



Dynamic Presentations for Illustration Purposes

Dissertation

zur Erlangung des akademischen Grades

Doktoringenieur (Dr.-Ing.)

angenommen durch die Fakultät für Informatik
der Otto-von-Guericke-Universität Magdeburg

von: Diplom-Informatiker Roland Jesse
geb. am 6. September 1974 in Salzwedel

Gutachterinnen / Gutachter:

Prof. Dr. Thomas Strothotte
Prof. Dr. Heidrun Schumann
Prof. Dr. Bernhard Preim

Ort und Datum des Promotionskolloquiums: Magdeburg, den 26. März 2004

Roland Jesse **Dynamic Presentations for Illustration Purposes**
Otto-von-Guericke University of Magdeburg, Germany.

© 2004 Roland Jesse
All rights reserved.

Abstract

Illustrations provide visual support for transmitting a communication goal with regard to an illustration subject and a target audience. A set of presentation variables exists supporting this task. This set has proven valuable for expressing illustration information. To broaden the spectrum of illustration techniques, this work introduces the notion of dynamic presentations for illustration purposes. Herein, temporally parameterised dynamics aim at maximising the expression capabilities while economising as much on cognitive load as possible. To this end, a collection of components is presented: (1) an overview of fundamentals of dynamics including cognitive aspects; (2) a temporal model for parameterising presentation of dynamics; (3) a set of exemplary dynamics with some special emphasis on non-photorealistic renderings; and (4) some aspects on system modelling for implementing the introduced concepts. Lastly, use of the presented components is exemplified by drawing on some applications using dynamics for illustration purposes. Inspirations for future research are derived for broadening the set of dynamic presentation techniques. This is expected to widen their field of application as well as further promoting the potential of dynamics as an independent expression dimension.

Kurzfassung

Illustrationen bieten visuelle Unterstützung für die Vermittlung eines Kommunikationszieles gegenüber einer Zielgruppe mit Bezug auf ein Illustrationsobjekt. Für diesen Zweck existiert bereits eine Gruppe von Präsentationsvariablen. Diese haben sich als wertvoll zur Repräsentation von Informationen in Illustrationen erwiesen. Um dieses Spektrum von Illustrationstechniken zu erweitern, werden in dieser Arbeit dynamische Darstellungen zu Illustrationszwecken eingeführt. Auf der einen Seite zielen hierbei zeitlich parameterisierte dynamische Darstellungen auf eine Maximierung der Ausdrucksmöglichkeiten ab, während auf der anderen Seite die kognitive Last möglichst gering gehalten wird. Zu diesem Zweck wird eine Reihe von Komponenten vorgestellt: (1) ein Überblick über Grundlagen dynamischer Darstellungen einschließlich kognitiver Aspekte; (2) ein Zeitmodell zur Parameterisierung von dynamischen Präsentationen; (3) eine Sammlung von dynamischen Darstellungstechniken, wobei ein besonderes Augenmerk auf nicht-fotorealistischen Methoden liegt sowie (4) einige Aspekte der Systemmodellierung zur Implementierung der vorgestellten Konzepte. Schlussendlich wird die Praktikabilität der geschilderten Methoden am Beispiel einiger Anwendungen aufgezeigt, die dynamische Darstellungen zu Illustrationszwecken verwenden. Daraus werden Inspirationen für weiterführende Forschungsaufgaben abgeleitet, die auf eine Ausweitung der Menge dynamischer Darstellungsmethoden abzielen. Es wird erwartet, dass damit eine Erweiterung des Anwendungsfeldes dynamischer Präsentationsmethoden ebenso erreicht werden kann wie das weitere Propagieren des Potenzials von Dynamik als eine eigenständige Ausdrucksdimension.

Acknowledgements

This thesis emerged from research conducted at the Department of Simulation of Graphics (ISG) of the University of Magdeburg, Germany. I am particular grateful to Prof. Dr. Thomas Strothotte, who has been my supervisor for the work of this thesis. He let me explore and be free in my research. His large experience in the domain, enthusiasm for research and incredible energy have provided a continuous stimulus for my research work.

I am indebted to Prof. Dr. Heidrun Schumann and Prof. Dr. Bernhard Preim for accepting to work as reviewers of this thesis. Their valuable comments and advice have significantly improved this thesis and its presentation.

I am grateful to the support and assistance I received from several colleagues. Parts of this thesis and collateral work were developed in collaboration with Axel Panning, Bernd Nettelbeck, Felix Ritter, Ingolf Geist, Iryna Davydova, Kai-Uwe Sattler, Knud Pehrs, Oliver Dunemann, Thomas Funke, and Tobias Isenberg. The results of these co-operations are published as: Dunemann et al. [2002a], Jesse and Isenberg [2003], Jesse et al. [2003], Dunemann et al. [2002b], Jesse et al. [2002], Dunemann et al. [2001], Jesse et al. [2000] and Jesse et al. [2001].

I owe my gratitude to the members of the secretariat of the ISG. Their general help, support, and organising skills are outstanding. The same respect is due to the technical staff at the ISG, whose thorough and bearing support proved helpful.

My thanks are due to Christian Döring, Ingolf Geist, and Dr. Jochen Schneider for proofreading parts of this thesis. A final word of thanks goes to Nadine Schulz for her critical analysis and editing of this thesis.

Contents

1	Introduction	1
1.1	Illustration—Visualisation—Presentation	2
1.2	Objectives of this Work	4
1.3	Results and Contributions	5
1.4	Structure of this Thesis	6
2	Fundamentals of Dynamics	9
2.1	Fundamentals of Dynamics Perception	9
2.1.1	Seeing Movement	9
2.1.2	Apparent Movement	10
2.1.3	Recognition of Motion Patterns	12
2.1.4	The Value and Risk of Motion	13
2.1.5	Supportive Object Perception in Illustrations	15
2.2	Dynamics Stimulus Window	15
2.3	Dynamics and the Representation of Temporal Change	18
2.3.1	Flow Along a Time Line	18
2.3.2	Flow Representation of an Impulse Response Function	21
2.3.3	Dynamic Metaphor	22
2.3.4	Dynamic Textures	23
2.3.5	Discussion	24
2.4	Hierarchy Model of Dynamic Presentation Variables	25
2.4.1	Partition of Dynamics into a Layer Model	26
2.4.2	Composite Dynamics Patterns	29
2.4.3	Local versus Global Dynamics	30
2.5	Summary	30
3	Illustrative Dynamics by Motion and Non-Realism	31
3.1	Illustration Target Function	32
3.1.1	Context	32
3.1.2	Framework Notation	33
3.1.3	Model Basis	34
3.1.4	Function Definition	36

3.1.5	Evaluation and Limitation	39
3.2	Classic and Static Illustration Techniques	40
3.2.1	Classification	40
3.2.2	Change of Colour	45
3.2.3	Transparency	45
3.2.4	Fisheye	47
3.2.5	Discussion	48
3.3	Motion Techniques	48
3.3.1	Oscillations	49
3.3.2	Structural Changes	53
3.3.3	Motion-enhanced Information Mural	60
3.4	Dynamic Changes of Rendering Styles	65
3.4.1	Hybrid Presentations	65
3.4.2	Overview of Style Blending	66
3.4.3	Alpha-Blending	68
3.4.4	Emissive Colour Blending	70
3.4.5	Comparing Style Transitions	71
3.5	Zoom-based Distortion Histories	72
3.5.1	Zoom Basis	74
3.5.2	Trace Line Support for Zoom Histories	75
3.5.3	The Chewing Gum Zoom History	79
3.5.4	Discussion	84
3.6	Classification Based on Effect on Scene Coherence	85
3.7	Summary	86
4	Temporal Control of Dynamics	89
4.1	Requirements	89
4.1.1	Event-based Requirements	91
4.1.2	Interval-based Requirements	92
4.1.3	Structural Requirements	94
4.2	Classification of Temporal Models	96
4.2.1	Time Lines	96
4.2.2	Graph-based Models	98
4.2.3	Petri Net Models	105
4.2.4	Object-oriented Models	106
4.2.5	Temporal Logic	112
4.2.6	Discussion	115
4.3	Model for Temporal Presentation	116
4.3.1	Behaviour in a Single Trajectory	116
4.3.2	Concurrent Trajectories	121
4.4	Specification of Control Function	124
4.4.1	Notation	124

4.4.2	Function	125
4.4.3	Graphical Representation	127
4.4.4	Fulfilling the Requirements	128
4.5	Script-Based Controlling of Dynamics	129
4.5.1	Script Basis	130
4.5.2	Script Composition	131
4.6	Summary	132
5	Toolkits for Temporally Constrained Dynamic Presentations	133
5.1	Implementation Basis	133
5.1.1	Scene Graph	133
5.1.2	Layout of Toolkits	135
5.2	Motion Toolkit	135
5.2.1	Oscillating Motion	136
5.2.2	Motion by Structural Change	138
5.3	OpenNPAR Integration of Dynamic Non-Realism	139
5.3.1	Fundamental Data Structure	140
5.3.2	Use of Hybrid Rendering Styles	141
5.3.3	Transition Between Styles	143
5.4	Temporal Server	145
5.4.1	System Design	146
5.4.2	Dynamics Management Unit	148
5.4.3	Web Service Interface	148
5.5	Workbench for Information Fusion	150
5.5.1	Context	150
5.5.2	Requirements	151
5.5.3	Components and Layout	152
5.5.4	Interaction and Temporal Fusion Characteristics	153
5.6	Summary	155
6	Applications	157
6.1	Enhancing Illustrations with Search Engines	157
6.1.1	Scenario	158
6.1.2	Context on Search Engine Handling	158
6.1.3	Design of the Illustration Process	160
6.1.4	Search Result Evaluation	162
6.1.5	Search Result Illustration	163
6.1.6	Mapping of Search Results onto Rendering Parameterisation	164
6.2	Information Fusion	169
6.2.1	Principle Fusion Result Presentation	169
6.2.2	Climate Data	170
6.2.3	Exemplary Scatterplot	173

6.2.4	Online Aggregation	175
6.3	Model Presentations	177
6.3.1	Presentation of Technical Objects	177
6.3.2	Examination of Anatomical Objects	179
6.3.3	Motion-enhanced Model Exploration	180
6.4	Summary	181
7	Concluding Remarks	183
7.1	Summary of Contributions	184
7.2	Further Research Directions	186
7.2.1	Extend Use of Distortion Histories	186
7.2.2	Extend General Field of Dynamics	187
7.2.3	Extend Field of Applications	187
7.2.4	Application-centred User Studies	188
	Bibliography	189
	Appendix	I
A	Description of Document Formats	III
A.1	Definition of the <i>Tigeop</i> Schema	III
A.2	Exemplary XML Document According to the <i>Tigeop</i> -Schema	V
A.3	WSDL Schema	VI
A.3.1	Description of the <i>Tigeop</i> Interface	VI
A.3.2	Description of the <i>Tigeop</i> Implementation	VIII
B	List of Supervised Works	XI
	Index	XIII

1 Introduction

In today's world, digital technologies and a steady increase of information density are of growing influence in daily and scientific life. Illustrations take part in the interplay of presentation systems. Such systems are used to dig into the bulk of available information sources with the goal of filtering out relevant information for a specific task. Thereby, illustration systems specifically target examination of geometric models. Supportive illustrations help a user to explore an available model and any possible annotation information provided along with it. Examples include the analysis of technical objects by an engineer, the examination of anatomical objects by medical students, or the investigation of geometric information spaces constructed as part of a broader information retrieval process by means of a visualisation pipeline.

Illustrations are subject to increased personalisation. That is, users utilise as much of the technical illustration basis as needed to meet their illustration goal. Along with the illustration models, these goals get more and more complex. This results in steadily more dense illustrative presentations. To prevent visual overload, flexible and powerful illustration techniques are required. This request is further deepened by the demand to reflect dynamics in illustration models. Possible causes for such dynamics span a broad range of areas. A technical illustration model may be provided with dynamic functionality such as flows of fluids or gases through a pipeline system. Another example of dynamics in an illustration model is in a flexible information retrieval process. Such process consists of a set of individual information transformation components possibly including a visualisation pipeline providing a geometric illustration model [Baeza-Yates and Ribeiro-Neto, 2002]. This model is of dynamic nature as it reflects a varying composition of employed information transformation components throughout the whole retrieval process. Such model dynamics can be reflected in multiple ways, such as textual annotations or a time line reflecting dynamic model behaviour.

Besides the set of existing presentation techniques, motion is regarded as an expression dimension of its own [Bartram, 2001]. It allows to provide changing presentations reflecting dynamics of the illustration model. Thereby, motion is characterised by a rich expressiveness combined with a low perceptive load as long as it is constrained accordingly. Based on this presentational power of motion, *dynamics* can be created that include motion without being limited to it. Thereby, any visual variation of a presentation over time is regarded as a dynamic presentation, also referred to as dynamics. Such dynamics can be used to reflect either dynamic

characteristics of an illustration model or any other model behaviour. Thus, the information that is to be conveyed is encoded by means of dynamics.

A special challenge in the construction of dynamic presentation techniques is in respecting their perceptual and cognitive limitations. These limitations provide a demand for constraining and parameterisation in the temporal dimension of dynamics.

The overall goal of an illustration is in communicating information about an illustration model to a targeted user. This is specifically achieved by directing user attention to relevant aspects of the illustration model. For this purpose, illustrative presentations are used. A variety of such techniques exists and is used in illustration systems. Due to increased complexity and personalisation of information models, these techniques suffice only partially in the goal to match behaviour of an illustration model visually. A task for dynamics-enhanced illustration systems is now to provide techniques and solutions for information management systems, that allow to extend the available expression set and reflect dynamic characteristics of illustration models and their context.

1.1 Illustration—Visualisation—Presentation

This work targets presentations for illustration purposes. Figure 1.1 points out the relationship and dependency between an illustration and the presentation used therein. Furthermore, the figure embeds visualisation and arranges all three terms forming a hierarchic layer model. This model provides the taxonomy of these fundamental terms in the context of this work.

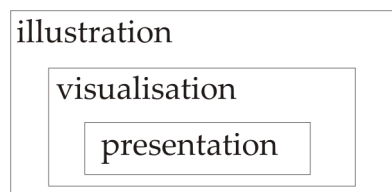


Figure 1.1: The relationship of illustration, visualisation, and presentation.

Illustrations are centred on communicating information to satisfy a user’s goal of information seeking. For this purpose, they might make use of visual metaphors as defined by visualisation techniques. For this reason, the hierarchic model is designed such that illustrations form as an extension of visualisation whereas both build on presentations as principle basis.

Illustration

The topmost layer of the hierarchy model is represented as *illustration*. The purpose of illustration is the visual presentation of an *illustration subject* or *model* to an *illustration target*. The illustration subject is referred to as the illustration model whereas the target is a user with the overall goal of fulfilling an illustration target function by this kind of presentation. For this purpose, a set of possible annotations may be used to lead the user throughout the illustration process. These annotations are typically of textual nature [Hartmann et al., 2002]. An example is presented by Ritter et al. [2003] where *illustrative shadows* are used to present the annotating text in the shadow of objects as integral part of the presented illustration model.

While illustrations are typically enriched by annotations, their accentuation is on a visual rather than textual metaphor as a means of conveying information [Beall et al., 1996]. Visual presentation techniques are used to direct user attention to specific aspects of the illustration model.

Visualisation

The potential for gaining information is no longer in the exclusive and automatic extraction of information from existing data sources. It is *visualisation* which supports the user in gaining information and deciding on influencing this process [Wong, 1999]. Such visualisation forms the second level of the representational layer model addressed here. Subject of visualisation is the computer-based visual presentation of data and information for the purpose of human interpretation depending on application and context of the visualisation data [Schumann and Müller, 2000]. This data-centred characteristic distinguishes visualisation from illustrations which are centred on the goal of communicating information.

Visualisation techniques and their geometric results might serve as basis for an illustration. For the context of this work, illustrations based on visualisation is referred to as *data-driven* illustration. Alternatively, illustrative presentations of geometric models are labelled as *geometry-based* illustrations.

Presentation

The lowest level of this layer model is referred to as *presentation*. Thereby, presentation is understood as any kind of visual output that is perceived by a targeted user. In contrast to visualisation as subject of the parent layer, a presentation describes only the concrete visual output as gained by using a set of available presentation variables. Any information about the source or ways of production of presented entities is not of relevance with regard to a presentation description.

A concrete implementation of presentation systems might span output by means of multiple media [Bordegoni et al., 1997]. Presentations for illustration purposes

as subject of this work concentrate on use of graphical media. Even though integration of multimedia components in illustration systems might be a valuable goal to achieve, it is not focused here but left as a task for further work.

1.2 Objectives of this Work

The main goal of this work is to extend the set of available presentation techniques. Even though a variety of such techniques already exists, these do not always suffice in meeting modern information seeking needs. A specific challenge in current illustration scenarios is reflection of dynamic behaviour in illustration models. Such model dynamics may be caused by a variety of reasons. For geometric illustration models, dynamic behaviour might be flow of functionality throughout the model. Examples for this case include physical flow such as gases running through a series of pipeline systems of the model or logic flow expressing interplay of individual model components. For an engine model, a specific logic flow may include the fly-wheel as well as the whole cooling aggregate due to internal model semantics. For a data-driven illustration model, dynamic behaviour may be reasoned in a streaming data source, in temporal data, or in variations of the data caused by operations constantly applied to it.

A further challenge for current illustration systems lies in the limited number of available presentation techniques. An increase in complexity of illustration models along with steadily refined illustration goals of a user result in the need to reflect increasingly more model characteristics. Thereby, classic presentation variables are often used to represent the illustration model only. For a geometric model, its individual parts are rendered by their respective shape, colour, and position. Data-driven illustrations use these presentation variables to reflect different attribute dimensions of the original data set. Specifically for complex geometric models and high-dimensional illustration data, the available space of expression variables is easily exhausted.

To address the above shortcomings of the set of presentation techniques currently available, this set is to be enriched by *dynamics*. Thereby, the notion of dynamics is understood as presentations that change over time. In order to exemplify the variety of expression capabilities of dynamics, a set of such techniques is to be developed.

For the purpose of evaluating the potential of dynamics as well as any possible drawbacks, cognitive fundamentals have to be derived from published studies. A specific focus should be placed on respective value and risk of motion, as motion forms an elemental basis of dynamic presentation techniques.

Depending on the results of deriving cognitive fundamentals of dynamics, a model for constraining dynamic presentations is to be constructed. The goal of such constraining is to allow use of dynamics up to their full potential while avoid-

ing cognitive and perceptual overload due to their over-excessive use. A constraint model for dynamics includes temporal modelling aspects. The respective set of temporal characteristics is to be worked out, that defines all necessary parts of controlling and parameterising dynamic presentation techniques for illustration purposes. Based on these characteristics, it is to be evaluated whether and how available temporal modelling approaches may be suitably employed for the constraining task at hand.

1.3 Results and Contributions

Contributions of this work are split into a set of areas. The core result of this thesis is in the introduction of a notion of *dynamics* as a set of presentation techniques resulting in varying presentations. A set of exemplary dynamics is developed. This set spans motion by oscillations as well as structural changes, dynamic use of hybrid rendering styles, and distortion histories at the example of fisheye zoom extensions. These dynamics form a basis for further investigations regarding the development of dynamic presentation techniques either from scratch or by defining dynamic extensions of existing presentation techniques.

In support of dynamics, underlying cognitive and perceptual fundamentals are analysed. Notions of a *dynamics stimulus window* as well as a *hierarchy model for dynamics* materialise these fundamentals and provide them in a compact fashion for general use. Based on the fundamentals, dynamics are *temporally constrained*. This allows to use dynamic presentations as effectively as possible. A set of requirements for a supportive temporal model is propagated. In addition to presenting a model fulfilling these requirements, underlying cognitive and perceptive *fundamentals* are analysed.

The individual parts of this thesis are combined by a discussion of exemplary *applications*. These span a variety of possible illustration subjects to present an overview of the range of possible applications of dynamics: simple presentational illustrations, information retrieval processes, and the reflection of dynamic sources of information for an illustration. These illustration scenarios are expected to serve as a starting point to motivate further use of dynamic presentation techniques for illustration purposes.

The contribution of this work is in the design and definition of dynamic presentation techniques that are temporally constrained on the basis of an evaluation of cognition-related studies. Such an integrated approach of using dynamic presentations for illustration purposes has not yet been addressed before. It is shown that new presentation techniques extend the available means of reflecting dynamic characteristics and behaviour of illustration models. Thereby, the set of expression dimensions is extended to address the increase of information density in today's world. The analysis of cognitive fundamentals of dynamics substantiates the

achieved temporal presentation constraints. These constraints ensure that the targeted user is not subject to cognitive overload. By using dynamic presentation techniques, illustration systems are provided with a means of appropriately presenting dynamic illustration models. This allows to address modern information needs.

1.4 Structure of this Thesis

This thesis is split into four major parts as outlined in Figure 1.2. The first part presents a foundation of dynamics. This is followed by the development of a set of dynamic presentation techniques. Temporal aspects of dynamic presentations are discussed in the third part whereas the last main part addresses implementation and application work.

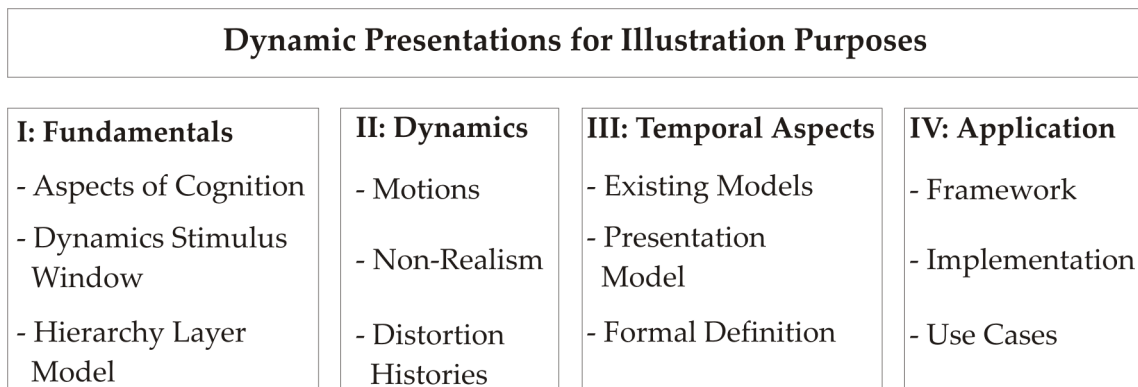


Figure 1.2: Structure of this thesis. Four main parts are outlined. The *fundamentals* are outlined in Chapter 2, *dynamics* are subject of Chapter 3, *temporal aspects* of controlling and parameterising dynamics are addressed in Chapter 4, and implementations as well as exemplary *application* scenarios are presented in Chapters 5 and 6, respectively.

In detail, this work is structured as follows: The next chapter concentrates on fundamentals of dynamics. First of all, aspects of perceiving dynamics are discussed. This is based on an analysis of published studies addressing various aspects of dynamics perception. Thereby, a specific focus is put on motion. This is due to motion acting as an elemental dynamic presentation technique. Findings of the discussed literature study are collected in two definitions: the notion of a dynamics stimulus window and construction of a hierarchy model of dynamic presentation variables. The dynamics stimulus window summarises perceptual information related to temporal stimulus characteristics. Based on this information, the window defines the temporal frame of dynamic presentations for the purpose of maximising the use of their expression potential. The hierarchy model of dynamic presentation techniques classifies different kinds of dynamics and partitions these classes into disjunct layers. These layers are arranged according to their respective perceptual influence. In addition to both presented definitions, that summarise fundamentals

of dynamics for the remaining parts of this thesis, the relationship between dynamics and the representation of time is analysed. This is done because an intrinsic characteristic of dynamics is the representation of changes over time. The presented overview of temporal change representation points out that dynamics not only play well in this regard but promise to support further illustration areas equally well.

The second main part of this thesis is represented by Chapter 3. Here, a set of exemplary dynamic presentation techniques is introduced. These are divided into three main groups: motion, non-realism, and distortion histories. For each group, some dynamic presentation techniques are constructed that express the respective group's main characteristics. In order to embed the dynamic presentation techniques in a flexible framework, the notion of an illustration target function is presented. A target function is used to express an illustration goal and its context. This helps to identify specific illustration components and to determine appropriate use of dynamics to fulfil the overall illustration goal. Additionally, the illustration target function is accompanied with the discussion of a presentation framework notation. This notation is used consistently throughout the remaining parts of the thesis.

Based on the fundamentals of Chapter 2, temporal control of dynamics is addressed in Chapter 4. For this purpose, a set of temporal requirements is presented. These requirements describe the functional frame that is to be fulfilled by a temporal model used for dynamics constraining. A variety of already existing temporal modelling approaches is analysed with respect to the requirements. These models are classified according to their modelling principles. A temporal model for parameterisation and constraining of dynamics is developed next. This model is based on elements and modelling approaches which are derived out of the different modelling classes. This new model fulfils all presented temporal requirements for dynamics constraints. A formal specification of a control function meeting this model is introduced. This function allows to provide a formal basis of an implementation using the shown temporal model. To deepen the support of such an implementation, a model for script-based control of dynamic presentations is introduced. Such script allows real-world illustration applications to employ dynamics with integrated temporal constraints.

Chapter 5 and 6 form the fourth and last part of this thesis. Implementation of the concepts introduced in Chapters 3 and 4 are addressed in Chapter 5. Four main components define the implementation basis: a motion toolkit, an integration of dynamics in *OpenNPAR*, a temporal server, and a workbench for information fusion. The first two components address dynamics as discussed in Chapter 3. The temporal server acts as application basis for the model of Chapter 4. Finally, the workbench provides an application testbed for complex illustration tasks that may be enriched by temporally constrained dynamics.

A set of concrete application examples is shown in Chapter 6. First of all, an illustration scenario is presented where dynamic presentations are used to reflect

dynamics of the illustration model. Such model dynamics are gained by retrieving illustration information through queries sent to a search engine. Depending on results retrieved from such search engine, the illustration presentation is modified. Secondly, a set of application areas in the context of information fusion is presented. These examples are framed by a presentation of some characteristics of information fusion. Finally, use of dynamics for presenting geometric models is shown. A set of two distinct types of models is used: technical models and anatomical models. In addition to the remaining illustration scenarios presented earlier, this reveals the flexibility of possible use of dynamic presentations for illustration purposes.

Chapter 7 concludes this thesis. On one hand, this provides a summary of the achieved work and contributions. On the other hand, prominent research directions are put forward. These directions are derived from the presentation framework introduced within this thesis and are expected to extend it in a most valuable way. Thereby, these research directions cover refinement of concepts and algorithms presented within this thesis as well as the development of new dynamic presentation techniques and two concrete real-world application scenarios.

2 Fundamentals of Dynamics

The perception of dynamics evolved as a fundamental key element of the human cognition system [Gregory, 1998, p. 98]. Stimuli caused by motion are processed by the brain to allow for fast handling of dynamics perception. This builds the basis for fast response and reaction times. Historically, fast reactions were of vital importance. However, even though this has changed to some degree in modern civilised environments, fast reaction to changing events still proves to be a key element of the human cognitive system. This way, any design of illustration systems can take advantage of perception capabilities that stem from evolutionary survival needs.

This chapter contributes an analysis of the fundamentals of dynamics perception. Furthermore, the notion of a *dynamics stimulus window* is introduced. This defines the temporal space and limitations for the parameterisation of dynamics. A discussion of a set of existing systems addressing the representation of temporal data analyses the value of dynamics for this purpose. Finally, a *hierarchy model of dynamic presentation variables* helps to order the different techniques according to their perceptual influence.

2.1 Fundamentals of Dynamics Perception

The discussion of an analysis of the value of dynamics is based on available studies about perception of motion. First of all, some basis of motion recognition is defined. This specifically includes various kinds of movement perception. After a confrontation of the value and risk of motion, specific advantages of motion with regard to object recognition are discussed.

2.1.1 Seeing Movement

Specific anatomic characteristics of the eye hold responsible for recognition of motion [Gregory, 1998, ch. 6]. Two alternative movement processing systems with regard to human perception are presented by Gregory: image-retina movement and eye-head movement.

The image-retina system provides recognition of movement caused by moving objects that run along the retina while the eyes are held still. Sequential firing of the retina's receptors causes information about movement being deduced by the brain.

This perception property proves powerful compared to the remaining characteristics of the retina. Especially the edges of the retina are sensitive only to movement, not to the perception of static objects. That is, a moving object is perceived as such but the shape and colour of the object are not recognised. Stopping an object's motion therefore stops its respective perception.

Eye-head movement describes an alternative way of perceiving motion. The moving objects do not change their position regarding to the retina. The image of a respective moving object remains stationary upon the retina. In order to still see the movement, the eyes follow the objects along their motion trajectories. The required commands for eye movement as well as any possibly needed head movement are processed by the brain along with the image information provided by the retina. This way, the brain itself provides the necessary scenic background and maps stationary retinal information onto respective real-world motion.

2.1.2 Apparent Movement

Apparent movement plays a vital role in the field of dynamics perception. It is denoted as a form of motion that is implicitly deduced [Goldstein, 2002, ch. 8]. This perception is explained by phenomenological methods. Motion is perceived without moving stimuli. Instead it is induced by the perception of another object.

Different kinds of apparent movement exist. An overview is presented in Figure 2.1 at the example of a pinpoint of light. Any real movement is presented as a straight arrow. Dashed arrows illustrate perceived movement. Normal movement of light is perceived as it is. Thus, it is considered as *real* movement which is included in the figure for reference. The remaining cases presented in the figure point out specific cases of apparent movement.

Stroboscopic motion is perceived in case two, where a pair of lights flash shortly after each other. The perceived movement implies that the light moves from one location to the other. A set of parameters influences this motion: distance of the pinpoints of light, their brightness, and the temporal distance of the flashes. Increased brightness results in increased motion perception. The smaller the spatial and temporal distance of both positions, the easier is the motion perceived. Some specific details about the temporal frame for stroboscopic motion will be discussed shortly in Section 2.2 addressing the dynamics stimulus window.

The effect of stroboscopic motion is widely used in a variety of application areas [Carmesin and Arndt, 1995, Krekelberg and Lappe, 1999, Kruse et al., 1996, Lappe and Krekelberg, 1998, Zagier, 1997]. Movies and animations in general depend on this effect. Both are composed of a sequence of images where consecutive images do not need to vary too much from each other. This results in the perception of a smooth and continuous motion. In case consecutive images differ or in case of a too restrict temporal frame, frame coherency is lost. This results in perception of a

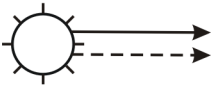

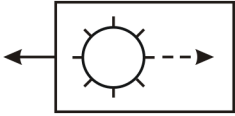
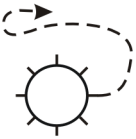
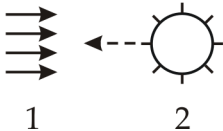
Apparent Movement	Arrangement	Setup
1) real		light source moves
2) stroboscopic		lights flash alternatively
3) induced		static light source in a moving environment
4) autokinetic		stationary pinpoint of light in the dark
5) aftereffect		lasting display of static image after motion

Figure 2.1: Five different setups that result in the perception of apparent motion. (based on [Guski, 1996, p. 172])

lack in smoothness.

Another kind of apparent movement is denoted in the figure as *induced* motion. This case specifically underlines that motion is always relative with regard to a reference point (cf. chapter five in Masuch [2001]). In the figure's arrangement, the pinpoint of light is perceived to be moving to the right even though its position is not changed. Instead, the environment of the light is subject of a translation to the left. As the viewer focuses on the light, the environment functions as a positional reference and the light appears to move.

An *autokinetic* motion describes the perception of automatically self-moving objects. It is derived from the illusion of movement that occurs when a stationary pinpoint of light is displayed in a totally dark room. That is, a very small or sharp point is perceived to be moving in case it is presented just by itself in a completely dark environment. After some amount of time of watching this setup, the perceptive system is tricked to see the light moving randomly [Sherif, 1935]¹. This perception cannot be controlled reliably which makes any practical use of this effect a challenging task.

¹ The study published by Sherif [1935] is the first known publication on the effect of autokinetic movement. Others picked up on this observation and support it [Comalli, Jr. et al., 1957, Nijhawan, 1997].

A common case of apparent movement is the *aftereffect*. After an ongoing observation of a steadily moving setup that abruptly stops, the stopped objects seem to move in the opposite direction as before. This effect holds valid for the duration of a couple of seconds. In the arrangement shown in the figure, the light was moving steadily from left to right before it stopped. Afterwards—as indicated by the dashed line—the light is perceived to move to the left.

Other well-known examples of the aftereffect include the waterfall and archimedic spiral. After a lasting gaze at a waterfall with its permanent motion and an abrupt stopping of this motion (or, more realistically, after changing focus to a neutral and static area) the impression of an oppositely directed motion is engendered. The archimedic spiral shown in Figure 2.2 rotates continuously around its centre. As soon as the rotation stops, an apparently opposite rotation is seen.



Figure 2.2: Archimedic Spiral.

The different effects of apparent movement promise to be helpful for the communication of information in an illustration system. Generated images and especially dynamic presentations are enriched by providing expression of motion without a need to invest valuable resources from the set of available presentation variables. This specifically holds as there is ample evidence that the same area of the brain are activated by both types of movement: apparent movement created by flashing lights and real movement created by actual movement through space [Stevens et al., 2000].

2.1.3 Recognition of Motion Patterns

Early studies express that the visual gathering of groups of semantically connected data is eased in case these groups are expressed by similar motion patterns instead

of static displays [Johansson, 1964]. Ware [2000] presents another survey by Johansson [1973] revealing that even abstract motion patterns are conceived as representations of concrete facts [Ware, 2000, p. 237f]. Visual metaphors defined by motion patterns can even be resolved in case the motion is only visible for parts of a second. Likewise, the causal connection of data is easily perceivable by users if represented by either the same motion patterns or patterns that are temporally connected [Ware, 2000, p. 235f]. Bartram et al. [2001] show that object movements are recognised faster and less error-prone than changes in colour or shape of an object.

Furthermore, Gregory [1998] presents demonstrations of Johansson [1975] showing how little information is needed in order to see moving humans and animals. In the example presented, lights are attached to the joints of arms and legs in an otherwise completely dark environment. In case they move, the people in the scene are recognised as such. Even a distinction between male and female can be made because of slight differences of their respective movements.

2.1.4 The Value and Risk of Motion

Motion is deemed as an expression dimension of its own. It allows easy perception of any changes in the presentation without putting a high cognitive load on the user. The early study by Johansson [1964] points out that changes in two dimensions are easily perceived as a motion in 3D. That is, different frequencies in the x - and y -directions of a stimulus pattern provide perceptions involving motion in the z -direction. Even though this cognitive conclusion does follow some restrictions (such as velocity), it shows that motion provides a rich expression capability. In case animation is effectively used, this cognitive task can be transformed into a perceptual one [Robertson et al., 1991]. The expression potential of motion is picked up by Bartram [1998, 2001] for the purpose of enriching information visualisations by motion. To support this attempt, a set of user studies has been carried out [Bartram, 1997a,b, Ware et al., 1999]. These point out a valuable potential for increasing user interface bandwidth by motion.

Motion provides an extensive interpretation scope. Complex psychological impressions can be produced by simple actions that are relatively inexpensive to compute [Lethbridge and Ware, 1990]. This allows to make information accessible which is not directly expressed in available data but is semantically encoded therein. A set of studies by Arthur et al. [1993] and Ware et al. [1993] reveals that motion provides more valuable perceptual clues than stereopsis. According to Ware and Franck [1996], motion is also a more effective method than stereo in regard to the disambiguation of three-dimensional graphs.

However, as promising as the use of motion as a display dimension of its own sounds, an eventual cognitive overload caused by motion should be avoided. For this reason, any real implementation either needs to follow a set of restriction guide-

lines or needs to be questioned at all. By analysing the results of a set of user studies on the perception of motion, Pylyshyn et al. [1993] present several basic properties of visual spatial attention. One is »that it is possible to track about 4 randomly moving objects and to keep them distinct from visually identical distractors, so that events taking place on the tracked targets can be quickly detected and identified« [Pylyshyn et al., 1993, p. 21]. This puts a considerable restriction on the set of motion techniques that can be employed simultaneously in a scene, even though the fixed limit of four objects is loosened by the authors. Thus, several (up to five) items can be precued from among a larger set of items. The cued items will be treated by the human visual system as though they were the only ones in the scene [Ibid.]. Thereby, the cognitive task depends on various kinds of stimuli. Chey et al. [1997] discuss neural dynamics of motion processing at the example of speed discrimination. Four dimensions define the respective motion parameter space: stimulus contrast, dot density, duration, and spatial frequency. While the first two of them are specifically related to speed discrimination, the latter two directly affect motion perception and influence the motion parameter space.

A more scenario based reason for limiting the use of motion by means of animation is presented by Harrison [1995]. In a scenario of an online help system that includes still graphics as well as animated visuals, the animations did not succeed in providing either considerably more information or a more sustainable learning experience. This result is questioned to some degree as the animations provided in the study were segmented to emphasise each stop of a procedural task. That is, the task at hand was not specifically designed to benefit from animated assets in the help system. This study is picked up by Morrison et al. [2000]. They extend the argument of an animation's limited use for conveying information by analysing the available literature on user studies targeting the perception of animation. In most cases, animations fail the expectations because they are difficult to perceive or because they mismatch the user's conception of motion, which seem to be more often discrete rather than continuous.

Hudson and Parkes [2003] present a study on possible visual overload by using animated graphical layering. Their setup includes animated transparent layers. These permit the user to selectively view elements that overlap. Compared to manual rearrangement of overlapping layers, animation helps to lower the cognitive load for interactive selection tasks. However, the study points out that visual cluttering is not completely avoided by substituting static layers with animated ones. The proposed solution combines transparent animations with alternating contrasting bands that progress over the display. This way, the interaction object to be emphasised is made prominent to peripheral vision.

2.1.5 Supportive Object Perception in Illustrations

A central requirement for the illustration of geometric objects is the guarantee to present all parts of an object such that they can be visually distinguished. Figure-ground perception provides for the distinction of objects from their scenic background as well as other objects. The field of perception is thereby divided into a set of distinct areas. This process happens pre-attentively. That is, figure-ground perception is neither controlled directly by the perceiving person nor influenced by a knowingly controlled attention. A set of characteristics of the visual information presented determines the success of this perceptive process. An appropriate parameterisation of an illustration system helps to make use of these characteristics in order to influence figure-ground perception.

In case of a dynamic scenery, figure-ground perception is based on the decoupling of surface recognition of a dynamic object compared to surfaces of static scene elements. Early studies show that the distinction of an object from its background is based on detection of an edge marking the junction from one to the other [Gibson et al., 1969]². Shifting this edge indicates a motion of the respective object.

Maglio and Campbell [2000] discuss a set of experiments addressing expressive power of motions by periphery presentations. For this purpose, they employ a two-task-study: While subjects work on a text, additional text is displayed in the periphery margin. In case this additional text scrolls continuously at constant speed, the subjects are more distracted than in case the text is scrolled slowly and on demand. Furthermore, answers to a questionnaire point out, that in the latter case, more information is extracted from the text by the subjects. The use of discrete targeting dynamics proves more useful than its continuous and ongoing counterpart.

2.2 Dynamics Stimulus Window

In order to assure the perception of different stimuli as such, they are to be presented to the user with a minimum distance defined by the *temporal-order threshold* of 20 – 40 ms. Thus, events are *not* perceived to happen separately and consecutively unless they are separated by at least this interval. This minimum event separation time span is also referred to as *inter-stimulus interval* [Hirsh and Sherrick, 1961, Pöppel, 1997, von Steinbüchel et al., 1996]. The temporal-order threshold can be subject to training. However, this training mainly addresses brain-injured patients. In case of a healthy audience, the temporal window of 20 – 40 ms holds [Mates et al., 2001].

So called *stimulus parameters* determine the synchronisation window. These describe inputs to the perceptive system that cause an action. A reaction at the lower

² Referenced by Guski [1996].

boundary of the stimulus window (20 ms) is caused by modifying two stimuli simultaneously with regard to their quality and spatial location. A variation occurring at the same position, exclusively, results in a reaction at the upper limit of 40 ms [Swisher and Hirsh, 1972]³.

Further support for the specification of an perception-based interval separating event is provided by an evaluation by Kanabus et al. [2002]: The minimal temporal interval enclosing the perception of a correct distinction of two acoustic or visual events exceeds 40 ms. This number is based on a 75% quota⁴ of correct answers of all test persons. This directly leads to the conclusion that a frame rate of not necessarily more than 25 fps suffices to produce an impression of smooth dynamics.

Studies of various stimulus ranges (acoustic and visual) furthermore show, that the human time-organisation system is independent from periphery sensory mechanisms [Efron, 1963, Tallal et al., 1998, von Steinbüchel et al., 1999b,a]⁵. Thus, it directly relates to the lower boundaries of the event identification.

The maximum possible time frame allowing the perception of an event as being unique and separated from others varies between two and three seconds [Pöppel, 1994]. Pöppel [1997] references a set of studies supporting this [Elbert, 1991, Pöppel, 1971, 1978]. Furthermore, any comparison of two stimuli has to take place in less than four seconds in order to ensure that the first stimulus does not fade. As a side effect, unrelated events are recognised as belonging together in case they occur in a time frame of up to two to three seconds. In case a visual event exceeds this limit, it is likely not to be perceived as an individual unit by the user.

An example of these dynamic stimulus capabilities is the Necker cube by the Swiss crystallographer Louis Albert Necker. He recognised in 1832 that the cubic shapes spontaneously reverse in perspective. This only holds for a cube drawn in an orthographic projection as shown in Figure 2.3. The human perspective system receives the two-dimensional shadow of a cube as a three-dimensional object. However, the 2D image does not distinguish the front and back faces. Either one may be perceived as being in front. This cube and its attached perception information are presented by Pöppel [1997]. Even though this cube is a static image, the perception of its perspective characteristic is dynamic. In case, subjects are able to perceive the perspectives of the cube, there is an automatic shift between both perspectives. This shift occurs in regular intervals of approximately three seconds duration [von Steinbüchel et al., 1996].

Some interesting support for the derivation of the temporal presentation frame is provided by studies on *apparent movement* (see also Subsection 2.1.2 above). An example is presented in Figure 2.4. It has been observed that the shown line (a) is perceived to be moving from left to right in case it is presented at its two distinct

3 Referenced by Kanabus et al. [2002].

4 A quota of 75% is deemed as the boundary between the denomination of *correct* and *approximate* results in such user studies [Tallal et al., 1998].

5 Referenced by Kanabus et al. [2002].

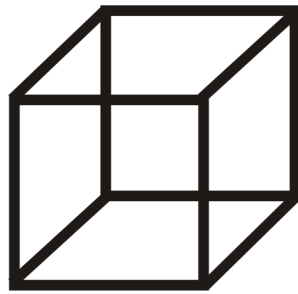


Figure 2.3: The Necker cube. As long as the perspectives are recognised at all, the subject's perception switches between both perspectives at regular intervals of circa 3 s. (following Pöppel [1997] and based on data presented by von Steinbüchel et al. [1996])

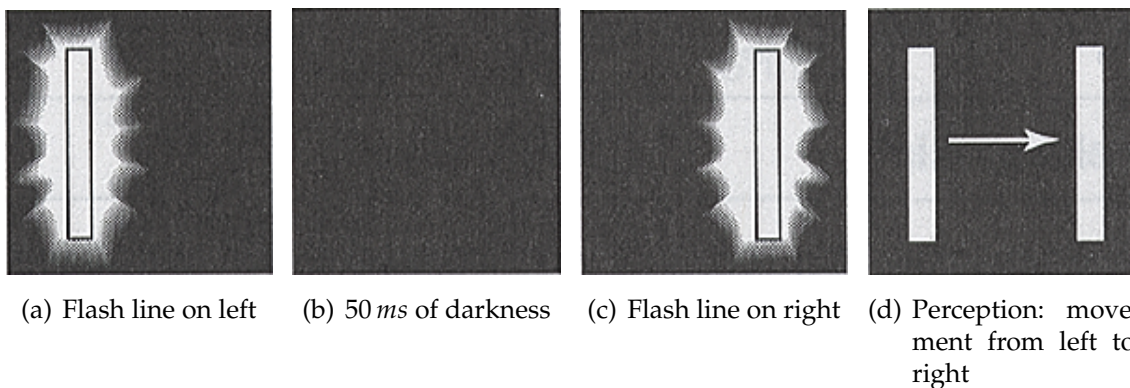


Figure 2.4: (a) Flashing a light in one position and then, (b) after a brief pause, (c) flashing it in another position, creates (d) an illusion called *apparent movement* [Goldstein, 2002].

positions in the time frame of 50 ms, as denoted in image (b) of the figure. The motion is induced by any arbitrary real-world object changing its position from A (left) to B (right). How much apparent movement follows the outlined motion stimulus time frame is documented by Goldstein [2002] as outlined in Figure 2.5. The lower limit of movement perception is herein extended from 40 ms to a time frame of 30 – 60 ms. Other temporal constraints hold as described above.

Whereas some visual perception parameters (such as colour) depend on cultural influences, the temporal limits for event perception do not [Gerstner and Fazio, 1995, Schleidt et al., 1987]. Furthermore, the aforementioned time spans are pre-semantic, that is, independent of the concrete task at hand [Pöppel, 1997, p. 59]. Therefore, the transition between two distinct dynamic presentation techniques is best presented in a time frame ranging from 30 ms to three seconds. This time span is referred to as the *dynamics stimulus window*.

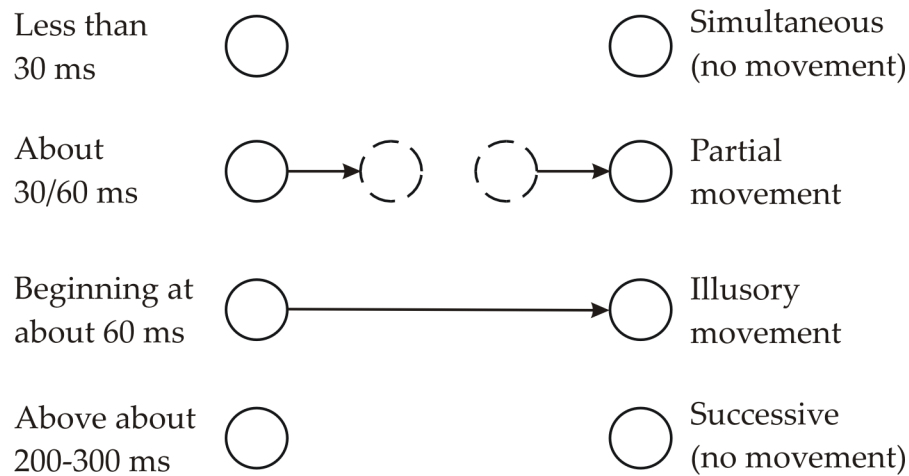


Figure 2.5: The perception of apparent movement depends on the time interval between the flashing of two lights. As the time interval is increased, the observer's perception goes through the stages shown in the figure [Goldstein, 2002].

2.3 Dynamics and the Representation of Temporal Change

An intrinsic characteristic of dynamics is the representation of changes over time. This holds especially for animations. Presentation of varying images directly correlates to temporal changes of the state of the animated subject. In order to reflect this bounding of dynamics and time representation, a selected set of approaches is presented here. This outlines various attempts to use dynamics for the representation of temporal behaviour of their subject. This composed collection presents an overview of dynamics-based time representation is presented. Each system is chosen according to its appropriateness and how well suited the system is with regard to representing its underlying presentation role.

For each presented system, its presentation domain is outlined. The domain describes the concrete application example for the respective system. While most systems are designed to be independent from any concrete application domain, these domains still provide some background on system design considerations. One effected aspect is the dimensionality of presented information. This is mainly provided by the available data sets and therefore the application domain. Any relevant aspects of the systems' internal temporal modelling are discussed in Section 4.2.

2.3.1 Flow Along a Time Line

The most basic kind of a diagram representing temporal behaviour are *time lines*. These use a two-dimensional, Cartesian coordinate system. One axis represents time. This is typically the horizontal axis. The other axis is divided into a set of

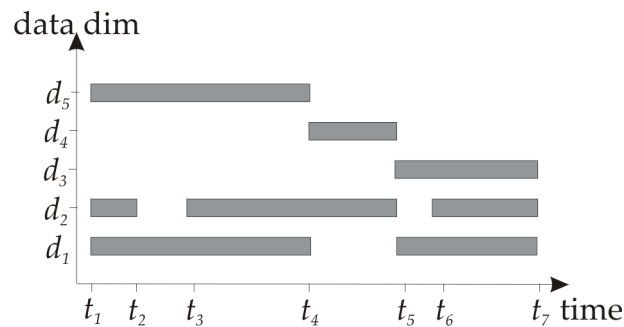


Figure 2.6: Simple example of a temporal diagram using a time line.

level for the display of temporal data. A simple example is shown in Figure 2.6.

The construction of a diagram on the basis of a time axis is presented by Tufte [1990] on the example of the railroad schedules from the Japanese Shinkansen trains. The horizontal axis of the diagram is reserved for the temporal dimension. Respective schedules for the train stations are placed along the vertical axis. In order to allow for a readable presentation of multiple train schedules, a line is drawn for every station connecting all points in time when a train stops there. As a side effect, this allows to visually deduce connections between trains at a station. The price for the hard-coded connections of train events is its inapplicability to express temporal uncertainty in such a diagram based on time axes.

For the visual display of temporal information, the *Time Line Browser* system is presented by Cousins and Kahn [1991]. At the example of patient data, this browser allows to detect temporal relationships among time-ordered data. In support of this, all temporal data is represented as events on a time line. Using a set of five operations on the time line (*slice*, *filter*, *overlay*, *new*, and *add*) allows the user to browse the set of events and locate any specific data of interest.

LifeLines as introduced by Plaisant et al. [1996] are an extension of the classic concept of time lines. A specific extension introduced is the concept of facets. These facets represent vertical segments that group similar temporal entities. For the purpose of providing a means to explore all data and structures therein, the facets can be opened and closed as desired. This possibly prevents the diagram from being cluttered by an overwhelming set of represented groups of data. Thereby, a temporal zoom-in/zoom-out is allowed for examined ranges of time. The dynamic rearrangement of events and intervals is specifically displayed. Hierarchical time lines with time cues are presented by Jensen [2003]. These extend the notion of facets and allow to collapse and expand individual temporal entities as opposed to whole groups of entities.

A simplified version of *LifeLines* are *ganttt charts*. These charts lack the ability of showing different facets of the same task. However, compared with *LifeLines*, *ganttt charts* provide the additional ability of displaying hierarchies in the temporal data. This makes them well suitable for handling of task-oriented models as illustrated

in Figure 2.7. Beale et al. [2001] present another variation of using a time line model for the representation of temporal behaviour. In order to support a CSCW project that is based on a chat system, temporal processes in the chat are presented by use of two time lines.

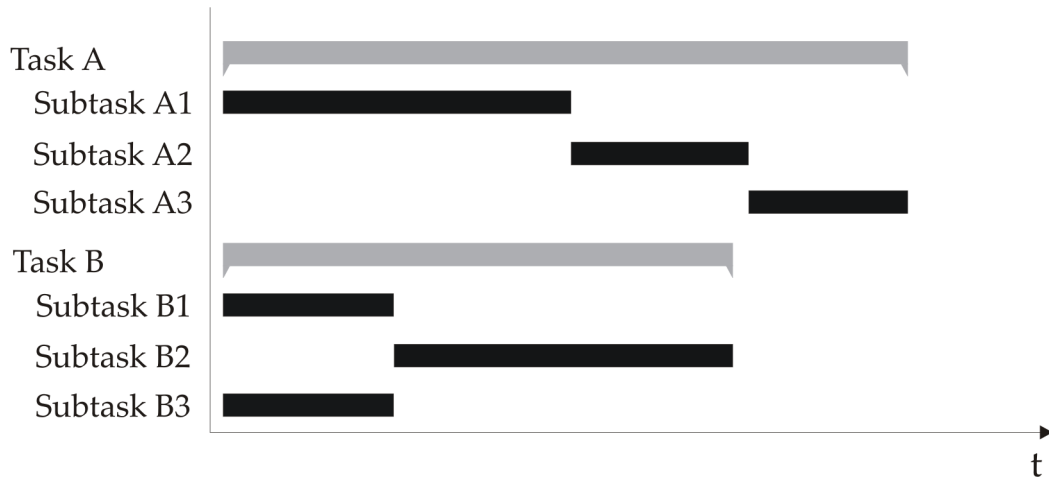


Figure 2.7: Exemplary gantt chart.

Diagrams that employ not only one but two time axes are presented as *Sets of Possible Occurrences* (SOPOs) by Rit [1986]. The two axes respectively represent the begin and end of an interval. This way, a single point in the diagram specifies a whole interval. A single SOPO is created as an area including all intervals meeting a set of modelling criteria including earliest and latest start, earliest and latest end, and minimum as well as maximum durations. An example is shown in Figure 2.8. However, compared with other diagrams based on time lines, SOPOs not necessarily provide an intuitive interface to the representation of temporal data.

Another approach is presented by Havre et al. [2002] with the system *ThemeRiver*. Therein, a river metaphor is used in order to represent multiple data dimensions that vary over time. Figure 2.9 presents an example. Input data is represented by visual data streams that flow along the time axis. Different input dimensions are thereby mapped onto different flows that are encoded by respectively varying colours. As these flows change over time, their width changes according to the data value at the respective points in time. Additional context information may be provided. In the figure, this is done at the top of the display window. Specific events are placed at their related time points. This way, these annotation events indicate possible causes for patterns in the river's flow.

Weber et al. [2001] introduce an extension of a time line by bounding the line and forming a spiral. Figure 2.10 illustrates this idea. Thereby, time-series information is mapped on the spiral and the respective spiral subparts are colourised accordingly. As analysed by Weber et al. [2001], use of the spiral results in about the same spatial

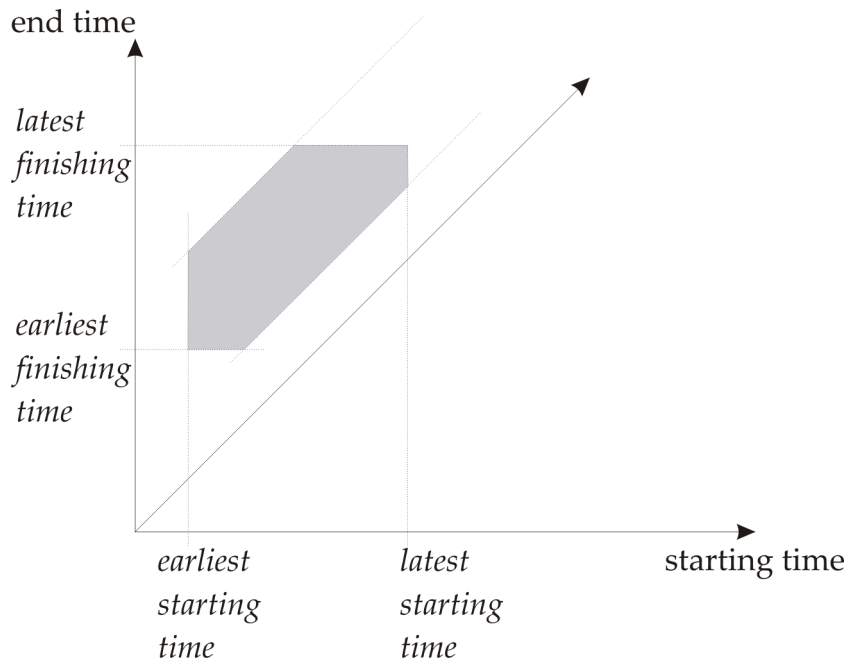


Figure 2.8: Example of a diagram showing *Sets of Possible Occurrences*. (based on Kosara et al. [2001])

requirements as a classical time line approach. In contrast, periodic patterns can easier be compared in the spiral than in the line as repetitive data sets may be placed more close to each other.

2.3.2 Flow Representation of an Impulse Response Function

For the purpose of generating real-time dynamic deformable models using simulators, James and Fatahalian [2003] introduce a precomputed data-driven state space modelling approach. That is, they precompute dynamic deformable scenes for interactive exploration. For the purpose of the final simulation responding to interactions as expected, these are precomputed as well. To robustly support runtime interactions that correlate with their precomputed counterparts, parameterised impulse response functions are sampled. The progression of this sampling is presented as a visual annotation during runtime.

Figure 2.11 shows a snapshot of using a flow representation of the impulse response function's temporal progress as presented by James and Fatahalian [2003]. Subject of the dynamic deformation is a dinosaur as shown in the lower right part of the window. The remaining part of the presented screen space is used for illustrating the respective flow of function evaluation. The bright dots near the top of the display represent the current state of functional progress. The remaining value space of the function is shown in order to provide necessary context on past and upcoming positions.

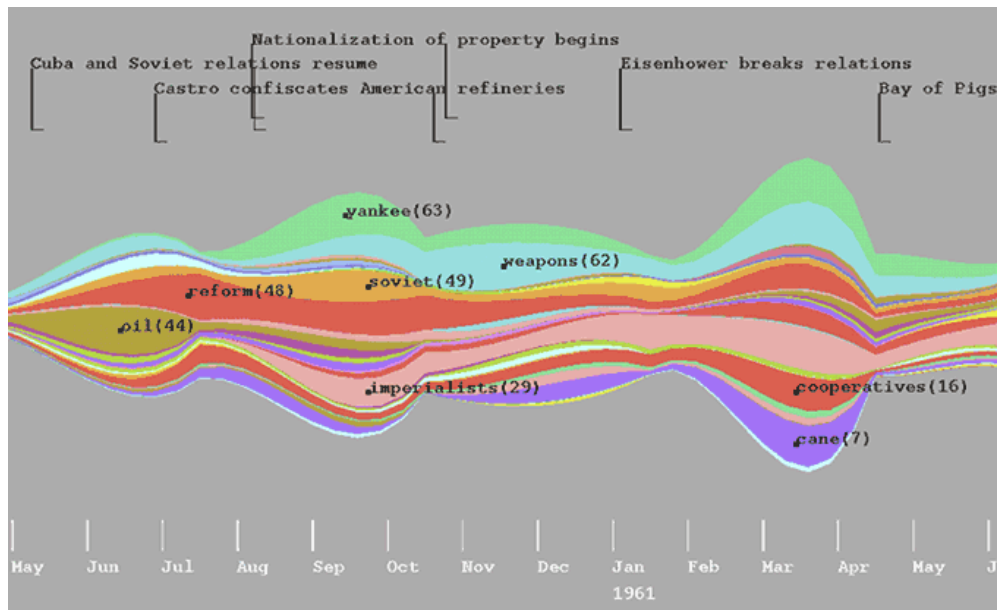


Figure 2.9: Exemplary use of the *ThemeRiver* system for the presentation of multiple input dimensions along a time line. The river metaphor enables to directly compare individual dimensions at any specific point in time. Some temporal dependent annotation information is presented above the river display. [Havre et al., 2002]

Both presentations—the flow through the value space of the impulse response function as well as the deformation of the dinosaur—are shown synchronously. On one side, this helps to understand the function evaluation, on the other side, this supports the user in interacting with the dinosaur’s deformation.

2.3.3 Dynamic Metaphor

The system *Tardis* as presented by Carpendale et al. [1999] addresses the representation of landscape dynamics over time. At the example of a simulated landscape the effect of fire covering a large area is illustrated. For this purpose, a set of landscape patterns is used. These patterns are mapped onto a spatio-temporal block as shown in Figure 2.12. For the purpose of supporting development of ecosystems, *Tardis* uses three dimensions for presentation: two spatial and one temporal. These are constructed by piling up two-dimensional landscapes along a temporal axis. The overall sum of these staples form the Tardis-block.

As the fire-affected landscape changes over time, the respective staple in the block changes with respect to its surrounding staples, as well. This is revealed by a direct comparison of both images in Figure 2.12. The left image shows the complete block. The righthand image represents only the changes for the individual staples over time. For each specific point in time the changes reflected by its staple directly correlate to the changes in the underlying data set. That is, landscape changes

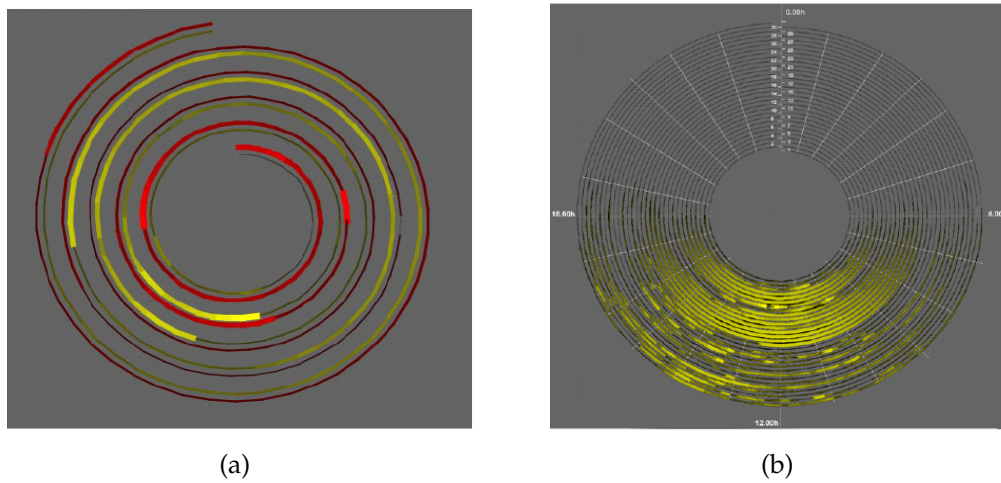


Figure 2.10: Mapping of information onto a spiral as an extension of a simple time line [Weber et al., 2001]. Subfigure (a) shows the mapping of stock values of two companies in five years on two spirals (respectively encoded in red and yellow). The use of annotative scales on a spiral is shown in (b).

caused by fire are shown.

To provide an interaction technique for the temporal landscape patterns, a book metaphor is introduced. In order to compare two specific time points of the data set, the spatial-temporal block may be opened similar to a book. This presents the user with two opposite pages: one to the left side of the block and one to the right. This is illustrated in Figure 2.13. Each of these sides represents one of the respective points in time, that are currently of interest. As the granularity of the block is variable, the temporal difference between the two pages might vary as well. This allows for the interactive analysis of a wide span of temporal intervals.

2.3.4 Dynamic Textures

The idea of mapping information onto textures is presented by van Wijk [1991] as *Spot noise*. The principle of this technique is outlined in Figure 2.14. Modelling primitives are spots of random intensity such as the example spot shown in the leftmost image of the figure. A texture is generated by drawing these spots at various positions on a plane. Finally, the spots are blended together randomly, which results in the final texture as shown in the rightmost image of the figure.

Mapping of information onto this texture is now achieved by variations of the spots. These variations include rotation, scaling, and bending by deformations of spots using a mesh [de Leeuw and van Wijk, 1995]. For the presentation of vector data, each spot thereby reflects some input vector. It is first rotated according to the vector's orientation. Scaling is applied to the spot according to the respective value of the vector. Deformations may finally be used in order to reflect overall patterns in the vector set.

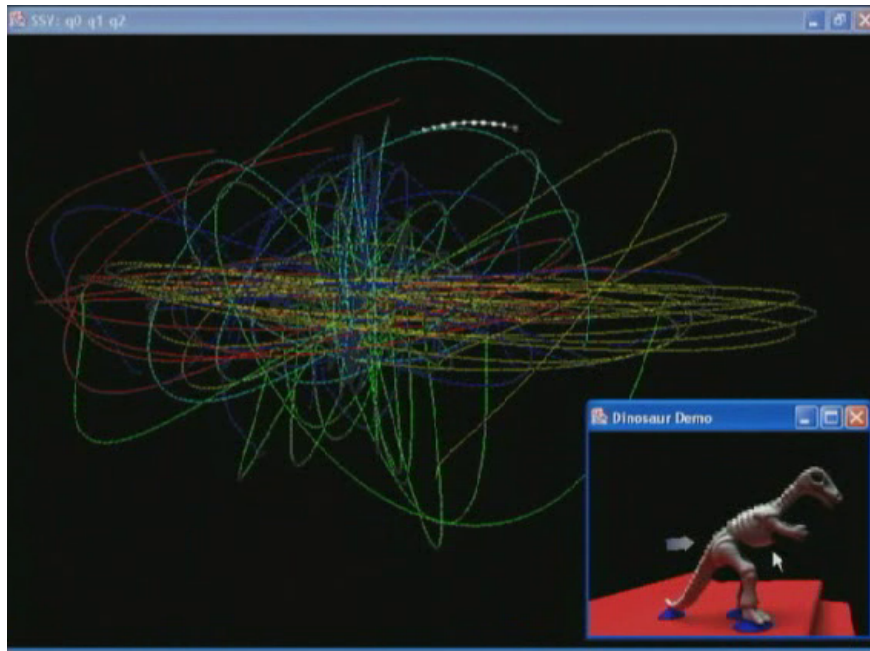


Figure 2.11: Snapshot of an animation showing impulse response function time-steps coloured by an impulse palette index. (The snapshot is taken from a video accompanying the work presented by James and Fatahalian [2003].)

The *Spot noise* technique is extended by de Leeuw and van Liere [1999] to represent the dynamic flow of information. Temporal information is thereby represented by *variations* of the constructed texture. Possible applications of dynamic *Spot noise* specifically include the illustration of vector data such as climate data or numerical simulation results. The sports are thereby positioned along streamlines and particle paths. In contrast to arrows or streamlines, the generated texture allows for a continuous display of flowing data. Figure 2.15 shows a snapshot of a dynamic *Spot noise* presentation of numerical simulation data.

2.3.5 Discussion

Dynamic presentations prove to be well suited for the representation of time and temporal behaviour. Various methods exist for the mapping of temporal information onto visual parameters. Dynamics fit this task specifically well as they are characterised by a presentation that changes over time. Two systems have been presented which use this change and represent flow-based information of the underlying data model. These techniques are contrasted by two cases of dynamics which employ visual metaphors for a representation of temporal data. One uses a metaphor-based approach and maps data onto a spatio-temporal block. The other employs dynamic textures which allows to use the geometric shape of presented

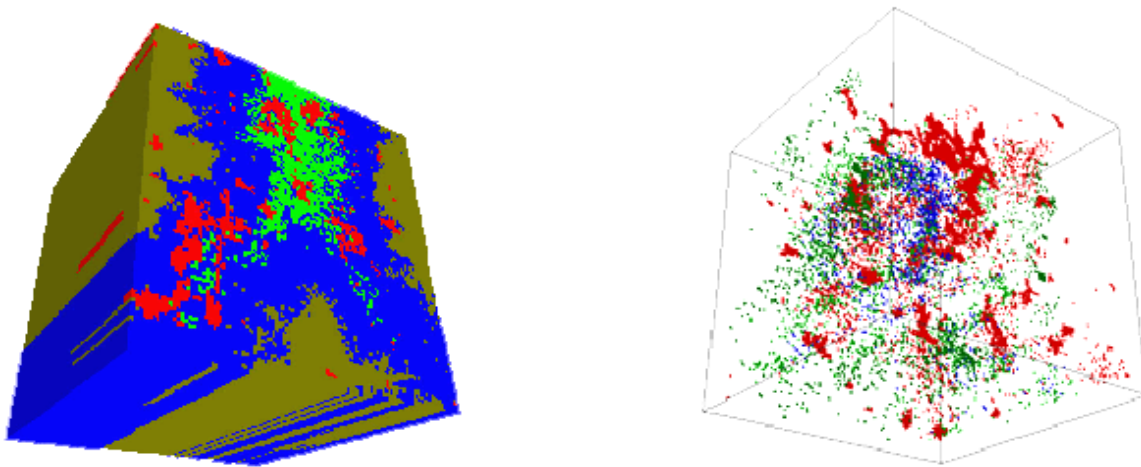


Figure 2.12: Landscape patterns as used by the *Tardis* system. [Carpendale et al., 1999]

objects to convey further data attributes.

