        <g id="curtains">
            <image x="0" y="0" width="200" height="200"
                xlink:href="curtains.png"/>
        </g>
    </g>
</svg>
```

Figure 81: Window.svg

A.2 X3D Documents

A.2.1 Boxtable.x3d

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN"
    "http://www.web3d.org/specifications/x3d-3.0.dtd">

<X3D version='3.1' profile='Immersive'
    xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance'
    xsd:noNamespaceSchemaLocation=
        'http://www.web3d.org/specifications/x3d-3.1.xsd'>
    <head>
        <meta name='filename' content='x3dcube.x3d'/>
    </head>
    <Scene>

    <NavigationInfo type='"EXAMINE" "WALK" "FLY" "ANY"'
        transitionType='"ANIMATE"' transitionTime='1.0'
        transitionComplete=''/>
    <Shape>
        <Appearance>
            <Material/>
            <ImageTexture url=' "bordtexture.png"
                "bordtexture.png" '/>
        </Appearance>
        <IndexedFaceSet colorPerVertex='false'
            creaseAngle='0.5' coordIndex='0 1 3 2 -1 4 5 7 6
            -1 6 7 1 0 -1 2 3 5 4 -1 6 0 2 4 -1 1 7 5 3 -1'
            texCoordIndex='0 1 3 2 -1 0 1 3 2 -1 0 1 3 2 -1 0
            1 3 2 -1 0 1 3 2 -1 0 1 3 2 -1'>
            <Color color='0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0'/>
            <Coordinate point='-2 1 1 -2 -1 1 2 1 1 2 -1 1 2 1
            -1 2 -1 -1 -2 1 -1 -2 -1 -1'/>
            <TextureCoordinate point='0 1 0 0 1 1 1 0'/>
        </IndexedFaceSet>
    </Shape>
</Scene>
</X3D>
```

Figure 82: Boxtable.x3d

A.3 Pictures Used in Documents

These are the referenced images in Section 4.1.4.

