Lean Modeling - The intelligent use of geometrical abstraction in 3D animations

Andreas Butz[1] and **Antonio Krüger**[2]

Abstract. The purpose of this paper is to discuss the usefulness and applicability of automatically generated geometrical abstractions in the automatic generation of 3D animations under time-critical conditions.

The paper describes two components currently under development, one for the computation of geometrical abstractions, the other for the automatic generation of 3D animations.

It shows how they can be combined to yield an intelligent level of detail control which improves the expressivity of visual communication and saves computing capacity at the same time (adaption to the cognitive and computational resources).

1 The Framework

The increasing complexity of user interfaces and presentation systems, the graphical capabilities of available computer hardware and the tendency towards the use of multi-modal computer-human interaction raised the demand for automatic generation not only of natural language, but also graphics and animation.

The two systems we describe in this paper are ongoing work in the $\mathcal{PPP}$[3] Project at the German Research Center for Artificial Intelligence. $\mathcal{PPP}$ is an intelligent multimedia presentation system [2], that presents information about the usage and function of technical devices and produces personalized interactive operating instructions that are adapted to the person interacting with the system.

$\mathcal{PPP}$ achieves the adaptivity of its presentations by generating their whole structure as well as all their parts from scratch. Instead of storing a limited set of preproduced text, graphics and animations in a large database, it generates them on the fly, using techniques of NL generation and automated graphics generation.

These newly generated presentation parts can reflect all aspects of the presentation situation. They allow cross-references between each other (e.g. texts referring to pictures) and are tailored to the overall document structure and their function in this structure, which is a clear advantage over the currently used "patchwork" technique with preproduced presentation parts.

On the other hand the creation from scratch involves much more computational effort than a pure combination of given presentation components.

2 Graphical Abstraction in Computer Graphics

During the last twenty years significant progress has been made in the field of photorealistic computer graphics. The common goal in this field is to generate graphics by means of a computer in a most realistic fashion. The results achieved are impressive and the gap between photographic material and computer graphics has been reduced significantly. These realistic graphics are used in many application areas such as entertainment and advertising.

However the increasing amount of information in graphics is leading to a twofold problem because on the one hand the generation of computer graphics is becoming computationally more intensive and on the other hand the resulting graphics are often overloaded with information. This causes a distraction of the viewer's attention from relevant aspects of the graphics. A good example where this problem has been detected is the field of documentations and technical manuals. Authors of device descriptions prefer to rely on graphical means other than photographs or photorealistic computer images to create presentations of objects and their use.

In order to reduce the waste of *computational* resources (i.e. computer memory and cpu time) and of *cognitive* resources (the capability of the viewer to process the presented information in an appropriate way) it is necessary to filter and customize the graphical information.[4] For these simplifications we chose the term *stylistic abstractions*.[5] They are not technically caused, but made intentionally in order to follow a certain presentation goal. Examples of such goals are: "Draw the viewers attention to a certain object property" or "Show an object in its spatial context without defocusing it".

2.1 Automatic Generation of Stylistic Abstractions

In order to satisfy presentation goals, human designers follow one or more design directives to improve the communicative expressiveness of an illustration by means of stylistic abstraction. They try to

- avoid overloading the graphics with information and
- focus the viewer's attention to the relevant parts of the graphics.

Further directives are the wishes to

- keep a presentation as simple as possible and
- increase the uniformity within a series of pictures.

Which of these directives are used depends on a variety of parameters such as purpose of the graphics, viewer profile and presentation context (including the mood of the designer). To enable a graphics generation component to generate automatically stylistic abstractions at least the following two things must be done:

1. abstraction operators must be developed and
2. the conditions determining a certain technique must be specified.

[1] Universität des Saarlandes, Saarbrücken
[2] Universität des Saarlandes, Saarbrücken
[3] Personalized Plan-Based Presenter

[4] For more information on resource adaptivity refer to [13].
[5] In contrast to the notion of a *technical abstraction*, which is caused by technical constraints (e.g. screen resolution or number of available colors). For more details refer to [9].

ECAI 96. *12th European Conference on Artificial Intelligence*
Edited by W. Wahlster
Published in 1996 by John Wiley & Sons, Ltd.