The expressive power of dynamics promises to be useful for more than just the representation of time. Therefore, a hierarchy model of dynamic presentation variables is introduced in the following section. This helps to classify dynamic techniques and to point out their respective effects.

2.4 Hierarchy Model of Dynamic Presentation Variables

Dynamics span a broad range of presentation techniques: from simple colour blinking to sophisticated motion patterns. Changes in rendering styles of hybrid presentations are considered to be dynamic as well. All the individual techniques are therefore respective subsets of dynamics. As motion is part of this collection, dynamic presentations in general promise to share at least some of its expressive power.

For a discussion of using well parameterised dynamics for illustrations, it is meaningful to differentiate between their dynamic character and influence. This section concludes the chapter on fundamentals of dynamics and addresses the notion of different dynamics with regard to each other. The individual techniques are arranged in a set of four layers: colour blinking, variable rendering styles, distortions, and motion. For the purpose of maximising an effective use of dynamics, the techniques of these layers are ranked according to their perceptive influence. Based on this ranking, a set of general design guidelines for the use of dynamics is derived. This is supported by a discussion of presentation effects of the respective layers.

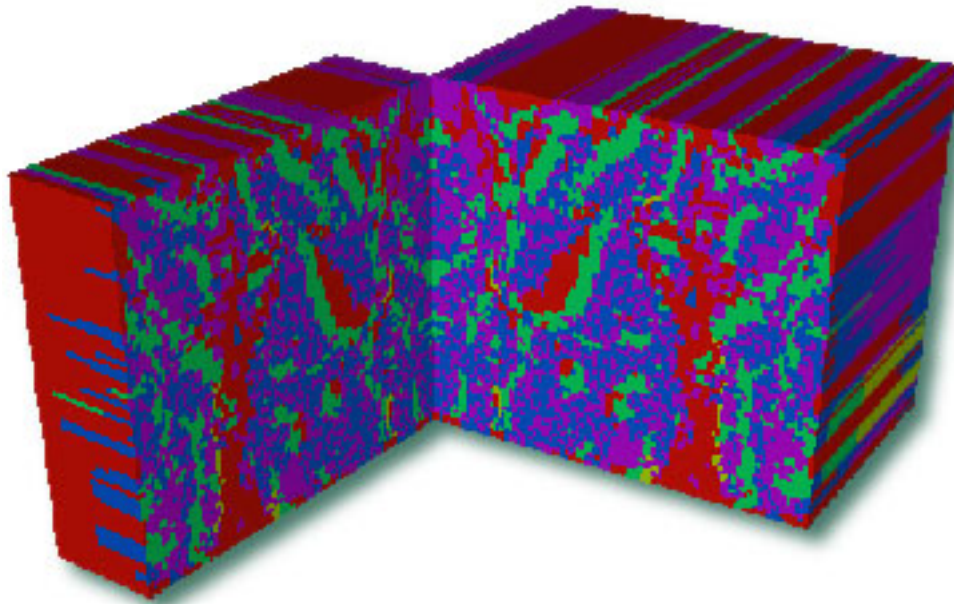


Figure 2.13: The book metaphor introduced by the *Tardis* system. In order to compare represented data characteristics at specific points in time, the user may browse through the temporal block as through a book. In case the block is *opened*, the opposite *pages* represent consecutive points in time. [Carpendale et al., 1999]

2.4.1 Partition of Dynamics into a Layer Model

A partition of dynamic presentation techniques in a layer model is presented in Figure 2.16. As of its fast perception and ease of use, colour blinking forms the bottom layer of the model. Section 3.2 will discuss the change of colour as a classic illustration technique which can mainly be classified as being of static nature. However, a continuous and repetitive use of colour changes results in blinking. This blinking is regarded as most basic form of dynamics that has long been used in awareness tools [Bartram, 2001]. For the purpose of attracting and directing visual attention, much use is made of blinking as a human interrupt.

Variable rendering styles form the next layer. These include changes of a specific rendering style as well as combinations of different styles and modifications thereof. Details of the design and implementation of dynamic changes of rendering styles will be presented in Section 3.4. Generally, the set of techniques on this layer can be divided in two categories: (1) fading of otherwise static renderings by use of varying transparency and (2) the simultaneous use of multiple rendering styles forming hybrid renderings. The effect of variable rendering styles is always of local nature. That is, the dynamics of this layer only influence a directly affected object. The remainder of the scene is not touched.

Even though in-depth studies on their affect on cognitive load have not yet been

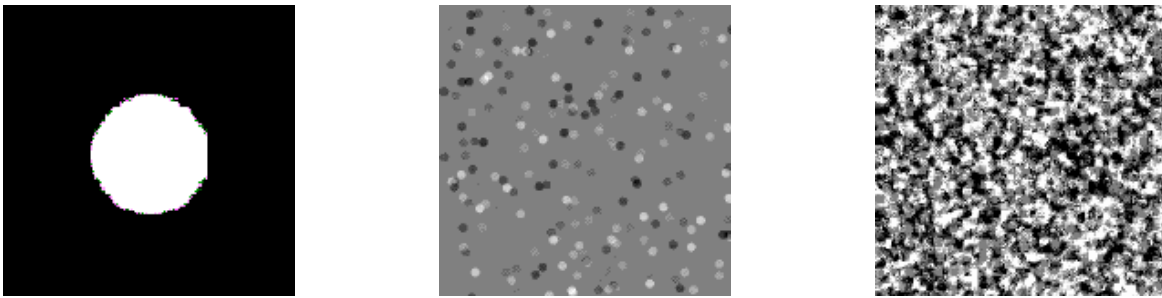


Figure 2.14: The principle of *Spot noise* texture construction. A single spot is shown to the left. The middle image represents an in-between state of drawing a set of spots on a plane. Blending them together randomly, finally results in the texture of the rightmost image. [de Leeuw and van Liere, 1999]

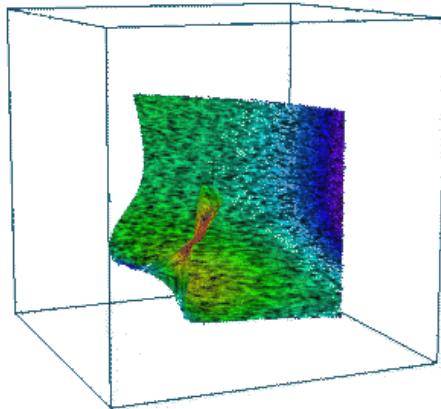


Figure 2.15: Dynamics by texture variation as introduced by the *SpotNoise* system. [de Leeuw and van Liere, 1999]

published, variable rendering styles are expected to share a fair amount of influence in this regard with blinking. This assumption is based on studies comparing cognitive load and power of blinking and motion [Bartram, 2001, ch. 5]. However, rendering style variations are parameterisable in a more extended and flexible way than blinking. This is due to the circumstance that all parameterisation dimensions of blinking are available for dynamic rendering styles as well. Whereas blinking switches constantly between two alternating colours, two styles alternate analogously. Furthermore, the whole range of parameter dimensions for each rendering style employed adds to this set. Overall, this rich parameterisation space allows for a more fine-grained control of dynamics on this layer compared to blinking. This positively effects its perceptive influence and justifies these two layer's ordering.

The third layer represents dynamics by distortions. These are constructed by modifying an object's shape, size, or both. A broad range of distortion techniques is available [Carpendale et al., 1997]. Generally, the catalogue of distortion techniques

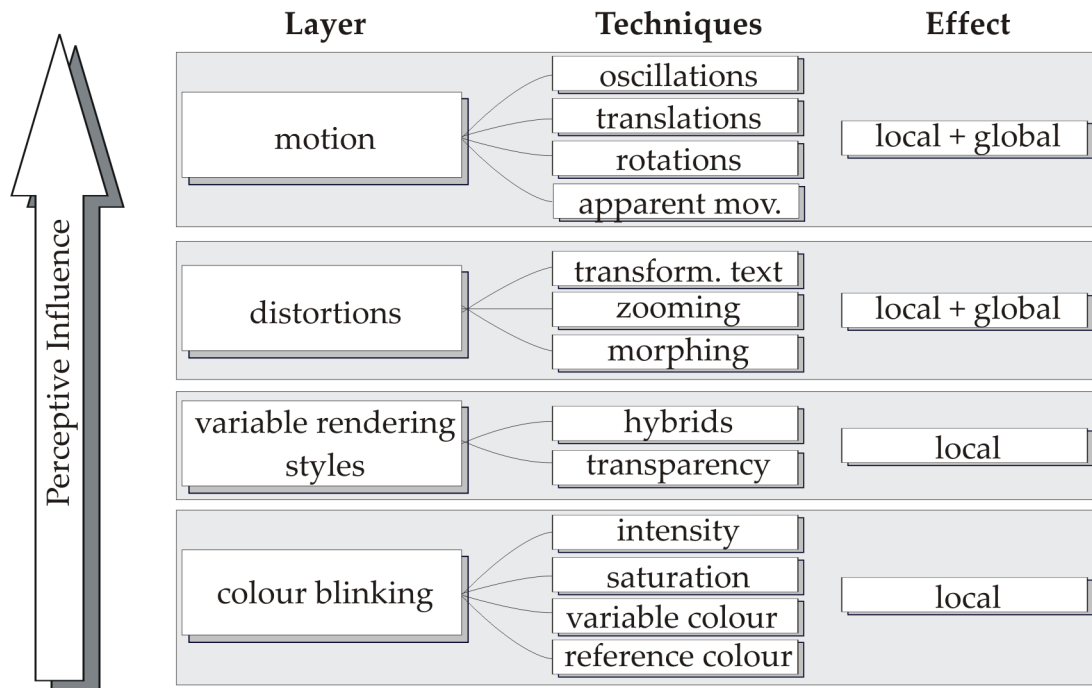


Figure 2.16: Partition of dynamic presentation techniques in a layer model.

consists of morphing [Blanz and Vetter, 1999, Cohen-Or et al., 1998], zooming [Furnas, 1986, Preim et al., 1997], and transformation such as variable text presentations [Maglio and Campbell, 2000]. Dynamic distortion histories based on zooming will be presented in Section 3.5.

Distortion techniques provide for local effect as well as global effect. Local effect can be generated by modifying only a well-defined region. Examples are the morphing of an object in case the smallest possible convex hull of the object is thereby not changed in size. Furthermore, text transformations are characterised by local effect as long as the textual region is not modified. This is illustrated by Figure 2.17 where the legibility of text in a region is turned up in order to provide dual use of image space by presenting text explanations for images *within* image space [Chigona and Strothotte, 2002a].

Any change of an object's size and shape influences not only the object itself, but also the surrounding parts of the scene. This is regarded as global effect. For the above example of dual use of image space by text modifications, this holds in case the textual region is modified as illustrated in Figure 2.18.

The layer with the highest perception influence of the model is represented by motion. Thereby, the notion of motion includes explicit motion techniques as well as apparent motion techniques. Explicit motion spans the whole range of modifications of an image over time caused by a position change or modification of the kind of object presentation [Masuch, 2001, ch. 5]. Specifically, explicit motion techniques

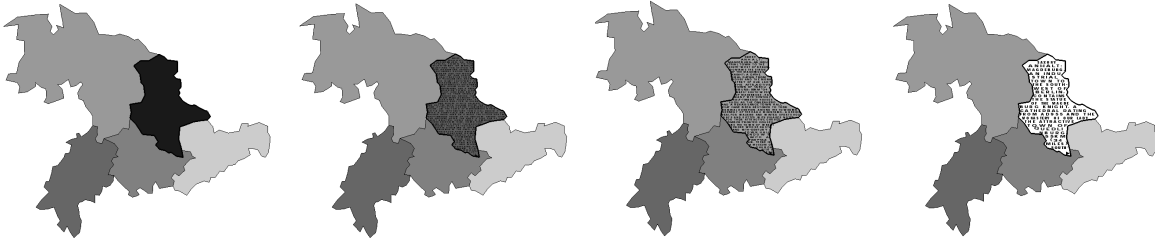


Figure 2.17: Dual use of image space by text modifications of a region. The effected region is modified. As the modification influences only the region itself, the effect of this distortion is of local nature. [Chigona and Strothotte, 2002b]

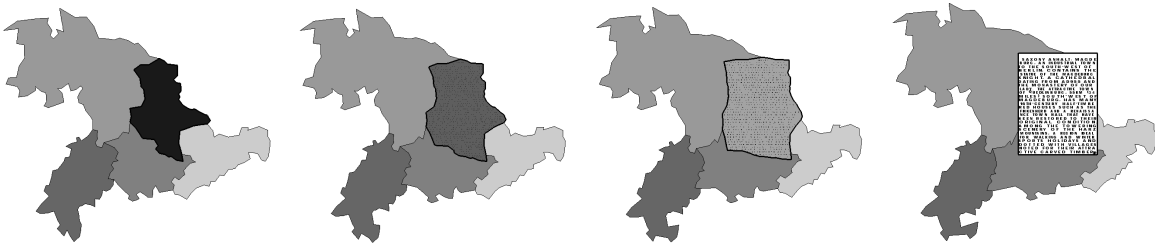


Figure 2.18: Variation of Figure 2.17 where the effected region is modified in shape and size [Chigona and Strothotte, 2002b]. This also influences the surrounding scene elements. Therefore, the effect is of global nature.

consist of rotations, translations, and combinations thereof. Motion can be of single, repeating, or oscillating character. As introduced above in the discussion of the dynamics stimulus window, apparent motion describes the *illusion* of an explicit motion technique (see also [Goldstein, 2002, ch. 8]).

Similar to the distortions layer, the effect of motion may either be of local or of global effect. Translation-based motions affect not only the moving object itself but its surroundings as well. Therefore, these are respected as global dynamics. In case of rotation-based motions, the effect depends on the rotation centre and shape of the affected object. Rotations around an regular object's geometric centre provide local dynamics whereas any other rotation results in a global effect. Any oscillation inherits the characteristics of its underlying translation, rotation, or combination thereof.

2.4.2 Composite Dynamics Patterns

Techniques from individual layers may of course be combined in order to form composite dynamics patterns. The effect of composites is determined by logical derivation: In case one of the involved techniques provides a global effect, the whole combination provides a global effect.

2.4.3 Local versus Global Dynamics

The visual effect of the two lowest layers in the hierarchy model provides local effect only. However, global judgements to objects can still be derived. This is supported by a study addressing the perception of global object information based on the integration of local stimuli [Gerlach et al., 2002]. For this, stored structural knowledge about affected objects holds responsible. Areas previously associated with access to stored structural knowledge are found during the processing of recognisable stimuli.

On the other hand, the study also reveals that even an exclusive use of global object information allows for the recognition of concrete objects. This cognitive resolving task is based on the same structural knowledge as before. It is to be noted, that the effect does not hold for unrecognisable stimuli. That is, natural objects are easier and faster recognised than artifacts.

2.5 Summary

It is the intention of this chapter to outline fundamentals of dynamics for the purpose of presenting their expressive potential as well as a motivation for constraints of their use. The effectiveness of an illustration results from task suitability of all illustration techniques used [Knight, 2001]. Any single generic collection of presentation variables does not suit all possible application areas. This holds regardless of them being of static or dynamic nature.

It has been shown that dynamic presentation techniques are capable of communicating their classic illustration goal: the representation of time and temporal behaviour. Furthermore, the potential of dynamics to enrich the expression set of an illustration system is discussed. As pointed out, any use of dynamics needs to be subject of constraining parameterisation. The dynamics stimulus window in combination with a hierarchic layer model of the set of dynamics provide the formal basis for this.

The following chapter introduces a set of selected dynamic presentation techniques in detail. Temporal aspects of controlling dynamics-enhanced presentations will be discussed in Chapter 3.

3 Illustrative Dynamics by Motion and Non-Realism

The essence of illustration is a form of communicating information about a model. While a variety of illustration techniques exists for this purpose, most of them can be classified as being static by nature. These do not necessarily provide enough communication bandwidth in order to meet the illustration goals.

Use of photorealism has the advantage of a representation of rendered objects that appears to be true to the original. However, for an illustration goal of presenting some—possibly complex—context information, the availability of a variety of detail information does not necessarily increase the communication bandwidth. In order to focus on objects of interest in a scene, this information can be aggregated and simplified visually. This directly leads to non-photorealistic representations. To be precise, the set of non-photorealistic images includes all renditions that are not photorealistically illuminated.

This way, non-photorealism acts as an extension of the set of classic presentation variables as presented by Noik [1994]. So far, these variables include shape, position, materials, and metaphorical mappings. As discussed in the previous chapter, some work has been done to evaluate the potential of motion as a presentation dimension of its own. This chapter contributes a set of dynamic presentation techniques. These include motion-based methods, distortions, and changing combinations of multiple rendering styles, simultaneously used.

The chapter is divided into two major parts. These parts are illustrated in Figure 3.1: At first, the notion of an illustration target function is defined. This function provides the context and goals of an illustration. In addition, the notation of a presentation framework is introduced here. This framework is used consistently throughout the remainder of this work. An overview of classic and static illustration techniques follows. The second part of this chapter introduces dynamic presentation techniques in Sections 3.3 up to 3.5. Three types of dynamics are discussed: motion-based dynamics, hybrid rendering style variations, and distortion histories.

The figure points out how this chapter is glued to the remaining parts of this work. Fundamentals of dynamics are presented in the previous chapter. The next chapter picks up the dynamics presented here. Thereby, temporal controlling of dynamics will be introduced on a script basis.

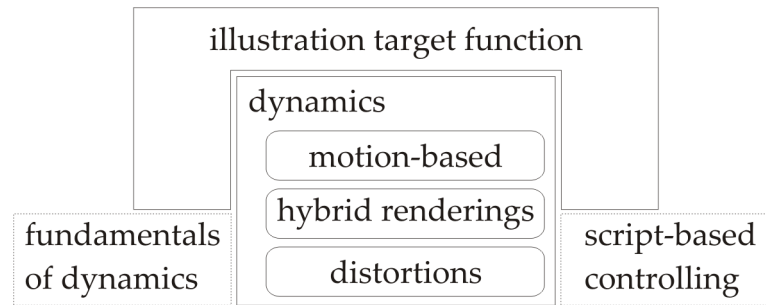


Figure 3.1: Structural chapter overview.

3.1 Illustration Target Function

Target functions are used to express expected behaviour of a system and provide a means of comparing the system's output with these expectations. They are commonly used in a variety of areas such as automatic learning of neural networks [Fukumizu, 1999].

In order to determine the effectiveness of an illustration and the adequacy of employed illustration techniques, the illustration goal needs to be materialised. This is accomplished by defining an illustration target function. In addition to expressing the illustration goal, this function respects the represented world as well as its modelled counterpart and a user model. Following Wegner [1997], the illustration target function captures semantic properties of the illustration domain by a syntactic representation for the pragmatic benefit of users. Thereby, the *user information needs* [Baeza-Yates and Ribeiro-Neto, 2002] are a key element in retrieving information by means of an illustration.

Four steps are used in this section to describe the illustration target function. First of all, some context on target functions and user models is presented in Subsection 3.1.1. Before the definition of the function is given in Subsection 3.1.4, the notation of the underlying framework is outlined in 3.1.2 along with the target function's model basis in 3.1.3. Finally, Subsection 3.1.5 discusses evaluation issues with regard to illustration modelling.

3.1.1 Context

Use of an iteratively defined target function for modelling purposes is presented by Zorc [1995]. This function concentrates on defining a single object of interest. Contextual information is not explicitly respected. That is, the model wholly concentrates on the modelling subject without regard to external model constraints, the represented world, or the modelling user. Steps of iteration are used for gradually refining the model described by the function. For the example of optical film presented by Zorc [1995], this results in steady refinements of multiple thin layers

defining the film.

Cohen et al. [2001] motivate the use of an *user model* for the automatic evaluation of system responses. The context of their work is in fulfilling the complex information needs a user has. They employ the user model to address the question of when and how a system should react to user interaction. The model is evaluated before any output of the system is presented. For that purpose, the system needs to pass a set of tests before reaction is shown. Each test compares internal system information with assumptions about the user's needs and information based on the user model. Depending on the test results, the system might acquire supportive user interaction in order to solve the task at hand.

The role of an user model plays in interactive illustrations is also of central nature. Interactivity of an illustration directly leads to higher load put on the user. This may be rewarded with an increase in illustration result quality [Beall et al., 1996]. For the purpose of targeting this repayment of interactive illustrations, an illustration target function has to include a user model. This model allows to respect the user's goals and capabilities as well as system restrictions that address possible illustration constraints.

3.1.2 Framework Notation

The design of the presentation framework is formally based on the work presented by Kreuzeler and Schumann [2002]. Their model was developed in the context of Visual Data Mining. But its general nature allows for application in other domains as well. In principle, the model is based on the definition of *information objects* IO_i that combine to an *information set* \mathbf{IM} :

$$\mathbf{IM} = \{IO_1, \dots, IO_n\} \text{ with } IO_i = IO_j \Leftrightarrow i = j, 1 \leq i, j \leq n, \text{ and } i, j, n \in \mathbb{N}.$$

Each information object represents some real world data. In order to parameterise these objects according to their represented data characteristics, an attribute function *attr* is provided:

$$\mathbf{AM} = \text{attr}(\{IO_1, IO_2, \dots, IO_n\}) = \{A_1, A_2, \dots, A_k\}$$

with $A_i = A_j \Leftrightarrow i = j, 1 \leq i, j \leq n$, and $i, j, k, n \in \mathbb{N}$. Thereby, \mathbf{AM} is the *attribute set* of the respective information objects. The individual attributes of this set act as dimensions of information objects and span an information space \mathbf{IR} . For the purpose of defining relations between IO_i the *information structure* \mathbf{IS} is introduced as $\mathbf{IS} \subseteq \mathbf{IM} \times \mathbf{IM}$.

This framework formalisation has proved applicable for a variety of application areas. As already noted, its roots are in Visual Data Mining. By extending the formal basis with a set of application dependent preprocessing functions, the framework can be adapted to the various stages of data processing and presentation.

Kreuseler and Schumann [2002] present examples for three such cases: preprocessing, reduction of the information space, and visualisation. For preprocessing, interactive user-driven approaches are used as well as automatic obtainment of structures and patterns in the data by algorithmic computational procedures. For reduction of the gained information space, a set of exemplary algorithms is employed. This includes visual previews by *data tables*, *Self-Organising Maps*, and *Dynamic Hierarchy Computation*. Exemplary visualisation techniques presented for use within this framework include *Magic Eye View* for hierarchy visualisation and *Shape Vis* for exploration of multidimensional information sets.

3.1.3 Model Basis

An interactive system which is supported by a visually enriched user interface reflects multi layered representation models. This set of models consists of a representation of a world combined with a human or mechanical interpretation [Wegner, 1997]. For illustrations, this set's granularity is refined by distinguishing between a *represented world* and a *modelled world* and furthermore including a distinct *user model*.

The highest abstraction level acts as represented world, which reflects entirely the system's underlying semantics. For different illustration contexts, different represented worlds exist. Overall, this level is classified as \mathcal{W}^{rep} , a set of represented worlds:

$$\mathcal{W}^{rep} = \{world_1^{rep}, world_2^{rep}, \dots, world_n^{rep}\} \quad n \in \mathbb{N}.$$

Thereby, each represented world describes a set consisting of the user model, the scene model, relations between different components of the scene, and operations defined in this world:

$$world^{rep} = \{user, scene, relations, operations\}.$$

A typical user model is determined by:

$$user = \{percept, actions, scope\}.$$

The perception of the user as well as her or his actions are represented by the sets *percept* and *actions*. The *scope* describes a set of user model parameters that influences the individual target function of a user with regard to the system.

The *scene model* describes a geometric description of the illustration subject. With regard to the illustration target function, the source of this model is not of relevance. Specifically, a distinction between geometry-based and data-driven illustration models is not addressed. Both may fit well into an illustration scenario. Examples of geometry-based models typically include medical or technical objects that are to be explained by an illustration. A common source for data-driven models is

the whole field regarded as *information visualisation* wherein arbitrary—and often numerical—data is mapped onto geometric representations.

Semantic background for individual elements of the scene model is provided by *relations*. These relations define an ordering of all scene elements as well as any groups of these elements. Various structures may be used to express the relations. Typical examples include *relational filters* and *scene graphs* [Strauß and Carey, 1992]. It is to be noted specifically that relations define the context of a set of objects and not how to render them.

A subset of the represented world is expressed as modelled world:

$$world^{mod} = world^{rep} \setminus user = \{scene, relations, operations\}.$$

That is, the modelled world reflects the represented world excluding the user model. A concrete instance of $world^{mod}$ includes all information of $world^{rep}$ that is relevant for fulfilling the requirements defined by the user model.

Another part of the modelled world are *operations* bridging between the scene and the user model. These operations include interaction techniques, use of presentation variables, and scene modifications. Modifications target the scene elements and are a combination of state changes as well as adding new elements to the scene and removing elements from the scene. An explicit part of the operations is the temporal model. A concrete temporal model for this purpose will be discussed in the following chapter. This ensures consistency as well as regard to temporal constraints for all presentations.

Figure 3.2 shows the individual components building the model basis of the illustration target function. The individual parts are grouped according to their respective role in the function.

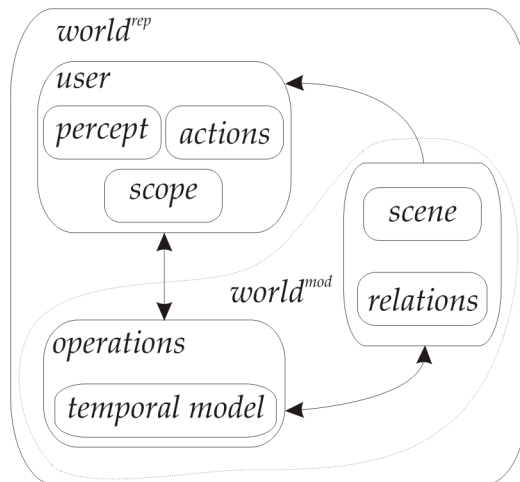


Figure 3.2: Schematic representation of the illustration target function's components.

3.1.4 Function Definition

The illustration target function itf is a tuple $(\mathcal{S}, \mathcal{M}, \mathcal{P})$ consisting of the user's scope \mathcal{S} , the illustration model \mathcal{M} , and the system properties \mathcal{P} where:

- \mathcal{S} specifies the user's scope and is given as $\mathcal{S} = \{s_1, \dots, s_n\}$. It is defined by the formulation of a set of questions to be answered by the illustration process.
- \mathcal{M} represents the illustration model and correlates to the scene model to be illustrated and *relations* from the above model basis. Therefore, \mathcal{M} constitutes as $\mathcal{M} = \{\text{scene}, \text{relations}\}$.
- \mathcal{P} denotes properties of the illustration system and includes operations and system restrictions: $\mathcal{P} = \{OP, SR\}$.

The function describes the illustration model as well as all questions to be answered by the illustration. It implicitly bridges between the user's target function with regard to the illustration and any system properties providing the illustration context.

Correlating the illustration model \mathcal{M} with the presentation framework, *scene* directly corresponds to an information space \mathbf{IM} whereas *relations* merge in an information structure \mathbf{IS} , thus $\mathcal{M} = \{\mathbf{IM}, \mathbf{IS}\}$. Individual objects $IO_i \in \mathbf{IM}$ are representations of real world objects in an illustration scene. The model is not only the illustration subject but provides the overall context limitations. This context might be enriched by another illustration component: a set of predefined model annotations. While these might include some static annotation material, dynamic querying of a variety of annotation sources is common. This will be discussed in some more detail with specific examples in Chapter 6.

Operations as part of the system properties \mathcal{P} are defined as a set of functions:

$$OP = \{doi(IO), \mathcal{D}(\mathbf{IM}), INT(\mathbf{IM})\}.$$

The set SR of system restrictions is defined as:

$$SR = SR_\delta \cup SR_m.$$

Part of the operation set is a degree of interest function, $doi(IO)$, that determines an interest value for any given object $IO_i \in \mathbf{IM}$ in the scene. This value correlates with the user's scope as defined above but is also influenced by system behaviour and user interaction. Support of this interaction as well as any restrictions is represented by $INT(\mathbf{IM})$. That is, for the whole illustration model presented by an illustration system, a set of interaction techniques is available. This provides an interface to the model and available annotation information. In general this set holds valid for the whole model as represented by \mathbf{IM} . The definition allows to respect specific ways

of interaction for individual objects, though:

$$INT(\mathbf{IM}) = \{int_1(IO_1), int_2(IO_2), \dots, int_n(IO_n)\} \quad n \in \mathbb{N}.$$

This is reasoned in providing different means of interacting with different illustration entities, such as text and 3D renderings.

In addition to the $doi()$ and $INT()$ functions, a set of presentation techniques is available as a system property. This is denoted as $\mathcal{D}_p(\mathbf{IM})$. In the context of this work, this set specifically includes dynamic presentation techniques. However, this is not a fundamental restriction required by the illustration target function definition. Precisely, $\mathcal{D}_p = \mathcal{D}_s \cup \mathcal{D}_d$ includes all available presentation variables as a combination of static (\mathcal{D}_s) and dynamic (\mathcal{D}_d) techniques:

$$\mathcal{D}_p = \mathcal{D}_s \cup \mathcal{D}_d = \{d(IO) : d(IO) \in \mathcal{D}_s \text{ or } d(IO) \in \mathcal{D}_d\}.$$

Different presentations may be used for different objects. Furthermore, elements of this set do not need to be disjoint. That is, any two information objects may well be presented using the same presentation technique. Overall, the parameterisation of \mathcal{D}_p represents the attribute set \mathbf{AM} of all information objects. The discussion in the remaining parts of this chapter concentrate on \mathcal{D}_d . For simplicity reasons, this will be referred to as \mathcal{D} .

The union, $SR_\delta \cup SR_m$, of the sets SR_δ and SR_m defines the set of system restrictions. Similar to the presentation techniques defined above, this set consists of all the elements of SR_δ and SR_m together:

$$SR_\delta \cup SR_m = \{sr : sr \in SR_\delta \text{ or } sr \in SR_m\}.$$

Thereby, elements of SR_δ represent the temporal restrictions manifested by a temporal parameterisation function. Details of such a function will be discussed in the following chapter. Elements of SR_m are application dependent restrictions that need to be modelled explicitly.

Using the illustration target function for describing an illustration goal is outlined in Example 3.1. Figure 3.3 shows the engine model used in the example.

Example 3.1 Given is a geometric model of an engine. Using this model, all elements of the cooling system are to be illustrated as well as flow through this system.

The scope is defined as $\mathcal{S} = \{s_1, s_2, s_3\}$ with:

- s_1 Which parts make up the cooling system?
- s_2 What is the flow through the cooling system?
- s_3 Is the workload of the system balanced?

The information space \mathbf{IM} of the model \mathcal{M} is given as a triangulated face set. The model's information structure for defining relations of all $IO_i \in \mathbf{IM}$ is modelled by a scene graph containing all geometric elements ordered with respect to each other.

The system properties are defined as $\mathcal{P} = \{doi(\mathbf{IM}), \mathcal{D}, INT(\mathbf{IM})\}$. The degree of interest is defined as a boolean function:

$$doi(\mathbf{IM}) = \begin{cases} 1 & \text{for all elements of the cooling system} \\ 0 & \text{for all other parts of the geometric model.} \end{cases}$$

Assuming

$$\mathbf{IS}_{cs} = \{IO : IO \in \text{cooling system}\}, \quad (3.1)$$

the *doi* function is defined as:

$$doi(IO) = \begin{cases} 1 & \forall IO \in \mathbf{IS}_{cs} \\ 0 & \text{otherwise.} \end{cases}$$

The method set constitutes of $\mathcal{D} = \mathcal{D}_m \cup \mathcal{D}_{cs}$ with:

- \mathcal{D}_m global presentation of the whole model, and
- \mathcal{D}_{cs} specific presentation of all $IO \in \mathbf{IS}_{cs}$ as defined in equation (3.1).

Following the interests expressed by \mathcal{S} , the specific case of $\mathcal{D}_m \subset \mathcal{D}_{cs}$ holds. This requires that no part of the engine model is presented with a method $d \in \mathcal{D}_{cs}$ in case it does not belong to the cooling system.

Finally, the set of interaction techniques required for fulfilling the target function is a union $INT(\mathbf{IM}) = INT_g(\mathbf{IM}) \cup INT_{cs}(\mathbf{IM})$. Thereby, INT_g specifies global interaction and navigation techniques, suitable for exploring the whole engine model. Interaction with all objects $IO : doi(IO) = 1$ is expressed by:

$$INT_{cs} = \{int(IO) : IO \in \mathbf{IS}_{cs}\}$$

with \mathbf{IS}_{cs} from equation (3.1) and $int(IO)$ providing a means of selecting an object. This way, the individual parts of the cooling system can be examined in detail whereas the remaining engine model may be explored for context purposes. \square

More details about the specifics of rendering in Example 3.1 as well as more images of the resulting illustration will be presented in Section 3.4.1 and in a concrete illustration scenario in Chapter 6.

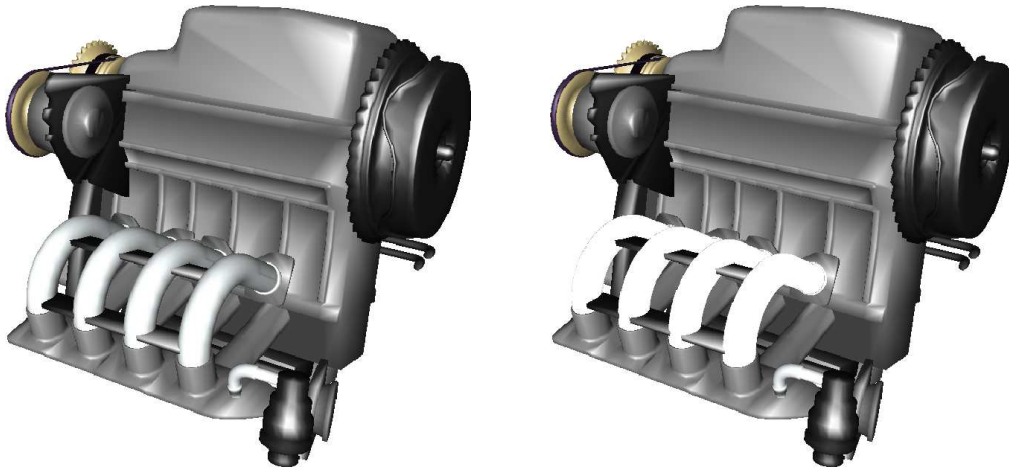


Figure 3.3: Geometric engine model as used in Example 3.1. The left image shows the illustration model of an engine without any emphasis techniques employed. The righthand image shows an exemplary use of hybrid rendering styles for pointing out specifics of the engine’s cooling system.

3.1.5 Evaluation and Limitation

A fully automated evaluation of the target function is not subject of this subsection. This is specifically due to the intention of the function as described above. The goal of the illustration target function is to express the illustration task. This supports the modelling process. This support is due to the central goal of the modelling process being to meet the illustration task. This can only be achieved in case this task is both: well known and specified clearly. Therefore, a fully defined illustration target function points out interests to be addressed by an illustration and some directions of meeting these interests.

Some restrictions apply to any instantiation of the function, though. These address the relationship between the modelled and the represented world. This relationship is to be *complete* and *sound*. Soundness requires that the modelled world $world^{mod}$ captures behaviour in the represented world $world^{rep}$. Completeness builds on soundness by requiring that *all relevant* behaviour of $world^{rep}$ is captured by $world^{mod}$. Thereby, the notion of *relevance* with regard to capturing behaviour correlates to fulfilling the scope S of the user model. These two relationships limit the overall flexibility of the target function. *Soundness* restricts the creation of a modelled world such that it does not depend on information external to the illustration target function. *Completeness* prohibits to leave out information from the represented world that is of relevance to fulfilling the function. Figure 3.4 points out this two-way caused limitation of the illustration target function.

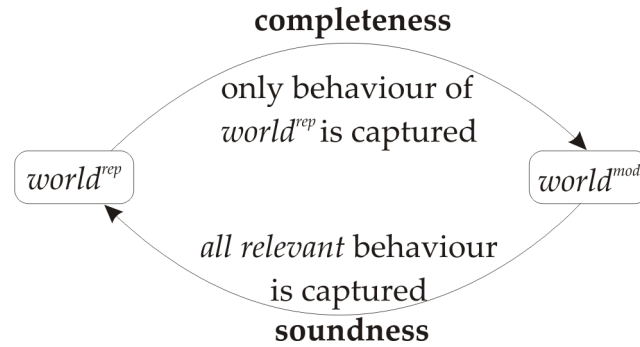


Figure 3.4: Restrictions of the illustration target function due to limited expressability of the represented world by the modelled world (based on Wegner [1997]).

3.2 Classic and Static Illustration Techniques

Before a set of dynamic presentation techniques is presented in the main part of this chapter, this section discusses classic presentation techniques. These form the basis for dynamic enhancements. All presented techniques are in principle of static nature. As the following sections will point out, at least some of the techniques shown here might be used as part of composing dynamic presentations.

Specifically, subsections 3.2.2 to 3.2.4 present selected classic techniques that are used for dynamics construction later on. This presentation is based on a discussion of two classification systems for illustration techniques in subsection 3.2.1. Finally, subsection 3.2.5 concludes this excursion into classic presentations.

3.2.1 Classification

Various approaches have been published providing overviews of presentation techniques [Noik, 1994, Preim and Ritter, 2002, Schroeder et al., 1998, Schumann and Müller, 2000]. This section presents two approaches for classifying illustrative presentation variables: an early overview of presentation emphasis techniques by Noik [1994] and a more recent classification by Preim and Ritter [2002]. Both collections specifically target techniques suitable for illustration purposes and provide classifications of these techniques.

Noik's Six-Dimensional Presentation Space

Noik [1994] describes an abstract space of presentation techniques for emphasis purposes. This space addresses an initiation of a common framework for comparing emphasis techniques in the context of relational data visualisation. For this purpose, Noik divides his presentation space into six dimensions. Each dimension describes an emphasis property of presentation. In order to refine the six-dimensional gran-

ularity to some degree, a set of subtypes is created for some of the properties. The resulting classification is shown in Table 3.5.

Emphasis Dimension	Subtype	Comment/Description
Transformation	Vis-to-Vis	The emphasised view is obtained from the normal view.
	Graph-to-Vis	The emphasised view is obtained from graph topology.
Emphasis Technique	Implicit	Derive emphasis from effect of point perspective in 3D.
	Filtered	Display of a subset of elements and suppressing the rest.
	Distorted	Distortion of sizes, shapes, and positions.
	Adorned	Variations of other variables such as colour, shading, and line style, and thickness.
Priorities	Supplied	Priorities supplied by user or a priority algorithm.
	Built-in api	Priority computation by a built-in priority algorithm.
	Built-in dist	Computation of the distance by a built-in priority algorithm.
	Client api	Possibility of client-specific importances.
	Client dist	Possibility of client-specific distances.
Focal Points	None	No notion of focal points used.
	Single	One single focal point used.
	Multiple	Multiple focal points used.
Animation	None provided	Indicate whether or not animation is used for emphasis.
Inputs	Sequences	One-dimensional data.
	Hierarchies	Hierarchical graphs.
	Flat graphs	General graphs.
	Nested graphs	Graphs with hierarchic nesting allowed.
	Beyond nested graphs	Arbitrary graphical presentation.

Table 3.5: Six-dimensional space of presentation emphasis techniques presented by Noik [1994].

The *transformation* dimension describes the kinds of transformation an emphasis algorithm uses. Two types are presented here: a *visualisation-to-visualisation* transformation where the emphasis is derived from the normal view and a *graph-to-visualisation* transformation characterised by using a graph topology for determining an emphasis.

The somewhat misleading name *emphasis techniques* for the second presentation dimension describes types of views for adding emphasis to a presentation. An *im-*

PLICIT technique derives an emphasis by evaluating point perspective in 3D. Nearby points are presented more obtrusively, that is, they are possibly loomed large, whereas distant points are presented in a less dominant manner. A *filtered* technique uses ways of determining a subset of the original scene and presenting only this by use of an emphasis technique. The remaining parts of the scene are either untouched or completely suppressed. *Distorted* presentations modify objects in shape, size, or position. Emphasis techniques are classified as being *adorned* in case any other presentation variable is used. These include—without being limited to—colour, shading, line style, and thickness.

Priorities describe means of obtaining relevance values usable for applying any emphasis. Two supportive definitions are used for priority classification: An *a priori importance (api)* determines the global importance of a given point or object in a scene; and the *distance (dist)* determines the conceptual distance between any two points or objects. This conceptual distance describes the pathlength between the two respective end points.

Three different classes of priority designation are defined: supplied priorities, built-in priorities, and client priorities. A *supplied* priority results from either user interaction or a pre-defined priority algorithm. The *built-in* priorities represent on-demand calculations based on either:

- A priori importance, or
- Distance values.

The same distinction holds for *client* priorities. The notion of a *client* respects application dependent functions. That is, each possible client in the sense of this classification directly correlates to an application where the respective function is defined.

The notion of a *focal point* describes a point selected as current focus of interest. In case an emphasis technique works independently from any specific focus of interest, no focal point needs to be supported. In case focal points are used, the above mentioned priority is affected. For a *single* focal point, the priority of a point or object increases along with its a priori importance and decreases with its distance from the focal point. *Multiple* focal points extend the priority function by multiple variation degrees. Any precise instantiation of a priority function in this case is to be specified on a per-case basis.

Whether or not *animation* is used by an emphasis technique is subject of the fifth dimension of this classification. This use of animation is bound to variations in priorities. In case the priority values change, animation may help to eliminate or reduce abrupt transitions in presentation variables. The notion of animation used in this context represents a subset of the overall range of dynamic presentations as only positional changes (i. e. motion) are addressed.

The set of *inputs* making up the sixth dimension is ordered by increasing expressiveness. Each level of input represents possible data sources required by an empha-

sis technique, whereby each level is fully contained in the more expressive one. The most simplest case is a *sequence* representing one-dimensional data. *Hierarchies*, *flat graphs*, and *nested graphs* describe varying complexities of graphs as input. Finally, *beyond nested graphs* is simply a synonym for unrestricted input for an emphasis technique. It is to be noted that Noik's notion of an input source concentrates on graph-based emphasis techniques. This way, geometric input topologies are not addressed.

Given the classification presented in the table, any presentation technique may be characterised according to its emphasis dimensions. This directly implies that techniques are not placed directly into one of the dimensions. Instead, different grades of completeness are reached with regard to the respective property dimensions.

Preim and Ritter's Emphasis Classification

A different approach to classify emphasis presentations is presented by Preim and Ritter [2002]. Here, the presentation techniques act as keys to the classification. For each individual technique, its completeness with regard to a presentation property is given. These properties correlate to some degree with the dimensions presented above. However, Preim and Ritter specifically address the presentation context of medical visualisations. Table 3.6 lists this classification of emphasis techniques and their respective characteristics.

The parameter space for the classified techniques includes the applicability of any given technique. This may either affect a single or *multiple objects*. Based on this distinction, the geometric context of an object can be respected during its emphasis. Single objects can be targeted by any of the presented techniques. In case multiple objects can be affected by a technique, the relation between a specific object of interest and other objects may be illustrated. Techniques targeting only a single object allow to concentrate on representation of the object's state.

Whether or not an emphasis technique *ensures visibility* of a targeted object is addressed by another classification parameter. Depending on the illustration goal as expressed by the illustration target function, visibility of any specific object of interest might be a requirement. This parameter is associated with the previous one. In case a technique targets only a single object but cannot ensure this object's visibility, the resulting emphasis provides no visual effect.

The *grade of variation* describes the effect an emphasis technique has with regard to the overall scene. A *local* grade affects only the targeted object whereas a *global* grade results in a variation of the whole scene. An in-between grade of affecting not only the targeted object but its direct surrounding as well is regarded as *regional* grade of variation. This is to be distinguished from the former parameter addressing the level of emphasis targeting. Some techniques, such as transparency applied to all objects besides the illustration target, only address a single object but result in a global grade of variation.

Besides respecting the individual parameters for the listed techniques, the classification also includes an analysis of the respective implementation overhead. This varies considerably. Even though the technique-oriented parameters determine appropriateness of an emphasis technique, any real-world application might respect its cost of implementation as well. Depending on the application's context, some unclassified notes describing specific advantages or restrictions may deepen decisions made during illustration design.

Emphasis technique	Multiple objects	Visibility ensured	Grade of variation	Implementation overhead	Notes
1. Change of colour	yes	no	local	minimal	emphasis by increased saturation
2. Shadow Shadow volume	yes yes	rather not yes	local local	moderate high	
3. Contour lines	yes	no	local	high	demarcation to other objects
4. Transparency of hiding objects all other objects	cond. no	yes yes	regional global	moderate minimal	
5. Adjustment of view direction	no	yes	global	high	unnatural view in most cases
6. Cutaway view	yes	yes	regional	high	good for volume data
7. Bounding box	cond.	no	local	minimal	generally inappropriate as single technique
8. Crosshair	cond.	no	local	moderate	
9. Arrows	yes	no	local	high	
10. 3D fisheye	cond.	rather yes	regional	high	inappropriate for med. diagnosis

Table 3.6: Emphasis techniques and their characteristics according to Preim and Ritter [2002].

The following three subsections discuss selected specific classic illustration techniques in more detail. These techniques form the basis for some of the dynamic presentation techniques presented in the remaining sections of this chapter.

3.2.2 Change of Colour

The simple change of an object's colour provides an expressive tool for object emphasis. Preim and Ritter [2002] differentiate between two variants of emphasis by colour:

- Deployment of one colour that is globally used for the emphasis of all objects.
- Consideration of the current object's colour. Emphasis is gained by a modification of the objects previous colour. An exemplary option of doing this is by substituting a colour with low saturation by a colour of the same hue but with higher saturation.

A set of cognitive limitations is to be regarded in order to preserve expressiveness. An emphasis is ineffective in case the chosen colour occurs at surrounding objects as well. Furthermore, colour is mainly used for objects of at least moderate size or filling of long or strong lines. In case of small objects, the perception of differences in colour remains as a challenge as colour perception is affected by an object's surrounding [Foster, 2003, Wuerger et al., 2000].

Generally, the cultural effect of colour is to be considered when designing an illustration making use of it. Stankowski and Dushek [1994] present a colour classification with regard to its emotional effect. They point out that variations of the yellow-red area activate the emotional state positively whereas the green-blue-purple areas provide a negative effect. Specifically, dark purple is perceived as a representation of grief and yellow-black combinations as positive appeal. Making use of this cultural background for use of colour in an illustration environment allows to communicate an illustration message that helps to meet the illustration goal.

The value of blinking as a dynamic variation of colour changes is discussed in Section 2.4. Using a change in an object's emissive colour as a basis of dynamic rendering styles is addressed in Section 3.4.1.

3.2.3 Transparency

Transparency provides a means of ensuring visibility of an object of interest [Hamel et al., 1998]. It is an alleviated variation of simply disguising all objects in a scene that possibly hide the target object. In contrast to completely hiding objects, transparency allows to preserve scene-related context as all objects remain visible. Depending on the illustration target function, two different ways of applying transparency to an object may be used: (1) using transparency on objects hiding the object of interest or (2) making the whole scene transparent with exception of the object of interest.

The first case targets visibility of the opaque object of interest. It is emphasised by applying transparency to all objects that hide this object with regard to the current

camera position and viewing direction. This way, the formerly hidden object gets visible without a need to adjust the overall scene layout and camera position [Helbing et al., 1998]. Therefore, the grade of variation is of regional nature. For each affected object, a level of transparency needs to be determined. This level specifies how much the object's colour contributes to rendering the scene. It needs to be low enough to provide the desired see-through effect but simultaneously high enough to keep the object visible.

The second case of applying transparency to the whole scene with the exception of the object of interest addresses the problem in case (1) where too many objects hide the target. In this case, the see-through effect may not be achievable as the remaining colour values of all those objects accumulate and prohibit a visual effect of the final rendering of the hidden object. This is avoided by applying an exhaustive amount of transparency to the whole scene and just rendering the object of interest opaque. An example is shown in Figure 3.5.

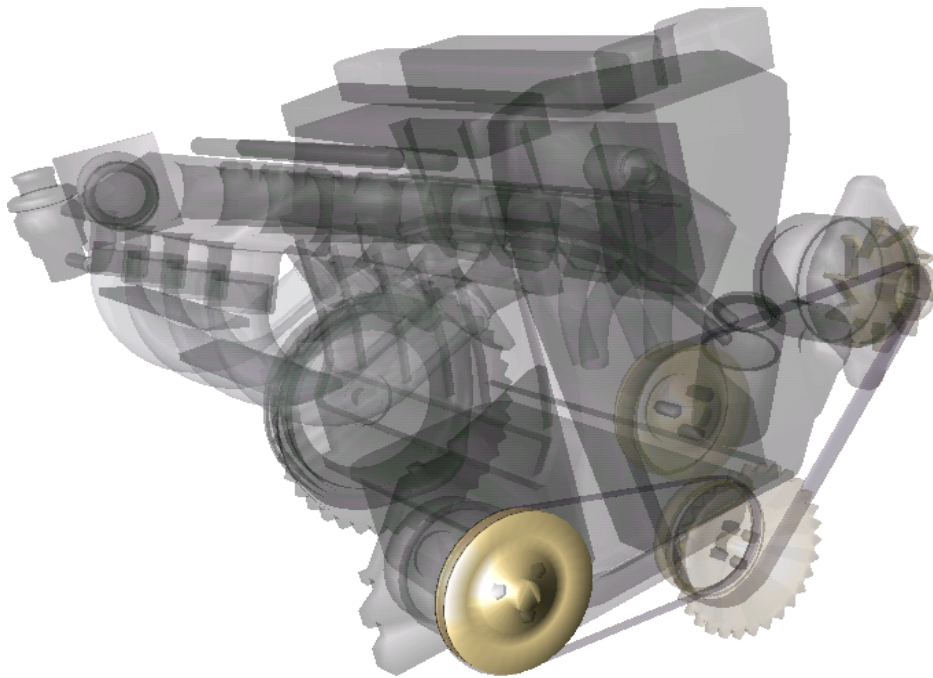


Figure 3.5: Use of transparency for the whole scene with the exception of an object of interest for the purpose of emphasising this object.

As a variation of transparency, wireframe rendering may be used. Overall, the achieved effect provides exclusive emphasis of the opaque object with regard to the remaining scene. Complete visibility is ensured at the cost of a global grade of variation.

An alternative to these two classical approaches for using transparency for emphasis is in providing an emphasis effect similar to a change in colour as discussed previously above. Thereby, the visual presentation of a single object is changed

with regard to its surrounding scene. Depending on the grade of transparency applied, the object is accentuated by a low amount of transparency or de-accentuated as it gets invisible due to complete transparency [Snowden and Verstraten, 1999]. This use of transparency is of local effect as only the presentation of the targeted object is modified. In contrast to both above approaches, visibility of the targeted object cannot automatically be ensured.

Transparency may also be used as a basis for dynamics of hybrid rendering styles. Section 3.4 discusses this in detail.

3.2.4 Fisheye

The notion of *fish-eye* distortion for presentation purposes is presented by Furnas [1986]. For the purpose of examining hierarchical graphs, a 2D view is created that allows for local browsing of subgraphs of interest. This is supported by defining a degree of interest (*doi*) function as well as an a priori importance (*api*) for all nodes in the graph. Using a distance function determining the pathlength between any two objects, the *doi* can be derived from the *api*. The local area of the object with the highest *doi* is then presented with some increased detail at the cost of presenting the remaining scene elements in a more dense fashion.

Fisheye distortions are applicable in a variety of areas. The span of applications includes data-driven illustration tasks as well as geometry-based illustrations. Examples of the former include *Tree-Maps* [Johnson and Shneiderman, 1991], hyper-text illustrations [Noik, 1993, Bartram et al., 1995, Yang et al., 2002], hierarchy presentations like *Cone Trees* [Robertson et al., 1991] or *hyperbolic layouts* [Lamping et al., 1995, Lamping and Rao, 1996, Munzner, 1997, Robinson, 1998]. Geometry-based fisheye illustrations include *pliable surfaces* [Carpendale et al., 1995], combinations of fisheye with text in 2D [Rauschenbach et al., 2001], and 3D [Preim et al., 1997]. A survey of illustration techniques including fisheye variations is presented by Herman et al. [2000].

The first extension of a fisheye zoom from 2D to 3D is presented by Mackinlay et al. [1991] with their *Perspective Wall*. Thereby, the third dimension is created by mapping the overall display onto a surface (the wall) that is folded at borders close to the area with highest *doi*. As the folded surface is of two-dimensional nature, the *Perspective Wall* can be respected as a $2\frac{1}{2}$ D presentation instead of true three-dimensionality. Another approach for using a fisheye zoom in 3D is illustrated by Carpendale et al. [1995]. Here, the information is mapped onto a stretchable surface that blobs out the area of interest while the remaining part of the scene is scaled down accordingly. The use of additional navigation hints, such as a cartographical grid, is provided by Carpendale et al. [1997]. Raab and Ruger [1996] present a 3D fisheye algorithm for coherent zooming. This algorithm allows to flexibly adjust the size of all objects affected by a zoom operation.

A fisheye view is usually a highly interactive emphasis technique [Olwal and Feiner, 2003]. While some part of the available information is presented in detail (the focus), the rest is still available in smaller size (the detail). This interaction is referred to as *focus and context browsing* [Leung and Apperley, 1994]. A variety of further interaction techniques exist that are build on top of fisheye zooms. These include *Magic Lenses* by Bier et al. [1993] for interactive filtering of information, *Macroscope* by Lieberman [1994] which combines multiple translucent layers for the purpose of providing extended pan and zooming interaction.

Section 3.5 presents two dynamic variations of a fisheye zoom that concentrate on preserving structural information of the zoomed objects. This information is presented by means of navigational hints which are build dynamically on the basis of the zoomed object.

3.2.5 Discussion

Two classification systems for emphasis techniques are presented in this section. Both of these systems mainly concentrate on static presentation techniques. Still, they provide different approaches for classifying presentation techniques. One characterises the respective techniques according to their property dimensions. The other uses the techniques instead of their properties as key to the classification. These orthogonal approaches point out the flexibility in respecting presentation techniques.

Overall, the presented analysis of the two classification systems show the variety of classic and static presentation techniques. Dynamic techniques as presented in the following are not isolated from these techniques but are based on static ones and extend them in various ways. The following sections present such extensions.

3.3 Motion Techniques

This section is the first out of three presenting dynamic presentation techniques. Based on the fundamentals of dynamics as discussed in Chapter 2, the set of presentation variables for an illustration system can be enhanced using dynamic techniques. Using temporal constraints as presented in Chapter 4 helps to make use of dynamics up to their full expression potential. It is specifically to be noted that this does not aim at an exclusive use of dynamics for any presentation. Instead, dynamics *extend* the set of presentation capabilities and are to be used in combination with any classic and static illustration techniques.

For the discussion of not only motion but all dynamic presentation techniques, the framework presented in Section 3.1.2 is used. That is, the illustration model is presented as an information set \mathbf{IM} with a set of attributes \mathbf{AM} for all information objects $IO_i \in \mathbf{IM}$. For the presentation of these objects a set of presentation

techniques \mathcal{D}_p is used. This set denotes all available techniques. As introduced in Section 3.1.4, the set of dynamic presentation techniques is denoted as \mathcal{D}_d . Again, for reasons of simplicity of the discussion, this will be referred to as \mathcal{D} .

The individual techniques of set \mathcal{D} are labelled $d(IO)$. That is, each $d \in \mathcal{D}$ may be used to present a specific information object. This relation is not mutually exclusive. Therefore, for any two objects IO_i and IO_j with $i \neq j$, the same presentation technique may be used: $d_k(IO_i) = d_k(IO_j)$. Any concrete parameterisation and presentation of d depends on the dynamics class.

The first class introduced is motion. That is, a set $\mathcal{D}_M \subseteq \mathcal{D}$ is defined with $\mathcal{D}_M = \{d_1, \dots, d_n\}$ ($n \in \mathbb{N}$) that provides presentation techniques with moving effects for information objects IO . Further classes of dynamics include changing rendering styles and distortions as presented in upcoming sections.

Three different subclasses of motion techniques are presented here: oscillations, structural changes, and a motion-enhanced information mural. These techniques do not introduce any new motion algorithms. Instead, they make use of existing approaches and introduce an entry point for dynamic presentation techniques. Specific applications making use of motion are not included in this section but are part of Section 6.2 in the application chapter.

3.3.1 Oscillations

Repeating motion patterns provide an expressive means of communicating information about groups of objects [Johansson, 1964]. This subsection introduces oscillations as motion in the framework of dynamic presentation techniques. First of all, construction of information objects is discussed as underlying prerequisite. This is followed by a description of translations and rotations as basis techniques of oscillations. Ways of specifying an oscillation's parameter space completes this part.

Construction of Information Objects

According to the presentation framework, information objects IO are described by a set of n attributes. These have a continuous range of values. In an n -dimensional information space, each IO is described by $\mathbf{AM} = \{A_1, \dots, A_n\} \in \mathbf{IR}$. Individual attributes A_i with $1 \leq i \leq n$ specify the role of the respective information object IO in its information space \mathbf{IR} . This concept is generalised such that concrete positioning information of IO is not required for applying oscillation.

This way, oscillation can be applied to an information set without depending on specifics on the origin of information objects. Examples of such origins include the output of a mapping pipeline as it is used in information visualisation systems or the manual design and construction of a geometric model that is subject of an illustration. A concrete example of the former case is presented in the context of

information fusion in Section 6.2.3, an example of the latter case is shown with the illustration of a generator model in Section 6.3.3.

Basis Techniques

Two types of basis techniques for oscillations may be used:

1. *Translation*: The position of an object *IO* is constantly changed. In case translation modification affects multiple scene objects, their influence on scene coherence is regarded as a group. That possibly allows for these objects to change their spatial structure without influencing other (possibly unrelated) parts of the display.
2. *Rotation*: Hereby, an information object *IO* is rotated constantly. The rotation centre is thereby part of the parameter space. An advantage of using the object's central point is that the scene's coherence is not considerably invalidated. However, spotting this motion might require an intense cognitive approach by the user. To avoid this, the rotation centre might vary from the object's centre. It does not necessarily need to be a point inside the volume of the object. This flexibility is of course paid for by the potential of invalidating the scene coherence.

Both techniques can be combined to create sophisticated motion patterns. In addition, multiple methods can be used simultaneously in a scene to express different data characteristics by means of different motion application.

An oscillation is defined by repeating a motion continuously. For translation-based motions, three variations of translation paths are defined:

1. a straight path between two fixed positions,
2. a fixed path passing key positions, and
3. a free path restricted by an area.

These paths are illustrated in Figure 3.6. Depending on their respective path lengths, the three different motion trajectories vary in influence on scene coherence. The first two cases possibly cause local, regional, and global effects on the scene. Their positional influence is not restricted. This is different for the third case of paths that are restricted to an area. This area naturally limits the oscillation's influence to local neighbourhood and the region spanned by the area. It is possible that the area extends over the complete scene borders. In this case influence of this path would be of global nature. However, this specific case is regarded as regional influence as well, because the illustration designer specifically modelled the region as spanning the whole scene. This way, the notion of a region applies by chance to a larger area as usual.

Two distinct cases exist for rotation-based oscillations:

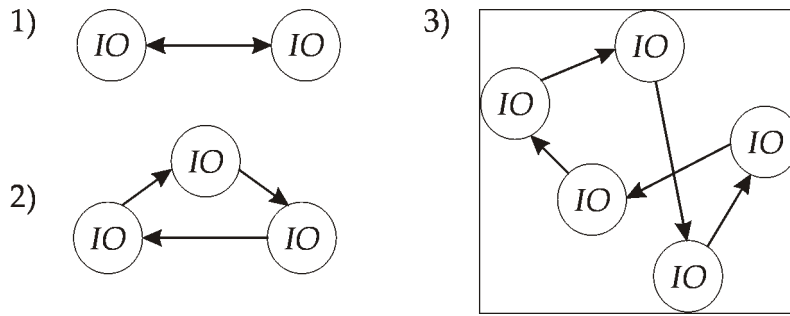


Figure 3.6: Three ways of defining translation paths for oscillating motion trajectories: 1) moves an object along a straight path between two fixed positions, 2) moves an object along a fixed path passing key positions, and 3) constraints an otherwise free path by a region.

1. regular shape with midpoint rotation and
2. irregular shape with rotation around arbitrary point.

The first case preserves scene consistency and has only local influence. Case two possibly affects remaining parts of the scene as well. Figure 3.7 represents these two cases. In case the rotation centre is located at the midpoint of an information object with regular shape, its local region is affected. In case the object is either of irregular shape or the rotation centre is chosen arbitrarily, global effects on the scene coherence may result.

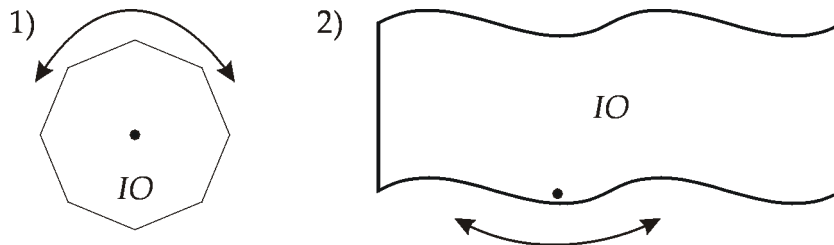


Figure 3.7: Effects of rotation-based oscillations. The dots mark the respective rotation centres.

Parameter Space for Motion Mapping

For the purpose of mapping oscillations onto information objects in an illustration, these objects are grouped into information structures of similar objects: $\mathbf{IS}_M = \{IO_i \mid IO_i \in \mathbf{IM}, i \in \mathbb{N}\}$. Following the algorithmic preprocessing description of the framework as discussed by Kreuzeler and Schumann [2002], a measuring function $s(IO_i, IO_j) = s_{ij}$ is used to decide whether or not two information objects IO belong to the same group. As already demonstrated by Ankerst et al. [1998], this similarity measurement is a domain-specific task. Depending on context and user scope defined by the illustration target function, the construction of \mathbf{IS}_M is achieved

independently for each illustration scenario. An example will be presented in Section 6.2.3 for the context of representing groups of information objects retrieved out of data from the field of bioinformatics.

The parameter space for presentation techniques $d \in \mathcal{D}_M$ is determined by the grouping function s . For each group of translations, rotations, or combinations thereof, a set of parameters may be modified:

- frequency of the oscillation,
- amplitude of the basis technique, and
- motion progression function describing the trajectory appropriately depending on a concrete application context at hand.

The evaluation of these parameters is handled by an oscillation function $f_{OS}(\mathbf{IS}_M, s, \mathcal{D}_{OS}, \delta)$ with:

- \mathbf{IS}_M specifying the information structure containing all information objects affected by oscillating motion.
- s being the measuring function determining the groups of information objects in \mathbf{IS}_M .
- \mathcal{D}_{OS} providing the set of available oscillating motion techniques. This set consists of the basis techniques and varying combinations thereof.
- δ being the temporal parameterisation function presented in the following chapter.

The oscillation function maps oscillation techniques $d_M \in \mathcal{D}_{OS}$ onto groups of information objects $\mathbf{IS} \in \mathbf{IS}_M$ that correlate with each other according to the measuring function s . The resulting dynamic presentation is temporally constrained by δ for the purpose of complying to cognitive and temporal restrictions as outlined in Chapters 2 and 4.

3.3.2 Structural Changes

A classical approach to motion by structural changes is flow representation. Such flows either affect particle systems as introduced by Reeves [1983] or are represented as flows on a surface. Particle systems basically describe object-based motion. As discussed for the case of oscillations above, they are not subject of this section.

Early works on flow representations by structural changes of surfaces are drawn by Max [1981] at the example of ocean waves. This work was picked up and refined continuously [Fournier and Reeves, 1986, Peachey, 1986, Tso and Barsky, 1987]. These approaches target structural change of a surface such as a wave. Others address flow of patterns on a surface [Turk, 1991, Stam, 2003] or dynamic smoothing out of surfaces [Desbrun et al., 1999].

Procedural models may be used for the description of structural changes of objects. Continuing the work on ocean wave representations, such a model is presented by Hinsinger et al. [2002]. Jeschke et al. [2003] extend this approach by allowing to model breaking of such waves. Functions are used in this model for describing the movement and appearance of the waves.

In contrast to oscillations as discussed above, the approach of functionally describing a motion of structural changes is continued here. Two examples are presented for this purpose: a folding function as well as the description of a free-form function with user-specified constraints of a structural change. Before the discussion of both techniques, some underlying foundation regarding the organisation of information objects is presented.

Organisation of Information Objects

The motion by structural changes as presented here targets object surfaces. Changes of this surface are applied on a per-face basis. That is, a specific motion technique only affects a single surface of the respective information object. Thereby, local influence of the motion technique is ensured by not only touching no other information object but also limiting the changes to parts of the targeted information object.

The overall set of presentations using dynamic structural changes is given as $\mathcal{D}_{ST}(\mathbf{IM})$. This set is a union of sets of structural change-based presentations $\mathcal{D}_{ST}(IO)$ for the single information objects making up the information set:

$$\mathcal{D}_{ST}(\mathbf{IM}) = \mathcal{D}_{ST}(IO_1) \cup \mathcal{D}_{ST}(IO_2) \cup \dots \cup \mathcal{D}_{ST}(IO_n) \quad IO_i \in \mathbf{IM}, 1 \leq i \leq n, n \in \mathbb{N}.$$

These individual sets are collections of motions applied to surface-based subparts of the respective information object:

$$\mathcal{D}_{ST}(IO) = \{d_{ST_1}(IO), d_{ST_2}(IO), \dots, d_{ST_m}(IO)\} \quad m \in \mathbb{N}. \quad (3.2)$$

Here, m describes the number of presentation techniques resulting in structural changes of surfaces of this information object. This number is not fixed but depends on the object's number of faces s :

$$IO = \{fc_1, fc_2, \dots, fc_s\} \quad s \in \mathbb{N}, s \leq m.$$

Each face might be subject of an individual local motion. This way, $m = s$ holds. That is, the number of structural motion techniques for an object potentially equals

the number of faces of the object. However, sets of faces may be grouped for the purpose of applying such motion to the group instead of just single faces. These groups are denoted as FC consisting of faces fc :

$$FC \subseteq IO = \{fc_1, fc_2, \dots, fc_g\} \quad g \in \mathbb{N}, g \leq s.$$

Therefore, the definition of the available set of motion presentations for an information object as expressed in equation (3.2) changes to:

$$\mathcal{D}_{ST}(IO) = \mathcal{D}_{ST}(FC) = \{d_{ST_1}(fc_i), d_{ST_2}(fc_i), \dots, d_{ST_m}(fc_i)\} \quad fc_i \in FC, 1 \leq i \leq g.$$

In case a face set group consists of a single face only, it's presentation is denoted as:

$$d_{ST}(fc) \Leftrightarrow d_{ST}(FC) \text{ holds and } FC = \{fc\}.$$

This extends the overall number of possible motions applicable for the information object. Naïvely, this results in $m = s!$, that is, the number of applicable motion patterns to an information object spans all possible permutations of combining the object's faces into sets. This number is limited to some degree. First of all, double pairs of faces are regarded equally. Furthermore, visual cluttering is to be avoided by not applying a technique d_{ST_i} to a face fc_i as well as to a group of faces including fc_i :

$$d_{ST_i}(fc_a) \Leftrightarrow \nexists d_{ST_i}(fc_b) \text{ with } \{fc_a, fc_b\} \in FC \text{ and } fc_a = fc_b.$$

Figure 3.8 illustrates this grouping of object faces. The leftmost image shows the set of sixteen faces defining a part of an information object. A subgroup of this face set is highlighted in the middle image of the figure. This subgroup may be used for applying motion based on structural changes of the individual faces in the group. Using such motion on two disjunct groups of the overall face set is sketched out in the rightmost image. Either the same motion pattern or two patterns different from each other may be used on both groups. In case the same pattern is used, some repeating attributes of the information object can be reflected. Using different techniques outlines varying structures of these attributes.

