Position Paper (draft February 1, 1996)

AD REM*: Accessing Distributed Resources via Executable Media

Gino van den Bergen Kees Huizing Wim Nuij Kees van Overveld
Remco Veltkamp Huub van de Wetering

Department of Mathematics and Computing Science
Eindhoven University of Technology
PO Box 513, 5600 MB Eindhoven, The Netherlands
Email: [gino—keesh—wsinwaan—wsinkvo—remco—wstahw]@win.tue.nl

Abstract

This paper describes the AD REM research and development initiative. The concept of 'executable media' is introduced as an object-oriented extension of conventional multi media. Based on executable media, an architecture is proposed to support access to distributed information resources.

1 Introduction

Over the last two or three years, the concept of Multi Media (MM) has aroused a tremendous growth of interest, both from the side of applications and from the side of research and development initiatives. The inherently multi-disciplinary nature of MM has drawn a host of researchers from various backgrounds (computer architecture, user interface design, the visual arts, computer graphics) to assess the added values of MM and to enhance the functionality of traditional information media by means of integration and multi-modal access. The AD REM research initiative, to be discussed below, aims at bringing together research results and proposing unifying paradigms for further development of MM applications. It derives from known principles and techniques that are used in present day MM-applications, and it introduces new extensions thereto. The integration of these principles and techniques into a working, demonstrable, and potentially commercialisable system is seminal to the AD REM project.

In order to describe the central ideas of AD REM, and to contrast these with current MM concepts, this document first contains a short list of definitions of MM-related concepts in section 2. Next, in section 3, some properties of current MM are discussed where an emphasis is put on the inherent limitations. Section 4 lists some extensions to the current MM concepts that might alleviate these shortcomings. To contrast the AD REM-view to MM with current MM, we introduce the term 'executable media', an extension to the traditional information exchange media based on object-orientedness and distributed resources. This leads the way to an introduction of the AD REM project in section 5. This section focusses on the organisation of AD REM and the relation between AD REM and other MM-related research initiatives. Finally, section 6 gives a somewhat more detailed account of the current state of AD REM and the new sub-projects that have been defined now.

*The Latin expression 'ad rem' translates to 'to the point'.

2 Present day MM – a summary of concepts

Present day MM-applications have resulted from some recent technological developments:

- high performance PC's with high resolution raster graphics displays have become available;
- the CD-ROM as information carrier has been introduced;
- the JPEG and MPEG standards for efficient image and video compression have become widely accepted;
- audio cards based on MIDI and other standards for the exchange of digital sound and music have become a standard device in most PC's.

Apart from these technological innovations, the development of MM in the commercial market can be explained by the following observations:

- it is relatively cheap to produce, reproduce, and distribute MM-documents (see below for the definitions of MM-related terminology);
- it requires relatively little specialised technological knowledge to create MM-documents;
- graphical and audible presentation of information is highly appreciated by a broad audience;
- interactive computer games already had gained an immense popularity; many MM-applications are (deliberately) reminiscent to computer games.

The MM phenomenon took shape in a relatively short time as a result of the turbulent interaction between market and technology. Therefore, no widely accepted definition of MM has emerged yet. Facing the risk of ignoring many relevant and promising recent developments in the periphery of the MM-field, in this paper we consider MM as a combination of the following concepts.

MM-platform : a computer with hardware for high-quality reproduction of image and sound together with

  - a CD-ROM player for so called local MM-applications, or
  - a network access for so called remote MM-applications (e.g. Internet access via Mosaic for World Wide Web applications).

MM-item : a text, image or sound fragment, a video/animation fragment or an interaction widget. An interaction widget is a graphical control device such as a push button or a slide controller, allowing an MM-user (see below) to navigate through an MM-document (see below).

MM-document : a collection of coherent MM-items in combination with an MM-dialog structure (see below) defined on top of them.