2.2 Methods for Stylistic Abstractions

The graphics generation process can be seen as a *rendering pipeline*[5]. Fed with an object-model[6] this pipeline generates in several steps a two-dimensional object depiction. This includes for example the choice of convenient camera and light source positions or the definition of a projection mechanism. Methods for stylistic abstraction can be applied at any step of this pipeline. So, for example, an object's depiction generated with the Gouraud shading method contains less information than the same object's depiction obtained with the more realistic Phong shading method or even raytracing or texture mapping. Other approaches do not at all try to produce photorealistic results. With the help of a *sketch-renderer* [12] it is possible to produce results that are very similar to hand-drawn graphics.

Most of these abstraction methods filter the information towards the end of the rendering pipeline. In order to gain more flexibility in the results we decided to abstract at the beginning of the pipeline, e.g. the methods operate directly on the object's representation. This means instead of using the highest detailed version of an object's constellation, a geometrically simplified version is first computed and then used as an image source.

Inspired by Feiner's work on geometric approximations on domain objects [4] we have developed a binary operator which reduces the complexity of an object's model by *merging* primitive modeling components. Figure 1 shows an example for the abstraction of the modeling primitives cube and cylinder. Two separate objects are merged into a new object that occupies the same space. It can be considered as a graphical abstraction since the amount of object parts and the representational effort (e.g. the number of polygons) has been reduced. Object parts are closed under the merging operator, so that it can be used more than once and also on its own results. So it is possible to increase the level of abstraction step by step.

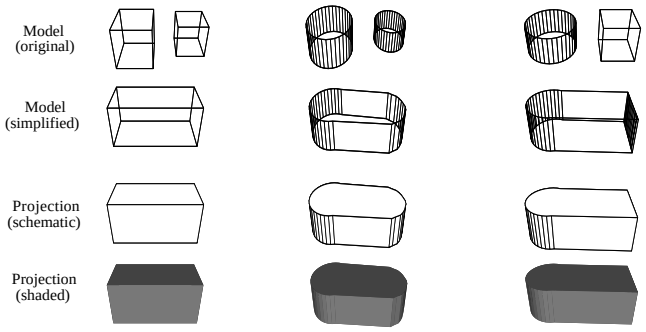

Figure 1. The *merging* operator for 3D-models

In this situation the amount of information is much higher at the beginning of the pipeline than at the end. By a conscious selection of the filtered aspects the results can be tailored much better to a specific presentation goal. Another advantage is that a reduction of information in earlier steps generally reduces computational effort at later stages.

2.3 Applying Stylistic Abstractions

The abstraction method of merging object primitives introduced above has been implemented in the prototype system PROXIMA[8].

[6] in our case: polygon-based wire frames with surface-descriptions

The system is given a certain presentation goal that must be fulfilled in the resulting abstracted graphics. For this purpose PROXIMA first derives a *visibility constraint* from the presentation goal for each object part. In a second step these constraints are used to decide which of the object's parts can be simplified by using the merging operator.

Looking at an example we can see how PROXIMA computes abstractions. Figure 2(a) shows the depiction of the object constellation of a modem's circuit board. Consider, for example, the presentation goal as:

Focus the viewer's attention to the *switch* and to the *LED*

From this goal PROXIMA first derives that since the switch and LED are foreground objects, they should remain *identifiable* in the resulting graphics. Other objects are constrained with the help of a database containing information about significant attributes of the circuit board and with the use of default rules. These rules control the over-all abstraction level of the graphics. They are used for example to ensure a proper classifaction of the abstracted background objects (in this case: the circuit board). Knowledge about significant attributes of the circuit board is important in order to decide which objects contribute to which extent to this task (for a more detailed description of PROXIMA's behaviour see [9, 8]). The result of the abstraction process

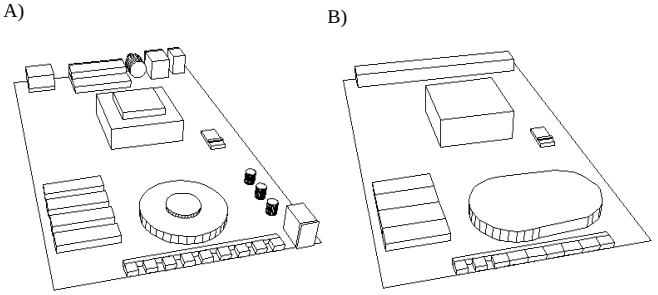

Figure 2. Example of PROXIMA's abstraction capability.

in figure 2(b) illustrates this issue. The foreground objects (*switch* and *LED*) remained unchanged. In contrast to that some background objects vanished completely. Consider for example the large feature in the front and in the right of the circuit board . This part is not a significant one and must not necessarily be displayed in order to fulfill the presentation goal. Other background objects and parts are more important to provide landmark information , like the sockets on the rear end of the circuit board. The higher level abstractions of these objects are obtained by applying the merging operator more than once.