Principle Motion Basis

As stated above, motion by structural change refers to modification of surfaces. These modifications are constructed by altering vertex positions. Description of this vertex altering is done by a function $f_v(v)$. This function is a tuple $(sr, func, amp, freq, \mathcal{P})$ with:

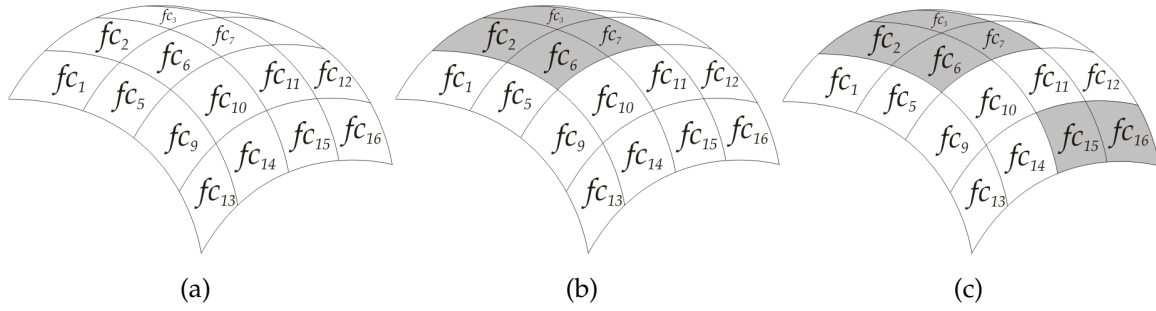


Figure 3.8: Sketch of a part of an information object and a set of faces defining this part's surface. Image (a) shows the raw set of faces. A concrete group of faces which may be subject to motion by structural change is emphasised in (b). To the right, (c) shows the definition of two face sets which are defined independently from each other.

- sr describing the rate at which the function is sampled,
- $func$ being the motion function which describes the motion trajectory of the affected point,
- amp being the maximum amplitude constraint valid for all sampled function values,
- $freq$ being the frequency constraint, specifying how often the function is iterated per second,
- \mathcal{P} holding a variable set of parameters depending on a concrete function definition.

Figure 3.9 points out the principle of this function. The figure shows the surface defined as a group of faces $FC = \{fc_1, fc_2, \dots, fc_{16}\}$ as introduced in Figure 3.8. This group is modelled by a set of 25 vertices v_i ($1 \leq i \leq 25$). Now, a subgroup of faces is defined to which a motion by structural change is to be applied:

$$FC_{ST} \subset FC = \{fc_2, fc_3, fc_6, fc_7\}.$$

This equals the scenario discussed for Figure 3.8(b) above. This set of faces is now modified by use of $f_v(v_8)$. Modification of the point's position results in changes of all faces edged by the point. The concrete positional function values of $f_v(v_8)$ are determined by the motion function $func$ and constrained by amp and $freq$.

For the purpose of smooth face modifications, an application using this function $f_v(v)$ should not only respect alteration of a single vertex at a time. Instead, the respective functions $f_v(v)$ for the surrounding vertices are to be synchronised. In the specific example presented here, these are $v_2 \dots v_5$, v_7 , v_9 , and $v_{12} \dots v_{14}$. Synchronisation of this kind is not inherently ensured by each $f_v(v)$ automatically. This is due to limiting the function's influence to one vertex at a time and to provide for a maximum of flexibility in using it.

The sample rate sr has no direct effect on the motion pattern. Instead it influences presentation performance and is to be used as a knob for controlling the motion

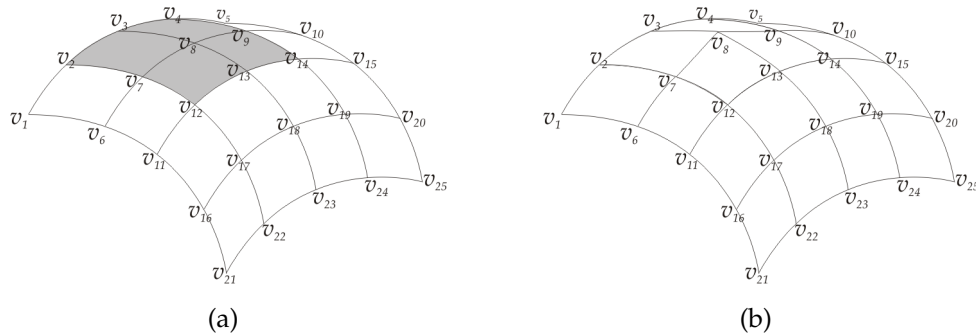


Figure 3.9: Principle motion basis for motion by structural change targeting surfaces. To the left, (a) shows the face set defining the surface. The faces subject to motion are highlighted. In (b), these faces are changed in their structure by altering vertex v_8 .

externally. This proves useful for integrating motion by structural changes with the temporal presentation framework as presented in the following chapter.

The following two subsections present exemplary motion techniques based on structural changes. First, a motion based on folding is described. Secondly, a notion of describing the change by a free-form function is introduced. This allows for a flexible description of motion depending on the respective field of application. Both concrete examples show that the above definition of the function forms as a basis for motion by structural change. This basis provides enough flexibility to construct different motion techniques.

Structural Change by Folding

This section introduces motion based on structural change of surfaces by folding. Thereby, each $d_{ST}(fc)$ is defined by modifying fc such that an effect of folding this face is gained. This effect is specifically of local influence and applied to each face separately.

Figure 3.10 shows a sketch that outlines the folding. It is a two-dimensional representation defined by a cut through the z axis parallel to the (x, y) -plane. This exemplary cut is made for simplicity reasons. The folding is not restricted to take place in any axis-parallelism. Instead it is based on the notion of a *fold plane*. This plane is determined by any three neighbouring vertices of the current vertex of interest. In this case, this is just the (x, y) -plane.

A folding is defined by a tuple of five parameters:

$$fold = \{amp_{min}, amp_{max}, loop_{min}, loop_{max}, lap_{fold}\}.$$

The description depends on function $f_v(v)$ as it defines a set of two amplitudes instead of just one and uses the notion of *loops* in contrast to frequency.

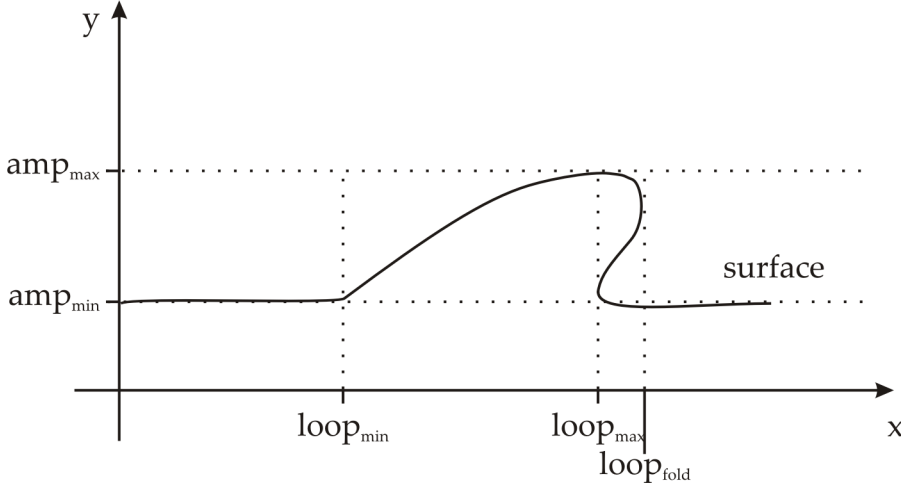


Figure 3.10: 2D cut view of folding a surface.

The amplitude range of the fold is given as $amp_{max} - amp_{min}$. In between, the current amplitude is determined at each step of the folding process:

$$amp_{cur} = \int_{i=0}^{i \leq \pi} |\sin(i)| \cdot stretch \quad (3.3)$$

Thereby, *stretch* is defined as a constant stretch factor that ensures restriction of the amplitude to its range of $amp_{max} - amp_{min}$. Using only the absolute value of the *sin* at each step i of iteration ensures to avoid a *negative folding* effect.

Similarly to the amplitude, the length of the fold equals the distance of $loop_{max} - loop_{min}$. An amount of overlapping of a fold is labelled as lap_{fold} . To be precise, this describes the maximum expansion of the lap whereas the *amount of overlapping* equals $lap_{fold} - loop_{max}$. The folding may be applied repetitively to the face. Depending on the distance between two folds, both may overlap. If these folds are given as $fold_a$ and $fold_b$, this implies:

$$loop_{max_a} \leq loop_{min_b} \leq lap_{fold_a}.$$

The current loop value in range $[loop_{min}, lap_{loop}]$ is determined at each step i by:

$$\begin{aligned} loop_{cur} &= loop_{min} + i \cdot f_{lap}(i) \\ f_{lap}(i) &= \begin{cases} \cos(i) \cdot i & \Leftrightarrow \pi/2 < i \leq \pi \\ i & \text{otherwise} \end{cases} \end{aligned} \quad (3.4)$$

The lap function $f_{lap}(i)$ helps to achieve the horizontal fold effect as it is sketched in Figure 3.10. Overall, iterations of the fold function $f_v(v)$ are normalised to the range $[0, \pi]$. This is only done to achieve the fold effect and does not restrict application of the folding spatially.

The single steps to be carried out for achieving a motion by structural change based on folding are sketched out in Algorithm 3.1. The end of the presentation is marked temporally in line 7. All other steps of the algorithm follow the above discussion.

Algorithm 3.1 Motion by structural change based on folding.

Require: \mathbf{IS}_{ST}

- 1: Determine current vertex of interest $v_i \in \mathbf{IS}_{ST}$
 - 2: Determine *fold plane* for v_i
 - 3: **repeat**
 - 4: Determine amp_{cur} based on Equation 3.3
 - 5: Determine position $loop_{cur}$ based on Equation 3.4
 - 6: Render image
 - 7: **until** temporal end of presentation
-

The resulting effect of this motion is shown in Figure 3.11 at the example of a flat plane. From left to right, this plane is shown in its original planar state and with increasing amplitude. The folding is applied starting at the plane's upper right hand corner.

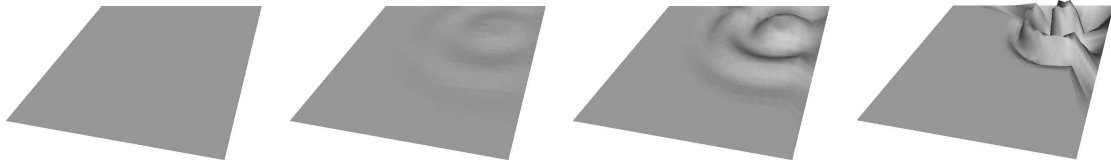


Figure 3.11: Exemplary use of a folding motion.

Free-Form Function

Creating a motion by structural changes of surfaces as outlined at the example of folding above can be generalised to some degree. This is done by using a free-form function to describe the changes in structure of a face. Depending on concrete application context, this function is provided by modelling it as part of the illustration target function or by specifying it interactively. The free-form modelling of structural changes of object surfaces is supported by a set of function primitives. This set is not limited by nature but defined by a concrete implementation of this motion technique. Section 5.2 presents such an exemplary implementation which includes basic trigonometric functions.

Trigonometry is used to construct an exemplary free-form function. This function defines waves on a surface. Such a wave is specified in the illustration model which is part of the illustration target function. For the example shown here, this

specification uses a simplified version of the folding as described above. If i describes the current loop position, amp_{cur} the current amplitude value of the wave, and $stretch$ a user-defined stretch factor for the motion, the free-form function values are given as:

$$\begin{aligned} func &= i \\ amp_{cur} &= \int_{i=0}^{i \leq \pi} \sin(i - ((amp_{max} - amp_{min}) \cdot stretch)). \end{aligned}$$

The resulting effects of this function definition are shown in Figures 3.12 and 3.13. The first set of images applies waves to the complete set of surfaces of a geometrical model consisting of three information objects. Each object represents a letter. The objects are modelled by a set of relatively large faces. From left to right, repetitive waves are applied to an increasing set of surfaces. This helps to gradually illustrate the organisational structure of the model. In order to make use of this illustration in a context which is not of object-intrinsic nature, this motion-enhanced information set may be placed in a motion-less context. There, the folding helps to outline specific parts of the overall scene consisting of a more complex information set.

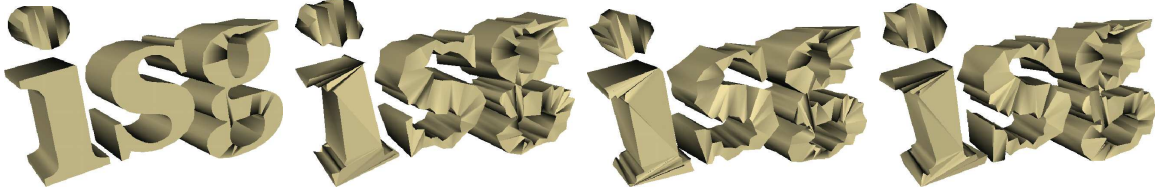


Figure 3.12: Snapshots expressing the effect of a local motion by structural change applied to the complete set surfaces in the model.

This is done in the example shown by the second set of images in Figure 3.13. Here, the overall information set consists of a geometric model of an eye. This set is composed of a variety of information objects defining the individual parts of this eye:

$$\mathbf{IM} = \{IO_i\} \quad 1 \leq i \leq |\mathbf{IM}|, IO_i = \text{part of the eye.}$$

From this set, an information structure \mathbf{IS}_{ST} is derived. The elements of this structure are subject to a folding motion. Specifically, this structure is defined as:

$$\mathbf{IS}_{ST} = \{IO_j\} \quad 1 \leq j \leq i, IO_j = \text{external eye muscle.}$$

That is, the waving motion is used to illustrate the external eye muscles in contrast to the remaining model. This adds to the colour coding of the model presentation. Colour is used to emphasise an information structure containing all muscles of the eye, including the external ones. If \mathbf{IS}_{col} labels this structure, $\mathbf{IS}_{ST} \subseteq \mathbf{IS}_{col}$ holds. Therefore, using motion helps in pointing out the location differences of external muscles with respect to the set of all eye muscles.

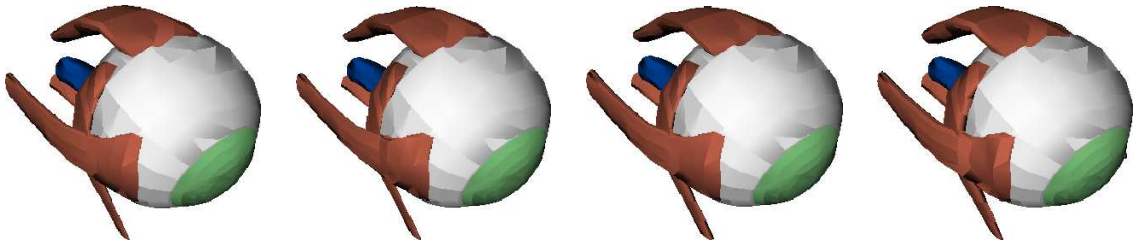


Figure 3.13: Snapshots expressing the effect of a local motion by structural change applied to only parts of the model. The images show different status of folding the eye movement muscles to the left and at the eye's top. This allows to disambiguate them from the eye's focal length muscles which are coloured the same and located nearby.

Effects on Scene Coherence

In contrast to object-based oscillations, motion by structural changes provides local effects to a scene only. If \mathbf{IS}_{ST} is an information structure containing all objects which are subject to structural motion, no object $IO \in \mathbf{IM} \setminus \mathbf{IS}_{ST}$ exists that is influenced by structural motion as well. This locality of motion influence insures that scene coherence is not invalidated.

3.3.3 Motion-enhanced Information Mural

This subsection introduces the use of motion for the purpose of enhancing the *information mural* as it is presented by Jerding and Stasko [1998]. The information mural is an illustration technique that provides a representation of an entire information space and fits this representation onto an available display window. This is achieved by providing a miniature version of the original information space. Creation of this version is supported by use of visual mapping of input data onto small information objects, variations in size and colour as well as intensity specifications of these objects. As the whole information space is presented at once, the information mural allows to browse the information space and to focus on items appearing as interesting.

The original mural by Jerding and Stasko [1998] is a two-dimensional illustration technique. All information objects are constructed by mapping input data onto coloured pixels forming lines in the mural. For the purpose of a motion-enhanced mural presented here, this presentation is extended to 3D. This way, the mural lines are modified to form cuboids. These cuboids allow to use their depth value as a further expression dimension. This helps to loosen the spatial presentation limits to some degree. Furthermore, using a three-dimensional presentation helps to explore the information space interactively by providing another degree of freedom for possible viewing directions. Precisely, instead of just providing a frontal view of the mural as in the original algorithm, it may be rotated around the different axes for detailed investigation.

Motion is used to enhance the mural and provide some means for presenting structural information of input data distribution. Using threshold values for the different dimensions of input data represented in the mural, a set of motion techniques is parameterised accordingly. This provides a practical illustration algorithm combining oscillations and motion by structural change. The presentation of a motion-enhanced information mural is divided into three parts:

1. construction of the information objects defining the mural;
2. ways of mapping motion onto these objects;
3. a practical step-by-step algorithm for constructing illustrations using the motion-enhanced mural.

Construction of Information Objects

As an information mural's main goal is to display the complete information set in a display, all $IO_i \in \mathbf{IM}$ are created initially. An update of the input data set necessarily results in an update of \mathbf{IM} . The same holds in case of streaming where the amount of available input data steadily increases. Any changes resulting from user interaction are expressed by a manipulation of the attribute set \mathbf{AM} and information structures \mathbf{IS} .

The creation of IO_i correlates to the available input data sets. The information mural technique is based on the idea of mapping an input data image onto a representation image. Thereby, the notation of an *image* does not necessarily describe the output of a rendering task but the collection of $k \times l$ data points, where $k, l \in \mathbb{N}$. The input image is an information space with dimensions $n \times m$. This is to be mapped onto the representation image with resolution $i \times j$ with $i, j, n, m \in \mathbb{N}$ and $i \leq n, j \leq m$. As a result, an information set is constructed consisting of $i \times j$ information objects: $\mathbf{IM} = \{IO_{11}, IO_{12}, \dots, IO_{ij}\}$. Construction of the individual $IO_i \in \mathbf{IM}$ follows the original mural algorithm as outlined by Jerding and Stasko [1998]. Overall, this creates a 2D image representing the original information space. The third dimension is now used to convey an additional attribute of the input data. The values of this attribute dimension are simply mapped onto the depth of their respective information objects.

The geometric shape of an information object is a vertical line in 2D respectively a cuboid in 3D. This allows to order all IO_i along the x axis. Other object characteristics as well as the mural restrictions are ensured by assigning appropriate attributes $A_i \in \mathbf{AM}$. These attributes include position, shape modification, and colour. The x position is determined by appropriately moving along the axis according to the overall coordinate maximum of the display. Figure 3.14 shows the principle mapping of the mural as well as the layout of all information objects. Mapping of a concrete example of the mural is presented in Chapter 6.

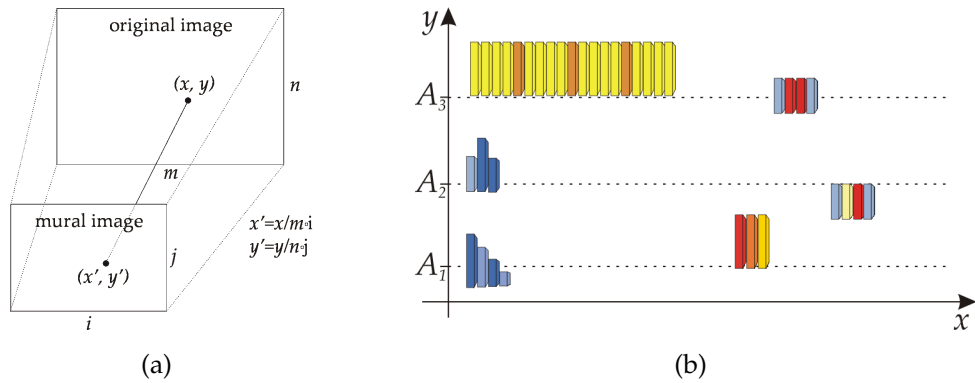


Figure 3.14: Principle mural layout: Subfigure (a) shows mapping of an $n \times m$ input image onto its $i \times j$ representation [Jerding and Stasko, 1998]. In (b), a sketch of the 3D layout is shown where data is mapped onto cuboids and all three spatial dimensions are used to express attributes of the information space. (based on Pehrs [2004])

Main Interest Scalar

The notation of a *main interest scalar* is introduced for the case of high-dimensional input data exceeding the capacity of direct mapping onto the 3D information mural. This scalar represents a specific attribute dimension of the original information space. Assumed, the attribute set of this space consists of k dimensions and the main interest scalar is an attribute A_s , a set of $s \times (k - 1)$ virtual input images is available. Thereby, $r = k - 1$ represents respective data dimensions that vary while s changes. As a result, an illustration is constructed that outlines variations of r input dimensions with respect to the current interest scalar dimension. In order to illustrate all such variation combinations, multiple presentations need to be constructed with a varying main interest scalar s .

Motion Mapping

In order to extract patterns from data distribution, a set of information structures $\{\mathbf{IS}_{M_1}, \dots, \mathbf{IS}_{M_n}\}$ is composed that merges similar objects into groups. The overall number n of these structures is less than or equal to the number of information objects in the information set \mathbf{IM} . The metrics used to obtain similarities and structures are parameterisable interactively and depend on the specific application domain at hand. Exploration at arbitrary levels of detail are thereby possible. The user is actively involved in supervising and steering the search for patterns.

Parameters controllable by the user are referred to as *threshold dimensions*. Elements of these dimensions directly represent an emphasis limit of respective input dimensions. The threshold dimension set is given as:

$$\mathbf{TD} = \{th_1, th_2, \dots, th_n\} \quad n \in \mathbb{N}, n \leq \dim(\mathbf{IR}), n \leq \dim(\mathbf{IM}).$$

Thereby, n does neither exceed the overall number of available attributes in \mathbf{AM} that is specified as $\dim(\mathbf{IR})$ nor the number of information objects defined in the information space \mathbf{IR} ($\dim(\mathbf{IM})$). The individual threshold dimensions represent values of attributes $A \in \mathbf{AM}$ that are used for determining whether or not a respectively affected information object belongs to an information structure \mathbf{IS}_M :

$$\mathbf{IS}_M = \{IO : \Delta_{th}(IO, th_i) > 0\}.$$

This definition makes use of a threshold function Δ_{th} determining the distance between a threshold and the respective attribute value of an information object IO . Using the attribute function $attr(IO, A_i)$, which returns the value of an attribute A_i of a given information object, the distance Δ_{th} is defined as:

$$\Delta_{th}(IO, th) = attr(IO, A_i) - th_i \quad 1 \leq i \leq n.$$

The index of attribute A_i and the threshold th_i are the same as both specify the same dimension of the original information space. Precisely, only values of the same attribute dimensions in the input data can be compared with each other. It is specifically to be noted that a threshold defined as minimum function reverses this definition. That is, in this case all information objects with an attribute value *smaller* than the threshold are included in \mathbf{IS}_M .

All information objects $IO_i \in \mathbf{IS}_M$ are subject to a presentation with a motion technique $d_i \in \mathcal{D}_M$ whereby:

$$\mathcal{D}_M = \mathcal{D}_{OS} \cup \mathcal{D}_{ST}.$$

That is, the set of available motion techniques for enhancing an information mural results as a union of oscillation techniques and structural changes. Parameterisation of individual motion techniques is influenced by a *motion constraint factor* (mcf):

$$mcf = \frac{\Delta_{th}}{max - min}.$$

The *max* and *min* values correspond to the limits of the overall value range of all attributes in \mathbf{IR} . Therefore, the mcf normalises the dynamic parameters of all available motion techniques. As the mcf depends on the individual thresholds, a set of such factors exists. Each mcf corresponds to a specific threshold used for grouping information objects. This way, distinct motion parameterisations are derived that allow different presentations of different groups and patterns in the underlying data. Similar to the motion techniques discussed in the previous sections, a *mural motion function* $f_{MM}(\mathbf{IS}_M, \mathbf{TD}, \mathcal{D}_M, mcf, \delta)$ is defined employing the following components:

- IS_M** specifying the information structure containing all information objects affected by motion.
- TD** specifying the set of threshold dimensions, whereby each $th_i \in \mathbf{TD}$ acts as key into the respective groups of information objects in **IS_M**.
- D_M** providing the set of available motion techniques.
- mcf* being the distance-based motion constraint factor for parameterisation of $d_m \in \mathcal{D}_M$.
- δ being the temporal parameterisation function from Chapter 4.

This function applies motion techniques to a subset of all available information objects in an information mural. Objects included within this mural are specified by structure **IS_M**. The thresholds of set **TD** are used to differentiate classes in this structure. Different classes may be illustrated by different motion techniques $d_m \in \mathcal{D}_M$. Furthermore, the individual techniques are to be parameterised differently. This parameterisation is based on *mcf*. Last, but not least, presentation parameterisation as provided by δ is also supported by *mcf* which is specific to motion in an information mural.

Practical Use

For the purpose of describing the practical use of the motion-enhanced information mural, a step-by-step algorithm for doing so is presented here. This approach illustrates the principle use of the mural for dynamic presentation. Section 6.2.2 shows a concrete example of using this mural technique for illustration of a climate data set.

Algorithm 3.2 describes the general approach for using the motion-enhanced information mural for illustrating a high-dimensional information space. The overall goal of this illustration as much as any other is in fulfilling the illustration target function. Thus, this illustration is an interactive process and all steps with the exception of 3, 7, and 8 require input from the user.

3.4 Dynamic Changes of Rendering Styles

In order to achieve rendering style dynamics, hybrid presentations are required. These hybrids combine use of different styles in a scene simultaneously. Dynamics are created by blending between styles.

At first, an overview of hybrid presentations is given. An introduction to style blending follows. Two concrete approaches of such blending are discussed and

Algorithm 3.2 Practical use of the motion-enhanced information mural.

Require: Modelling of the illustration target function itf .

```

1: while user scope of  $\text{itf}$  not fulfilled do
2:   for all  $A_s \in \text{AM}$  do
3:     Construction of mural presentation.
4:     Selection of threshold dimensions TD.
5:     while Scope of  $\text{itf}$  with regard to  $A_s$  not fulfilled do
6:       Select  $th_i \in \text{TD}$ .
7:       Designate parameterisation of  $d_i \in \mathcal{D}_M$  by means of  $f_{MM}$ .
8:       Illustration presentation.
9:       Interactive exploration with respect to  $\text{itf}$ .
10:    end while
11:  end for
12: end while

```

round up this section. The work presented here results out of a cooperation with Bernd Nettelbeck and Tobias Isenberg [Nettelbeck, 2003, Jesse and Isenberg, 2003].

3.4.1 Hybrid Presentations

In recent years, non-photorealistic rendering (NPR) has received a great deal of attention. Two books have been published which give a comprehensive overview about the subject [Gooch and Gooch, 2001, Strothotte and Schlechtweg, 2002]. However, the group of non-photorealistic rendering styles are typically considered to be separated from photorealistic styles. Only little attempt has been made to combine the two into individual renditions in a systematic way.

Saito and Takahashi [1990] were the first to add non-photorealistic elements such as silhouettes to photorealistically shaded objects in order to create more expressive renditions. Gooch et al. [1998] combine an adapted photorealistic shading with non-photorealistic elements such as silhouettes to create a non-photorealistic lighting model. Their main application domain is automatic technical illustration. Gooch et al. [1999] explore techniques to speed up the computation of the lighting model in order to allow for interactive technical illustrations. Although both create a somewhat hybrid style, they apply it coherently to the whole scene. I. e., the rendering itself only uses one style. It is, thus, not hybrid in the sense of this work.

In contrast, Masuch and Strothotte [2001] combine different styles in one image. They use a photorealistic rendering or photographic image as a background and add a non-photorealistically rendered object as foreground in order to visualise the uncertainty and the degree of trust in the reconstruction of ancient architecture. When using a photo, it is natural that only still images are produced (see also Strothotte et al. [1999]). However, when including the NPR renditions into the

photorealistic background, real-time animations can be created. On the other hand, a continuous transition from the photorealistic style to non-photorealism or vice versa cannot be achieved by the presented approach.

Ritter et al. [2003] use additional silhouettes for semi-transparent objects in interactive medical illustrations to increase the user's ability to recognise these objects.

Halper et al. [2002] present an interface for easy combination of various different rendering styles. Being conceived as a tool for designers to assemble and experiment with renditions it makes it easy to come up with new combinations of styles including the possibility to create hybrid renditions.

A different kind of hybrid animation is used in movies such as »Who Framed Roger Rabbit« (1988) or »Space Jam« (1996). In this type of movies, real footage is combined with hand-drawn cartoon characters. Johnston [2002] improves the appearance of the cartoon characters by creating toon renditions using the light positions from the real scene. When combining it with real video footage, the more coherent $2\frac{1}{2}$ D look of the character with correct highlights is achieved by estimating normals from the hand-drawn silhouette using a set of heuristics.

Jesse and Isenberg [2003] describe a hybrid rendering system which combines non-photorealistic styles with regular photorealistic shading techniques. In order to provide a maximum of rendering and application flexibility, the use of each rendering style is controlled either interactively or by a script based scene description. The overall script structure as well as a specific application example is presented by Jesse [2003].

3.4.2 Overview of Style Blending

Non-photorealistic rendering techniques can roughly be divided into surface shading techniques and line-oriented methods. In addition, combinations of both are possible. Usually, both variants of NPR techniques can be used individually, e. g., a cartoon shading can be used as well as just displaying the visible silhouettes of a model. However, when combining NPR techniques with photorealistically rendered models this does not necessarily hold anymore. For example, using a photorealistically shaded model and only displaying one object of the model using silhouettes can be very confusing since the rest of the model is visible through the silhouette lines (see Figure 3.15). Thus, in cases where line-based NPR techniques are used, it is usually advisable to combine these techniques with some kind of surface shading, either photorealistic or non-photorealistic (see Figure 3.16).

As an example for a line-based NPR technique, stylised silhouette rendering is employed [Isenberg et al., 2003]. This implies that line stylisation and rendering happens after the model has been rendered because a correct z-buffer is necessary for the hidden line removal of the silhouette.

For a smooth transition between the regular photorealistic shading of an object's

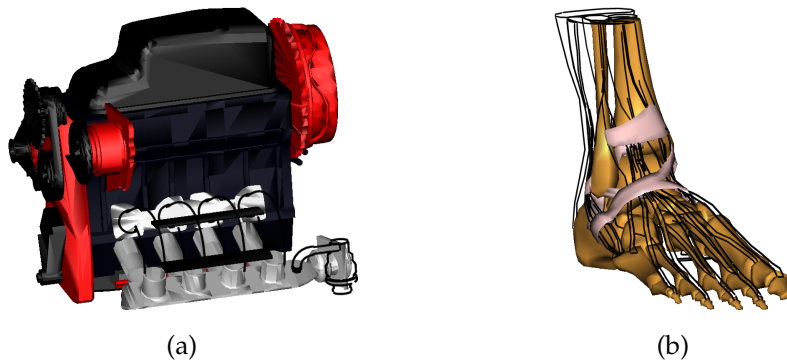


Figure 3.15: When the shading of an object is exchanged with a line-based NPR technique, the photorealistically rendered remainder of the model is seen in the background, which is very distracting for many applications.

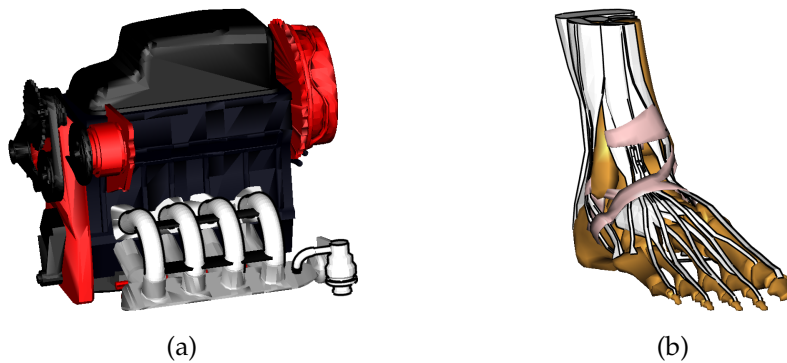


Figure 3.16: Using a shading to enhance the appearance of the line-based NPR style. In contrast to Figure 3.15, hidden surfaces do not shine through the non-photorealistically rendered parts of the models.

surface to non-photorealistic shading, one style has to be continuously de-emphasised while the other style is successively emphasised. Typically, this type of transition is achieved by using α -blending. This method will be discussed in Section 3.4.3. An alternative way to create the background shading for line-based NPR styles as mentioned above will be introduced in Section 3.4.4.

3.4.3 α -Blending

An intuitive implementation of a transition between different rendering styles produces renditions of both styles and uses the α -channel for blending. That is, transparency is used in order to allow for the perception of different object representations and their respective rendering styles. Besides the use of classical illumination models for photorealistic rendering, this allows for a transition to various NPR-shadings of a model, such as gray-scale or Gooch shading [Gooch et al., 1998].

Figure 3.17 shows an example of using variations in a non-photorealistically ren-

dered scene. In the upper row of the figure, three renditions are shown that reflect increased transparency from left to right. In contrast, the lower row shows variations of the scene's rendering where the shapes' surface colour is changed as well as the colour of shape edges. This illustrates the bandwidth of variations in a rendering that also form the basis of the dynamics as presented here.

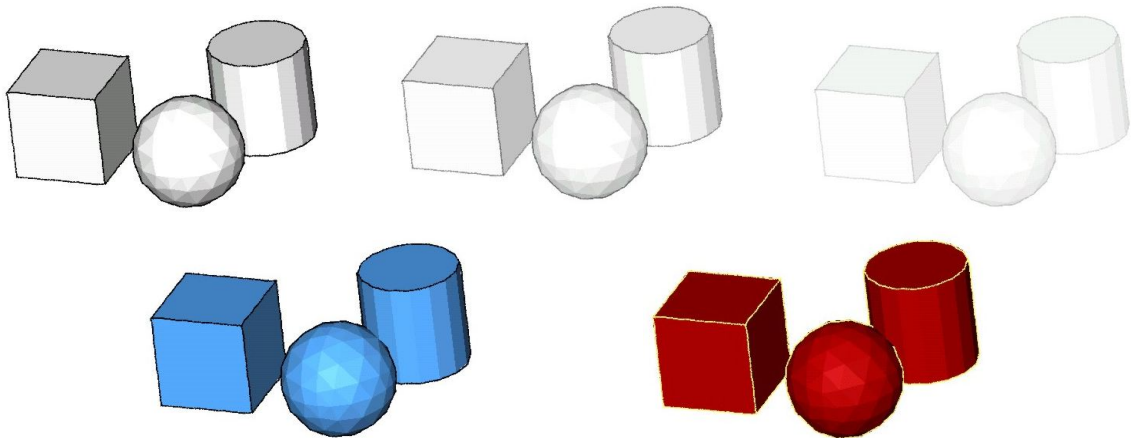


Figure 3.17: Variations of rendering a scene non-photorealistically based on transparency in contrast to colour changes [Nettelbeck, 2003].

Different kinds of blending can be used with regard to the order of rendering of all objects. When using a *normal* blending order, all objects are rendered as they are represented in the geometric model. An alternative is *delayed* blending which cares about opaque objects first and renders transparent objects afterwards. By using *sorted* blending, all opaque and transparent objects are rendered in an order depending on their respective distance from the camera's position.

This is illustrated by Figure 3.18. Normal blending is used in 3.18(a). Clearly visible is a partial occlusion of the cube by the sphere, caused by an unfortunate ordering of both objects in the geometric scene description. On the contrary, the correct ordering of the sphere with regard to the cylinder causes the latter to stay completely visible throughout the transparent sphere. Figure 3.18(b) shows the same scene as rendered with sorted blending. That is, all objects are handled according to their distance from the camera. As a result, the sphere appears transparent to both remaining objects.

This directly leads to the *see-through effect* in α -blending: As one object is rendered in two different styles, these styles visually interact with each other by being visible through their respective transparent counterpart. This effect is well visible in the third and fourth image of Figure 3.19 that shows a series of snapshots from an animation created by blending an eye's muscle continuously from a realistic rendering style to silhouette line style by steady adjustments to the α -channel. A naïve approach to avoid the (unwanted) see-through effect is to render the NPR shading

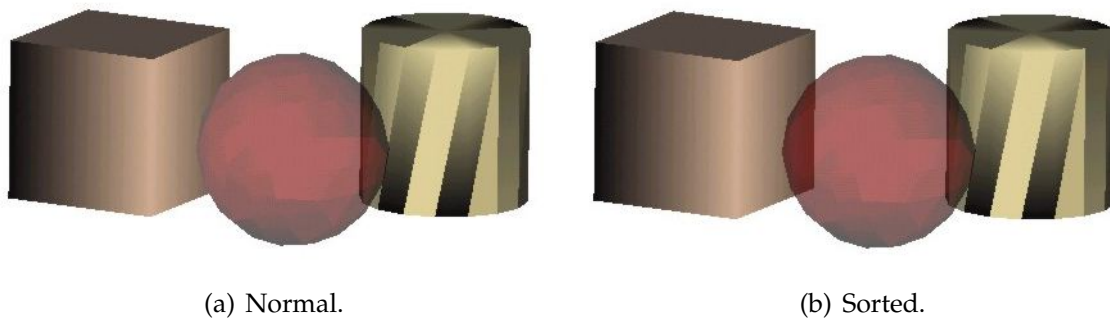


Figure 3.18: Problems with sorting when using α -blending [Nettelbeck, 2003].

non-transparently first, followed by an opaque rendering of the realistic shading style. The latter is then continuously made more transparent until it is completely blended out. This guarantees a smooth transition from a realistic rendering style to a non-realistic style.

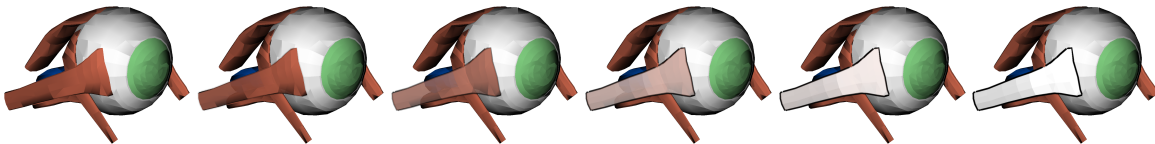


Figure 3.19: Snapshots from an animation created by blending from a realistic rendition to a non-realistic rendition regarding a part of the model. For this purpose, α -blending is used. Note the *see-through effect* in the third and fourth image causing faces hidden by the blended object to shine through for some period of time during the animation.

However, this approach is challenged by the problem that the size of non-photorealistically rendered objects usually extends the size of its respective realistic counterparts slightly. An example of this effect are stylistic silhouette renditions. This especially holds for the case of silhouette lines shaped as waves, but is valid for other line shapes as well. As an alternative, the stencil buffer might be used in order to render the face content in an opaque manner early on and blend in the face border. As of now, the see-through effect is just accepted as a rendering artifact.

Another challenge when dealing with simultaneous rendering of two different styles for the same object is in possible flickering caused by »triangle fights«. As triangles in both styles are located at approximately the same depth with regard to the camera viewpoint, they appear to flicker because of limited z -buffer resolution. At some angles, one of two respective triangles appears to be closer to the viewer, at other angles the other triangle does. In order to avoid this visually unpleasant effect, the triangles need to be layered accordingly. This is achieved by using *OpenGL's* polygon offset for defining how to offset specific triangles with respect to others.

3.4.4 Emissive Colour Blending

An alternative to modifying the α -channel of a material is in adjusting the emissive colour of objects in a scene. In order to blend from one style to another, the RGB values of the colour are continuously changed from 0.0 to 1.0. A vector of (0.0, 0.0, 0.0) does not influence the colour representation of an object's material at all. A colour triple of 1.0 values results in the object being rendered completely white. By incrementally selecting appropriate steps in between, blending from a realistic rendering style to a non-realistic style can be achieved. Similarly to α -blending, the remaining part of the scene is not influenced, as all modifications are local and no light source is touched. Figure 3.20 shows a series of snapshots taken from an animation created by increasing the RGB values of an eye's muscle from 0.0 to 1.0.

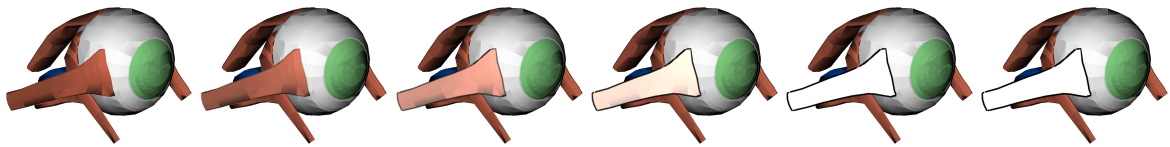


Figure 3.20: Snapshots from an animation created by changing the emissive colour RGB values of a muscle continually from 0.0 to 1.0. Note that no shading artifacts are visible for this specific muscle and that the *see-through effect* from Figure 3.19 is avoided.

In order to achieve the effect of white lines on black background, the RGB values are reduced to the range $(-1.0, -1.0, -1.0)$. This way an effect of the illumination model is used that is not possible in physics. The objects »suck in« the light from regular shading. To get an even better appearance, the background of the rendition can be changed to black as well.

Depending on the choice of colours to be used as RGB values, two visual metaphors can be created: the effect of sketching with a pencil on a white sheet of paper or the effect of using chalk on a blackboard. Only black and white pen colours are to be used in case a rendition without any shading artifacts is to be achieved. Using alternative colours not only results in shading still being represented but in the visual metaphors being less perceivable.

An effect of modifying the emissive colour of an object in order to blend from one style to another is in completely covering the affected faces. This holds especially for large RGB values $(1.0 - \epsilon)$, with a small ϵ . On one hand, this explains why shading artifacts are avoided. Furthermore, only a single shading technique is used and no transition between two shadings and transparency made necessary. On the other hand, this prevents other NPR shading techniques—such as Gooch shading [Gooch et al., 1998]—from being usable simultaneously. Figure 3.21 shows a series of snapshots from blending different rendering styles based on changing the emissive colour of the affected objects.



Figure 3.21: Snapshots of an illustration emphasising the individual bones of a toe by use of dynamic rendering styles achieved through changes of the emissive colour of the affected bones.

3.4.5 Comparing Style Transitions

Given the above discussion, dynamics by variation of hybrid rendering styles are defined as:

$$\mathcal{D}_{RS} = \mathcal{D}_{RS_\alpha} \cup \mathcal{D}_{RS_{col}}.$$

That is, dynamic rendering styles are formed as a union of blending by transparency changes in \mathcal{D}_{RS_α} and blending by adjusting the emissive colour of an object in $\mathcal{D}_{RS_{col}}$. This definition allows for styles derived as combinations of both subsets.

Figure 3.22 presents two snapshots of the same scene. One object in the scene is rendered with a silhouette line shape whereas the remaining objects in the scene are rendered with a photorealistic style. Figure 3.22(a) represents the use of α -blending for this purpose and the result of changing the emissive colour is shown in Figure 3.22(b), respectively.

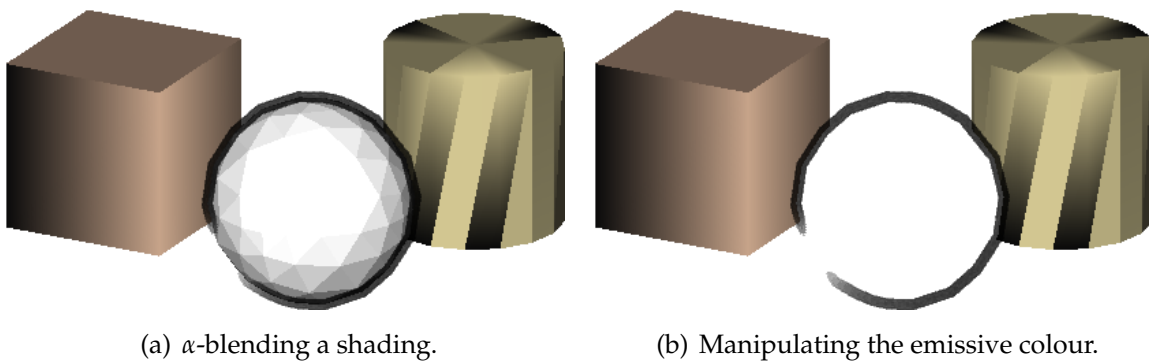


Figure 3.22: Comparison of both methods for generating hybrid renditions. In both cases, the silhouette in form of a line stroke is combined with a background [Nettelbeck, 2003].

As can be seen clearly, the shading artifacts are still visible in the left image, that is for α -blending. Even though flat shading is used in this specific case in order

to illustrate the effect, it holds for advanced shading techniques as well. Besides avoiding the shading effect, Figure 3.22(b) expresses the *pen on paper* metaphor.

Both blending techniques are capable of fulfilling the requirements according to cognitive limitations as expressed by the dynamics stimulus window from Section 2.2. Stepping over the lower window boundary is automatically ensured as each individual rendering style already fulfils this requirement. Therefore, any combination of multiple styles does not fall below the lower temporal boundary. In order to guarantee not to extend the dynamic stimulus window's upper limit as well, any style changes need to be parameterised accordingly. For the worst case of very complex scenes and low rendering bandwidth, a transition from one rendering style to another can be presented in as less as two steps—one for each end of the blending pipeline. However, both presented blending techniques are designed to provide for smooth transitions between different styles within the complete range of the dynamic stimulus window.

The use of more than just one shading technique is only possible with α -blending. This requires two rendering passes of the scene, though: once using a classic photo-realistic style and once for the desired NPR style. The resulting drawback in rendering time is at least partially compensated by α -blending often being implemented directly in graphics hardware. Due to the visual metaphors achieved by the emissive colour technique, its rendering results appear more artistic. This allows for the perception of an extended degree of abstraction compared to α -blending.

3.5 Zoom-based Distortion Histories

The third part of this chapter's collection of dynamic presentation techniques is formed by distortion histories. These histories represent steady changes of an object's shape over time. Steady changes of shape can be divided in two distinct types:

1. Changes of an individual object's shape, and
2. Changes of the relative positions of a group of related objects.

Changing only a single object provides a means of dynamics without manipulating the overall scene construction. The object's spatial correlation to other objects is not affected. This technique is therefore to be used preferably when the shape consistency of a specific object is of minor importance compared to its influence on scene context. The change in shape can be done either by repetitive scaling or switching the shape's kind.

Changing the relative positions of a group of respective objects allows to express inherent relation characteristics of these objects. The respective objects can be manipulated by similar changes in shape. Additionally, translation changes as subject

of Section 3.3 help to create distortions providing an effect of explosion. The exemplary distortion techniques presented here are variations of the fisheye zoom. As discussed in Section 3.2.4, fisheye is a powerful tool for interactive exploration purposes. Magnification and demagnification work in tandem to create a distortion that emphasizes information of interest in an area of context information.

An overview of distortion techniques including fisheye zooms is presented by Leung and Apperley [1994]. The presented techniques have two problems in common:

1. A spatial challenge of fitting the information space onto the presentation space while preserving spatial relationships.
2. An information density challenge where the actual information is to be fitted into its context.

Addressing both challenges is a central design goal for various zooming techniques. The second case is specifically addressed by fisheyes. The extensions presented in this section target at providing a distortion history by spatial relationship information. This deepens context information communication.

Extending fisheye-based views with distortion history information is further supported by two studies carried out by Skopik and Gutwin [2003]. These studies address how navigating fisheyes is to be carried out with regard to memorability. This memorability describes the ability to find and go back to objects and respective features in the data. That is, during the process of examining some information object of interest supported by a fisheye zoom, visual presentation of connectivity and history information between the zoomed object and its un-zoomed original counterpart might prove helpful. These connectivity presentations of an illustration act like *landmarks* in an information space.

A combination of non-photorealistic rendering techniques with zoom metaphors promises to extend the expression capabilities of an illustration system. Context information for individual objects may potentially be lost by making those objects subject to zooming. This information can possibly be preserved by applying an appropriate NPR style for displaying information on the history of distorting the objects. Zooming possibly results in an invalidation of the spatial relation of two objects. In this case, a suitable parameterisation of NPR styles helps to convey this relation. This results in distortions without spatial gaps for the purpose of preserving spatial relations while local details are explored. Two techniques are presented for this purpose: speedlines supporting coherent zooming and the chewing gum zoom.¹ Before the discussion of these extension techniques for a fisheye zoom, some necessary basis of the zoom will be presented. Both techniques are subject of Subsection 3.5.2 and 3.5.3, respectively.

¹ This section presents the principles and design of both techniques. A prototypical implementation has been subject of a diploma thesis rendered in the context of this work [Davydova, 2003].

3.5.1 Zoom Basis

A classic zooming technique is the fisheye lens as proposed by Furnas [1986]. This visual metaphor is derived from the field of photography. A fisheye lens is a very wide-angle lens. It provides detailed information in the projection centre. With increasing distance from this centre, surrounding areas represent respectively less information. Using this metaphor for illustrating computer graphics provides detailed local focus information with less detailed global context. The first exemplary fisheye browser for 2D layouts is presented by Sarkar and Brown [1992].

A discussion of further works based on the fisheye lens is presented in Section 3.2.4. While the original fisheye algorithm by Furnas [1986] presented a 2D approach, some of the extensions provide a 3D presentation. One such extension is the *3D Zoom* by Raab and R uger [1996]. This introduces a 3D zooming technique based on *interval structures*. These structures reflect spatial object boundaries and are used for determining flexibility values for object movement and resizing in support of a zoom operation. Some details of this technique are discussed below as this algorithm forms the basis of both dynamic zooming techniques that follow.

Fundamental Zoom Algorithm

The *3D-Zoom* provides support for multiple focus points in a 3D scene. Spatial variations are based on the notion of an *interval structure*. The intervals of this structure specify the zoom boundaries and are derived from the object's bounding boxes. Figure 3.23 illustrates this. The lefthand image of the figure shows a 2D representation of intervals derived from two objects. It is obvious that intervals may overlap in case their respective objects overlap as well. The righthand image points out constraints with regard to neighbouring objects which will be discussed shortly.

The individual steps of Raab and R uger's zoom are outlined in Algorithm 3.3. Lines 1 to 5 of the algorithm describe the initial creation of the interval structure. First, all intervals are constructed by an evaluation of available object bounding boxes. The gained intervals are normalised such that for each spatial dimension (x, y, z) the overall value is *one*. The remaining lines 6 through 18 describe the algorithm's work necessary for each single zooming step.

For each step of the zoom, the interval structure is evaluated and updated. Based on the degree of interest of the available information objects ($\text{doi} (IO)$) the object's new size and position are determined. After finishing this process for the whole information set, some optional adjustments to individual objects are applied for an increase in fulfilling the user scope. These include deformations of objects affected by a low doi . As these objects are of less importance with regard to the user scope, they may be deformed by changing their aspect ratio. This provides some increase in communicating object relations specifically for objects far away from the user [Raab and R uger, 1996, Sec. 3.3]. Additionally, some low doi values may be

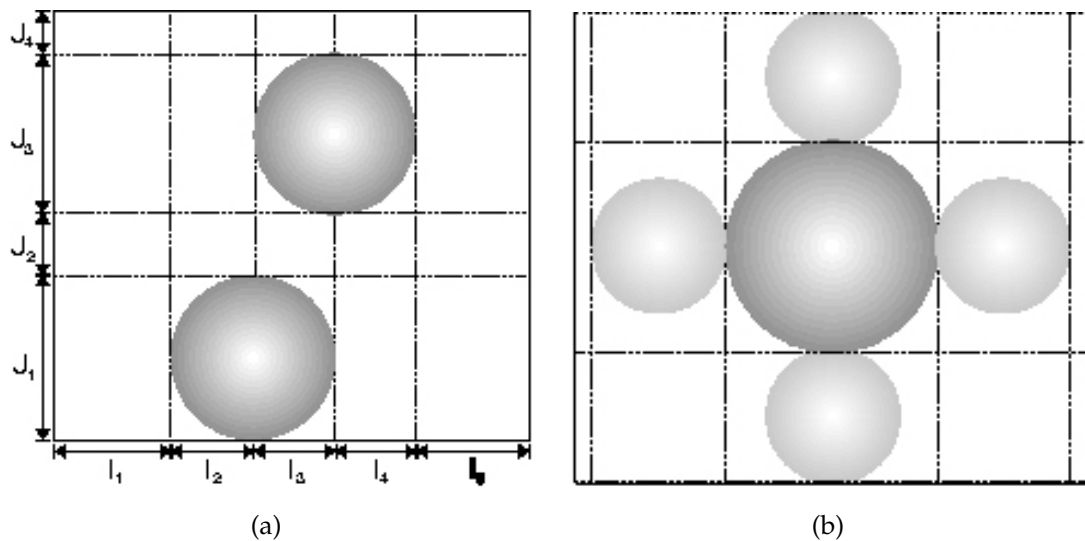


Figure 3.23: Interval structures of the *3D Zoom* [Raab and Rüger, 1996]. Subfigure (a) shows the 2D interval structure for the zoom, in (b) constraints with regard to neighbouring objects are shown.

mapped onto transparency for the purpose of deemphasising the respective objects. For this, care should be taken with regard to accumulating doi -based transparency values which possibly results in misinterpretations of the illustration.

3.5.2 Trace Line Support for Zoom Histories

This subsection introduces an enhancement to zooming that provides dynamically created traces of a zooming motion. These traces are shaped analogous to speedlines representing the moving objects' contours. This allows to present simultaneously the history of objects which are subject to a zooming motion and the objects them-self.

Context

Speedlines are classically used for the presentation of motion of objects in (computer generated) still images as outlined in Figure 3.24 [Masuch et al., 1999]. The motion of an object is thereby depicted by a set of contour lines that are drawn at intermediate motion steps. These lines illustrate the motion itself, the motion direction, and possibly some spatial relations of moving object parts. The motion trajectory is thereby depicted by a gradually decreasing intensity of drawn speedlines. Therefore, the original location of a moved object can still be derived after the motion has been completed.

Different types of speedlines are depicted in the figure. The two innermost cases use contour lines of the moving object as speedlines. Whereas the first of these

Algorithm 3.3 Basis for the zoom extensions following Raab and Ruger [1996].

```

1: Define set of intervals  $SI$  in each spatial dimension  $(x, y, z)$ 
2: for all  $i \in SI$  do
3:   Reduce size of empty  $i$ 
4:   Normalise  $i$  such that  $\sum i = 1$  for each dimension holds
5: end for
6: for all zoom steps do
7:   for all  $i \in SI$  do
8:     for all  $IO \in IM$  in  $i$  do
9:       Determine scale value  $s_{IO}^i$ 
10:      Define size of  $IO$  as  $\min(\text{doi}(IO), \text{size}(i \cdot s_{IO}^i))$ 
11:      Determine position of  $IO$ 
12:      Optional deformation of  $IO$  with low  $\text{doi}$ 
13:      Optional increase of  $\alpha$  for  $IO$  with low  $\text{doi}$ 
14:      Update  $i$ 
15:     end for
16:   end for
17:   Render zoom step
18: end for

```

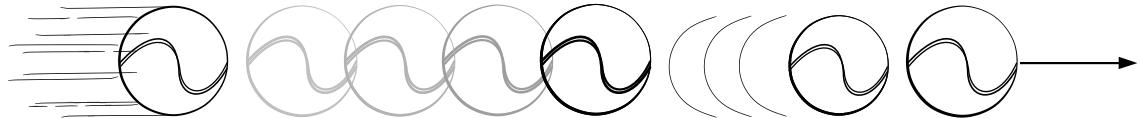


Figure 3.24: Speedline supported illustration of a moving ball. Contours are used respectively with varying transparency. In addition, parts of contours as well as an arrow extend the expression of dynamics. [Masuch et al., 1999]

two approaches makes use of the whole contour, a simplified version is derived for the other approach where only parts of the contour define the speedlines. The remaining two cases draw lines parallel to the motion trajectory. In the leftmost case, this helps to point out the trajectory along which the object moved so far. The rightmost approach for a speedline differs from all other cases as it adds an arrow to the speedline for the purpose of pointing out a future motion and not one that has already taken place.

Algorithm

The idea of using speedlines for motion representation is now introduced to zooming based on Algorithm 3.3. As these lines represent traces of objects along their zoom paths, the speedlines are referred to as *trace lines*. Two characteristics define these lines:

1. Trace lines repeat the contour of zoomed objects, thereby representing their shape and position along the zoom trajectory.
2. The intensity of a trace line decreases with an increase in distance of the zoomed object with respect to its origin.

The effect addressed by the trace line enhanced zoom is shown in Figure 3.25. This shows a scene which is simply a collection of six basic objects. Originally, all of these objects are placed in a bulk. A zoom is used to stretch the space between all objects. For this purpose, the doi of the cone is selected higher compared to all other objects. Therefore, these are moved away from the cone which stays in the centre. For each of the zoomed out objects, a set of trace lines is drawn that renders the respective motion path. This way, the objects' original composition is still communicated while all objects are placed with enough space in between to allow for individual exploration.

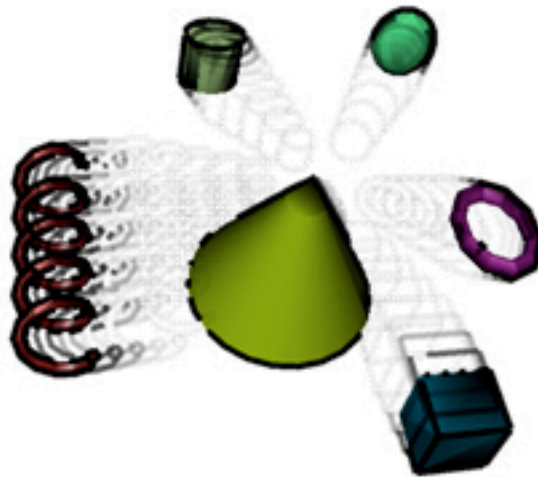


Figure 3.25: Effect achieved by a trace line supported zoom [Davydova, 2003].

Figure 3.26 shows another set of snapshots of a zoom animation supported by trace lines. The spatial relation of an object part that is moved out of its original spatial scope is outlined by trace lines which are rendered at intermediate animation steps.

For the creation of trace lines, a set of presentation techniques is maintained during an active zoom operation:

$$\forall IO \in \mathbf{IM} : \mathcal{D}_{TL}(IO) = \{d_{TL_1}(IO), \dots, d_{TL_n}(IO)\} \quad n \in \mathbb{N}.$$

Thereby, each d_{TL} describes a set of presentation characteristics:

- *Translation*: The position of the trace line.
- *Scale*: The scale factor of the trace line with regard to the original size of its respective information object before the zoom.
- *Intensity*: The current intensity of the trace line which is constantly decreased while the zoom operation is in progress.
- *Contour*: The shape of the trace line as derived from the information object.

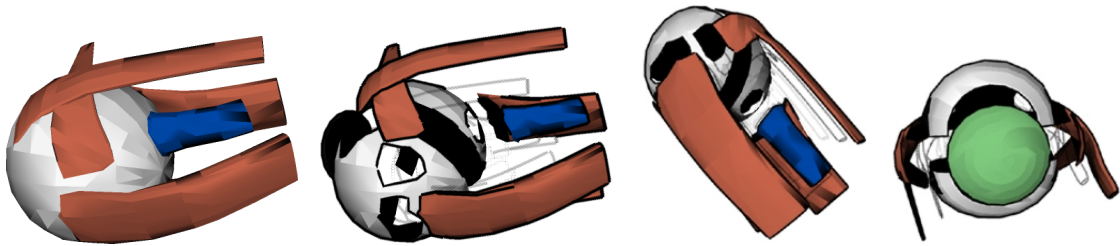


Figure 3.26: Use of speedline inspired trace hints in order to outline object relations after a zooming distortion. (courtesy of a co-work with Davydova [2003])

Using the complete contour information of the information object for the presentation of its trace lines might result in an overly cluttered illustration. To overcome this effect, only parts of the contour may be used for which the normals face away from the current zoom direction of the object. Figure 3.27 shows both of these cases. The lefthand image represents the use of complete contours for presenting traces. The righthand image simplifies the traces by using partial contours with back-facing normals. The intensity is adjusted during each step of zooming.

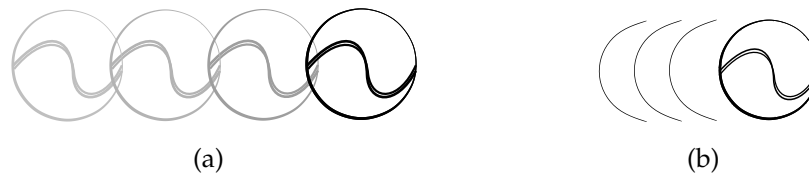


Figure 3.27: Creating traces based on contours of the respective information object. Subfigure 3.27(a) shows the naïve approach of using the complete contour. Subfigure 3.27(b) points out use of partial contours to simplify the presentation and provide for a less cluttered illustration.

While the distance of the information object to its original position before the zoom increases, the distance to the traces already drawn increases as well. If $\text{pos}(IO)$ determines the current position of an information object during a zooming operation and $\text{dist}(\text{pos}(IO_1), \text{pos}(IO_2))$ determines the distance of two such positions, the intensity (lightness) l is given as:

$$l_{cur}(IO) = \frac{l_{orig}(IO)}{\text{dist}(\text{pos}_{cur}(IO), \text{pos}_{orig}(IO))}. \quad (3.5)$$

Thereby, $l_{cur}(IO)$ describes the intensity of the trace described by the current d_{TL} , $l_{orig}(IO)$ describes the original intensity of the information object IO . As is obvious from the equation, the lightness of a trace decreases with an increase of the respective information object's distance from its origin.

As the last component of the trace description by d_{TL} , the contour of the information object is determined by a silhouette rendering. No specific silhouette extraction and rendering techniques are developed. Instead, existing approaches such as those described by Isenberg et al. [2003] are used.

Algorithm 3.4 summarises the general procedure used for trace line enhanced zooming. This basically extends the zoom basis as described above by the creation and constant maintenance of trace representations of each information object subject to zooming.

Algorithm 3.4 Trace line enhanced zoom operating after line 13 in Algorithm 3.3.

```

for all  $d_{TL} \in \mathcal{D}_{TL}$  do
  Determine  $l_{cur}(IO)$  according to equation (3.5)
end for
Define current  $d_{TL_{cur}}(IO)$ 
Add  $d_{TL_{cur}}(IO)$  to  $\mathcal{D}_{TL}$ 

```

3.5.3 The Chewing Gum Zoom History

The idea of this zoom extension builds on two objects being glued together by a chewing gum. In case these two objects get zoomed out from each other, the gum is stretched but outlines the spatial relation of both objects. This connectivity information is incrementally thinned out while the objects are moved farther away from each other. As soon as the distance of both objects with regard to each other exceeds a *connection threshold*, the glueing gum gets ripped up. Visually, both objects no longer belong to each other.

Using this idea for a zoom-based illustration helps to reflect the spatial dynamics history of a zoom-affected information object as well as its spatial relations with regard to other objects. Similar to the trace line enhanced zoom as presented above, the zoom path of an object is illustrated by the chewing gum. In contrast, though, this information is made more explicit by providing continuous connection from the zoomed object to its zoom origin. Removing this connection gum based on a connection threshold helps to avoid visual cluttering and to reflect spatial connectivity limitations.

Context

For the creation of a chewing gum, the gum needs to be derived from the shape and position of affected information objects. A variety of approaches exists for this purpose. These include point-based geometry as display primitive, such as the surface elements *surfals* by Pfister et al. [2000]. *Surfals* are based on work presented by Levoy and Whitted [1985] as well as the work by Grossman and Dally [1998]. Another approach for point-based geometry is the hierarchy of bounding spheres in the *QSplat* system [Rusinkiewicz and Levoy, 2000, 2001]. Boolean operations support variations of free-form solids such as a gum-like volume [Adams and Dutré, 2003]. Merging of polyhedral shapes with scattered features is discussed by Alexa [2000]. This merging acts as basis for a 3D morphing process. Morphing in the sense of transforming the gum volume by use of a metamorphosis function as presented by Cohen-Or et al. [1998] is used for animating the chewing gum throughout the zoom process below.

Algorithm

The chewing gum connecting an information object affected by zooming with its origin is defined by the following four characteristics:

1. The length of the chewing gum correlates to the zoom distance of the respective information object.
2. The amount of volume of the chewing gum does not change during zooming.
3. The chewing gum is ripped up if the zoom distance of the respective information object exceeds a connection threshold.
4. The shape of the chewing gum is described by a set of faces forming a solid with an inverse cubic bezier surface boundaries.

A basic sketch of constructing the chewing gum is outlined in Figure 3.28. For simplicity reasons, the figure shows a 2D representation. The real application in 3D works analogously whereby the gum's volume is derived by a shape preserving extrusion of the 2D volume. The figure shows an information object that is subject to a zoom operation. This causes the object to move from its origin position $p_0(x_0, y_0)$ to a zoom position $p_z(x_z, y_z)$. This zoom position changes constantly while the zoom is active.

The chewing gum itself is introduced as a representation $\mathcal{G}(IO)$ reflecting the spatial relation of an information object IO with regard to its un-zoomed origin. \mathcal{G} is defined as a set of transformations:

$$\mathcal{G}(IO) = \{G_t(IO) \mid 1 \leq t \leq n_z\}. \quad (3.6)$$

Thereby, n_z is defined as the overall number of zooming steps to be carried out. Each G in this set is defined by a chewing gum morph function:

$$G(IO) = f_{CG}(b_{CG}(IO), l, d_{min}).$$

That is, this function is defined as a set of constraints consisting of:

- b_{CG} being a function determining the connection diameters of the gum.
- l specifies the current length of the gum.
- d_{min} holds the current minimum diameter at position $l/2$.

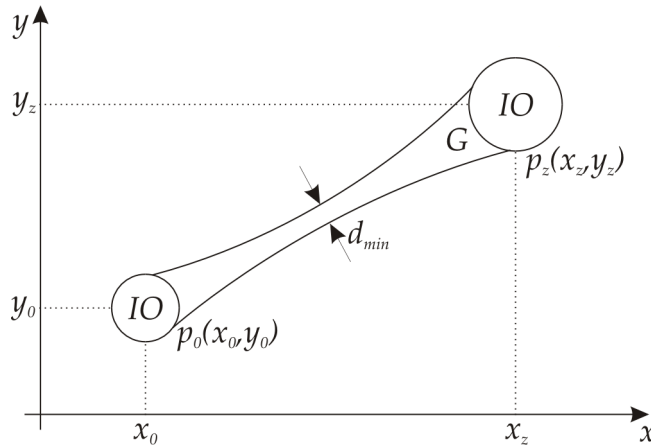


Figure 3.28: Sketch of the chewing gum in 2D.

The function b_{CG} determines the border diameters of the gum based on the respective information object. This is a three step procedure illustrated in Algorithm 3.5. First of all, the gum is attached to the faces of the information object which have normals pointing towards the zoom direction. Secondly, the midpoint of the gum is determined based on the middle of these faces. Finally, the width of the chewing gum is derived by using the information object's width ($y_t - y_0$) scaled by the user-supplied *range* factor. As this factor is applied to only one half of the information object's overall width, symmetry is gained by multiplying the result with 2.

The function f_{CG} is initially used once to determine the starting shape of the gum at position p_0 . For each zooming step, the function is used to update the end shape of the gum at the current zoom position p_t .

Besides the shape and position of the respective information object, $G(IO)$ is defined by two constraints defining the chewing gum: the length of the gum and its minimum diameter at position $l/2$. These constraints are pointed out in Figure 3.28.

This definition of gradually deforming the chewing gum follows the idea of a distance field metamorphosis as presented by Cohen-Or et al. [1998]. This metamor-

Algorithm 3.5 Border diameter function b_{CG} for the chewing gum zoom.