MM-player : the computer program running on an MM-platform to disclose the information in an MM-document to an MM-user. This takes place in a dialogue with the MM-player according to the dialogue structure encoded in the MM-document.

MM-dialogue structure : a set of rules defining the effect of interaction of the MM-user with each of the interaction widgets. Also the timing and synchronisation of the several MM-items within the MM-document are controlled from within the MM-dialogue structure.

Local MM-application : an MM-application that runs on a stand-alone MM-platform, where the MM-document (and optionally also the required MM-player) are loaded from the locally present CD-ROM player.

Remote MM-application : an MM-application whose MM-items are distributed and loaded from a network. At any time, only parts of the MM-document need be resident in the MM-player. Typically, the required MM-player is part of the local environment, but services can be provided by remote processes.

MM-user : a person who retrieves information from an MM-document via the MM-dialogue structure.

MM-author : a person who assembles an MM-document, making use of an MM-authoring tool (see below), which is intended to be played by a given MM-player. Assembling an MM-document comprises collecting and installing MM-items, and specifying the MM-dialogue structure.

MM-authoring tool : a computer program for assembling MM-documents.

Current MM-media : text, (still) images, video/animation, sound, digitally encoded music. These media are connected in a coherent manner by means of the dialogue as it executes under control of the MM-player.

3 Properties and limitations of current MM-applications

From the previous list of definitions, we can conclude some properties and limitations of current MM. Phrased in key words, these are the following.

Statical : an MM-document is encoded in a CD-ROM or on an FTP-server and it cannot change during an MM-dialogue. Only the order of access of the several MM-items of the MM-document may change under control of the MM-dialogue.

Uni-directional : except for the information of the MM-user to direct the MM-dialogue via the interaction widgets, all information is directed from the MM-document to the MM-user.

2-D : the images and video/animation fragments in most current MM-applications are 2-D. This means that the MM-user has no control over camera standpoints or illumination conditions.

Non-programable : during an MM-session, the MM-player is conceptually the only computer program running. The hard wired functionality of this MM-player therefore fully determines the functionality of the MM-application. Moreover, the instruction set of the MM-player is typically restricted to instructions for showing texts, images or video fragments and playing sound fragments or music scores. Therefore, the action repertoire of an arbitrary MM-application is limited to a sequence of these kinds of actions, possibly affected only by user commands intermediated via the interaction widgets.

Hard-wired data-retrieval structure : to disclose MM-items in an MM-document, the MM-author installs so called links in the MM-document, mostly coupled to interaction widgets. These links comprise (conceptually) a hypertext structure, and this is the only available structure to retrieve information from the MM-document.

4 Possible extensions of the MM-concept: executable media

As an extension of the current MM-concept we propose the introduction of executable media. The MM-concept, thus extended, will be denoted as EM to avoid confusion. Executable media are a means to alleviate the limitations listed in section 3. The crucial aspects are introduced below.

Dynamic : There exist two ways to escape the invariability of current MM-documents:

- by means of computation: in current MM, MM-items contain canned information in the form of files. By constrast, an EM-item can be seen as a process that may perform computations and thus may exhibit an autonomous behaviour, maybe interacting with the behaviours of other EM-items and/or the EM-user. Examples of the resulting enhanced flexibility are computer games, such as flight simulators, where the displayed images are not pre-recorded on CD-ROM, but are computed as a function of the (time-dependent) behaviour of the player. A standard type MM-player however, is not capable of doing arbitrary computations. Therefore, applications such as flight simulators are not reckoned to be standard MM-applications such as encyclopedia and more traditional adventure games. With the EM-player as introduced in section 5, however, arbitrary computations and hence arbitrary simulations can be programmed. Therefore we will use the flight simulator as our running example throughout this section.
- the second means to escape the static nature of current MM-documents is to offer an 'underwater' link to the global network. Via this link, the information in an EM-document can be updated on-the-fly. For a flight simulator, this may mean for example the following. Suppose we approach JFK Airport in New York. Then in order to give an increased touch of reality to the simulation, the current weather conditions above New York are obtained via an automatic query to a meteorologic database in that region. One step further is that the planes and flight manoeuvres of other players, anywhere on the global network, are brought into view as well. This option of real-time updates of EM-applications via the global network implies that an EM-document, once purchased, is actually equivalent to a life-long subscription on an ever-lasting information source. Moreover, since the information that is communicated to an EM-player consists of executable objects rather than traditional static files (see below), the behaviour of EM-applications also may be extended long after they have been first installed.