PROXIMA has been put to use successfully in a variety of scenarios. It emerged originally from the WIP project [1], an intelligent multimodal presentation system. Other application areas are the semi-automated illustration design [10] and the generation of hypergraphics [9, 2]. The following sections will show how PROXIMA can be used to improve the results of an animation generation system.

3 Animation Generation

Some situations or processes such as the movements of objects and the spatial constellation of a scene can be shown much more clearly in the form of a 3D animation than in a text or static graphic. The

second system we describe, which is named CATHI,[7] generates 3D animation scripts from given model worlds and communicative goals based on a context-sensitive 'animation grammar' and a generation context database.

It then controls the rendering of these scripts and shows the resulting animations with the smallest possible delay on the screen. Figure 3 shows an overview of the architecture of CATHI at the current state of implementation.

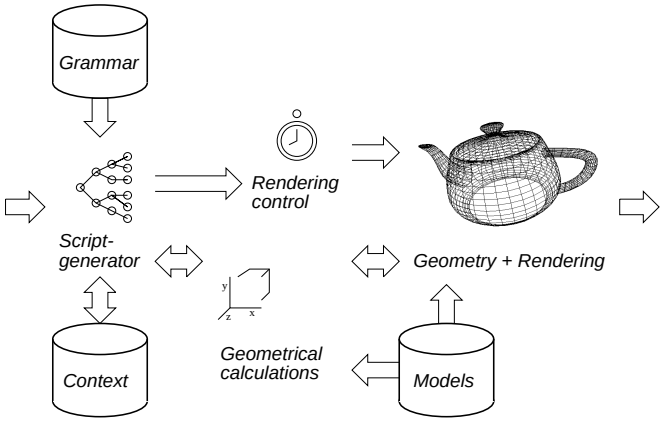

Figure 3. Architecture of the CATHI system

3.1 Script Generation

The CATHI system uses a given 3D model world for which it calculates the necessary camera positions and light settings, and it has a set of grammar-like rules that describe animation sequences in a scene-independent way. The input of the script generator are high-level descriptions of the animation to be generated. For example

Localize object A in its surroundings

Show movement B of object A

The localization of a small object in the scene, for example, implies setting the camera to a fitting position from which the object can be seen in its surroundings, followed by a shot zooming in on the object. Zooming in on an object in turn implies keeping the camera position and varying the camera's viewangle at the same time. Likewise the grammar rules recursively specify the subsequences of the animation scripts.

During the generation process the expansion of the rules is guided by a generation context that describes the current state of the scene and external parameters such as the graphical capabilities of the output device (renderer or visualization tool). The rules themselves specify recursively the subsequences of a certain animation sequence as either parallel or successive in time. They also contain conditional descriptions that select between several possible sets of subsequences depending on the current generation context.

The generation process works incrementally and allows rendering of the early parts of a script and displaying of the corresponding pieces of animation while further parts of the script are still being generated. This strategy is also applied in newer Natural Language generation systems [7] that face the need to generate output under real-time conditions and it represents the most important difference between CATHI and other existing animation generation approaches [6, 3].

Incremental generation presumes the fact that once a subsequence of the animation script is generated, it stays unchanged by further generation and can be forwarded immediately to the renderer or visualization tool, (Otherwise we would run into major difficulties). This can be achieved by the concept of directed and context-driven recursive expansion of the animation grammar rules described above. This strategy involves no backtracking through trial and error.

During this process, the generator always “knows” which objects are important for the current scene. It obtains this information from the initially given abstract presentation goal as well as from the current state of the scene, the animation grammar and the structural description of the model world.

At the moment this information is used to control the camera parameters such as eyepoint, aimpoint, viewangle, focal distance and depth of field. Objects that are important have to be inside the camera's field of view and they have to be in focus (in photographic terms: inside the depth of field). Unimportant objects can be out of focus or even outside the frame if they aren't needed to create the visual context for the main objects.