Require: $range$: user-defined range factor in $[0, 1]$

Require: p_0 : initial position of IO before zoom

Require: p_t : current zoom position of IO

1: $sf \leftarrow$ faces of IO where normals point towards zoom direction

2: $cg_{mid} \leftarrow mid(sf)$

3: $cg_{width} = 2 \cdot range \cdot (y(p_t) - y(p_0))$

Return: $bcg(cg_{mid}, cg_{width})$

phosis describes a transformation of an object's shape. This is a two-step function consisting of a *rigid* and an *elastic* transformation. For this purpose, the distance field calculations are done on a volumetric representation of the object in question. A set of anchor points controls the *warp function*. These anchor points are described by a sequence of keyframe models. The remaining field distances get interpolated.

Applied to the chewing gum morph function, this metamorphosis results in an iteratively defined chewing gum. Each iteration corresponds to one zoom step. At each step intermediate images from zooming are reused. As a result, the chewing gum is stretched depending on the current position with regard to start and end of the zoom. The chewing gum is transformed from its original source shape G_S to its final target shape G_T . The source shape of the chewing gum is defined absolutely at the point in time, the zoom is initiated. The target shape depends on the zoom progress and is not necessarily known initially. All intermediate steps of the zoom are derived as a morph as stated by Equation (3.6). This leaves each entry $G \in \mathcal{G}$ as:

$$G(IO) = \begin{cases} G_S(IO) & \text{initial chewing gum,} \\ G_T(IO) & \text{final chewing gum,} \\ G_t(IO) & \text{gum constantly adjusted at each zooming step.} \end{cases}$$

Thereby, $G_S(IO)$ and $G_T(IO)$ are special cases of $G_t(IO)$ with $t = 0$ and $t = n_z$, respectively. The only two anchor points in the sense of Cohen-Or et al. [1998] are given as the initial chewing gum $G_S(IO)$ and the gum with the minimum diameter corresponding to the *connection threshold* d_ϵ . This latter case defines a rip up of the chewing gum before the zoom operation is finished. Therefore, two cases of finishing the gum presentation are defined:

$$G_t = G_T \Leftrightarrow \begin{cases} t = n_z \\ d_{min} \leq d_\epsilon \end{cases}$$

The current minimum diameter of the chewing gum at each zoom step, d_{min} , is determined logarithmically on the basis of l . This ensures that the gum is thinned out smoothly throughout the zoom process. Overall, Algorithm 3.6 points out the assignment of the gum at each zoom step. The special case of ripping up the chewing

gum when its minimum diameter gets too small is handled in line 4 where the gum is set to 0.

Algorithm 3.6 Chewing gum function f_{CG} applied at each zoom step.

Require: p_z : current position of IO

Require: d_ϵ : user-defined minimum chewing gum diameter

- 1: $l = p_z - p_0$
- 2: $d_{min} = \log(l/2)$
- 3: **if** $d_{min} \leq d_\epsilon$ **then**
- 4: $G(IO) \leftarrow 0$
- 5: **else**
- 6: $G(IO) \leftarrow \{b_{CG}(IO), l, d_{min}\}$
- 7: **end if**

Return: $G(IO)$

The individual parts forming the chewing gum can now be assembled as illustrated in Algorithm 3.7. This enhances the basis algorithm for zooming (3.3) by the chewing gum. Algorithms 3.5 and 3.6 are used as parts of the assembly. Integration of the individual chewing gum steps into the original Algorithm 3.3 takes part right before the first step as well as between step three and four. This latter case ensures that the chewing gum is updated at every iteration of the overall zoom loop.

Algorithm 3.7 Zoom enhanced by a chewing gum for spatial connectivity.

Require: Steps 1 to 5 of Algorithm 3.3

- 1: Create initial chewing gum $G_0(IO)$ based on Algorithm 3.5
- 2: $G_p(IO) = G_0(IO)$
- 3: $p_z = p_0$

Require: Steps 6 to 14 of Algorithm 3.3

Require: $\mathbf{IS} \leftarrow \mathbf{IM} \cup G_p(IO)$

- 4: Update p_t
- 5: $G_t(IO) \leftarrow f_{CG}$ by Algorithm 3.6
- 6: $\mathbf{IS} = \mathbf{IS} \setminus G_p(IO)$
- 7: **if** $G_t(IO) \neq 0$ **then**
- 8: $\mathbf{IS} = \mathbf{IS} \cup G_t(IO)$
- 9: **end if**

Return: \mathbf{IS} and continue Algorithm 3.3

Figure 3.29 shows a series of exemplary snapshots of a presentation where a prototypical implementation of the chewing gum is employed. A gum is applied to the individual objects in the scene. From left to right, the figure represents different stages throughout the zoom process. The gums are stretched and thinned out, respectively. They finally break up as shown in part in the rightmost image.



Figure 3.29: Exemplary snapshots of a presentation using a prototypical implementation of the chewing gum [Davydova, 2003]. Different gums are used for the individual objects. From left to right, different stages throughout the zoom process are shown.

3.5.4 Discussion

Two approaches are presented above for the purpose of defining dynamic distortion histories based on extending a zoom presentation: trace lines and a chewing gum. Both serve as examples for principle approaches of extending an already existing dynamic technique. These extensions may be used for communicating history information about an object which is subject to zoom-based distortion. This provides not only information about the particularly affected object but possibly about other objects as well.

Both extensions provide a means of spatially limiting the influence they have on scene coherence. Early drawn trace lines are gradually faded out by an increase of their transparency while the zoom progresses. Thinning out the chewing gum is one of its central characteristics. Furthermore, a connection threshold helps to stop an active chewing gum illustration in case its overall spatial effect exceeds the desired goal as expressed in the illustration target function. As of these constraints, both zoom extensions avoid a global effect on scene coherence and restrict it to be of regional influence.

The combination of the trace line zoom extension with the chewing gum does not promise to provide for an extended expressiveness. Both approaches affect the same spatial region: the distortion volume spanned by the object which is subject to zooming. In order to avoid interference of both techniques, a combination is only reasonable after a rip up of the chewing gum. This rip up is caused by the goal of limiting the influence of the zoom on scene coherence, though. Using any consecutive combinations of zoom extensions results in extending this spatial window. This extends the overall effect from a regional to a global influence.

3.6 Classification Based on Effect on Scene Coherence

Table 3.10 classifies the dynamic presentation techniques discussed in this chapter with respect to their influence on scene coherence. This puts all techniques in relation with regard to each other.

Three kinds of *translations* are defined for oscillations as sketched in Figure 3.6 of Section 3.3.1:

Techniques	Effect on scene coherence
motion	
oscillations	
translations (1)	regional, global
translations (2)	regional, global
translations (3)	regional
rotations (1)	local, regional
rotations (2)	regional, global
structural change	local
mural	local, regional
hybrid styles	local
distortion histories	
chewing gum zoom	regional, global
trace line zoom	regional, global

Table 3.10: Effect on scene coherence of presented dynamic presentation techniques.

1. movement of an object along a straight path between two fixed positions,
2. movement of an object along a fixed path passing key positions, and
3. movement along a free path in a constraining region.

These classes differ in their respective degrees of freedom of applying motion to affected information objects. The first two classes possibly result in not only a regional, but a global effect on scene coherence. Restriction of motion to a region is an inherent characteristic of the third class. This directly correlates with its effect on scene coherence.

The two kinds of *rotation* differ in the shape of the affected information object and the rotation centre as sketched in Figure 3.7:

1. The rotation centre is placed at the midpoint of the affected object. In case of a regular shaped object, local effect on scene coherence results. In case of an irregular shaped object, the effect is regional.
2. Rotation around an arbitrary point with regard to the object results in either regional or global effect on scene coherence.

As motion by *structural change* is applied to the surface of an object only, its effect is of local nature. The information mural combines oscillating and structural motion techniques. However, the mural algorithm applies motion only to individual information objects or groups thereof. Thus, global effect on scene coherence is avoided.

Dynamics by variations of *hybrid rendering styles* provide a local effect on scene coherence. This is due to the principle of rendering styles: These are always isolated to the objects they are applied to.

The effect of *distortion histories* depends on the underlying distortion techniques. For the zoom used here, scene coherence is effected either regional or globally. Alternative distortions possibly restrict this effect to a region only.

The results of this classification will be picked up in the next chapter. Thereby, the different effects on scene coherence will be used as part of a script layout. Concrete scripts which are materialised on the basis of this layout are to be used for parameterising and controlling dynamics.

3.7 Summary

This chapter provides an overview of exemplary dynamic presentation techniques. These techniques cover a broad range of ways to construct dynamics. First of all, the most intuitive approach of creating animations based on motion is presented. Thereby, two inherently different approaches for creating motions are addressed: oscillations affecting whole information objects and motion by structural change which influences only local areas of a specific object. Another approach to address dynamics is presented by variations of hybrid rendering styles. Thereby, no translation-based animation is used to achieve a dynamic presentation. The collection of dynamic techniques is further filled with a discussion of two approaches for dynamically presenting distortion histories: trace lines and a chewing gum.

The presentation of dynamics throughout this chapter is framed by two components: the notion of an illustration target function and a classification of dynamics based on their effect on scene coherence. The former provides the necessary context for an illustration to address a specific task and user goal. The latter will be picked up in the next chapter. Therein, the classification is used as part of a script-layout for parameterisation of dynamics. Such script may be used for instantiating a concrete illustration meeting the user's goals as defined by the illustration target function.

4 Temporal Control of Dynamics

For the purpose of controlling dynamics as motivated in Chapter 2, temporal constraints need to be complied to. These constraints address a set of various temporal characteristics. By defining seven requirements for temporal behaviour of any dynamic presentation, these characteristics are placed into a structured framework.

The main contribution of this chapter is the introduction of a temporal control function for parameterising dynamic presentations. As a basis of this function, a temporal model is presented which fulfils the defined requirements. In support of this model, an overview of already existing temporal models is presented. These models are evaluated and classified with regard to the requirements.

4.1 Requirements

The temporal control of dynamics is a manifold task. As motivated in the previous chapter, any use of dynamic presentations needs to be restricted in order to employ dynamics to their full potential. Technique-based parameterisation of dynamics is subject of the next chapter. Here, the focus is on temporal constraints.

In order to describe temporal behaviour, the notion of a *temporal entity* is introduced. Such entity represents a presentation unit associated with some temporal characteristic such as points in time or intervals. Thereby, a presentation unit describes some aspect of a presentation. An exemplary presentation unit is the emphasis of a specific object in a scene for a defined duration by use of a concrete presentation technique.

For temporal management of dynamic presentations in an illustration system, the following basic requirements can be derived:

R1: Comparison of events; In order to schedule the usage of presentation techniques, temporal entities need to be comparable. These entities label presentation actions, such as `present`, `start`, `stop`, and `fade`. Comparable events allow for these actions to follow each other or be combined to presentation composites. Any possibly derived comparison of intervals by consideration of bordering events is explicitly not subject of this requirement.

R2: Insertion of new events; In order to support input of system-external information in a presentation system, new events need to be inserted into the temporal presentation model. These events form external sources of temporal

information and include user interaction as well as interaction with external processes and databases, such as knowledge-bases, search engines, or fusions thereof. A specific challenge for the task of inserting new events into the temporal model is in the preservation of temporal constraints, orders, relations, and dependencies.

- R3: Relational operations on intervals;** In addition to the consideration of pure temporal events, intervals need to be manageable, too. Allen [1983] introduces the notion of thirteen basic interval relationships. These form the basis for the management of intervals with regard to the integration and evaluation of the dynamics stimulus window as presented in the previous chapter. Relational interval operations allow for an accordingly restricted use of concurrency of dynamic presentation techniques.
- R4: Merging of intervals;** Besides the introduction of new events and intervals to the temporal model, the reduction of the model's dimensionality helps to reflect the dynamics of presentation techniques. In case events are removed from the model, the affected intervals need to be adjusted. This adjustment results in merging at least two intervals in order to close the gained gap. Furthermore, the combination and merging of intervals helps to express the respective correlation of presentation techniques and presented information.
- R5: Discretisation of intervals;** Similar to the motivation for merging intervals in **R4**, some given interval might need to be divided into a set of intervals. This reflects the need to parameterise the dynamics of an individual interval as temporal entity.
- R6: Constrained concurrency;** Multiple dynamic presentation techniques may be employed simultaneously. In order to comply to cognitive restrictions as expressed in the previous chapter, any use of these techniques possibly needs to be subject to restricting parameterisation. The main focus of these restrictions is in limiting the concurrency of presentation techniques.
- R7: Preservation of temporal consistency;** A presentation using either static or dynamic techniques or combinations thereof is to be modelled consistently. That is, at any given point in time, the state of the presentation model needs to be defined. This state includes currently active presentations as well as scheduling of upcoming ones. The notion of consistency furthermore spans annotation information provided by all previous requirements.

These requirements form three basic categories: event-based requirements, those based on intervals, and structural requirements for global temporal control. The following subsections discuss these categories.

4.1.1 Event-based Requirements

Parts of the requirements target temporal events. This subsection discusses these requirements. Of specific interest is thereby the use of events as time steps in a temporal model. This discussion is accompanied by a reflection on the correlation of events as temporal modelling entities versus events as basis for presentation of specific data.

Handling of Events

The first two requirements specify handling of events. These events can be considered as points in time and are specifically not duration-based. This way, **R1** and **R2** are useful for specifying momentaneous actions in the temporal model and allow for user interaction.

Requirement **R1** describes comparison of events. Usually, any two points in time can be compared easily based on their time stamps. This requirement is listed to provide a means of validating various approaches of addressing singular events in different temporal models. Such models do not necessarily need to be based on points in time as their modelling basis. In case they do not, it will be discussed whether and how they address handling of events or use an interval-based modelling alternative instead.

Based on **R1**, an insertion of new events is required by **R2**. Specifically, this addresses handling of events which are not known prior to modelling a presentation. One source of such events is user interaction. Any interaction possibly invalidates the current state of a presentation's temporal model. This results in a need of updating the state if events are inserted.

Correlation to Event-based Feature Tracking

The notion of *event-based* presentations has recently been used for data-driven illustrations [Reinders et al., 2001, Post et al., 2003]. Thereby, events are detected in flow visualisation techniques. For time-dependent data sets, the evolution of features are described. Detecting and extracting specific events during an evolution allows to illustrate temporal characteristics of features.

The intention of this chapter is to develop a temporal model for controlling dynamic presentation techniques. In support of this, the requirements target specific aspects of such model. Thereby, the notion of *event-based* temporal entities is used in the tradition of temporal modelling [Allen, 1983, 1991, Freksa, 1992a, Bettini et al., 2000, Artale et al., 2001].

Both uses of the notion of events differ in their underlying intentions. On one hand, *event-based* presentations target illustration of selected time-dependent data out of a larger data set. On the other hand, *event-based* temporal modelling tar-

gets specification of time-related behaviour in support of an illustration. These two event-based use cases coexist. For the context of this work, the notion of events for temporal modelling is used. Thereby, events serve as a fundamental key of the model requirements discussed here.

4.1.2 Interval-based Requirements

Requirements **R3** through **R5** are related to modelling of intervals. First of all, the set of reference intervals as defined by Allen will be discussed. This describes the basis for **R3**. The discussion is followed by reflections on handling interaction and its influence on determinism of temporal models. This is implicitly contained in **R4** and **R5**.

Interval Relations

Allen [1983] introduces the notion of thirteen reference intervals for the description of temporal relations between two entities *A* and *B*: *A* happens entirely before *B*, *A* meets *B* (which means that *B* starts at the same temporal instant that *A* ends), *A* overlaps *B* (where the begin of *B* happens before the end of *A*), *A* starts *B* (in contrast to the meet-relation, *A* and *B* now start at the same time), *A* occurs during *B*, and *A* finishes *B* (that is the end of *A* results in an simultaneous end of *B*). These six relations may be reversed and an additional similarity relation representing *A* equals *B* completes the set. In order to clarify these relations to some degree, they are illustrated in Figure 4.1.

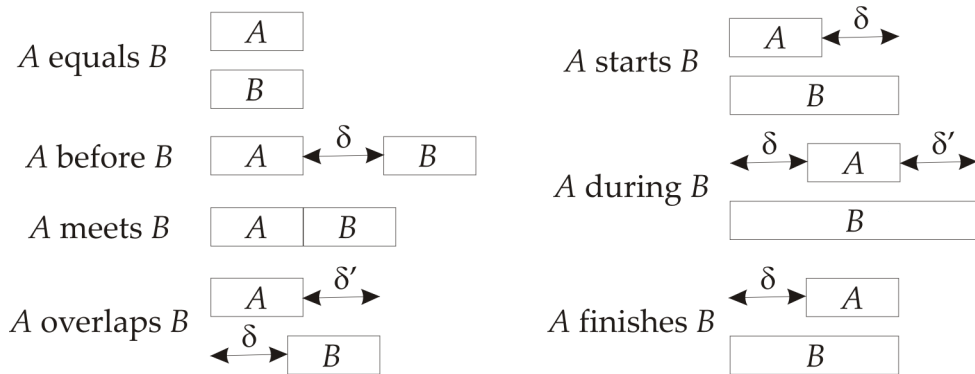


Figure 4.1: Allen’s set of temporal relations. With the exception of *equals*, all relations can be reverted. Overall, this results in thirteen interval relations being defined.

These relations are based on the exclusive consideration of time as intervals. Singular time points are not considered but modelled as intervals with duration $t < \epsilon_{time}$ where ϵ_{time} specifies a lower interval threshold for respecting events. The

relations introduced by Allen are *complete* in the sense that they allow the description of all possible relations of two given intervals. Thereby, relative time specifications are used exclusively. Modelling of absolute times is avoided. In order to ease modelling of the variety of relations between intervals, Allen uses his notion of so called *reference intervals*. These are used for the description of groups (clusters) of intervals, whereas the temporal dependencies in each of these groups is already calculated completely. As a side effect, these reference intervals are usable for the construction of object hierarchies and controlling propagation therein. This eases the representation of complex temporal models.

This model is picked up by Freksa [1992a] who expands it by the notion of *Semi-Intervals*. These are characterised by start- and endpoints only. For these interval borders, the relations $>$, $=$, and $<$ are defined. These relations are transitive. An additional restriction is put on them by constraining the begin of an event (interval) to always take place before its end. Two relations are therefore satisfactory in order to describe the relation between two arbitrary intervals. The equality relation ($=$) is derived as a combination of both. Furthermore, this model allows to describe the *complete* temporal knowledge about a set of intervals using a more simplified notion compared to Allen.

Freksa [1992b] picks up this notion, which is defined as *conceptual neighbourhood* and extends it to spatial knowledge. Orientation information is thereby limited to relative positions without including the degree of variation of a given position according to its orientation. As an example, relations specifying left/right or front/back orientation are part of this model, whereas relations expressing »too much on left/right site« are not.

Interaction and Determinism of Temporal Models

Handling of interval flexibility as described by requirements **R4** and **R5** is reasoned and motivated by a variety of system characteristics. Freedman et al. [1996] present a two-folded classification of temporal systems: time-driven and interrupt-driven systems. This distinction is based on the comparison of an exemplary telecommunication application and computerised control systems. Common to both is the »integration of temporal specifications into data and control flow« [Freedman et al., 1996]. Interrupt-driven systems represent some class of temporal pushing behaviour. New temporal information is provided on a per-availability basis. Time-driven systems are characterised by temporal determinism being employed in a *periodic* way where physical constraints determine the frequency of evaluating temporal inputs and sending out temporal control events. The specific kind of these physical constraints is defined by the concrete operating environment. Examples include sensors sending out signals or output sampled from a computing operation. Timing requirements are thereby associated with reactivity regarding detection and processing of alarms such as interrupts or sensor signals.

Fleischmann et al. [2001] present an internet media lab system for supporting knowledge discovery in mixed realities composed of awareness, memory space, and knowledge discovery. Their media lab consists of a set of *categories* providing access to various information storage areas. This allows for provision of collaborative information pools. The categories are either predefined (which results in a static set of categories) or are obtained by working with the alternative »Semantic Web« interface. Access to these categories is provided by a time line interface. Thereby, the time points act as access points for the categories.

Wahl and Rothermel [1994] analyse the usability of temporal models for *interactive* multi-media presentations. They conclude that modelling temporal behaviour of individual presentation components cannot be sufficiently supported by exclusive use of a time line as long as interactivity of the presentation is desired. This is reasoned in the requirement of a complete pre-modelling of all events in case of a time line, which correlates to a deterministic model. But, interactive systems are intrinsically characterised by not conforming to this model. Therefore, alternative models are required in order to support modelling of nondeterministic temporal behaviour.

A distinction between *internal* and *external* nondeterministic events is introduced by Santos et al. [1999]. Even though their work is presented in the context of temporal consistency of hyper-media documents, the idea can be applied for the context of this work, too. Internal events correlate to temporal relations in the geometry-based illustration model. External events are introduced by means of user interaction. This distinction proves useful for the designation as well as for solving temporal inconsistency. Inconsistencies caused by internal events are to be solved by a temporal evaluation entity of the system. In case an external event causes a temporal inconsistency, the latter might not be solvable. This is due to the unpredictability of external events such as user interaction. A solution for this challenge is in guaranteeing temporal determinism by the temporal evaluation entity. This is achievable by conforming to the above requirements **R4** and **R5**.

4.1.3 Structural Requirements

Requirements **R6** and **R7** are classified as being of structural nature. That is, both do not describe specification of temporal entities directly. Instead they address contextual constraints to be put onto the modelling. Whereas **R6** describes concurrency and limitations thereof, **R7** requires consistency of the temporal model.

Concurrency

Concurrency occurs when presentation techniques in an illustration system overlap in time. This is also referred to as parallelism. Fundamentals of dynamics perception are discussed in Section 2.1. These include limits of the cognitive system with

regard to multiple, simultaneous dynamics patterns.

As it is an illustration system's task to communicate information, these limitations need to be respected. This directly leads to requirement **R6** addressing concurrency. A specific emphasis of **R6** is in constraining concurrency. That is, a temporal model has not only to allow for concurrent presentations but also to provide some means of limiting parallelism. This ensures to comply to cognitive restrictions and model a presentation that meets its communicative goal.

Temporal Consistency

Consistency of temporal entities needs to be ensured for the purpose of defining a temporal model for dynamics presentations. This is expressed by requirement **R7**. The notion of temporal consistency as introduced by Courtiat and de Oliveira [1996] describes matching *begin* and *end* pairs in a reachability graph. The description and definition of this graph and its consistency is to be supported just as much as ways of validation. Other ways of discussing consistency make use of a notion of temporal coherence [Lascares and Oberlander, 1993, Yang and Das, 1994].

Due to the restrictions of computational systems, the lack of infinitely computation and communication resources leads to the possible violation of constraints between status in an interface. Dix and Abowd [1996] characterise this discrepancy as *temporal incoherence*. They recommend the use of a tolerance interval which is to be used at system modelling time. Using this interval allows to regard temporal constraints in a flexible manner. This is achieved by placing constraint entities in a temporal frame defined by the tolerance interval. Constraints are then transferred from the entity to the interval, which allows to overcome possible constraint violations.

Bourdev [1998] defines and uses the term *coherence of non-photorealistic rendering*. He thereby uses a combination of *temporal coherence* and *arc-length coherence*. Temporal coherence in this context is defined by disallowing distracting trembling of silhouette strokes over frames. The arc-length coherence describes maintaining of a constant period of repetition of a pattern in a given stroke. Both goals are mutually exclusive.

The notion of temporal coherence is also referred to as *frame-to-frame coherency*. Masuch et al. [1998] use it in the context of line drawings for illustrative purposes. A parameterised line model is presented for the purpose of preserving frame coherent animations of characteristic line deviations. A focus of the rendering parameterisation is in the reconstruction of line segments in order to achieve the desired coherence. Modelling and automatic derivation of temporal events is not specifically considered.

For the remainder of this work, the term *temporal coherence* describes the use of dynamic presentation techniques according to temporal constraints defined by the dynamics stimulus window as presented in Section 2.2.

4.2 Classification of Temporal Models

The following subsections present existing approaches for temporal modelling. For this purpose, a task-oriented presentation of the models is chosen. For each presented model, its principle is outlined and its potential for fulfilling the aforementioned requirements is sounded. In order to allow for a comparison of the presented models, the notion of *temporal entities* is used as basic modelling elements. These entities possibly correspond to dynamic presentation techniques that are to be scheduled in an illustration system. For the individual models presented, temporal entities might specify different targets, depending on the specific context at hand.

The contribution of this section is in the presentation of an overview and classification of temporal models. A specific and unique accentuation is thereby put on its appropriateness for constraining graphical presentation and illustration systems.

4.2.1 Time Lines

As outlined in Section 2.3, time lines are a classical technique for the representation of temporal data. However, a time line model can also be used for the construction of temporal models.

A basis for temporal modelling by time lines is presented by Keim [2002]. Even though his work presents the representation of multidimensional data by pointed plots, the author introduces the notion of one-dimensional data typically being temporal data. Each time point is thereby associated with one or more data values. Individual events make up the temporal model. The same notion of single time points as model basis consolidates time lines. This is already used by Leith and Cunningham [1997] who present time lines as an »intuitive modelling data structure.« The authors use it for modelling readings of simple sentences in order to analyse linguistic categories of the subjective text.

A concrete definition of time lines as a temporal model basis is presented by Jónsson and Frank [2000]. They introduce the time line as a combination of state variables and intervals, where the evolution of a state variable is expressed as a sequence of intervals, connected by temporal constraints. Smith et al. [2001] use a similar notion and define the time line by a series of events and required system responses. The latter represent reactions on events. The requirements of these reactions results from the application of the model. The authors use the time line model as an interaction key of an editor for temporal model requirements. This editor is used for the formalisation of temporal models forming the basis for automatic model checkers such as *Spin* as presented by Holzmann [1997].

Kosara and Miksch [2001] present the *AsbruView* system and enrich use of time lines to 3D. The third dimension is used to model and convey further information. Time axis are employed without a scale but for indication of ordering directions.

Temporal constraints are employed by means of *time annotations*. These are composed of: earliest starting shift, latest starting shift, earliest finishing shift, and latest finishing shift. Furthermore, a minimum and a maximum duration might be specified. This allows to model sequences, parallel entities denoted as »some-together« relation, any-order, and cyclical relations.

The comparison of exact time steps is easily derived from the events' respective position on the time line. This fulfils requirement **R1**. New events may be inserted into the time line model as long as their time stamps are known exactly. The insertion of events without exact time stamps is only supported in a handicapped way: In order to handle inexact events, an interpolation scheme may be developed. A temporally fuzzy event e_f is thereby placed just between two existing events e_a and e_b , as long as $e_a \leq e_f \leq e_b$ holds. Thus, requirement **R2** is supported in part. In order to handle intervals in the model, their respective beginning and ending events are used. This loses the notion of the interval as such as long as no additional meta data is stored as well. **R3** is affected as not being fully supported.

The notion of time lines for temporal modelling is easy to grasp as of its simplistic nature. This simplicity reasons the main disadvantage of the concept, though. Time lines are of restricted nature and do not provide a flexible modelling basis. Only exact time stamps can be handled. Therefore, only exact points in time that are known can be modelled. A consideration of uncertain and possibly undefined parts of an event set is not provided. This prevents support of requirements **R4** and **R5**, that is interaction support by non-determinism.

Concurrency in its simplest form is supported by time lines via modelling of a set of parallel temporal axes. Any way of expressing constraints to this parallelism cannot be expressed, though. **R6** is only partially supported.

Temporal consistency as denoted by requirement **R7** is not supported by a time line based temporal model as well. As branching of a time line is not allowed, *begin* and *end* pairs with regard to reachability do always match. Therefore, the notion of a possible *inconsistency* in this regard does not apply.

A Reference Scenario

In order to illustrate the scheduling of hierarchic composition compared with a time line model, a concrete assignment of a set of dynamic presentation techniques $\mathcal{D} = \{d_1, d_2, d_3, d_4, d_5, d_6\}$ is presented in Example 4.1.

Example 4.1 The set $\mathcal{D} = \{d_1, \dots, d_6\}$ used in Figure 4.2 may be instantiated by the following use of concrete dynamic presentation techniques combined with classic presentations:

- d_1 rotation,
- d_2 change in rendering style,

- d_3 static realistic rendering,
- d_4 motion-less but otherwise unrestricted,
- d_5 oscillating translation, and
- d_6 static non-photorealistic rendering.

□

This shows the dynamics-enhanced presentation of multiple objects. Use of a motion technique d_1 for the duration of $t_1 \rightarrow t_2$ might possibly effect the same object as the use of d_4 in the time-frame of $t_2 \rightarrow t_3$. It is to be noted explicitly that d_4 may be decomposed further in order to define its presentation in some more detail. This allows for the effected object to be presented dynamically for the complete duration $t_1 \rightarrow t_3$: first by d_1 , than by d_4 .

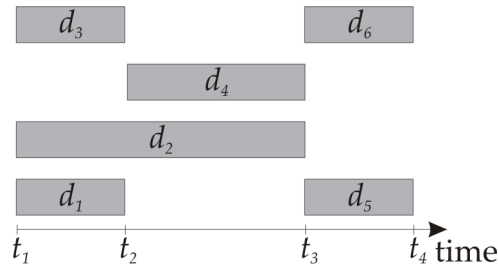


Figure 4.2: Time line representation of the exemplary reference scenario. This scenario represents the temporal composition of a dynamic presentation as described in Example 4.1.

4.2.2 Graph-based Models

Different approaches exist for temporal modelling based on graphs. A first overview is presented by Allen [1991] who presents three types of graphs:

1. graphs using exact time stamps,
2. constrained propagation graphs, and
3. duration-based graphs.

All three approaches form the basis of further temporal graph models. Two of these are presented: Firefly and Hyperstories.

It is to be noted that other forms of temporal models make use of graphs, too. Indeed, these models employ graphs for representation purposes whereas modelling and fulfilment of the listed requirements is achieved by other means. An example for such a case are petri nets which will be discussed later on.

Graph with Exact Time Stamps

For a model based on *exact time stamps*, two dates are to be specified for each temporal entity: its earliest possible starting point as well as its latest possible starting point. In case that only the linear order of events is known, pseudo dates can be used in order to comply to the model constraints. Figure 4.3 shows the basic model structure whereas a concrete example of using the model in order to schedule a set of events is shown in Figure 4.4.

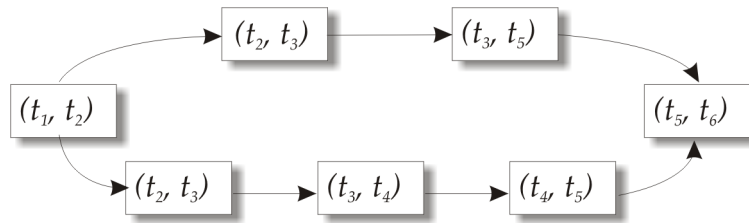


Figure 4.3: Temporal modelling based on exact time stamps. Each block represents a temporal entity characterised by its earliest and latest possible starting time.

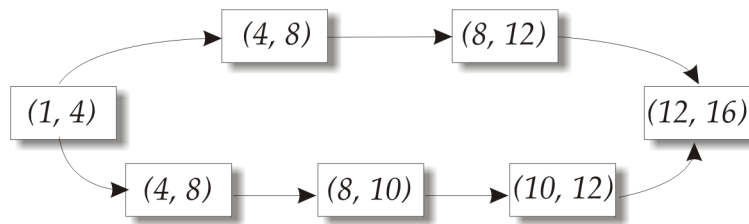


Figure 4.4: A concrete allocation of the scenario from Figure 4.3 showing the temporal model based on exact time stamps. The numbers express time stamps in an abstract notion.

This graph naturally supports requirements **R1** and **R2**. The model's basis is formed by events. Comparison is explicitly provided by placing the individual temporal entities in the graph. The same holds for inserting an event into the graph.

Handling of temporal intervals is not supported by this model. Even though some interval information may be derived by relating events to each other, this information is not handled explicitly in the graph. Therefore, requirements **R3** through **R5** are not supported.

Concurrency in the sense of requirement **R6** is supported in part. Modelling of parallel temporal entities is possible by simply adding branches to the graph. Any constraints to be applied to concurrent branches cannot be expressed explicitly, though.

Support for requirement **R7** is provided as much as for all other graph-based models as well. As the graph is of acyclic nature, its temporal state is always consistent. This fulfils this requirement entirely.

Graph Based on Constrained Propagation

Constrained propagation describes a graph-based modelling approach that is based on temporal relations. The graph is defined by placing events at the nodes and temporal relations at the edges. The insertion of new events into an existing graph may result in the necessity for a complete reordering.

Figure 4.5 points out a snapshot of a constrained propagation graph during its modelling phase. The time points shown in the figure directly correlate to their counterparts of the reference scenario as shown in Figure 4.2. At this stage, the relations between the shown time points are still very flexible. Introducing the missing point in time t_2 results to the graph is shown in Figure 4.6. Therein, all time points are placed correctly with respect to each other.

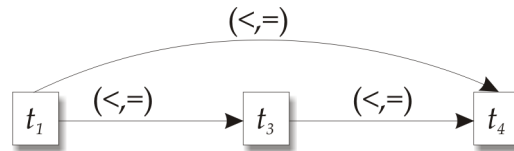


Figure 4.5: Temporal modelling of the reference scenario from Figure 4.2 by use of constrained propagation. Temporal relations are used in order to express dependencies between model entities. (based on Allen [1991])

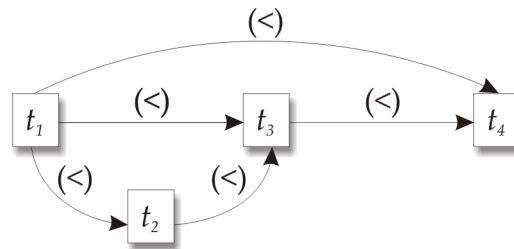


Figure 4.6: Temporal modelling by constrained propagation. In contrast to Figure 4.5, the presentation at t_2 has been inserted into this model. This results in a need to update some of the specified relations. (based on Allen [1991])

As the model of constrained propagation is built on relations between events, requirement **R1** is supported well. Inserting new events as expressed by **R2** possibly leads to modifications of relations in large parts of the graph. Any deduction of intervals from the model proves even more challenging. A precise notion of intervals is not provided. However, as Allen [1991] points out, alternative approaches for modelling constrained propagation based on intervals instead of events are possible. Thereby, interval representation is expressed by point-based constraints. On one hand, this provides support for fulfilling requirement **R3**. On the other hand, this approach does not specifically meet complete handling of intervals in the sense of **R4** and **R5** as splitting or merging sets of constrained events is not provided.

This graph does not specifically provide a means of specifying concurrency. Specifically constraining concurrency as expressed in **R6** cannot be modelled. Support of requirement **R7** basically follows the notion of exact time stamps above. That is, this requirement is met due to the graphs inherent nature of fulfilling consistency as long as it is of acyclic nature. For constrained propagation, this is always the case [Allen, 1991].

Duration-based Graph

In contrast to both former approaches, the *duration-based* temporal modelling approach, as outlined in Figure 4.7, makes use of intervals as model basis. The single nodes in the graph still contain time stamps: the earliest possible starting time as well as the latest possible starting time for any given event in the system. These time stamps do not make up the model basis, though. They are derived from intervals defining event durations. These durations are placed at the graph's edges and denoted as intervals I in Figure 4.7. The graph is constructed as a PERT network [Stoyan and Daley, 1984]. Partial ordering of events is maintained in an acyclic directed graph. The first and the last entity in this graph are explicitly specified, thus ensuring fulfilment of requirement **R7**.

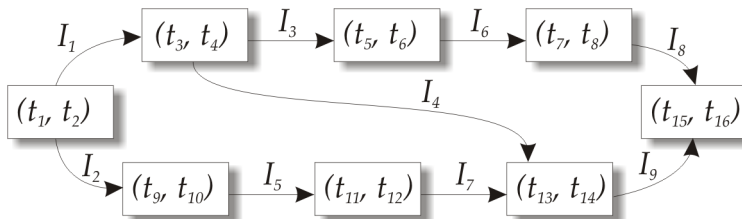


Figure 4.7: Exemplary PERT network representing temporal modelling by duration-based interval specifications. (based on Allen [1991])

Figures 4.8 through 4.10 present an exemplary use of duration based modelling. The first node in the graphs represents the initial temporal entity. The earliest possible starting time is set to 0. As the models execution is to be started as early as possible, the latest possible starting time is set to 0 as well. All other time points and durations are left open. This is expressed by Figure 4.8.

The edges in the graph are assigned with respective durations of executing the temporal entities of the model. Figure 4.9 shows the graph with all durations specified. Using these duration values, all earliest possible starting times of the temporal entities can be derived as shown in the figure. The latest possible starting time for the last entity in the graph is set to its earliest possible counterpart. Similar to the first entity this is due to the desired effect of ending the graph's execution as early as possible.

Based on the complete knowledge of the last entity's temporal settings, the remaining graph can be calculated. The result is shown in Figure 4.10. For each tem-

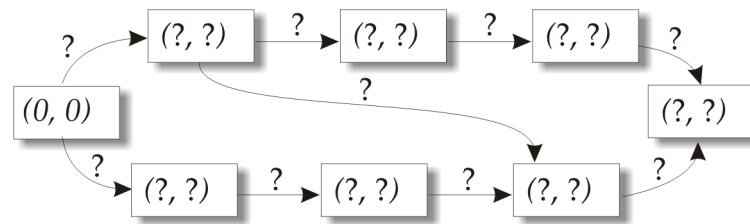


Figure 4.8: The PERT network from Figure 4.7 with still no durations being specified. The earliest possible starting times are set to 0 expressing the motivation to start as soon as possible.

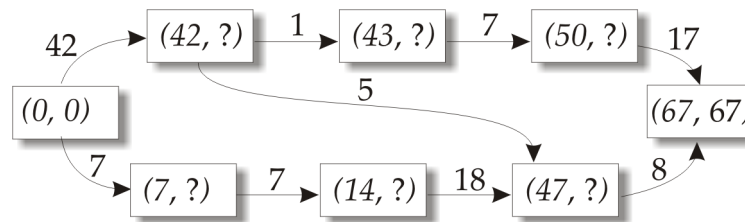


Figure 4.9: The exemplary graph from Figure 4.8 with durations specified. The earliest starting points are derived.

poral entity in the graph, its respective earliest and latest possible starting points in time are now known. Using these intervals for scheduling allows to comply to the desired temporal model.

As the example points out, events are not specified explicitly but derived from the respective durations of the temporal entities. This still supports requirement **R1** as the derived points in time can be compared by the respective positions of entities in the graph. Inserting new events as required by **R2** is not supported.

Handling of intervals is the model’s principle basis. Requirements **R3** through **R5** are all fully supported. An insertion of new duration elements into the graph only affects all or part of the entities that temporally follow the new entity. An explicit handling of concurrency is not provided by the model. Branches in the graph are not necessarily related to each other which prevents constraints as defined in **R6** to be modelled.

The temporal graph representing the reference scenario by duration-based tem-

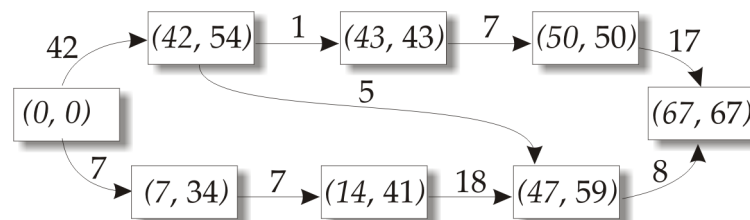


Figure 4.10: Complete calculation of the exemplary graph from Figure 4.9. This includes all earliest starting points as well as all latest possible starting points in order to meet the presentation constraints.

poral modelling is shown in Figure 4.11.

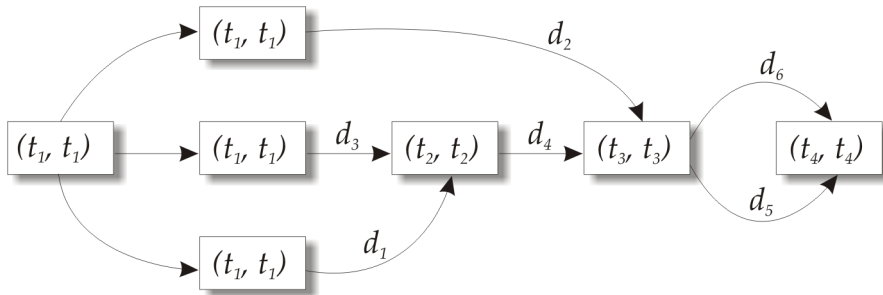


Figure 4.11: Using a duration-based temporal graph for modelling the reference scenario of Figure 4.2.

Further Temporal Graph Models

Firefly: The temporal graph provided by the *Firefly* system presented by Buchanan and Zellweger [1992] consists of three modelling entities: square nodes, circular nodes, and edges. Square nodes indicate starting and ending times of temporal entities. Circular nodes provide for intra-object synchronisation. That is, they provide for a granularity refinement compared to square nodes. For this purpose, circular nodes may be placed in between any two square nodes which discretises the respective interval (fulfilling **R5**). Edges finally combine all nodes and provide temporal relations between events. The relations are defined by annotating edges with constraints. The exemplary *Firefly* graph notation for the reference scenario is shown in Figure 4.12.

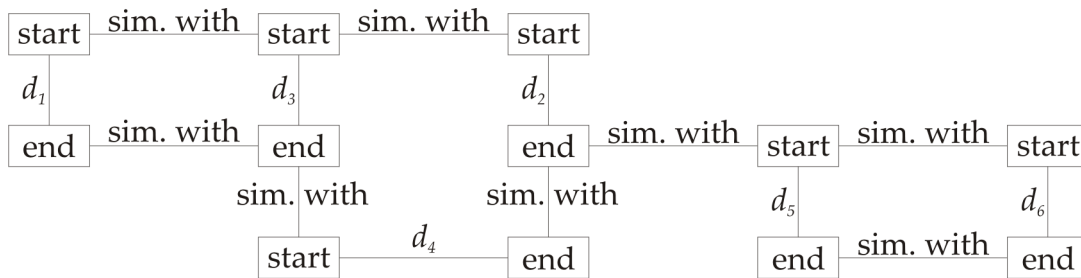


Figure 4.12: Firefly's graph notation for the reference scenario of Figure 4.2.

Two types of constraints may be used in a *Firefly* model: temporal equalities and temporal inequalities. Equality constraints either specify that two events happen simultaneous with each other or that both events differ by a fixed temporal distance. That is, one event occurs with a distance of t before the other. The

simultaneous with constraint provides concurrency modelling as noted by requirement **R6**. Any explicit restriction of a maximum number of parallel components cannot be expressed in the model. Inequality constraints open the deterministic limitation of a fixed temporal specification. This support for indeterminism includes two expression classes: preceding one event by another one by a fixed time and limiting this relation by an additional upper limit. Even though these constraints underpin support for requirement **R1** as it is inherent to graph-based models, they also restrict compliance of **R2**. Insertion of new events possibly invalidates the graph and might therefore require the graph's reconstruction.

Relations on intervals, as required by **R3**, are not supported directly by *Firefly* graphs as these are based on the modelling of events as points in time. Similarly, merging intervals (**R4**) needs to be handled indirectly by merging respective interval borders. In this case, constraints need to be re-evaluated. Buchanan and Zellweger [1993] present an analysis tool for consistency checking of a *Firefly* graph. Even though this does not solve any inconsistencies possibly introduced by **R4**, it helps to identify problem-causing model entities. This directly influences support of the temporal consistency requirement **R7**.

Hyperstories: A variation of the *Firefly* model is presented by Kim and Song [1995]. Their *Hyperstories* system uses the concept of elastic time for the scheduling of temporal entities. These entities represent elements of a story, such as text or video. Such entities are to be arranged in a flexible manner for the purpose of meeting a scheduling goal that fits all available entities into a given time range.

Instead of events as a model basis, this system is build on top of intervals. Compared to the *Firefly* model, this directly affects an almost opposite compliance to requirements **R2** through **R5**. Introduction of new *events* into the model (**R2**) is not supported directly. Instead, they need to be derived from intervals. The interval requirements **R3**, **R4**, and **R5** are met because central key characteristics of the model are the fulfilment of Allen's interval relations (**R3**), the merging of intervals (**R4**), and the discretisation of intervals (**R5**) for the purpose of flexible scheduling of temporal entities.

The handling of concurrency as defined by requirement **R6** in the *Hyperstories* model is similar to the *Firefly* model. Different branches in the scheduling graph represent different parallel temporal sequences. This fulfils the requirement to some extent. Though, just as in the *Firefly* model, no notion of limiting the set of parallel sequences is available in the model. Requirement **R7** is met as the graph provides complete reachability as an intrinsic characteristic.

4.2.3 Petri Net Models

The notion of petri nets for the modelling and specification of a broad family of reactive systems is introduced by Petri [1962]. An overview of possibilities to utilise temporal actions of a user by employing petri nets is presented by Palanque and Bastide [1996]. Petri nets are defined as a graph that connects a set of states. Transitions represent actions that may move from state to state along the graph's edges. By characterising states with temporal information, intervals and concurrency can be handled.

Formally, petri nets form a state chart. A generic notion of state charts is presented by Harel [1987, 1988]. Conditions for the temporal order of modelling entities can be formulated exactly by these charts. A state chart is defined as a tuple $(Q, \Sigma, \delta, q_0, F)$. Thereby, Q describes a finite set of states, Σ the set of input values, $q_0 \in Q$ a designated starting state and $F \subseteq Q$ the set of end states.

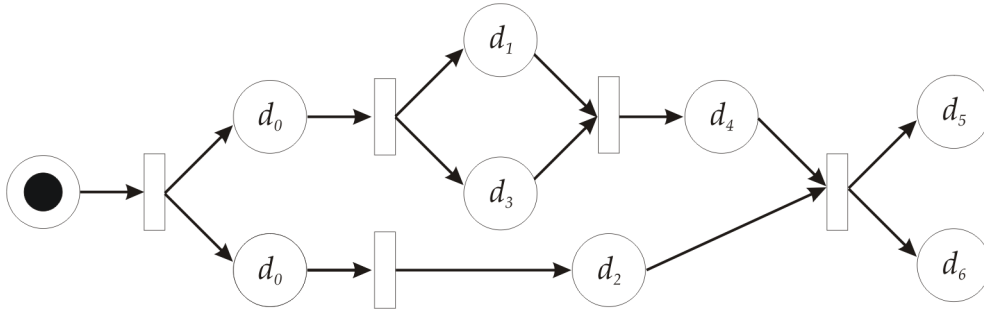


Figure 4.13: Petri net model for the reference scenario of Figure 4.2.

Petri nets are usually represented as state diagrams. Figure 4.13 presents such a diagram for the exemplary reference scenario. Therein, state transitions are expressed directly. Each state describes a temporal interval. The interval starts at the event of entering the state by a transition. Leaving a state ends its represented interval. As any transition only seizes one state at a time, all intervals are disjunct.

Petri nets easily fulfil requirement **R1**, as the comparison of events can be derived from the respective state placements in the graph. The insertion of new events is a natural process in a petri net. For any possible case of invalidating constraints and relations in the network, a solution can be derived [Keller et al., 1994]. Therefore, requirement **R2** is met.

Handling of the whole set of Allen's thirteen temporal relations as expressed in **R3** is supported by petri nets [Little and Ghafoor, 1990]. Any merging and discretisation of intervals as stated by requirements **R4** and **R5** can be achieved as long as no intermediate or branching states are affected. That is, for a set of states $S = \{s_1, s_2, s_3\}$, the merging of s_1 and s_3 is prohibited by s_2 if, and only if, $s_1^e \leq s_2^b \wedge s_2^e \leq s_3^b$. That is, the beginning of state s_2 happens later than the end of s_1 , and s_2 finishes before s_3 begins.

Concurrency and its constraints as required by **R6** need to be modelled explicitly by use of multiple transitions. In case of a *single* transaction floating through the petri network, the concurrency of two or more presentations cannot be represented by the temporal model. However, a set of extensions for petri nets has been presented that addresses concurrency constraints [Biberstein et al., 1997, Kiepuszewski et al., 2002, Little and Ghafoor, 1993]. Using such a model helps to fulfil **R6** in its entirety.

An explicit temporal modelling approach addressing true concurrency is introduced as *evolving concurrent object petri nets: Co-nets* [Aoumeur, 2002]. Such nets allow to model behaviour of information systems with a rewriting-logic based semantics. Object-orientation and true concurrency support is achieved by dividing the model construction up onto a set of three layers: data-layer, object-level, and meta-level [Aoumeur and Saake, 2000]. This decomposition of the modelling process allows to adjust to changes of the underlying system behaviour and therefore support requirement **R6** to its complete extent.

Requirement **R7** is met as long as the petri net is given as an acyclic graph. Then, the graph ordering represents temporal consistency. The case of cyclic petri nets is discussed by Allan et al. [1995]: If a petri net is cyclic, a state may possibly be reached through which the net has already passed. As the decisions made throughout the petri net execution are deterministic, the respective behaviour of the net repeats. Thus, any cyclic parts in the graph result in cyclic presentations. This invalidates **R7**.

4.2.4 Object-oriented Models

Defining temporal entities as a set of objects allows to model their scheduling by use of object-oriented design principles. A temporal scheduling scenario is composed by modelling the set of objects with relation to each other. Thereby, temporal order is achieved by means of object attributes. Classically, these attributes are defined as methods to be invoked on concrete object instances.

The following subsections present six different models for object-oriented temporal behaviour specifications. In the order presented, these modes are based on each other.

Interaction Diagrams

The concept of *interaction diagrams* is introduced by Jacobson et al. [1992] and Booch [1993]. Some practical use of them is described by the UML modelling specification [OMG, 2003]. Based on *event trace diagrams* [Rumbaugh et al., 1991] these diagrams are used in order to schedule the temporal behaviour of object-oriented system components during the system's design process. Thereby, methods as reaction to temporal events are encapsulated. These methods provide a means of complying to

requirement **R1**. As methods are generated and passed from object to object freely and in an unrestricted manner, requirement **R2** is also met.

The modelling of temporal behaviour and constraints of objects and their presentation methods is expressed graphically. An exemplary interaction diagram for modelling a dynamics-based presentation according to the reference scenario is shown in Figure 4.14.

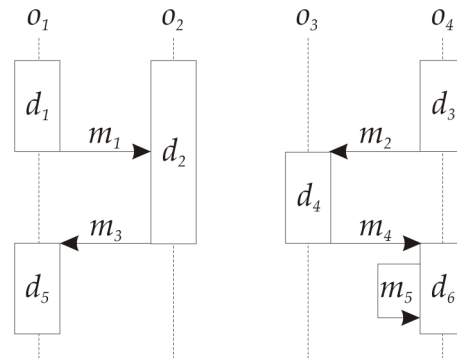


Figure 4.14: Temporal order of message passing between objects in an interaction diagram at the example of the reference scenario from Figure 4.2.

The dashed lines express instances of the system's time axis. All lines represent the same time axis. That is, no *distribution* of temporal behaviour is shown in the figure. Each axis shows the presentation of an object o_i over time. The rectangles specify the duration of an active dynamic presentation technique for the respective object.

The methods m_1 through m_5 , which are passed between the objects represent temporal synchronisation points. Any temporal entity might be modelled without an initiating method as shown for d_1 , d_2 , and d_3 . Sending m_2 at the end of d_1 results in a modified presentation of d_2 . This intra-object style modification is also possible to be caused by self-referential methods. An example is shown for d_6 which is changed during its life cycle by passing m_5 to itself.

The notion of intervals is derived from combinations of events that build interval borders. This allows to derive interval relations from event specifications and thereby supporting requirement **R3**. The merging of intervals requires to modify the set of available events. Support of such a model modification is restricted to object flexibility. That is, requirement **R4** is only met in case affected objects provide methods for handling such coarser temporal granularity.

As the modelled intervals correlate with objects in the diagrams, an arbitrary discretisation of intervals is not provided by the model. As an objects presentation and its underlying temporal structure might change to some degree, requirement **R5** is supported at least partially.

Concurrency of multiple actions is ensured by managing separate sets of events for different objects. However, a means of automatically constraining the sequen-

tiality is not provided. Requirement **R6** is therefore only partially supported.

Ensuring temporal consistency of an interaction diagram model is left as a modelling task. That is, no automatic validation of consistency is provided as required by **R7**.

Interaction Machines

Wegner [1997] extends the notion of interaction diagrams to *interaction machines*. The purpose of these machines is to advance from pure computing-based algorithmic programming approaches to interaction controlled computer systems.

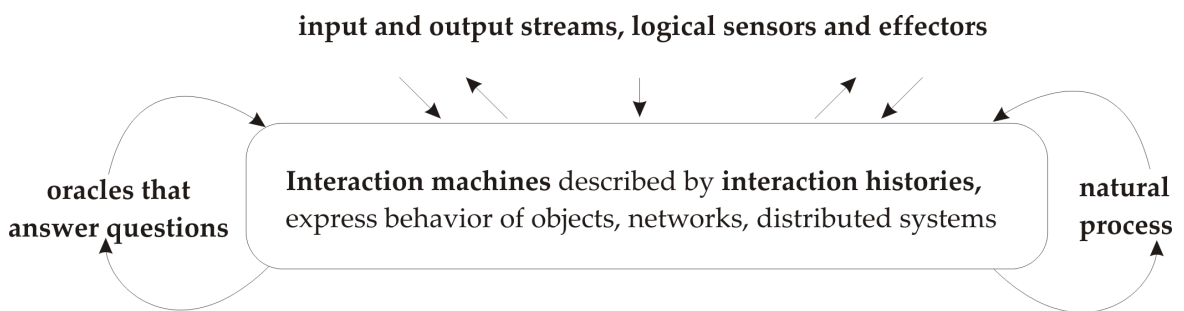


Figure 4.15: Interaction machines as defined by Wegner [1997].

Figure 4.15 shows the principle of interaction machines as presented by Wegner [1997]. This basically shows that the observable behaviour of these machines is characterised by *interaction histories*. Information about temporal actions of state changes in the machine are stored in these histories. Model entities that are temporally affected, like a specific emphasis of parts of a geometric model, are marked with time-stamped traces. Sets of these traces form the action intervals. Histories are capable of handling non-sequential time-stamp traces.

The temporal model of interaction machines follows the model presented for interaction diagrams with the exception of providing additional history information. Even though this helps in the process of modelling an interaction-based temporal scenario, it does not affect support of the requirement set **R1** through **R7**. Therefore, the requirement support presented for interaction diagrams holds for interaction machines as well.

Interval Diagrams

Weber [2000] extends both former models and introduces the notion of interval diagrams for the modelling and preservation of interval coherence in the sense of requirement **R7**. While concentrating on the integration of visual and non-visual user interfaces, the author introduces *hierarchic temporal intervals* in order to comply

to human perception of time. The hierarchic character of the model results from its handling of intervals. Even single events are respected as being intervals. Therefore, requirements **R1** and **R2** are not directly supported. Hierarchic combinations of individual intervals to *interval composites* form the temporal model. By reusing the interval relations from Allen [1983], qualitative and quantitative intervals and relations can be defined by the model. This fully supports **R3**.

While this does not extend the semantics of intervals compared to interaction diagrams or machines, it makes the distinction between single events and durations explicit. Each point in time possibly belongs not only to one interval but to a set of intervals. This allows to respect temporal constraints of finer granularity. Especially the possibility of a point in time to belong to an interval as well as to one of its sub-intervals is of relevance for controlling dynamic presentations. This allows to directly bind to the composite character of dynamic presentation techniques as outlined in Section 2.4. Furthermore, it provides interval merging and discretisation as required by **R4** and **R5**.

The nested character of the interval specification allows for the modelling of concurrent events. This only holds in case the upper limit of any given interval is specified as an interval as well. As this requires *forwarding relations*, a need for meeting Allen's *during* relation can be derived, whereby requirement **R6** is only fulfilled partially.

Hierarchic Composition

Herrtwich and Delgrossi [1990]¹ present modelling of temporal relations by use of a tree structure. This model is an extension to the standard document architecture ODA [ISO]. The documents which are subject of ODA are to be respected as temporal entities in case they are stamped with temporal characteristics and temporal relations between multiple documents are defined [Bertino and Ferrari, 1998].

The tree is composed of *timed objects* for the purpose of expressing temporal requirements for a multimedia presentation. Two kinds of timed objects are used: basic timed objects and composite objects. Basic objects are denoted by leaves of the tree. Composite objects are represented by internal nodes. The duration of a basic object may be specified either explicitly or implicitly. An explicit modelling is done by associating the object with a respective interval representing the object's presentation time. An implicit specification results from deriving the presentation interval from the intervals of the object's associated content portions.

This model does not provide for a specification of exact time points as required by **R1** and **R2**. Instead, *ordering* of multiple presentations is modelled with respect to each other. An exemplary hierarchic composition model representing the equivalent to the reference scenario is outlined in Figure 4.16.

¹ Referenced by Section 7 of Bertino and Ferrari [1998].

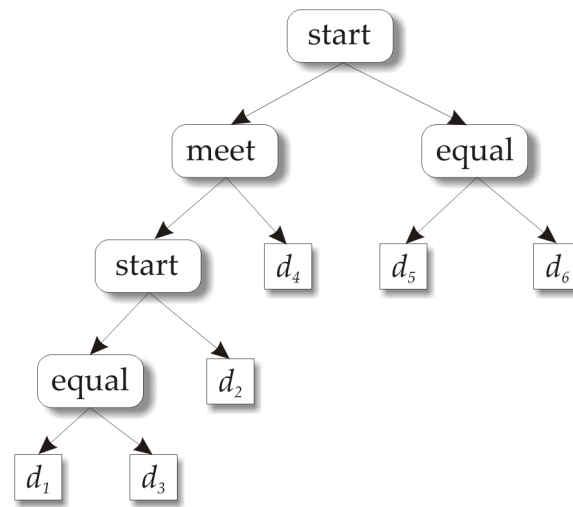


Figure 4.16: Graph showing the hierarchic composition model of the reference scenario of Figure 4.2.

Support of requirement **R3** is essential for hierarchic compositions. It is the set of interval relations that is used for model construction. Handling of interval merging and discretisation is not automatically provided by the model. Depending on the composition of relations used for all modelled intervals, both operations can be achieved manually, though. Thus, requirements **R4** and **R5** are supported in part.

Modelling of concurrency is not explicitly prohibited by the model. However, specific support for synchronisation extending the *equal* relationship is not provided. For this reason, an explicit constraining of concurrency is not provided which leaves only rudimentary support of requirement **R6**.

As all entities of the model are specified by a tree structure, consistency in the sense of requirement **R7** is supported well. The reachability criteria is met by travelling along any path from the tree's root node to the respective leaf.

Path Expressions

Other extensions to ODA for the handling of temporal entities are presented by Hoepner [1991]. In contrast to concentrating on modelling temporal logics, these extensions base temporal scheduling on rearrangements of an object layout structure.

Two basic concepts form this model: actions and path expressions. An action represents a temporal entity or, more precisely, its presentation. Path expressions were introduced first by Campbell and Habermann [1974] and are now used for the specification of synchronisation constraints.

Temporal scheduling is achieved by combining actions with path operators. Assuming that A and B are temporal actions, the ordered list of operators that may be used for this purpose is: *parallel-last* ($A \wedge B$) denoting a simultaneous start

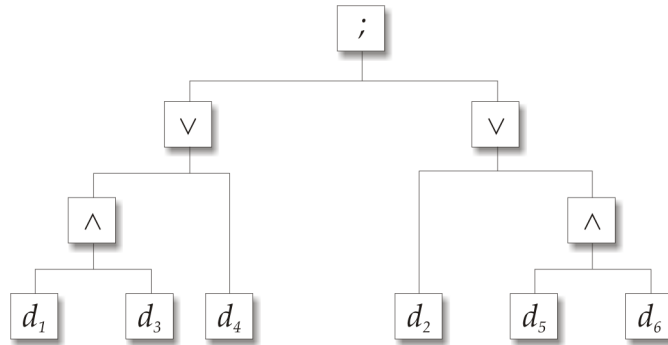


Figure 4.17: Example of using path expressions for modelling the scenario from Figure 4.2.

of A and B where the ending of the composition is determined by the last ending of either A or B ; *parallel-first* ($A \vee B$) denoting the same operator as before with the difference that the first ending of either A or B determines the composite's ending; *sequential* ($A; B$) denoting a start of B after A has ended; *selective* ($A|B$) stating that either A or B may be executed; *repetition* (A^{i*}) noting how often (i) an action might be repeated; and *concurrency* ($n : A$) resulting in n executions of A simultaneously. An example of using these operators for defining path expressions is shown in Figure 4.17 for the reference scenario from Figure 4.2.

Similar to the hierarchic composition model above, the path expressions operators do not provide information about single events or relations thereof. Requirements **R1** and **R2** are therefore not met. Relational operations on intervals (**R3**) are a central characteristic of the model, though. In order to provide for merging as well as discretisation of intervals (**R4** and **R5**), any preservation of the validity of relations between all affected actions needs to be handled manually as it is not provided by the operators. Even though an explicit operator is provided for handling of concurrency (**R6**), an automatic constraint concurrency mechanism is not available. Consistency in the sense of requirement **R7** is always preserved by the model.

Active Objects

The active objects model is an object-oriented model based on time lines [Gibbs, 1991, Gibbs et al., 1993, 1994]. Objects are denoted as being *active* because they may activate methods even in case no causing event is sent to the object. Therefore, each active object represents a temporal entity. The methods defined on these entities control temporal activities: *start*, *stop*, *pause*, and *resume*. These methods denote actions on individual objects. For the purpose of defining a whole model schedule, a set of further scheduling method groups is defined: coordinate management methods, composition methods, and synchronisation methods.

Coordinate management methods hold responsible for transitions between differ-

ent time granularities used throughout the model. This allows to specify temporal behaviour of a single entity with another granularity than behaviour of the overall model. Coordinations in this sense support requirements **R1** and **R3** as events and intervals of different granularities need to be put into relation to each other during a coordination process.

Composition methods allow to merge different temporal entities and form composite entities. Thereby, these entities are placed into a temporal sequence. This directly affects support of requirement **R4** which addresses the merging of intervals. This support is only of partial nature, though, as only sequential intervals may be merged. A reverse operation of construction of new intervals by discretisation as stated by requirement **R5** is not provided.

Synchronisation of multiple temporal entities is supported. This handles concurrency in the sense of **R6**. The overall number of simultaneously active entities is not handled explicitly, though.

Support of **R2** is provided by inserting new objects into the model. As no overall evaluation entity holds responsible for consistency validation throughout the model, requirement **R7** is not met. Figure 4.18 presents the reference scenario from Figure 4.2 modelled with active objects.

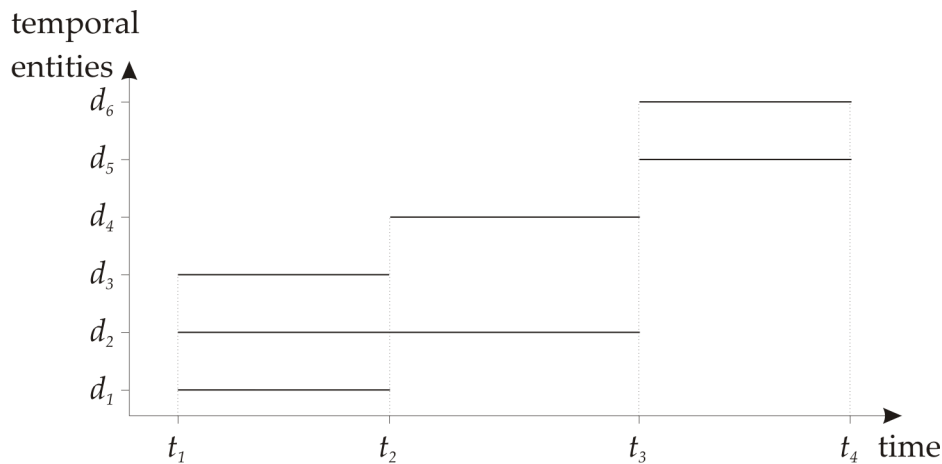


Figure 4.18: Time line with *Active Objects* modeled on the basis of the reference scenario shown in Figure 4.2.

4.2.5 Temporal Logic

Instead of using temporal entities as the principle modelling basis, temporal logic uses a state-based approach to modelling temporal behaviour. This is achieved by defining temporal languages or temporally extending existing ones. Two exemplary approaches are presented and their respective fulfilment of the temporal

requirements discussed. Inherent group characteristics of temporal logic with regard to these requirements are not specifically outlined as different languages may address handling of temporal logic individually. These discussions are followed by a brief overview of a set of temporal models based on grammars. Such models aim at introducing a language targeting exclusive modelling of temporal behaviour.

TCSP as Exemplary Language Extension

Based on the language of *Communicating Sequential Process*, Davies et al. [1991] present TCSP as *Timed Communicating Sequential Process*. This language aims at specifying temporal scenarios for synchronisation purposes. A set of operators is used. These operators explicitly depend on time: *termination*, *skip*, *timed delay*, *timed prefix*, *external choice*, *internal choice*, *timeout*, *sequential composition*, *relabelling*, *binary parallel composition*, *asynchronous parallel composition*, *recursive program*, *input from channel*, *synchronised parallel composition*, *hybrid parallel composition*, *event prefix*, and *interrupt*.

The modelling basis of TCSP are `time units`. Functioning as events, these units directly fulfil requirements **R1** and **R2**, as relations on them are defined by the `timed delay` and `prefix` operations as well as combinations thereof. In support of **R2**, new events can always be introduced to the model.

Intervals are defined out of event compositions. Operations on these compositions correlate to those on events. Ates et al. [1996] show that TCSP is powerful enough in order to fulfil all of Allen's interval relations. Using event compositions recursively on intervals defined by compositions helps to support requirement **R4**. That is, intervals might be merged by treating them as compositions of either events or other intervals. This also supports requirement **R5** to some extent. The composition of an interval can be split up again. Such splitting basically results in an event discretisation. This splitting is restricted to the finest granularity formerly used for any given interval.

Concurrency of time units is provided by `parallel composition` operations. However, inter-object synchronisation is not specified by TCSP. This results in only limited fulfilment of requirement **R6**.

Theoretically, any language-based temporal model is capable of providing automatic consistency validation and therefore supports the fulfilment of requirement **R7**. An automatic validation tool for TCSP is not available, though. As no graph-based validation mechanism is provided as well, **R7** is not met entirely.

Evolving Temporal Logic

Conrad and Saake [1996] introduce handling of dynamic object behaviour to temporal logic. They develop this extension as *Evolving Temporal Logic (ETL)*. According to Conrad et al. [1997], the specification language used for this purpose can be considered as an extension of the object-oriented language *Troll* [Jungclaus et al., 1996].

ETL provides behaviour specification based on events. To comply to dynamics in object behaviour, conditions can be specified for these events. Such conditions allow to directly fulfil requirements **R1** and **R2**.

Intervals are derived from event specifications. Thereby, the notion of *life cycles* for object behaviour is introduced. Temporal conditions and constraints do not operate on these life cycles directly but on their underlying events. Relational operations on intervals as required by **R3** are still supported to some degree, though. This is achieved by defining a set of axioms. These axioms express temporal properties of objects that hold valid for the duration of an interval. Support of requirements **R4** and **R5** is provided analogously. Both—merging of intervals as well as their discretisation—depend on constraints defined for respective interval-limiting events.

By introducing a concept of *local time* for an object, requirement **R6** is supported. Synchronisation occurs due to communication between objects. By an according use of these communication channels, constrained concurrency can be achieved.

Similar to *TCSP* as discussed above, requirement **R7** is met in principle as the given formalised specification of *ETL* allows for verification of temporal consistency. As neither automatic proof system tools nor graph-based verification are available, only partial support of **R7** results.

Other Temporal Description Logics

On the example of specifying temporal logic of reactive and concurrent systems, Manna and Pnueli [1992] introduce a set of operators for the construction of temporal formulas. Artale et al. [2001] take this further by introducing a temporal description logic that is based on the *snapshot representation* of abstract temporal databases. Thereby, a temporal database is regarded as a map from time points to standard (relational) databases. This direct bounding to databases is of intrinsic nature for the presented *DLR_{US}* description logic. The logic's idea of a snapshot-notion for temporal modelling holds for other fields of application as well, though.