Multi-directional : Because an EM-player may execute arbitrary computations (simulations), controlled by input from the EM-user, and because a transparent link to the global network is part of the EM-concept, the information stream is truly bi-directional. This means that groupware applications (where several users can explore and modify the same application domain where they are real-time visually informed about each other's actions) are inherently supported by the EM-concept.

3-D : In the case of a flight simulator, it is crucial that the simulated world may be viewed from arbitrary spatial directions. It is not feasible to offer all possible views as pre-computed 2-D images in an MM-document. Instead, a 3-D model of the geometry of the simulated scenery is present as EM-items, and the required views are generated on the basis of this model. This is possible if the computer, running the flight simulator, is equipped with sufficiently efficient 3-D graphics hardware and software to compute a real-time conversion of a 3-D geometric model to a 2-D image. We assume that the proposed EM-player possesses these features (which are becoming quite common nowadays).

Programmable : In contrast to a conventional MM-player, the EM-player is freely programmable, changeable and extendable even during a running simulation. To this aim, an EM-document contains both standard MM-items and code fragments for the EM-player. The code is interpreted by the EM-player. It may contain instructions to do computations, instructions to redefine existing functionality of the player, and instruction to extend its functionality. These code fragments may be read during a running EM-session from a local EM-document or communicated over the global network link.

Effectively, there is no longer a distinction such as between an MM-document and an MM-player: an EM-document may contain extensions or modifications to a 'standard' EM-player in order to customise it for obtaining task-specific behaviour. Conversely, EM-items may be generated on the spot by dedicated methods, part of a customised EM-player, and unlike MM-items, they don't have to be resident a priori in the EM-doccument.

For example, consider a conventional MM-document on the classical Greek civilisation. An MM-item in this context could be a digitised photo of the Parthenon. The EM-equivalent is a 3-D geometric model of the Parthenon. This means that the EM-user may enjoy a full walk-through, zoom in on arbitrary details (as far as they occur in the 3-D model) and look at them from arbitrary directions. He may adjust the light conditions at wish. Next he may want to verify how the Parthenon would look with Dorian columns instead of Ionian columns. All necessary functions (e.g. to implement interactive walk-throughs to explore 3-D geometric scenes) could be coded as part of the geometric model, or they could be suitably installed extensions to the generic EM-player, to customise it (in this example) for the task of a 'virtual environment viewer'. In the case sufficient attributes are present in the 3-D geometric model the MM-user may want to establish such issues as the centroid of the temple or the distribution of the roof's weight over the columns. Also, if adequate dynamic simulation and collision handling capabilities are installed, he may experiment with the effect of pushing over one of the corner columns. This all can be realised by interpreting code fragments in EM-items from the EM-document. Finally, the exchange format of these geometric EM-items has to be sufficiently generic in order to import the Parthenon including its full behaviour in the flight simulator discussed above, or to let it serve in a training application for Urban Design and Planning.

Variable data-retrieval structure : Every EM application is embedded in a world wide network structure via a transparent global network link, with sources of information that may be valuable for a given application. Also, the information fragments that are communicated over the global network from one EM-player to another may posses both a static information contents and a dynamic, autonomous behaviour, provided by code fragments. Therefore it is an attractive option to consider sophisticated ways to perform remote data queries. In particular, one can imagine that queries trigger agents that search the global network in an autonomous way to retrieve desired information, and, using pattern matching methods or other (application dependent) techniques possibly derived from artificial intelligence, interrogate remote information sources. In some cases, this may go unnoticed by EM-users who are engaged in EM-dialogues with some of the involved EM-players.

