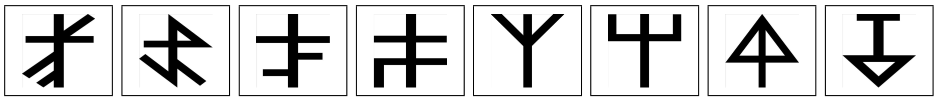
part ii. delivery & presentation

if you linger for a long time in one place you'd almost think there must be something there
¡black¿wittgenstein¡/black¿



2

reading directives In this part we will look at the issues involved in delivery and presentation, primarily from a technical perspective. We will argue the importance of codecs (read compression), and we will discuss the criteria for selecting a particular codec, as well as the standards that have been developed for packaging multimedia content in an effective way. In chapter 4, we will discuss multimedia presentation platforms, and we will look at the Microsoft DirectX 9 platform in somewhat greater detail.

Essential sections are section 3.1, which introduces codecs, 3.2, which discusses the MPEG-4 and SMIL standards and section 4.1, with puts the development of high-end multimedia platforms in a historical perspective. Sections 3.3 and 4.3 can safely be skipped on first reading.

perspectives As you can see below, the topics introduced in this part are not only relevant from a technical perspective. Other perspectives are equally valid:

perspectives – delivery & presentation

- technical – codec selection
- political – market vs. consortium
- sociological – digital services
- legal – copyright protection
- scientific – experience design
- computer science – computational support
- futuristic – global & personal information
- commercial – WMV, Quicktime, RealONE

For example, the issues of copyrights and copyright protection are hot topics, since the rise of the internet is obviously a threat to the tradional industries of music and film distribution.

essay topics Since many of the interesting topics will only be hinted, you may select on or more topics for further investigation and study. As essay titles I would suggest:

- multimedia standards – MPEG4
- XML-based multimedia – SMIL
- multimedia technology – the DirectX 9 toolbox

When you write the essay, then assess first from which perspective you will tackle the subject. When you approach the material from a technical perspective, then make sure that you do understand the technical issues in sufficient detail.

the artwork

1. logo – a drawing by Soutine, it is (almost) my personal logo, and also decorates the cover of OO.
2. signs – property marks, Signs, p. 76, 77.
3. photographs – Jaap Stahlie[1], commissioned work.

[1]www.jaapstahlie.com

3. codecs and standards

without compression delivery is virtually impossible

learning objectives

After reading this chapter you should be able to demonstrate the necessity of compression, to discuss criteria for the selection of codecs and mention some of the alternatives, to characterize the MPEG-4 and SMIL standards, to explain the difference between MPEG-4 and MPEG-2, and to speculate about the feasibility of a semantic multimedia web.

Without compression and decompression, digital information delivery would be virtually impossible. In this chapter we will take a more detailed look at compression and decompression. It contains the information that you may possibly need to decide on a suitable compression and decompression scheme (codec) for your future multimedia productions. We will also discuss the standards that may govern the future (multimedia) Web, including MPEG-4, SMIL and RM3D. We will explore to what extent these standards allow us to realize the optimal multimedia platform, that is one that embodies digital convergence in its full potential. Finally, we will investigate how these ideas may ultimately lead to a (multimedia) semantic web.

1

3.1 codecs

Back to the everyday reality of the technology that surrounds us. What can we expect to become of networked multimedia? Let one thing be clear *compression is the key to effective delivery*

There can be no misunderstanding about this, although you may wonder why you need to bother with compression (and decompression). The answer is simple. You need to be aware of the size of what you put on the web and the demands that imposes on the network. Consider the table, taken from Vasudev and Li (1997), below.

media	uncompressed	compressed
voice 8k samples/sec, 8 bits/sample	64 kbps	2-4 kbps
slow motion video 10fps 176x120 8 bits	5.07 Mbps	8-16 kbps
audio conference 8k samples/sec 8bits	64 kbps	16-64 kbps
video conference 15 fps 352x240 8bits	30.4 Mbps	64-768 kbps
audio (stereo) 44.1 k samples/s 16 bits	1.5 Mbps	128k-1.5Mbps
video 15 fps 352x240 15 fps 8 bits	30.4 Mbps	384 kbps
video (CDROM) 30 fps 352x240 8 bits	60.8 Mbps	1.5-4 Mbps
video (broadcast) 30 fps 720x480 8 bits	248.8 Mbps	3-8 Mbps
HDTV 59.9 fps 1280x720 8 bits	1.3 Gbps	20 Mbps

You'll see that, taking the various types of connection in mind

(phone: 56 Kb/s, ISDN: 64-128 Kb/s, cable: 0.5-1 Mb/s, DSL: 0.5-2 Mb/s)

you must be careful to select a media type that is suitable for your target audience. And then again, choosing the right compression scheme might make the difference between being able to deliver or not being able to do so. Fortunately,

images, video and audio are amenable to compression

Why this is so is explained in Vasudev and Li (1997). Compression is feasible because of, on the one hand, the statistical redundancy in the signal, and the irrelevance of particular information from a perceptual perspective on the other hand. Redundancy comes about by both spatial correlation, between neighboring pixels, and temporal correlation, between successive frames.

The actual process of encoding and decoding may be depicted as follows:

codec = (en)coder + decoder

signal $\rightarrow$ source coder $\rightarrow$ channel coder (encoding)

signal $\leftarrow$ source decoder $\leftarrow$ channel decoder (decoding)

Of course, the coded signal must be transmitted accross some channel, but this is outside the scope of the coding and decoding issue. With this diagram in mind we can specify the *codec design problem*:

From a systems design viewpoint, one can restate the codec design problem as a bit rate minimization problem, meeting (among others) constraints concerning: specified levels of signal quality, implementation complexity, and communication delay (start coding – end decoding).

2

compression methods

As explained in Vasudev and Li (1997), there is a large variety of compression (and corresponding decompression) methods, including model-based methods, as for example the object-based MPEG-4 method that will be discussed later, and waveform-based methods, for which we generally make a distinction between lossless and lossy methods. Hufmann coding is an example of a lossless method, and methods based on Fourier transforms are generally lossy. Lossy means that actual data is lost, so that after decompression there may be a loss of (perceptual) quality.

Leaving a more detailed description of compression methods to the diligent students' own research, it should come as no surprise that when selecting a compression method, there are a number of tradeoffs, with respect to, for example, coding efficiency, the complexity of the coder and decoder, and the signal quality. In summary, the follwoing issues should be considered:

tradeoffs

- *resilience to transmission errors*
- *degradations in decoder output – lossless or lossy*
- *data representation – browsing & inspection*
- *data modalities – audio & video.*
- *transcoding to other formats – interoperability*
- *coding efficiency – compression ratio*
- *coder complexity – processor and memory requirements*
- *signal quality – bit error probability, signal/noise ratio*

For example, when we select a particular coder-decoder scheme we must consider whether we can guarantee resilience to transmission errors and how these will affect the users' experience. And to what extent we are willing to accept degradations in decoder output, that is lossy output. Another issue in selecting a method of compression is whether the (compressed) data representation allows for browsing & inspection. And, for particular applications, such as conferencing, we should be worried about the interplay of data modalities,in particular, audio & video. With regard to the many existing codecs and the variety of platforms we may desire the possibility of transcoding to other formats to achieve, for example, exchange of media objects between tools, as is already common for image processing tools.

compression standards

Given the importance of codecs it should come as no surprise that much effort has been put in developing standards, such as JPEG for images and MPEG for audio and video.

Most of you have heard of MP3 (the audio format), and at least some of you should be familiar with MPEG-2 video encoding (which is used for DVDs).

Now, from a somewhat more abstract perspective, we can, again following Vasudev and Li (1997), make a distinction between a *pixel-based approach* (coding the raw signal so to speak) and an *object-based approach*, that uses segmentation and a more advanced scheme of description.

- *pixel-based* – MPEG-1, MPEG-2, H3.20, H3.24
- *object-based* – MPEG-4

As will be explained in more detail when discussing the MPEG-4 standard in section 3.2, there are a number of advantages with an object-based approach. There is, however, also a price to pay. Usually (object) segmentation does not come for free, but requires additional effort in the phase of authoring and coding.

MPEG-1 To conclude this section on codecs, let's look in somewhat more detail at what is involved in coding and decoding a video signal according to the MPEG-1 standard.

MPEG-1 video compression uses both *intra-frame analysis*, for the compression of individual frames (which are like images), as well as. *inter-frame analysis*, to detect redundant blocks or invariants between frames.

The MPEG-1 encoded signal itself is a sequence of so-called I, P and B frames.

frames

- I: intra-frames – independent images
- P: computed from closest frame using DCT (or from P frame)
- B: computed from two closest P or I frames

Decoding takes place by first selecting I-frames, then P-frames, and finally B-frames. When an error occurs, a safeguard is provided by the I-frames, which stand on themselves.

Subsequent standards were developed to accomodate for more complex signals and greater functionality. These include MPEG-2, for higher pixel resolution and data rate, MPEG-3, to support HDTV, MPEG-4, to allow for object-based compression, and MPEG-7, which supports content description. We will elaborate on MPEG-4 in the next section, and briefly discuss MPEG-7 at the end of this chapter.

example(s) – *gigaport*

GigaPort[2] is a project focussing on the development and use of advanced and innovative Internet technology. The project, as can be read on the website,

[2]www.gigaport.nl/info/en/about/home.jsp

focuses on research on next-generation networks and the implementation of a next-generation network for the research community.

Topics for research include:

GigaPort

- optical network technologies - models for network architecture, optical network components and light path provisioning.
- high performance routing and switching - new routing technologies and transport protocols, with a focus on scalability and stability robustness when using data-intensive applications with a high bandwidth demand.
- management and monitoring - incident response in hybrid networks (IP and optical combined) and technologies for network performance monitoring, measuring and reporting.
- grids and access - models, interfaces and protocols for user access to network and grid facilities.
- test methodology - effective testing methods and designing tests for new technologies and network components.

As one of the contributions, internationally, the development of optical technology is claimed, in particular *lambda* networking, networking on a specific wavelength. Locally, the projects has contributed to the introduction of fibre-optic networks in some major cities in the Netherlands.

research directions– *digital video formats*

In the online version you will find a brief overview of *digital video technology*, written by Andy Tanenbaum, as well as some examples of videos of our university, encoded at various bitrates for different viewers.

What is the situation? For traditional television, there are three standards. The american (US) standard, NTSC, is adopted in North-America, South-America and Japan. The european standard, PAL, whuch seems to be technically superior, is adopted by the rest of the world, except France and the eastern-european countries, which have adopted the other european standard, SECAM. An overview of the technical properties of these standards, with permission taken from Tanenbaum's account, is given below.

system	**spatial resolution**	**frame rate**	**mbps**
NTSC	704 x 480	30	243 mbps
PAL/SECAM	720 x 576	25	249 mbps

Obviously real-time distribution of a more than 200 mbps signal is not possible, using the nowadays available internet connections. Even with compression on the fly, the signal would require 25 mbps, or 36 mbps with audio. Storing the signal on disk is hardly an alternative, considering that one hour would require 12 gigabytes.

When looking at the differences between streaming video (that is transmitted real-time) and storing video on disk, we may observe the following tradeoffs:

item	**streaming**	**downloaded**
bandwidth	equal to the display rate	may be arbitrarily small
disk storage	none	the entire file must be stored
startup delay	almost none	equal to the download time
resolution	depends on available bandwidth	depends on available disk storage

So, what are our options? Apart from the quite successful MPEG encodings, which have found their way in the DVD, there are a number of proprietary formats used for transmitting video over the internet: Quicktime, introduced by Apple, early 1990s, for local viewing; RealVideo, streaming video from RealNetworks; and Windows Media, a proprietary encoding scheme fromMicrosoft. Examples of these formats, encoded for various bitrates are available at Video at VU.

Apparently, there is some need for digital video on the internet, for example as propaganda for attracting students, for looking at news items at a time that suits you, and (now that digital video cameras become affordable) for sharing details of your family life.

Is digital video all there is? Certainly not! In the next section, we will deal with standards that allow for incorporating (streaming) digital video as an element in a compound multimedia presentation, possibly synchronized with other items, including synthetic graphics. Online, you will find some examples of digital video that are used as texture maps in 3D space. These examples are based on the technology presented in section **??**, and use the streaming video codec from Real Networks that is integrated as a rich media extension in the *blaxxun* Contact 3D VRML plugin.

comparison of codecs A review of codecs[3], including Envivio MPEG-4, QuickTime 6, RealNetworks 9 en Windows Media 9 was published januari 2005 by the European Broadcast Union[4]. It appeared that The Real Networks codecs came out best, closely followed by the Windows Media 9 result. Ckeck it out!

3

[3]www.ebu.ch/trev_301-samviq.pdf
[4]www.ebu.ch/trev_home.html

3.2 standards

Imagine what it would be like to live in a world without standards. You may get the experience when you travel around and find that there is a totally different socket for electricity in every place that you visit.

Now before we continue, you must realize that there are two types of standards: *de facto* market standards (enforced by sales politics) and committee standards (that are approved by some official organization). For the latter type of standards to become effective, they need consent of the majority of market players.