In order to describe scheduling of interaction events, temporal models based on grammars are used by Moran [1981], Reisner [1981], and Shneiderman [1981]. These approaches share the implicit assumption that the sum of all individual steps define the duration of an action. These steps include interaction events and the notion of an action corresponds to a temporal entity.

4.2.6 Discussion

A summary of the discussed models of this section is presented in Table 4.2. This table lists all models and their respective support of the requirements from Section 4.1.

Model	R1	R2	R3	R4	R5	R6	R7
Time lines	☑	☐	☐	☐	☐	☐	○
Graph-based	☑	(no group characteristics)					☑
Exact Time Stamps	✓	☑	○	○	○	☐	✓
Constrained Propagation	✓	☐	☑	☐	☐	○	✓
Duration-based	✓	○	☑	☑	☑	☐	✓
Firefly	✓	☐	○	○	☑	☐	✓
Hyperstories	✓	○	☑	☐	☑	☐	✓
Petri nets	☑	☑	☑	☐	☐	☑	☐
Object-oriented	(no inherent group characteristics)						
Interaction Diagrams	☑	☑	☑	☐	☐	☐	○
Interaction Machines	☑	☑	☑	☐	☐	☐	○
Interval Diagrams	○	○	☑	☑	☑	☐	☑
Hierarchic Composition	○	○	☑	☐	☐	☐	☑
Path Expressions	○	○	☑	☐	☐	☐	☑
Active Objects	☑	☑	☑	☐	○	☐	○
Temporal Logic	(no inherent group characteristics)						
TCSP	☑	☑	☑	☑	☐	☐	☐
ETL	☑	☑	☐	☐	☐	☑	☐

Table 4.2: Evaluation of temporal models with regard to the requirements of parameterising dynamic presentations. Checked boxes (☑) indicate that a requirement is met directly. A check mark without a box (✓) resembles indirect fulfilment of a requirement in case the model's class already fulfils it. An empty box (☐) stands for marginal support of a requirement whereas a circle (○) stands for a missed requirement.

As can be expected, not one single model supports all requirements equally well. It is of specific interest, though, that petri nets and language-based temporal logic approaches provide the best matches. Both models are powerful with regard to temporal model specification. However, especially in the case of models extending existing programming languages, this is paid for by a task-dependence that is not easily integrated in an arbitrary modelling scenario. More precisely, modelling any temporal scenario by this approach requires use of a specific programming language in question.

An outstanding advantage of petri nets for temporal modelling is its possibility to provide for modelling of constrained concurrency. Even though, not every petri net model provides this, a set of petri net model extensions doing so is available.

The variety of presented object-oriented modelling techniques promises to provide for a set of useful modelling approaches. These may need to be derived from combinations of the models. This holds especially with regard to support of events as well as intervals in a single model.

The following section presents the model for temporal presentation used throughout the remainder of this work. This model is constructed by using a set of concepts from petri nets combined with object-oriented modelling techniques. A temporal presentation function will be presented providing a program-based interface to the model.

4.3 Model for Temporal Presentation

Based on the above evaluation of temporal models, this section introduces a temporal presentation framework for illustrations enhanced by dynamics. For this purpose, the notion of a *dynamics trajectory* is used. Such a trajectory describes the scheduling of dynamic presentation techniques along a single time line. Multiple trajectories make use of multiple parallel time lines and define concurrent presentation of dynamics.

This section is divided into two main parts: behaviour in a single trajectory and modelling of concurrent trajectories. Besides complying to the requirements listed in Section 4.1, some motivation for this is provided by three main characteristics associated with temporal information of the presentation of any object [Prabhakaran, 1997, p. 128]:

1. Time instant of the object presentation,
2. Duration of its presentation, and
3. Synchronisation of the object presentation with those of others.

The first two characteristics basically describe temporal actions in a single dynamics trajectory. The third one supports concurrency and requires multiple trajectories.

A concrete control function based on the model presented here will be discussed in the next section.

4.3.1 Behaviour in a Single Trajectory

Scheduling presentation entities on a single trajectory builds the basis for temporal modelling of a dynamics-enhanced presentation. Use of concurrent trajectories as discussed in Subsection 4.3.2 enhances this concept and requires some additional constraints.

Events

The most atomic temporal term of the model is an event. Similar to the graph-based model of exact time stamps, each event directly corresponds to a specific point in time. That is, an event is of instantaneous nature. The set of all events in a scenario is given as $E = \{e_1, \dots, e_n\}$.

A set of comparison relations is defined for the purpose of scheduling events along a trajectory's time line. Comparing an event $e_i \in E$ with another event $e_j \in E$, both may be equal to each other ($e_i = e_j$), the first might be placed earlier than the other ($e_i < e_j$) or vice versa. For flexible handling of ranges of events, combinations may be used. The overall set of six relations defined for events is summarised in Table 4.3.

notation	relation
$e_1 < e_2$	e_1 happens before e_2 .
$e_1 > e_2$	e_1 happens after e_2 .
$e_1 = e_2$	e_1 and e_2 represent the same point in time.
$e_1 \leq e_2$	e_1 precedes or equals e_2 .
$e_1 \geq e_2$	e_1 does not happen before e_2 .
$e_1 \neq e_2$	e_1 and e_2 are disjunct.

Table 4.3: Set of relations defined for events.

Any set of events belonging together may be combined into an *event group* denoted as E_G . Such an event group defines an event-composite. Each group can be used for parameterisation of a temporal entity associated with it. Precisely, each event $e_i \in E_G$ represents a point in time at which the state of the temporal entity might change. Behaviour between two events in a group is not explicitly defined for at any point in time. This defines event groups as a kind of interval with holes. Groups may overlap. That is, an event that is element of any one event group E_{G_1} may also be element of group E_{G_2} . A group is only limited by its elements. This directly implies that any given group may contain other groups either in whole or in part.

The insertion of new events into a concrete temporal model instance is handled by an event function f_e . This function uses the set of relations defined above for comparing the inserted event with the existing event set E . Initially, a new event is not put into any specific event group. This mapping of events onto groups is addressed by a separate group function f_g .

This group function provides a set of grouping cases. First of all, a newly inserted event may equal an already existing event. This directly leads to the creation of a new event group that contains both events. Any other groups are not affected. As a second case, the new event is simply inserted into an already existing event group. For each new event, this may be repeated depending on the number of affected

temporal entities. Besides the insertion of new events, to more cases are handled by f_g : removal and rearrangement. As much as any event can be inserted to an event group, it may also be removed from it in case its causing context does not hold anymore. Rearrangement does not affect the number of events in a group. It is used for internal reorganisation. This results in the creation of subgroups that may be used for fine grained parameterisation purposes. The function's operation is defined as

$$f_g(e_i, E_{G_j}) = \begin{cases} E_{G_k} & \Leftrightarrow e_k \in E_{G_j} \text{ and } e_i = e_k & \text{[double entry]} \\ E_{G_j} \cup \{e_i\} & & \text{[insertion]} \\ E_{G_j} - \{e_i\} & & \text{[removal]} \\ E_{G_j} \cup E_{G_l} & | E_{G_l} \subseteq E_{G_j} & \text{[rearrangement].} \end{cases}$$

Example 4.2 illustrates the insertion of an event.

Example 4.2 Given is an event group $E_G = \{e_1, e_2, e_3\}$ describing the temporal behaviour of an object's presentation. Starting at tie point in time represented by e_1 , the object is rotated repeatedly around the same axis. Event e_2 causes this rotation to change into an oscillating rotation, that is, the direction of rotation is repeatedly changed while the rotation axis remains the same. A doubling of the rotation's frequency is modelled with e_3 . A user interaction event e_4 is now introduced into this group. This event is characterised by $e_2 < e_4 < e_3$. The effect of e_4 is an immediate stop of the rotation. As e_4 happens before e_3 , the change in frequency causes no visual effect. \square

Intervals

To schedule the number of employed dynamic techniques in a single trajectory, use of each given technique is modelled with a constraint of its use over time. For managing temporal dependencies a set of relations is used. This is based on the interval relations discussed in Section 4.1.2. A relation between two given intervals I_i and I_j is expressed by $I_i \circ I_j$ with $\circ \in \{<, \leq, =, \geq, >\}$. An interval $I = \{e_{begin}, e_{end}\}$ with an implicit interval duration of $e_{end} - e_{begin}$ is given as a further model primitive. For easier notation, the interval borders can be specified by I^b and I^e directly.

Given a set \mathcal{D} of available dynamic presentation techniques, $\mathcal{D} = \{d_1, \dots, d_j\}$, and a set \mathcal{I} of intervals, $\mathcal{I} = \{I_1, \dots, I_k\}$, the mapping from \mathcal{I} to \mathcal{D} is specified by a dynamics function $f_d : \mathcal{I} \rightarrow \mathcal{D}$. For the application of consecutive dynamics patterns this function maps disjunctive intervals onto disjunctive methods:

$$f_d(I_i) = d_n \quad 1 \leq i \leq k, 1 \leq n \leq j.$$

Modifiable dynamics patterns are characterised by a set of temporal dependent attributes $\mathbf{AM} = \{a_1, \dots, a_k\}$. This denotes the dynamics methods as $d_n^{\mathbf{AM}} \in \mathcal{D}$ with

$1 \leq n \leq j$. Individual interpretations of **AM** are determined by use of an attribute function:

$$f_a(I_i) = a_o \quad 1 \leq i \leq k, 1 \leq o \leq k.$$

In order to allow modifications of the parameter space during the time span $I_i^e - I_i^b$, *interpolation* of respective interpretations a_o is possible.

As noted above, the set of known events for the overall system runtime is named E . To determine the correlation of any given event $e_i \in E$ to an interval $I \in \mathcal{I}$, a function $\psi : E \rightarrow \mathcal{I}$ is defined as noted in equation (4.1).

$$\psi(e) = \begin{cases} I_0 & \text{single interval} \\ I_i \Leftrightarrow I_i^b \leq e \leq I_i^e & |i = [1, 2] \text{ double interval} \\ I_j \Leftrightarrow I_j^b \leq e \leq I_j^e & |1 \leq j \leq n \text{ otherwise} \end{cases} \quad (4.1)$$

Thereby, $\psi(e)$ maps the event's point in time onto its corresponding interval by using plain comparison operations. As I^b and I^e represent events, the relations expressed in Table 4.3 apply.

An interval can be discretised by use of an interpolation function $\tau : I \rightarrow \mathcal{I}$. Thereby, a change between different instances of the parameter space can be accomplished fluently. This allows for acceleration and deceleration as well as for fading individual motion techniques.

Exemplary Interval Layers

For illustration purposes, a set of interval layers is presented in Figure 4.19. It shows use of different representations of temporal behaviour depending on the number of modelled time intervals.

The *single interval* level represents the most simplistic model complexity. Applying any presentation method—including dynamics—is not timely restricted but the method's influence is shown for the whole interval duration. As soon as the interval ends, i.e. at point I_1^e , any possibly active dynamic display vanishes.

This method level allows for easy interaction support as the user is not disturbed by any dynamic activities. Thus, possible manipulations of object positions are avoided along with invalidations of (parts of) the scene coherence. Preemptive visual clues provided by use of this method level are possibly not satisfactory enough for all user requirements.

The *double interval* provides two different ways of representing dynamics: temporally limited continuous dynamics as well as a sequence of two consecutive dynamic techniques. Their duration results from the interval bounds of I_1 and I_2 . As $I_1^e = I_2^b$, a continuous motion presentation is limited by $I_1^e - I_1^b$ which is the complete duration of the first interval. The remaining time can be used analogously to the single interval level.

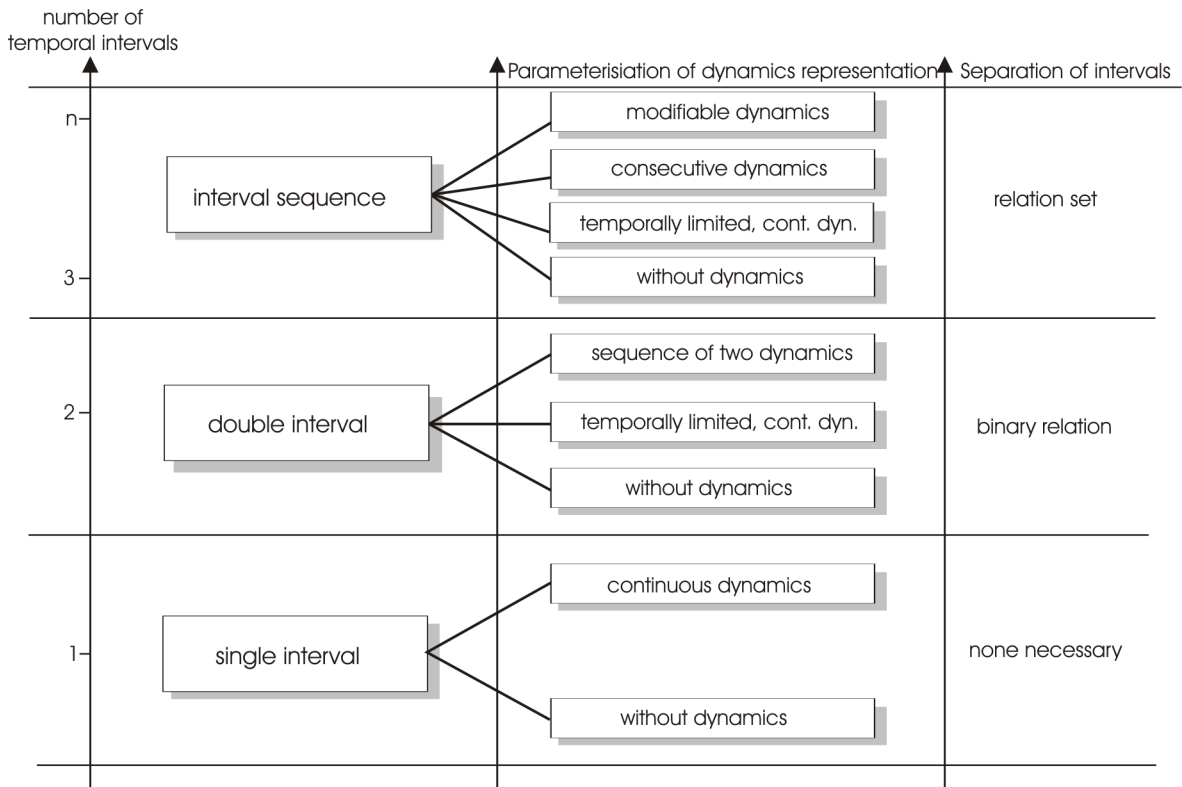


Figure 4.19: Layer model for temporally constrained dynamics presentation.

Sophisticated dynamics patterns may be employed for a given *interval sequence*. First of all, variations provided on both former interval levels are applicable on this level as well. This explicitly includes the exclusive use of classic presentation variables only, or a combined use of these and object based dynamics. Application of dynamics on this level is parameterisable by means of two categories: display of consecutive dynamics patterns or display of modifiable dynamics patterns.

Presentation Example

Specifying temporal behaviour in a single dynamics trajectory allows to model consecutive presentations. These presentations may either be definite or flexible. In the following, a respective example is shown for both cases.

Example 4.3 describes a model for a *definite* scheduling of a single object presentation. Temporal status of this description is definite as the temporal entity is static by nature. Its exact starting event is known as well as its precise duration of presentation.

Example 4.3 Present the object of interest with dynamic presentation d_j starting at event e_k for the duration of exactly three seconds. \square

A *flexible* presentation description is given in Example 4.4. Possibly depending on presentation of other objects, specific presentation of the object in question is started in a temporal frame defined by two events. The precise duration of this presentation is flexible as well. This possibly influences scheduling of another presentation based on a flexible description.

Example 4.4 Present the object of interest with a fading presentation technique d_j at some point in time between event e_m and event e_n until the fading is completed. \square

4.3.2 Concurrent Trajectories

As discussed in Section 2.1.4 any simultaneous use of dynamic presentation variables needs to be subject to restricting parameterisation for the purpose of complying to cognitive limitations. In support of this, a concurrency layer model for dynamics is introduced here. Based on this model, a parameterisation function is defined for the purpose of evaluating the number of simultaneously active temporal entities and allowing to constrain them.

Layer Model

A set of four different layers is defined for temporally restricting concurrent dynamics trajectories. These are illustrated in Figure 4.20.

The simplest layer of this model employs a single trajectory only. This directly correlates to the scenario presented in the previous subsection. Along the time line of this trajectory, dynamics can be scheduled freely without any further need of temporal restriction.

On top of this, a layer is placed consisting of two parallel trajectories. This allows for a combination of up to two concurrently dynamic presentations. Each of these may be scheduled individually as described above. From the cognitive point of view, no restriction needs to be put onto two trajectories. For the purpose of using both as separate presentation paths, though, any concurrently presented dynamics should be parameterised such that they clearly distinct from each other.

Adding another trajectory to the double layer results in *tripled* concurrency. Besides the distinction property of the previous layer, further synchronisation is to be provided here. This synchronisation holds responsible for ensuring scene consistency as well as avoiding the dynamics to interfere with each other. Scene consistency is affected in case a dynamic presentation causes objects to be taken out of their former context. This happens in case objects belonging together get affected by

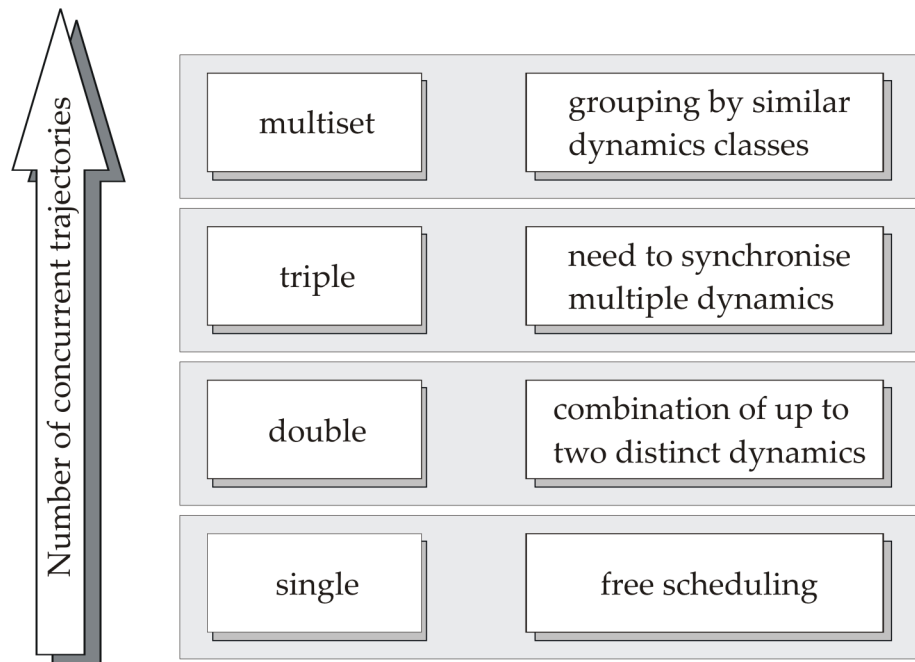


Figure 4.20: Concurrency layer model for restricted use of dynamic presentation variables.

different dynamic presentation techniques, respectively. Interference of dynamics may occur specifically in case involved dynamics include positional change. This way, affected objects possibly collide, which is to be avoided.

Simultaneously employing more than three dynamics trajectories is subject of the *multiset* layer. Any restriction valid for the previous layer holds for this one as well. In addition, the number of simultaneous dynamics eventually needs to be limited. This is specifically due to visual overload caused by multiple dynamics as outlined in Section 2.1.4. This is furthermore combined with temporal limitations as defined by the dynamics stimulus window (see Section 2.2). Besides a pure limitation of the number of simultaneously allowed dynamics, different techniques can be combined to form *dynamics classes*. As elements of such classes share visual attributes, a whole class provides less cognitive load than the sum of its members on their own. The backside of merging dynamics into classes is in loosing some of their expression capabilities. Dynamics of a given task are not freely parameterisable but need to follow constraints of the classes.

Concurrency Evaluation

Based on the concurrency layer model, a concurrency evaluation function γ is defined. This function takes scheduled dynamic presentations and evaluates the schedules based on the set of currently active dynamic trajectories. This is a transaction-based operation consisting of six single steps:

1. *Start transaction:* A snapshot of the system's temporal state is taken and any other modification of this state is blocked until the transaction is committed.
2. *Determine pre-evaluation layer:* All currently active dynamics trajectories are evaluated and their respective status is analysed.
3. *Evaluation of scheduled dynamics:* All scheduled techniques are arranged such that an optimum distribution along as less trajectories as possible is constructed.
4. *Determine post-evaluation layer:* The state from step (2) and the result from step (3) are merged and a new state of concurrency is constructed. This state corresponds to one of the layers.
5. *Re-scheduling of dynamics:* The resulting state from step (4) is evaluated with respect to the layer model and the former state of step (2). In case, the layer has changed compared to step (2), the scheduling of the dynamics is re-calculated.
6. *Committing transaction:* The overall transaction as finalised in step (5) is committed. This also re-opens the system to further modifications.

Constructing a transaction for handling of scheduled dynamics ensures their encapsulation. All outstanding dynamic presentations are handled the same. That is, either all of them are applied unmodified, all are blocked or all are constrained. This respects any possible reason for grouping the scheduled techniques in the first place.

Constraining Concurrency

Parallel partitioning of employment of dynamic presentation techniques helps to make use of their full range of expression capabilities. For the purpose of minimising possible caveats, implementations of the parameter function f_p are to be restricted appropriately. A possible switch back to classic presentation variables by according parameterisation of f_p is provided as well. This allows to avoid any drawback in interaction support as possibly caused by dynamics such as motion.

Three different ways of constraining dynamic presentations may be used:

Restricting dynamic character; The parameterisation of an individual dynamic presentation technique is modified such that the dynamic character is loosened. In case of a motion technique this can be achieved by either lowering the amplitude or the frequency of the motion. In case of rendering style dynamics, the time frame of the blending may be stretched.

Merging multiple dynamic techniques; Instead of modifying a single technique, multiple dynamic presentations are merged. Overall, this results in a reduced number of active dynamics. The merging of techniques is only to be applied

in case the same information objects are effected. Otherwise, the communication goal of individual techniques gets easily de-constructed.

Fading out dynamic techniques; In case both above approaches fail in fulfilling the constraining requirements, selected techniques might need to be removed from the presentation. In order to not disturb the overall presentation composition, any dynamic presentation is not just switched off abruptly but faded out smoothly. The fading parameterisation is to be chosen such that it complies with the launching of any possibly new technique in the presentation, regardless if this technique is of static or dynamic nature.

4.4 Specification of Control Function

This section presents the specification of a control function that is used as basis for temporal control of dynamics. For the purpose of controlling presentation methods, a combination of absolute dates and duration specification is used in the context of this work. This allows for generation of a complete linear order of all events as well as generation of temporal networks with a fine grained model granularity in accordance with Bettini et al. [2000].

4.4.1 Notation

The notion of a *specification* is meant as the description of desired behaviour of a dynamic presentation system, while avoiding concrete references to its methods or implementation details. The specification of the temporal control function builds on the presentation framework notation introduced in the previous chapter. Thus, the framework is extended by means of temporal constraints and parameterisation.

In order to name the different relation sets, some variables are introduced. The parameter set for a specific presentation is denoted as δ . All defined rendering styles (methods) are represented by \mathcal{D} :

$$\mathcal{D} = \{d_1, d_2, \dots, d_n\} \quad n \in \mathbb{N}.$$

The available interval set is named \mathcal{I} whereas an individual interval is labelled I :

$$\mathcal{I} = \{I_1, I_2, \dots, I_m\} \quad m \in \mathbb{N}.$$

The set of time stamped events is described by E . The functions τ and ψ are used for operations on events and intervals. Discretisation of an interval resulting in an interval set is accomplished by τ and the mapping of any specific event onto its respective interval results from ψ .

The specification of the temporal model is given as tuple $SPEC = \langle S, OP, REL, X, REQ \rangle$. The elements of S represent all *sorts (types)*. S^* names the set of all

sort chains. For each $s_a \in S^*$ and each $s_b \in S$, the set of all possible operators is named OP_{s_a, s_b} . The elements op of OP_{s_a, s_b} are therefore projections $op : S^* \times S \rightarrow S$. For each $s_a \in S^*$ represents REL_{s_a} the set of all possible relations. The set $\langle S, OP, REL \rangle$ is commonly referred to as signature of the specification. X marks the defined variable set. Last, but not least, names $REQ \subset F_{SIG}(X)$ the set of formulas that define the algebra of the specification.

For the benefit of readability this syntax is simplified to some degree. The different tuple components are presented in a tabular form whereby each set is prefaced with a keyword. The sorts are presented after the keyword *sorts*, the operations follow *opns*, relations are listed after *rels*. All variables are prefaced by *vars* and the formulas follow *reqs*.

4.4.2 Function

A function $\delta(\mathcal{D}, \mathbf{AM}, \mathcal{I}, \mathcal{F})$ is defined for temporal parameterisation of dynamic presentations. Thereby, \mathcal{D} names the set of available dynamic presentation techniques, \mathbf{AM} represents the set of visual attributes of presented objects as it is modified by the function, \mathcal{I} names the set of all currently defined temporal intervals, and \mathcal{F} holds a set of transformation functions.

Presentation parameterisation $\delta(\mathcal{D}, \mathbf{AM}, \mathcal{I}, \mathcal{F})$

sorts: $I, \mathcal{I}, E, \mathcal{D}, \mathbf{AM}, \mathcal{W}^{rep}$

opns: $\psi : E \rightarrow I$

$\tau : I \rightarrow \mathcal{I}$

$+$: $\mathcal{I} \rightarrow \mathcal{I}$

dsw : $\mathcal{I} \rightarrow \mathcal{I}$

f_a : $\mathcal{I} \rightarrow \mathbf{AM}$

f_d : $\mathcal{I} \rightarrow \mathcal{D}$

f_e : $E \rightarrow E$

f_g : $E \rightarrow E$

rels: $<, \leq, =, \neq, >, \geq : ee$

$\circ = \{<, \leq, =, \geq, >\} : II$

$\subset, \subseteq, = : \mathcal{I} \mathcal{I}$

$\neq : \mathcal{D} \mathcal{D}$

$\neq : \mathcal{W}^{rep} \mathcal{W}^{rep}$

vars: $e_1, \dots, e_n : E$

$I_1, \dots, I_n : \mathcal{I}$

$d_1, \dots, d_n : \mathcal{D}$

$world_1^{rep}, \dots, world_n^{rep} : \mathcal{W}^{rep}$

reqs: $\psi(e_n) < \psi(e_m) \Leftrightarrow e_n < e_m$

limiting interval borders:

$\tau : \forall I_i \in \mathcal{I} \Rightarrow I_1^b \leq I_i^b \wedge I_n^e \geq I_i^e$ $d_a \neq d_b \Leftrightarrow f_d(I_a) \neq f_d(I_b) \wedge I_a \neq I_b$ $\mathbf{AM}_a \neq \mathbf{AM}_b \Leftrightarrow f_a(I_a) \neq f_a(I_b) \wedge I_a \neq I_b$ <p>cardinality of the dynamics stimulus window:</p> $dsw(\mathcal{I}) \subseteq \mathcal{I} \wedge dsw(\mathcal{I}) \leq \mathcal{I} $ <p>limiting interval lengths:</p> $I_a < I_b \Leftrightarrow I_a^b \geq I_b^b \wedge I_a^e < I_b^e$ $I_a = I_b \Leftrightarrow I_a^b = I_b^b \wedge I_a^e = I_b^e$ $I_a > I_b \Leftrightarrow I_a^b \leq I_b^b \wedge I_a^e > I_b^e$ $\mathcal{I} + \lambda = \mathcal{I}$ <p>interval continuity:</p> $\mathcal{I}_a = \{I_1, \dots, I_a\} \wedge \mathcal{I}_b = \{I_{a+1}, \dots, I_b\} \Rightarrow \mathcal{I}_a + \mathcal{I}_b = \{I_1, \dots, I_b\}$ <p>transitivity:</p> $\mathcal{I}_a \subseteq \mathcal{I}_b \wedge \mathcal{I}_b \subseteq \mathcal{I}_c \Rightarrow \mathcal{I}_a \subseteq \mathcal{I}_c$ <p>seclusiveness of dynamics:</p> $d_a \neq d_b \Leftrightarrow world_a^{rep} \neq world_b^{rep}$

Table 4.4: Specification of a parameterisation function for dynamic presentations.

Table 4.4 presents the specification of the temporal presentation function. This function basically maps instances of the overall set of representable worlds (\mathcal{W}^{rep}) onto a concrete presentation. A represented world $world^{rep} \in \mathcal{W}^{rep}$ holds an illustration model. This illustration model represents the information space $\mathbf{IM} = \{IO_1, \dots, IO_n\}$. Each represented world also includes an illustration target function, and some contextual scope for the illustration. Some more details of represented worlds are discussed in the context of the illustration target function in Section 3.1.

The set of transformation functions is defined as $\mathcal{F} = \{\psi, \tau, dsw, f_a, f_d, f_e, f_g\}$. Principles of the respective modes of operation for the individual functions directly correspond to the temporal presentation model as introduced in Section 4.3. Any concrete function parameterisation is application specific. Examples will be presented in Chapter 6.

For each predefined or otherwise determined interval, a combination of dynamic presentation techniques ($d_i \in \mathcal{D}$) can be defined and parameterised. The intervals determining a presentation are both—complete and self-contained. That is, any time stamp (event) can be mapped onto an interval (by use of ψ) and no interval of the model exceeds the interval set for the overall presentation. This is denoted in the table as *limiting interval borders*. The respective constraints for a concrete allocation of \mathcal{I} are outlined in the previous section. These include *interval length limitations* as well as *continuity constraints* and *transitivity*.

4.4.3 Graphical Representation

The graphical representation of the presented model and its control function is defined as a temporal graph. This graph uses some concepts from petri nets as it represents flow through a network of states with a set of transformations in between. In addition, the graph supports modelling of different temporal granularities by respecting events as well as intervals. Explicit support of constraint concurrency can be expressed in the graph.

All temporal entities in the graph are marked with shadows. This visually represents their temporal floating character and distinguishes them from transformations that are represented as plain objects.

Events

For the purpose of placing events in the graph, each event is represented by a rectangular block. An excerpt of placing an event group containing two events into the graph is shown to the left of Figure 4.21. Both events in the figure define the temporal border of a presentation d_i . That is, d_i operates on an interval spawned by e_b and e_e . To the right of the figure, consecutive scheduling of event-based qdynamic presentation techniques is shown.

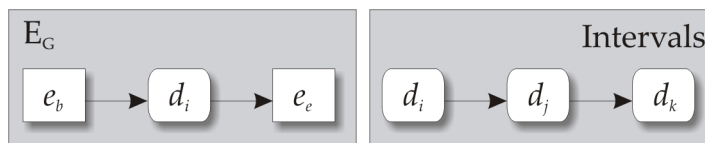


Figure 4.21: Graphical representation of events and intervals for the δ transformation function. To the left, the interval limitation by e_b and e_e is shown. To the right, consecutive scheduling of a set of dynamics is pointed out.

Intervals

Intervals are represented in the graph by rectangular blocks with rounded corners. This visually distinguishes them from events. Intervals are implicitly created out of any two elements of an event group. For the example shown in Figure 4.21, the individual presentations $d_{\{i,j,k\}}$ are active for their respective time spans of an interval. This representation furthermore helps to express the lack of precise time stamps. This lack is due to the circumstance that the graph does not provide direct support of modelling precise time stamps for interval borders. However, any points in time can be specified by using the notion of event groups as discussed above. Scheduling of a set of interval-based presentations on a single dynamics trajectory is expressed in the righthand side of Figure 4.21.

Concurrency

Explicit support of expressing concurrency in the graph is provided by a δ transformer. This transformer contains a set of transformation functions. These directly correlate with the γ , τ and dsw elements of the control function. The transformer is placed at each point in the graph where it either branches or merges the presentation. Figure 4.22 points out both cases. On the lefthand side of the figure, γ and τ control an initiation of concurrency by branching the graph. A single dynamics trajectory is divided up into a set of three trajectories. Internally, the transformer may make use of the dynamics function f_d for proper parameterisation of the individual techniques. This has no visual impact on the graph.

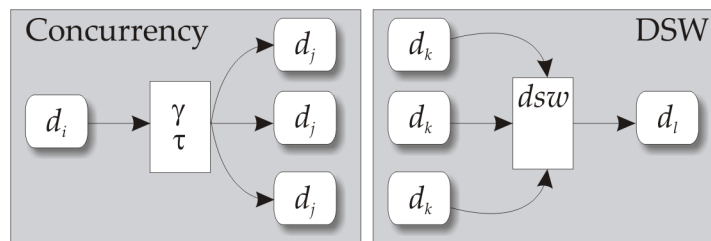


Figure 4.22: Graphical representation of concurrency constraining by the δ transformer. The transformer contains three constraining elements for this purpose: ψ , τ , and dsw . For clarity purposes, these are divided into two groups here: one for ψ and τ and one for dsw .

Reference Scenario

Figure 4.23 shows the graph for the exemplary reference scenario presented in Figure 4.2 of Section 4.2. Compared directly with the petri net model from Figure 4.13, it is to be noted that less steps are needed to express the scenario. This is specifically reasoned in the concurrency transformers that handle immediate evaluation and verification at all branching and merging points.

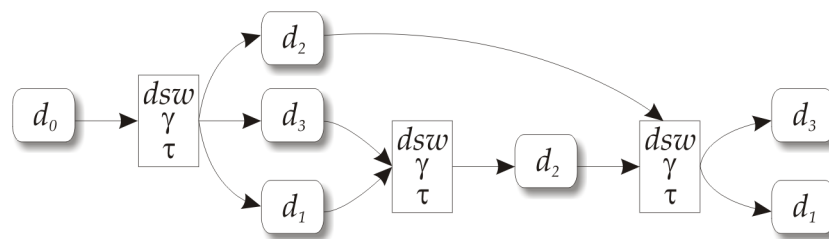


Figure 4.23: Model of the reference scenario from Figure 4.2.

4.4.4 Fulfilling the Requirements

Table 4.5 lists the elements of the temporal control function for dynamics that respectively fulfil the requirements introduced in Section 4.1. Both event-based requirements are basically addressed by handling of event groups and supportive functions. Relational operations on intervals as required by **R3** is naturally subject of the dynamics stimulus window function dsw and the relational operators defined on intervals (I) and interval sets (\mathcal{I}). The main interval handling functions ψ and τ address merging and discretisation of intervals which are subject of **R4** and **R5**. Even though constraining concurrency (**R6**) is explicitly handled by γ , this is additionally supported by ψ and τ as well. Both of these functions also ensure temporal consistency as required by **R7**. This is furthermore provided by the graph being of acyclic nature.

Requirements	R1	R2	R3	R4	R5	R6	R7
Function	$E_G,$	$E_G,$	$dsw,$	ψ	τ	γ	$\psi, \tau,$
Elements	$\text{rels}(E)$	f_e, f_g	$\text{rels}(I, \mathcal{I})$			(ψ, τ)	graph

Table 4.5: Fulfilling the requirements presented in Section 4.1 by elements of the temporal control function.

As a result, the temporal control function δ presented in this section is capable of fulfilling all stated requirements for temporal management of dynamic presentations in an illustration system. Compared with existing temporal models as discussed in Section 4.2, this provides a considerable potential for constraining the dynamic presentations of the previous chapter. This allows for these presentations to comply to cognitive fundamentals as outlined in Chapter 2.

4.5 Script-Based Controlling of Dynamics

Using a script for parameterisation of dynamics allows to provide for a means of an interface to consistently control illustrations enriched by these dynamic presentation techniques. Such consistent interface regards the different characteristics and effects of the individual techniques.

As motivated by the fundamentals of dynamics in Chapter 2, use of dynamic presentations needs to be constrained and accordingly parameterised. Temporal control provides one such option of restricting dynamics parameterisation. While the following chapter presents a *dynamics management unit* for this purpose, the script as outlined here bridges between the motivations for controlling dynamics and a concrete implementation of it.

4.5.1 Script Basis

Continuing the temporal presentation framework from Section 4.4.2, the function δ is used in the script for temporally constraining any use of dynamic presentation variables. Precisely, δ is used to limit the set of currently active dynamic presentation techniques $d_i \in \mathcal{D}$. The set of presentation techniques \mathcal{D} is composed as:

$$\mathcal{D} = \mathcal{D}_{OS} \cup \mathcal{D}_{ST} \cup \mathcal{D}_{RS} \cup \mathcal{D}_{DI}.$$

Thereby, the individual sets of techniques correspond to the dynamics presented throughout Chapter 3:

- \mathcal{D}_{OS} refers to the set of oscillating motion from Section 3.3.1.
- \mathcal{D}_{ST} describing the set of motion by structural changes from Section 3.3.2.
- \mathcal{D}_{RS} including the set of dynamic combinations of hybrid rendering styles as discussed in Section 3.4.
- \mathcal{D}_{DI} holding the set of distortion-based dynamics as presented in Section 3.5.

Each currently active or scheduled dynamic presentation of an information object IO is denoted as $d(IO)$ with $d \in \mathcal{D}$. Each information object is part of the information set describing the illustration model: $IO \in \mathbf{IM}$. Out of this information set, multiple information structures may be derived:

- \mathbf{IS}_{OS} structure containing all information objects subject to oscillating motion.
- \mathbf{IS}_{ST} all information objects subject to motion by structural changes.
- \mathbf{IS}_{RS} all information structures subject to dynamic changes of hybrid rendering styles.
- \mathbf{IS}_{DI} all information objects to be distorted by \mathcal{D}_{DI} .

These structures may be combined, that is, any specific information object may belong to more than one structure at a time.

4.5.2 Script Composition

Figure 4.24 outlines the structure of the individual script components. The dashed lines point out *is-part-of*-relationships between elements of the script. The connection lines between the individual script components annotated with cardinalities describe quantitative relationships between elements. Two different kinds of such relationships exist:

1. *One-to-one* (1 : 1) relationship: An instance of an element type is connected with exactly one instance of another type.
2. *One-to-many* (1 : n) relationship: An instance of an element type is connected with multiple instances of another type. This may either be an open relationship (1 : n) or a fixed relationship with a constrained upper limit of n (such as 1 – 3).

An example for the second case is the relationship between *temporal presentation constraints* δ and *scene coherence constraints*. The former constraints refer to as many as three coherence constraints whereas each instance of the latter is combined with exactly one instance of δ .

Each dynamic presentation $d(IO)$ (with $d \in \mathcal{D}$, $IO \in \mathbf{IM}$) is parameterised by a parameter list. The layout of this list in the script specification is generic. This allows for a flexible description of different types of dynamic presentation techniques. A parameter list PL is given as:

$$PL = \{ps_i\} \text{ with } ps_i = \{\text{key}, \text{value}\} \quad i \in \mathbb{N}.$$

That is, the list consists of a set of parameters ps_i . Each of these parameters is defined by a *key-value* pair. The concrete interpretation of the content of such pairs in a parameter list is left to the implementation of the concrete dynamic presentation technique at hand.

As stated above, the parameterisation function δ holds responsible for temporal control of concurrent dynamic presentations. Representation of the temporal constraints described by δ is expressed by a combination of *scene coherence constraints* and *temporal presentation descriptions*. Recalling Section 3.6, the former constraints include any combination out of the local, regional, or global character of a dynamic presentation technique.

A presentation description is composed as $pd = \{\text{start}, \text{duration}\}$. With respect to each instance of a presentation $d(IO)$ this description is combined with a 1 : 1 relationship. Overall, the temporal presentation constraints as defined by δ provide the available set of presentation descriptions by a 1 : n relationship.

A concrete implementation of parameterising dynamically enhanced presentations on a script bases will be shown in the next chapter. Therein, XML is used which allows to integrate the presented dynamic techniques in an webservice-based illustration environment.

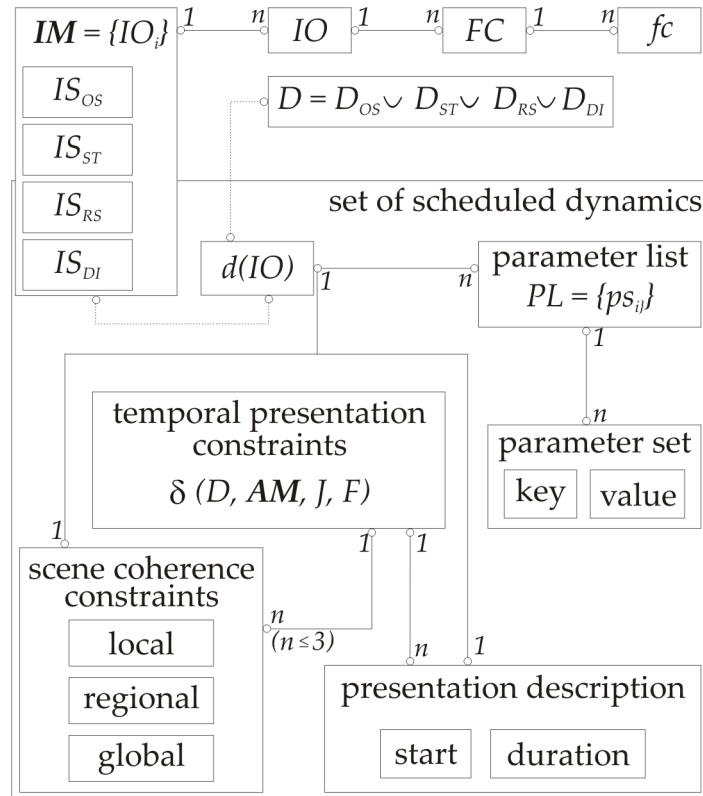


Figure 4.24: Structure of the individual components of a script for controlling dynamic presentations.

4.6 Summary

Subject of this chapter is the development of a temporal model. The purpose of this model is support of constraining and parameterisation of dynamic presentations for illustration purposes as presented in the previous chapter. In support of this, a set of requirements is introduced for such a temporal model. Based on these requirements, an overview of existing temporal models is presented. These models are classified according to their characteristics. Resulting classes along with their respective models are evaluated with regard to the listed requirements. Based on the evaluation results, a temporal model is developed that fully meets the temporal presentation requirements. For this model, a control function is presented as well as a graphical notation. The results of this chapter will be picked up in the following. Chapter 5 includes discussion of a *Temporal Server* based on this model whereas Chapter 6 presents a set of applications making use of temporally constrained dynamic presentations.

5 Toolkits for Temporally Constrained Dynamic Presentations

Figure 5.1 shows the four main components which form the basis for an implementation of temporally constrained dynamic presentations as presented in this work. An overview of the common basis for these components is presented in Section 5.1 along with a discussion of the relationship between the different implemented parts. This is followed by a discussion of a *Motion Toolkit*, an integration of dynamic non-realism in *OpenNPAR*, a *Temporal Server*, and a *Workbench for Information Fusion* in Sections 5.2 to 5.5, respectively.

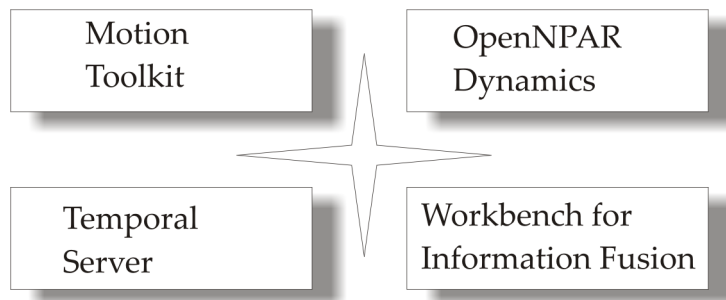


Figure 5.1: Four main components forming the basis for temporally constrained dynamic presentations.

5.1 Implementation Basis

The individual components implemented in the context of this work share some common basis and stand in relation to each other. This basis as well as the relations are subject of this section. First of all, using a scene graph for materialising geometry information is introduced in Subsection 5.1.1. Secondly, Subsection 5.1.2 presents the principle interplay of the four toolkit components. Sections that follow will discuss specifics of the separate components, respectively.

5.1.1 Scene Graph

Presentations for illustration purposes target handling of geometric models. The origin of these model may either be a design construction process or some (semi-)

automated mapping task such as an information visualisation pipeline. The geometry at hand is part of the illustration model. As expressed in the context of an illustration target function in Section 3.1, this model is part of the modelled world $world^{mod}$ and contains the *scene* and *relations* therein.

The scene itself is expressed as an information set **IM** consisting of information objects *IO*. A scene graph [Strauß and Carey, 1992] is used to materialise the individual objects. Furthermore, this graph allows to reflect relations between the different objects as well. Relations are modelled by placing objects in the graph according to their respective reference to each other.

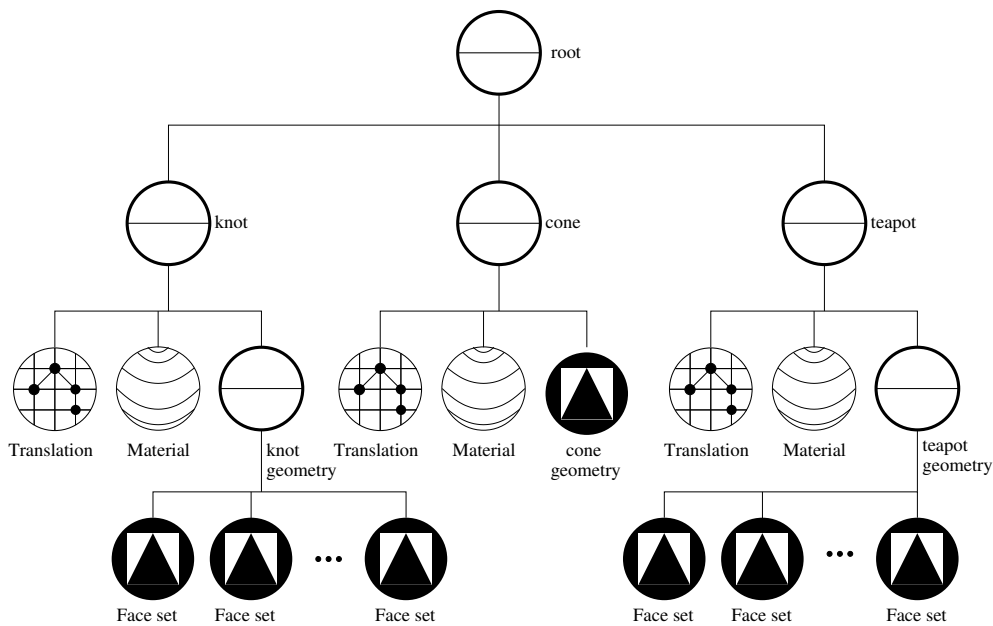


Figure 5.2: Simple example for a scene graph layout representing a geometric model consisting of an information set $\mathbf{IM} = \{IO_{knot}, IO_{cone}, IO_{teapot}\}$.

Figure 5.2 shows a simple example scene graph to illustrate this principle. This graph represents an information set consisting of three information objects: a knot, a cone, and a teapot. This is an arbitrary collection of objects, not corresponding to a realistic illustration scenario and only used for simplicity reasons here. For each of these objects, some transformation and material information is stored in the graph. The example shows that different granularity level may be used for representing the geometry of each object. While the cone is specified directly by a graphical primitive, the knot as well as the teapot are constructed out of a set of primitives. A resulting rendition of this exemplary scene graph is shown in Figure 5.3.

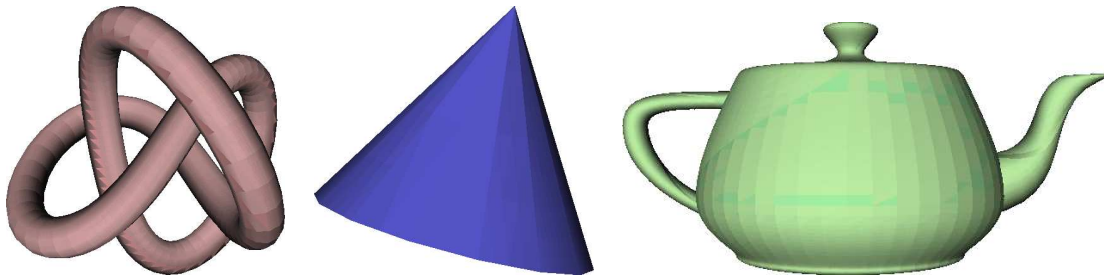


Figure 5.3: Resulting rendition of the exemplary scene graph from Figure 5.2.

5.1.2 Layout of Toolkits

Out of the four toolkit components, two are inherently graphics-based: the *Motion Toolkit* and dynamics as part of *OpenNPAR*¹. Both handle geometry by use of a scene graph. The precise scene graph implementation used here is *Open Inventor* [Wernecke, 1994a,b]. As of this, an abstraction from the underlying graphics hardware is achieved. Figure 5.4 illustrates the resulting abstraction layer of the implemented components with regard to their graphical basis. Thereby, the figure references *OpenGL* as an intermediate layer between concrete graphics hardware and *Open Inventor*. Even though this is no inherent requirement of the abstraction layer or the *Open Inventor* specification as presented by Wernecke [1994a,b], any available real-world implementation of *Open Inventor* is using *OpenGL* for this purpose.

The remaining two toolkit components are not directly graphics-based but serve as illustration helper function and integration testbed for complex illustration scenarios. A *Temporal Server* evaluates temporal constraints as presented in the previous chapter. This way, it provides support of controlling dynamic presentations. A *Workbench for Information Fusion* represents a multi-component application basis for information retrieval purposes. The illustration parts of this workbench allow to make use of *OpenNPAR* as well as the *Motion Toolkit*.

5.2 Motion Toolkit

The algorithmic basis for motion as a dynamic presentation variable is subject of Section 3.3. There, a distinction is made between motion by oscillation and motion by structural change. As a further extension, a motion-enhanced information mural is described which combines both approaches for motion in an illustration algorithm. This section describes implementation issues of the first two motion techniques. The information mural is subject of a more in-depth discussion in the

¹ *OpenNPAR* is a toolkit for non-photorealistic rendering and animation [Halper et al., 2003]. Some details of this toolkit and an extension thereof will be discussed in Section 5.3.

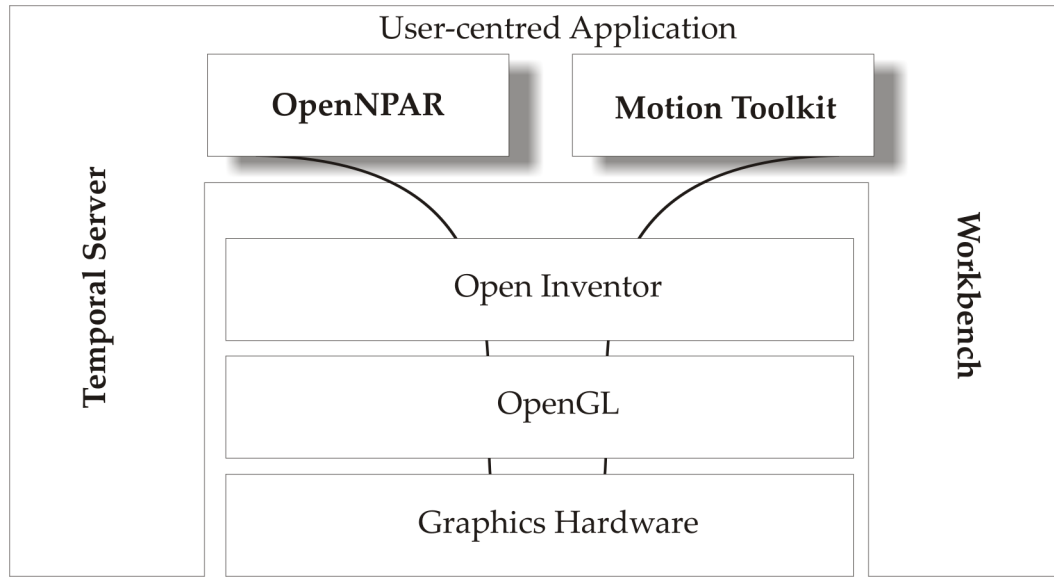


Figure 5.4: Abstraction layer of the implemented toolkit components and their relation to underlying graphics toolkits.

next chapter at the example of a concrete illustration scenario.

For each of the two approaches presented, the respective scene graph construction is shown. Specific elements and structures are introduced that provide motion effects based on the scene graph. Subsection 5.2.1 targets oscillation-based motion whereas implementation of motion by structural changes is subject of Subsection 5.2.2.

5.2.1 Oscillating Motion

An oscillation function $f_{OS}(\mathbf{IS}_M, s, \mathcal{D}_M, \delta)$ is introduced in Section 3.3.1. The parameter space for oscillations is addressed by this function. With regard to the scene graph implementation, this parameter space is encoded in a set of *nodekits*. A nodekit is a collection of nodes combined to provide uniform access to a sub scene graph representing a geometric object. The individual objects of the information structure \mathbf{IS}_M to which the motion is applied are represented separately in respective nodekits. That is, for each $IO_M \in \mathbf{IS}_M$, a nodekit instance is created. The measuring function s holds and manages a list of all these objects. The method set \mathcal{D}_M of all available oscillation techniques is given as a collection of nodekits forming the basis of oscillations:

- *ShuttleKit*: providing a transition between a set of key positions. A cosine function is used from the underlying *Open Inventor* implementation to create a smooth motion.

- *PendelKit*: providing an oscillating rotation based on a set of key positions.
- *PulseKit*: providing a pulsating oscillation. The pulse is characterised as an accelerated variation of the above nodekits.

Using these oscillation primitives, different motion techniques can be constructed by varying parameterisation. For this purpose, each nodekit consists of a variety of parameterisation nodes that address classic presentation variables as well as dynamic ones. As need arises, this set can easily be extended by creating a new nodekit based on any existing one. The new kit inherits all behaviour from its predecessor and only adds what is missing.

An exemplary oscillation nodekit is given in Figure 5.5 for an oscillating *shuttle node*. Motion is thereby controlled by a path with two endpoints, which are specified by translations. The oscillation frequency is noted as a *float* value for the number of cycles per second. Other motion definitions are integrated in the nodekit structure analogously.

A *separator* node is used for encapsulation of motion with regard to the remaining scene. This node saves all configuration of the rendering system's status machine. After successful traversal of the nodekit scene graph, this configuration is restored. This prevents motion effects to influence any information object *IO* not contained in IS_M .

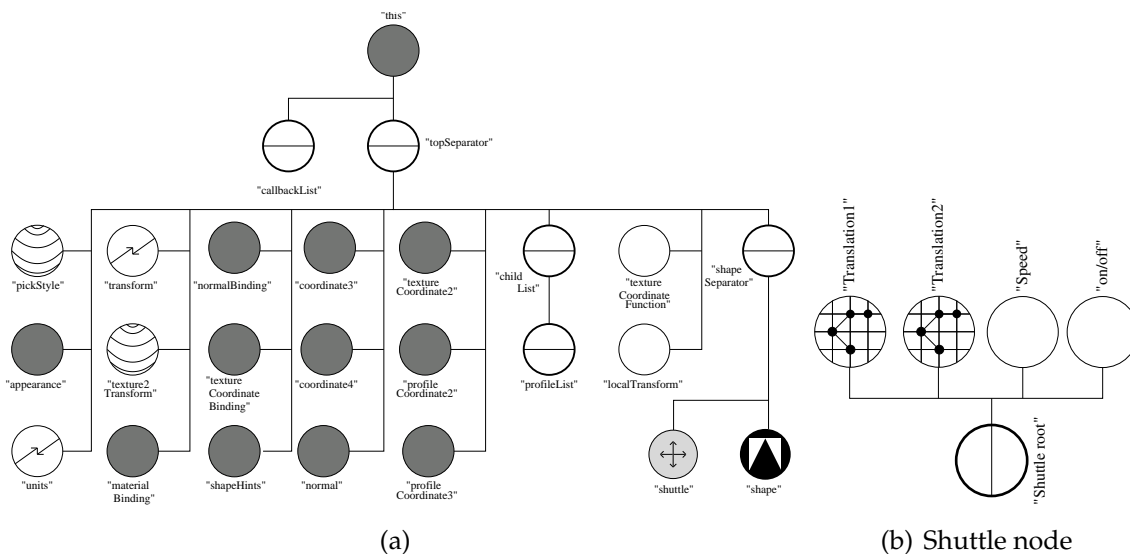


Figure 5.5: Structure of implementing oscillating motion. Subfigure (a) shows the nodekit containing an information object. The details of a shuttle node as part of this nodekit are outlined in (b).

5.2.2 Motion by Structural Change

The general layout of handling information objects $IO_i \in \mathbf{IS}_M$ in the scene graph is the same for motion by structural changes as for oscillating motion as described above. The motion is achieved by manipulating individual vertices of the geometry subgraphs. The concrete vertex manipulation is based on evaluating a motion function $f_v(v)$ which operates on a vertex v . This evaluation is done by use of an *engine*. Such an engine takes a set of input values, uses a transfer function description to operate on these values, and outputs a set of resulting vertex positions. Figure 5.6(a) shows the scene graph layout for this process.

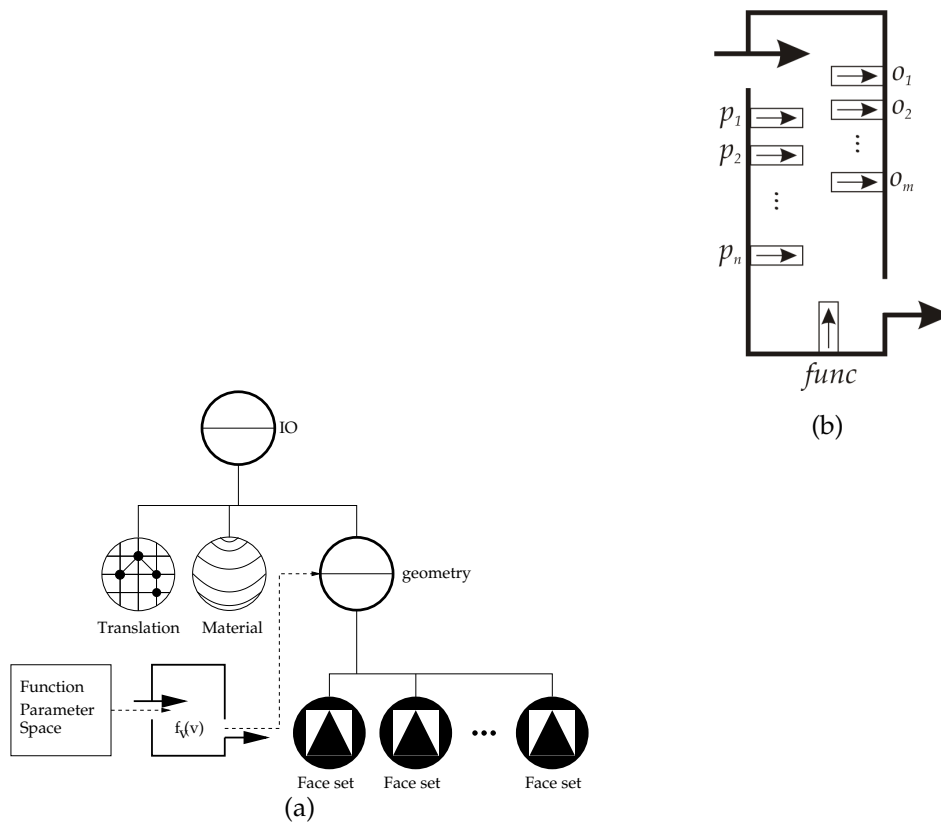


Figure 5.6: Scene graph layout for motion by structural change. Subfigure (a) shows the general scene graph including the information object which is subject to motion as well as the engine causing vertex modifications. The engine layout for free form function evaluation is shown in some more details in Subfigure (b).

A more detailed view of the engine used for determining vertex transformations is shown in Figure 5.6(b). The engine allows for a set of input parameters. These form a parameter list PL :

$$PL = \{p_1, p_2, \dots, p_n\}.$$

Furthermore, a free-form definition of the vertex manipulation is provided by func-

tion *func*. Combining both of these inputs allows the engine to produce a set of outputs forming the result list *RL* which is used for parameterising motion of vertex *v*:

$$func(PL) = RL = \{o_1, o_2, \dots, o_m\}.$$

For the specification of *func*, a variety of primitive functions may be used that operate on *PL*. The concrete set of such functions is based on using the class *SoCalculator* from *Open Inventor* for implementing the free-form function evaluation. Basically, the following list builds the functional basis:

Scalar functions

$\cos(p_i)$	cosine function (p_i in radians)
$\sin(p_i)$	sinus function
$\tan(p_i)$	tangent function
$a \cos(p_i)$	arc cosine function
$a \sin(p_i)$	arc sinus function
$a \tan(p_i)$	arc tangent function
$a \tan 2(p_k, p_i)$	arc tangent function of two variables (p_k, p_i).
$\cosh(p_i)$	hyperbolic cosine function
$\sinh(p_i)$	hyperbolic sinus function
$\tanh(p_i)$	hyperbolic tangent function
$\text{sqrt}(p_i)$	square root function ($\sqrt{p_i}$)
$\text{pow}(p_i, p_k)$	p_i raised to the power of p_k
$\text{exp}(p_i)$	e to the power of p_i
$\log(p_i)$	natural logarithm of p_i
$\log_{10}(p_i)$	base-10 logarithm of p_i
$\text{ceil}(p_i)$	rounds p_i upwards to the nearest integer
$\text{floor}(p_i)$	rounds p_i downwards to the nearest integer
$\text{fabs}(p_i)$	absolute value
$\text{fmod}(p_i, p_k)$	remainder of dividing p_i by p_k
$\text{rand}(p_i)$	pseudo-random value between 0 and 1

Vector functions

$\text{cross}(p_i, p_k)$	cross product of p_i and p_k
$\text{dot}(p_i, p_k)$	dot product of p_i and p_k (returns scalar value)
$\text{length}(p_i)$	length of p_i (returns scalar value)
$\text{normalize}(p_i)$	returns normalized version of p_i
$p_i[p_k]$	access components in p_i (p_k should be a scalar value in the range $[0, 2]$)

Using these primitive functions, a mapping of $PL \rightarrow RL$ can be defined. In case, this

functional basis is not sufficient for some specific application context, the concrete engine instance may be substituted by a more appropriate reimplementaion as long as it conforms to the *engine* interface provided by *Open Inventor*.

5.3 OpenNPAR Integration of Dynamic Non-Realism

This section contributes a discussion of the implementation of dynamic changes of rendering styles as introduced in Section 3.4. As the implementation basis, *OpenNPAR* is used. By extending this already existing system, a variety of implemented rendering styles can be used for creating sophisticated dynamic presentation patterns.

Most parts of this extension to *OpenNPAR* are implemented as part of a diploma thesis rendered in the context of this work [Nettelbeck, 2003]. Part of this work has also been published otherwise [Jesse and Isenberg, 2003].

For the purpose of clarifying the implementation basis, Subsection 5.3.1 points out some fundamental information about an essential data structure used throughout the *OpenNPAR* system. While this discussion includes all relevant information needed for the dynamic extensions discussed in Subsections 5.3.2 and 5.3.3, it is kept as brief as possible.

5.3.1 Fundamental Data Structure

The basis for using any NPR style in this system is a photorealistic rendition of the scene [Halper, 2003, ch. 3]. To allow for fast edge lookup, a data structure is used here that provides local neighbourhood information for individual edges and their affected faces. Baumgart [1975] presents the *WingedEdge* data structure as meeting this criteria. Approaches of continuously maintaining such structure are discussed by Glassner [1991]. Figure 5.7 shows a schematic view of the *WingedEdge* data structure as presented by Glassner.

The structure is composed of four types of elements:

1. *Shapes*: are defined as compositions of instances of the three types below. A shape represents the geometry of an information object.
2. *Faces*: are defined by lists of edges. For efficiency reasons, some supportive data (such as normals) and functions (such as `getCenter` and `directionTurn`) are stored along with each face.
3. *Edges*: separate faces from each other. For this purpose, references to the faces are stored with each edge.

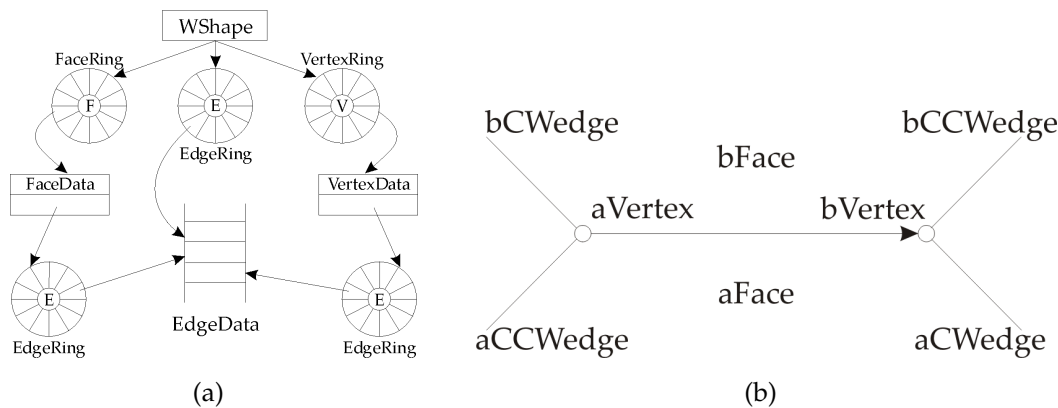


Figure 5.7: The *WingedEdge* data structure as introduced by Baumgart [1975]. Subfigure (a) shows a schematic view of the *WingedEdge* data structure [Glassner, 1991, p. 192]. The information stored along with each edge in the structure is shown in Subfigure (b).

4. *Vertices*: are defined by their positions and normals. In addition, each vertex keeps track of the shape it belongs to, stores all edges around this vertex, and maintains a radian map. For each pair of edges, this map contains the angle in between in radians.

Faces, edges, and vertices are stored in respective lists. These lists are represented in the figure as rings which make the double-linked character of the respective lists obvious.

5.3.2 Use of Hybrid Rendering Styles

Before the implementation of a dynamic transition of multiple rendering styles is discussed in Subsection 5.3.3, use of more than one rendering style in a scene based on *OpenNPAR* is presented here. For this purpose, the following paragraph introduces *OpenNPAR*'s way of defining non-photorealistic rendering styles. This is followed by a discussion of an extension of this behaviour which allows to employ multiple styles simultaneously.

Exclusive Use of a Single NPR Style

In order to apply an *OpenNPAR* rendering style to an information object in a scene, its respective scene graph needs to be prepared accordingly. The initial geometry graph is preserved and integrated into a wrapper scene graph. This wrapper consists of an *WingedEdge* containing the geometry as a subgraph as well as NPR supplements. The preparation process furthermore adds a material node for each scene object in case it does not yet exist. This node is part of the geometry subgraph and holds responsible for blending styles.

The general scene graph layout for rendering of non-photorealistic images using *OpenNPAR* is shown in Figure 5.8. The *NPR line rendering* subgraph represents the stylisation pipeline allowing for a variety of styles to be used. As an example, stylised silhouettes as discussed by Isenberg et al. [2003] are employed here. The silhouette edges need to be detected first. After concatenating the edges appropriately, hidden lines are removed. This is followed by a parameterisation of stroke styles. Finally, the strokes can be rendered.