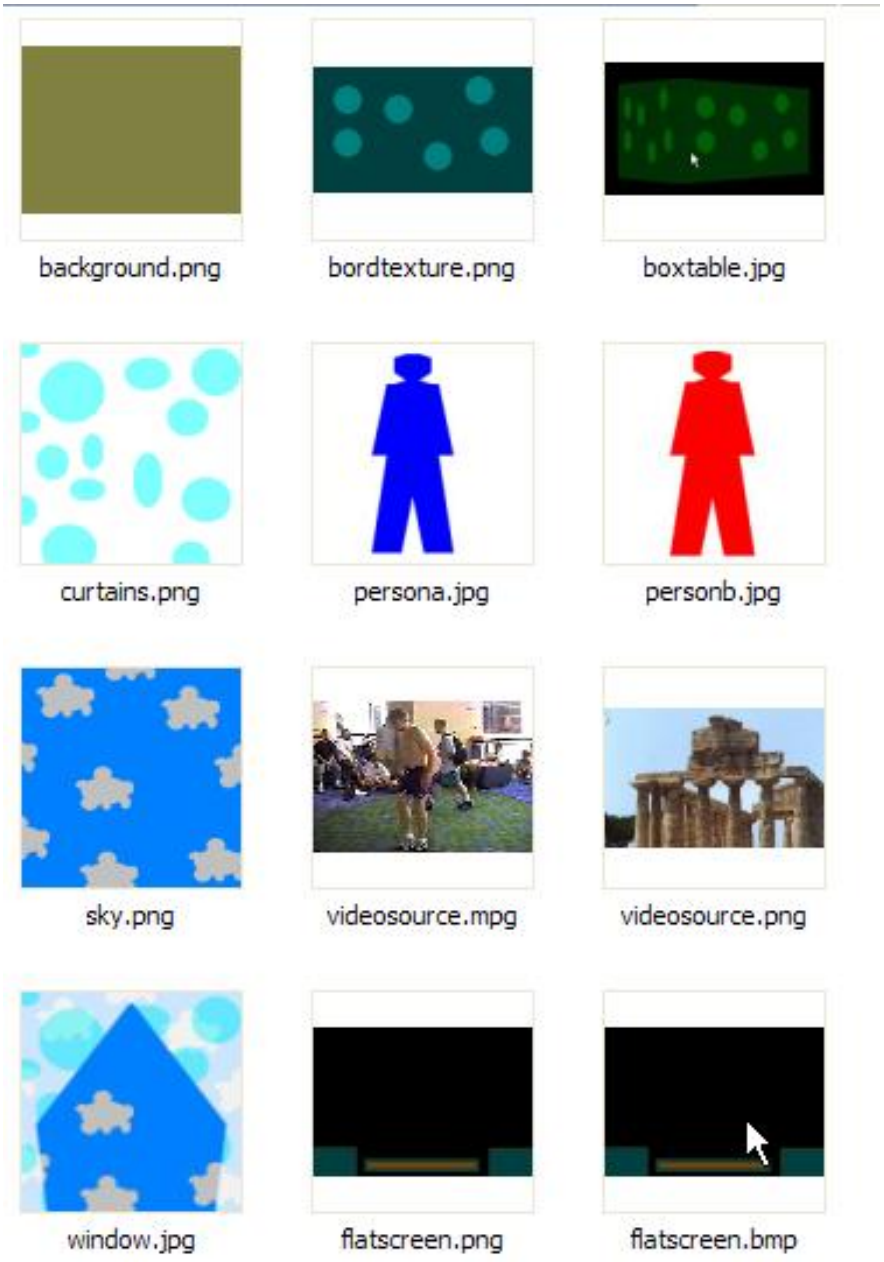


Figure 83: Referenced images

A.4 SMIL documents

A.4.1 Embeddedinline.smil

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 2.0//EN"
    "http://www.w3.org/2001/SMIL20/SMIL20.dtd">
<!- Namespaces:  SMIL, SVG and X3D. A player supporting all
    three standards are needed.  Try X-Smiles -!>
<smil xmlns="http://www.w3.org/2001/SMIL20/Language"
    xmlns:svg="http://www.w3.org/2000/svg" version="1.1"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"
    xsd:noNamespaceSchemaLocation="http://www.web3d.org/
    specifications/x3d-3.1.xsd">
<head>
<layout>
    <root-layout width="1024" height="768" />
    <region id="background_region" left="0%" top="0%"
        z-index="1" />
    <region id="svgpersona_region" left="85%" top="45%"
        width="150 "height="300" />
    <region id="svgpersonb_region" left="15%" top="45%"
        width="150" height="300" />
    <region id="flatscreen_region" left="25%" top="15%"
        width="300" height="200" z-index="20"/>
    <region id="playedvideo" left="27%" top="17%"
        width="270" height="170" z-index="30"/>
    <region id="window_region" left="75%" top="20%"
        width="200" height="200" />
    <region id="boxtable_region" left="45%" top="70"
        width="300" height="100" />
</layout>
</head>
<body>
<!- All the objects are displayed in parallel and starts
    simultaneously -!>
```

```
<par>
<!- SMIL, Background image, wall and floor -!>
<img src="background.png" region="background_region"
    fit="fill" dur="indefinite"/>
<!- SVG, Person A -!>
<svg:svg region="svgpersona_region">
    <svg:desc>Person A(persona)
    </svg:desc>
    <svg:defs>
        <svg:clipPath id="persona">
        <svg:polygon points="50,40 66,35 84,35 100,40 100,60
            84,70 100,75 110,75 130,170 110,170 130,300
            90,300 75,225 60,300 20,300 40,170 20,170
            40,75 50,75 66,70 50,60"/>
        </svg:clipPath>
    </svg:defs>
    <svg:g clip-path="url(#persona)" visibility="visible"
        opacity="1.0">
        <svg:g>
            <svg:image x="0" y="0" width="500" height="500"
                xlink:href="personA.mpg" />
        </svg:g>
    </svg:g>
</svg:svg>
<!- SVG, Person B. Has the same shape as Person A -!>
<svg:svg region="svgpersonb_region">
    <svg:desc>Person B(personb)
    </svg:desc>
    <svg:defs>
        <svg:clipPath id="personb">
        <svg:polygon points="50,40 66,35 84,35 100,40 100,60
            84,70 100,75 110,75 130,170 110,170 130,300
            90,300 75,225 60,300 20,300 40,170 20,170
            40,75 50,75 66,70 50,60"/>
        </svg:clipPath>
    </svg:defs>
    <svg:g clip-path="url(#personb)" visibility="visible"
```

```
    opacity="1.0">
        <svg:g>
            <svg:image x="0" y="0" width="500" height="500"
                xlink:href="personB.mpg" />
        </svg:g>
    </svg:g>
</svg:svg>
<!- X3D, Table / box on the floor-!>
<xsd:X3D region="boxtable_region" version='3.1'
    profile='Immersive' >
    <xsd:head>
    </xsd:head>
    <xsd:Scene>
    <xsd:Shape>
        <xsd:Box/>
        <xsd:Appearance>
            <xsd:ImageTexture url='bordtexture.png'/>
        </xsd:Appearance>
    </xsd:Shape>
    </xsd:Scene>
</xsd:X3D>
<!- SMIL, A TV playing a video stream.  The z-indexes stacks
    the video upon the image of a TV box -!>
<img src="flatscreen.png" region="flatscreen_region"
    fit="fill" dur="indefinite" />
<video src="videosource.mpg" region="playedvideo" fit="fill"
    repeatCount="indefinite" type="mpg" />
<!- SVG, A transparent window with a clouded sky on the
    outside -!>
<svg:svg region="window_region">
    <svg:desc>A cloudy and blue sky (sky.png) is seen outside
        thin white curtains with green dots on them
    </svg:desc>
    <!- Background sky->
    <svg:image x="0" y="0" width="200" height="200"
        xlink:href="sky.png"/>
    <svg:defs>
```

```
    <svg:clipPath id="curtaincoordinates">
        <svg:polygon stroke="black" stroke-width="1"
            points="0,0 200,0 200,200 175,200 185,120
            100,10 15,120 25,200 0,200" />
        </svg:clipPath>
        <svg:clipPath id="skycoordinates">
            <svg:polygon stroke="black" stroke-width="0"
                points="175,200 185,120 100,10
                15,120 25,200" />
        </svg:clipPath>
    </svg:defs>
    <svg:g clip-path="url(#curtaincoordinates)"
        visibility="visible" opacity=".8" >
        <svg:g id="curtains">
            <svg:image x="0" y="0" width="200" height="200"
                xlink:href="curtains.png"/>
        </svg:g>
    </svg:g>
</svg:svg>
</par>
</body>
</smil>
```

Table 5: Embeddedinline.smil

A.4.2 Referenced.smil

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 2.0//EN"
    "http://www.w3.org/2001/SMIL20/SMIL20.dtd">
<!- Namespaces referenced in linked documents are
    SVG and X3D. A player supporting all three
    standards are needed.  ->
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
<head>
<layout>
    <root-layout width="1024" height="768" />
```

```
    <region id="background_region" left="0%" top="0%"
        z-index="1"/>
    <region id="svgpersona_region" left="85%" top="45%"
        width="150" height="300" />
    <region id="svgpersonb_region" left="15%" top="45%"
        width="150" height="300" />
    <region id="flatscreen_region" left="25%" top="15%"
        width="300" height="200" z-index="20"/>
    <region id="playedvideo" left="27%" top="17%"
        width="270" height="170" z-index="30"/>
    <region id="window_region" left="75%" top="20%"
        width="200" height="200" />
    <region id="boxtable_region" left="45%" top="70%"
        width="300" height="100" />
</layout>
</head>
<body>
<!- All the objects are displayed/played in parallel
    and starts simultaneously ->
<par>
<!- SMIL, Background image, wall and floor ->
    <img src="background.png" region="background_region"
        fit="fill" dur="indefinite" />
<!- SVG, Person A ->
    <!- First element of switch will show if
        available, alternatively second element->
    <switch>
        <ref type="image/svg+xml" src="persona.svg"
            region="svgpersona_region"/>
        <img src="persona.png" region="svgpersona_region"/>
        </switch>
<!- SVG, Person B ->
    <switch>
        <ref type="image/svg+xml" src="personb.svg"
            region="svgpersonb_region"/>
        <img src="personb.png" region="svgpersonb_region"/>
    </switch>
```

```
<!- X3D, Table / box on the floor->
    <switch>
        <ref type="model/x3d+xml" src="boxtable.x3d"
            region="boxtable_region"/>
        <img src="boxtable.png" region="boxtable_region"/>
    </switch>
<!- SMIL, A TV playing a video stream.  The z-indexes
    stacks the video upon the image of a TV box ->
    <img src="flatscreen.bmp" region="flatscreen_region"
        fit="fill" dur="indefinite" />
    <video src="videosource.mpg" region="playedvideo"
        fit="fill" repeatCount="indefinite"
        type="mpg" />
<!- SVG, A transparent window with a clouded sky on
    the outside ->
    <switch>
        <ref type="image/svg+xml" src="window.svg"
            region="window_region"/>
        <img src="window.png" region="window_region"/>
    </switch>
</par>
</body>
</smil>
```