For multimedia on the web, we will discuss three standards and RM3D which was once proposed as a standard and is now only of historical significance.

standards

- XML – eXtensible Markup Language (SGML)
- MPEG-4 – coding audio-visual information
- SMIL – Synchronized Multimedia Integration Language
- RM3D – (Web3D) Rich Media 3D (extensions of X3D/VRML)

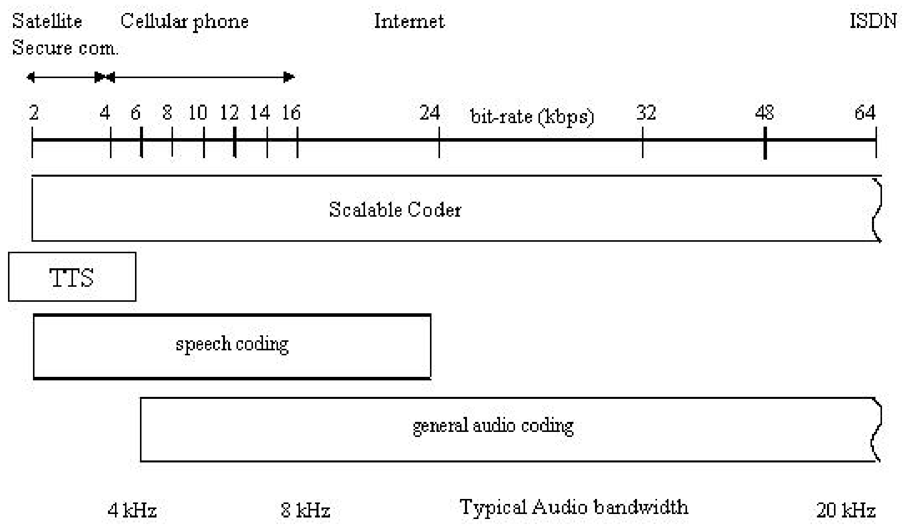
XML, the *eXtensible Markup Language*, is becoming widely accepted. It is being used to replace HTML, as well as a data exchange format for, for example, business-to-business transactions. XML is derived from SGML (Structured Generalized Markup Language) that has found many applications in document processing. As SGML, XML is a generic language, in that it allows for the specification of actual markup languages. Each of the other three standards mentioned allows for a syntactic encoding using XML.

MPEG-4 aims at providing "the standardized technological elements enabling the integration of production, distribution and content access paradigms of digital television, interactive graphics and multimedia", Koenen (2000). A preliminary version of the standard has been approved in 1999. Extensions in specific domains are still in progress.

SMIL, the *Synchronized Multimedia Integration Language*, has been proposed by the W3C "to enable the authoring of TV-like multimedia presentations, on the Web". The SMIL language is an easy to learn HTML-like language. SMIL presentations can be composed of streaming audio, streaming video, images, text or any other media type, W3C (2001). SMIL-1 has become a W3C recommendation in 1998. SMIL-2 is at the moment of writing still in a draft stage.

RM3D, *Rich Media 3D*, is not a standard as MPEG-4 and SMIL, since it does currently not have any formal status. The RM3D working group arose out of the X3D working group, that addressed the encoding of VRML97 in XML. Since there were many disagreements on what should be the core of X3D and how extensions accomodating VRML97 and more should be dealt with, the RM3D working group was founded in 2000 to address the topics of extensibility and the integration with rich media, in particular video and digital television.

remarks Now, from this description it may seem as if these groups work in total isolation from eachother. Fortunately, that is not true. MPEG-4, which is the most encompassing of these standards, allows for an encoding both in SMIL and X3D. The X3D and RM3D working groups, moreover, have advised the MPEG-4 commitee on how to integrate 3D scene description and human avatar animation in MPEG-4. And finally, there have been rather intense discussions between the SMIL and RM3D working groups on the timing model needed to control animation and dynamic properties of media objects.



4

MPEG-4

The MPEG standards (in particular 1,2 and 3) have been a great success, as testified by the popularity of mp3 and DVD video.

Now, what can we expect from MPEG-4? Will MPEG-4 provide *multimedia for our time*, as claimed in Koenen (1999). The author, Rob Koenen, is senior consultant at the dutch KPN telecom research lab, active member of the MPEG-4 working group and editor of the MPEG-4 standard document.

> *"Perhaps the most immediate need for MPEG-4 is defensive. It supplies tools with which to create uniform (and top-quality) audio and video encoders on the Internet, preempting what may become an unmanageable tangle of proprietary formats."*

Indeed, if we are looking for a general characterization it would be that MPEG-4 is primarily

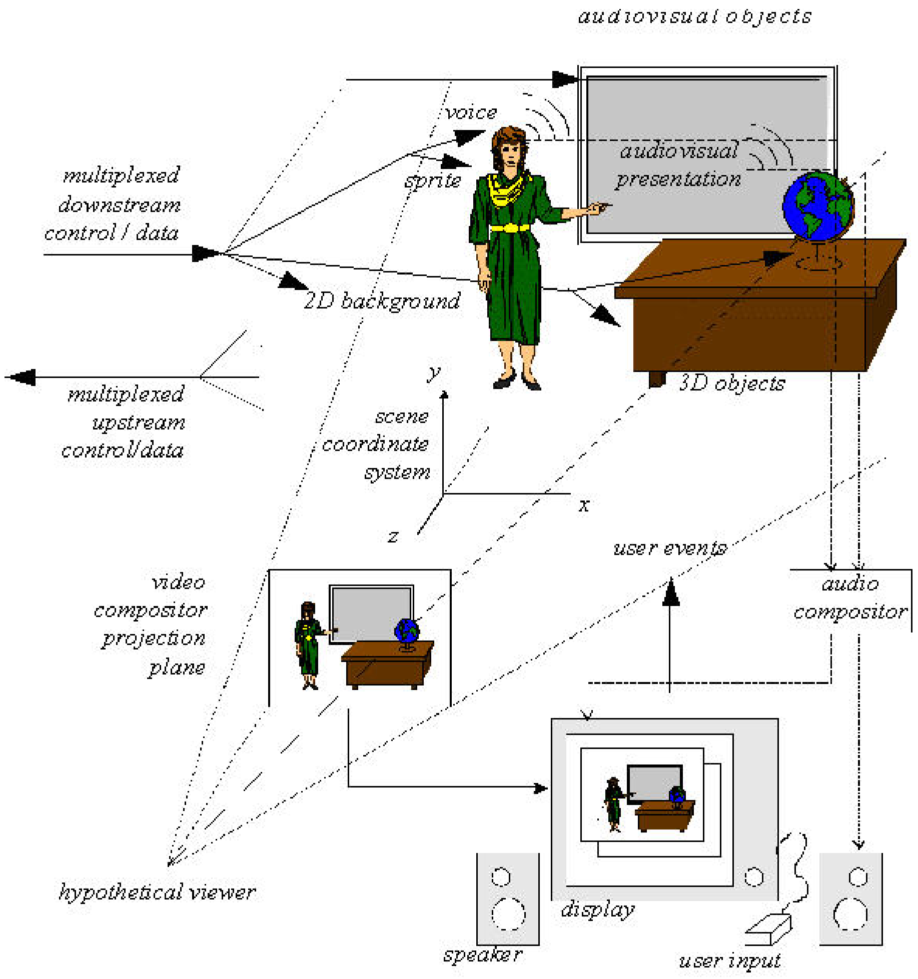
MPEG-4

a toolbox of advanced compression algorithms for audiovisual information and, moreover, one that is suitable for a variety of display devices and networks, including low bitrate mobile networks. MPEG-4 supports scalability on a variety of levels:

scalability

- *bitrate* – switching to lower bitrates
- *bandwidth* – dynamically discard data
- *encoder and decoder complexity* – signal quality

Dependent on network resources and platform capabilities, the 'right' level of signal quality can be determined by selecting the optimal codec, dynamically.



5

media objects It is fair to say that MPEG-4 is a rather ambitious standard. It aims at offering support for a great variety of audiovisual information, including still images, video, audio, text, (synthetic) talking heads and synthesized speech, synthetic graphics and 3D scenes, streamed data applied to media objects, and user interaction – e.g. changes of viewpoint.

Let's give an example, taken from the MPEG-4 standard document.

example

> *Imagine, a talking figure standing next to a desk and a projection screen, explaining the contents of a video that is being projected on the screen, pointing at a globe that stands on the desk. The user that is watching that scene decides to change from viewpoint to get a better look at the globe ...*

How would you describe such a scene? How would you encode it? And how would you approach decoding and user interaction?

The solution lies in defining *media objects* and a suitable notion of composition of media objects.

media objects

- *media objects* – units of aural, visual or audiovisual content
- *composition* – to create compound media objects (audiovisual scene)
- *transport* – multiplex and synchronize data associated with media objects
- *interaction* – feedback from users' interaction with audiovisual scene

For 3D-scene description, MPEG-4 builds on concepts taken from VRML (Virtual Reality Modeling Language, discussed in chapter 7).

Composition, basically, amounts to building a *scene graph*, that is a tree-like structure that specifies the relationship between the various simple and compound media objects. Composition allows for placing media objects anywhere in a given coordinate system, applying transforms to change the appearance of a media object, applying streamed data to media objects, and modifying the users viewpoint.

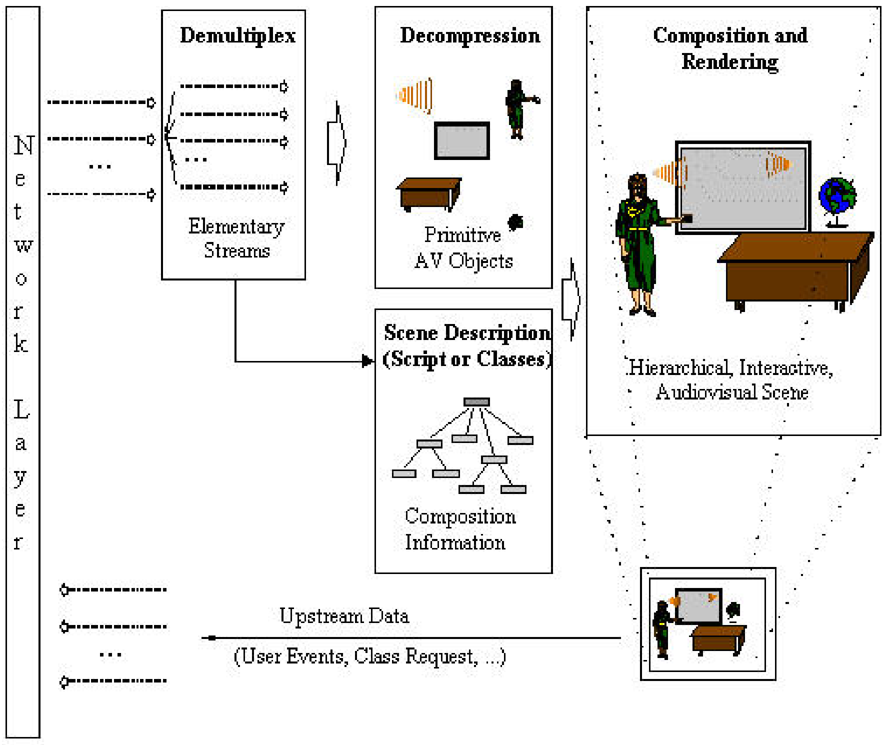
So, when we have a multimedia presentation or audiovisual scene, we need to get it accross some network and deliver it to the end-user, or as phrased in Koenen (2000):

transport

> *The data stream (Elementary Streams) that result from the coding process can be transmitted or stored separately and need to be composed so as to create the actual multimedia presentation at the receivers side.*

At a system level, MPEG-4 offers the following functionalities to achieve this:

- BIFS (Binary Format for Scenes) – describes spatio-temporal arrangements of (media) objects in the scene
- OD (Object Descriptor) – defines the relationship between the elementary streams associated with an object
- *event routing* – to handle user interaction



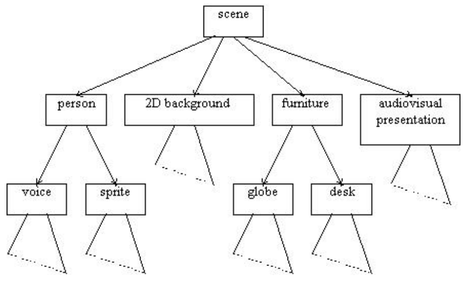
Network Layer
Demultiplex
Elementary Streams
Decompression
Primitive AV Objects
Scene Description (Script or Classes)
Composition Information
Composition and Rendering
Hierarchical, Interactive, Audiovisual Scene
Upstream Data
(User Events, Class Request, ...)

6

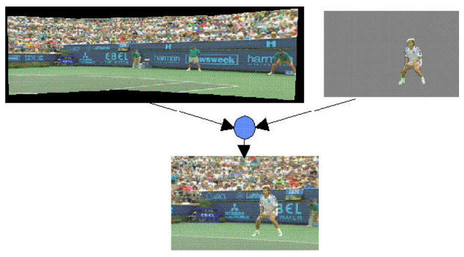
In addition, MPEG-4 defines a set of functionalities For the delivery of streamed data, DMIF, which stands for

Delivery Multimedia Integration Framework

that allows for transparent interaction with resources, irrespective of whether these are available from local storage, come from broadcast, or must be obtained from some remote site. Also transparency with respect to network type is supported. *Quality of Service* is only supoorted to the extent that it ispossible to indicate needs for bandwidth and transmission rate. It is however the responsability of the network provider to realize any of this.



(a) scene graph



(b) sprites

7

authoring What MPEG-4 offers may be summarized as follows

benefits

- *end-users* – interactive media accross all platforms and networks
- *providers* – transparent information for transport optimization
- *authors* – reusable content, protection and flexibility

In effect, although MPEG-4 is primarily concerned with efficient encoding and scalable transport and delivery, the *object-based* approach has also clear advantages from an authoring perspective.