5 Introduction to the AD REM-project

The AD REM project has its roots in one of the sub-projects of an earlier project: a major joint research program (the so called DenK project, [Bunt 92], [Ahn 94], [Ahn 95]) by the IPO, TUE and KUB in multi-modal human-computer interaction based on a cooperative agent model. In the context of DenK,

Eric Peeters and co-workers built the GDP anymation system, and developed the language LOOKS™ (Language for Object Oriented Kinematic Specification, see [Peeters 93a], [Peeters 94], [Peeters 95]).

Many of the proposals in section 4 depend on the availability of an EM-player that is on-the-fly extendable, modifyable and freely programmable. A second seminal ingredient is the availability of a standardised format for the exchange of information (in Object-Oriented terminology: attributes) and executable code and procedure calls (in Object-Oriented terminology: methods and messages). The GDP, developed in the Eindhoven Computer Graphics group [Peeters 95] is a first implementation of a prototype EM-player according to the previously discussed principles. It is a software system that interprets and executes fragments of script written in the language LOOKS in order to generate interactive computer animation sequences.

The LOOKS language is an Object-Oriented prototype of the exchange format for information and code as introduced above. The GDP in combination with LOOKS demonstrates all of the above features, albeit that the current implementation may not be sufficiently efficient, robust, and versatile to accommodate with industry standard quality criteria. Also, system classes still have to be added to the GDP in order to support the full 3-D geometry, user interaction, dynamic simulation and network capabilities we alluded to in section 4.

Some of the most important properties of LOOKS/GDP that are implemented thus far include the following:

- Object-orientedness, including genericity, multiple repeated inheritance and redefinition.
- Asynchronous communication of language fragments, viz. class definitions, object instantiations, en method calls. All of such LOOKS fragments may be sent to a running GDP (EM-player) and they have immediate effect after successfull parsing has been completed. This makes the EM-player programmable as introduced in the previous section.
- Runtime dynamically linkable system classes in pre-compiled code for higher performance. The exchange format has to be platform independent, and therefore it has to be interpreted. This yields non-optimal execution performance, which is remedied by compiled code (at the expence of platform independency).
- A large and increasing collection of system classes for 3-D geometry, including kinematics, some simple forms of kinematic control (pseudo inverse kinematics, [vO93b]), dynamics (using the methods from [vO94], [Overveld 94]), user interaction based on traditional windows and (software) event processing.
- Multi-user access: a GDP may be wired to several input channels to receive LOOKS fragments from several active agents affecting the same running simulation. Moreover, each of these agents may have his own window with an independently controlled synthetic camera to provided multiple views on the simulation in progress.
- Pseudo concurrency with a discrete-time based synchronisation mechanism.

6 State of the art of AD REM; detailed description of the sub projects

In order to arrive at a prototype AD REM system with full demonstratable functionality, completion of the GDP/LOOKS combination has been adopted as the common prior interest project of the Eindhoven Computer Graphics Group. Now that the first implementation of the GDP is available and the LOOKS 1.0 language definition has been released, the project definition is based on a model of loosely coupled functional components. These components come in 2 categories: generic and specific components.

Generic components. The heart of the GDP is a parser/interpreter for the object-oriented language LOOKS. This subsystem maintains two internal data structures: a table containing all currently defined classes with, for each class, pre-compiled representations of the methods of that class; and a table containing all currently instantiated objects.

The class table is initialised with all so called system-classes: the classes that implement the basic functionality of the GDP (including standard variable types, vector algebra, geometric model representation, kinematics, dynamics, user interaction, viewing, rendering, etc.). During runtime, the contents of the class table is extended with new class definitions that are communicated to the running GDP as fragments of LOOKS code. These new class definitions may inherit from earlier classes, methods may be redefined, etc.