3.2 Output Delay

Since the generation process and the output of the rendered animation overlap with the smallest possible delay, the time available for the script generation is generally limited upwards by the duration of the animation. The delay with which the animation can be rendered is limited downwards by the increments of the generator, i.e. the size of the pieces of script it can forward to the renderer.

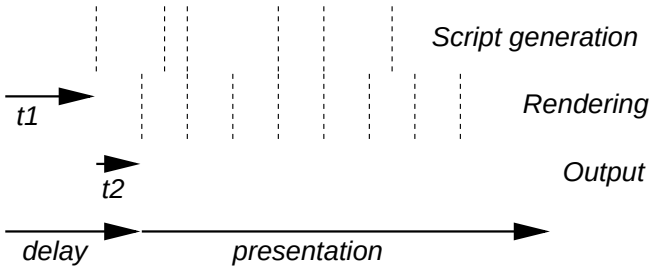

Figure 4. Sources of delay

Figure 4 shows the timing of these processes in a real-time generation situation. Increments in every process are marked by dashed lines. The delay from the start of the generation process until the start of the renderer is dictated by the first increment the generator produces, i.e. the time $t1$ until it forwards its first piece of animation script to the rendering control component.

The second delay is caused by the renderer and is typically the duration of the rendering process for the first frame plus its display on the screen. This delay is neglectible here but it plays an important role for the fluency of the animation because it dictates the frame rate (increments of the renderer).

The overall delay from the start of script generation until the start of the presentation is the sum of the two which means that it is mainly set by the increment size of the script generator. It is the task

[7] Computer Animation Tool for Help and Information systems

of the grammar-writer to specify an animation grammar that allows the smallest possible increments.

In fact this is a somewhat simplified description that leaves out several aspects (the first increment isn't always the largest and what if generation time surpasses presentation time?) but this is beyond the scope of this paper. The fact to remember is that through the formulation of the animation grammar we can only minimize the delay from the start of script generation until the output of the first frame.

3.3 Output Speed

Another factor that cannot be influenced by either the grammar or generation algorithm is the rendering speed. The time needed to render a single frame depends on two other factors: image quality (i.e. complexity of the shading model) and complexity of the scene (number of polygons, textures, lights). Increasing one of these three factors implies the reduction of at least one of the two others given a limited computing capacity (see figure 5). If we assume that the

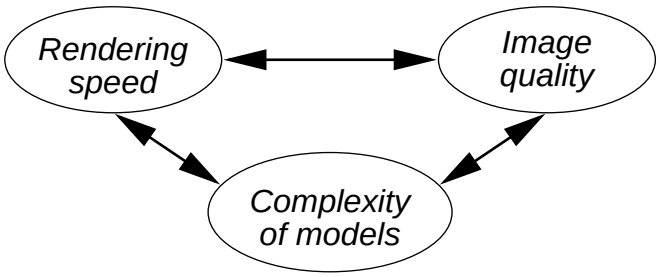

Figure 5. Depending factors

complexity of the scene is fixed, and that we need a certain animation speed to achieve fluent motions, our only choice is to decrease image quality (i.e. go down from Phong or Gouraud shading to flat shading or even wire frames). The three factors are dependent on each other and the overall quality is limited by the graphical and computing capabilities of the hardware we use.

4 Achieving Fluent Output

In the current state of implementation CATHI selects the shading model based on the available graphics hardware so that it can show fluent animations at the cost of image quality. This is the same concept that is applied in many VRML viewers where the scene is shown as wire frames during camera motions and is rendered with shading and textures as soon as the camera stops. VRML however contains yet another concept to increase rendering speed.

4.1 Conventional LOD Selection

In a VRML world objects can be described at different levels of detail (LOD) and the representation of a certain object in the scene is chosen from these LODs depending on the object's distance to the camera. This choice presumes no knowledge at all about the scene, since it can be made simply based on geometrical computations.

Currently the necessary abstractions for these different LODs are mostly modeled by humans or in rare cases the result of interpolation algorithms [11]. Information about, for example, relevant parts of an object are not processed at all. This technique increases the amount of data necessary to describe a VRML world since several descriptions for the same object are needed, but it speeds up the viewing process afterwards.