Table 6: Referenced.smil

B Appendix: Ambulant Player

B.1 Ambulant Player Interfaces

- **Refcounting protocol:** A low level interface shared by many objects.
- **Player Interface:** This is the top level object.
- **Parser Interface:** Describes the interfaces to the XML parser.
- **Datasource interface:** The interface used to get external data into the program.
- **Playable interface:** This interface makes media items appear on the screen.
- **Layout interface:** This interface is used to determine where those media items show up.
- **GUI window interface:** This interface is used to create new windows.
- **Animation interface:** The interface used for SMIL animation.

The Ambulant team [30]

B.2 Ambulant Player Objects

- **Clocks:** These advance a virtual time.
- **Event processor:** This is the main loop plus the event/callback mechanism.
- **Document:** The representation of a SMIL document.
- **Node:** The representation of the DOM tree.
- **Transitions:** Classes to do visual transitions.
- **Timeline:** This is a description of another scheduler: the MMS scheduler. This scheduler has a much simpler structure than the SMIL 2.0 scheduler.

The Ambulant team [30]

B.3 UML Diagrams

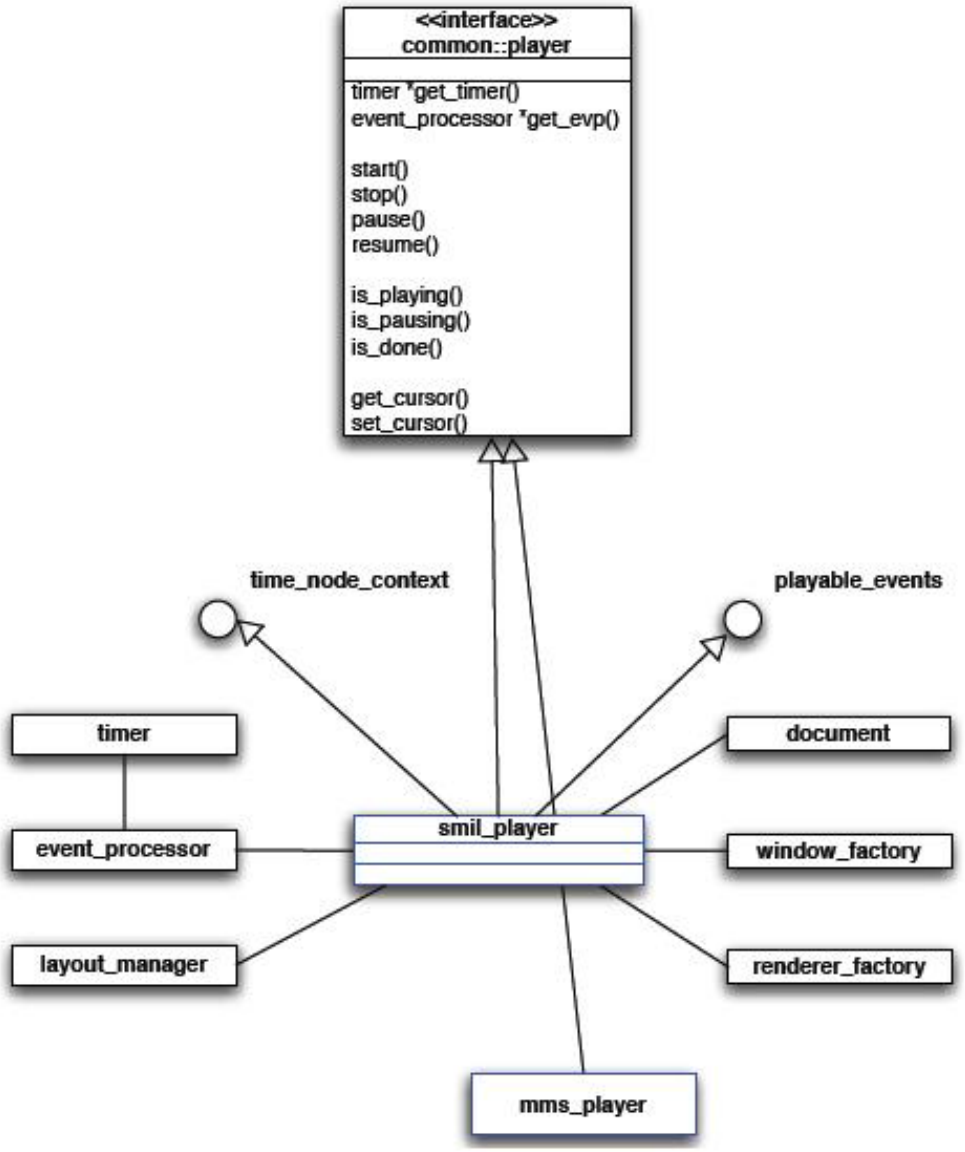


Figure 84: UML diagram for player
[30]