The object table contains all instantiated objects. Objects may be instantiated as a side effect of the execution of a method, or a fragment of LOOKS code may contain an instruction to instantiate a new object during runtime. The time dependent behaviour of objects is dictated by the (pseudo concurrent) execution of methods. Again, a method may start execution as a result of another method or as the effect of a fragment of LOOKS code, issued during runtime.

Generic components of the AD REM project include:

- system integration (i.e. sufficiently versatile and coherent system classes),
- software engineering aspects (separating platform dependent functionality, modularity, portability aspects),
- software interfacing and file format conversions to accommodate links to other systems that offer conventional MM-services and/or 3-D geometric data exchange,
- testing and (small) demo applications,
- communication with possible customers and/or users.

Specific components. The specific components include: (1) management of (geometric) constraints; (2) geometric datastructures for efficient management of large 3-D scenes, including the detection of collisions of moving geometric objects; (3) more sophisticated methods for user interaction with complex geometric scenes where objects may be moving; (4) kernel engineering (the internal execution mechanism, inter-process communication, event management, the internal aspects of constraint management); (5) dynamic simulation (the simulation of kinematic and dynamic systems, rigidity and inertia effects, collision response, dynamic constraints); (6) network aspects (data encapsulation, i.e. the way to embed traditional MM-items in LOOKS fragments, protection for multiple writable data, protocols for negotiation and communication between two possibly differently configured GDP's, information hiding issues, synchronisation).

In the roughly 10 man-years of development effort of GDP/LOOKS, most of the above items have received attention, and the current GDP implementation contains at least rudimentary functionality in most of the above areas. Below we give a more detailed account of the current status and the initiated sub projects of AD REM:

(1) geometric constraints As yet, a limited class of geometric constraints to govern the motion of articulated linked rigid objects is supported. Both hard constraints ([Overveld 92] [vO93a]) and soft constraints ([vO93b]) for the six main types of object-object connections (telescope joint, revolute joint, cylinder joint, sphere joint, helix joint and planar joint) are solved by relaxation. Some added functionality includes range limits and (relative) velocity limits, and non-trivial control algorithms can be built on these functions. The new research encompasses the design and

implementation of a framework for the management of different types of constraints, loops in the constraint network, and overconstained and underconstrained specifications [Veltkamp 94]. The idea is to exploit the structure of the constraint network [Freuder 94] by grouping clusters of constraints and variables into higher-order constraints [Tyugu 94] which may be satisfied by specialized solvers.

(2) geometric datastructures The current datastructure for geometric scenes supports dynamic hierarchies. That is, hierarchical relations (similar to PHIGS [Blake 93], [(ANSI) 88]) may be specified between rigid components. The geometric relations between parents and children can be any of the above joint types plus arbitrary affine transforms. Moreover, these hierarchical relations may be dynamically changed. In the proposed research, efficiency aspects of this data structure will be taken into account, where we focuss on two main issues:

- paging strategies to restrict the size of the resident datastructure to contain only the currently visible portions;
- spatial indexing strategies to support rapid overlap detection (cf. [Boyse 79], [Dobkin 83], [Hopcroft 83], [Uchiki 83], [Preparata 85], [Hahn 88], [Moore 88], [Samet 90], [Zyda 93], [Garcia-Alo 94], and variations thereof) and collision detection of the (assumed polyhedral) rigid components in a scene.

(3) user interaction with changing scenes The interaction functionality in the system classes of the GDP consists of the traditional window, panel, menu, and button style interaction, together with the associated event management and callback machinery, plus selection and navigation in the synthetic image by means of a 3-D geometric manipulation widget (3-D cursor) [vO89]. In the proposed research, the functionality of this widget will be significantly enhanced with a repertoire of navigation, selection and manipulation modes. The control of the widget will be made transparent with respect of the applied physical device (either mouse, 3-D mouse, 6-D bat, or data glove). Earlier work on interaction with dynamic animations is [Ball 94]; see also [Strauss 92].