Besides the camera distance there are several other parameters that could control the choice of LODs in an animation. We could, for example, consider the motion speed of moving objects. Objects that are moving fast across the frame don't need as much detail as those that are fixed.

Another parameter is the focal distance together with the depth of field (DOF). Objects that are out of focus don't need much detail, since they're shown more or less fuzzy anyway. But in contrast to DOF rendering computations that are quite expensive, geometrical abstraction even reduces rendering time. If we take the model of the *rendering pipeline* mentioned earlier, DOF rendering is done at the very end of the pipeline while geometrical abstractions are applied quite at the beginning and reduce the amount of data in further computations.

Finally, we could abstract objects depending on their position in the field of view. As a rule of thumb, objects in the center of the image would then be considered more important and would be shown fully detailed while objects towards the sides are shown at higher abstraction levels. All these parameters can be computed from the bare 3D representation of the scene and need no further information.

4.2 Intelligent LOD Selection

In contrast to the methods above, a script generator like the one in CATHI already has the information that would be needed to apply the LOD feature in a more intelligent way. As described earlier, the script generator uses information about the relevance of objects in the scene for the control of camera settings. This knowledge could be used as well to control the LOD of a geometric object in the scene.

Objects that are important in the animation, either because they are known to be 'main actors' or because their visibility and perceptibility is needed to create the visual context, could be shown in full detail, while others can be shown at higher abstraction levels using only a fraction of their original geometrical complexity.

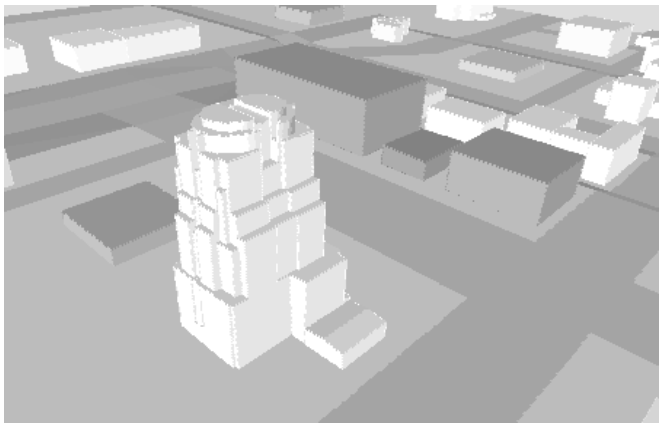

Figure 6. Partially abstracted cityscape

Figure 6 shows a scene with houses and streets in which only one house is shown detailed while the others are shown at higher abstraction levels. This clearly directs the visual focus towards the house in question.

The necessary geometrical abstractions could be modeled or computed in advance and stored in the model world as alternative geometrical representations of one object, or they can be generated automatically on demand, and we can specify the degree of abstraction of each object using the techniques described earlier.

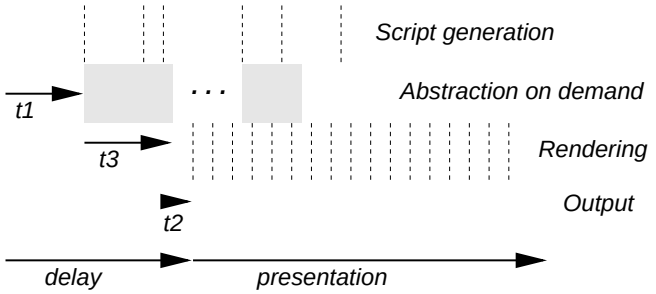

Figure 7. One more source of delay

While preproduced abstractions would cause no further delay for the output of the animation, they restrict us to the cases and situations we have thought of in advance or (if we want to store all possible abstractions) they would be very resource-intensive. In addition, the philosophy of the $\mathcal{PPP}$ system (generation from scratch for maximum flexibility) implies a generation of the necessary abstractions on demand during the animation generation. The computation of these abstractions introduces another source of delay to the animation output but allows the maximum flexibility in the use of different abstraction levels. Figure 7 shows a modified timing diagram for this situation.

Whereas the overall delay from the start of script generation until the start of the animation output has grown, the rendering speed itself is increased significantly. This allows a higher frame rate in the rendering process and more fluent animation in the final output. As in the case of VRML, where a higher amount of data causes longer loading times, this delay can easily be accepted for the effect of fluent animations.