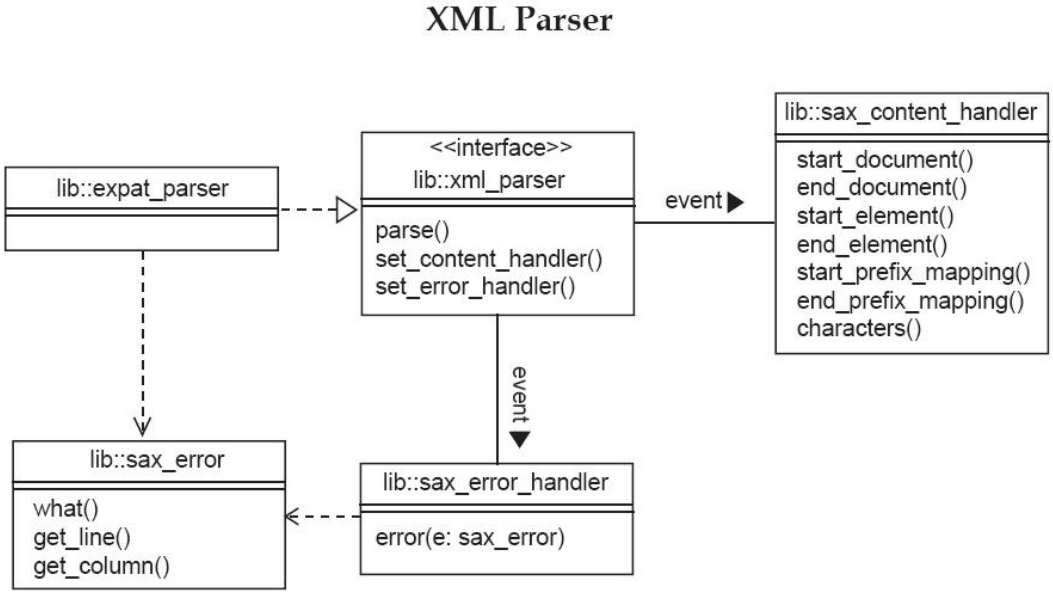


Figure 85: UML diagram XML parser
[30]

C Appendix: Namespaces

C.1 Namespaces

To customize SMIL with extensions like 3D, transparency and custom shapes there is an need for namespaces. Namespaces are defined as:

> In general, a namespace is an abstract container, which is or could be filled by names, or technical terms, or words, and these represent real-world things.
>
> As a rule, names in a namespace cannot have more than one meaning, that is, two or more things cannot share the same name. A namespace is also called a context, as the valid meaning of a name can change depending on what namespace applies. Names in it can represent objects as well as concepts.

Wikipedia [72]

Here, an example namespace (SMIL 2.0) reference is created:

```
xmlns:prefix="http://www.example.com/SMIL20/Extensions"
```