(4) kernel engineering The GDP kernel is engineered round a virtual machine which drives the pseudo concurrent method calls, and the maintenance of class and object tables. In the proposed research, the kernel will also be responsible for constraint maintenance. We are working the friction between the global and compelling nature of constraints and the data encapsulation and information hiding principles of Object-Orientedness [Blake 91]. To solve this friction, we propose a strict separation of the communication schemes for constraint handling from the normal communication between objects by message passing [Veltkamp 93] [Veltkamp 95]. Communication between constraint and the object is done via events and data-flow. The LOOKS interface to define constraints in a declarative style has to be elaborated. Furthermore, some re-engineering will improve the memory efficiency of the GDP as well as reduce the call overhead due to the dynamic allocation of intermediate internal objects.

(5) dynamics For some types of dynamics constraints (point-to-point, point-to-world and line hinge joints), the current GDP supports numeric machinery based on relaxation that evaluates the motion equations associated with the effects of inertia, external forces and reaction forces (based on [vO94], [Barenbrug 94]). Still, much can be gained both in terms of compuational efficiency and (language) interface versatility. The proposed research will focuss on, on the one hand, enhancing the repertoire of dynamic simulation features (e.g. massless components, friction, collision handling, ropes, more sophisticated dynamics constraints) and on the other hand on more efficient numerical techniques (iterative techniques, approximations based on perturbation methods)

to handle the sets of coupled algebraic differential equations that implement the dynamic simulation. Earlier work on the simulation of dynamical systems includes [Witkin 90], [Metaxas 92], [Barzel 88], [Herr 90], [Isaacs 87], [Isaacs 88], [Haumann 88], [Wilhelms 88], [Armstrong 85], [Haug 84], [Platt 92].

(6) networking Currently, multi-user access to the GDP has been implemented via a software switch internally based on UNIX pipes. This supports configurations where one GDP communicates with 2 separate users. For a much more advanced multi-GDP network, the connection between these switch boxes and the underlying global network requires protocols for synchronisation, data encapsulation, remote procedure calls, and routing. Other relevant aspects are protection against method calls to non-existing remote objects, instantiation of objects from non-existing remote classes, and attempts to define classes that inherit from non-existing remote classes. Earlier work on distributed interactive systems is reported in [Buchanan 92], [Gibbs 92], [Brondmo 91], and [Liestol 94].

The current AD REM development team (the Computer Graphics group of Eindhoven University of Technology) possesses sufficient knowledge and experience to handle all of the above design and implementation issues. Nevertheless, a much faster progress of the development of a full-fledged AD REM system, a broader functionality, and more advanced implementation technology for such issues as network aspects and system architecture, can be assured if connections with other research initiatives can be established.

Publications related to the AD REM project

[vO91] C. W. A. M. van Overveld. The generalized display processor as an approach to real time interactive 3-d computer animation. *The Journal of Visualisation and Computer Animation*, 2(1):16–25, 1991.

[Peeters 91] Eric A. J. Peeters. The generalized display processor. Hand-out. Workshop Object-Based Concurrent Computing, ECOOP (Geneve), 1991.

[Peeters 92] Eric A. J. Peeters and Rita Roelfs. An informal description of looks. Technical report, Technical University of Eindhoven, 1992.

[Peeters 93a] E. A. J. Peeters. Design and implementation of an object-oriented, interactive animation system. In Christine Mingins, William Haebich, John Potter, and Bertrand Meyer, editors, *Technology of Object-Oriented Languages and Systems, TOOLS 12 & 9*, pages 255–267. Prentice Hall, 1993.

[Peeters 93b] E. A. J. Peeters. Syntax and semantics of looks. Technical report, Technical University of Eindhoven, 1993.

[Peeters 94] Eric A. J. Peeters and Dieter Hammer. Concurrency issues in object-oriented computer animation. In *Workshop Concurrent Object-Based Systems (COBS), Dallas*, 1994.