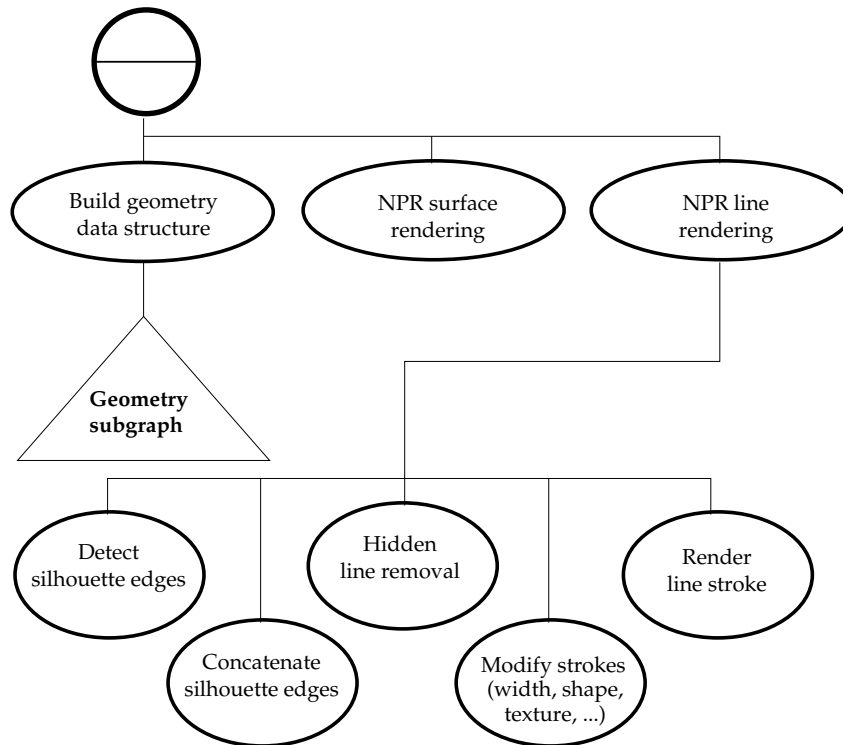


Figure 5.8: General scene graph composition for presentation in a non-photorealistic style using *OpenNPAR*.

A set of *OpenNPAR* nodes holds responsible for maintaining the *WingedEdge* data structure:

- *SbWingedEdge*: is the data structure itself.
- *SoGenerateWingedEdge*: creates an instance of the structure and saves it for future references.
- *SoWingedEdgeElement*: keeps a pointer to the instance of the *WingedEdge* data structure as created by *SoGenerateWingedEdge*.

In case a new instance of the *WingedEdge* is created, the pointer stored in *SoWingedEdgeElement* is overwritten. This allows to automatically use the most recent in-

stance of an *WingedEdge* for rendering. Figure 5.9 shows this interplay of the above nodes.

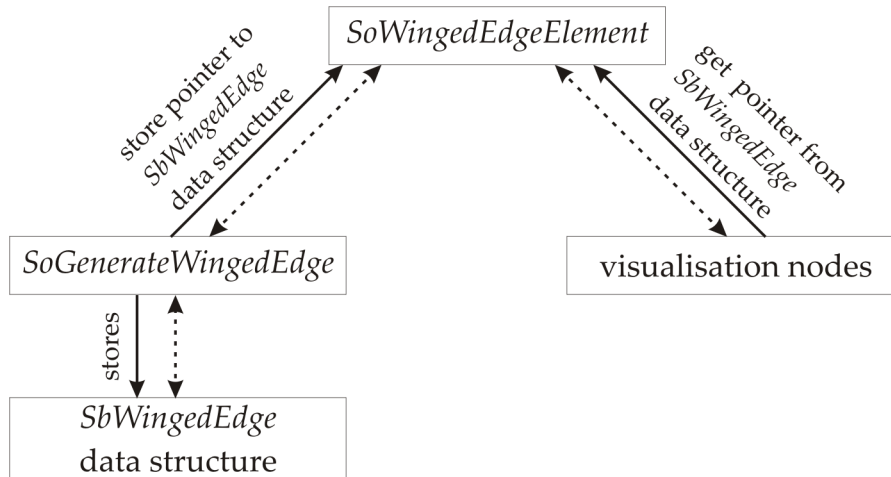


Figure 5.9: Interplay of the nodes *SbWingedEdge*, *SoGenerateWingedEdge*, and *SoWingedEdgeElement*. The dashed arrows represent processing data of the *SbWingedEdge* data structure.

Simultaneous Use of Multiple NPR Styles

In order to employ more than a single NPR style in a scene, the scene graph from Figure 5.8 is extended as shown in Figure 5.10. The geometry subgraph is divided into a number of further subgraphs. For each of these, a separate *WingedEdge* data structure is constructed. This allows to store geometry data for all respective objects in the scene. Thereby, the scene graph layout is designed such that hierarchical object handling is possible. This ensures a maximum of flexibility with regard to applying individual styles to either whole objects or parts of an object that belong together.

For each rendering style, a separate style pipeline is to be put into the scene graph. The set of geometry data nodes is referenced by separate selection nodes. These map a style onto its respective object or object hierarchy. Parameterisation of the individual styles remains as described above, but is now independent from each other.

A set of new nodes is provided for *OpenNPAR* to allow multiple rendering styles to be used simultaneous in a scene:

- *SoWingedEdgeListElement*: holds pointers to *SbWingedEdges*.
- *SoStoreWingedEdge*: stores each pointer as created by a *SoGenerateWingedEdge* node in an instance of *SoWingedEdgeListElement*.

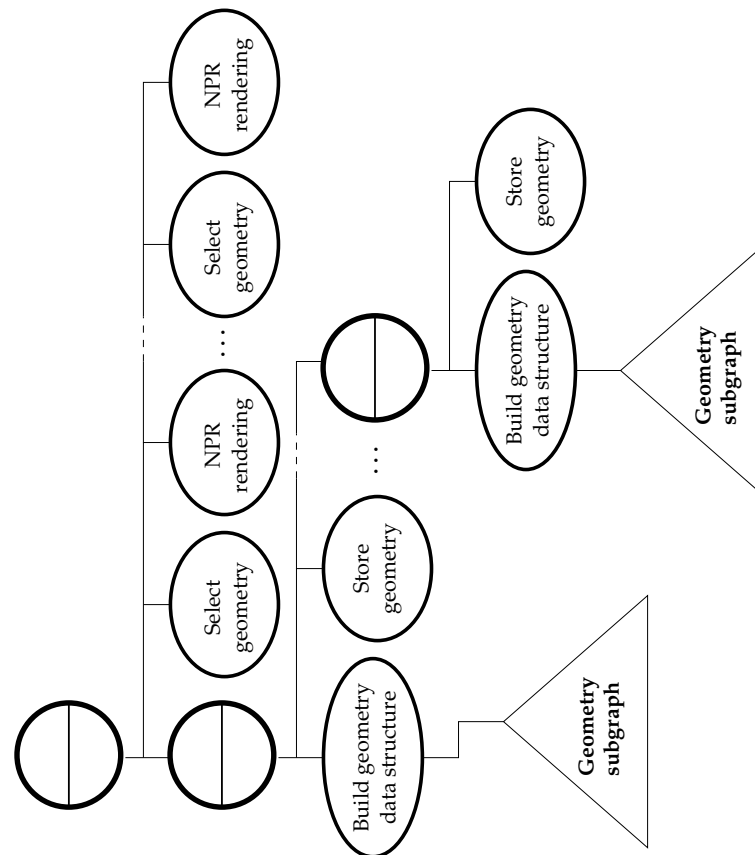


Figure 5.10: Scene graph extended to allow for multiple styles being used simultaneously as well as to apply a style to parts of the scene only. The *NPR rendering* subgraphs follow the description presented in Figure 5.8.

- *SoSelectWingedEdge*: contains the address of an associated *SoStoreWingedEdge* node. This address is used to access the *SoWingedEdgeListElement*, retrieve the pointer of the respective *SbWingedEdge* and store it in a *SoWingedEdgeElement*.

Figure 5.11 shows the interplay of these nodes on the basis of the scenario shown in Figure 5.9 above. The figure points out how to overcome the shortcomings of each *SoWingedEdgeElement* storing only the most recent pointer to a *SbWingedEdge*. The new nodes allow to take this pointer, store it permanently, and provide access as demand arises. The general interface of handling the different classes used for accessing the *WingedEdge* data structure is not affected by the new nodes. Therefore, this extension is easily integrated into existing systems whereby legacy code is ensured to be supported as well.

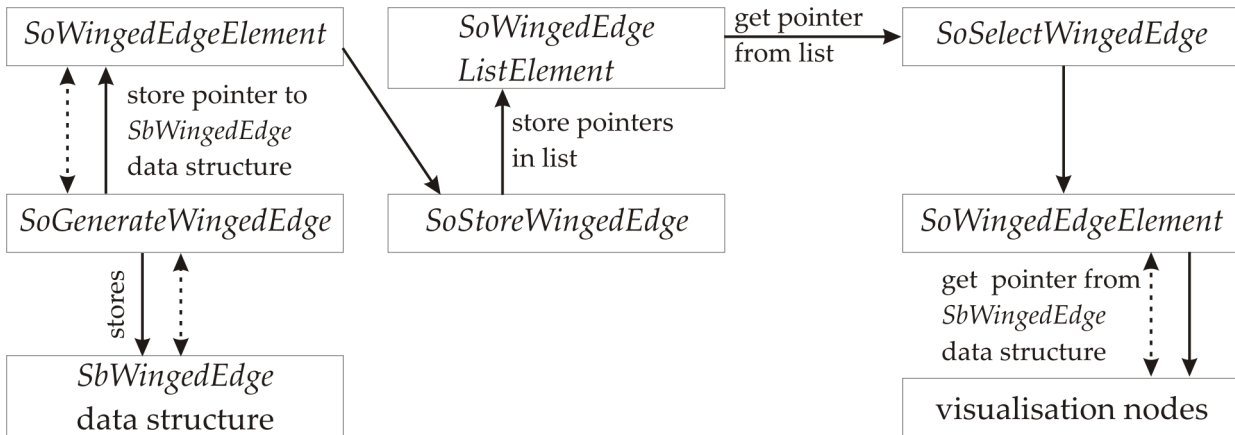


Figure 5.11: Interplay of the nodes maintaining the *SbWingedEdge* data structure. This is an extension to the scenario of Figure 5.9 for handling multiple rendering styles simultaneously.

5.3.3 Transition Between Styles

The information structure \mathbf{IS}_{RS} holds all information objects that are subject to a dynamic presentation of multiple rendering styles. Dynamics are thereby achieved by blending from one style to another as described in Section 3.4.1. In order to blend the presentation of an information object $IO_i \in \mathbf{IS}_{RS}$ from one rendering style to another, the object is rendered twice: once for the source style and once for the target style.

Two blending approaches are used for dynamic rendering styles:

1. blending by variation of transparency and
2. blending by adjustments of an object's emissive colour.

Both kinds of transition are achieved by use of a *Transition Engine*. This engine is used to provide a set of control values affecting either transparency of an object's material, its emissive colour, or both. Figure 5.12 points out this interplay. The engine uses some temporal parameterisation values as input. These are transformed into two different kind of outputs:

1. control values for the source rendering style and
2. control values for the target rendering style.

That is, the output fields of the engine are connected to the material node of the sub scene graph used for rendering the source style as well as to the material node which holds responsible for rendering the target style. Depending on the type of transition, these material nodes are then modified automatically.

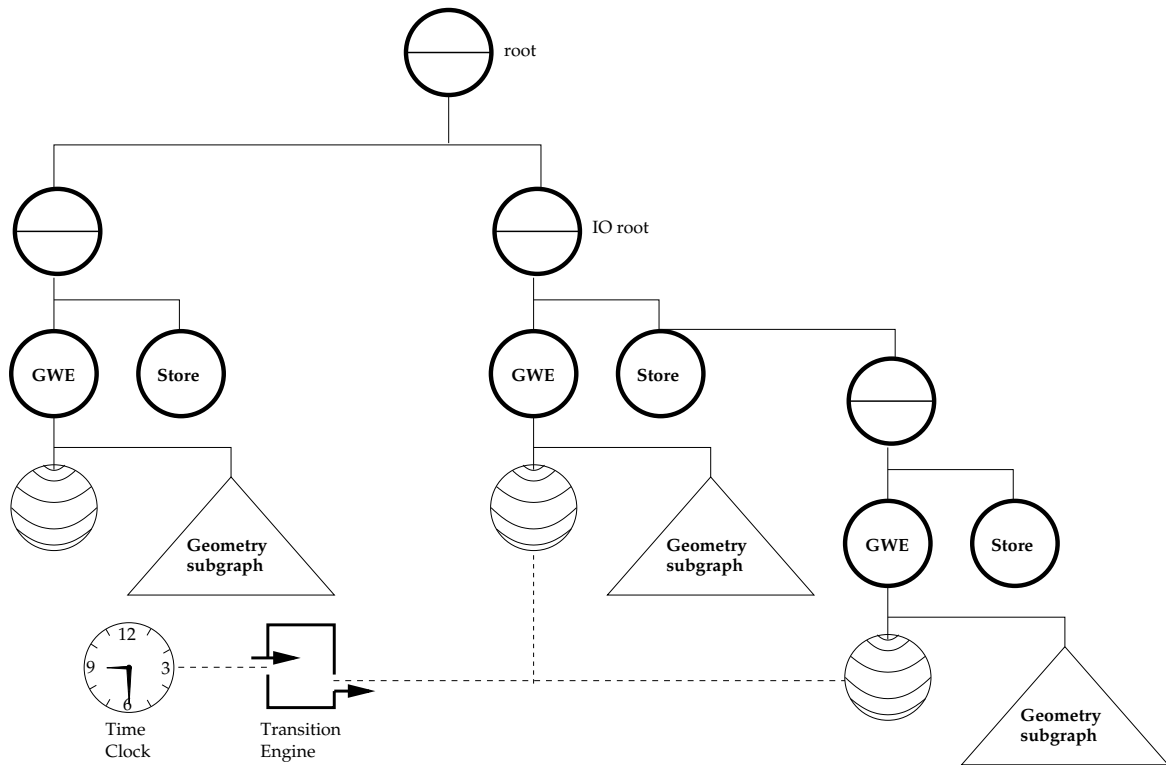


Figure 5.12: Achieving dynamic rendering styles by use of a *Transition Engine*. This engine provides control values for continuously adjusting transparency and emissive colour values throughout the transition process.

In case of a transparency-based transition, the respective materials' α -elements are connected with the engine. Whereas the transparency of the source style is gradually increased, transparency of the target style subgraph decreases. In case the transition is complete, the engine connection for both styles are reversed and the transition is restarted.

If a transition is based on changes in an object's emissive colour, a pair of engine outputs is used for each rendering style subgraph. This pair consists of:

1. a *vector* for changing the light emission and
2. a *scalar* for adjusting the α -channel of the source style subgraph.

Adjusting the transparency of the source rendering style is used to avoid overlap of both styles. Furthermore, using this pair of engine outputs allows to merge both kinds of transition in order to overcome the see-through effect as discussed in Section 3.4.1.

5.4 Temporal Server

The principle of evaluating temporal presentation constraints is discussed in Chapter 4. Here, a *Temporal Server* component is presented that is part of a framework of *temporally influenced geometry presentation (Tigeop)*. This framework furthermore consists of a specification of parameterisation documents that build the foundation of a script-based controlling of dynamic presentation techniques for illustration purposes. These documents are described in Appendix A.

First of all, the design and layout of the server component is presented. This is followed by a discussion of its core: the *Dynamics Management Unit*. A presentation of the web service interfaces provided by the framework rounds up this section.

5.4.1 System Design

The overall layout of the temporal server is shown in Figure 5.13. This system consists of a set of components build around the *Dynamics Management Unit (DMU)*. The *DMU* holds responsible for evaluating temporal constraints of a presentation and scheduling of individual presentation techniques. Its working principle is described in the following subsection.

Other parts of the server layout include a set of catalogues used for maintaining available dynamic presentation techniques as well as intervals valid throughout the duration of an illustration. These catalogues are referenced as *Set of Dynamics D* and *Interval Catalogue*, respectively. Whereas the set of available dynamic presentation techniques is not modified by the *DMU*, the interval catalogue is constantly updated.

The design of handling concrete instances of illustration clients follows Silva et al. [1997]. They present a *Proxy* design pattern in a client/server-based environment. This pattern is employed here to construct and maintain a list of clients currently operating in the context of the *Tigeop* framework.

Communication between different instances of illustration clients and the *Temporal Server* is done on a *Web Service* basis. Following the discussion presented by Stal [2002], this provides the basis of employing a wide variety of illustration client systems. All kinds of illustration systems should be able to be connected to the *Temporal Server*. This flexibility is provided by using a set of well-defined documents to bridge the interface-gap between such illustration clients and the server. Subsection 5.4.3 below discusses the principles of this communication in some more detail. Before this is done, the following Subsection 5.4.2 points out specifics of the *Dynamics Management Unit*.

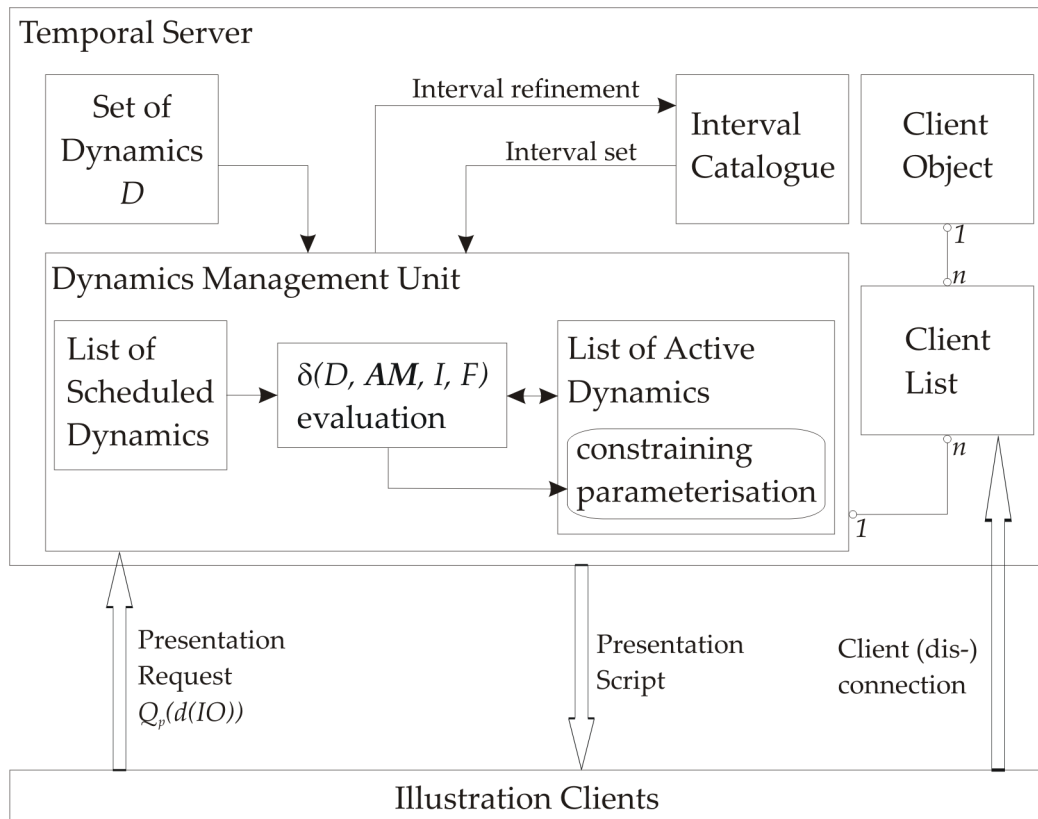


Figure 5.13: Layout of the *Temporal Server* for temporal control of presentations on a script basis.

5.4.2 Dynamics Management Unit

The *Dynamics Management Unit (DMU)* builds the core of the *Temporal Server*. It evaluates the fundamental temporal constraints as presented in Chapter 2 by use of the parameterisation function δ presented in Section 4.4. Thereby, it controls the temporal parameterisation of any illustration output. This is done on two levels:

1. Respective dynamic illustration techniques are to be specified; and
2. The temporal parameters for these techniques need to be determined.

For temporal constraining purpose, the *DMU* maintains a list of scheduled dynamic presentation techniques. Each time, an illustration client sends a request $Q_p(d(IO))$ of using dynamics for the presentation of an information object IO , one out of four possible operation reactions is applied to the object by the *DMU*:

1. *Pass*: The request $d(IO)$ is allowed immediately and the dynamic presentation may be used without any further constraints.

2. **Constrain:** The request is allowed but the dynamic character of the presentation technique $d(IO)$ in question is constrained. This effects dynamic characteristics such as the frequency or type of d .
3. **Schedule:** The request is not passed immediately but the presentation of $d(IO)$ is added to the queue of scheduled dynamics.
4. **Regret:** The request for $d(IO)$ is not allowed to pass.

Temporal mapping is done based on an analysis of a given set of intervals. This is retrieved from the overall interval catalogue maintained by the *Temporal Server*. Interval consistency is checked by δ as outlined in Section 4.4.

Overall, by use of δ , the *DMU* combines the set of scheduled dynamics and the set of intervals for the purpose of updating and parameterising the set of currently active dynamics. For each illustration client connected to the server, this results in a constrained presentation using dynamics such that their expression potential is used up to its maximum whereas cognitive overload due to its over-excessive use is avoided.

5.4.3 Web Service Interface

Context

Distributed applications are characterised by consisting of a set of individual application components. Typically, these components interplay on the basis of middleware technologies such as *Corba* [Dogac et al., 1996, Roantree et al., 1996], *HLA* [Lorenz et al., 1997, Lorenz and Ritter, 1997, Straßburger et al., 1998, Jesse and Schumann, 2000], or *Grid Services* [Foster et al., 2002, Furmento et al., 2001].

An alternative approach to combine a set of application components is regarded to as a *web service environment* whereby the individual components are referred to as *web services* [Yang and Papazoglou, 2002]. In order to allow for interoperability of these individual services, a communication interface protocol is needed. One such protocol is *SOAP* as defined by the W3C.

SOAP is a protocol for the exchange of structured XML documents in distributed environments. It is described by a set of four documents:

1. The *Primer* [Mitra, 2003] targets the interested end user and provides a general overview of *SOAP*.
2. The *Messaging Framework* [Gudgin et al., 2003a] targets developers. Details of *SOAP* messages are documented as well as the request/response mechanism and the possibility to transport *SOAP* messages by use of arbitrary protocols.
3. The *Adjuncts* [Gudgin et al., 2003b] point out optional protocol extensions. The transport of *SOAP* messages via HTTP is documented here as well.

4. Finally, the *Assertions and Test Collection* [Haas et al., 2003] complete the protocol specification by documenting numerous test cases for the validation of *SOAP* processors.

Overall, the documents describing *SOAP* provide an open interface description for web services. On this basis, flexible frameworks can be constructed which provide adaptivity for new application areas as demand arises. For this reason, web services form the basis of defining the interface for the *Temporal Server*.

Layout of Documents

A set of three documents describes the *Tigeop* service:

1. The *Tigeop implementation description* provides information about accessing the *Tigeop* service from within illustration applications.
2. The *Tigeop interface description* documents information about the service provided by the *Temporal Server*.
3. The *Tigeop Schema* describing the structure of documents as provided by the *Temporal Server*. These documents control an illustration enhanced by dynamic presentation techniques.

The first two documents are instances of the *Web Service Description Language (WSDL)* as defined by Christensen et al. [2001]. Such *WSDL* documents consist of four different elements:

1. *Messages*: specify structures and types of operations provided by the web service.
2. *Bindings*: describe the protocol implementation of the web service. The common basis for bindings is the *SOAP* protocol.
3. *PortTypes*: define the interface of a web service in terms of operations and their input and output messages.
4. *Ports*: define the bindings used by the system.

The concrete instances of the two *WSDL* documents describing the *Tigeop* service are shown in Appendix A.3.2 for the implementation description and A.3.1 for the interface description. This latter description follows the script as discussed in Section 4.5 whereas the former document simply defines access information for the server.

The output script constructed by the *Temporal Server* follows the syntax described as *XML Schema*² as defined by the W3C. The *Tigeop* Schema is shown in detail in Appendix A.1. It contains the three following elements: a set of employed dynamic presentation techniques \mathcal{D} , a reference to the geometric illustration model, referenced as *scene description*, as well as a set of object presentation definitions.

5.5 Workbench for Information Fusion

The process known as information fusion targets retrieval of information from a set of heterogeneous data sources. First of all, some context information about fusion is given in the next subsection. Secondly, a set of fusion requirements is derived. These requirements motivate the individual implemented components of the workbench for information fusion. Layout of these components is presented in an architectural fashion. This architecture is accompanied by a description of integrating the *Temporal Server* for handling temporal fusion characteristics. Further details on using information fusion will be presented in the following chapter at the example of a set of applications.

5.5.1 Context

Given the current state of the art in database technology and modern communication networks large volumes of data cannot only be stored and managed but are subject to the providence of world-wide distribution and access. The resulting increase in the number of available information sources leads to what users perceive as information overload. Information providers often represent their data in a heterogeneous fashion regarding structure and semantics. The task of presenting deduced information can be derived. Attempts have been made to access varying data sources via a uniform user interface, thereby hiding the heterogenous nature of the underlying data [Bellgard et al., 1999, Fischer et al., 1999]. These efforts are restricted to specific application domains and are not generally applicable. In addition, they do not answer the open question of limited expression capabilities provided by the set of classic presentation variables like colour, shape, transparency, and position of an object.

Information fusion is seen as a process of integrating heterogeneous data sources, examining stored data, and extracting possibly hidden information [Dunemann et al., 2002b]. It helps to handle and access potentially large data sets. For that purpose, the fusion process is defined as a concatenation of various fusion operators. These operators include but are not limited to database integration, machine

² Information about XML Schema, according tools, and documentation is available at <http://www.w3.org/XML/Schema>.

learning, and data mining techniques. The execution of the fusion process is of iterative nature and controlled by user interaction. Behaviour of information fusion as an interactive process is characterised by temporal parameters regarding two aspects: user interaction and process execution.

5.5.2 Requirements

The objective of information fusion results in important requirements of methods and techniques for interactive information exploration and manipulation. Though various potential fusion applications result in different requirements, a set of tasks can be identified which are similar for a wide range of fusion applications [Jesse and Strothotte, 2001]:

Data access: At first, we have to support a uniform access to different sources. This involves usage of database gateways in order to hide the heterogeneity, to access Web sources via the appropriated protocols, and extracting semistructured data from these sources, as well as query translation, optimisation, and processing.

Data integration: An integrated view should represent data from the different sources in a homogeneous model. This involves mending conflicts at schema or instance level and dealing with aspects of data quality. In addition, inter-source relationships have to be represented and managed at the global layer.

Analysis and abstraction: Filtering or condensing data and extracting dependencies or abstractions offers the opportunity to yield information of a new quality. The notion of new quality depends on the concrete application.

Presentation and processing: The discovered information has to be presented according to the problem domain or be prepared for further processing. The illustration needs to allow a user exploration of the information space and offer ways of interactive manipulation of the fusion process as well as of the fused data.

Representation of meta-information: An important prerequisite for fusion is the existence of information about sources, fusion objects, and the problem domain. Such meta-information should be managed by the system and updated or extended during the fusion process, e.g. for optimisation and interactive exploration purposes.

This section concentrates on the presentation and processing stage. Of interest are aspects of illustration and its specific characteristics in the fusion process. Information fusion is a process which is heavily influenced by user interaction. On one

hand, users need to decide on kinds of fusion operations to apply. On the other hand, the process of fusion needs to be presented in a comprehensible manner. After all, a possibility is needed to modify single aspects of the fusion process.

Access to such fusion characteristics is provided by illustrations. Navigation techniques for exploring complex information spaces are outlined by Strothotte [1998]. Special emphasis is put on various zooming methods, including fish-eye views, distortions, and displacements. Such techniques along with individual implementations of dynamics may be integrated as part of a workbench for information fusion. This characterises such a workbench as an implementation testbed for dynamics as presented earlier.

5.5.3 Components and Layout

To provide a testbed for user-centred dynamic presentations in support of information fusion, an architecture has been developed [Dunemann et al., 2002a, Jesse et al., 2001, Dunemann et al., 2001, 2002b]. This architecture is shown in Figure 5.14 and allows to fulfil the requirements of information fusion as presented above. This architecture basically consists of two main components:

1. *InFuse*: as a database-centred and component-based middleware system. This system includes with *FraQL* [Sattler et al., 2000] a query processor for data management of heterogeneous sources. On top of this query processor, a *fusion engine* holds responsible for processing and managing meta data of a fusion process.
2. *Angie*: provides different means of illustrating information about the fusion process, its composition, and results. For this purpose, *Angie* is composed of a set of plugins for access to a variety of illustration techniques. These include dynamics.

Dividing up the architecture into these two main components allows to address the listed fusion requirements in a flexible manner. The fusion process consists of several steps which partially depend on each other. The *fusion engine* allows to manage the components and information needed for this flexibility. The definition and persistence of processes is handled as well as control of process execution. Supportive process information is stored in the *meta data repository*. A set of information gathering operators is available within *InFuse* as well. These operators act on data at various stages throughout fusion process execution and perform operations such as data mining or machine learning algorithms.

The front end *Angie* is designed component-based as well. It basically consists of a core application holding responsible for window management as well as user interaction, and a set of plugins providing illustration techniques. These plugins

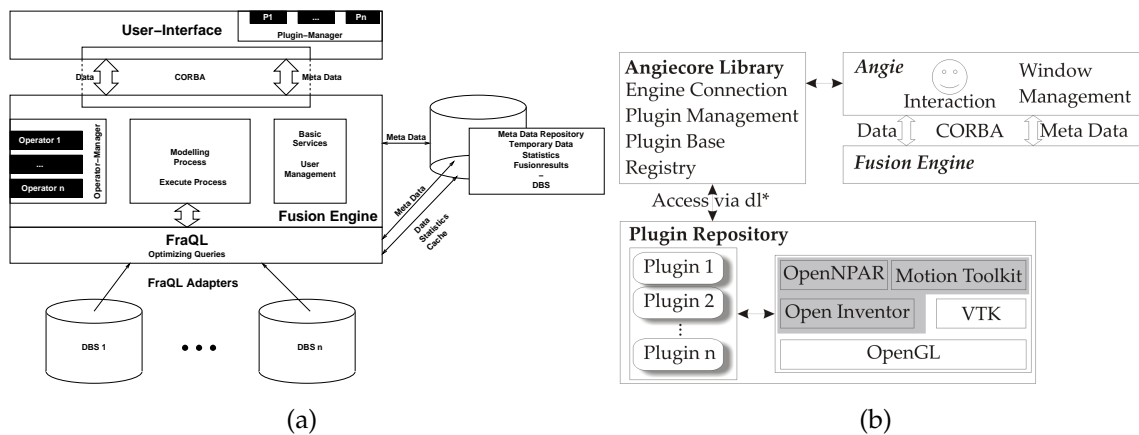


Figure 5.14: Architectural layout of the *Workbench for Information Fusion* [Dunemann et al., 2002b]. Subfigure (a) shows the global architecture of the underlying *InFuse* layout whereas (b) shows the layout of the front end *Angie* integrating dynamic illustrations.

provide an interface to the individual implementations of dynamic illustration techniques presented earlier throughout this chapter. This way, *Angie* characterises the workbench as an integration of individual and independent techniques. In addition to this integration, the *visualisation toolkit (VTK)* [Schroeder et al., 1998] adds to the set of techniques employed by the various plugins. *Corba* is used for communication between *InFuse* and *Angie* to provide a maximum of interface flexibility.

5.5.4 Interaction and Temporal Fusion Characteristics

A fusion process is characterised as a composition of multiple fusion operations as sketched in Figure 5.15. These operations are arranged in varying order as any given operation possibly depends on results produced by another operation. Such dependencies and respective operation scheduling result in temporal information inherent to the fusion process execution. To reflect these information in the execution graph, temporal components may be inserted as shown in Figure 5.16.

User interaction is expressed by events which occur consecutively. Each interaction event might cause some action that changes the current state of the system. This change is valid until another user event occurs or until it is outdated by system progress. The effect of any user based event is therefore valid for a specific time interval. Therefore, any supporting illustration technique needs to be temporally parameterised to ease interaction and follow its temporal constraints.

The temporal nature of the fusion process execution itself results out of the execution of fusion operators. Two operators with a dependency between them are always executed consecutively. Independent operators can possibly be executed in parallel. The order of operator execution results as an acyclic graph. At every execution step, intermediate results are gained. These may be illustrated in order to

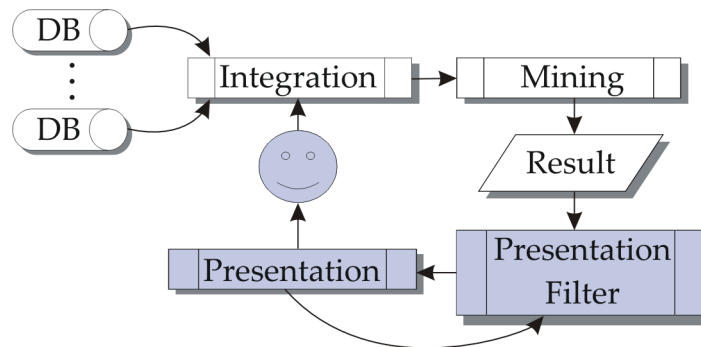


Figure 5.15: Gaining information in a fusion network. This graph layouts the principle structure of an exemplary information fusion process.

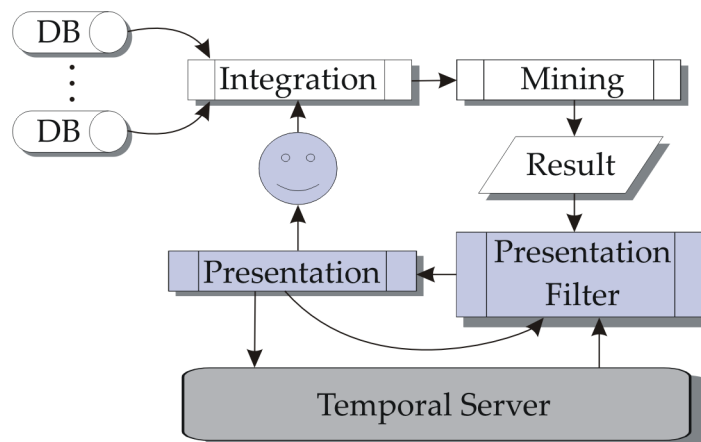


Figure 5.16: Extension of the fusion execution network from Figure 5.15. The network is enhanced by temporal components controlling the result presentation based on an analysis of process execution supported by the *Temporal Server*.

support the user in interacting with fusion process execution. The temporal component of each result set closely correlates to the time dependent position of its causal operator in the execution graph. Some results are gained earlier than others. Some results are kept valid for longer periods whereas others need to be revised as parameterisation of their corresponding operators is to be changed depending on user decisions. The completeness of a result set needs to be expressible by an accompanying illustration. Temporal influence on presentation variables is suitable for complying to this requirement.

5.6 Summary

This Chapter contributes implementation issues regarding dynamics as discussed in Chapter 3 as well as a supportive *Temporal Server* based on the discussion from Chapter 4. In doing so, this chapter bridges from the two conceptual chapters 3

and 4 to the application-oriented part of this thesis. The following chapter deepens this application part further by presenting some examples of using the implementations shown here to construct illustration scenarios enriched by dynamic presentations.

6 Applications

A set of different applications is shown here for the purpose of pointing out possible uses of dynamic presentation techniques for illustration purposes. The examples shown span a broad range of application fields. First of all, dynamic rendering styles are applied to a presentation for reflecting dynamic behaviour of the illustration model. Thereby, model dynamics are gained by retrieving illustration information from a search engine. This is followed by the discussion of a set of applications framed by the common context of information fusion. This set contains use of the motion-enhanced information mural for illustration of climate data, a scatter plot enriched by motion to reveal underlying data characteristics, and construction of an abstract dynamic landscape in support of online aggregation. Finally, a set of illustrations is shown that address exploration of geometric models or parts thereof. These models are of technical nature as well as stemming from the field of anatomy.

The contribution of this chapter is in pointing out the potentials of dynamics for enriching illustrations. Thereby, use of dynamics enhances the expression set of the shown presentations. On one hand, this supports illustration of dynamic behaviour of illustration models. On the other hand, dynamics extend the means of directing user attention to specific aspects of an illustration model.

6.1 Enhancing Illustrations with Search Engines

Classic illustration systems are often enriched by manually prepared model annotations. Subject of current work is the automatic coupling of a model with supportive external information sources. The use of a knowledge base as the fundamental source of illustration information is presented by Hartmann et al. [2002]. That is, a geometric model is illustrated whereby annotating information as well as some structural information about the model is derived from a knowledge base. The presented system is an extension of the *Text Illustrator* by [Schlechtweg and Strothotte, 1999, 2000].

However, in most cases these external annotation information sources are of static nature. Documents need to be collected and knowledge bases need to be composed. These processes are characterised as being highly interactive and dependent on manual input. This section addresses the coupling of an illustration model's objects with queries sent to a search engine. This way, the search engine is employed as a dynamic source of illustration annotations. They are dynamic in a sense that

there is mainly no manual updating of the information storage required. The retrieval of this dynamic information is used in order to control and parameterise use of dynamic presentation techniques. The selection of these dynamic techniques concentrates on fading transitions between different rendering styles.

The contribution of this section is in enhancing illustrations by accessing external dynamic annotation sources. For this purpose, dynamic presentation techniques are used. These specifically include non-realistic rendering styles. Implementation issues of this section's contribution are published as part of a diploma thesis conducted in the context of this work [Funke, 2003].

6.1.1 Scenario

For this specific search scenario, *Google*¹ is used as search engine. However, this is not a restriction immanent to the system. In principle, any accessible search engine can be used. *Google* is only chosen for practicability reasons. Besides its classic web access, *Google* provides a web service based programming interface for querying the search engine directly.² This allows for easy integration into the illustration system as presented here.

The exemplary subject of the illustration is an engine model. A realistically rendered view of this model is shown in Figure 6.1. The engine model consists of multiple parts that are of interest for the illustration process. These specifically include the cooling system (1), the flywheel (2), and the propulsion system (3). Search queries are now used for these parts in order to retrieve and finally express any possible inter-object relations and to provide some information not yet provided by the model. This may be some new annotations as well as just a keyword collection or an importance value derived from the number of retrieved search results.

6.1.2 Context on Search Engine Handling

Two techniques for the search for related pages in the World Wide Web are presented by Dean and Henzinger [1999]. However, instead of using traditional search engine queries, the authors present the search for related web pages based on their URL. This allows for the identification of related pages without a need to analyse the source web page first. A similar approach is addressed by the application presented here as the search queries to be performed are in part build on the basis of available short model labels.

A search engine for 3D models is presented by Funkhouser et al. [2003]. Their system provides access to a set of models by a query interface. Different kinds

1 The *Google* search engine can be accessed via <http://www.google.com/>.

2 Further information about *Google*'s programming interface is available at <http://www.google.com/apis/>.

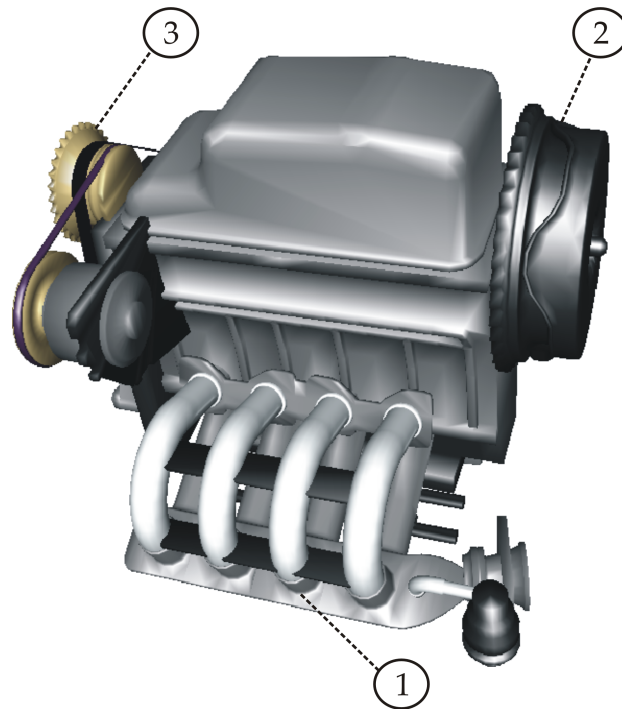


Figure 6.1: The illustration model.

of queries are accepted for model lookup: text-based queries, similarity queries or combinations out of these two. A text-based query references the model by name or some context that is possibly stored in the model database. In order to settle for a similarity query, a VRML model is to be provided and the search engine returns a set of similar models. The degree of similarity is based on a shape-analysis of the provided model. Regardless of the type of the query, the 16 most similar models found in the database will be returned by the search engine.

Challenges in the evaluation of a web search are discussed by Hawking et al. [1999]. The authors explicitly address common ranking systems as used by existing search engines. As a result of an analysis of the rankings combined with an evaluation framework, weaknesses of rankings are pointed out. The authors state, that even though modern search engine ranking systems have been subject to a steady refinement, the rankings are to be handled with care regarding their expressiveness.

A set of techniques for the presentation of multimedia web search results are outlined by Mukherjea et al. [1996]. These presentations are constructed in 3D and include a tabular listing of all results as well as a scatter plot. Both are of interactive nature which allows for an analysis of the available search results without the need to directly access the underlying web pages. At the example of the system *Grouper*, a dynamic clustering interface for web search results is presented by Zamir and Etzioni [1999]. Herein, clusters are created out of the results retrieved from the search engine. This allows to present some more meta information about results as

in case of classical search engines where the querying user is confronted with a list of document snippets. This approach is extended by the *Kartoo* meta search engine³ that presents the search results by circular glyph representations of the clusters. The size of these glyphs depends on the respective result rankings.

6.1.3 Design of the Illustration Process

Use of search engines for the purpose of enhancing an illustration of a geometric model is a highly interactive task. In support of this, the following subsection points out three elemental classes of queries useful for this task. An overview of the system's architecture is presented thereafter.

Querying for Illustration

The kinds of query to be executed can be divided into the following three classes:

Querying individual objects; For a specific object of the illustration model, a query is submitted to the search engine. This query includes the object's name or any possibly available annotation. The results retrieved can then be used in order to reflect additional information about the object.

Querying Multiple Objects; The same queries as used for individual objects can be applied to multiple objects in parallel. That is, different search queries are sent to the search engine in order to retrieve information about respectively different objects. The individual queries are sent independently from each other. That is, they are OR-combined. This allows to reveal formerly hidden inter-object relationships. Finally, it possibly leads to an observation of semantically linked object clusters in an illustration model.

External Documents; In case, external documents are available that provide some information about the illustration model, these can be included in the search scenario. In this case, some excerpts of the document—such as specifically marked keywords—are added to the set of potential search terms. The results retrieved for the document may be combined with results from both query classes above. The similarity between the results for both types of queries is then to be presented. This helps to illustrate matching of an appropriateness of the object-document relationships.

An interactive refinement of the search strategy is possible for each querying class. The presentation of this section concentrates on the first two classes. However, support of including external documents into the illustration scenario can easily be integrated in the framework as presented. A test implementation has already been

³ The *Kartoo* meta search engine can be accessed via <http://www.kartoo.com/>.

approached. Even though first results sound promising, a set of questions were raised that are subject to future work.

Architecture

Figure 6.2 presents the principle architecture of the *search illustration system*. This system is designed in a component-based manner in order to allow for easy integration of new components as well as easy reuse. As shown on the left side of the figure, this architecture integrates the *OpenNPAR* system as well as the *Temporal Server*. Both serve as parameterisation basis for the illustration client.

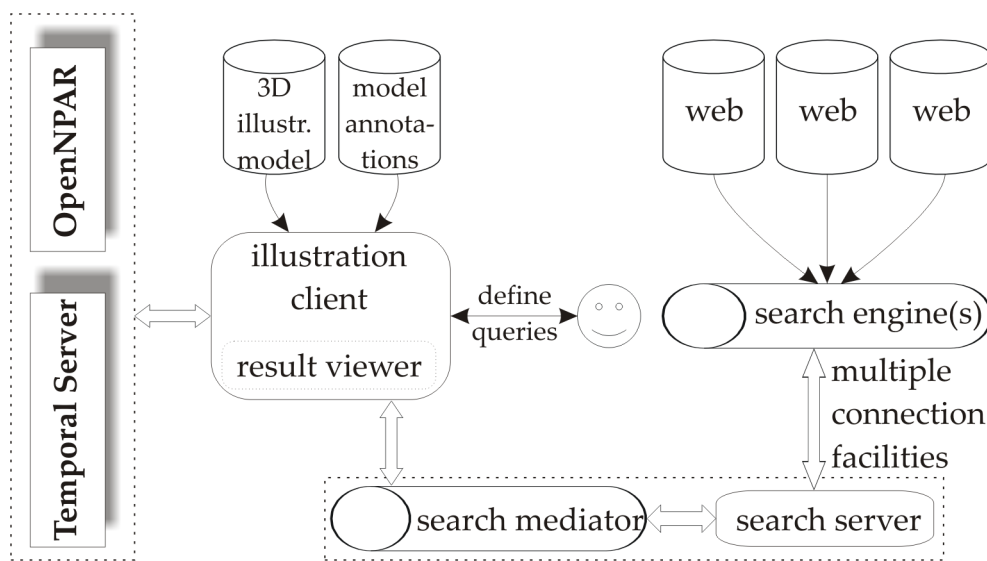


Figure 6.2: Architecture for the illustration system combining geometric model presentation with information gathering by querying search engines.

A central component of the system is the *search server*. This server holds responsible for any communication with search engines. Its design is of generic nature and allows to address a set of different search engines. However, as of now, access to *Google* has been implemented only.

The *illustration client* acts as the interface to the user. It provides a dynamically enhanced rendition of the *3D illustration model* as well as interaction facilities for the definition of queries and respective selection of objects. Therefore, the client provides the presentation of the geometric model as well as any supporting illustration information. For this purpose, it includes a *result viewer*. This viewer presents the search results in order to allow for their direct exploration by the user. The result viewer is discussed in some detail in Section 6.1.5 whereas specific rendering effects of the illustration client are subject of Section 6.1.6.

A *search mediator* bridges between the client illustration application and the server. It helps to decouple both components. This way, multiple illustration clients may

benefit from information retrieved by the search server. Furthermore, the decoupling allows to provide web service interoperability whenever desired.

The architecture allows for single components to easily fit into a web service environment. This provides for relatively fast and easy creation and integration of alternative application domains. An example is the *Temporal Server* as a component providing integration of temporal constraints in order to parameterise the simultaneous use of dynamic presentation variables.

6.1.4 Search Result Evaluation

Retrieving information from search engines results in multi-dimensional information sets. Combined with available query information these information sets form *ResultSets* as provided by the *search server*. A *ResultSet* is defined as a six-tuple (q, u, t, p, d, r) where

- q is the search term,
- u is the URL of the result page,
- t is the title of the result document,
- p is the first part of the result providing a document summary,
- d holds an excerpt from a possibly existing DMOZ⁴ entry, and
- r represents a boolean value indicating whether or not related web pages are available (specific to *Google*).

Multiple strategies may be employed for the analysis of the *ResultSets* for the sake of extracting as much information as possible [Baeza-Yates and Ribeiro-Neto, 2002, p. 99ff]. For the work presented here, the *ResultSet* elements q, u, t, p , and combinations thereof are evaluated. URL matching is done in two ways: counting occurrences of q in u and counting of multiple instances of u for separate terms of q . The latter case gets active in case the search for multiple objects results in a result URL retrieved more than once. This way, some similarity for the respective objects can be derived. The former case of URL matching builds the basis for evaluating other pairs of occurrences of q in any of $\{u, t, p\}$ as well. Besides an analysis of occurrences of q in u , the same can be determined for q in t , q in p , and combinations thereof. Matching of $q \in p$ is to be treated with a lower score than matching of $q \in u$ and $q \in t$. This values the importance of a document's location and its title compared to the document body.

Besides an analysis of the *ResultSet* contents, ranking information is used for parameterisation of search result presentation. In order to consider ranking, some con-

4 DMOZ is the Open Directory Project accessible at <http://www.dmoz.org/>. This directory is also available via <http://directory.google.com/>.

text information about results retrieved from the search engine is evaluated. The ordering of the results is not given arbitrarily. Instead it represents some information about the relevance of the results with regard to the query submitted. This ordering varies from search engine to search engine. Google's ranking system is explained to some degree by Page and Brin [1998] as well as Calishain and Dornfest [2003]. This ranking system is popularity-based, as a web page gets ranked high in case it is often referenced by other pages.

6.1.5 Search Result Illustration

In addition to using search results for rendering parameterisation as will be presented in Subsection 6.1.6, they are illustrated in a result viewer. This allows to provide some background information on the dynamic illustration as well as the ability for the user to examine the results by itself. As both—the dynamic illustration presentation and the search results—correlate with each other, the result viewer supports the user in crossing the cognitive gap of resolving these relations. For this purpose, elements of the illustration model which are subject to a search query are presented along with their respective search results.

The result viewer follows the principle idea of the *Kartoo* meta search engine. That is, the viewer does not simply list all retrieved search results on a textual basis. Instead, search results regarding their respective objects of the illustration model are evaluated and a graphical representation of this analysis is created. Figure 6.3 illustrates this.

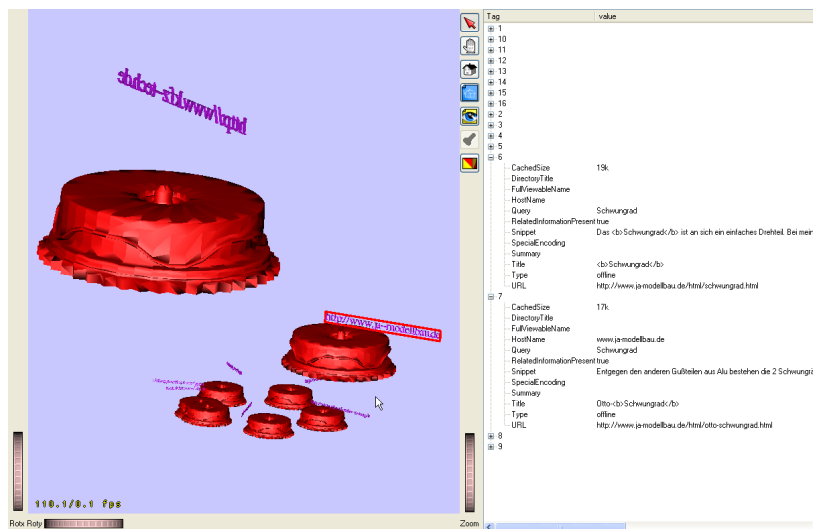


Figure 6.3: Screenshot of the *result viewer*.

At the left side of the viewer, a graphical representation of the search results is shown. This scene always represents the search results retrieved for a single object.

Multiple instances of the viewer may be opened at any time in order to investigate results for different objects. To allow for a detailed in-depth analysis of the presented results, the right hand view of the result viewer window lists all results textually. A complete listing of all search result data possibly exceeds the available presentation space. Therefore, only relevant parts are shown. The relevance is specified interactively in two ways: browsing the textual result list combined with selection of interest items and selection of results of interest in the graphical scene view. This latter case is shown in Figure 6.3 where an object of interest is selected in the scene view to the left and its respective search results are presented on the right hand site.

The scenic presentation shows a selected set of all search results. The object which is subject of the respective search is used as the geometric basis for the glyphs shown. Each of these glyphs is accompanied by a textual label containing the domain of the result's URL. All objects are arranged along a virtual spiral. Even though the spiral itself is not shown explicitly, this ordering helps to provide some maximum visibility of all objects and their labels simultaneously. Each shown result is scaled according to its relevance. A higher relevance causes an increased scale factor. The relevance value may be determined by any means outlined in the previous subsection. Currently, the number of occurrences of the search terms in the result's title and summary are used.

6.1.6 Mapping of Search Results onto Rendering Parameterisation

All defined rendering styles are available as set \mathcal{D} . Each element of this set is defined by its rendering parameterisation. Therefore, the parameter set for a specific presentation is denoted as $d_i \in \mathcal{D}$. The parameterisation of each d_i depends on the respective attribute set \mathbf{AM} of the represented illustration object IO . The parameterisation function δ provides temporal description of the dynamic rendering.

All objects IO_i that are currently affected by a dynamic presentation reflecting the evaluation of causally related search results are summarised in an information structure \mathbf{IS} . Therefore, such a structure includes all affected objects in each of the specific parameterisation cases as presented below.

The Illustration Client Application

A screenshot of the illustration client application is provided in Figure 6.4. As the main task of the application is the presentation of the illustration model, the rendering window takes up the largest part of the application space. This is accompanied with a set of annotation windows to the right. These are grouped by a set of tabbed widgets. The *Scene* tab which is displayed in the figure provides a tree view of the geometric model description at the top and all currently active search queries at the bottom. Other tabbed widgets provide information on annotations for the

model, any collected search results which are grouped into *clusters*, and illustration settings.

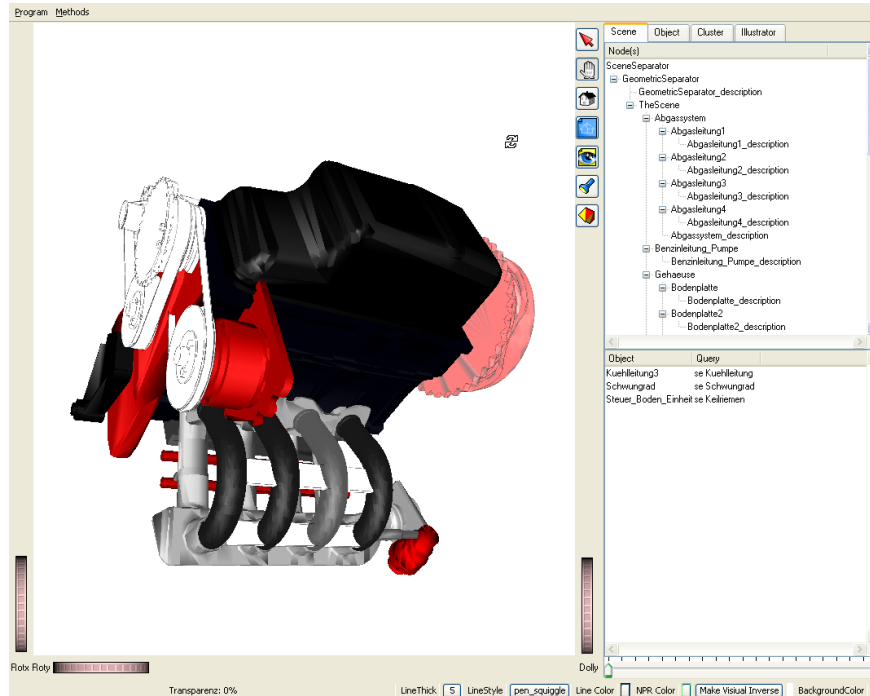


Figure 6.4: Screenshot of the illustration client application. The main part of the application's window is consumed by the rendering display of the geometric illustration model. To its right, a set of varying annotation spaces is provided.

Active Search Queries

An information structure \mathbf{IS}_a is composed of all objects that are currently subject of an active search query. This may explicitly affect multiple objects. For instance, a user might simultaneously search for the engine's cooling system parts, the fly-wheel, and parts of the propulsion system. These individual objects $IO_i \in \mathbf{IS}_a$ are emphasised by transparency blinking as illustrated in Figure 6.5. This provides some visual progress feedback. As soon as the search action is finished or a search timeout is reached, the blinking is stopped.

As any other collection of simultaneous use of dynamic presentation techniques, the respective application of an emphasis to an information object $IO_i \in \mathbf{IS}_a$ is subject to restriction by the *Temporal Server*. Figure 6.6 points out how this restriction is applied. Overall, a set of dynamic presentations $\mathcal{D}(\mathbf{IS}_a)$ is available for emphasising specific objects or parts thereof. In this specific case, the set includes blinking in addition to the techniques introduced in Chapter 3. This plain extension of \mathcal{D} shows that dynamic presentation techniques are handled in a flexible manner and

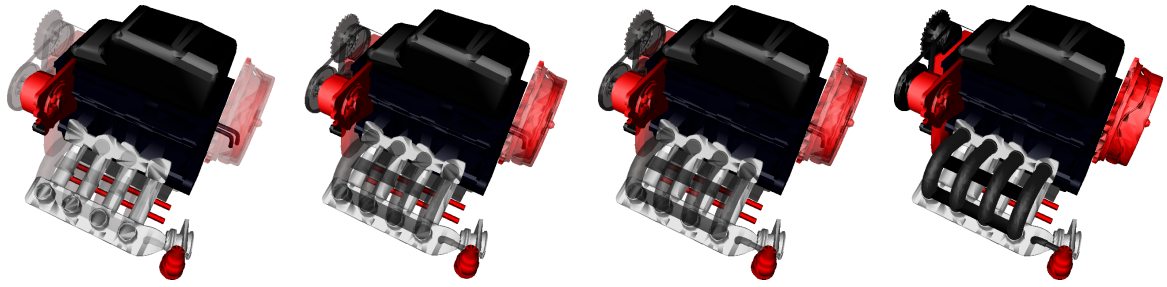


Figure 6.5: Snapshots of transparency blinking in order to illustrate objects affected by active search queries.

can easily be enriched by any dynamics not included so far. In case a specific dynamic presentation is to be used by the illustration system, a presentation query is sent to the *dynamics management unit (DMU)*. In the example shown in the figure, this is done by $Q_p(d_i(IO_{a_k}))$. Thereby, $d_i(IO_{a_k})$ represents blinking of object IO_{a_k} . The presentation query Q_p is evaluated by the *DMU* with respect to the temporal presentation parameterisation function δ . As discussed in Section 5.4.2 this results in the request to be allowed (pass), to be constrained, scheduled, or regretted. This general principle holds for the application of any other dynamic presentation technique used for an illustration enhanced by search engine queries as well.

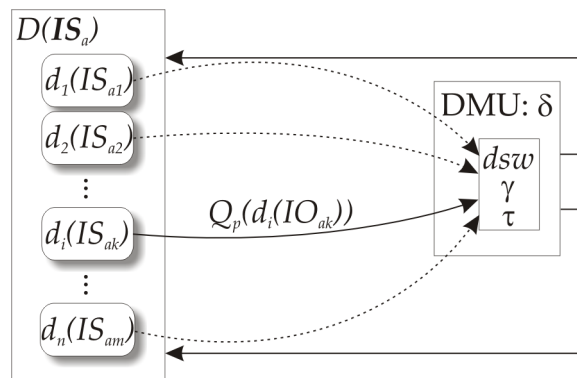


Figure 6.6: Using presentation queries for restricted use of dynamic presentation techniques by the illustration client. The currently active query is labelled as $Q_p(d_i(IO_{a_k}))$. Scheduled and finished queries are marked by dashed arrow lines.

Use of Transparency-based Rendering Style Dynamics

In order to illustrate by rendering parameterisation based on search query results, an information structure IS_s is constructed. This structure is based on the evaluation of a former search targeting a *single* object such as the cooling system. That is, all objects $IO_i \in IS_s$ are determined out of results retrieved from a search engine.

Figure 6.7 shows an illustration with an emphasised \mathbf{IS}_s containing the four tubes of the cooling system. Searching for their respective object labelling, these tubes are found to belong together as indicated by matching search results. Even though this relationship seems obvious to the user by itself, the method proves to be generally applicable.

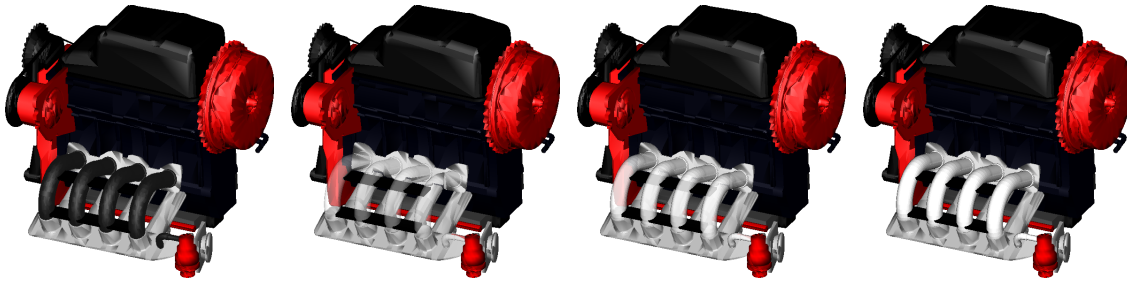


Figure 6.7: Illustrating a group of objects based on an evaluation of a respective search query. The illustration is supported by blending of two rendering styles using changes in transparency.

The transparency-based dynamic rendering style $d_t \in \mathcal{D}$ used for all $IO_i \in \mathbf{IS}_s$ is constructed by a cyclic transition from a realistically shaded rendering to an NPR shading. This transition is achieved by a double rendering of the objects. That is, the dynamic style is a composition of two other styles, or short $d_t \leftarrow d_r \times d_n$. By gradually adjusting the respective style's transparency values, the styles are transformed into each other. The drawback in rendering time resulting from the double rendering passes is somewhat compensated by fast α -blending as it is often implemented directly in hardware.

Use of Colour-based Rendering Style Dynamics

The illustration of Figure 6.7 reveals the see-through effect as it is discussed in Section 3.4.3. In order to overcome this effect here, the above information structure \mathbf{IS}_s may be illustrated by an alternative dynamic presentation technique. Instead of the transparency-based transition technique d_t , an alternative technique $d_c \leftarrow d_r \times d_n$ is constructed that employs the emissive colour of each $IO_i \in \mathbf{IS}_s$ for the change from one style to the other. This colour's RGB values are gradually increased from 0.0 to 1.0. A colour vector of (0.0, 0.0, 0.0) does not effect the material rendering at all. Setting all RGB values to 1.0 naturally results in the respective objects being rendered completely white. The stroke technique is blended in simultaneously. This way, only one rendering pass is needed.

Snapshots of the application of d_c to all $IO_i \in \mathbf{IS}_s$ are shown in Figure 6.8. This reveals a site-effect, though. The use of a completely saturated emissive colour on a stylised silhouette rendering as shown here loses some shape information. This is best expressed by the rightmost image of the figure.

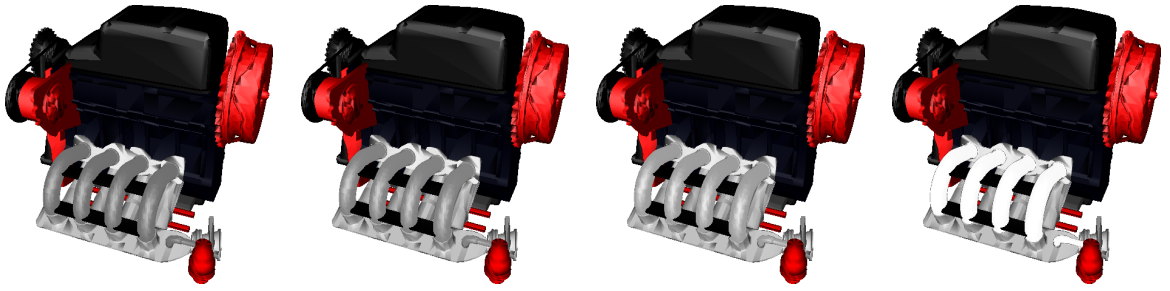


Figure 6.8: Illustration of the same group of objects as in Figure 6.7. Instead of blending between the different rendering styles by varying transparency, colour changes are used.

Combined Presentation of Multiple Search Results

In order to express different search results and their relevance, the mapping effects presented above can be combined. For this purpose, an information structure $\mathbf{IS}_m \leftarrow \mathbf{IS}_s \times \mathbf{IS}_s$ is constructed, whereby \mathbf{IS}_m labels an information structure containing objects affected by multiple searches and \mathbf{IS}_s respectively represent structures due to single search queries. This structure not only contains the information objects affected by a single search but all information objects that are subject to any of the set of multiple queries sent. Typically, these objects are grouped into sets of information structures. Each of these structures contains objects that belong together and are affected by the same search query.

The illustration of \mathbf{IS}_m results in the employment of more than one dynamic presentation technique in the scene. That is, instead of constructing a single dynamic style $d \in \mathcal{D}$, a set of techniques $\mathcal{D}_m \subset \mathcal{D}$ is used. Each $d_m \in \mathcal{D}_m$ can now be used for the illustration of all $IO_m \in \mathbf{IS}_i \subset \mathbf{IS}_m$.

This is illustrated in Figure 6.9. Three information structures are derived from the evaluation of all available search queries. One structure contains a cooling system tube, one structure consists of the flywheel, and one structure includes the propulsion system. Subject of dynamics-based illustration is therefore the set $\mathbf{IS}_m = \{\mathbf{IS}_{m_1}, \mathbf{IS}_{m_2}, \mathbf{IS}_{m_3}\}$. These are presented by means of $\mathcal{D}_m = \{d_{m_1}, d_{m_2}, d_{m_3}\}$. To be precise, all $IO_{m_i} \in \mathbf{IS}_{m_1}$ are rendered by the dynamic presentation technique d_{m_1} , all $IO_{m_j} \in \mathbf{IS}_{m_2}$ by d_{m_2} , and all $IO_{m_k} \in \mathbf{IS}_{m_3}$ by d_{m_3} . All of these dynamics are instantiated by varying hybrid rendering styles, parameterised differently for each respective information structure.

6.2 Information Fusion

The main context and requirements of information fusion are presented in Section 5.5. This section shows a set of exemplary application scenarios. First of all, an exemplary temporal scenario is presented in Subsection 6.2.2. This is followed by an illustration of climate data in Subsection 6.2.2, an exemplary scatter plot in

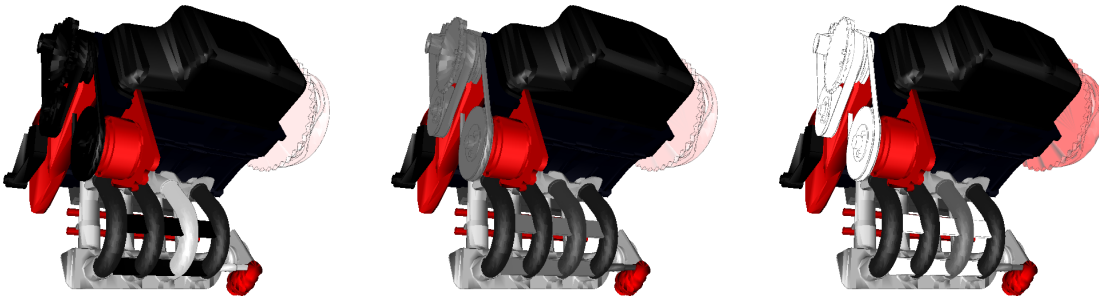


Figure 6.9: Illustration of multiple information sets resulting from a set of queries sent to a search engine. As different parts of the illustration model are also affected differently by search results, their presentation varies from each other.

6.2.3 and a discussion of online aggregation in Subsection 6.2.4.

6.2.1 Principle Fusion Result Presentation

Fig. 6.10 shows an exemplary set of presentation methods and their scheduling order throughout the duration of a session of an information fusion process execution. Different fusion operations lead to different result sets that are available during the illustration process. These result sets are referred to as:

$$RS = \{r_1, r_2, \dots, r_n\} \quad n \in \mathbb{N}.$$

Each result set $r_i \in RS$ (with $i = 1 \dots n$) is illustrated by a set of information objects forming respective information structures:

$$IS_{r_i} = \{IO_1, IO_2, \dots, IO_m\} \quad 1 \leq i \leq n, m \in \mathbb{N}.$$

A set of four dynamic presentation techniques is used for illustrating these information structures: $\mathcal{D} = \{d_1, d_2, d_3, d_4\}$. These are represented by bars in the figure. Detailed information about the information sets and presentation parameterisation is left out of the figure for brevity.

After a preparation period of the illustration session, the execution of the fusion process is started at time $t_0 = start$ and finished at time $t_{10} (act_3)$ because of a terminating user interaction event. Beginning at t_1 , the user is presented with visual representations of gained results of executed fusion operators. Some results might not be represented visually but only be used internally. An example for such case is result set r_3 .