Figure 86: Namespace

The above `prefix` is the reference to the particular namespace used within the SMIL document and `http://www.example.com/SMIL20/Extensions` is the URL of the namespace. This prefix tells the player that the attribute in the SMIL document is not a standard attribute but still a valid attribute because it is defined in the namespace `xmlns:prefix`.The SMIL player then recognizes the attribute and knows how to handle it. Namespaces are added in the `<smil>` attribute:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language"
    xmlns:prefix="http://www.example.com/SMIL20/
    Extensions">
```

Figure 87: Namespace added to SMIL

D Appendix: Gaim

D.1 About Gaim

Gaim is a multi-protocol instant messaging client compatible with AIM (Oscar and TOC protocols), Yahoo, ICQ, MSN Messenger, Gadu-Gadu, IRC, Jabber and Zephyr networks.

Gaim users can log in to multiple accounts on multiple IM networks simultaneously. This means that you can be chatting with friends on AOL Instant Messenger, talking to a friend on Yahoo Messenger, and sitting in an IRC channel all at the same time.

Gaim supports many features of the various networks, such as file transfer (coming soon), away messages, typing notification, and MSN window closing notification.

It also goes beyond that and provides many unique features. A few popular features are Buddy Pounces, which give the ability to notify you, send a message, play a sound, or run a program when a specific buddy goes away, signs online, or returns from idle; and plugins, consisting of text replacement, a buddy ticker, extended message notification, iconify on away, and more.

Softpedia - Gaim 2.0.0 beta 3 [73]

D.2 Working Functionality

1. Currently the following things are working:
2. SIP Message Handling
3. TCP
4. DNS SRV resolving
5. registration on SIP server
6. sending / receiving instant messages

7. presence subscribe / notify for simple status types and PUA == PA

   Bugs / Problems / Missing:

8. SIP over UDP for compatibility with non 2.0 servers
9. resending SIP messages (not sure if needed with TCP)
10. auto login (currently just works after disconnect/connect, don't know why...)
11. uploading presence information (PA != PUA)

Softpedia - Gaim 2.0.0 beta 3 [73]

E Appendix: SIP

E.1 SIP Header Fields

- **To:** The logical recipient of the request. May contain a SIP or SIPS URI. An example of SIP URI is *To: Anton <sip:anton@bratislava.com>*.
- **From:** The logical address of the initiator of the request. An example is
  From: Heidi <sip:heidi@bratislava.com>.
- **CSeq:** CSeq is an abbreviation for Command Sequence and is the way to identify and order transactions. *CSeq: 4711 INVITE* is an example which consists of a sequence number and a method.
- **Call-ID:** A unique identifier. Groups together a series of messages. All requests and responses in a dialog must have the same Call-ID. A cryptographically random Call-ID is recommended because it protects more against hijacking and collision with other Call-IDs.
- **Max-Forwards:** Limits the number of hops in the network to protect against endless loops.
- **Via:** Indicates the path taken by the request and contains information about the transport. protocol that is used.

RFC3261 [74]

Table 7: SIP header fields

F Appendix: SIMPLE

The SIP MESSAGE method does not currently support any sense of a session. Instant messages sent using this method are treated like pager messages. Each message stands alone, and is not linked into a conversation. There has been recent interest in the idea of a SIP based instant message session, where the user experience is more akin to a a text conference or a chat room. This document proposes the idea of treating SIP instant message sessions as a media type that can be initiated using the same SIP mechanisms as for any other media type.

In this approach, a SIP endpoint that wishes to initiate a text chat session would send an INVITE request with an SDP body that describes the session [2]. The sender and recipient then negotiate MESSAGE sessions using normal SIP conventions.

IETF Internet-draft, SIP instant message sessions [10]

G Appendix: Software Tools

- **Gaim:** Instant Message Client [31].
- **AMBULANT Open SMIL Player:** SMIL player [75].
- **Microsoft Windows XP Professional SP2:** Operating system [76].
- **MikTex:** TeX implementation [77].
- **LaTeX:** Document format of the Master's Thesis [78].
- **TeXnicCenter:** Developing LaTeXdocuments [79].
- **Microsoft Wordpad:** Writing and investigating source code.
- **Microsoft Office Visio Professional 2003 SP2:** Illustrations and diagrams [80].
- **GMail Drive:** Backup of Master's Thesis during work [81].
- **Xj3D:** Viewing of X3D files[71].
- **X-Smiles:** Viewing SMIL with X3D/SVG [70].
- **Corel Paint Shop Pro X Version 10.0:** Editing and format conversion of digital images [82].

H Appendix: SOSIMPLE

SOSIMPLE is a SIP/SIMPLE approach based on a open source protocol stacks, standards, fully distributed and decentralized system for peer-to-peer VoIP and IM building on existing SIP components, compatibility with existing SIP are maintained. The communication messages are passing directly between the users. SOSIMPLE is developed by David A. Bryan and Bruce B. Lowekamp at College of William and Mary, USA and partially supported by the Cisco University Research Program. According to the authors a draft of the P2P SIP protocol based on SOSIMPLE is submitted to the IETF [83].

H.0.1 Keywords About SOSIMPLE

- Peer-to-peer VoIP/IM
- No central server
- No naming authority