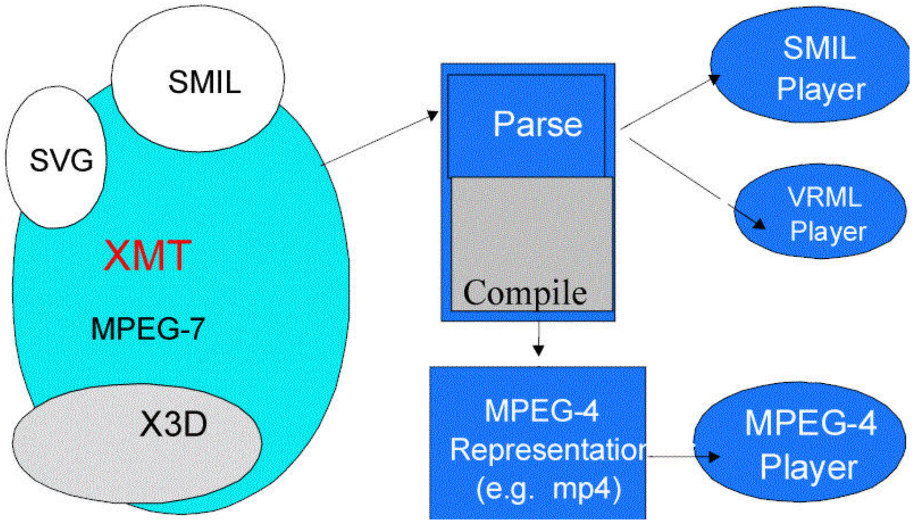
One advantage is the possibility of reuse. For example, one and the same background can be reused for multiplepresentations or plays, so you could imagine that even an amateur game might be 'located' at the centre-court of Roland Garros or Wimbledon.

Another, perhaps not so obvious, advantage is that provisions have been made for

managing intellectual property

of media objects.

And finally, media objects may potentially be annotated with meta-information to facilitate information retrieval.



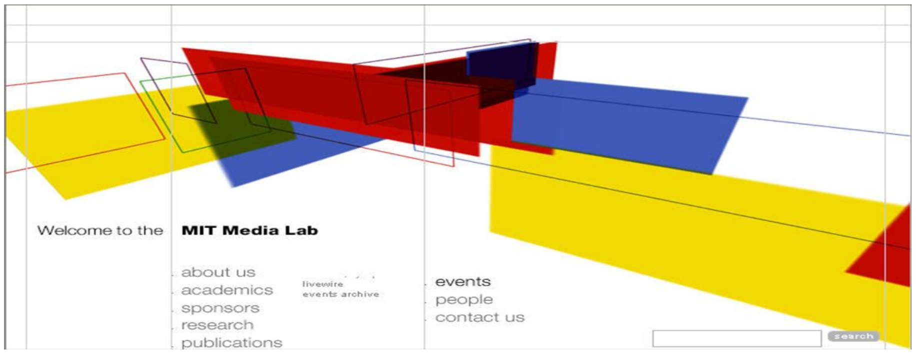
8

syntax In addition to the binary formats, MPEG-4 also specifies a syntactical format, called XMT, which stands for *eXtensible MPEG-4 Textual format*.

XMT

- XMT contains a subset of X3D
- SMIL is mapped (incompletely) to XMT

when discussing RM3D which is of interest from a historic perspective, we will further establish what the relations between, respectively MPEG-4, SMIL and RM3D are, and in particular where there is disagreement, for example with respect to the timing model underlying animations and the temporal control of media objects.



9

example(s) – *structured audio*

The Machine Listening Group[5] of the MIT Media Lab[6] is developing a suite of tools for *structered audio*, which means *transmitting sound by describing it rather than compressing it.* It is claimed that tools based on the MPEG-4 standard will be the future platform for computer music, audio for gaming, streaming Internet radio, and other multimedia applications.

The structured audio project is part of a more encompassing research effort of the Music, Mind and Machine Group[7] of the MIT Media Lab, which *envisages a new future of audio technologies and interactive applications that will change the way music is conceived, created, transmitted and experienced*,

SMIL

SMIL is pronounced as *smile.* SMIL, the Synchronized Multimedia Integration Language, has been inspired by the Amsterdam Hypermedia Model (AHM). In fact, the dutch research group at CWI that developed the AHM actively participated in the SMIL 1.0 committee. Moreover, they have started a commercial spinoff to create an editor for SMIL, based on the editor they developed for CMIF. The name of the editor is GRINS. Get it?

As indicated before SMIL is intended to be used for

TV-like multimedia presentations

The SMIL language is an XML application, resembling HTML. SMIL presentations can be written using a simple text-editor or any of the more advanced tools, such as GRINS. There is a variety of SMIL players. The most wellknown perhaps is the RealNetworks G8 players, that allows for incorporating RealAudio and RealVideo in SMIL presentations.

parallel and sequential

> *Authoring a SMIL presentation comes down, basically, to name media components for text, images,audio and video with URLs, and to schedule their presentation either in parallel or in sequence.*

Quoting the SMIL 2.0 working draft, we can characterize the SMIL presentation characteristics as follows:

presentation characteristics

- The presentation is composed from several components that are accessible via URL's, e.g. files stored on a Web server.
- The components have different media types, such as audio, video, image or text. The begin and end times of different components are specified relative to events in other media components. For example, in a slide show, a particular slide is displayed when the narrator in the audio starts talking about it.

[5]sound.media.mit.edu/mpeg4
[6]www.media.mit.edu
[7]sound.media.mit.edu

- Familiar looking control buttons such as stop, fast-forward and rewind allow the user to interrupt the presentation and to move forwards or backwards to another point in the presentation.
- Additional functions are "random access", i.e. the presentation can be started anywhere, and "slow motion", i.e. the presentation is played slower than at its original speed.
- The user can follow hyperlinks embedded in the presentation.

Where HTML has become successful as a means to write simple hypertext content, the SMIL language is meant to become a vehicle of choice for writing *synchronized hypermedia*. The working draft mentions a number of possible applications, for example a photoalbun with spoken comments, multimedia training courses, product demos with explanatory text, timed slide presentations, onlime music with controls.

As an example, let's consider an interactive news bulletin, where you have a choice between viewing a weather report or listening to some story about, for example, the decline of another technology stock. Here is how that could be written in SMIL:

example

```
<par>
   <a href=" #Story"> <img src="button1.jpg"/> </a>
   <a href=" #Weather"> <img src="button2.jpg"/></a>
    <excl>
        <par id="Story" begin="0s">
          <video src="video1.mpg"/>
          <text src="captions.html"/>
        </par>

        <par id="Weather">
          <img src="weather.jpg"/>
          <audio src="weather-rpt.mp3"/>
        </par>
    </excl>
</par>
```

Notice that there are two *parallel* (PAR) tags, and one *exclusive* (EXCL) tag. The *exclusive* tag has been introduced in SMIL 2.0 to allow for making an exclusive choice,so that only one of the items can be selected at a particular time. The SMIL 2.0 working draft defines a number of elements and attributes to control presentation, synchronization and interactivity, extending the functionality of SMIL 1.0.

Before discussing how the functionality proposed in the SMIL 2.0working draft may be realized, we might reflect on how to position SMIL with respect to the many other approaches to provide multimedia on the web. As other approaches we may think of *flash*, dynamic HTML (using javascript), or java applets. In the SMIL 2.0 working draft we read the following comment:

history

> *Experience from both the CD-ROM community and from the Web multimedia community suggested that it would be beneficial to adopt a declarative format for expressing media synchronization on the Web as an alternative and complementary approach to scripting languages.*
>
> *Following a workshop in October 1996, W3C established a first working group on synchronized multimedia in March 1997. This group focused on the design of a declarative language and the work gave rise to SMIL 1.0 becoming a W3C Recommendation in June 1998.*

In summary, SMIL 2.0 proposes a *declarative format* to describe the temporal behavior of a multimedia presentation, associate hyperlinks with media objects, describe the form of the presentation on a screen, and specify interactivity in multimedia presentations. Now,why such a fuzz about "declarative format"? Isn't scripting more exciting? And aren't the tools more powerful? Ok, ok. I don't want to go into that right now. Let's just consider a *declarative format* to be more elegant. Ok?

To support the functionality proposed for SMIL 2.0 the working draft lists a number of modules that specify the interfaces for accessing the attributes of the various elements. SMIL 2.0 offers modules for animation, content control, layout, linking, media objects, meta information, timing and synchronization, and transition effects.

This modular approach allows to reuse SMIL syntax and semantics in other XML-based languages, in particular those that need to represent timing and synchronization. For example:

module-based reuse

- SMIL modules could be used to provide lightweight multimedia functionality on mobile phones, and to integrate timing into profiles such as the WAP forum's WML language, or XHTML Basic.
- SMIL timing, content control, and media objects could be used to coordinate broadcast and Web content in an enhanced-TV application.
- SMIL Animation is being used to integrate animation into W3C's Scalable Vector Graphics language (SVG).
- Several SMIL modules are being considered as part of a textual representation for MPEG4.

The SMIL 2.0 working draft is at the moment of writing being finalized. It specifies a number of language profiles topromote the reuse of SMIL modules. It also improves on the accessibility features of SMIL 1.0, which allows for, for example,, replacing captions by audio descriptions.

In conclusion, SMIL 2.0 is an interesting standard, for a number of reasons. For one, SMIL 2.0 has solid theoretical underpinnings in a well-understood, partly formalized, hypermedia model (AHM). Secondly, it proposes interesting functionality, with which authors can make nice applications. In the third place, it specifies a high level declarative format, which is both expressive and flexible. And finally, it is an open standard (as opposed to proprietary standard). So everybody can join in and produce players for it!

10

RM3D – not a standard

The web started with simple HTML hypertext pages. After some time static images were allowed. Now, there is support for all kinds of user interaction, embedded multimedia and even synchronized hypermedia. But despite all the graphics and fancy animations, everything remains flat. Perhaps surprisingly, the need for a 3D web standard arose in the early days of the web. In 1994, the acronym VRML was coined by Tim Berners-Lee, to stand for *Virtual Reality Markup Language*. But, since 3D on the web is not about text but more about worlds, VRML came to stand for *Virtual Reality Modeling Language*. Since 1994, a lot of progress has been made.

www.web3d.org

- VRML 1.0 – *static 3D worlds*
- VRML 2.0 or VRML97 – *dynamic behaviors*
- VRML200x – *extensions*
- X3D – *XML syntax*
- RM3D – *Rich Media in 3D*

In 1997, VRML2 was accepted as a standard, offering rich means to create 3D worlds with dynamic behavior and user interaction. VRML97 (which is the same as VRML2) was, however, not the success it was expected to be, due to (among others) incompatibility between browsers, incomplete implementations of the standards, and high performance requirements.

As a consequence, the Web3D Consortium (formerly the VRML Consortium) broadened its focus, and started thinking about extensions or modifications of VRML97 and an XML version of VRML (X3D). Some among the X3D working group felt the need to rethink the premisses underlying VRML and started the Rich Media Working Group:

groups.yahoo.com/group/rm3d/

The Web3D Rich Media Working Group was formed to develop a Rich Media standard format (RM3D) for use in next-generation media devices. It is a highly active group with participants from a broad range of companies including 3Dlabs, ATI, Eyematic, OpenWorlds, Out of the Blue Design, Shout Interactive, Sony, Uma, and others.

In particular:

RM3D

> *The Web3D Consortium initiative is fueled by a clear need for a standard high performance Rich Media format. Bringing together content creators with successful graphics hardware and software experts to define RM3D will ensure that the new standard addresses authoring and delivery of a new breed of interactive applications.*

The working group is active in a number of areas including, for example, multi-texturing and the integration of video and other streaming media in 3D worlds.

Among the driving forces in the RM3D group are Chris Marrin and Richter Rafey, both from Sony, that proposed *Blendo*, a rich media extension of VRML. Blendo has a strongly typed object model, which is much more strictly defined than the VRML object model, to support both declarative and programmatic extensions. It is interesting to note that the premisse underlying the Blendo proposal confirms (again) the primacy of the TV metaphor. That is to say, what Blendo intends to support are TV-like presentations which allow for user interaction such as the selection of items or playing a game. Target platforms for Blendo include graphic PCs, set-top boxes, and the Sony Playstation!

11

requirements The focus of the RM3D working group is not *syntax* (as it is primarily for the X3D working group) but *semantics*, that is to enhance the VRML97 standard to effectively incorporate rich media. Let's look in more detail at the requirements as specified in the RM3Ddraft proposal.

requirements

- *rich media* – audio, video, images, 2D & 3D graphics (with support for temporal behavior, streaming and synchronisation)
- *applicability* – specific application areas, as determined by commercial needs and experience of working group members

The RM3D group aims at interoperability with other standards.

- *interoperability* – VRML97, X3D, MPEG-4, XML (DOM access)

In particular, an XML syntax is being defined in parallel (including interfaces for the DOM). And, there is mutual interest and exchange of ideas between the MPEG-4 and RM3D working group.

As mentioned before, the RM3D working group has a strong focus on defining an object model (that acts as a common model for the representation of objects and their capabilities) and suitable mechanisms for extensibility (allowing for the integration of new objects defined in Java or C++, and associated scripting primitives and declarative constructs).

Notice that extensibility also requires the definition of a declarative format, so that the content author need not bother with programmatic issues.

The RM3D proposal should result in effective 3D media presentations. So as additional requirements we may, following the working draft, mention: high-quality realtime rendering, for realtime interactive media experiences; platform adaptability, with query functions for programmatic behavior selection; predictable behavior, that is a well-defined order of execution; a high precision number systems, greater than single-precision IEEE floating point numbers; and minimal size, that is both download size and memory footprint.

Now, one may be tempted to ask how the RM3D proposals is related to the other standard proposals such as MPEG-4 and SMIL, discussed previously. Briefly put, paraphrased from one of Chris Marrin's messages on the RM3D mailing list

SMIL is closer to the author and *RM3D is closer to the implementer.*

MPEG-4, in this respect is even further away from the author since its chief focus is on compression and delivery across a network.

RM3D takes 3D scene description as a starting point and looks at pragmatic ways to integrate rich media. Since 3D is itself already computationally intensive, there are many issues thatarise in finding efficient implementations for the proposed solutions.