As time advances and process execution progresses, the parameterisation of any used method d is subject to change. An exemplary change of illustrating result set r_1 happens at time t_2 because of internal process events. Thereby, user interaction is not involved. On the other hand, an interaction event at $t_7 = act_2$ leads to the extraction of result set r_4 and its immediate presentation. More process execution

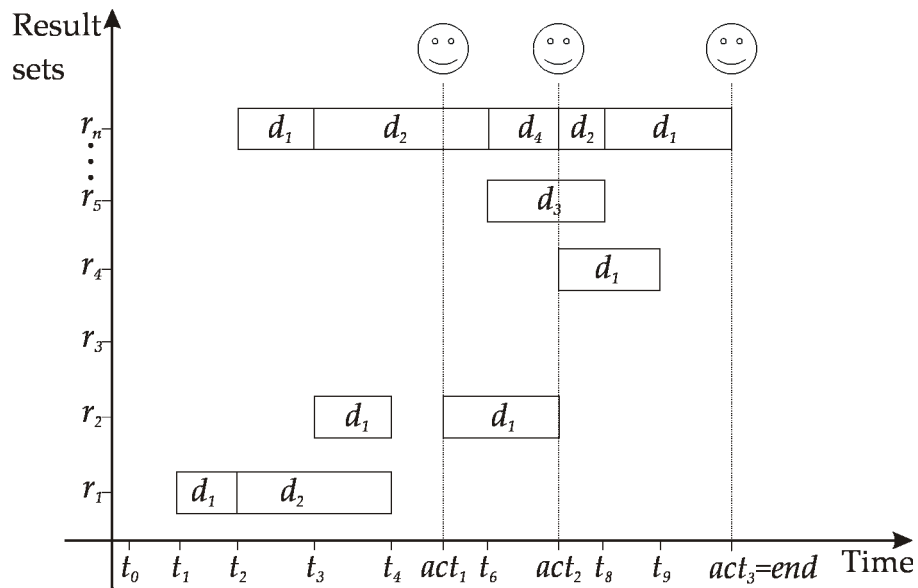


Figure 6.10: Scheduling of presentation methods during fusion process execution. Time points $t_5, t_7,$ and t_{10} are defined by user interaction events and are denoted as $act_1, act_2,$ and $act_3,$ respectively. Note that result set r_3 does not lead to any visual display and that only one result set (r_n) is valid until the final user event.

time is obviously needed in order to gain r_5 after user action happening at $t_5 = act_1$. The illustration of result set r_n changes multiple times throughout the whole session depending on both—user interaction and fusion process events.

All of $r_1, r_2, r_4,$ and r_5 represent intermediate result sets that are discarded and not used for final information gathering. Only r_2 is discarded explicitly by the user at time t_7 (which is equivalent to act_2). As discussed above, r_3 is only used internally and not represented visually.

Figure 6.11 shows the temporal graph for this exemplary presentation schedule. This graph follows the notation introduced in Chapter 4.

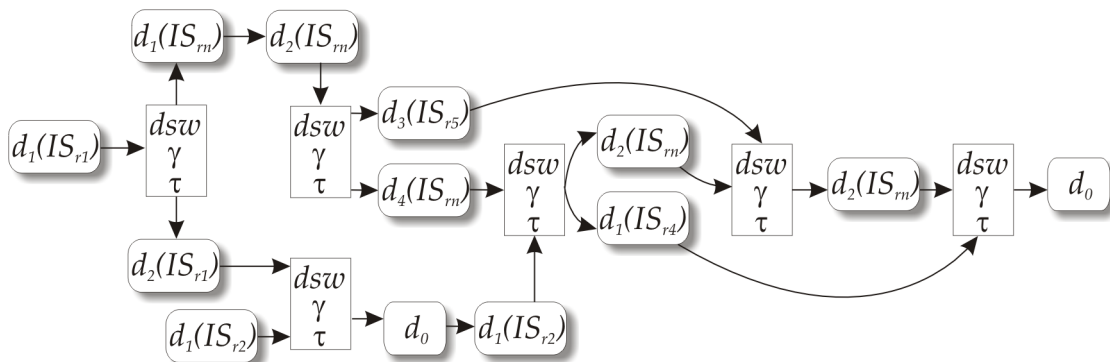


Figure 6.11: Temporal graph of the presentation schedule of Figure 6.10. This model uses the notation introduced in Chapter 4.

6.2.2 Climate Data

As introduced in Section 5.5, a goal of information fusion is in communicating information hidden in a collection of data retrieved from a variety of sources. This subsection shows an exemplary case of illustrating climate data as retrieved by simulation. For illustration, the motion-enhanced information mural is used. This illustration scenario is a result of an cooperation with Knud Pehrs and manifested in a diploma thesis [Pehrs, 2004].

Scenario

Possible interest in climate data spans a wide range of potential analysis areas. A common approach is the investigation of a curvilinear grid defined by longitude, latitude, and altitude. This allows to compare different area's data and is useful for weather forecasts. Another approach is to investigate data for a specific location only by sampling data at concrete location points. This preserves a reasonable amount of data and allows to examine data spanning a more extensive time frame. This latter case forms the basis of the work presented here. Precisely, daily weather data are used.

As available weather databases provide incomplete data in this regard, *ClimGen* is used to generate synthetic daily climate data.⁵ This tool allows for the generation of data containing total solar radiation, maximum and minimum temperature, rainfall, and wind-run. In order to generate any reliable data, the simulator needs to be parameterised accordingly. This is done by specifying a set of *location parameters*.

As an exemplary location, the place Magdeburg (Germany) is used. The location »Magdeburg« is described by a latitude of 52.13° , a longitude of 11.62° , and an elevation of 79.00 m . *ClimGen* allows various degrees of input data completeness. Possibly missing parameters are estimated by the program. The following monthly mean values for minimum temperature, maximum temperature, and precipitation are used as basis for data generation:

⁵ More information about *ClimGen*, its availability, and a comparison compendium of *ClimGen* and other climate data generators is available at <http://c100.bsyse.wsu.edu/climgen/> For information about why the program is used at all, see Jesse [2002].

month	$mean(T_{min})$	$mean(T_{max})$	$mean(prec.)$
January	-1.8°C	13.0°C	34.8 mm
February	-0.6°C	12.0°C	30.0 mm
March	0.6°C	13.0°C	35.0 mm
April	4.6°C	16.9°C	39.2 mm
May	6.1°C	37.1°C	47.6 mm
June	14.3°C	24.3°C	63.8 mm
July	13.0°C	29.0°C	58.4 mm
August	12.4°C	24.5°C	52.3 mm
September	10.5°C	20.5°C	38.1 mm
October	5.4°C	15.7°C	33.8 mm
November	-5.6°C	10.9°C	37.3 mm
December	-7.0°C	11.2°C	39.2 mm

This input data is partially retrieved from freely accessible web sources⁶ and partially estimated. Using this input, *ClimGen* generates usable data about precipitation and temperature. Due to the estimation of some parameters, possibly generated data about solar radiation and wind-speed are not of reliable nature and therefore not used.

Approaches for Climate Data Illustration

The field of climate data illustration is very diverse and only a brief overview of existing techniques is outlined here.

Max and Crawfis [1995] present cloud rendering methods for the purpose of analysing clouds and their behaviour over time. The presented methods include mapping of a 3D texture to cloudiness contour surfaces as well as volume rendering of clouds by splatting.

A set of web based applications is available that target the collection, handling, visualisation, and interpretation of climate data. Such a system is presented by Collins and Schweitzer [1997]. Their *GrADS* system provides a variety of techniques to present climate data. These techniques include but are not limited to graphs, scatterplots, contours, and streamlines. Another example of a web based climate data visualisation application is *FERRET* [Hankin et al., 1998]. Its set of provided techniques spans from various plots to different contour displays.

A different approach is shown by de Leeuw and van Liere [1999]. They use textures as expression variables for the purpose of presenting temporal data flow. As input data changes over time, texture does too. An exemplary application shows the temporal relation between atmospheric pollution and changes in wind fields.

⁶ Precisely, some climate data about Magdeburg is retrieved from <http://www.worldclimate.com/>, <http://www.stadtklima.de/>, and the home page of the German Environment Agency (\gg Umwelt-Bundesamt \ll).

Most *geographic information systems (GIS)* are capable of handling some amount of climate data. Their purpose is in management of geographic knowledge. A *GIS* usually works map based and provides support for raster analysis of complex areas. Exemplary *GIS* systems are *ArcInfo* and *ArcView*.

Mural Parameterisation

The initial attribute set **AM** is created automatically depending on data dimensions such as temperature and precipitation. Based on the discussions in Section 4.5 and 5.4, the mural presentation is described by a script document. Listing 6.1 shows an excerpt of such a document. This excerpt describes $d_1(IO_6)$ which starts after 11 s of the illustration and holds valid for a duration of 10 s. A snapshot of an illustration client including the mural presentation based on the document this excerpt belongs to is shown in Figure 6.12.

Listing 6.1: Document excerpt for generating the presentation shown in Figure 6.12

```

...
<sceneobjects>
  <object id="6" name="wideSpanningDay">
    <present method="1" start="PT11S" duration="PT10S" />
  </object>
</sceneobjects>
...

```

The threshold dimensions controllable by the user are the mean temperature threshold and the precipitation threshold. This allows to construct a structure \mathbf{IS}_m containing all IO_i with a represented temperature span larger than the threshold as well as a precipitation higher than specified. If the motion threshold function $mt(IO_i, T_{thres}, P_{thres})$ defines a function returning all objects meeting this criteria, the information structure results as $\mathbf{IS}_m = \{IO_1, IO_2, \dots, IO_n\}$ with $IO_i = mt(IO_i, T_{thres}, P_{thres})$ and $1 \leq i, j \leq n$. The objects $IO_i \in \mathbf{IS}_m$ are emphasised by applying an oscillating motion.

6.2.3 Exemplary Scatterplot

For the purpose of representing fusion results, a wide range of illustration techniques is available. An extensive overview of a toolkit providing supportive visualisation methods is given by Schroeder et al. [1998]. Figure 6.13 shows a snapshot of the information fusion workbench including a motion-enhanced scatterplot presentation. The snapshot also includes a list of meta-data representations to the left and the layout of a fusion process. Executing this process results in the shown scatterplot illustration. The concrete data forming the basis of the fusion process and its resulting illustration stems from a financial application scenario. Dunemann et al.

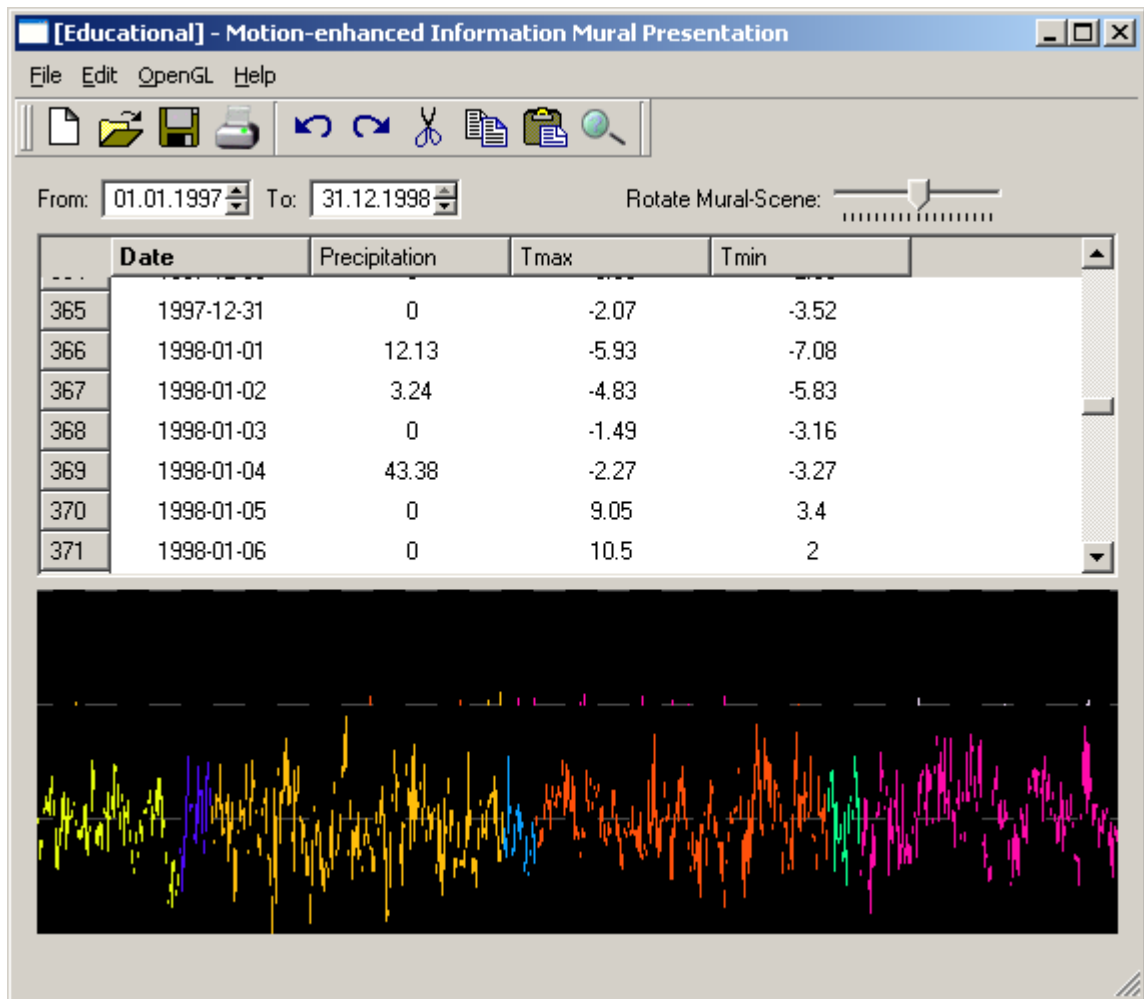


Figure 6.12: Snapshot of the illustration application using the *Motion-enhanced Information Mural*.

[2002a] describe this scenario in detail. The overall goal of this application case is in determining specific classes of bank customers for the purpose of intensifying the bank-customer relationship. An alternative application scenario for this workbench is discussed by Dunemann et al. [2002b] at the context of bioinformatics.

Especially for large data sets such a scatterplot includes the potential of hiding relevant information by overlapping object locations. A motion-enhanced scatterplot supports in finding such overloaded plot areas. For this purpose, an oscillating motion is applied to selected information objects. Assumed, the information set \mathbf{IM}_{SP} defines the overall scatterplot presentation. In order to apply oscillation to objects therein, an according information structure \mathbf{IS}_{OS} is composed. The individual objects contained in this structure are determined by a scatterplot motion function $\text{spmF}(\mathbf{IM}_{SP}, IO_i)$. For each information object $IO_i \in \mathbf{IM}_{SP}$, this function returns an overlap value of the object's position with regard to all remaining

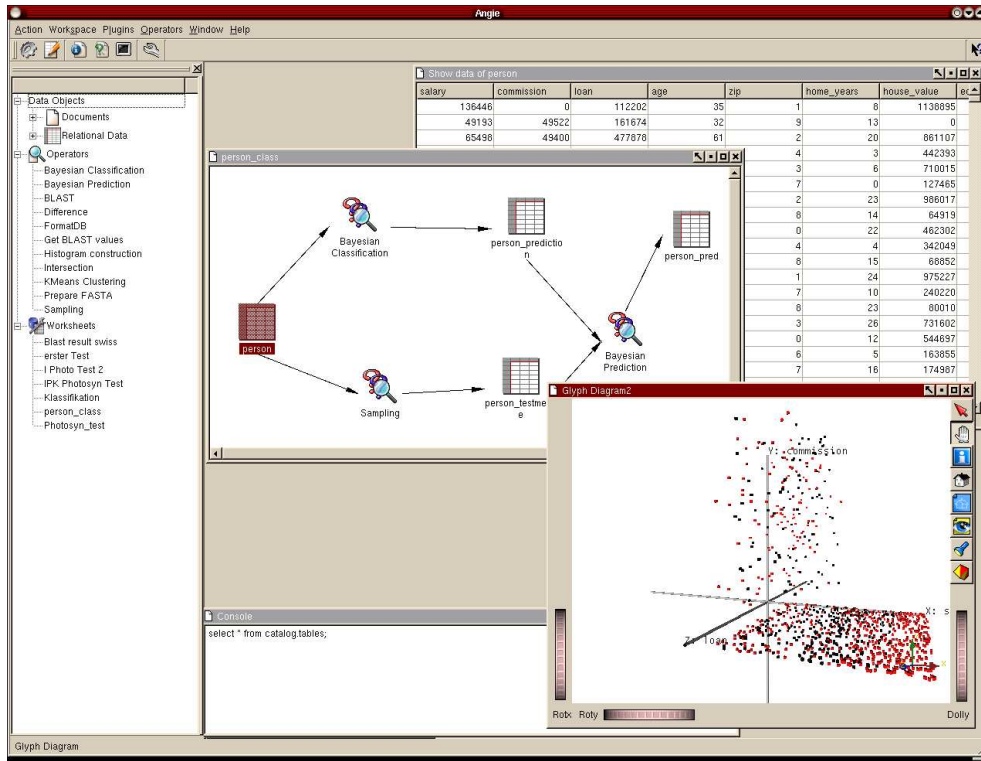


Figure 6.13: Snapshot of the *workbench for information fusion* including a fusion process based on financial data along with a motion-enhanced scatterplot illustration [Dunemann et al., 2002a].

objects ($\mathbf{IM}_{SP} \setminus IO_i$). For including an object in \mathbf{IS}_{OS} , an overlap threshold ϵ_{sp} is defined. Putting these components together results in the following composition of the information structure \mathbf{IS}_{OS} :

$$\mathbf{IS}_{OS} = \{IO_i \mid IO_i \in \mathbf{IM}_{SP} \wedge \text{spmF}(\mathbf{IM}_{SP}, IO_i) > \epsilon_{sp}\}.$$

All information objects of this structure are subject to oscillating motion. Using this dynamic presentation allows to visually identify classes in the underlying data. These classes may support the application user in the original task of bank-customer relationship optimisation.

6.2.4 Online Aggregation

Many data analysis scenarios are challenged by questions about how to provide access to data sources and how to represent the data. Aggregation is deemed as a common basis for providing access to data objects with similar characteristics. As aggregation is generally a time consuming batch operation, online aggregation was developed as an extension to provide access to intermediate results that are gradually refined.

For the purpose of representing data to the user, a variety of illustration techniques exists. Some of these techniques were developed by following a visualisation metaphor. One such metaphor is an abstract information landscape that is created by dynamically mapping currently available aggregation data onto geometric landscape objects. This enables an adaptive illustration that adjusts to online aggregation. The specific context on aggregation is discussed in detail by Jesse et al. [2003].

Landscape Mapping

The purpose of this section is not to present an enrichment of the catalogue of available visualisation techniques as presented by Schroeder et al. [1998], Schumann and Müller [2000], or Ware [2000]. Rather, a general description of methods for the representation of aggregated database content is targeted. For this purpose, an abstract landscape metaphor is used. That is, all data is mapped onto objects that form a landscape.

The individual landscape objects represent aggregation data. The attribute set **AM** of the data is mapped onto presentation dimensions such as position according to the landscape coordinates, the height of a landscape object, its material information, and rendering style information. An information object *IO* is positioned as a cuboid on a two-dimensional landscape grid which is described as $\mathcal{L} = \{p_0, \overline{d}_x, \overline{d}_y\}$. Thereby, the landscape consists of an origin p_0 and two vectors spanning the x and y coordinate axis, respectively. The height and material attributes of information objects represent attributes of the respective data sets as retrieved in the aggregation process.

Resulting snapshots of a constructed landscape are shown in Figure 6.14. From left to right, the images represent increased availability of data as retrieved from a database management system. That is, the information set **IM** is initially empty and individual information objects *IO* are added with increased data availability.

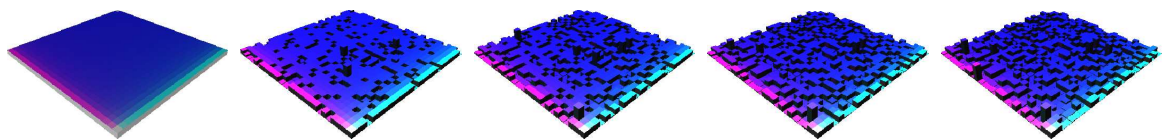


Figure 6.14: Landscape snapshots. The leftmost image represents a landscape without any mapped data. The remaining images show the representation of an increasing amount of mapped data.

Layered Styles

Figure 6.15 shows some results of applying hybrid rendering styles to the landscape illustration for the purpose of latency representation. Any possible delay in the retrieval of aggregated data is mapped onto the presentation of currently investigated

information objects $IO_i \in \mathbf{IS}$. As outlined in Section 3.4.1, multiple rendering styles are combined to a hybrid composition. Selected attributes $A_m, \dots, A_n \in \mathbf{AM}$ with $1 \leq (m, n) \leq k$ (cf. section on *landscape mapping above*) are used to parameterise the respective rendering style. Such attributes include—without being limited to—line style, line thickness, texturing, and colouring.

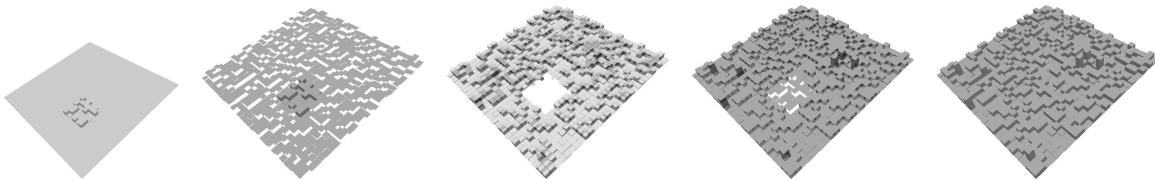


Figure 6.15: Rendering of the landscape with hybrid styles as presented in Section 3.4.1. The different styles are used to emphasise possible delays in data retrieval as caused by the database management system. The leftmost image shows an initial data set. The remaining images represent the addition of context data whereby the initial set still remains visible. The styles for this initial set and the remaining scene are gradually combined and finally form the representation shown in the rightmost image.

In order to link to the current status of aggregated data retrieval, the rendering parameters are chosen such that the rendering produces images that appear partially incomplete. This visual incompleteness is gradually refined. In case all data is finally retrieved completely, the result shown is to be constructed by a single homogeneous rendering style.

A discussion of interaction issues for feedback in the online aggregation process is provided by Jesse et al. [2003]. This includes an analysis of focus and context techniques for browsing the aggregation landscape, the presentation of an interaction layer model for influencing aggregation and resulting query refinement and aggregation control.

6.3 Model Presentations

This section addresses the use of dynamic presentation techniques for an illustrative presentation of geometric models. Subsection 6.3.1 presents use of hybrid rendering styles for illustration purposes in a technical model whereas use of the same techniques in a medical object is subject of Subsection 6.3.2. Finally, Subsection 6.3.3 presents use of oscillating motion to support exploration of a technical generator model.

6.3.1 Presentation of Technical Objects

The construction and maintenance life cycle for engineering models spans a broad area of activities. This includes the refinement of the model or parts thereof during

the *detailed design phase* [Fox and Salustri, 1994]. As an example, the model of an engine is used here. In order to examine the utilisation of individual engine elements, the design engineer is supported by visual feedback on the status of current items of interest.

Hybrid rendering styles may be used to highlight specific model parts by presenting them in another rendering style in contrast to the remaining parts of the model. Listing 6.2 shows an excerpt of a *Tigeop* document instance describing the varying use of an NPR style for the tube parts of a geometric engine model. Each tube is assigned a unique object id (4–7 in this case) and a name. The highlighting is done consecutively for all tubes according to the starting time stamps and durations as specified in the document.

Listing 6.2: Excerpt of a possible *Tigeop* documents describing the presentation shown in Figure 6.16

```

...
<sceneobjects >
  <object id="4" name="tube1">
    <present method="2" start="PT7.5S" duration="PT4.2S" />
  </object >
  <object id="5" name="tube2">
    <present method="2" start="PT12S" duration="PT3.7S" />
  </object >
  <object id="6" name="tube3">
    <present method="2" start="PT14S" duration="PT4.2S" />
  </object >
  <object id="7" name="tube4">
    <present method="2" start="PT18.5S" duration="PT1.3S" />
  </object >
</sceneobjects >
...

```

Figure 6.16 shows a series of snapshots from a dynamic presentation illustrating the utilisation of the engine model. The current item of interest is its cooling system. Four tubes form this system and are supported by a cooling aggregate located at the lower right side. The snapshots express the consecutive usage of all individual tubes with a smooth transition from one tube to the next. The work load of the small extra tube connected to the cooling aggregate is emphasised incrementally while going through the main tubes. The trained design engineer is now in charge of interpreting the scenario in order to decide on a potential lack of capacity for the bandwidth of this connection tube.

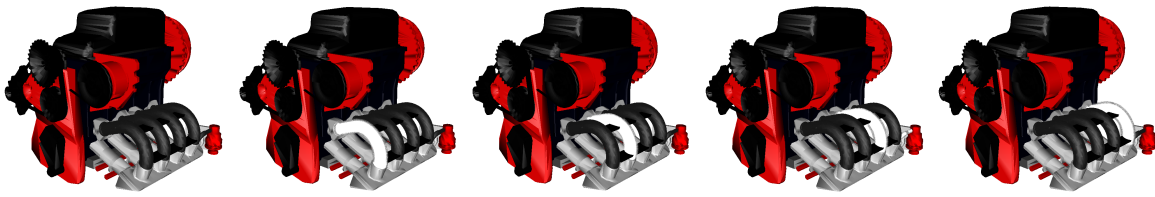


Figure 6.16: Snapshots of an engine model. The tubes of the engine’s cooling system are gradually emphasised by being rendered in a line shaped silhouette style. The style changes reflect the possible utilisation of the single tubes and their interplay with the cooling aggregate located ad the bottom right side.

6.3.2 Examination of Anatomical Objects

A typical goal in medical studies is in teaching students anatomical context of specific body parts. As stated by Ritter et al. [2000], a good way to approach this goal is in active exploration and examination of a geometric model of the object in question.

A series of snapshots for the example of a foot model presentation is shown in Figure 6.17. Therein, dynamic changes of hybrid rendering style are used to communicate the spatial relationship of phalanges—the bones of the toes. Beginning at a cuneiform bone, a silhouette line rendering is used in order to gradually emphasise the individual bones forming a toe and associated bones. Precisely, the order of bones as pointed out in the figure is: *Os cuneiformie I*, *Os metatarsale I*, *Phalanx proximalis I*, and *Phalanx distalis I*.

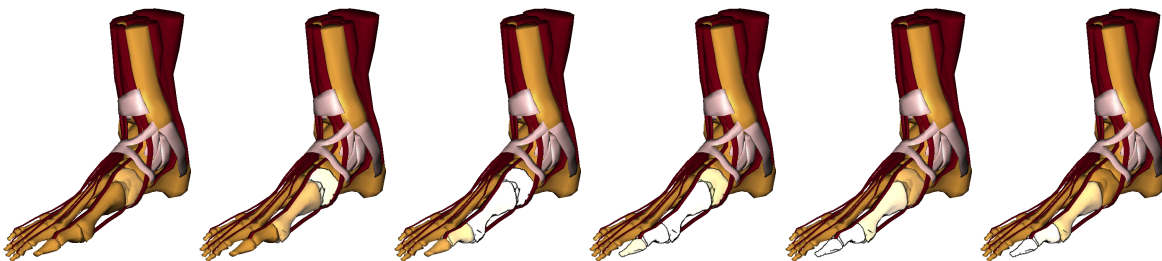


Figure 6.17: Snapshots of an animation pointing out specific parts of an anatomical example. *Os cuneiformie I*, *Os metatarsale I*, *Phalanx proximalis I*, and *Phalanx distalis I* are emphasised in order to illustrate their relationship.

This presentation can be run cyclical at varying speeds. Now, a sophisticated learning environment could possibly provide multiple parameterisations of the animation cycle to the students. By spanning the whole spectrum of the presented stimulus intervals and analysing the learning effect of the students, studies about respectively modified learning curves can be derived.

6.3.3 Motion-enhanced Model Exploration

The final example of this section addressing illustrative presentation of geometrical models makes use of motion instead of dynamic rendering styles. For this purpose, the *Motion Toolkit* has been integrated into an interactive system for user centred assembly of a geometric model that is presented by [Ritter et al., 2000]. This system allows to gain information about a model by means of active exploration.

The model of interest is the bearing device of a generator as shown in Figure 6.18. Specifically, Figure 6.18(a) shows the complete bearing in its fully assembled state. It consists of a shaft, a surrounding shield, and a set of supportive parts such as screws. All these parts are available as individual objects $IO_i \in \mathbf{IM}_B$ where \mathbf{IM}_B describes the whole bearing.

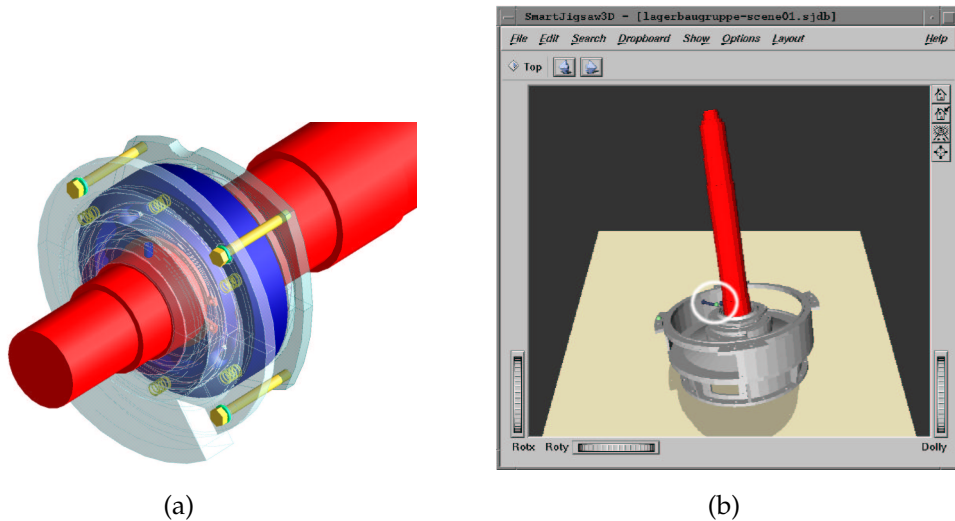


Figure 6.18: Constructing a bearing device out of its components. The completely assembled device is shown in 6.18(a) whereas 6.18(b) shows a construction snapshot in an early stage of assembly.

The task of active exploration of the model is accomplished by letting the user assemble the whole model out of its individual components. Figure 6.18(b) shows a snapshot of such an assembly task at an early construction state.

Oscillating motion is used to support the user in the exploration process. Specifically, motion is applied to provide visual feedback on objects which are currently of interest. The interest value of such objects is determined by an evaluation of user interaction. Two functions are defined in support of this:

1. $\text{interact}(\mathbf{IM}_B)$ returns the current object $IO_i \in \mathbf{IM}_B$ the user is interacting with.
2. $\text{class}(IO)$ determines the model class of an object. The notion of a class is part of the illustration model.

Using these functions, an information structure \mathbf{IS}_{OS} is maintained which consists of all objects of interest:

$$\mathbf{IS}_{OS} = \{IO_i \mid IO_i \in \mathbf{IM}_B \wedge \text{class}(IO_i) = \text{class}(\text{interact}(\mathbf{IM}_B))\}$$

The individual objects $IO_i \in \mathbf{IS}_{OS}$ are subject to oscillating motion. For the example shown in Figure 6.18(b), \mathbf{IS}_{OS} contains all screws of the bearing device model. All of the screws are positioned inside of the shield surrounding the shaft. For this reason, the screws are mainly hidden which prevents them from being available to the user immediately. The motion helps to reveal the screws. The motion can be faded out interactively in order to progress with the assembly process.

6.4 Summary

This chapter describes a set of exemplary applications which point out the use of dynamic presentations for illustration purposes. The individual dynamic techniques introduced in Chapter 3 are used along with the temporal parameterisation model of Chapter 4 by means of the respective components of Chapter 5.

The set of applications spans a broad range of areas. At first, an illustration scenario is presented where dynamic presentations are used to reflect dynamic behaviour of the illustration model. These latter dynamics are caused by using search engines to retrieve information about the illustration model. This information is of dynamic nature as the concrete contents of the results retrieved from a search engine are not known beforehand. Dynamic rendering styles support the resulting illustration and are used to reflect the results retrieved from the search engine along with its correlation to the illustration model.

This application example is followed by a set of illustrations rooted in the field of information fusion. First of all, the principle presentation of fusion results is shown. Use of the motion-enhanced information mural is presented at the example of climate data. This is followed by employing oscillating motion to enrich a scatter plot presentation of a financial data set. Finally, the creation of an illustration supporting online aggregation by means of a landscape metaphor rounds up the set of information fusion presentations. The chapter is completed with a set of illustrative presentations of geometric models. These models are of technical as well as of medical nature. For the purpose of illustrating the models, dynamic rendering styles are used as well as oscillating motion.

Overall, the applications shown point out that dynamic presentations can be used to enrich a variety of illustrations. The selection is intended to serve as a starting point for investigations of further illustration scenarios which might profit from dynamic presentations. The set of applications shown is by no means complete but can only serve to reveal the diversity of potential illustrations based on dynamics.

7 Concluding Remarks

Along with the steady increase of information density goes an increase of information need in daily life as well as in the scientific world. Illustrations play a vital role in satisfying this need. They form as an element in the set of presentation systems which communicate information by means of visual presentations. In support of this, a variety of presentation techniques is available. These techniques are characterised by a broad spectrum of property dimensions. Some techniques are specifically employed to ensure visibility of an object which is part of an illustration model. Examples include according use of transparency as well as appropriate translations of the object in question. Other illustrative presentation techniques concentrate on the task of emphasising an illustration object of interest. Examples are changes of colour of the respective object or use of a supportive helper object such as a bounding box, a crosshair, or arrows.

While these existing illustration techniques provide a broad means of presenting a variety of information, there is a need for visually reflecting more and more complex illustration scenarios. In such scenarios, stored information may be available in a constantly varying state. Furthermore, complex illustration goals of a user require advanced illustration techniques.

To overcome these deficits, this thesis introduces the concept of *dynamic presentations* for illustration purposes. Thereby, the notion of *dynamics* is defined as presentations that change over time. Dynamics are constructed in multiple ways and include motion as well as variations and combinations of other already existing techniques. Such dynamics support illustrations in multiple ways. As an intrinsic characteristic, they allow to represent temporal behaviour of an illustration model. In addition, they help to support visibility of objects as well as specific emphasis. The concrete value of a dynamic presentation technique depends on its underlying presentation variables.

The following section summarises the main contribution of this thesis: The models and techniques for dynamic presentations for illustration purposes. Section 7.2 outlines further research directions stemming from the work presented in this thesis.

7.1 Summary of Contributions

For the purpose of constructing the cognitive basis for dynamic presentations, an analysis of published studies on the perception of varying presentations is provided. The results of this analysis are manifested in the definition of a *dynamics stimulus window* and a *hierarchy layer model* of dynamics. Both of these frame all further investigations and provide limitations to which all developed models and techniques need to comply. For this reason, this in-depth analysis of user-centred fundamentals serves as a substitute for carrying out a set of user studies on the general usefulness of dynamics. This does not attempt to role out such studies for the purpose of evaluating presentation techniques. Instead, it values the pre-application character of the techniques targeted by this work. While a set of exemplary applications points out prospective use of dynamic presentations for illustration purposes, dynamics and supportive models as presented in this thesis are of generic nature. Whether a dynamic presentation technique is more valuable than any other depends on the precise application at hand.

The diversity of dynamics is outlined by a discussion of classes of multiple such techniques. These classes define the spectrum of exemplary dynamic presentation techniques: motion, dynamic changes of rendering styles, and zoom-based distortion histories. This way, dynamics are composed as a collection of techniques ranging from classic motion-based animations to variable non-photorealism and extensions of presentation techniques which are already of dynamic nature.

The presented set of dynamics by motion is divided into three areas: oscillations, structural changes, and a motion-enhanced information mural. Oscillations describe object-based motion, structural changes affect object surfaces, and the information mural is a classic illustration technique which is extended to make use of both other motion dynamics.

Dynamics by *changes of rendering styles* are based on non-photorealism as presentation technique. *Hybrid renderings* are created which combine this non-photorealism with classic realistic renditions. Thus, such renderings contribute to the goal of developing new presentation techniques. Dynamic presentations are derived by continuously blending between different rendering styles. Two different approaches are shown for such blending: adjustments of transparencies and changes in an object's emissive colour. The discussion of rendering style dynamics is accompanied with a description of the see-through effect. This effect possibly occurs when different rendering styles are used simultaneously. This is rather undesirable, since it results in parts of the scene to shine through even though they are originally hidden by other scene elements. Strategies are presented for avoiding this effect while still maintaining the derived rendering style transitions.

Presentation of distortion histories form the third class of exemplary dynamics. At the specific case of a fisheye-zoom, two ways of constructing a zoom history are introduced: trace lines and a chewing gum. The trace lines use the concept of speed

lines to indicate zoom progression. The chewing gum provides a means of inserting constantly updated intermediate objects in a scene which is subject to distortion. These objects reflect the distortion history by being stretched continuously until they rip up.

All discussed presentation techniques are analysed and compared to each other regarding their influence on scene coherence. Thereby, scene coherence is defined as the influence of a dynamic presentation—which is applied to a specific object—on the remaining scene’s presentation. Using the results of this comparison allows to support design decisions for an illustration scenario. To embed use of dynamics for illustrations in a broader set of presentation techniques, supportive overviews and classifications of such techniques are discussed.

For the purpose of designing illustrations that include dynamic presentation techniques, the notion of an *illustration target function* is introduced. This function includes all elements of the illustration model along with supportive context information as well as relations and operations defined for the model. The definition of a target function for a specific illustration task at hand helps to identify and materialise the overall illustration goal as well as necessary presentation techniques to meet this goal.

A further contribution of this thesis is the construction of a *temporal model* for parameterising and constraining dynamic presentations. This model’s design is based on a set of requirements that are identified to support the control of dynamics with respect to the cognitive limitations presented earlier. An evaluation and classification of existing temporal modelling approaches provides further design inspirations, as modelling ideas of these approaches are reused. The developed model is accompanied by the specification of a temporal control function. This function serves as basis for materialising any temporal parameterisation of dynamics. Presented examples for concrete use of this function include the design of a presentation script as well as an implementation of a dynamics management unit.

The individual contributions are instantiated by a set of implemented system components which form a temporally constrained illustration framework based on dynamic presentation techniques.

As a summary, Figure 7.1 provides an abstract view on the relationship of the individual parts of this thesis. Concrete real-world scenarios build the basis of the illustration context within the overall goal to achieve and manage information. Such scenarios are addressed by illustration models as manifested by an illustration target function. This function expresses an illustration goal which is targeted by compositions of presentation techniques. These techniques include dynamics without being limited to them. At the core of an illustration stands the user who defines limitations of any presentation based on cognitive restrictions. These are addressed by temporal constraints which parameterise a presentation. Overall, all components need to interplay such that the user is supported in the search for desired information in the real world.

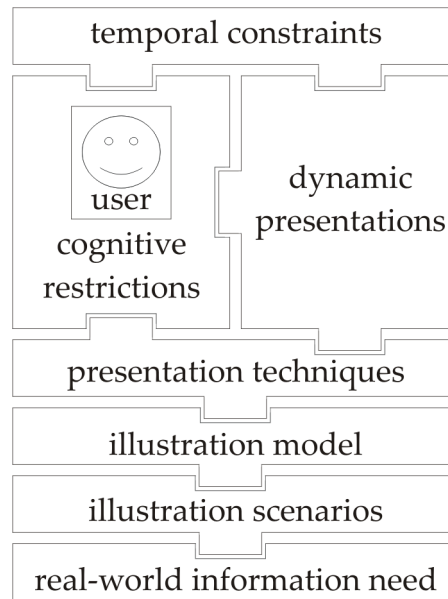


Figure 7.1: Interplay of the individual contributions of this thesis and their overall context.

7.2 Further Research Directions

Throughout this thesis, the notion of dynamic presentation techniques was discussed along with supportive models. Thereby, the first crucial step towards enriching illustrations by dynamics is made. However, much remains to be explored before the potential for dynamics-enhanced illustrations can be realised to its full extent. In the following, interesting research directions resulting from this thesis are discussed.

7.2.1 Extend Use of Distortion Histories

All dynamic presentation techniques discussed throughout this thesis are characterised by some prototypical status. They are designed to illustrate the notion of dynamics as presentation which varies over time. Thus, all of these techniques provide potential for further refinement and extensions.

As an example, distortion histories can and should be extended to further distortions besides the fisheye-zoom. The trace lines used to present coherent zooming stem from the concept of speed lines rooted in motion representation in still images. As much as use of these lines migrated from still images to fisheye-zoom, it can continue to move on to other presentations as well. As the lines reflect spatial history information, those presentations should be motion-based or provide alternative means of spatial variation. The second dynamic presentation technique introduced for display of distortion histories is the chewing gum. This gum rep-

resents an intermediate object, created exclusively for representation of distortion progression. Similar to trace lines, this follows a spatial-oriented dynamic presentation approach. Thus, the chewing gum may possibly be of use in any presentation which results in spatial variation. As in the case of trace lines, this includes motion-based techniques as well as further distortions.

7.2.2 Extend General Field of Dynamics

The set of dynamic presentation techniques introduced in this thesis provides an idea of the broad range of dynamics. This set spans a variety of underlying presentation techniques: motion, different and hybrid rendering styles, and distortions.

However, all of the techniques discussed so far concentrate on visual presentations only. The derived notion of dynamics does not yet value any integration of multi-media components in illustration systems. Specifically, the extension of dynamics to include sound and variations thereof promises to add considerably to the set of dynamic presentation techniques.

Supportive studies on perceptual issues of sound are published by the International Community for Auditory Display (ICAD¹). In the context of this community and related conferences, approaches to sonification as data-controlled sound have been presented. Integration of these approaches with dynamic graphical display techniques poses as challenge worth to be pursued.

7.2.3 Extend Field of Applications

The applications discussed throughout this thesis provide an impression of the variety of principle use of dynamics for illustration purposes. Further improvements can be achieved for each application shown as much as new illustration scenarios can be created with dynamics as an illustrative expression dimension.

A selected continuation of one of the presented applications is in support of external documents in the illustration scenario of querying search engines for illustration information. As of now, the respective illustration client concentrates on handling queries defined interactively during runtime of an illustration-supported model exploration. External documents may provide textual descriptions and annotations for the illustration model. This text and parts thereof may now be used to refine queries sent to the search engine. In case the documents are composed by experts with valuable background knowledge about the illustration model, this promises to enhance search result quality considerably. Support for such external model documents has already been implemented as an early draft. First results

¹ Information about ICAD—the International Community for Auditory Display—is available at <http://www.icad.org/>.

sound promising, even though test documents were created without a specific domain knowledge at hand. Furthermore, first tests point out that appropriate interaction techniques need to be developed to access not only whole documents for inclusion in the search process but to extract most relevant subparts of a document. Such document access may even be combined with linguistic text analysis tools in order to optimise the model-text relationship before the user is involved in document handling.

Communication of conflict information serves as basis for a potential application of dynamics for data-driven illustrations. Such an illustration serves to represent information gained throughout a retrieval application in the context of information fusion. Thereby, a possible user information goal is to determine the quality of intermediate and final results. During a fusion process, different stages of data integration are employed. At each level of integration, resulting data may be subject to conflicts and mismatches of the integration process. Such dirty data may now be mapped onto dynamic presentation. This helps to reveal possible errors throughout the retrieval process as early as possible. Depending on the integration level, appropriate interaction can result to solve any conflicts. By using different dynamics for different layers, the user is directly informed about the respectively responsible system components.

An exemplary new application area for geometry-based illustrations enhanced by dynamic presentation techniques is in support of industrial design. Here, illustrations allow to reflect different stages throughout a design process. An example is the design of a car engine. Different parts of this engine are designed independently from each other. At various stages throughout the overall design process, parts of the engine model are combined. Supportive illustrations point out relationships between these parts as well as conflicts or weaknesses in the engine composition. Dynamics may be used here to point out such conflicts and weaknesses. According parameterisation of concrete dynamic presentation techniques employed for this purpose helps to respect the grade of conflict. This supports the designer in the process of identifying intermediate model weaknesses as early as possible during the design phase.

7.2.4 Application-centred User Studies

As discussed in the previous section, a set of user studies may be carried out to evaluate the potential of dynamics for specific application areas. Such studies should specifically target the application of dynamics for a concrete task at hand. This allows to respect the context of the application domain. By using this context information at the designing stage of a user study, this study can address application specific needs that possibly influence the value of dynamics in the concrete case at hand.

Bibliography

- Bart Adams and Philip Dutré. Interactive Boolean Operations on Surfel-Bounded Solids. In John C. Hart, editor, *Proceedings of ACM SIGGRAPH 2003*, volume 22 of *Annual Conference Series*, pages 651–656. ACM, 2003.
- Marc Alexa. Merging Polyhedral Shapes with Scattered Features. *The Visual Computer*, 16(1):26–37, 2000. doi: citeseer.nj.nec.com/alexa00merging.html.
- Vicki H. Allan, Reese B. Jones, Randall M. Lee, and Stephen J. Allan. Software Pipelining. *ACM Computing Surveys*, 27(3):367–432, 1995.
- James F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, 1983.
- James F. Allen. Time and Time again: The Many Ways to Represent Time. *International Journal of Intelligent Systems*, 6(4):341–355, 1991.
- Mihael Ankerst, Stefan Berchtold, and Daniel Keim. Similarity Clustering for an Enhanced Visualization of Multidimensional Data. In G. Wills and J. Dill, editors, *Proceedings of Information Visualization*, pages 52–62. IEEE, 1998.
- Nasreddine Aoumeur. *Specifying and Validating Consistent and Dynamically Evolving Concurrent Information Systems: An Object Petri-net Based Approach*. PhD thesis, University of Magdeburg, Germany, 2002.
- Nasreddine Aoumeur and Gunter Saake. An Appropriate Semantics for Distributed Active Object-Oriented Databases on the Basis of the Co-nets Approach. In H. Fouchel and Y. L. Roger, editors, *International Conference on Software Engineering Applied to Networking and Parallel / Distributed Computing (SNPD '00)*, pages 541–548. International Association for Computer and Information Science (ACIS), 2000.
- Alessandro Artale, Enrico Franconi, Frank Wolter, Milenko Mosurovic, and Michael Zakharyashev. The DLR_{US} Temporal Description Logic. In *Workshop Notes of the Int. Workshop on Description Logics (DL-01)*, Stanford University, CA, August 2001.

- K. W. Arthur, K. S. Booth, and Colin Ware. Evaluating 3d task performance for fish tank virtual worlds. *ACM Transactions on Office Information Systems*, 11(3):239–265, 1993.
- Ahmet Feyzi Ates, Murat Bilgic, Senro Saito, and Behcet Sarikaya. Using Timed CSP for Specification, Verification, and Simulation of Multimedia Synchronization. *IEEE Journal on Selected Areas in Communication*, 14(1):126–137, 1996.
- Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern information retrieval*, volume 5. ACM Press, 2002.
- Linda R. Bartram. Can Motion Increase User Interface Bandwidth in Complex Systems? In *Proceedings of 1997 IEEE Conference on Systems, Man and Cybernetics*, pages 1686–1692, 1997a.
- Linda R. Bartram. Perceptual and Interpretative Properties of Motion for Information Visualization. Technical Report CMPT-TR-1997-15, School of Computing Science, Simon Fraser University, 1997b.
- Linda R. Bartram. Enhancing Visualizations With Motion. In *Hot Topics: Information Visualization 1998*, pages 13–16, North Carolina, USA, 1998.
- Linda R. Bartram. *Enhancing Information Visualization with Motion*. PhD thesis, Simon Fraser University, 2001.
- Linda R. Bartram, Albert Ho, John Dill, and Frank Henigman. The Continuous Zoom: A Constrained Fisheye Technique for Viewing and Navigating Large Information Spaces. In *UIST '95: Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 207–216. ACM Press, 1995.
- Linda R. Bartram, Colin Ware, and Tom Calvert. Moving Icons: Detection And Distraction. In *Proceedings of Interact 2001*, Tokyo, 2001.
- Bruce G. Baumgart. A Polyhedral Representation for Computer Vision. In *Proceedings for the National Computer Conference*, volume 44, pages 589–596. AFIPS, 1975. doi: <http://citeseer.nj.nec.com/context/232104/0>.
- Marcus Beale, Marty Einstein, Scott McCrickard, Chris North, and Purvi Saraiya. Visualizing Communication Timelines Containing Sparsely Distributed Clusters. In *Proceedings of IEEE InfoVis 2001 Symposium*. IEEE, 2001.
- Jeff E. Beall, Adam M. Doppelt, and John F. Hughes. Developing an Interactive Illustration: Using Java and the Web to Make It Worthwhile. In *Proceedings of 3D and Multimedia on the Internet, WWW and Networks*, Computer Graphics, 1996. doi: citeseer.nj.nec.com/47998.html.

- Matthew I. Bellgard, Hong Lian Hiew, Adam Hunter, and Michael Wiebrands. OR-BIT: an integrated environment for user-customized bioinformatics tools. *Bioinformatics*, 15(10):847–851, 1999.
- Elisa Bertino and Elena Ferrari. Temporal Synchronization Models for Multimedia Data. *IEEE Transactions on Knowledge and Data Engineering*, 10(4):612–631, 1998.
- Claudio Bettini, Sushil Jajodia, and Sean X. Wang. *Time Granularities*. Springer-Verlag, Berlin Heidelberg New York, 2000.
- Olivier Biberstein, Didier Buchs, and N. Guelfi. CO-OPN/2: A Concurrent Object-Oriented Formalism. In *Proceedings of Second IFIP Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS)*, pages 57–72. Chapman and Hall, 1997. doi: citeseer.nj.nec.com/biberstein97coopn.html.
- Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony D. DeRose. Toolglass and Magic Lenses: The See-Through Interface. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics*, pages 73–80, 1993.
- Volker Blanz and Thomas Vetter. A Morphable Model for the Synthesis of 3D Faces. In *Proceedings of ACM SIGGRAPH 1999, Annual Conference Series*, pages 187–194. Addison Wesley Longman, 1999. doi: citeseer.nj.nec.com/blanz99morphable.html.
- Grady Booch. *Object-Oriented Analysis and Design with Applications*, volume 2. Addison-Wesley, 1993.
- M. Bordegoni, G. Faconti, M. T. Maybury, T. Rist, S. Ruggieri, P. Trahanias, and M. Wilson. A Standard Reference Model for Intelligent Multimedia Presentation Systems. *Computer Standards and Interfaces*, 18(6-7):477–496, December 1997.
- Lubomir Bourdev. Rendering Nonphotorealistic Strokes with Temporal and Arc-Length Coherence. Master's thesis, Brown University, 1998.
- M. Cecelia Buchanan and Polle T. Zellweger. Specifying Temporal Behavior in Hypermedia Documents. In *Proceedings of ACM Conference on Hypertext*, pages 262–271, 1992.
- M. Cecelia Buchanan and Polle T. Zellweger. Automatically Generating Consistent Schedules for Multimedia Documents. *Multimedia Systems*, 1(2):55–67, September 1993.
- Tara Calishain and Rael Dornfest. *Google Hacks*. O'Reilly, 2003.

- R.H. Campbell and A.N. Habermann. The Specification of Process Synchronization by Path Expressions. In G. Goos and J. Hartmanis, editors, *Operating Systems*, number 16 in Lecture Notes in Computer Science, pages 89–102. Springer-Verlag, 1974.
- Hans-Otto Carmesin and Stefan Arndt. Neuronal Self-Organization of Motion Percepts. ZKW-Bericht 6/95, Zentrum für Kognitionswissenschaften, Universität Bremen, 1995.
- M. Sheelagh T. Carpendale, David J. Cowperthwaite, and F. David Fracchia. 3-Dimensional Pliable Surfaces: For the Effective Presentation of Visual Information. In *UIST 1995: Proceedings of the 8th ACM symposium on User interface and software technology*, pages 217–226, 1995.
- M. Sheelagh T. Carpendale, David J. Cowperthwaite, and F. David Fracchia. Making distortions comprehensible. In *Proceedings of the 1997 IEEE Symposium on Visual Languages (VL '97)*, pages 36–45. IEEE Computer Society Press, 1997.
- M. Sheelagh T. Carpendale, David J. Cowperthwaite, M. Tigges, A. Fall, and F. David Fracchia. The Tardis: A Visual Exploration Environment for Landscape Dynamics. In *Proceedings of the Conference on Visual Data Exploration and Analysis*, 1999.
- Jonathan Chey, Stephen Grossberg, and Ennio Mingolla. Neural dynamics of motion processing and speed discrimination. *Vision Research*, (38):2769–2786, 1997.
- Wallace Chigona and Thomas Strothotte. Contextualized Text Explanation for Visualizations. In *Proceedings of the 2nd International Symposium on Smart Graphics*, pages 27–34, New York, 2002a. ACM Press.
- Wallace Chigona and Thomas Strothotte. Distortion for readability of contextualized text explanations for visualizations. In *Proceedings of 6th International Conference Information Visualization*, pages 289–294. IEEE Computer Society, 2002b.
- Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. Web Services Description Language (WSDL) 1.1. <http://www.w3.org/TR/wsdl>, March 2001.
- Robin Cohen, Peter Vanderheyden, and Michael Fleming. Meeting Users' Information Needs Through Interaction. In *Proceedings of International Conference on Intelligent User Interfaces (IUI)*, 2001.
- Daniel Cohen-Or, David Levin, and Amira Solomovici. Three-Dimensional Distance Field Metamorphosis. *ACM Transactions on Graphics*, 17(2):116–141, 1998.

- Julia Collins and Roland Schweitzer. Find it, Plot it, Grab it: Distributing Climate Data Via the Web. In *Sixth International World Wide Web Conference*, Santa Clara, California, USA, 1997. Online Proceedings, available at <http://www.cdc.noaa.gov/~jac/www.97/>.
- Peter E. Comalli, Jr., Heinz Werner, and Seymour Wapner. Studies in physiognomic perception III: Effect of directional dynamics and meaning-induced sets on autokinetic motions. *The Journal of Psychology*, 43:289–299, 1957.
- Stefan Conrad and Gunter Saake. Specifying Evolving Temporal Behaviour. In J. Fidadeiro and P.-Y. Schobbens, editors, *Proceedings of the 2nd Workshop of the ModelAge Project*, pages 51–65. Departamento de Informatica, Universidade de Lisboa, 1996.
- Stefan Conrad, Can Türker, and Gunter Saake. Towards Agent-Oriented Specification of Information Systems. In *Proc. of the 1st Int. Conf. on Autonomous Agents*, pages 502–503. ACM Press, 1997.
- Jean-Pierre Courtiat and Roberto C. de Oliveira. Proving Temporal Consistency in a New Multimedia Synchronization Model. In *ACM Multimedia*, pages 141–152. ACM Press, 1996.
- Steve B. Cousins and Michael G. Kahn. The Visual Display of Temporal Information. *Artificial Intelligence in Medicine*, (3):341–357, 1991.
- J. Davies, D.M. Jackson, J.N. Reed, G.M. Reed, A.W. Roscoe, and S.A. Schneider. Timed CSP: Theory and Practice. In Jacobus Willem de Bakker, C. Huizing, W.P. De Roever, and G. Rosenberg, editors, *Real-Time: Theory and Practice*, number 600 in Lecture Notes in Computer Science, pages 640–675. Springer-Verlag, 1991.
- Iryna Davydova. Nicht-Fotorealistische Verzerrungen zu Illustrationszwecken. Diploma thesis, University of Magdeburg, 2003.
- Willem Cornelis de Leeuw and Robert van Liere. Spotting Structure in Complex Time Dependent Flow. In Hans Hagen, Gregory M. Nielson, and Frits H. Post, editors, *Scientific visualization*, pages 47–53. IEEE, 1999.
- Willem Cornelis de Leeuw and Jarke J. van Wijk. Enhanced spot noise for vector field visualization. In G.M. Nielson and D. Silver, editors, *Proceedings Visualization '95*, pages 233–239. IEEE Computer Society Press, 1995.
- Jeffrey Dean and Monika R. Henzinger. Finding Related Pages in the World Wide Web. *Computer Networks*, 31(11-16):1467–1479, 1999.
- Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow. In *Proceedings of ACM SIGGRAPH*, Annual Conference Series, pages 317–324. ACM, ACM, 1999.

- Alan Dix and Gegory Abowd. Delays and Temporal Incoherence Due to Mediated Status-Status Mappings. *SIGCHI Bulletin*, 28(2):47–49, April 1996.
- A. Dogac, C. Dengi, E. Kilic, G. Ozhan, F. Ozcan, S. Nural, C. Evrendilek, U. Halici, B. Arpinar, P. Koksall, and S. Mancuhan. A Multidatabase System Implementation on CORBA. In *RIDE'96, Proc. of the 6th International Workshop on Research Issues in Data Engineering: Interoperability in Nontraditional Database Systems*, pages 2–11, Los Alamitos, CA, 1996. IEEE Computer Society Press.
- Oliver Dunemann, Ingolf Geist, Roland Jesse, Gunter Saake, and Kai-Uwe Sattler. InFuse – Eine datenbankbasierte Plattform für die Informationsfusion. In Andreas Heuer, Frank Leymann, and Denny Priebe, editors, *Datenbanksysteme in Büro, Technik und Wissenschaft, BTW 2001*, pages 9–25, Oldenburg, 2001. Springer.
- Oliver Dunemann, Ingolf Geist, Roland Jesse, Gunter Saake, and Kai-Uwe Sattler. Informationsfusion auf heterogenen Datenbeständen. *Informatik, Forschung und Entwicklung*, 17(3):112–122, September 2002a.
- Oliver Dunemann, Ingolf Geist, Roland Jesse, Kai-Uwe Sattler, and Andreas Stephanik. A Database-Supported Workbench for Information Fusion: Infuse. In Christian S. Jensen, Keith G. Jeffery, Jaroslav Pokorny, Simonas Šaltenis, Elisa Bertino, Klemens Böhm, and Matthias Jarke, editors, *8th International Conference on Extending Database Technology (EDBT) Proceedings, Lecture Notes in Computer Science*, pages 756–758. Springer, March 2002b.
- R. Efron. The effect of handedness on the perception of simultaneity and temporal order. *Brain*, (86):261–284, 1963.
- T. Elbert. The processing of temporal intervals reflected by CNV-like brain potentials. *Psychophysiology*, 28:648–655, 1991.
- S. Fischer, J. Crabtree, B. Brunk, M. Gibson, and G. C. Overton. bioWidgets: data interaction components for genomics. *Bioinformatics*, 15(10):837–846, 1999.
- Monika Fleischmann, Wolfgang Strauss, Jasminko Novak, Stefan Paal, Boris Müller, and Gabriele Blome. netzspannung.org: an internet media lab for knowledge discovery in mixed realities. In *cast01 // living in mixed realities*, pages 121–129. FhG – Institut Medienkommunikation (IMK), 2001.
- David H. Foster. Does colour constancy exist? *Trends in Cognitive Sciences*, 7(10): 439–443, October 2003.
- Ian Foster, Carl Kesselman, Jeffrey M. Nick, and Steven Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Technical report, Open Grid Service Infrastructure WG, Global Grid Forum, June 2002.

- Alain Fournier and William T. Reeves. A Simple Model of Ocean Waves. In *Proceedings of ACM SIGGRAPH*, volume 20 of *Annual Conference Proceedings*, pages 75–84. ACM, ACM, 1986.
- Mark S. Fox and Fil Salustri. A Model for On-Off Systems Engineering. In *Proceedings of the Workshop on Artificial Intelligence and Systems Engineering*, Seattle, U.S.A., August 1994. AAAI. doi: <http://citeseer.nj.nec.com/fox94model.html>.
- Paul Freedman, Daniel Gaudreau, Raouf Boutaba, and Ahmed Mehaoua. Two Real-Time Solitudes: computerized control and telecommunications. In *Real-Time Applications Workshop*, pages 87–90. IEEE, 1996. doi: citeseer.nj.nec.com/freedman96two.html.
- Christian Freksa. Temporal Reasoning Based on Semi-Intervals. *Artificial Intelligence*, 52(1-2):199–227, 1992a.
- Christian Freksa. Using Orientation Information for Qualitative Spatial Reasoning. In A. U. Frank, I. Campari, and U. Formentini, editors, *Proceedings Int. Conference on Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, pages 162–178, Berlin, 1992b. Springer.
- Kenji Fukumizu. Learning in neural networks and an integrable system. *Applied Mathematics of Discrete Integrable Systems*, pages 23–41, 1999.
- Thomas Funke. Anreicherung von Illustrationen durch Anfragen an Suchmaschinen. Diploma thesis, University of Magdeburg, 2003.
- Thomas Funkhouser, Patrick Min, Michael Kazhdan, Joyce Chen, Alex Halderman, and David Dobkin. A Search Engine for 3D Models. *ACM Transactions on Graphics*, 22(1):83–105, 2003. doi: citeseer.nj.nec.com/funkhouser02search.html.
- Nathalie Furmento, Anthony Mayer, Stephen McGough, Steven Newhouse, Tony Field, and John Darlington. Optimisation of Component-based Applications within a Grid Environment. In *Proceedings of Super Computing*, 2001.
- George W. Furnas. Generalized fisheye views. In *Proceedings of ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 16–23, 1986.
- C. Gerlach, C.T. Aaside, G.W. Humphreys, A. Gade, O.B. Paulson, and I. Law. Brain activity related to integrative processes in visual object recognition: bottom-up integration and the modulatory influence of stored knowledge. *Neuropsychologia*, 40(8):1254–1267, 2002.
- G.E. Gerstner and V.A. Fazio. Evidence of a universal perceptual unit in mammals. *Ethology*, 101:89–100, 1995.

- Simon Gibbs. Composite multimedia and active objects. In *Proceedings of the International Conference on Object-Oriented Programming: Systems, Languages, and Applications*, pages 97–112, 1991.
- Simon Gibbs, Christian Breiteneder, and Dennis Tsichritzis. Audio/video databases: An object-oriented approach. In *Proceedings of the 9th International Conference on Data Engineering*, pages 381–390, 1993.
- Simon Gibbs, Christian Breiteneder, and Dennis Tsichritzis. Data Modelling of Time-Based Media. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 91–102, 1994.
- J.J. Gibson, G.A. Kaplan, H.N.J. Reynolds, and K. Wheeler. The change from visible to invisible: a study of optical transitions. *Perception & Psychophysics*, (5):113–116, 1969.
- Andrew Glassner. Maintaining winged-edge models. In James Arvo, editor, *Graphics Gems II*, pages 191–201. 1991.
- E. Bruce Goldstein. *Sensation and perception*, volume 6. Thomson Learning, 2002.
- Amy Gooch, Bruce Gooch, Peter Shirley, and Elaine Cohen. A Non-Photorealistic Lighting Model for Automatic Technical Illustration. In *Proceedings of ACM SIGGRAPH 1998, CGPACS*, pages 447–452. ACM SIGGRAPH, 1998.
- Bruce Gooch and Amy Gooch. *Non-Photorealistic Rendering*. A K Peters, Ltd., Natick, 2001. ISBN 1-56881-133-0. doi: <http://www.akpeters.com/book.asp?bID=131>.
- Bruce Gooch, Peter-Pike J. Sloan, Amy Gooch, Peter Shirley, and Richard Riesenfeld. Interactive Technical Illustration. In *1999 ACM Symposium on Interactive 3D Graphics*, pages 31–38. ACM, 1999.
- Richard L. Gregory. *Eye and Brain. The Psychology of Seeing*, volume 5. Oxford University Press, 1998.
- J. P. Grossman and William J. Dally. Point Sample Rendering. In *Eurographics Rendering Workshop*, pages 181–192, 1998.
- Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, and Henrik Frystyk Nielsen. SOAP Version 1.2 Part 1: Messaging Framework. Online document, available at <http://www.w3.org/TR/2003/REC-soap12-part1-20030624/>, June 2003a. W3C Recommendation.
- Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, and Henrik Frystyk Nielsen. SOAP Version 1.2 Part 2: Adjuncts. Online document, available at <http://www.w3.org/TR/2003/REC-soap12-part2-20030624/>, June 2003b. W3C Recommendation.

- Rainer Guski. *Wahrnehmen – ein Lehrbuch*. Kohlhammer, 1996.
- Hugo Haas, Oisín Hurley, Anish Karmarkar, Jeff Mischkinsky, Mark Jones, Lynne Thompson, and Richard Martin. SOAP Version 1.2 Specification Assertions and Test Collection. Online document, available at <http://www.w3.org/TR/2003/REC-soap12-testcollection-20030624/>, June 2003. W3C Recommendation.
- Nick Halper. *Supportive Presentation for Computer Games*. Dissertation, University of Magdeburg, 2003.
- Nick Halper, Tobias Isenberg, Felix Ritter, Bert Freudenberg, Oscar Meruvia, Stefan Schlechtweg, and Thomas Strothotte. OpenNPAR: A System for Developing, Programming, and Designing Non-Photorealistic Animation and Rendering. In Jon Rokne, Reinhard Klein, and Wenping Wang, editors, *Proceedings of Pacific Graphics*, pages 424–428. IEEE Computer Society, 2003. Short paper.
- Nick Halper, Stefan Schlechtweg, and Thomas Strothotte. Creating Non-Photorealistic Images the Designer’s Way. In Stephen N. Spencer, editor, *NPAR 2002*, pages 97–104. ACM SIGGRAPH, 2002.
- Jörg Hamel, Stefan Schlechtweg, and Thomas Strothotte. An Approach to Visualizing Transparency in Computer-Generated Line Drawings. In *Proceedings of IEEE Information Visualization*, pages 151–156. IEEE Computer Society, 1998.
- Steve Hankin, Jerry Davison, and D.E. Harrison. Web Visualization and Extraction of Gridded Climate Data with the FERRET Program. Online document, available at http://www.pmel.noaa.gov/ferret/ferret_climate_server.html, 1998.
- David Harel. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, 8(3):231–274, 1987. doi: citeseer.nj.nec.com/harel87statecharts.html.
- David Harel. On Visual Formalisms. *Communications of the ACM*, 31:514–531, 1988.
- Susan M. Harrison. A Comparison of Still, Animated, or Nonillustrated On-Line Help with Written or Spoken Instructions in a Graphical User Interface. In I.R. Katz, R. Mack, L. Marks, M.B. Rosson, and J. Nielsen, editors, *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 82–89. ACM, 1995.
- Knut Hartmann, Stefan Schlechtweg, Ralf Helbing, and Thomas Strothotte. Knowledge-Supported Graphical Illustration of Texts. In Maria De Marsico, Stefano Levialdi, and Emanuele Panizzi, editors, *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI 2002)*, pages 300–307. ACM Press, 2002.

- Susan Havre, Elizabeth Hetzler, Paul Whitney, and Lucy Nowell. ThemeRiver: Visualizing Thematic Changes in Large Document Collections. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):9–20, January-March 2002.
- David Hawking, Nick Craswell, and Donna Harman. Results and Challenges in Web Search Evaluation. *Computer Networks*, 31(11-16):1321–1330, 1999.
- Ralf Helbing, Knut Hartmann, and Thomas Strothotte. Dynamic Visual Emphasis in Interactive Technical Documentation. In Thomas Rist, editor, *Proc. of the Workshop on Combining AI and Graphics for the Interface of the Future*, pages 82–90, 1998.
- Ivan Herman, Guy Melançon, and M. Scott Marshall. Graph Visualization and Navigation in Information Visualization: a Survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, 2000.
- R.G. Herrtwich and L. Delgrossi. ODA-Based Data Modelling in Multimedia Systems. Technical Report tr-90-043, International Computer Science Institute, Berkley, 1990.
- Damien Hinsinger, Fabrice Neyret, and Marie-Paule Cani. Interactive Animation of Ocean Waves. In *Symposium on Computer Animation*, 2002.
- I. J. Hirsh and C. E. Sherrick. Perceived order in different sense modalities. *Journal of Exp. Psychology*, 62:423–432, 1961.
- P. Hoepner. Synchronizing the Presentation of Multimedia Objects—ODA Extensions. In *Proceedings of the First Eurographics Workshop on Multimedia, Systems, Interaction, and Applications*, pages 87–100, 1991.
- Gerard J. Holzmann. The Model Checker Spin. *Software Engineering*, 23(5):279–295, 1997. doi: citeseer.nj.nec.com/holzmann97model.html.
- James Hudson and Alan Parkes. Visual Overloading. In Constantine Stephanidis, editor, *HCI International 2003. Adjunct Proceedings*, pages 67–68. Crete University Press, 2003.
- Tobias Isenberg, Bert Freudenberg, Nick Halper, Stefan Schlechtweg, and Thomas Strothotte. A Developer’s Guide to Silhouette Algorithms for Polygonal Models. *IEEE Computer Graphics and Applications*, 23(4):28–37, 2003.
- ISO. Office Document Architecture (ODA): An Interchange Format. no. 8613, 1986.
- Ivar Jacobson, Magnus Christerson, Patrik Jonsson, and Gunnar Övergaard. *Object-Oriented Software Engineering*. Addison-Wesley, Reading, MA, 1992.