- No message passes through external proxies
- Advantages of a proxy based system are preserved
- No critical proxy down-time
- Ad-hoc situations
- Less expensive to join, more scalable
- Creating realms of only invited users
- User Mobility Support
- Reuse and compatibility using SIP/SIMPLE

Benefits of using SOSIMPLE [83]

H.0.2 SOSIMPLE Architecture

SOSIMPLE is based on a distributed architecture, there are no central servers or authorities to control the users. The system is also very scalable and the capacity grows

with the number of users. SOSIMPLE is based on a Distributed Hash Table (DHT) similar to Naptster and Gnutella.

Nodes connect to a few other nodes in the overlay network. These nodes are the destination of the SIP messages. Each node acts as User Agent (UA) and proxy at the same time. Together all the nodes replace the need for Proxies and Registrars. The nodes are organized in a Distributed Hash Table [84], each node has its own ID created by combining a hash code, the IP-address and the port number. Advantage is taken by knowing that the user often contacts the same persons again and again. All the messages to maintain the Distributed Hash Table are SIP messages. To have peer-to-peer functionality be built into SIP only a few new headers are needed. The `SIP REGISTER` message pass information between the overlay nodes (see Appendix H for Join and Locate operations).

H.0.3 Future of SOSIMPLE

One important issue due to SOSIMPLE is the handling of keys and authentication. This problem has to be solved to make P2P SIP secure. Public key authentication is not covered in the SIP standard. How routing is to be handled is also an issue. Social routing (means the users social preferences) are taken into the consideration. And the interfaces to existing SIP networks also have to be covered in the future.

H.0.4 SOSIMPLE Client

During e-mail correspondence with one of the developers of the SOSIMPLE architecture, David A. Bryan, it was clear that no implemented version of a SOSIMPLE client are still available. But a tip from D. A. Bryan that an Italian, Enrico Marocco of Telecom Italia, was followed up and this Italian was currently developing a client that could be used in this Master's Thesis for research purposes. Still a great deal of functionality is missing and the stability is not very good. The implemented version was not received early enough to be tested and used as a part of this Master's Thesis.

H.1 Operations

H.1.1 Join

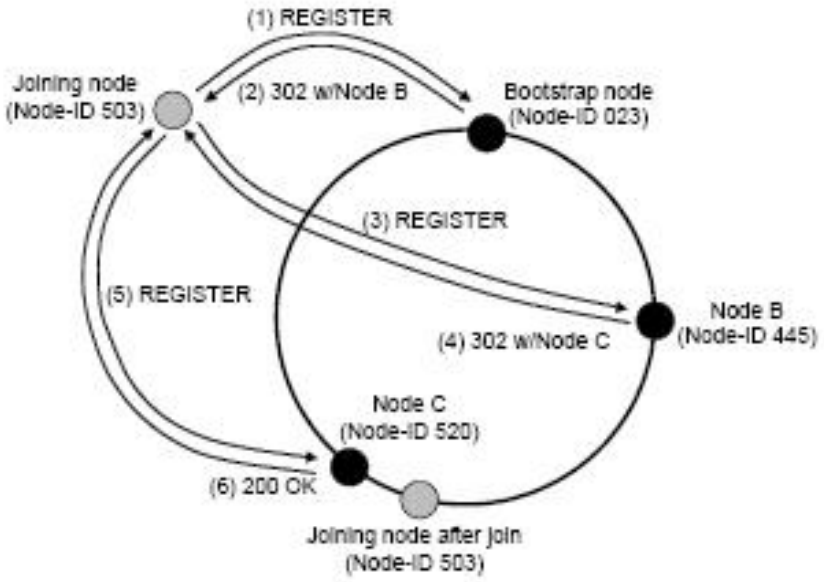


Figure 88: Node joining the overlay
[83]

When a node wants to join an overlay, it must:

1. Locate a node in the overlay (the bootstrap node)
2. Calculate its own node ID
3. Pass the ID in a REGISTER message to the bootstrap node
4. If the bootstrap node is not responsible for the overlay it responds with a SIP 302 Moved Temporarily response and information about which node in the nearest region that can place the joining node in the overlay
5. The joining node now pass his ID in a new REGISTER message to the closer node
6. If this new node is responsible for the overlay, a SIP 200 OK is returned, otherwise the earlier steps are repeated

7. Registration is complete

H.1.2 Locate Another Node

When a node wants to find another node it must:

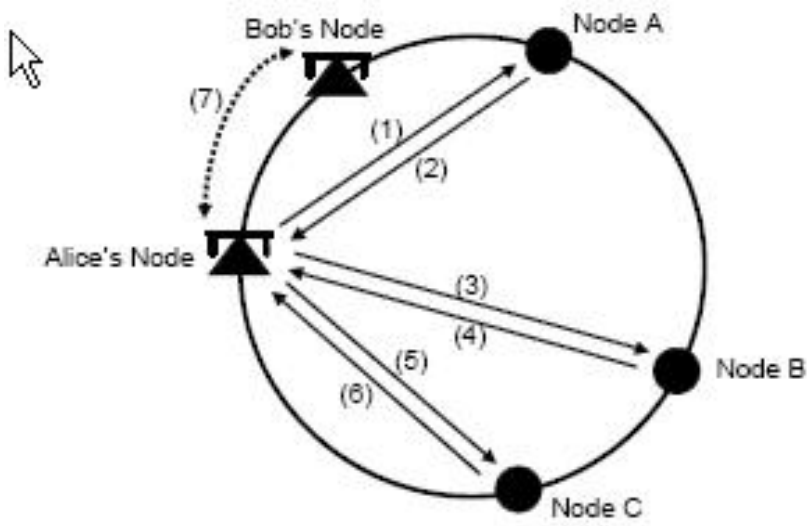


Figure 89: Node allocating another node and starting communication [83]

1. Hash the user name to find the Resource-ID
2. Look in the nodes own finger table and finds the node with the Node-ID closest to the Resource-ID
3. the other node responds with SIP 302 Moved Temporarily including the node it think is the closest
4. Repeats the previous step until the actual Resource is found
5. The Resource node maps the connection between the username and the IP address of the node connection to it
6. The Resource node then returns a 200 OK and the registration data

7. A conventional SIP or SIMPLE session between the actual nodes using regular SIP mechanisms can be established

8. The SMIL documents then may flow directly between the nodes

I Appendix: MPEG-4 Interactive

I.1 IndexedFaceSet

IndexedFaceSet describes how an object should be created of individual faces and polygons.

"IndexedFaceSet

ccw optionally supported. set_colorIndex optionally supported. set_normalIndex optionally supported. normal optionally supported. Only convex indexed face sets supported. Hence, convex optionally supported. For creaseAngle, only 0 and ? radians supported. normalIndex optionally supported.

Face list shall be well-defined as follows:

1. Each face is terminated with -1, including the last face in the array.
2. Each face contains at least three non-coincident vertices.
3. A given coordIndex is not repeated in a face.
4. The vertices of a face shall define a planar polygon.
5. The vertices of a face shall not define a self-intersecting polygon."

MPEG-4 interactive profile [43]

J Appendix: Directed Acyclic Graph

In computer science and mathematics, a directed acyclic graph, also called a dag or DAG, is a directed graph with no directed cycles; that is, for any vertex v, there is no nonempty directed path starting and ending on v. DAGs appear in models where it does not make sense for a vertex to have a path to itself; for example, if an edge u->v indicates that v is a part of u, such a path would indicate that u is a part of itself, which is impossible.

Wikipedia [85]

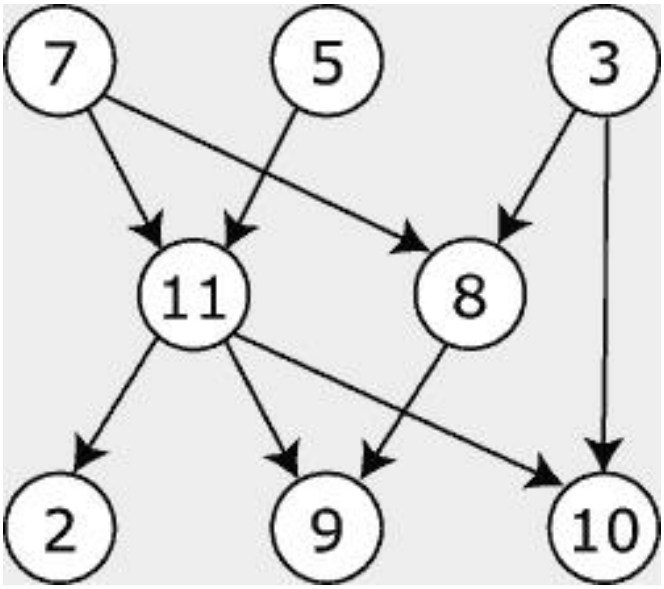


Figure 90: Directed acyclic graph
[85]

K Appendix: X3D Nodes

K.1 Abstract Nodes

K.1.1 X3DTextureCoordinateNode