12

timing model RM3D provides a declarative format formany interesting features, such as for example texturing objects with video. In comparison to VRML, RM3D is meant to provide more temporal control over time-based media objects and animations. However, there is strong disagreement among the working group members as to what time model the dynamic capabilities of RM3D should be based on. As we read in the working draft:

working draft

Since there are three vastly different proposals for this section (time model), the original <RM3D> 97 text is kept. Once the issues concerning time-dependent nodes are resolved, this section can be modified appropriately.

Now, what are the options? Each of the standards discussed to far provides us with a particular solution to timing. Summarizing, we have a time model based on a spring metaphor in MPEG-4, the notion of cascading time in SMIL (inspired by cascading stylesheets for HTML) and timing based on the routing of events in RM3D/VRML.

The MPEG-4 standard introduces the *spring metaphor* for dealing with temporal layout.

MPEG-4 – spring metaphor

- duration – minimal, maximal, optimal

The spring metaphor amounts to the ability to shrink or stretch a media object within given bounds (minimum, maximum) to cope with, for example, network delays.

The SMIL standard is based on a model that allows for propagating durations and time manipulations in a hierarchy of media elements. Therefore it may be referred to as a cascading modelof time.

SMIL – cascading time

- time container – speed, accelerate, decelerate, reverse, synchronize

Media objects, in SMIL, are stored in some sort of container of which the timing properties can be manipulated.

```
<seq speed="2.0">
   <video src="movie1.mpg" dur="10s"/>
   <video src="movie2.mpg" dur="10s"/>
   <img src="img1.jpg" begin="2s" dur="10s">
               <animateMotion from="-100,0" to="0,0" dur="10s"/>
   </img>
   <video src="movie4.mpg" dur="10s"/>
</seq>
```

In the example above,we see that the speed is set to *2.0*, which will affect the pacing of each of the individual media elements belonging to that (sequential) group. The duration of each of the elements is specified in relation to the parent container. In addition, SMIL offers the possibility to synchronize media objects to control, for example, the end time of parallel media objects.

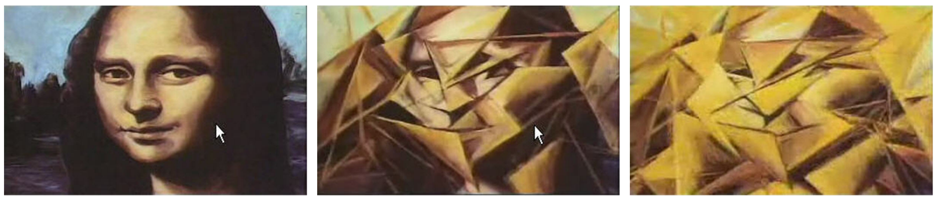
VRML97's capabilities for timing rely primarily on the existence of a *TimeSensor* thatsends out time events that may be routed to other objects.

RM3D/VRML – event routing

- *TimeSensor* – isActive, start, end, cycleTime, fraction, loop

When a *TimeSensor* starts to emit time events, it also sends out an event notifying other objects that it has become active. Dependent on itsso-called *cycleTime*, it sends out the fraction it covered since it started. This fraction may be send to one of the standard interpolators or a script so that some value can be set, such as for example the orientation, dependent on the fraction of the time intercal that has passed. When the *TimeSensor* is made to loop, this is done repeatedly. Although time in VRML is absolute, the frequency with which fraction events are emitted depends on the implementation and processor speed.

Lacking consensus about a better model, this model has provisionally been adopted, with some modifications, for RM3D. Nevertheless, the SMIL cascading time model has raised an interest in the RM3D working group, to the extent that Chris Marrin remarked (in the mailing list) *"we could go to school here"*. One possibility for RM3D would be to introduce *time containers* that allow for a temporal transform of their children nodes, in a similar way as grouping containers allow for spatial transforms of their children nodes. However, that would amount to a dual hierarchy, one to control (spatial) rendering and one to control temporal characteristics. Merging the two hierarchies, as is (implicitly) the case in SMIL, might not be such a good idea, since the rendering and timing semantics of the objects involved might be radically different. An interesting problem, indeed, but there seems to be no easy solution.



13

example(s) – *rich internet applications*

In a seminar held by *Lost Boys*, which is a dutch subdivison if Icon Media Lab[8], *rich internet applications* (RIA), were presented as the new solutions to present applications on the web. As indicated by Macromedia[9], who is one of the leading companies in this fiwld, *experience matters*, and so plain html pages pages do not suffice since they require the user to move from one page to another in a quite unintuitive fashion. Macromedia presents its new line of *flash*-based products to create such *rich internet applications*. An alternative solution, based on general W3C recommendations, is proposed by BackBase[10]. Interestingly enough, using either technology, many of the paricipants of the seminar indicated a strong preference for a backbuuton, having similar functionality as the often used backbutton in general internet browsers.

[8]www.iconmedialab.com

[9]www.macromedia.com/resources/business/rich_internet_apps/whitepapers.html

[10]www.backbase.com

research directions– *meta standards*

All these standards! Wouldn't it be nice to have one single standard that encompasses them all? No, it would not! Simply, because such a standard is inconceivable, unless you take some proprietary standard or a particular platform as the defacto standard (which is the way some people look at the Microsoft win32 platform, ignoring the differences between 95/98/NT/2000/XP/...). In fact, there is a standard that acts as a glue between the various standards for multimedia, namely XML. XML allows for the interchange of data between various multimedia applications, that is the transformation of one encoding into another one. But this is only syntax. What about the semantics?

Both with regard to delivery and presentation the MPEG-4 proposal makes an attempt to delineate chunks of core fuctionality that may be shared between applications. With regard to presentation, SMIL may serve as an example. SMIL applications themselves already (re)use functionality from the basic set of XML-related technologies, for example to access the document structure through the DOM (Document Object Model). In addition, SMIL defines components that it may potentially share with other applications. For example, SMIL shares its animation facilities with SVG (the Scalable Vector Graphics format recommended by the Web Consortium).

The issue in sharing is, obviously, how to relate constructs in the syntax to their operational support. When it is possible to define a common base of operational support for a variety of multimedia applications we would approach our desired meta standard, it seems. A partial solution to this problem has been proposed in the now almost forgotten HyTime standard for time-based hypermedia. HyTime introduces the notion of *architectural forms* as a means to express the operational support needed for the interpretation of particular encodings, such as for example synchronization or navigation over bi-directional links. Apart from a base module, HyTime compliant architectures may include a units measurement module, a module for dealing with location addresses, a module to support hyperlinks, a scheduling module and a rendition module.

To conclude, wouldn't it be wonderful if, for example, animation support could be shared between rich media X3D and SMIL? Yes, it would! But as you may remember from the discussion on the timing models used by the various standards, there is still to much divergence to make this a realoistic option.

14

3.3 a multimedia semantic web?

To finish this chapter, let's reflect on where we are now with 'multimedia' on the web. Due to refined compression schemes and standards for authoring and delivery, we seemed to have made great progress in realizing *networked multimedia.* But does this progress match what has been achieved for the dominant media type of the web, that is text or more precisely textual documents with markup?

web content

- *1st generation* – hand-coded HTML pages
- *2nd generation* – templates with content and style
- *3rd generation* – rich markup with metadata (XML)

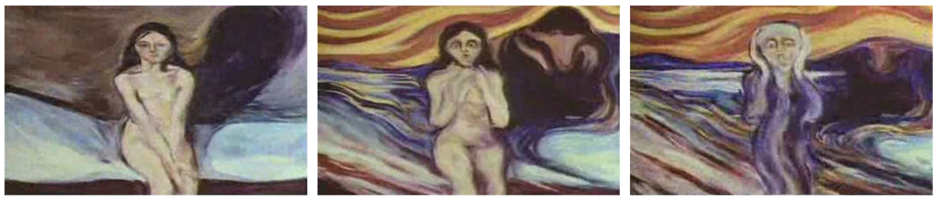
Commonly, a distinction is made between successive generations of web content, with the first generation being simple hand-coded HTML pages. The second generation may be characterized as HTML pages that are generated on demand, for example by filling in templates with contents retrieved from a database. The third generation is envisaged to make use of rich markup, using XML, that reflects the (semantic) content of the document more directly, possibly augmented with (semantic) meta-data that describe the content in a way that allows machines, for example search engines, to process it. The great vision underlying the third generation of web content is commonly refered to as the *the semantic web.* which enhances the functionality of the current web by deploying knowledge representation and inference technology from Artificial Intelligence, using a technology known as the *Resource Description Framework* (RDF). As phrased in Ossenbruggen et. al. (2001), the semantic web will bring

structure to the meaningful content of web pages,

thus allowing computer programs,such as search engines and intelligent agents, to do their job more effectively. For search engines this means more effective information retrieval, and for agents better opportunities to provide meaningful services.

A great vision indeed. So where are we with multimedia? As an example, take a *shockwave* or *flash* presentation showing the various musea in Amsterdam. How would you attach meaning to it, so that it might become an element of a semantic structure? Perhaps you wonder what meaning could be attached to it? That should not be too difficult to think of. The (meta) information attached to such a presentation should state (minimally) that the location is Amsterdam, that the sites of interest are musea, and (possibly) that the perspective is touristic. In that way, when you search for touristic information about musea in Amsterdam, your search engine should have no trouble in selecting that presentation. Now, the answer to the question how meaning can be attached to a presentation is already given, namely by specifying meta-information in some format (of which the only requirement is that it is machine-processable). For our *shockwave*or *flash* presentation we cannot dothis in a straightforward manner. But for MPEG-4 encoded material, as well as for SMIL content, such facilities are readily available.

Should we then always duplicate our authoring effort by providing (meta) information, on top of the information that is already contained in the presentation? No, in some cases, we can also rely to some extent on content-based search or feature extraction, as will be discussed in the following chapters.



15

Resource Description Framework – the Dublin Core

The Resource Description Framework, as the W3C/RDF[11] site informs us *integrates a variety of applications from library catalogs and world-wide directories to syndication and aggregation of news, software, and content to personal collections of music, photos, and events using XML as an interchange syntax.* The RDF specifications provide, in addition *a lightweight ontology system to support the exchange of knowledge on the Web.*

The Dublin Core Metadata Initiative[12] is an open forum engaged in the development of interoperable online metadata standards that support a broad range of purposes and business models.

What exactly is meta-data? As phrased in the RDF Primer[13]

meta data

> Metadata is data about data. Specifically, the term refers to data used to identify, describe, or locate information resources, whether these resources are physical or electronic. While structured metadata processed by computers is relatively new, the basic concept of metadata has been used for many years in helping manage and use large collections of information. Library card catalogs are a familiar example of such metadata.

The Dublin Core proposes a small number of elements, to be used to give information about a resource, such as an electronic document on the Web. Consider the following example:

Dublin Core example

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns #"
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:dcterms="http://purl.org/dc/terms/">
```

[11]www.w3.org/RDF
[12]dublincore.org
[13]www.w3.org/TR/rdf-primer