- Doug L. James and Kayvon Fatahalian. Precomputing Interactive Dynamic Deformable Scenes. In John C. Hart, editor, *Proceedings of ACM SIGGRAPH 2003*, volume 22 of *Annual Conference Series*, pages 879–887. ACM, ACM, 2003.
- Matt Jensen. Visualizing Complex Semantic Timelines. Technical Report NBTR2003-001, NewsBlip, 2003.
- Dean F. Jerding and John T. Stasko. The Information Mural: A Technique for Displaying and Navigating Large Information Spaces. *IEEE Transactions on Visualization and Computer Graphics*, 4(3):257–271, 1998.
- Stefan Jeschke, Hermann Birkholz, and Heidrun Schumann. A Procedural Model for Interactive Animation of Breaking Ocean Waves. In *WSCG Posters Proceedings*, 2003.
- Roland Jesse. Motion enhanced Information Mural for Climate Data Visualisation. In Konrad Wojciechowski, editor, *Proceedings of the International Conference on Computer Vision and Graphics (ICCVG 2002)*, volume 1, pages 374–379, 2002.
- Roland Jesse. Script-based presentation of simulation results. In Thomas Schulze, Stefan Schlechtweg, and Volkmar Hinz, editors, *Simulation und Visualisierung 2003*, pages 441–452, Magdeburg, March 2003. SCS-European Publishing House.
- Roland Jesse, Ingolf Geist, and Oliver Dunemann. Konzeption einer datenbankbasierten Plattform für die Informationsfusion. Technical report, University of Magdeburg, Magdeburg, 2001.
- Roland Jesse and Tobias Isenberg. Use of hybrid rendering styles for presentation. In Dirk Bartz and Vaclav Skala, editors, *Poster Proceedings of WSCG 2003*, 2003.
- Roland Jesse, Silvio Lange, Thomas Schulze, and Ulrich Klein. Animation in einer HLA-Federation. In Thomas Schulze, Peter Lorenz, and Volkmar Hinz, editors, *Simulation und Visualisierung 2000*, pages 55–67, Magdeburg, März 2000. Otto-von-Guericke Universität, SCS European Publishing House.
- Roland Jesse, Felix Ritter, and Thomas Strothotte. Bewegung als präemptive Präsentationsvariable in einem interaktiven System. In Thomas Schulze, Stefan Schlechtweg, and Volkmar Hinz, editors, *Simulation und Visualisierung 2002*, pages 275–288, Magdeburg, 2002. University of Magdeburg, SCS European Publishing House.
- Roland Jesse, Gunter Saake, Kai-Uwe Sattler, and Thomas Strothotte. Dynamic Visualisation for Feedback-driven Online Aggregation. In Jochen Schneider, Thomas Strothotte, and Winfried Marotzki, editors, *Proceedings of the Workshop on Computational Visualistics, Media Informatics and Virtual Communities*, number 11 in Image Science, pages 79–90. DUV, 2003.

- Roland Jesse and Marco Schumann. RTI-Benchmark. In *HLA Forum 2000*. Fraunhofer Institute for Factory Operation and Automation, March 2000.
- Roland Jesse and Thomas Strothotte. Motion Enhanced Visualization in Support of Information Fusion. In Hamid R. Arabnia, editor, *Proceedings of International Conference on Imaging Science, Systems, and Technology (CISST'2001)*, pages 492–497. CSREA Press, June 2001.
- Gunnar Johansson. Perception of motion and changing form. *Scandinavian Journal of Psychology*, 5(3):181–208, 1964.
- Gunnar Johansson. Visual perception of biological motion and a model for its analysis. *Perception and Psychophysics*, 14(2):201–211, 1973.
- Gunnar Johansson. Visual motion perception. *Scientific American*, (232):66–89, 1975.
- Brian Johnson and Ben Shneiderman. Tree-Maps: A Space Filling Approach to the Visualization of Hierarchical Information Structures. In *Proceedings of the 2nd International IEEE Visualization Conference*, pages 284–291, 1991.
- Scott F. Johnston. Lumo: Illumination for Cell Animation. In Stephen N. Spencer, editor, *NPAR 2002*, pages 45–52. ACM SIGGRAPH, 2002.
- Ari K. Jónsson and Jeremy D. Frank. A Framework for Dynamic Constraint Reasoning using Procedural Constraints. In *Proceedings of the European Conference on Artificial Intelligence*, 2000. doi: citeseer.nj.nec.com/jonsson00framework.html.
- Ralf Jungclaus, Gunter Saake, Thorsten Hartmann, and Cristina Sernadas. TROLL – A Language for Object-Oriented Specification of Information Systems. *ACM Transactions on Information Systems*, 14(2):175–211, 1996.
- Magdalena Kanabus, Elżbieta Szelaĝ, Ewa Rojek, and Ernst Pöppel. Temporal order judgement for auditory and visual stimuli. *Acta Neurobiologiae Experimentalis*, 62: 263–270, 2002.
- Daniel A. Keim. Information Visualization and Visual Data Mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):1–8, 2002.
- Rudolf K. Keller, Marianne Ozkan, and Xijin Shen. Towards Comprehensive Support for the Dynamic Analysis of Petri Net Based Models. In *Application and Theory of Petri Nets*, pages 298–317, 1994. doi: citeseer.nj.nec.com/keller94towards.html.
- B. Kiepuszewski, A.H.M. Hofstede, and W.M.P. van der Aalst. Fundamentals of Control Flow in Workflows. QUT Technical report FIT-TR-2002-03, Queensland University of Technology, Brisbane, 2002.

- Michelle Y. Kim and Junehwa Song. Multimedia Documents with Elastic Time. In *Proceedings of the third ACM international conference on Multimedia*, pages 143–154. ACM Press, 1995.
- Claire Knight. Visualisation Effectiveness. In Hamid R. Arabnia, editor, *2001 International Conference on Imaging Science, Systems, and Technology (CISST'2001)*. CSREA Press, 2001.
- Robert Kosara, Peter Messner, and Silvia Miksch. Time and Tide Wait for No Diagram. Technical Report Asgaard-TR-2001-2, Vienna University of Technology, Institute of Software Technology and Interactive Systems, 2001.
- Robert Kosara and Silvia Miksch. Metaphors of Movement: A Visualization and User Interface for Time-Oriented, Skeletal Plans. *Artificial Intelligence in Medicine*, 22(2):111–131, 2001. doi: citeseer.nj.nec.com/article/kosara01metaphors.html.
- Bart Krekelberg and Markus Lappe. Temporal Recruitment along the Trajectory of Moving Objects and the Perception of Position. *Vision Research*, 39:2669–2679, 1999.
- Matthias Kreuzeler and Heidrun Schumann. A Flexible Approach for Visual Data Mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):39–51, January-March 2002.
- Peter Kruse, Hans-Otto Carmesin, L. Pahlke, Daniel Strüber, and Michael Stadler. Continuous Phase Transitions in the Perception of Multistable Visual Patterns. ZKW-Bericht 3/96, Zentrum für Kognitionswissenschaften, Universität Bremen, 1996.
- John Lamping and Ramana Rao. The hyperbolic browser: A focus+context technique for visualizing large hierarchies. *Journal of Visual Languages and Computing*, 7(1):33–55, 1996.
- John Lamping, Ramana Rao, and Peter Pirolli. A Focus+Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies. In *Human Factors in Computing Systems, CHI '95 Conference Proceedings*. ACM, 1995.
- Markus Lappe and Bart Krekelberg. The Position of Moving Objects. *Perception*, 27:1437–1449, 1998.
- Alex Lascarides and Jon Oberlander. Temporal Coherence and Defeasible Knowledge. *Theoretical Linguistics*, 19(1):1–35, 1993.
- Miguel F. Leith and Jim Cunningham. Modelling Linguistic Events. In *Proceedings of the International Conference on Temporal Logic*, 1997.

- T. Lethbridge and C. Ware. Animation using behavior functions. In T. Ichikawa, E. Jungert, and R. Korfhage, editors, *Visual Languages and Applications*, pages 237–252. Plenum Press, New York, 1990.
- Y. K. Leung and M. D. Apperley. A review and taxonomy of distortion-oriented presentation techniques. *ACM Transactions on Computer-Human Interaction*, 1(2): 126–160, 1994. doi: <http://citeseer.nj.nec.com/leung94review.html>.
- Marc Levoy and Turner Whitted. The use of points as a display primitive. Technical Report TR85-022, UNC-Chapel Hill Computer Science, 1985.
- Henry Lieberman. Powers of Ten Thousand: Navigating in Large Information Spaces. In *Symposium on User Interface Software Technology*, pages 15–16. ACM, 1994.
- Thomas D. C. Little and Arif Ghafoor. Synchronization and Storage Models for Multimedia Objects. *IEEE Journal on Selected Areas in Communication*, 8(3):32–49, 1990.
- Thomas D. C. Little and Arif Ghafoor. Interval-Based Conceptual Models for Time-Dependent Multimedia Data. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):551–563, 1993.
- Peter Lorenz, Heiko Dorwarth, Klaus-Christoph Ritter, and Thomas J. Schriber. Towards a Web Based Simulation Environment. In *Proceedings of the Winter Simulation Conference*, pages 1338–1344, 1997.
- Peter Lorenz and Klaus-Christoph Ritter. Skopeo – A Platform-Independent System Animation for the W3. In Oliver Deussen and Peter Lorenz, editors, *Proceedings of Simulation and Animation*, pages 12–23. SCS Europe, 1997.
- Jock D. Mackinlay, George G. Robertson, and Stuart K. Card. The perspective wall: Detail and context smoothly integrated. In *Proceedings of the ACM Conference on Computer Human Interaction: CHI'91*, pages 173–180, 1991.
- Paul P. Maglio and Christopher S. Campbell. Tradeoffs in Displaying Peripheral Information. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI 2000)*, volume 2 of *CHI Letters*, pages 241–248. ACM, 2000.
- Zohar Manna and Amir Pnueli. *The temporal logic of reactive and concurrent systems: Specification*. Springer, 1992.
- Maic Masuch. *Nicht-Photorealistische Visualisierungen: Von Bildern zu Animationen*. Dissertation, Otto-von-Guericke-Universität Magdeburg, Magdeburg, 2001.

-
- Maic Masuch, Stefan Schlechtweg, and Ronny Schulz. Speedlines: Depicting Motion in Motionless Pictures. In *SIGGRAPH 99 Conference Abstracts and Applications*, Computer Graphics Proceedings, Annual Conference Series, pages 277–279. ACM SIGGRAPH, 1999.
- Maic Masuch, Lars Schumann, and Stefan Schlechtweg. Animating Frame-to-Frame Coherent Line Drawings for Illustrative Purposes. In *Proceedings of Simulation und Visualisierung*, pages 101–112. SCS Europe, 1998.
- Maic Masuch and Thomas Strothotte, editors. *Virtuelle Zeitreise: Der Computervisualistikraum in der Ausstellung »Otto der Große, Magdeburg und Europa«*. Otto-von-Guericke-Universität Magdeburg, Institut für Simulation und Graphik, 2001. ISBN 3980487415. ISBN: 3-9804874-1-5.
- Jiří Mates, Nicole von Steinbüchel, Marc Wittmann, and Bernhard Treutwein. A system for the assessment and training of temporal-order discrimination. *Computer Methods and Programs in Biomedicine*, 64:125–131, 2001.
- Nelson Max and Roger Crawfis. *Advances in Scientific Visualization*. IS&T/SPIE Symposium on Electronic Imaging: Science and Technology, 1995.
- Nelson L. Max. Vectorized Procedural Models for Natural Terrain: Waves and Islands in the Sunset. *Computer Graphics*, 15:317–324, 1981.
- Nilo Mitra. SOAP Version 1.2 Part 0: Primer. Online document, available at <http://www.w3.org/TR/2003/REC-soap12-part0-20030624/>, June 2003. W3C Recommendation.
- T. P. Moran. The Command Language Grammar: a representation for the user interface of interactive systems. *International Journal of Man-Machine Studies*, 15: 2–50, 1981.
- Julie Bauer Morrison, Barbara Tversky, and Mireille Beirancourt. Animation: Does It Facilitate Learning? In Andreas Butz, Antonio Krüger, and Patrick Olivier, editors, *Proceedings of Smart Graphics*, AAAI 2000 Spring Symposium Series, pages 53–60. AAAI Press – American Association for Artificial Intelligence, 2000.
- Sougata Mukherjea, Kyoji Hirata, and Yoshinori Hara. Visualizing the Results of Multimedia Web Search Engines. In *Proceedings IEEE Symposium on Information Visualization*, pages 64–65, 1996.
- Tamara Munzner. H3: Laying Out Large Directed Graphs in 3D Hyperbolic Space. In *Proceedings of the 1997 IEEE Symposium on Information Visualization*, pages 2–10, 1997.

- Bernd Nettelbeck. Zeitlich gesteuerte Änderung von Darstellungsstilen. Diploma thesis, Otto-von-Guericke University of Magdeburg, 2003.
- Romi Nijhawan. Visual decomposition of colour through motion extrapolation. *Nature*, 386:66–69, 1997.
- Emanuel G. Noik. Exploring Large Hyperdocuments: Fisheye Views of Nested Networks. In *Hypertext*, pages 192–205, 1993.
- Emanuel G. Noik. A Space of Presentation Emphasis Techniques for Visualizing Graphs. In *Proceedings of Graphics Interface '94*, pages 225–233, 1994.
- Alex Olwal and Steven Feiner. Rubbing the Fisheye: Precise Touch-Screen Interaction with Gestures and Fisheye Views. In *Symposium on User Interface Software and Technology*, pages 83–84. ACM, 2003.
- OMG. Unified modeling language (uml), version 1.5. Online document, available at <http://www.omg.org/technology/documents/formal/uml.htm>, 2003.
- Larry Page and Sergey Brin. Google PageRank Explained. Online document, available at <http://www.google.com/technology/>, 1998.
- Philippe Palanque and Rémi Bastide. Time Modelling in Petri Nets for the Design of Interactive Systems. *SIGCHI Bulletin*, 28(2):43–46, April 1996.
- Darwyn R. Peachey. Modelling Waves and Surf. In *Proceedings of ACM SIGGRAPH*, volume 20 of *Annual Conference Series*, pages 65–74. ACM, ACM, 1986.
- Knud Pehrs. Illustration von Klimadaten. Diplom, University of Applied Sciences Oldenburg / Ostfriesland / Wilhelmshaven, 2004.
- Carl Adam Petri. *Kommunikation mit Automaten*. Schriften des Rheinisch-Westfälischen Institutes für Instrumentelle Mathematik, No. 2, Universität Bonn: Institut für Instrumentelle Mathematik, 1962. Also in English translation: Communication with Automata. Tech. Report RADC-TR-65-377, Vol. 1, Suppl 1, Applied Data Research, Princeton, NJ, 1966.
- Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar, and Markus Gross. Surfels: Surface elements as rendering primitives. In *Proceedings of ACM SIGGRAPH 2000*, Annual Conference Series, pages 335–342. ACM, ACM, 2000.
- Catherine Plaisant, Brett Milash, Anne Rose, Seth Widoff, and Ben Shneiderman. LifeLines: Visualizing Personal Histories. In *Proceedings of CHI*, pages 221–227, 1996. doi: citeseer.nj.nec.com/plaisant96lifelines.html.

- Frits H. Post, Benjamin Vrolijk, Helwig Hauser, Robert S. Laramée, and Helmut Doleisch. The State of the Art in Flow Visualization: Feature Extraction and Tracking. *Computer Graphics Forum*, 22(4), 2003.
- Ernst Pöppel. Oscillations as possible basis for time perception. *Studium Generale*, 24:85–107, 1971.
- Ernst Pöppel. *Handbook of Sensory Physiology and Perception*, volume 8, pages 713–729. Springer, 1978.
- Ernst Pöppel. Temporal mechanisms in perception. *Int. Rev. Neurobiol.*, 37:185–202, 1994.
- Ernst Pöppel. A hierarchical model of temporal perception. *Trends in Cognitive Sciences*, 1(2):56–61, 1997.
- B. Prabhakaran. *Multimedia Database Management Systems*, volume 375 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, 1997.
- Bernhard Preim, Andreas Raab, and Thomas Strothotte. Coherent Zooming of Illustrations with 3D-Graphics and Text. In *Proceedings of Graphics Interface*, pages 105–113, 1997.
- Bernhard Preim and Felix Ritter. Techniken zur interaktiven Hervorhebung von Objekten in medizinischen 3D-Visualisierungen. In Thomas Schulze, Stefan Schlechtweg, and Volkmar Hinz, editors, *Simulation und Visualisierung 2002*, pages 187–200, Ghent, Belgium, March 2002. SCS-Society for Computer Simulation Int., SCS European Publishing House.
- Zenon W. Pylyshyn, J. Burkell, B. Fisher, C. Sears, W. Schmidt, and L. Trick. Multiple parallel access in visual attention. *Canadian Journal of Experimental Psychology*, 1993.
- Andreas Raab and Michael Rüger. 3D-ZOOM: Interactive Visualisation of Structures and Relations in Complex Graphics. In B. Girod, H. Niemann, and H.-P. Seidel, editors, *3D Image Analysis and Synthesis*, pages 125–132. infix-Verlag, Sankt Augustin, 1996.
- Uwe Rauschenbach, Stefan Jeschke, and Heidrun Schumann. General Rectangular FishEye Views for 2D Graphics. *Computer and Graphics*, 25(4):609–617, 2001.
- William T. Reeves. Particle Systems – A Technique for Modelling a Class of Fuzzy Objects. In *Proceedings of ACM SIGGRAPH 1983*, volume 17 of *Annual Conference Proceedings*, pages 359–376. ACM, ACM, 1983.

- Freek Reinders, Frits H. Post, and Hans J. W. Spoelder. Visualization of Time-Dependent Data using Feature Tracking. In *The Visual Computer*, volume 17(1), pages 55–71. Springer, 2001.
- P. Reisner. Formal grammar and human factors design of an interactive graphics system. *IEEE Transactions on Software Engineering*, 2:229–240, 1981.
- Jean-Francois Rit. Propagating temporal constraints for scheduling. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 383–388. MIT Press, 1986.
- Felix Ritter, Bernhard Preim, Oliver Deussen, and Thomas Strothotte. Using a 3D Puzzle as a Metaphor for Learning Spatial Relations. In *Proceedings of Graphics Interface 2000*, pages 171–178, Montréal, Qc, Canada, May 2000.
- Felix Ritter, Henry Sonnet, Knut Hartmann, and Thomas Strothotte. Illustrative Shadows: Integrating 3D and 2D Information Displays. In W. Lewis Johnson, Elisabeth André, and John Domingue, editors, *Proceedings of 2003 International Conference on Intelligent User Interfaces (IUI; Miami, Florida January 12-15, 2003)*, pages 166–173, New York, 2003. ACM. ISBN 1-58113-586-6. doi: <http://doi.acm.org/10.1145/604045.604072>.
- M. Roantree, P. Hickey, A. Crilly, and J. Murphy. Metadata Modelling for Healthcare Applications in a Federated Database System. In *Trends in Distributed Systems: CORBA and Beyond, International Workshop TreDS '96, Aachen, Germany, Lecture Notes in Computer Science No. 1161*, pages 71–83. Springer-Verlag, 1996.
- G. Robertson, Jock D. Mackinlay, and S.K. Card. Cone trees: Animated 3d visualizations of hierarchical information. In *ACM CHI*, pages 189–194, 1991.
- Alan J. Robinson. Ebi hyperbolic viewer. Online document, available at <http://industry.ebi.ac.uk/~alan/components>, 1998. European Bioinformatics Institute.
- J. Rumbaugh, M. Blaha, W. Premerlai, and F. Eddy. *Object-oriented modelling and design*. Prentice Hall, Englewood Cliffs, NJ, 1991.
- Szymon Rusinkiewicz and Marc Levoy. QSplat: A Multiresolution Point Rendering System for Large Meshes. In *Proceedings of ACM SIGGRAPH 2000, Annual Conference Series*, pages 343–352. ACM, ACM, 2000.
- Szymon Rusinkiewicz and Marc Levoy. Streaming QSplat: A Viewer for Networked Visualization of Large, Dense Models. In *Symposium on Interactive 3D Graphics*, pages 63–68, 2001.

-
- Takafumi Saito and Tokiichiro Takahashi. Comprehensible Rendering of 3-D Shapes. In Forest Baskett, editor, *Proceedings of ACM SIGGRAPH 1990*, CGPACS, pages 197–206, New York, 1990. ACM SIGGRAPH, ACM Press.
- C. A. S. Santos, Paulo Nazareno Maia Sampaio, and Jean-Pierre Courtiat. Revisiting the concept of hypermedia document consistency. In *ACM Multimedia*, volume 2, pages 183–186, 1999. doi: citeseer.nj.nec.com/santos99revisiting.html.
- Manojit Sarkar and Marc H. Brown. Graphical Fisheye Views of Graphs. In *Proceedings of ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 83–91, 1992.
- Kai-Uwe Sattler, Stefan Conrad, and Gunter Saake. Adding Conflict Resolution Features to a Query Language for Database Federations. *Australian Journal of Information Systems*, 8(1):116–125, 2000.
- Stefan Schlechtweg and Thomas Strothotte. Illustrative Browsing: A New Method of Browsing in Long On-line Texts. In S. Brewster, A. Cawsey, and G. Cockton, editors, *Human-Computer Interaction – INTERACT '99*, volume II, pages 466–473. IFIP, 1999.
- Stefan Schlechtweg and Thomas Strothotte. Generating Scientific Illustrations in Digital Books. In *Smart Graphics. Papers from the 2000 AAAI Spring Symposium*, pages 8–15. AAAI Press, 2000.
- M. Schleidt, I. Eibl-Eibesfeldt, and Ernst Pöppel. A universal constant in temporal segmentation of human short-term behaviour. *Naturwissenschaften*, 74:289–290, 1987.
- Will Schroeder, Ken Martin, and Bill Lorensen. *The Visualization Toolkit – An Object-Oriented Approach to 3D Graphics*. Prentice Hall PTR, 2. edition, 1998.
- Heidrun Schumann and Wolfgang Müller. *Visualisierung*. Springer-Verlag, Berlin Heidelberg New York, 2000.
- M. Sherif. A study of some social factors in perception. *Archives of Psychology*, 27: 311–328, 1935.
- Ben Shneiderman. Multiparty grammars and related features for defining interactive systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 12(2):148–154, 1981.
- António Rito Silva, Francisco Assis Rosa, and Teresa Gonçalves. Distributed Proxy: A Design Pattern for Distributed Object Communication. In *The 4th Conference on Pattern Languages of Programming, PLoP '97*, 1997.

- Amy Skopik and Carl Gutwin. Finding Things in Fisheyes: Memorability in Distorted Spaces. In *Proceedings of the 2003 Conference on Graphics Interface (GI'03)*, 2003.
- Margaret H. Smith, Gerard J. Holzmann, and Kousha Etessami. Events and Constraints: A Graphical Editor for Capturing Logic Requirements of Programs. In *Fifth IEEE International Symposium on Requirements Engineering*, pages 14–22, 2001.
- Robert J. Snowden and Frans A. J. Verstraten. Motion transparency: making models of motion perception transparent. *Trends in Cognitive Sciences*, 3(10):369–377, October 1999.
- Michael Stal. Web Services: Beyond Component-based Computing. *Communications of the ACM*, 45(10):71–76, October 2002.
- Jos Stam. Flows on Surfaces of Arbitrary Topology. In John C. Hart, editor, *Proceedings of ACM SIGGRAPH 2003*, volume 22 of *Annual Conference Series*, pages 724–731. ACM, ACM, 2003.
- Anton Stankowski and Karl Dushek. *Visuelle Kommunikation*. Reimer, 2 edition, 1994.
- Jennifer A. Stevens, Pierre Fonlupt, Maggie Shiffrar, and Jean Decety. New aspects of motion perception: Selective neural encoding of apparent human movements. *NeuroReport*, 11:109–115, 2000.
- Dietrich Stoyan and Daryl J. Daley, editors. *Comparison Methods for Queues and Other Stochastic Models*. Wiley, 1984.
- Steffen Straßburger, Thomas Schulze, Ulrich Klein, and J. O. Henriksen. Internet-based Simulation using off-the-shelf Simulation Tools and HLA. In D. J. Medeiros and E. Watson, editors, *Proceedings of the 1998 Winter Simulation Conference*, Washington D.C., 1998.
- Paul S. Strauß and Rikk Carey. An object-oriented 3D graphics toolkit. In Edwin E. Catmull, editor, *Proceedings of ACM SIGGRAPH 1992*, volume 26 of *Annual Conference Series*, pages 341–349. ACM, 1992.
- Thomas Strothotte. *Computational Visualization – Graphics, Abstraction, and Interactivity*. Springer, Berlin Heidelberg New York, 1998.
- Thomas Strothotte, Matthias Puhle, Maic Masuch, Bert Freudenberg, Sebastian Kreiker, and Babette Ludowici. Visualizing Uncertainty in Virtual Reconstructions. In *Proceedings of Electronic Imaging & the Visual Arts, EVA Europe '99*, page 16, Berlin, 1999. VASARI, GFaI.

- Thomas Strothotte and Stefan Schlechtweg. *Non-Photorealistic Computer Graphics: Modeling, Rendering and Animation*. Morgan Kaufmann, San Francisco, 2002.
- L. Swisher and I. J. Hirsh. Brain damage and the ordering of two temporally successive stimuli. *Neuropsychologia*, 10:137–152, 1972.
- P. Tallal, M. M. Merzenich, S. Miller, and W. Jenkins. Language learning impairments: integrating basic science, technology, and remediation. *Exp. Brain Res.*, 123:210–219, 1998.
- P. Y. Tso and B. A. Barsky. Modelling and Rendering Waves: Wave-Tracing Using Beta-Splines and Reflective and Refractive Texture Mapping. *ACM Transactions on Graphics*, 6(3):191–214, 1987.
- Edward Tufte. *Envisioning Information*. Graphics Press, Cheshire, Conn., 1990.
- Greg Turk. Generating Textures on Arbitrary Surfaces Using Reaction-Diffusion. In *Proceedings of ACM SIGGRAPH*, volume 25 of *Annual Conference Series*, pages 289–298. ACM, ACM, 1991.
- Jarke J. van Wijk. Spot noise – texture synthesis for data visualization. In Thomas W. Sederberg, editor, *Computer Graphics (Proceedings of ACM SIGGRAPH)*, volume 25, pages 263–272, 1991.
- Nicole von Steinbüchel, Marc Wittmann, and Ernst Pöppel. Timing in Perceptual and Motor Tasks after Disturbances of the Brain. In M.A. Pastor and J. Artieda, editors, *Time, Internal Clocks and Movements*, pages 281–304. Elsevier Science, 1996.
- Nicole von Steinbüchel, Marc Wittmann, Hans Strasburger, and Elzbieta Szlag. Auditory temporal order judgement is impaired in patients with cortical lesions in posterior regions of the left hemisphere. *Neuroscience Letters*, 264:168–171, 1999a.
- Nicole von Steinbüchel, Marc Wittmann, and E. Szlag. Temporal constraints of perceiving, generating, and integrating information: Clinical indications. *Restorative Neurology and Neuroscience*, 14:167–182, 1999b.
- Thomas Wahl and Kurt Rothermel. Representing Time in Multimedia Systems. In *International Conference on Multimedia Computing and Systems*, pages 538–543, 1994. doi: citeseer.nj.nec.com/article/wahl94representing.html.
- Colin Ware. *Information Visualization: Perception for Design*. Interactive Technologies. Morgan Kaufmann Publishers, 2000.
- Colin Ware, K. Arthur, and K. S. Booth. Fish tank virtual reality. In *InterCHI*, pages 37–42, 1993.

- Colin Ware and Glenn Franck. Evaluating stereo and motion cues for visualizing information nets in three dimensions. *ACM Transactions on Graphics*, 15(2):121–140, April 1996.
- Colin Ware, Eric Neufeld, and Linda R. Bartram. Visualizing Causal Relations. In *Proceedings Information Visualization 1999*, San Francisco, USA, 1999.
- Gerhard Weber. *Temporale Modellierung multimedialer interaktiver Systeme*. Shaker Verlag, 2000. Habilitation.
- M. Weber, M. Alexa, and W. Mueller. Visualizing Time-Series on Spirals. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis'01)*, pages 7–14. IEEE Press, 2001.
- P. Wegner. Why interaction is more powerful than algorithms. *Communications of the ACM*, 40(5):81–91, 1997.
- Josie Wernecke. *The Inventor Mentor*, volume 2. Addison Wesley, 1994a.
- Josie Wernecke. *The Inventor Toolmaker*, volume 2. Addison Wesley, 1994b.
- Pak Chung Wong. Visual Data Mining. *IEEE Computer Graphics and Applications*, 19(5):20–21, September/October 1999.
- Sophie M. Wuerger, Huw Owens, and Stephen Westland. Blur tolerance in different colour directions. In *Proceedings of International Conference on Color in Graphics and Image*, pages 305–310, 2000.
- Christopher C. Yang, Hsinchun Chen, and Kay Hong. Exploring the World Wide Web with Self-Organizing Map. In *Proceedings of the WWW 2002 Conference*, 2002. Online document, available at <http://www2002.org/CDROM/poster/189.pdf>.
- Jian Yang and Mike. P. Papazoglou. Web Component: A Substrate for Web Service Reuse and Composition. In *Proceedings of the 14th International Conference on Advanced Information Systems Engineering (CAiSE'02)*, 2002.
- Myung K. Yang and Chita R. Das. An Evaluation of the Temporal Coherence Heuristic in Partial-Order Planning. *IEEE Transactions on Parallel and Distributed Systems*, 5(1):74–86, 1994.
- Ellen J. Scher Zagier. Defining and Refining Frameless Rendering. Technical Report TR97-008, University of North Carolina at Chapel Hill, 1997.
- Oren Zamir and Oren Etzioni. Grouper: A Dynamic Clustering Interface to Web Search Results. *Computer Networks*, 31(11-16):1361–1374, 1999.

Hrvoje Zorc. Strategy of Design of Sophisticated Optical Multilayers. *Fizika A*, 4(2): 263–269, 1995.

Appendix

A Description of Document Formats

A.1 Definition of the *Tigeop* Schema

Listing A.1 shows the *Tigeop* Schema definition. Individual types of the Schema represent direct mappings of the formal notations presented in Sections 4.3 and 4.4. Lines 11–18 of the listing define the overall structure of a document instance. As presented in Section 5.4.3, a *Tigeop* document consists of:

- A description of employed dynamics presentation techniques,
- The geometric scene description, and
- A set of object presentation definitions.

The *string* type for the geometry definition in line 13 is to be interpreted as containing a scene graph in Open Inventor format.

Presentation methods are defined in lines 19–33 and bridge between the attribute set **AM** of Kreuzeler and Schumann’s framework as outlined in Section 4.3 and the set \mathcal{D} of the temporal parameterisation function δ from Section 4.4. Attributes for the individual methods are defined as *methodProperty*s in lines 30–33. Their content model is empty. Therefore the *property* element for each *presentation method* conveys both the name and the value of a specific method attribute. Depending on the content of these attributes in a document instance, its resulting rendering is to be interpreted as producing either photorealistic or non-photorealistic image components.

The heart of a document in regard to the temporal modelling of hybrid rendering styles is the definition of object presentations in lines 34–48. These presentations represent the evaluation of individual rendering styles at defined time points for the given durations. The temporal dependencies conform to the above algebra definition. Similar to the method properties mentioned above, the content model for an object’s *present* description is empty which results in all parameters being specified as element attributes. Exemplary excerpts of document instances showing the use of hybrid rendering style descriptions are presented in the next section.

Listing A.1: XML Schema definition for *Tigeop*.

```
<?xml version = '1.0' encoding = 'UTF-8'?>  
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
```

```
<xsd:annotation>
  <xsd:documentation xml:lang="en">
    Schema for temporally influenced geometry presentation.
    Copyright (c) 2002–4 University of Magdeburg.
    All rights reserved.
  </xsd:documentation>
</xsd:annotation>
<xsd:element name="tigeop" type="tigeopType"/>

<xsd:complexType name="tigeopType">
  <xsd:sequence>
    <xsd:element name="presentationmethods"
      type="presentationMethodType"
      minOccurs="0"/>
    <xsd:element name="ivscene"
      type="xsd:string"
      minOccurs="1" maxOccurs="1"/>
    <xsd:element name="sceneobjects"
      type="sceneObjectsType"
      maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="presentationMethodType">
  <xsd:sequence>
    <xsd:element name="method"
      type="methodType"
      maxOccurs="unbounded">
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="methodType">
  <xsd:attribute name="id" type="xsd:integer"/>
  <xsd:attribute name="name" type="xsd:string"/>
  <xsd:sequence>
    <xsd:element name="property"
      type="methodPropertyType"
      maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="methodPropertyType">
  <xsd:attribute name="name" type="xsd:string"/>
  <xsd:attribute name="value" type="xsd:string"/>
</xsd:complexType>
```



```

<xsd:complexType name="sceneObjectsType">
  <xsd:sequence>
    <xsd:element name="object "
      type="sceneObjectType "
      maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="sceneObjectType">
  <xsd:attribute name="id " type="xsd:string"/>
  <xsd:attribute name="name " type="xsd:string"/>
  <xsd:element name="present " type="presentObjectType"/>
</xsd:complexType>
<xsd:complexType name="presentObjectType">
  <xsd:attribute name="method " type="xsd:integer"/>
  <xsd:attribute name="start " type="xsd:dateTime"/>
  <xsd:attribute name="duration " type="xsd:time"/>
</xsd:complexType>
</xsd:schema>

```

A.2 Exemplary XML Document According to the Tigeop-Schema

Listing A.2 shows a small example of a presentation document based on the above *Tigeop* Schema. This presentation consists of a scene with only one object. This object is rendered with silhouette NPR style using a chalk texture. Use of this NPR emphasis is only valid for a specific interval during the overall presentation.

Listing A.2: Example of a *Tigeop* presentation document.

```

<?xml version="1.0" encoding="UTF-8"?>
<tigeop xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="Tigeop.xsd">
  <presentationmethods>
    <method id="1" name="Chalky">
      <property name="texture" value="chalky.xpm" />
      <property name="linethickness" value="7" />
    </method>
  </presentationmethods>
  <ivscene>
    #Inventor V2.1 ascii

```

```
Separator {
  BaseColor {
    rgb      0.5 0.5 0.95
  }
  Complexity {
    value    0.2
  }
  Sphere {
  }
}
</ivscene>
<sceneobjects>
  <object id="1" name="Sphere">
    <present method="1" start="PT7.5S" duration="PT4.2S" />
  </object>
</sceneobjects>
</tigeop>
```

The temporal specifications are constructed as follows:

- Each time stamp respectively each specification of a duration starts with a *P*.
- Year, month, and day specifications follow.
- The specification of time stamps shorter than a day are prefixed with a *T*.
- Each number is followed by its »unit« (*H* for hours, *M* for minutes, *S* for seconds).

Undocumented time spans with regard to rendering styles are to be presented classically, that is by use of photorealistic rendering.

A.3 WSDL Schema

As discussed in Section 5.4.3, a WSDL document describes the interface and implementation of a web service protocol. The following two subsections present the according *Tigeop* documents.

A.3.1 Description of the *Tigeop* Interface

Listing A.3 shows the interface description of *Tigeop*. The following possible messages are specified by this WSDL description:

- `setScene` with *String* as parameter type which requires a scene description in Open Inventor format.
- `presentObject` with following parameter/type pairs:
 - `id` : *String*
 - `start` : *Date*
 - `duration` : *Date*
 - `property` : *StringSequence*

The operations that follow out of this specification are direct representations of the messages and therefore `setScene` and `presentObject`.

Listing A.3: *Tigeop* interface description.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="TigeopInterfaceDescription"
  targetNamespace="urn:TigeopInterface"
  xmlns:tns="urn:TigeopInterface"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">

  <wsdl:message name="setScene">
    <part name="name" type="xsd:string" />
  </wsdl:message>

  <wsdl:message name="presentObject">
    <part name="id" type="xsd:string" />
    <part name="start" type="xsd:dateTime" />
    <part name="duration" type="xsd:time" />
    <part name="property" type="stringSequence" />
  </wsdl:message>

  <wsdl:portType name="TigeopInterface">
    <wsdl:operation name="setScene">
      <wsdl:input message="tns:setScene" />
    </operation>
    <wsdl:operation name="presentObject">
      <wsdl:input message="tns:presentObject" />
    </wsdl:operation>
  </wsdl:portType>

  <wsdl:binding name="TigeopBinding"
```

```
        type="tns:TigeopInterface">
<!-- This is no classic RPC over HTTP-->
<!-- but a custom Qt transport.-->
<soap:binding
  style="rpc"
  transport="http://schemas.xmlsoap.org/soap/http" />
<wsdl:operation name="setScene">
  <soap:operation soapAction="urn:Tigeop" />
  <wsdl:input>
    <soap:body
      use="encoded"
      namespace="urn:Tigeop"
      encodingStyle=
        "http://schemas.xmlsoap.org/soap/encoding/" />
    </wsdl:input>
  </wsdl:operation>
<wsdl:operation name="presentObject">
  <soap:operation soapAction="urn:Tigeop" />
  <wsdl:input>
    <soap:body
      use="encoded"
      namespace="urn:Tigeop"
      encodingStyle=
        "http://schemas.xmlsoap.org/soap/encoding/"
    />
  </wsdl:input>
</wsdl:operation>
</wsdl:binding>
</wsdl:definitions>
```

A.3.2 Description of the *Tigeop* Implementation

The *Tigeop* implementation description is provided in Listing A.4. This description defines that the *Tigeop* service is provided via host *arthur.cs.uni-magdeburg.de* as a web service.

Listing A.4: *Tigeop* implementation description.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="TigeopImplementationDescription"
  targetNamespace="urn:TigeopImplementation"
  xmlns:tns="urn:TigeopImplementation"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
```

```
xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/">
<wSDL:import namespace="urn:TigeopInterface"
             location="TigeopInterfaceDescription.wSDL" />
<wSDL:service name="TigeopService">
  <wSDL:port name="TigeopPort"
            binding="TigeopBinding">
    <!-- Location of the Tigeop service -->
    <soap:address
      location="tigeop://arthur.cs.uni-magdeburg.de/" />
  </wSDL:port>
</wSDL:service>
</wSDL:definitions>
```


B List of Supervised Works

- Name:* Syed Kamran Ali Ahmad
Title: Visualization of Nuclear Scattering Data
Co-Advisor: Jens Krüger, Technical University of Munich
- Name:* Andrea Schauen
Title: A Particle System based Simulation of a Flow Process in Industry
Advisor: Brent Baxter, Siemens Corp. Research Princeton, NJ, USA
- Name:* Niklas Röber
Title: Visualization of Fuel Cell Simulations
Co-Advisors: Tobias Isenberg and Torsten Möller
- Name:* Bernd Nettelbeck
Title: Zeitlich gesteuerte Änderung von Darstellungsstilen
Co-Advisor: Tobias Isenberg
- Name:* Martin Kellermann
Title: Framework for an MEMS Animated Graphic Design Aid
Advisor: Dr. Renate Sitte
- Name:* Alexandra Baer
Title: Elements of a Draw Desk for Specific MEMS Devices in MAGDA
Advisor: Dr. Renate Sitte
- Name:* Verena von Hintzenstern
Title: Editing of MEMS Elements for Interactive Visualisation
Advisor: Dr. Renate Sitte
- Name:* Iryna Davydova
Title: Nichtfotorealistische Verzerrungen zu Illustrationszwecken
- Name:* Thomas Funke
Title: Anreicherung von Illustrationen durch Anfragen an Suchmaschinen
- Name:* Knud Pehrs
Title: Illustration von Klimadaten
Co-Advisor: Prof. Dr. Frank Klawonn (University of Applied Sciences Oldenburg / Ostfriesland / Wilhelmshaven)

Index

- 3D fisheye, 44
- 3D illustration model, 161
- 3D-Zoom, 74

- a priority importance, 42, 47
- absolute dates, 124
- abstract information landscape, 176
- abstraction
 - from graphics hardware, 135
- access to data sources, 175
- actions, 34, 110
- active exploration, 179
- active objects, 111
- active search queries, 165
- acyclic graph, 106, 154
- adaptive illustration, 176
- adaptivity for application areas, 149
- adjuncts, 149
- aftereffect, *see* movement→apparent
- aggregation, 175
 - control, 177
- alarms
 - detection and processing, 93
- α -blending, 68
- alter vertex positions, 55
- altitude, 171
- amount of overlapping, 57
- amount of volume, 80
- amplitude range, 57
- analysis and abstraction, 151
- anatomical context, 179
- Angie, 152

- animated
 - graphical layering, 14
 - transparent layers, 14
- application
 - telecommunication, 93
- applications, 157
- archimedic spiral, 12
- ArcInfo, 173
- ArcView, 173
- arrows, 44
- artifacts, *see* stimuli→unrecognisable
- AsbruView, 96
- aspect ratio, 75
- assertions and test collection, 149
- asynchronous parallel composition, 113
- attribute
 - function, 33
 - set, 33
- automatic learning, 32
- available information sources, 151
- awareness, 94
 - tools, 26
- axioms, 114

- bank-customer relationship, 174
- batch operation, 175
- bearing device, 180
- beyond nested graphs, 43
- binary parallel composition, 113
- bindings, 150
- bioinformatics, 174
- blinking, 165

- bounding box, 44
- branching states, 105
- browsing, 164
- built-in priorities, 42

- cartoon shading, 66
- change
 - of colour, 26, 45
 - over time, 18
- chewing gum
 - iteratively defined, 82
- chewing gum
 - morph function, 81
 - ripped up, 80
- circular glyph representations, 160
- civilised environments, 9
- classes of bank customers, 174
- client priorities, 42
- client/server environment, 147
- climate data, 24, 170
- cloud rendering, 172
- clusters, 159
- Co-nets, 106
- cognitive
 - load, 27
 - resolving task, 30
- cognitive limitations, 45
- cognitive restrictions, 95
- coherence
 - arc-length, 95
 - frame-to-frame, 95
- collaborative information pools, 94
- colour
 - cultural background, 45
 - dark purple, 45
 - emotional state, 45
 - green-blue-purple area, 45
 - grief, 45
 - negative effect, 45
 - positive appeal, 45
 - saturation, 45
 - yellow-black combinations, 45
 - yellow-red area, 45
- colour blinking, 26
- communicating sequential process, 113
- communication bandwidth, 31
- communication interface protocol, 149
- complete
 - knowledge, 101
 - reordering, 100
 - temporal knowledge, 93
- completeness, 43
 - of a result set, 155
- component-based, 161
 - middleware, 152
- components, 133
- composition methods, 111
- computerised control systems, 93
- computing operation, 93
- computing-based algorithms, 108
- conceptional neighbourhood, 93
- concrete location points, 171
- concurrency, 94, 111
 - constrained, 90
 - evaluation function, 122
- concurrent
 - systems, 114
 - trajectories, 121
- concurrent dynamics
 - layer model, 121
- cone trees, 47
- connection threshold, 80
- consecutive scheduling, 127
- constrain, 148
- constraining parameterisation, 30
- constraints
 - violation of, 95
- construction of information objects, 49, 61
- context information, 31
- continuity constraints, 127
- contour, 78
- contour lines, 44, 76
- contours, 172

-
- control function
 - temporal, 124
 - convex hull, 28
 - cooling aggregate, 178
 - coordinate management methods, 111
 - Corba, 148
 - cosine function, 136
 - crosshair, 44
 - crossing the cognitive gap, 163
 - CSCW, 20
 - curvilinear grid, 171
 - cutaway view, 44
 - cycles per second, 137
 - cyclic
 - petri nets, 106
 - relations, 97
 - daily weather data, 171
 - data
 - access, 151
 - daily weather, 171
 - extract patterns from distribution, 62
 - heterogeneous sources, 150
 - integration, 151
 - mining, 151
 - multidimensional, 96
 - real world, 33
 - semantically connected, 12
 - sources, 42
 - stream, 20
 - tables, 34
 - temporal, 96
 - temporal flow, 172
 - database
 - integration, 151
 - technology, 150
 - databases, 90
 - deformation
 - dinosaur, 21
 - degree
 - of interest, 47, 75
 - of similarity, 159
 - of variation, 93
 - degrees of freedom, 86
 - delayed blending, 68
 - demagnification, 73
 - demarcation, 44
 - detailed design phase, 178
 - detection of silhouette edges, 141
 - dimensionality, 90
 - dimensions, 33
 - directionTurn, 140
 - disguise objects, 45
 - distance, 41, 63, 79
 - function, 47
 - distance field metamorphosis, 82
 - distant points, 42
 - distinction property, 121
 - distorted presentation, 42
 - distortion
 - fisheye, 47
 - histories, 72
 - distributed applications, 148
 - DMOZ, 162
 - document summary, 162
 - dot density, 14
 - double
 - interval, 119
 - pairs of faces, 54
 - double-linked lists, 141
 - dual use of image space, 28
 - duration, 14
 - duration specification, 124
 - dynamic
 - hierarchy computation, 34
 - textures, 23
 - dynamics
 - by distortions, 28
 - by rendering styles, 65
 - use of, 164, 166, 167, 176, 177
 - classes, 122
 - cognition, 9

- combination with static techniques, 37
- composite patterns, 29
- controlling, 89
- effective use of, 25
- evaluation of schedules, 123
- fading out, 124
- full potential, 89
- fundamentals of, 9
- global, 29
- landscape, 22
- local versus global, 30
- merging, 124
- patterns
 - consecutive, 118, 120
 - modifiable, 118, 120
- perception, 9
- reduced number of, 124
- representation
 - parameterisation, 120
- set of, 146
- trajectory, 116
 - events, 117
 - intervals, 118
 - single, 116
- use of, 157
- dynamics management unit, 148
- dynamics stimulus window, 15–17, 72, 90, 122, 129
- edge
 - concatenation, 141
 - lookup, 140
- edges, 141
- effect on scene coherence, 85
- elastic
 - time, 104
 - transformation, 82
- emissive colour, 144
 - blending, 70
- emotional state, 45
- emphasis
 - classification, 43
 - property, 40
- engine, 138
- engineering models, 177
- ensure visibility, 43, 45
- ensuring scene consistency, 121
- essence of information, 31
- ETL, *see* evolving temporal logic
- event
 - function, 117
 - group, 117
 - prefix, 113
 - trace diagrams, 106
- events
 - as time steps, 91
 - being intervals, 109
 - comparison of, 89
 - complete linear order, 124
 - external, 94
 - handling of, 91
 - inexact, 97
 - insertion of, 89
 - internal, 94
 - nondeterministic, 94
 - pre-modelling of, 94
 - reactions on, 96
- evolving temporal logic, 114
- examination
 - of anatomical objects, 179
 - of results, 163
- exemplary scatterplot, 173
- expressive renditions, 65
- expressiveness, 42, 45
- external
 - choice, 113
 - documents, 187
- external documents, 160
- extracting information, 162
- extrusion, 81
- eye
 - anatomic characteristics, 9
 - geometric model of, 59

-
- faces, 140
 - facets, 19
 - FERRET, 172
 - filtered technique, 42
 - financial application scenario, 173
 - finishing shift, 97
 - Firefly, 103
 - fisheye, 47
 - fisheye text combination, 47
 - flat graphs, 43
 - flexible frameworks, 149
 - flow
 - of function evaluation, 21
 - of patterns on a surface, 53
 - representation, 53
 - focal point, *see* focus point
 - focus and context, 48
 - focus point, 42
 - fold plane, 56
 - foot model, 179
 - formulas
 - in a specification, 125
 - framework formalisation, 33
 - FraQL, 152
 - free-form
 - function, 58
 - function evaluation, 138
 - vertex manipulation, 138
 - fusion, 90
 - engine, 152
 - operators, 151
 - process, 151, 152
 - layout of, 173
 - gant charts, 19
 - geographic information systems, 173
 - geometric
 - description, 34
 - input topologies, 43
 - geometry
 - data nodes, 142
 - point-based, 80
 - getCenter, 140
 - GIS, 173
 - global
 - effect, 28
 - grade of variation, 43
 - object information, 30
 - Gooch shading, 68, 71
 - Google, 158
 - grade of variation, 43
 - GrADS, 172
 - granularity
 - level, 134
 - refinement, 103
 - graph
 - acyclic directed, 101
 - constrained propagation, 100
 - duration-based, 101
 - exact time stamps, 99
 - grid services, 149
 - group function, 117
 - group of related objects, 72
 - Grouper, 159
 - groups
 - merge similar objects, 62
 - helper function, 135
 - heterogeneous data sources, 150
 - hidden line removal, 141
 - hierarchic
 - composition, 109
 - temporal intervals, 108
 - time lines, 19
 - hierarchies, 43
 - hierarchy model
 - of concurrency layers, 121
 - of dynamic presentation variables, 25
 - of interval layers, 119
 - hierarchy of bounding spheres, 80
 - high-dimensional data, 62
 - HLA, 148
 - human time-organisation system, 16

- hybrid
 - parallel composition, 113
 - presentation, 65
 - rendering styles, 141
- hyper-media documents, 94
- hyperbolic layouts, 47
- Hyperstories, 104
- hypertext illustration, 47
- illumination model, 70
- illusion, 29
- illustration
 - effectiveness of, 32
 - goal, 30, 32
 - model, 126
 - data-driven, 3, 34
 - geometry-based, 3, 34, 94
 - result quality, 33
 - scenario, 34
 - subject, 34
 - target function, 32, 126
 - complete relations, 39
 - evaluation and limitation, 39
 - fully defined, 39
 - limit flexibility, 39
 - restrictions, 39
 - sound relations, 39
 - system properties, 36
 - techniques
 - classic and static, 40
 - classification systems, 40
- illustration domain
 - properties of, 32
- implementation, 133
 - basis, 133
 - overhead, 44
- implicit technique, 42
- importance value, 158
- impulse response function, 21
- inequality constraints, *see* temporal→
 - inequalities
- information
 - dimensionality, 18
 - gathering operators, 153
 - objects, 33
 - overload, 151
 - set, 33
 - space, 33
 - storage areas, 94
 - structure, 33
 - system-external, 89
- information density challenge, 73
- information fusion, 169
 - context, 150
 - testbed, 152
- information mural
 - motion-enhanced, 60
 - use of, 173
- information space
 - miniature version of, 60
- InFuse, 152
- input data image, 61
- input from channel, 113
- instance counting, 162
- intensity, 78, 79
- inter-object
 - relationships, 160
- inter-stimulus interval, 15
- interaction, 90
 - computer systems, 108
 - diagrams, 106
 - histories, 108
 - layer model, 177
 - machines, 108
 - pan and zoom, 48
 - techniques, 35
- interactive
 - definition of queries, 161
 - illustrations, 33
- intermediate results
 - gradually refined, 175
- internal
 - choice, 113
 - system information, 33

-
- internet media lab, 94
 - interoperability of services, 149
 - interplay
 - of nodes, 142
 - of toolkits, 135
 - interpolation
 - function, 119
 - scheme, 97
 - interpretation
 - human or mechanical, 34
 - interrupt, 113
 - interval
 - catalogue, 147
 - coherence, 108
 - composites, 109
 - diagrams, 108
 - flexibility, 93
 - handling functions, 129
 - length limitations, 127
 - limiting borders, 127
 - relations, 92
 - relationships, 90
 - sequence, 120
 - structure, 74
 - intervals
 - combination, 90
 - complete, 126
 - discretisation, 90
 - groups of, 93
 - management of, 90
 - merging and discretisation of, 129
 - merging of, 90
 - modelling of, 92
 - relational operations, 90
 - self-contained, 126
 - separation, 120
 - intra-object style modification, 107
 - invalidation of spatial relation, 73
 - irregular shape, 51
 - iterative fusion, 151
 - iteratively defined chewing gum, 82
 - itf, *see* illustration target function
 - Japanese Shinkansen trains, 19
 - Kartoo, 160, 163
 - keys to classification, 43
 - keyword collection, 158
 - knowledge
 - base, 90, 157
 - discovery, 94
 - landmarks, 73
 - landscape mapping, 176
 - lap function, 58
 - large data sets, 174
 - large volumes of data, 150
 - latency representation, 176
 - latitude, 171
 - layers, 25
 - layout
 - of a fusion process, 173
 - of documents, 149
 - of toolkits, 135
 - level of transparency, 46
 - LifeLines, 19
 - lightness, 79
 - limit function influence, 56
 - linguistic categories, 96
 - list of scheduled dynamics, 148
 - local
 - effect, 28
 - grade of variation, 43
 - time, 114
 - location parameters, 171
 - longitude, 171
 - machine learning, 151
 - macroscope, 48
 - Magdeburg, 171
 - magic eye view, 34
 - magic lenses, 48
 - magnification, 73
 - main interest scalar, 62
 - maintain
 - dynamics, 146

- intervals, 146
- maintenance life cycle, 177
- mapping
 - onto textures, 23
- material information, 134
- materialise individual objects, 134
- mean
 - temperature threshold, 173
 - values, 171
- measuring function, 52
- memorability, 73
- memory space, 94
- messages, 150
- messaging framework, 149
- meta data, 97, 152
- meta search engine, 160
- meta-data
 - representations, 173
- meta-information
 - representation of, 152
- metamorphosis function, 80
- metaphor
 - book, 23
 - dynamic, 22
 - river, *see* ThemeRiver
- method groups, 111
- methods
 - invoked on object instances, 106
 - line-oriented, 66
 - self-referential, 107
- middleware, 148
 - component-based, 152
- midpoint rotation, 51
- minimum function, 63
- mixed realities, 94
- model
 - automatic checkers, 96
 - exploration, 180
 - granularity, 124
 - graph-based, 98
 - organisational structure of, 59
 - task-oriented, 19
- model annotations
 - manually prepared, 157
 - predefined, 36
- model presentations, 177
- modelled world, 34
- models
 - object-oriented, 106
 - petri nets, 105
- modern communication networks, 150
- morphing, 28
- motion
 - abstract patterns, 13
 - apparent, *see* movement→apparent
 - autokinetic, 11
 - by structural change, 53
 - by structural change, 138
 - constraint factor, 63
 - explicit techniques, 29
 - function, 138
 - mapping, 52, 62
 - oscillation, 136
 - oscillations, 29
 - recognition, 9
 - rotations, 29
 - stroboscopic, 10
 - techniques, 48
 - threshold function, 173
 - toolkit, 135
 - trajectory, 10, 76
 - translations, 29
 - value and risk, 13
- motion-enhanced
 - information mural, 60
 - scatterplot, 173
- movement
 - apparent, 10, 16, 29
 - eye-head, 10
 - image-retina, 9
 - perception, 9
 - recognition, 9
- multi layered representation models, 34
- multimedia

- web search, 159
- multimedia presentation, 109
 - interactive, 94
- multiple
 - focus points, 42, 74
 - NPR styles, 142
 - objects, 43
 - translucent layers, 48
- multiset layer, 122
- mural
 - motion function, 64
 - parameterisation, 173
- navigating fisheye, 73
- nearby points, 42
- necer cube, 16
- negative folding effect, 57
- neighbouring vertices, 56
- nested graphs, 43
- neural networks, 32
- nodekits, 136
- non-photorealistic rendering, 65
 - coherence of, 95
- normal blending, 68
- NPR, 65
 - line rendering, 141
- numerical simulation data, 24
- object
 - attributes, 106
 - deformations, 75
 - flexibility, 107
 - hierarchies, 93
- object-document relationships, 160
- object-oriented
 - design principles, 106
 - model based on time lines, 111
- occurrence counting, 162
- ocean waves, 53
- ODA, *see* standard document architecture
- online aggregation, 175
- open interface description, 149
- Open Inventor, 135
- OpenGL, 135
- OpenNPAR, 135, 139
- operating environment, 93
- operations, 35
 - in a specification, 125
- OR-combined, 160
- ordering of scene elements, 35
- organisation of information objects, 53
- orientation information, 93
- oscillating motion, 136
- oscillation, 49
 - frequency, 137
 - function, 52, 136
- overlap value, 174
- overlapping object locations, 174
- page rank, 163
- parallel composition, 113
- parallel-first, 111
- parallel-last, 110
- parallelism, 94
- parameter list, 138
- parameterisation
 - documents, 146
 - function, 121
 - of stroke styles, 141
- particle paths, 24
- pass, 148
- path expressions, 110
- patient data, *see* time line browser
- pattern
 - landscape, 22
 - motion, 25
 - periodic, 21
- pen on paper metaphor, 71
- PendelKit, 137
- percept, 34
- perception
 - figure-ground, 15
 - of objects, 15

- perceptive influence, 28
- period of repetition, 95
- peripheral vision, 14
- permutations of object faces, 54
- Perspective Wall, 47
- PERT network, 101
- petri nets, 105
 - evolving concurrent object, 106
- physical constraints, 93
- pliable surfaces, 47
- point-based geometry, 80
- port types, 150
- ports, 150
- post-evaluation layer, 123
- practical use of a mural, 64
- pre-evaluation layer, 123
- precipitation, 171
 - threshold, 173
- preemptive visual clues, 119
- presentation
 - and processing, 152
 - aspect of, 89
 - context
 - medical visualisations, 43
 - definite, 120
 - example, 120
 - flexible, 120
 - framework, 33
 - model
 - state of, 90
 - of multiple search results, 168
 - of technical objects, 177
 - ordering, 109
 - performance, 56
 - periphery, 15
 - scheduling, 90
 - space, 40
 - techniques
 - overview, 40
 - temporal border of, 127
 - unit, 89
- preserve scene-related context, 45
- primer, 149
- primitive functions, 138
- priorities, 42
- priority
 - algorithm, 41
 - designation, 42
- programming interface, 158
- propagation, 93
- proxy design pattern, 147
- pseudo dates, 99
- PulseKit, 137
- QSplat, 80
- queries
 - sent to a search engine, 157
- query
 - interface, 158
 - processor, 152
 - refinement, 177
- querying
 - individual objects, 160
 - multiple objects, 160
- railroad schedules, *see* Japanese Shinkansen trains
- rainfall, 171
- range factor, 82
- ranking information, 163
- re-scheduling of dynamics, 123
- reachability graph, 95
- reaction times, 9
- reactive systems, 105, 114
- reactivity, 93
- readings of simple sentences, 96
- real world data, 33
- recursive program, 113
- reference intervals, 92
- regional
 - grade of variation, 43
- regret, 148
- regular shape, 51
- relabelling, 113

-
- related web pages, 162
 - relation
 - some-together, 97
 - relational filters, 35
 - relations, 33, 35
 - any order, 97
 - between objects, 134
 - complete, 93
 - forwarding, 109
 - in a specification, 125
 - inter-object, 158
 - transitive, 93
 - relative positions, 93
 - relevance, 164
 - repeating
 - attributes, 54
 - motion patterns, 49
 - repetition, 111
 - representation image, 61
 - represented world, 32, 34, 126
 - restricting dynamic character, 123
 - result
 - list, 138
 - page URL, 162
 - viewer, 161
 - ResultSet, 162
 - retina, 9
 - retrieval, 150
 - rewriting logic, 106
 - rigid transformation, 82
 - rings, 141
 - rotation, 50
 - around arbitrary point, 51
 - centre, 29
 - sampld output, 93
 - SbWingedEdge, 141
 - scale, 77
 - factor, 164
 - scatterplot, 172
 - motion function, 174
 - scene coherence
 - limit influence on, 84
 - motion, 50
 - scene description, 150
 - scene graph, 35, 133
 - uniform access, 136
 - wrapper, 141
 - scene model, 34
 - scene modifications, 35
 - schedule, 148
 - scope, 34
 - screws, 180
 - search
 - engine, 90, 157
 - for 3D models, 158
 - illustration system, 161
 - mediator, 162
 - result
 - evaluation, 162
 - illustration, 163
 - mapping onto rendering parameterisation, 164
 - server, 161
 - term, 162
 - timeout, 165
 - see-through effect, 46, 69–70, 145, 167
 - selection nodes, 142
 - selection of interest items, 164
 - selective, 111
 - self-organising maps, 34
 - semantic
 - background, 35
 - web, 94
 - semantically linked object clusters, 160
 - Semi-Intervals, 93
 - sensors, 93
 - separator node, 137
 - sequence, 43
 - sequential, 111
 - composition, 113
 - set of
 - catalogues, 146
 - control values, 144

- key positions, 137
- possible occurrences, 20
- shading artifacts, 70
- shadow volume, 44
- shaft, 180
- shape analysis, 159
- shape vis, 34
- shapes, 140
- shield, 180
- ShuttleKit, 136
- signals, 93
- signature, 125
- silhouette rendering, 79
- silhouettes, 66
- similarity queries, 159
- simultaneous dynamics
 - limit, 122
- single
 - interval, 119
 - NPR style, 141
 - objects, 43
- skip, 113
- smooth transition, 72
- snapshot representation, 114
- SOAP, 149
- SoCalculator, 138
- SoGenerateWingedEdge, 141
- solar radiation, 171
- SOPOs, *see* set of possible occurrences
- sorted blending, 68
- sorts
 - in a specification, 125
- SoSelectWingedEdge, 143
- SoStoreWingedEdge, 143
- SoWingedEdgeElement, 142
- SoWingedEdgeListElement, 143
- spatial
 - challenge, 73
 - correlation, 73
 - frequency, 14
 - gaps, 74
 - knowledge, 93
 - structure, 50
- spatio-temporal block, 22
- specification, 124
- speed discrimination, 14
- speedlines, 75
- spiral, 20
- splatting, 172
- splitting, 113
- spot noise, 23
- standard document architecture, 109
- starting shift, 97
- state
 - diagrams, 105
 - invalidation, 91
 - updating, 91
 - variables, 96
- states
 - in petri nets, 105
- steady changes of shape, 72
- steering search for patterns, 63
- stencil buffer, 69
- stimuli, 9
 - local, 30
 - recognisable, 30
 - unrecognisable, 30
- stimulus
 - contrast, 14
 - parameters, 15
 - ranges, 16
- stored structural knowledge, 30
- streamlines, 24, 172
- structural change
 - by folding, 56
- structural changes, 53
- styles
 - transition between, 143
- stylisation pipeline, 141
- stylised silhouette rendering, 67
- subjective text, 96
- suck in light, 70
- supplied priorities, 42
- surface shading techniques, 66

-
- surfals, 80
 - synchronisation
 - constraints, 110
 - intra-object, 103
 - methods, 111
 - window, 15
 - synchronised parallel composition, 113
 - system
 - expected behaviour of, 32
 - interrupt-driven, 93
 - properties, 36
 - response
 - automatic evaluation of, 33
 - restrictions, 37
 - semantics, 34
 - time-driven, 93
 - tabbed widgets, 164
 - Tardis, 22
 - target function, 32
 - evaluation and limitation, 39
 - fully defined, 39
 - illustration, 32
 - iteratively defined, 32
 - restrictions, 39
 - system properties, 36
 - taxonomy, 2
 - TCSP, 113
 - temperature, 171
 - span, 173
 - temporal
 - actions, 89, 108
 - actions of a user, 105
 - activities, 111
 - behaviour, 18
 - behaviour distribution, 107
 - change, 18
 - coherence, 95
 - complex models, 93
 - consistency, 95
 - constraints, 89, 90
 - control events, 93
 - control function, 124
 - databases, 114
 - dependencies, 90, 93
 - distance, 10
 - entity, 89
 - equalities, 103
 - evaluation entity, 94
 - event encapsulation, 106
 - external source of information, 90
 - frame, 10, 95
 - frequency of evaluation, 93
 - fusion characteristics, 153
 - granularities, 127
 - granularity, 109
 - graph, 127, 170
 - events, 127
 - intervals, 127, 128
 - reference scenario, 128
 - incoherence, 95
 - inconsistency, 94
 - inequalities, 103
 - languages, 112
 - logic, 112
 - model, 35
 - classification of, 96
 - graphical representation, 127
 - interaction and determinism, 93
 - modelling, 96
 - state-based approach, 112
 - networks, 124
 - nondeterministic behaviour, 94
 - orders, 90
 - parameterisation values, 144
 - presentation model, 116
 - preservation of consistency, 90
 - pushing behaviour, 93
 - reference scenario, 97
 - relations, 90, 92, 100
 - requirements, 89
 - event-based, 91
 - interval-based, 92
 - structural, 94

- restricted, 89
- server, 145
- structure, 107
- synchronisation points, 107
- zoom, 19
- temporal data flow, 172
- temporal-order threshold, 15
- temporally influenced geometry presentation, 145
- terminating user event, 169
- termination, 113
- text
 - legibility of, 28
 - transformations, 28
- Text Illustrator, 157
- text-based queries, 159
- textual result list, 164
- ThemeRiver, 20
- threshold
 - dimensions, 63
 - values, 61
- threshold dimensions, 173
- Tigeop, 145
 - documents, 149
 - service, 149
- time
 - absolute, 93
 - annotations, 97
 - as intervals, 92
 - axis, 19, 96, 107
 - granularities, 112
 - human perception of, 109
 - relative specification, 93
 - representation, 18
 - stamps, 101
 - units, 113
- time line, 18, 96
 - browser, 19
 - interface, 94
- time-stamped traces, 108
- timed
 - delay, 113
 - objects, 109
 - prefix, 113
- timeout, 113
- timing requirements, 93
- toolkits, 133
 - layout of, 135
- trace line, 75
- transaction
 - commit, 123
 - start, 123
- transfer function description, 138
- transformation
 - dimension, 41
 - functions, 125
 - transformer, 128
 - information, 134
- transformer, 128
- transition
 - between styles, 143
 - engine, 144
 - in petri nets, 105
- transitivity, 127
- translation, 50, 77
- transparency, 45, 68, 144
- tree structure, 109
- tree view, 164
- tree-maps, 47
- triangle fights, 70
- true concurrency, 106
- two pass rendering, 72
- UML, 106
- URL, 158
 - matching, 162
- use of presentation variables, 35
- user
 - assembly, 180
 - information needs, 32
 - interaction, 91, 94
 - interface
 - non-visual, 108
 - visual, 108

- model, 32, 33
 - parameters, 34
 - perception, 34
- validation, 95
- variable
 - rendering styles, 26
 - set, 125
 - text, 28
- variables
 - in a specification, 125
- vertex of interest, 56
- vertices, 141
- viewing directions
 - degree of freedom, 61
- virtual spiral, 164
- visual
 - attention
 - attracting and directing, 26
 - data mining, 33
 - metaphors, 13
- visualisation toolkit, 153
- visually identify, 175
- VTK, 153

- W3C, 149
- warp function, 82
- waterfall, 12
- waves
 - breaking of, 53
- weather forecasts, 171
- web search evaluation, 159
- web service, 148
 - description language, 149
 - environment, 149
 - interoperability, 162
- widgets
 - tabbed, 164
- wind-run, 171
- WingedEdge data structure, 140
- work load, 178
- workbench for information fusion, 150

- world-wide distribution, 150
- WSDL, 149

- XML Schema, 150

- yellow-black combinations, 45

- zoom
 - boundaries, 74
 - loop, 83
 - path
 - of an object, 80
- zooming, 28