```
X3DTextureCoordinateNode :  X3DGeometricPropertyNode{
    SFNode [in,out] metadata NULL [X3DMetadataObject]
}
```

Figure 91: X3DTextureCoordinateNode
Web3D Consortium [60]

K.1.2 X3DTextureNode

```
X3DTextureNode :  X3DAppearanceChildNode{
    SFNode [in,out] metadata NULL [X3DMetadataObject]
}
```

Figure 92: X3DTextureNode
Web3D Consortium [60]

K.1.3 X3DTexture2DNode

```
X3DTexture2DNode :  X3DTextureNode{
    SFNode [in,out] metadata NULL [X3DMetadataObject]
    SFBool []      repeatS TRUE
    SFBool []      repeatT TRUE
}
```

Figure 93: X3DTexture2DNode
Web3D Consortium [60]

K.2 Texture Nodes

K.2.1 ImageTexture

```
ImageTexture :  X3DTexture2DNode{
    SFNode [in,out] metadata NULL [X3DMetadataObject]
    MFString [in,out] url [] [urn]
    SFBool []     repeatS TRUE
    SFBool []     repeatT TRUE
}
```

Figure 94: ImageTexture
Web3D Consortium [60]

K.2.2 MovieTexture

```
MovieTexture :  X3DTexture2DNode, X3DSoundSourceNode,X3DUrlObject{
    SFBool [in,out] loop FALSE
    SFNode [in,out] metadata NULL [X3DMetadataObject]
    SFTime [in,out] resumeTime 0 (-?,?)
    SFTime [in,out] pauseTime 0 (-?,?)
    SFFloat [in,out] speed 1.0 (-?,?)
    SFTime [in,out] startTime 0 (-?,?)
    SFTime [in,out] stopTime 0 (-?,?)
    MFString [in,out] url [] [urn]
    SFBool [] repeatS TRUE
    SFBool [] repeatT TRUE
    SFTime [out] duration_changed
    SFTime [out] elapsedTime
    SFBool [out] isActive
    SFBool [out] isPaused
}
```

Figure 95: MovieTexture