This concept of conscious simplification gives us the opportunity to use LODs in a much more intelligent way than VRML does. By stylistic abstraction of irrelevant parts of a scene we can significantly reduce its complexity. This intelligent reduction (for which we chose the term ' Lean Modeling') results in increased rendering speed which permits either higher frame rates or a better shading model to be used.

Another advantage of this concept is that the visual focus in the animation can be intensified by these abstractions (see figure 6). Geometrical abstraction as such is another expressive tool to support visual communication, such as depth of field or illumination. This tool can already be considered in the script generation process and it can be used consciously in the formulation of the animation grammar.

The overall result is that by the conscious application of stylistic abstractions we gain not only rendering speed (through reduced complexity of the scene) but also clarity in the expression of communicative contents (through the use of abstraction as an expressive tool).

5 Conclusion

We have seen in this paper that it is possible to use automatically generated stylistic abstractions of 3D polygon models to reduce the complexity of 3D scenes. A main problem with the use of abstractions is the choice of the appropriate level of detail for the single objects in the scene.

In the automatic generation of 3D animations the information for the conscious use of stylistic abstractions is already present in the script generation process. The controlled simplification by stylistic abstraction can be used consciously to direct the visual focus.

Unlike 'dumb' animation systems (like VRML viewers), the use of abstracted models not only increases rendering speed but also yields an expressive tool that can be used to support the communicative contents of an animation. This is an important step towards resource-adaptive graphics generation taking into account not only computational but also cognitive resources.

ACKNOWLEDGEMENTS

The visualization software (geomview) that we use, was written at the National Science and Technology Research Center for Computation and Visualization of Geometric Structures at the University of Minnesota, USA

REFERENCES

[1] E. André, W. Finkler, W. Graf, T. Rist, A. Schauder, and W. Wahlster, `WIP: The Automatic Synthesis of Multimodal Presentations' , in *Intelligent Multimedia Interfaces*, ed., M.T. Maybury, 75–93, AAAI Press, (1993).

[2] E. Andre, W. Graf, and J. Heinsohn, `PPP - Personalized Plan-Based Presenter' , Technical report, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Saarbrücken, (1993).

[3] A. Butz, `BETTY: Ein System zur Planung und Generierung informativer Animationssequenzen' , Technical report, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Saarbrücken, (1995).

[4] S. Feiner, `Apex: An experiment in the automated creation of pictorial explanations' , *IEEE Computer Graphics and Applications*, **5(11)**, 117–123, (1985).

[5] *ISO/IEC IS 11072, Information processing systems - Computer Graphics - Computer Graphics Reference Model*, 1992.

[6] P. Karp and S. Feiner, `Automated Presentation Planning of Animation Using Task Decomposition with Heuristic Reasoning' , in *Graphics Interface '93*, pp. 17–21, (1993).

[7] A. Kilger and W. Finkler, `Incremental Generation for Real-Time Applications' , Technical report, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Saarbrücken, (1995).

[8] A. Krüger, `PROXIMA: Ein System zur Generierung graphischer Abstraktionen' , Technical report, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Saarbrücken, (1995).

[9] A. Krüger and T. Rist, `Since Less is often More: Methods for Stylistic Abstractions in 3D-Graphics ' , in *ACM Multimedia Workshop on Effective Abstractions in Multimedia Layout, Presentation, and Interaction*, ed., Kent Wittenburg, (1995).

[10] T. Rist, T. Krüger, G. Schneider, and D. Zimmermann, `AWI - A Workbench for Semi-Automated Illustration Design' , in *Advanced Visual Interfaces (Proceedings of AVI '94, Bari, Italy)*, 59–68, (1994).

[11] J. Rossignac and P. Borrel, `Multi-resolution 3d approximations for rendering complex scenes' , in *Geometric Modeling*, pp. 455–465. iVerlag, (1993).

[12] T. Strothotte, B. Preim, A. Raab, J. Schumann, and D. R. Forsey, `How to Render Frames and Influence People' , Technical report, Department of Simulation and Graphics, Magdeburg, (1995).

[13] *Sonderforschungsbereich 1542 – Forschungsantrag: Ressourcenadaptive kognitive Prozesse*, ed., H. Tack, Universität des Saarlandes, Saarbrücken, 1995.