```
<rdf:Description rdf:about="http://www.dlib.org/dlib/may98/miller/05miller.html">
 <dc:title>An Introduction to the Resource Description Framework</dc:title>
   <dc:creator>Eric J. Miller</dc:creator>
   <dc:description>The Resource Description Framework (RDF) is an
    infrastructure that enables the encoding, exchange and reuse of
    structured metadata. rdf is an application of xml that imposes needed
    structural constraints to provide unambiguous methods of expressing
    semantics. rdf additionally provides a means for publishing both
    human-readable and machine-processable vocabularies designed to
    encourage the reuse and extension of metadata semantics among
    disparate information communities. the structural constraints rdf
    imposes to support the consistent encoding and exchange of
    standardized metadata provides for the interchangeability of separate
    packages of metadata defined by different resource description
    communities. </dc:description>
 <dc:publisher>Corporation for National Research Initiatives</dc:publisher>
   <dc:subject>
     <rdf:Bag>
       <rdf:li>machine-readable catalog record formats</rdf:li>
       <rdf:li>applications of computer file organization and
        access methods</rdf:li>
     </rdf:Bag>
   </dc:subject>
   <dc:rights>Copyright  1998 Eric Miller</dc:rights>
   <dc:type>Electronic Document</dc:type>
   <dc:format>text/html</dc:format>
   <dc:language>en</dc:language>
 <dcterms:isPartOf rdf:resource="http://www.dlib.org/dlib/may98/05contents.html"/>
 </rdf:Description>
</rdf:RDF>
```

Items such as *title*, *creator*, *subject* and *description*, actually all tags with the prefix *dc*, belong to the Dublin Core and are used to give information about the document, which incidentally concerns an introduction to the Resource Description Framework. The example also shows how *rdf* constructs can be used together with the Dublin Core elements. The prefixes *rdf* and *dc* are used to distinguish between the distinct namespaces of respectively RDF and the Dublin Core.

The Dublin Core contains the following elements:

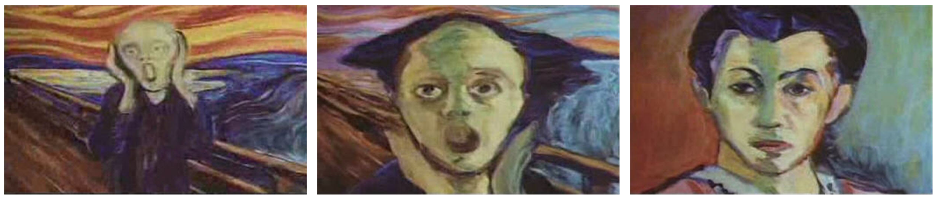
Dublin Core[14]

- *title* – name given to the resource
- *creator* – entity primarily responsible for making the content of the resource
- *subject* – topic of the content of the resource
- *description* – an account of the content of the resource
- *publisher* – entity responsible for making the resource available

[14]dublincore.org/documents/dces

- *contributor* – entity responsible for making contributions to the content of the resource
- *date* – date of an event in the lifecycle of the resource
- *type* – nature or genre of the content of the resource
- *format* – physical or digital manifestation of the resource
- *identifier* – unambiguous reference to the resource within a given context
- *source* – reference to a resource from which the present resource is derived
- *language* – language of the intellectual content of the resource
- *relation* – reference to a related resource
- *coverage* – extent or scope of the content of the resource
- *rights* – information about rights held in and over the resource

In section 10.3 we discuss an application in the domain of cultural heritage, where the Dublin Core elements are used to provide meta information about the information available for the conservation of contemporary artworks.



16

research directions– ***agents everywhere***

The web is an incredibly rich resource of information. Or, as phrased in Baeza-Yates and Ribeiro-Neto (1999):

information repository

> *The Web is becoming a universal repository of human knowledge and culture, which has allowed unprecedented sharing of ideas and information in a scale never seen before.*

Now, the problem (as many of you can acknowledge) is to get the information out of it. Of course, part of the problem is that we often do not know what we are looking for. But even if we do know, it is generally not so easy to find our way. Again using the phrasing of Baeza-Yates and Ribeiro-Neto (1999):

browsing & navigation

> *To satisfy his information need, the user might navigate the hyperspace of web links searching for information of interest. However, since the hyperspace is vast and almost unknown, such a navigation task is usually inefficient.*

The solution of the problem of *getting lost in hyperspace* proposed in Baeza-Yates and Ribeiro-Neto (1999) is information retrieval, in other words *query & search*. However, this may not so easily be accomplished. As observed in Baeza-Yates and Ribeiro-Neto (1999), The main obstacle is the absence of a well-defined data model for the Web, which implies that information definition and structure is frequently of low quality. Well, that is exactly the focus of the semanics web initiative, and in particular of the Resource Description Framework discussed above.

Standardizing knowledge representation and reasoning about web resources is certainly one (important) step. Another issue, however, is how to support the user in finding the proper resources and provide the user with assistance in accomplishing his task (even if this task is merely finding suitable entertainment).

What we need, in other words, is a unifying model (encompassing both a data model and a model of computation) that allows us to deal effectively with web resources, including multimedia objects. For such a model, we may look at another area of research and development, namely *intelligent agtents*, which provides us not only with a model but also with a suitable metaphor and the technology, based on and extending object-oriented technology, to realize intelligent assistance, Eliens (2000).

For convenience, we make a distinction between two kinds of agents, *information agents* and *presentation agents*.

information agent

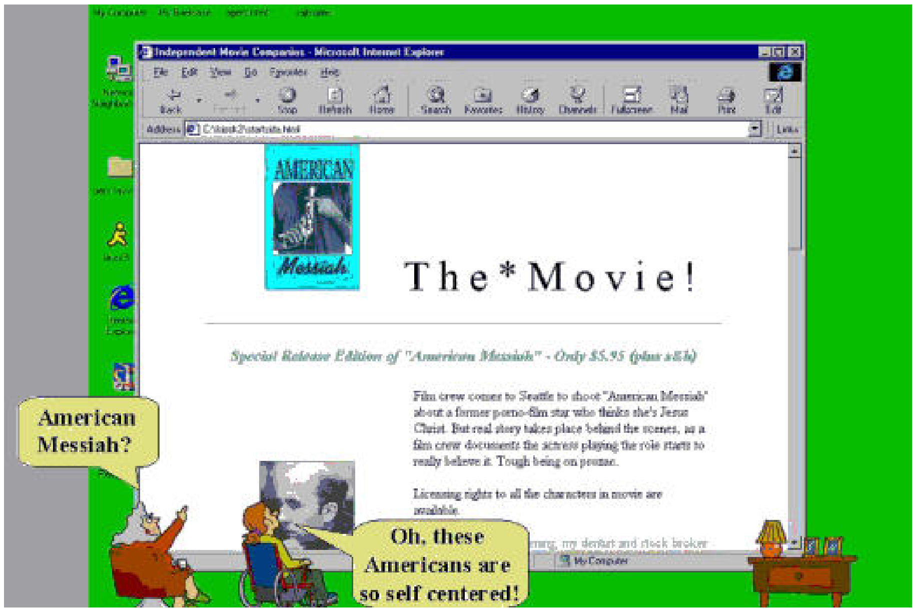
- gather information
- filter and select

Information agents are used to gather information. In addition, they filter the information and select those items that are relevant for the user. A key problem in developing information agents, however, is to find a proper representation of what the user considers to be relevant.

presentation agent

- access information
- find suitable mode of presentation

Complementary to the information agent is a *presentation agent* (having access to the information gathered) that displays the relevant information in a suitable way. Such a presentation agent can have many forms. To appetize your phantasy, you may look at the vision of *angelic guidance* presented in Broll et. al (2001). More concretely, my advice is to experiment with embodied agents that may present information in rich media 3D. In section **??**, we will present a framework for doing such experiments.


Independent Movie Companies - Microsoft Internet Explorer
American Messiah?
Oh, these Americans are so self centered!

AMERICAN Messiah

The*Movie!

Special Release Edition of "American Messiah" - Only $5.95 (plus s&h)

Film crew comes to Seattle to shoot "American Messiah" about a former porno-film star who thinks she's Jesus Christ. But real story takes place behind the scenes, as a film crew documents the actress playing the role starts to really believe it. Tough being on promac.

Licensing rights to all the characters in movie are available.

17

navigating information spaces Having *agents everywhere* might change our perspective on computing. But, it may also become quite annoying to be bothered by an agent each time that you try to interact with with your computer (you know what I mean!). However, as reported by Kristina Höök, even annoyance can be instrumental in keeping your attention to a particular task. In one of her projects, the *PERSONAS* project, which stands for

> *PERsonal and SOcial NAvigation through information spaceS*

the use of agents commenting on people navigating information space(s) is explored. As a note, the plural form of *spaces* is mine, to do justice to the plurality of information spaces.

As explained on the *PERSONAS* web site, which is listed with the acronyms, the *PERSONAS* project aims at:

PERSONAS

> *investigating a new approach to navigation through information spaces, based on a personalised and social navigational paradigm.*

The novel idea pursued in this project is to have agents (*Agneta* and *Frieda*) that are not helpful, but instead just give comments, sometimes with humor, but sometimes ironic or even sarcastic comments on the user's activities, in particular navigating an information space or (plain) web browsing. As can be read on the *PERSONAS* web site:

Agneta & Frieda

> *The AGNETA & FRIDA system seeks to integrate web-browsing and narrative into a joint mode. Below the browser window (on the desktop) are placed two female characters, sitting in their livingroom chairs, watching the browser during the session (more or less like watching television). Agneta and Frida (mother and daughter) physically react, comment, make ironic remarks about and develop stories around the information presented in the browser (primarily to each other), but are also sensitive to what the navigator is doing and possible malfunctions of the browser or server.*

In one of her talks, Kristina Höök observed that some users get really fed up with the comments delivered by *Agneta* and *Frieda*. So, as a compromise, the level of interference can be adjusted by the user, dependent on the task at hand.

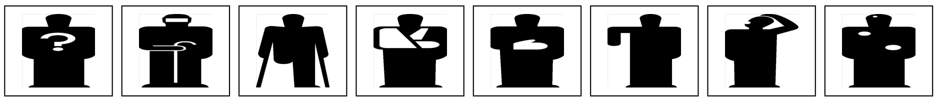
Agneta & Frieda

> *In this way they seek to attach emotional, comical or anecdotal connotations to the information and happenings in the browsing session. Through an activity slider, the navigator can decide on how active she wants the characters to be, depending on the purpose of the browsing session (serious information seeking, wayfinding, exploration or entertainment browsing).*

As you may gather, looking at the presentations accompanying this *introduction to multimedia* and Dialogs, I found the *PERSONAS* approach rather intriguing. Actually, the *PERSONAS* approach is related to the area of *affective computing*, see Picard (1998), which is an altogether different story.

The *Agneta* and *Frieda* software is available for download at the *PERSONAS* web site.

3.4 gluing it all together



18

questions

codecs and standards

1. (*) What role do standards play in *multimedia*? Why are standards necessary for compression and delivery. Discuss the MPEG-4 standard and indicate how it is related to other (possible) standards.

concepts

2. What is a *codec*?
3. Give a brief overview of current multimedia standards.

4. What criteria must a *(multimedia) semantic web* satisfy?

technology

5. What is the *data rate* for respectively (*compressed*) voice, audio and video?
6. Explain how a *codec* functions.
7. Which considerations can you mention for choosing a compression method?
8. Give a brief description of: XML, MPEG-4, SMIL, RM3D.

projects & further reading As a project, you may think of implementing for example JPEG compression, following Li and Drew (2004), or a SMIL-based application for cultural heritage.

You may further explore the technical issues on authoring DV material, using any of the Adobe[15], mentioned in appendix E. or compare

For further reading I advice you to take a look at the respective specifications of MPEG-4 and SMIL[16], and compare the functionality of MPEG-4 and SMIL-based presentation environments. An invaluable book dealing with the many technical aspects of compression and standards in Li and Drew (2004).

the artwork

1. costume designs – photographed from *Die Russchische Avantgarde und die Buhne 1890-1930*
2. theatre scene design, also from (above)
3. dance Erica Russel, Wiedermann (2004)
4. MPEG-4 – bits rates, from Koenen (2000).
5. MPEG-4 – scene positioning, from Koenen (2000).
6. MPEG-4 – up and downstream data, from Koenen (2000).
7. MPEG-4 – left: scene graph; right: sprites, from Koenen (2000).
8. MPEG-4 – syntax, from Koenen (2000).
9. MIT Media Lab[17] web site.
10. student work – *multimedia authoring I*, dutch windmill.
11. student work – *multimedia authoring I*, Schröder house.
12. student work – *multimedia authoring I*, train station.
13. animation – Joan Gratch, from Wiedermann (2004).
14. animation – Joan Gratch, from Wiedermann (2004).
15. animation – Joan Gratch, from Wiedermann (2004).
16. animation – Joan Gratch, from Wiedermann (2004).
17. Agneta and Frieda example.
18. signs – people, van Rooijen (2003), p. 246, 247.

[15]www.adobe.com/tutorials
[16]www.w3c.org/AudioVideo
[17]medai.mit.edu

Both the costume designs and theatre scene designs of the russian avantgarde movement are *expressionist* in nature. Yet, they show humanity and are in their own way very humorous. The dance animation by Erica Russell, using basic shapes and rhythms to express the movement of dance, is to some extent both solemn and equally humorous. The animations by Joan Gratch use *morphing*, to transform wellknown artworks into other equally wellknown artworks.

4. multimedia platforms

with DirectX 9 digital convergence becomes a reality

learning objectives

After reading this chapter you should be able to characterize the functionality of current multimedia platforms, to describe the capabilities of GPUs, to mention the components of the Microsoft DirectX 9 SDK, and to discuss how to integrate 3D and video.

Almost 15 years ago I bought my first multimedia PC, with Windows 3.1 Media Edition. This setup included a video capture card and a 4K baud modem. It was, if I remember well, a 100 Mhz machine, with 16 Mb memory and a 100 Mb disk. At that time, expensive as it was, the best I could afford. Some 4 years later, I acquired a Sun Sparc 1 multimedia workstation, with a video capture card and 3D hardware accelerator. It allowed for programming OpenGL in C++ with the GNU gcc compiler, and I could do live video texture mapping at a frame rate of about one per second. If you consider what is common nowadays, a 3Ghz machine with powerful GPU, 1 Gb of memory, a 1.5Mb cable or ADSL connection and over 100 Gb of disk space, you realize what progress has been made over the last 10 years.

In this chapter, we will look in more detail at the capability of current multimedia platforms, and we will explore the functionality of the Microsoft DirectX 0 platform. In the final section of this chapter, I will then report about the work I did with the DirectX 9 SDK to implement the ViP system, a presentation system that merges video and 3D.

1

4.1 developments in hardware and software

Following Moore's law (predicting the doubling of computing power every eighteen months), computer hardware has significantly improved. But perhaps more spectacular is the growth in computing power of dedicated multimedia hardware, in particular what is nowadays called the GPU (graphics processing unit). In Fernando and Kilgard (2003), he NVIDIA GeForce FX GPU is said to have 125 million of transistors, whereas the Intel 2.4GHz Pentium 4 contains only 55 million of transistors. Now, given the fact that the CPU (central processing unit) is a general purpose, or as some may like to call it, *universal* device, why is it necessary or desirable to have such specialized hardware, GPUs for graphics and, to be complete DSPs (digital signal processors) for audio?

a little bit of history

Almost evryone knows the stunning animation and effects in movies made possible by computer graphics, as for example the latest production of Pixar, *The Incredibles.* Such animation and effects are only possible by offline rendering, using factories of thousands of CPUs, crunching day and night to render all the frames.

At the basis of rendering lies traditional computer graphics technology. That is, the transformation of vertices (points in 3D space), rasterization (that is determining the pixel locations and pixel properties corresponding to the vertices), and finally the so-called raster operations (determining whether and how the pixels are written to the framebuffer). OpenGL, developed by SGI was the first commonly available software API (application programmers interface) to control the process of rendering. Later, Microsoft introduced Direct3D as an alternative for game programming on the PC platform.

The process outlined above is called the *graphics pipeline*. You put models, that is collections of vertices, in and you get (frames of) pixels out. This is indeed a simplification in that it does not explain how, for example, animation and lighting effects are obtained. To gain control over the computation done in the graphics pipeline, Pixar developed Renderman, which allows for specifying transformations on the models (vertices) as well as operations on the pixels (or fragments as they are called in Fernando and Kilgard (2003)) in a high level language. As vertex operations you may think of for example distortions of shape due to a force such as an explosion. As pixel operations, the coloring of pixels using textures (images) or special lighting and material properties. The languages for specifying such vertex or pixel operations are collectively called *shader* languages. Using offline rendering, almost anything is possible, as long as you specify it mathematically in a computationally feasible way.

The breakthrough in computer graphics hardware was to make such shading languages available for real-time computer graphics, in a way that allows, as Fernando and Kilgard (2003) phrase it, 3D game and application programmers and real-time 3D artists to use it in an effective way.

Leading to the programmable computer graphics hardware that we know today, Fernando and Kilgard (2003) distinguish between four generations of 3D accellerators.[18]

[18] The phrase GPU was introduced by NVIDIA to indicate that the capabilities of the GPU far exceed those of the VGA (video graphics array) originally introduced by IBM, which is

4 generations of GPU

- Before the introduction of the GPU, there only existed very expensive specialized hardware such as the machines from SGI.
- The first generation of GPU, including NVIDIA TNT2, ATI Rage and 3dfx Voodoo3, only supported rasterizing pre-transformed triangles and some limited texture operations.
- The second generation of GPUs, which were introduced around 1999, included the NVIDIA GeForce 2 and ATI Radeon 7500. They allowed for both 3D vertex transformations and some lighting, conformant with OpenGL and DirectX 7.
- The tird generation GPUs, including NVIDIA GeForce 3, Microsoft Xbox and ATI Radeon 8500, included both powerful vertex processing capabilities and some pixel-based configuration operations, exceeding those of OpenGL and DirectX 7.
- Finally, the fourth generation of GPUs, such as the NVIDIA GeForce FX and ATI Radeon 9700, allow for both complex vertex and pixel operations.

The capabilities of these latter generations GPUs motivated the development of high level shader languages, such as NVIDIA Cg and Microsoft HLSL. High level dedicated graphics hardware programming languages to control what may be called the programmable graphics pipeline.

the (programmable) graphics pipeline

Before discussing shading languages any further, let's look in some more detail at the graphics pipeline. But before that you must have an intuitive grasp of what is involved in rendering a scene.

Just imagine that you have created a model, say a teapot, in your favorite tool, for example Maya or 3D Studio Max. Such a model may be regarded as consisting of polygons, let's say triangles, and each vertex (point) of these triangles has apart from its position in (local) 3D space also a color. To render this model it must first be positioned in your scene, using so-called world coordinates. The *world transformation* is used to do this. The world transformation may change the position, rotation and scale of your object/model. Since your scene is looked at from one particular point of view we need to apply also a so-called *view transformation*, and to define how our view will be projected on a 2D plane, we must specify a *projection transformation.* Without going into the mathematical details, we may observe that these transformations can be expressed as 4x4 matrices and be combined in a single matrix, often referred to as the *world view projection matrix*, that can be applied to each of the vertices of our model. This combined transformation is the first stage in the process of rendering:

graphics pipeline

1. vertex transformation – apply world, view, projection transforms
2. assembly and rasterization – combine, clip and determine pixel locations
3. fragment texturing and coloring – determine pixel colors
4. raster operations – update pixel values

nothing more than a dumb framebuffer, requiring updates from the CPU.

The second phase, roughly, consists of cleaning up the collection of (transformed) vertices and determining the pixel locations that correspond to the model. Then, in the third phase, using interpolation or some more advanced method, coloring and lighting is applied, and finally a sequence of per-fragment or pixel operations is applied. Both OpenGL and Direct3D support among others an alpha (or transparency) test, a depth test and blending. The above characterized the fixed function graphics pipeline. Both the OpenGL and Direct3D API support the fixed function pipeline, offering many ways to set relevant parameters for, for example, applying lights, depth and texturing operations.

To understand what the programmable graphics pipeline can do for you, you would best look at some simple shader programs. In essence, the programmable pipeline allows you to perform arbitrary vertex operations and (almost) arbitrary pixel operations. For example, you can apply a time dependent morphing operation to your model. Or you can apply an amplification to the colors of your scene. But perhaps more interestingly, you can also apply an advanced lighting model to increase realism.

A simple morphing shader in ViP, see section 4.3.

2

a simple shader

When I began with programming shaders myself, I started with looking at examples from the DirectX SDK. Usually these examples were quite complex, and my attempt at modifying them often failed. Being raised in theoretical computer science, I changed strategy and developed my first shader program called *id*, which did nothing. Well, it just acted as the identity function. Then later I used this program as a starting point for writing more complex shader programs.

The *id* shader program is written in the DirectX 9 HLSL (high level shader language), and makes use of the DirectX Effects framework, which allows for specifying multiple vertex and pixel shaders, as well as multiple techniques and multiple passes in a single file.

The program starts with a declaration, specifying the global names for respectively the texture and the world/view/projection matrix. Also a texture sampler is declared, of which the function will become clear later.

HLSL declarations