Web3D Consortium [60]

L Appendix: XCAP

L.1 XCAP Client Operations

Retrieving:

- Document
- Element
- Attribute

Deleting:

- Document
- Element
- Attribute

Modifying:

- Document
- Element
- Attribute

Adding:

- Document
- Element
- Attribute

Jonathan Rosenberg [86]

M Appendix: SCTP

The Stream Control Transmission Protocol (SCTP) is an alternative to TCP. It is a reliable message based protocol on the same layer as UDP and TCP (transport layer). SIP (which is transport independent) may run over SCTP as well as UDP and TCP. SCTP manages congestion protocol in a manner similar to open one TCP connection for each message, removing delays introduced by "head of line blocking". Both SCTP and TCP manages packet size fragmentation which may occur when the packet to be sent are larger than the MTU of the network. SCTP also enables multi homing (i.e. multiple IP addresses) making it more reliable over lossy conditions. But TCP (which is already thoroughly tested as the transport protocol of SIP) is just as reliable as SCTP when the network is reliable [87].

N Appendix: Attached Files

The attached files consists of references (URLs made available offline) and source files. The contents are shown in Figures: 96, 97, 98 and 99.

Figure 96: References 1/3

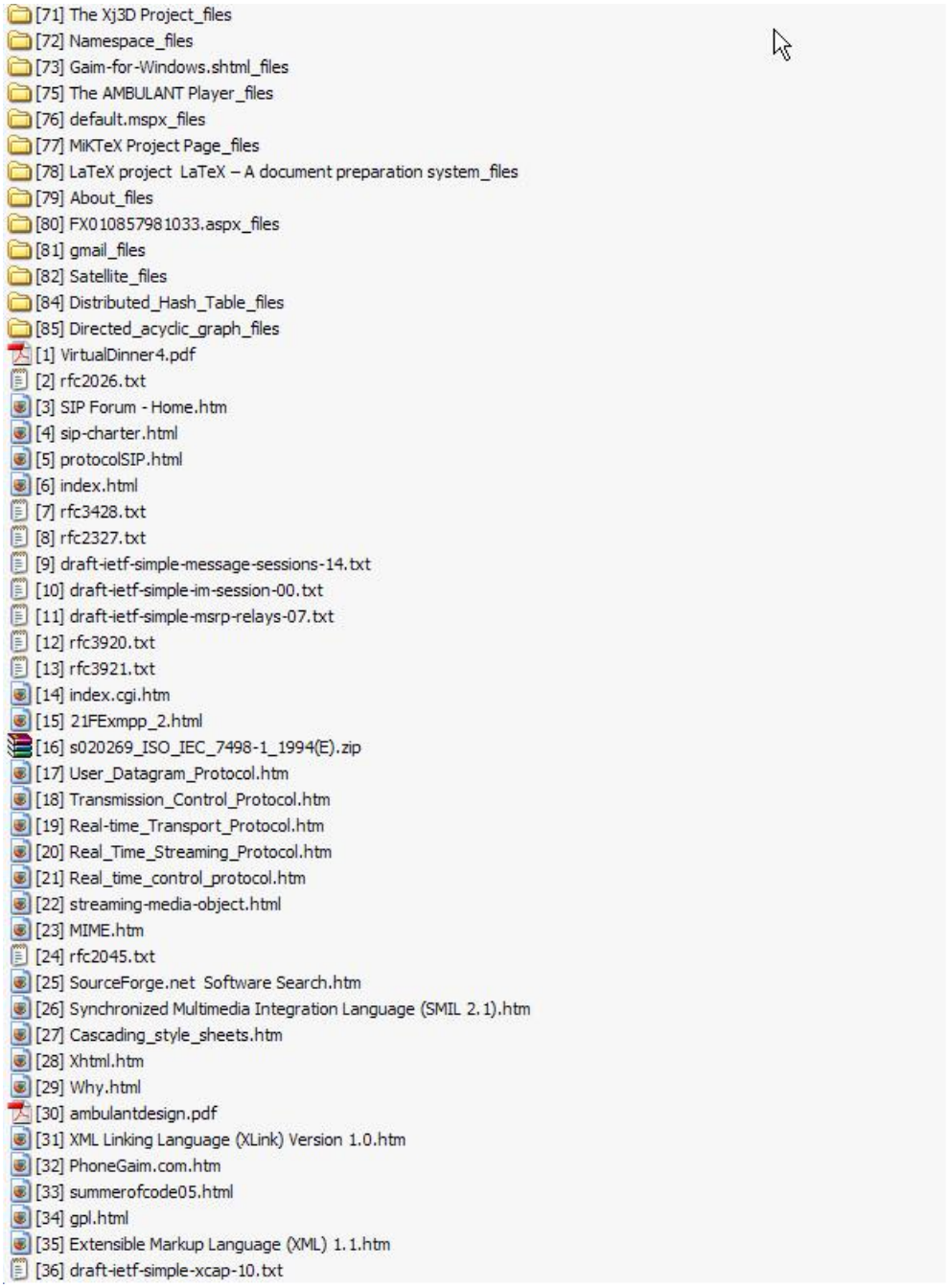


Figure 97: References 2/3

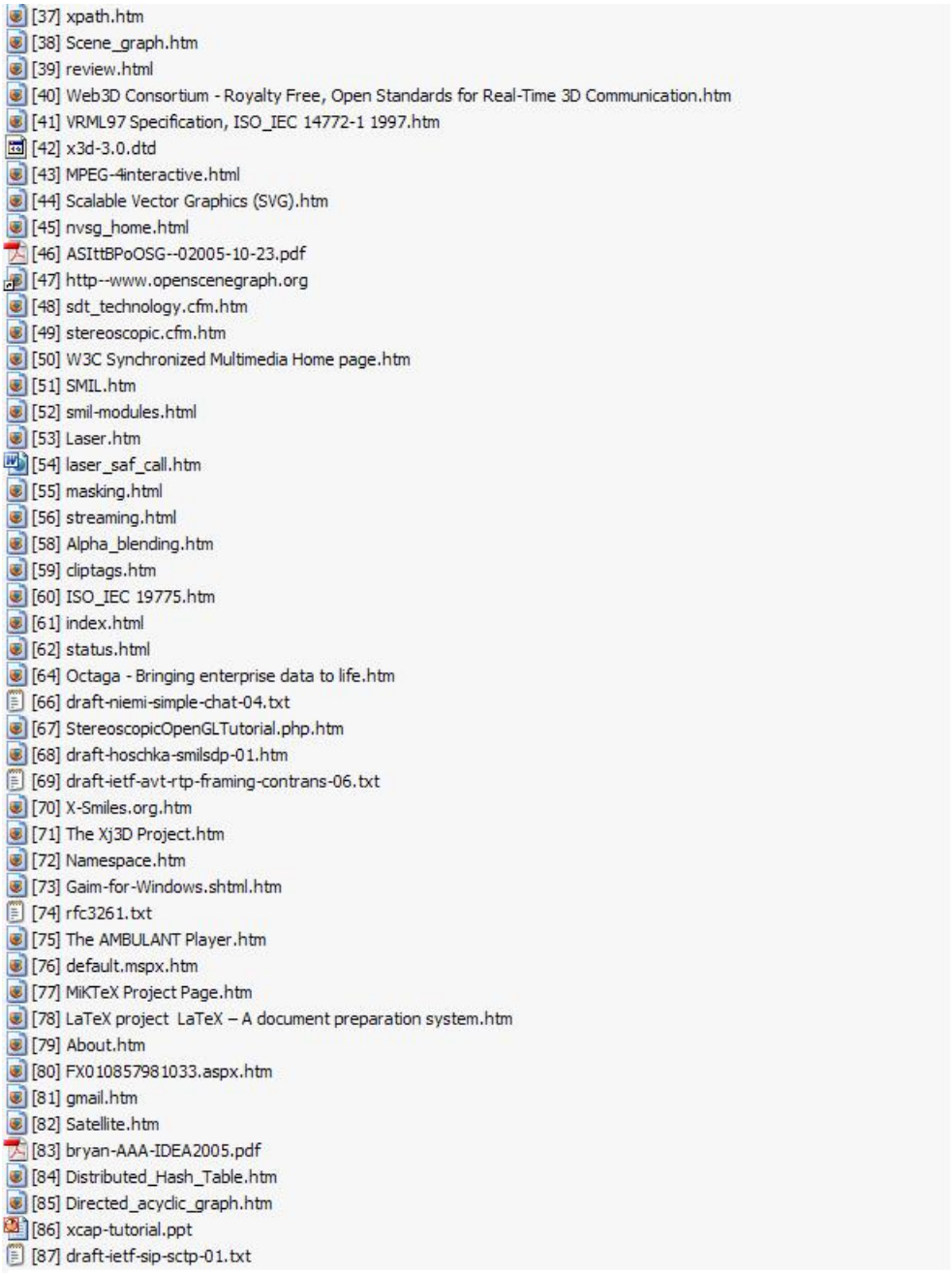


Figure 98: References 3/3

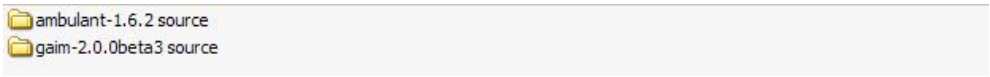


Figure 99: Source code