```
texture tex;
float4x4 wvp;       // World * View * Projection matrix

sampler tex_sampler = sampler_state
{
   texture = /¡tex/¿;
};
```

It then defines, respectively, the vertex shader input and output, as structures. This declaration follows the standard C-like syntax for specifying elements in a structure, except for the identifiers in capitals, which indicate the semantics of the fields, corresponding to pre-defined registers in the GPU data flow.

vertex shader data flow

```
struct vsinput {
   float4 position : POSITION;
   float3 normal : NORMAL;
   float2 uv : TEXCOORD0;
};
struct vsoutput {
   float4 position   : POSITION;   // vertex position
   float4 color    : COLOR0;     // vertex diffuse color
   float2 uv  : TEXCOORD0;  // vertex texture coords
};
```

When the *vs_id* function, given below, is called, the input arguments are filled by the registers corresponding to the semantics pf the input structure. Similarly, the output results in setting the registers corresponding to the semantics of the output structure.

vertex shader

```
vsoutput vs_id( vsinput vx ) {
   vsoutput vs;

   vs.position = mul(vx.position, wvp);
   vs.color = color;
   vs.uv = vx.uv;

   return vs;
}
```

The *vs_id* function does exactly what the fixed graphics pipeline would do when transforming vertices. It applies the transformation to the vertex and passes both color and texture sampling coordinates to the pixel shader.

The pixel shader has a single color as output, which is obtained by sampling the texture, using the (interpolated) vertex color to modify the result.

pixel shader

```
struct psoutput
{
    float4 color : COLOR0;
};


psoutput ps_id( vsoutput vs )
{
    psoutput ps;

    ps.color = tex2D(tex_sampler, vs.uv) * vs.color;

    return ps;
}
```

Note that the *tex_sampler* comes from the global declaration above. The function *text2D* is a built-in for obtaining a color value from the sampler.

Finally, for each technique and each pass within a technique, in our case one technique with one pass, it must be indicated which function must be used for respectively the vertex shader and the pixel shader.

technique selection

```
technique render_id
{
    pass P0
    {
        VertexShader = compile vs_1_1 vs_id();
        PixelShader  = compile ps_2_0 ps_id();
    }
}
```

To make actual use of this program, the effect must be invoked from a DirectX or OpenGL program using the interface offered by the API.

Examples of Impasto, see examples – impasto

3

a morphing shader Slightly more complex is an example adapted from the morphing shader that can be found in ATI's Rendermonkey. To make a shader that morphs a cube into a ball and back, you must manipulate the vertices and the normals of the cube. For this to work your cube must have sufficient vertices, which is a property you can set in the tool that you use to make a cube.

morphing (vertex) shader

```
float3 spherePos = normalize(vx.position.xyz);
float3 cubePos = 0.9 * vx.position.xyz;

float t = frac(speed * time);
t = smoothstep(0, 0.5, t) - smoothstep(0.5, 1, t);

// find the interpolation factor
float lrp = lerpMin + (lerpMax - lerpMin) * t;

// linearly interpolate the position and normal
vx.position.xyz = lerp(spherePos, cubePos, lrp);
vx.normal = lerp(sphereNormal, cubeNormal, lrp);

// apply the transformations
vs.position = mul(wvp, vx.position);
```

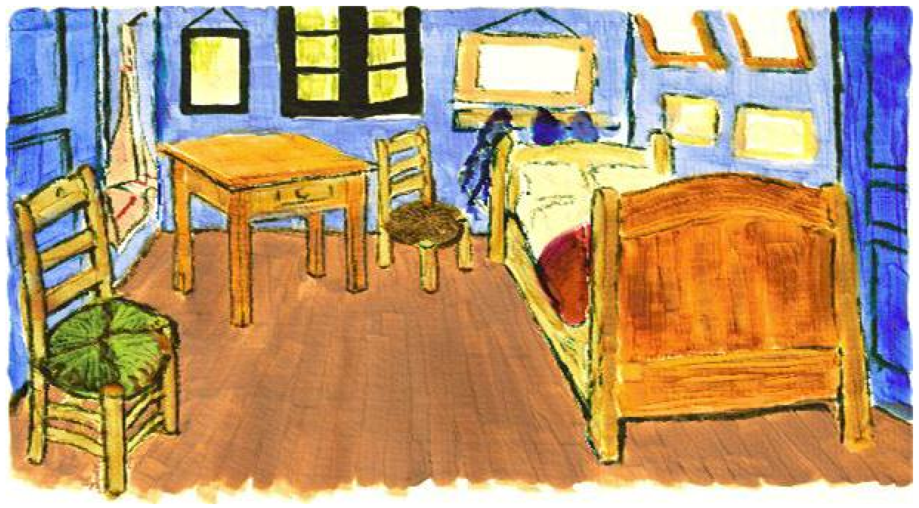
The example uses the built-in function *lerp*, that performs linear interpolation between two values using an interpolation factor between 0 and 1.

color amplification As an example of a pixel shader, look at the fragment defining an amplification of coloring below. It merely amplifies the RGB components of the colors when this exceeds a certain treshold.

coloring (pixel) shader

```
float4 x = tex2D(tex_sampler, vs.uv);
if (x.r ¿ x.g && x.r ¿ x.b) { x.r *= xi; x.g *= xd; x.b *= xd; }
else  if (x.g ¿ x.r && x.g ¿ x.b) { x.g *= xi; x.r *= xd; x.b *= xd; }
else  if (x.b ¿ x.r && x.b ¿ x.g) { x.b *= xi; x.r *= xd; x.g *= xd; }
ps.color = x;
```

When you develop shaders you must keep in mind that a pixel shader is generally invoked far more often than a vertex shader. For example a cube can be defined using 12 triangles of each tree vertices. However, the number of pixels generated by this might be up to a million. Therefore any complex computation in the pixel shader will be immediately noticable in the performance. For example, a slightly more complex pixel shader than the one above makes my NVIDIA GeForce FX 5200 accelerated 3 GHz machine drop to 5 frames per second!



rendering of van Gogh painting with Impasto

4

example(s) – *impasto*

IMPaSTo[19] is a realistic, interactive model for paint. It allows the user to create images in the style of famous painters as in the example above, which is after a painting of van Gogh. The *impasto* system implements a physical model of paint to simulate the effect of acrylic or oilpaint, using Cg shaders for real-time rendering, Baxter et al. (2004).

research directions – *the art of shader programming*

At first sight, shader programming seems to be an esoteric endeavor. However, as already indicated in this section, there are a number of high level languages for shader programming, including NVIDIA Cg and Microsoft HLSL. Cg is a platform independent language, suitable for both OpenGL and Direct3D. However, counter to what you might expect also Microsoft HLSL can be used for the OpenGL platform when you choose the proper runtime support.

To support the development of shaders there are, apart from a number of books, some powerful tools to write and test your shaders, in particular the already mentioned ATI Rendermonkey tool, the CgFx tool, which both produce HLSL code, as well as the Cg viewer and the effect tool that comes with the Microsoft DirectX 9 SDK.

Although I am only a beginning shader programmer myself, I find it truly amazing what shaders can do. For a good introducion I advice Fernando and Kilgard (2003). Futher you may consult Engel (2004a), Engel (2004b) and Engel (2005). Written from an artist's perspective is St-Laurent (2004).

[19]gamma.cs.unc.edu/IMPaSTo

Stacks and stacks of books on DirectX

5

4.2 DirectX 9 SDK

Many of the multimedia applications that you run on your PC, to play games, watch video, or browse through your photos, require some version of Direct X to be installed. The most widely distributed version of Direct X is 7.0. The latest version is the october release of 2004. This is version 9c. What is DirectX? And, more importantly, what can you do with it? In the DirectX documentation that comes with the SDK, we may read:

DirectX

> Microsoft DirectX is an advanced suite of multimedia application programming interfaces (APIs) built into Microsoft Windows; operating systems. DirectX provides a standard development platform for Windows-based PCs by enabling software developers to access specialized hardware features without having to write hardware-specific code. This technology was first introduced in 1995 and is a recognized standard for multimedia application development on the Windows platform.

Even if you don't use the DirectX SDK yourself, and to do that you must be a quite versatile programmer, then you will find that the tools or plugins that you use do depend on it. For example, the WildTangent[20] game engine plugin makes the DirectX 7 functionality available through both javascript and a Java interface. So understanding what DirectX has to offer may help you in understanding and exploiting the functionality of your favorite tool(s) and plugin(s).

DirectX 9.0 components

In contrast to OpenGL, the DirectX SDK is not only about (3D) graphics. In effect, it offers a wide range of software APIs and tools to assist the developer of multimedia applications. The components of the DirectX 9 SDK include:

DirectX 9 components

- Direct3D – for graphics, both 2D and 3D
- DirectInput – supporting a variety of input devices
- DirectPlay – for multiplayer networked games
- DirectSound – for high performance audio

[20]www.wildtangent.com

- DirectMusic – to manipulate (non-linear) musical tracks
- DirectShow – for capture and playback of multimedia (video) streams

In addition there is an API for setting up these components. Also, DirectX supports so-called *media objects*, which provide a standard interface to write audio and video encoders, decoders and effects.

Altogether, this is a truly impressive and complex collection of APIs. One way to become familiar with what the DirectX 9 SDK has to offer is to start up the sample browser that is part of the SDK and explore the demos. Another way is to read the online documentation that comes with the SDK, but perhaps a better way to learn is to make your choice from the large collection of introductory books, and start programming. At the end of this chapter, I will provide some hints about how to get on your way.

Direct3D

In the DirectX 9 SDK, Direct3D replaces the DirectDraw component of previous versions, providing a single API for all graphics programming. For Direct3D there is a set of simple, well-written tutorials in the online documentation, that you should start with to become familiar with the basics of graphics programming in DirectX.

Direct3D tutorials

- tutorial 1: creating a device
- tutorial 2: rendering vertices
- tutorial 3: using matrices
- tutorial 4: creating and using lights
- tutorial 5: using texture maps
- tutorial 6: using meshes

Before you start working with the tutorial examples though, you should acquire sufficient skill in C++ programming[21] and also some familiarity with Microsoft Visual Studio .NET.

One of the most intricate (that means difficult) aspects of programming Direct3D, and not only for novices, is the creation and manipulation of what is called the *device*. It is advisable to take over the default settings from an example, and only start experimenting with more advanced setting after you gained some experience.

DirectSound – the *drumpad* example

The DirectX SDK includes various utility libraries, for example the D3DX library for Direct3D, to simplify DirectX programming.

As an example of a class that you may create with DirectSound, using such a utility library, look at the *drumpad* below. The *drumpad* class can be integrated

[21] The DirectX 9 SDK also offers APIs for C# and VisualBasic .NET. See the *research directions* at the end of this section.

in your 3D program, using DirectInput, to create your own musical instrument. The header of the class, which is, with some syntactical modifications, taken from the SDK samples section, looks as follows:

```
class drumpad {
public:
    drumpad()
    ~drumpad();
    bool    initialize( DWORD dwNumElements, HWND hwnd );
    bool    load( DWORD dwID, const TCHAR* tcszFilename );
    bool    play( DWORD dwID );
protected:
    void    CleanUp();
    CSoundManager* m_lpSoundManager;
    CSound **       m_lpSamples;
};
```

The interface offers some methods for creating and destroying a *drumpad* object, initialization, loading sounds and for playing the sounds that you loaded. The *CSoundManager* is a class offered by the utility library for DirectSound.

The *play* function is surprisingly simple.

```
bool drumpad::play( DWORD id ) {
    m_lpSamples[id] -¿ Stop();
    m_lpSamples[id] -¿ Reset();
    m_lpSamples[id] -¿ Play( 0, 0 );
    return true;
}
```

The *id* parameter is a number that may be associated with for example a key on your keyboard or some other input device. Using the *drumpad* class allows you to make your own VJ program, as I did in the system I will describe in the next section. In case you are not familiar with either C++ or object-oriented programming, you should study object-oriented software development first. See for example Eliens (2000).

DirectShow

DirectShow is perhaps the most powerful component of the DirectX SDK. It is the component which made Mark Pesce remark that with the DirectX 9 SDK *digital convergence has become a reality.*[22] A technical reality, that is, Pesce (2003).

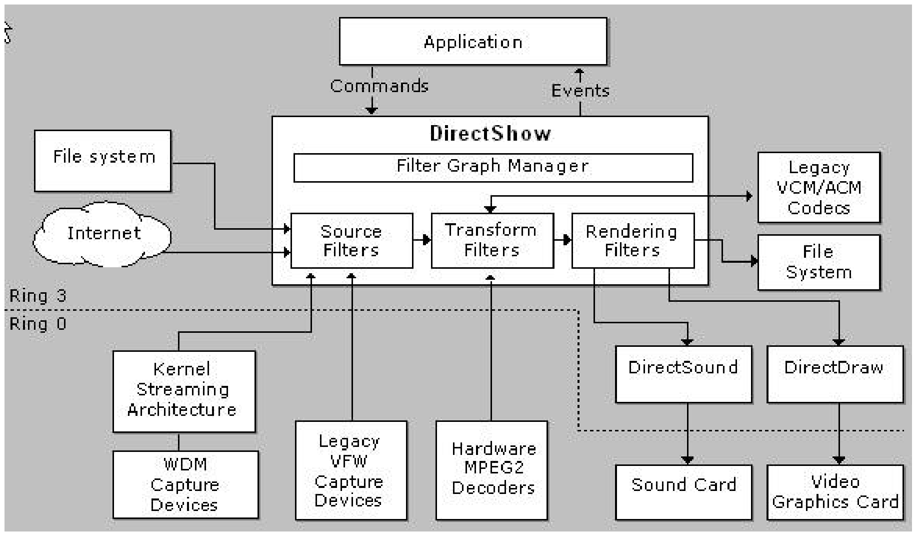
As we have seen in chapter 3, working with multimedia presents some major challenges:

multimedia challenges

[22] It is historically interesting to note that Mark Pesce may be regarded as the inventor, or initiator, of VRML, which was introduced in 1992 as the technology to realize a 3D web, as interlinked collection of 3D spaces.

- volume – multimedia streams contain large amounts of data, which must be processed very quickly.
- synchronization – audio and video must be synchronized so that it starts and stops at the same time, and plays at the same rate.
- delivery – data can come from many sources, including local files, computer networks, television broadcasts, and video cameras.
- formats – data comes in a variety of formats, such as Audio-Video Interleaved (AVI), Advanced Streaming Format (ASF), Motion Picture Experts Group (MPEG), and Digital Video (DV).
- devices – the programmer does not know in advance what hardware devices will be present on the end-user's system.

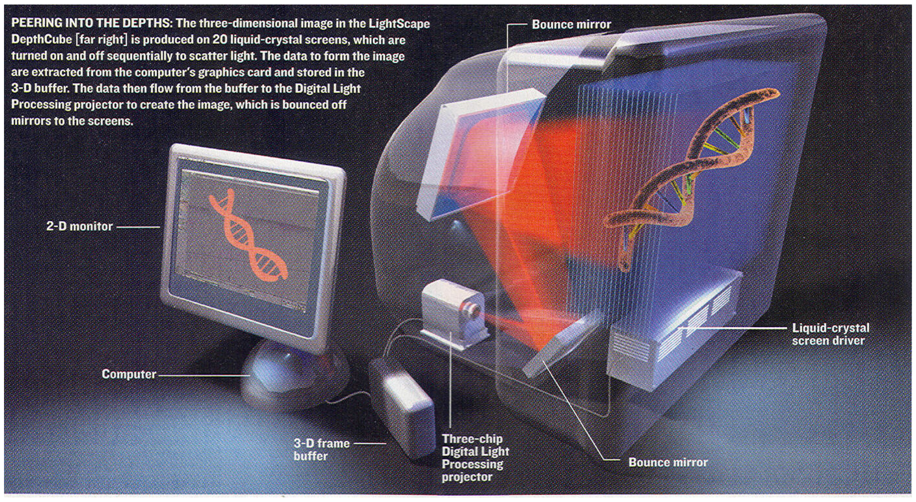
The DirectShow component was designed, as we learn from the online documentation, to address these challenges and to simplify the task of creating applications by isolating applications from the complexities of data transports, hardware differences and synchronization. The solution DirectShow provides is a modular architecture that allows the developer to set up a data flow graph consisting of *filters*. Such filters may be used for capturing data from, for example, a video camera or video file, for deploying a codec, through the audio compression manager (ACM) or video compression manager (VCM) interfaces, and for rendering, either to the file system or in the application using DirectSound and DirectDraw and Direct3D.



6

The diagram above, taken from the DirectX 9 SDK online documentation, shows the relations between an application, the DirectShow components, and some of the hardware and software components that DirectShow supports.

An interesting and convenient feature of the filter-based dataflow architecture of DirectShow is *SmartConnect*, which allows the developer to combine filters by indicating constraints on media properties such as format. The actual connections then, which involves linking input pins to output pins, is done automatically by searching for the right order of filters, and possibly the introduction of auxiliary filters to make things match.



PEERING INTO THE DEPTHS: The three-dimensional image in the LightScape DepthCube [far right] is produced on 20 liquid-crystal screens, which are turned on and off sequentially to scatter light. The data to form the image are extracted from the computer's graphics card and stored in the 3-D buffer. The data then flow from the buffer to the Digital Light Processing projector to create the image, which is bounced off mirrors to the screens.

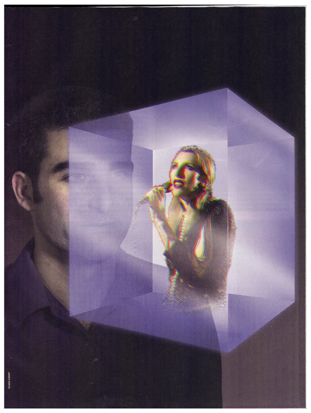
Bounce mirror
2-D monitor
Liquid-crystal screen driver
Computer
3-D frame buffer
Three-chip Digital Light Processing projector
Bounce mirror

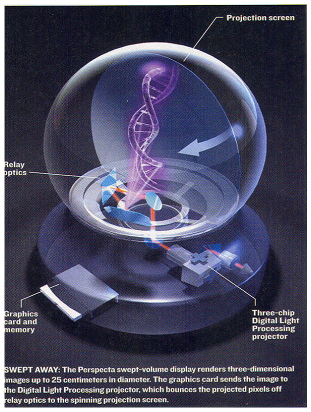
DepthCube, see example(s) – *3D vision*

7

DirectX application development

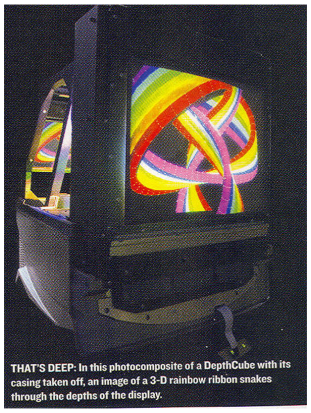
The examples that come with the DirectX'9 SDK use an application utility library, which includes a general application class that takes care of most of the details of creating an application and rendering window, initialization and event handling. For each of the SDK components there are numerous examples, ranging in difficulty from beginners to expert level. There are also a number of examples that illustrate how to mix the functionality of different SDK components, as for example the projection of video on 3D, which we will discuss in more detail in the next section.


3D vision



SWEPT AWAY: The Perspecta swept-volume display renders three-dimensional images up to 25 centimeters in diameter. The graphics card sends the image to the Digital Light Processing projector, which bounces the projected pixels off relay optics to the spinning projection screen.

Perspectra


THAT'S DEEP: In this photocomposite of a DepthCube with its casing taken off, an image of a 3-D rainbow ribbon snakes through the depths of the display.

DepthCube

8

example(s) – *3D vision*

Have you ever wondered how it would feel to be in Star Trek's holodeck, or experience your game in a truly spatial way, instead of on a flat LCD-display. In Sullivan (2005), an overview is given of technology that is being developed to enable volumetric display of 3D data, in particular the Perspecta swept-volume display (middle) and LightSpace DepthCube (right), that uses a projector behind a stack of 20 liquid-crystal screens.

The first approach of displaying volumetric data, taken by the Perspecta swept-volume display, is to project a sequence of images on a rotating sheet of reflective material to create the illusion of real volume. The psychological mechanism that enables us to see volumes in this way is the same as the mechanism that forces us to see motion in frame-based animation, at 24 frames per second, namely *persistence of vision*.

LightSpace DepthCube uses a stack of 20 transparent screens and alternates between these screens in a rapid way, thus creating the illusion of depth in a similar way. In comparison with other approaches of creating depth illusion, the solutions sketched above require no special eyewear and do not impose any strain on the spectator due to unnatural focussing as for example with autostereoscopic displays.

For rendering 3D images on either the Perspecta or DepthCube traditional rendering with for example OpenGL suffices, where the z-coordinate is taken as an indication for selecting a screen or depth position on the display. Rendering with depth, however, comes at a price. Where traditional rendering has to deal with, say 1024x748 pixels, the DepthCube for example needs to be able to display 1024x748x20, that is 15.3 million, *voxels* (the volumetric equivalent of a pixel) at a comparable framerate.

research directions– *the next generation multimedia platform*

Factors that may influence your choice of multimedia development platform include:

- platform-dependence – both hardware and OS
- programming language – C/C++, Java, .NET languages
- functionality – graphics, streaming media
- deployment – PC/PDA, local or networked, web deployment

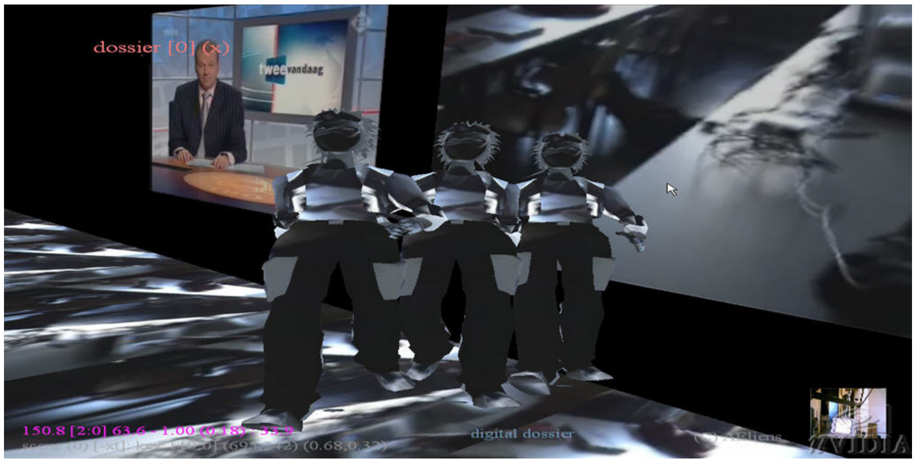
A first dividing line is whether you prefer to develop on/for Linux or Microsoft windows. Another dividing line, indeed, is your choice of programming language, C/C++, Java or .NET languages. Another factor that may influence your choice is the functionality you strive for. For example, Managed DirectX, for the .NET languages, provides only limited support for DirectShow and does not allow for capturing live video from a DV camera. And finally, it matters what deployment you wish to target for, mobile phone, PDAs or PCs, and whether you plan to make stand-alone applications or applications that must run in a web browser.

Apart from the hard-core programming environments such as the Microsoft DirectX 9 SDK, the Java Media Framework, OpenGL with OpenML extensions for streaming media, or the various open source (game development) toolkits, there are also high-level tools/environments, such as Macromedia Director MX, that allow you to create similar functionality with generally less effort, but also less control. In appendix E, a number of resources are listed that may assist you in determining your choice.

Given the range of possible options it is futile to speculate on what the future will offer. Nevertheless, whatever your choice is, it is good to keep in mind, quoting Bill Gates:

> *Software will be the single most important force in digital entertainment over the next decade.*

It should not come as a surprise that this statement is meant to promote a new initiative, XNA, which as the announcement says *is targeted to help contain the skyrocketing development costs and allow developers to concentrate on the unique content that differentiates their games.*



Animation in front of television news in ViP

9

4.3 merging video and 3D

In june 2003, I was approached by a theatre production company to advice on the use of "VR in theatre". As described in more detail in section 9.3, I explored what technology was available to realize such VR-augmented theatre. These explorations resulted in the development of the ViP system, that I once announced as follows:

www.virtualpoetry.tv

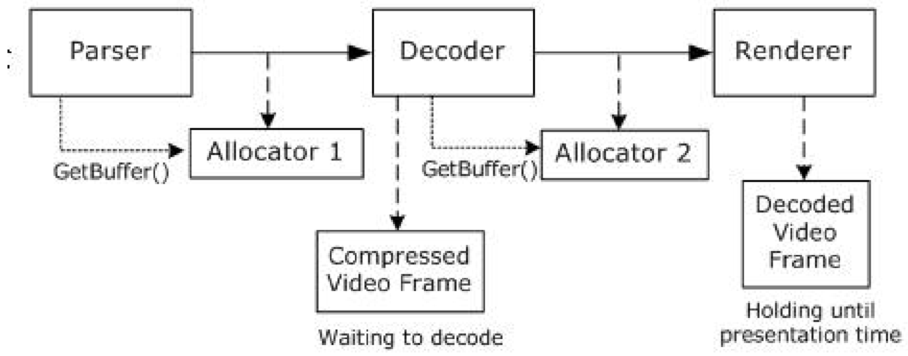
> *The ViP system enhances your party with innovative multimedia presentations.*
>
> *It supports multiple webcams and digital video cameras, mixed with video and images, enhanced by 3D animations and text, in an integrated fashion.*
>
> *For your party, we create a ViP presentation, with your content and special effects, to entertain your audience.*

In the course of time, I continued working on the system and, although I do not use it for parties, but rather for enlivening my lectures, it does include many of the features of a VJ system, such as the *drumpad* described in 3.2.

The major challenge, when I started its development, was to find an effective way to map live video from a low/medium resolution camera as textures onto 3D geometry. I started with looking at the ARToolkit but I was at the time not satisfied with its frame rate. Then, after some first explorations, I discovered that mapping video on 3D was a new (to some extent still experimental) built-in feature of the DirectX 9 SDK, in the form of the VMR9 (video mixing renderer) filter.

the Video Mixing Renderer filter

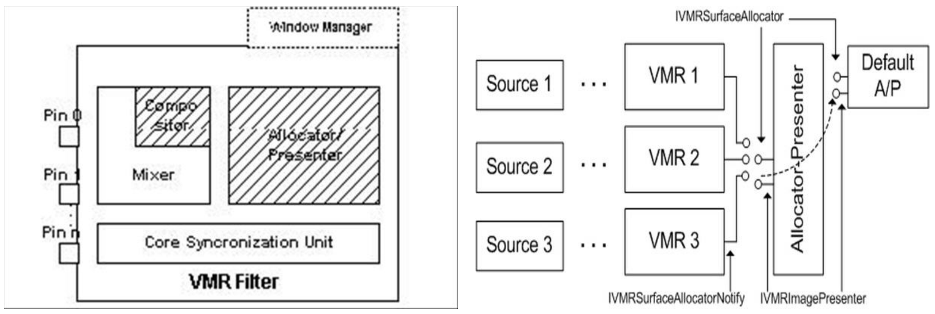
The VMR filter is a compound class that handles connections, mixing, compositing, as well as synchronization and presentation in an integrated fashion. But before discussing the VMR9 in more detail, let's look first at how a single media stream is processed by the filter graph, as depicted in the figure below.



10

Basically, the process consists of the phases of parsing, decoding and rendering. For each of these phases, dependent on respectively the source, format and display requirements, a different filter may be used. Synchronization can be either dtermined by the renderer, by pulling new frames in, or by the parser, as in the case of live capture, by pushing data on the stream, possibly causing the loss of data when decoding cannot keep up with the incoming stream.

The VMR was originally introduced to allow for mixing multiple video streams, and allowed for user-defined *compositor* and *allocator/presenter* components.



(a) VMR filter (b) multiple VMRs

11

Before the VMR9, images could be obtained from the video stream by intercepting this stream and copying frames to a texture surface. The VMR9, however, renders the frames directly on Direct3D surfaces, with (obviously) less overhead. Although the VMR9 supports multiple video pins, for combining multiple video streams, it does not allow for independent search or access to these streams. To do this you must deploy multiple video mixing renderers that are connected to a common allocator/presenter component, as depicted on the right of the figure above (b).

When using the VMR9 with Direct3D, the rendering of 3D scenes is driven by the rate at which the video frames are processed.

digital dossiers
knowledge
preservation
re-installation
contemporary art
nail sole shoes heel arthritus

Lecture on digital dossier for Abramovic, in ViP

12

the ViP system

In developing the ViP system, I proceeded from the requirement to project live video capture in 3D space. As noted previously, this means that incoming video drives the rendering of 3D scenes and that, hence, capture speed determines the rendering frame rate.

I started with adapting the simple *allocator/presenter* example from the DirectX 9 SDK, and developed a scene management system that could handle incoming textures from the video stream. The *scene* class interface, which allows for (one-time) initialization, time-dependent compositing, restore device setting and rendering textures, looks as follows:

```
class scene {
public:
        virtual int init(IDirect3DDevice9*);   // initialize scene (once)
            virtual int compose(float time);   // compose (in the case of an
     animation)
        virtual int restore(IDirect3DDevice9*);  // restore device settings
          virtual int render(IDirect3DDevice9* device, IDirect3DTexture9*
     texture);
protected:
```

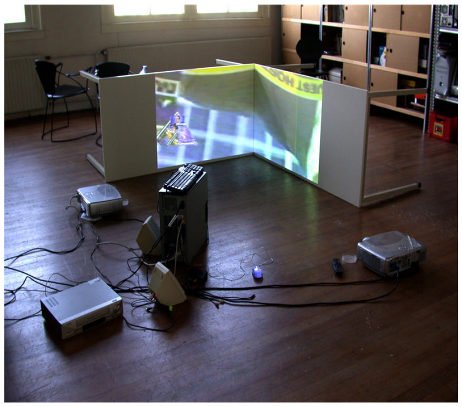
```
...
};
```

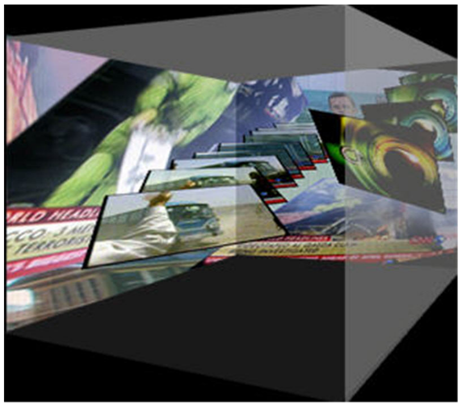
The scene graph itself was constructed as a tree, using both arrays of (sub) scenes as well as a dictionary for named scenes, which is traversed each time a video texture comes in. The requirements the scene management system had to satisfy are further indicated in section 9.3. Leaving further details aside, observe that for the simple case of one incoming video stream, transferring the texture by parameter suffices.

Later on, I adapted the *GamePlayer* which uses multiple video mixing renderes, and then the need arose to use a different way of indexing and accessing the textures from the video stream(s). So, since it is good practice in object-oriented software engineering to suppress parameters by using object instance variables, the interface for the *scene* class changed into:

```
class scene {
public:
        virtual int load();   // initialize scene (once)
        virtual int compose();  // compose (in the case of an animation)
        virtual int restore();  // restore device settings
        virtual int render();  // display the (sub) scene
protected:
...
};
```

Adopting the scene class as the unifying interface for all 3D objects and compound scenes proved to be a convenient way to control the complexity of the ViP application. However, for manipulating the textures and allocating shader effects to scenes, I needed a global data structure (dictionaries) to access these items by name, whenever needed. As a final remark, which is actually more concerned with the software engineering of such systems that its functionality per se, to be able to deal with the multiple variant libraries that existed in the various releases of DirectX 9, it was needed to develop the ViP library and its components as a collection of DLLs, to avoid the name and linking clashes that would otherwise occur.


installation


reality of TV news

13

example(s) – *reality of TV news*

The *Reality of TV news* project by Peter Frucht uses an interesting mix of technology:

- live video capture from the device of an external USB2.0 TV card
- live audio capture from the soundcard (line in)
- display of live audio and video with java3D (had to be invented)
- autonomous 3D objects with a specified lifetime
- collision behaviour (had to be invented)
- changing of texture-, material- and sound characteristics at runtime
- dual-screen display with each screen rotated toward the other by 45 degrees about the Y-axis
- 3D sound

In the project, as phrased by Peter Frucht, *the permanent flow of the alternating adverts and news reports are captured live and displayed in a 3D virtual-reality installation. The currently captured audio and video data is displayed on the surface of 3D shapes as short loops. The stream enters the 3D universe piece by piece (like water drops), in this way it is getting displaced in time and space - news reports and advertising will be displayed partly in the same time. By colliding to each other the 3D shapes exchange video material. This re-editing mixes the short loops together, for instance some pieces of advertising will appear while the newsreader speaks.*

The software was developed by Martin Bouma, Anthony Augustin and Peter Frucht himself, with jdk 1.5, java3d 1.31, Java Media Framework 2.1.1e. The primary technological background of the artist, Peter Frucht, was the book *CodeArt*[23], Trogemann & Viehoff (2004), by his former professor from the Media Art School in Cologne, Germany. The book is unfortunately only available in German, and should be translated in English!

[23]java.khm.de

research directions– *augmented reality*

In the theatre production that motivated the development of the ViP system, the idea was to have wearable LCD-projection glasses, with a head-mounted low resolution camera. This setup is common in *augmented reality* applications, where for example a historic site is enriched with graphics and text, laid on top of the (video rendered) view of the site. Since realtime image analysis is generally not feasible, either positioning and orientation information must be used, or simplified markers indicating the significant spots in the scene, to determine what information to use as an overlay and how it should be displayed.

The ARToolkit[24] is an advanced, freely available, toolkit, that uses fast marker recognition to determine the viewpoint of a spectator. The information that is returned on the recognition of a marker includes both position and orientation, which may be used by the application to draw the overlay graphics in accordance with the spectator's viewpoint.

Augnented reality is likely to become a hot thing. In april 2005 it was featured at BBC World[25], with a tour through Basel.

4.4 games for the people

14

questions

multimedia platforms

1. What components does a multimedia platform consist of? Discuss both hardware and software components.

concepts

2. Characterize the functionality of current multimedia platforms.
3. Explain the notions of vertex shader and pixel shader.
4. Indicate what solutions exist for merging video and 3D graphics.

technology

5. Characterize the capabilty of current GPUs.
6. What does HLSL stand for? Give some examples of what it is used for.
7. What are the components of the DirectX 9 SDK?
8. Explain how the VMR9 works. Give an example.

[24] artoolkit.sourceforge.net
[25] www.bbcworld.com/content/template_clickonline.asp?pageid=665&co_pageid=3

projects & further reading As a project, I suggest the development of shader programs using Rendermonkey[26] or the Cg Toolkit[27], or a simple game in DirectX.

You may further explore the possibilities of platform independent integration of 3D and media, by studying for example OpenML[28]. For further reading, among the many books about DirectX, I advice Luna (2003), Adams (2003) and Fay et al. (2004).

the artwork

1. dutch light – photographs from documentary film Dutch Light[29].
2. ViP – screenshot, with morphing shader, see section 4.3.
3. impasto – examples, see section 4.1
4. impasto – after a painting of van Gogh, using Cg shaders,
5. 3D vision, from Sullivan (2005), see example(s) section 4.2.
6. idem.
7. photographs of DirectX and multimedia books, by the author.
8. DirectX – diagram from online documentation.
9. ViP – screenshot, with the news and animations.
10. DirectX – diagram from online documentation.
11. DirectX – diagram from online documentation.
12. ViP – screenshot, featuring Abramovic.
13. Peter Frucht – *Reality of TV news*, see section 4.3.
14. signs – people, van Rooijen (2003), p. 248, 249.

The theme of the artwork of this chapter is *realism*. In the documentary *dutch light*, it was investigated whether the famous *dutch light* in 17th century painting really existed. The photographs shown here are a selection of shots that were taken on particular locations over a period of time. However, as an art historian formulated it in the documentary: *dutch light* is nothing but a *bag of tricks* shared by dutch 17th century painters. The examples from *impasto* demonstrated that, after all, realism is an arbitrary notion.

[26]www.ati.com/developer/RenderMOnkey
[27]www.nvidia.com/cg
[28]www.khronos.org/openml
[29]www.dutchlight.nl