[Peeters 95] Eric A. J. Peeters. *Design of an Object-Oriented Interactive Animation System*. PhD thesis, Technical University of Eindhoven, 1995.

Other references

[Ahn 94] R.M.C. Ahn, R.J. Beun, T. Borghuis, H.C. Bunt, and C.W.A.M. van Overveld. *The DenK-architecture: a pragmatic approach to user interfaces*. internal report IPO (manuscript 1009), IPO, EUT,Eindhoven, the Netherlands, 1994.

[Ahn 95] R.M.C. Ahn, R.J. Beun, T.Borghuis, H.C.Bunt, and C.W.A.M. van Overveld. The DenK-architecture: a pragmatic approach to user interfaces. *Artificial Intelligence Review (accepted for publication)*, 1995.

[(ANSI) 88] American National Standards Institute (ANSI). *American National Standard for Information Processing Systems - Programmer's Hierarchical Interactive Graphics System (PHIGS) Functional Description, Archive File Format , Clear-Text Encoding of Archive File,reference number ANSI X3.144-1988*. ANSI, New York, 1988.

[Armstrong 85] William Armstrong and Mark Green. The dynamics of articulated rigid bodies for purposes of animation. *The Visual Computer*, 1:231–240, 1985.

[Ball 94] J. Eugene Ball, Daniel T. Ling, David Pugh, Tim Skelly, and Andrew Stankosky. Reactor: A system for real-time, reactive animations. In *Proc. of ACM CHI'94 Conference on Human Factors in Computing Systems*, volume 2 of *DEMONSTRATIONS: CSCW*, pages 39–40, 1994.

[Barenbrug 94] B. Barenbrug. Using a constraint driven approach to dynamic simulation of rigid body movements in computer animation. Master's thesis, Eindhoven University of Technology, April 1994.

[Barzel 88] Ronen Barzel and Alan H. Barr. A Modeling System Based On Dynamic Constraints. *Computer Graphics (Proc. SIGGRAPH 88)*, 22(4):179–188, August 1988.

[Blake 91] E. H. Blake, C. Laffra, B. Freeman-Benson, and P. Wißkirchen. *Object and Constraint Paradigms for Graphics*, volume 22 of *SIGGRAPH Course Notes*. ACM SIGGRAPH, 1991.

[Blake 93] John W. Blake. *PHIGS and PHIGS PLUS, an introduction to 3D computer graphics*. Academic Press, London, UK, 1993.

[Boyse 79] John W. Boyse. Interference detection among solids and surfaces. *Communications of the ACM*, 22(1):3–9, January 1979.

[Brondmo 91] H.P. Brondmo and G. Davenport. Creating and viewing the *elastic charles – a hypermedia journal*. In McAleese R. and C. Green, editors, *Hypertext: state of the Art*, pages 43–51, Oxford, 1991. Intellect.

[Buchanan 92] M. Cecelia Buchanan and Polle T. Zellweger. Specifying temporal behavior in hypermedia documents. In *Proceedings of the ACM ECHT '92 Conference on Hypertext*, pages 262–271, 1992.

[Bunt 92] Harry Bunt and Robbert-Jan Beun. Denk: Dialogue management and knowledge acquisition. In *Think Quarterly*, volume 1, pages 15–24. The Institute for Language Technology and Artificial Intelligence, Tilburg University, P.O.Box 90153, 5000 LE Tilburg, The Netherlands, 1992.

[Dobkin 83] David P. Dobkin and David G. Kirkpatrick. Fast detection of polyhedral intersection. *Theoretical Computer Science*, 27(3):241–253, 1983.

[Freuder 94] Eugene Freuder. Exploiting structure in constraint satisfaction problems. In Mayoh et al. [Mayoh 94]. Proceedings NATO Advanced Study Institute on Constraint Programming, Pärnu, Estonia, August 1993.

[Garcia-Alo 94] Alejandro Garcia-Alonso, Nicolas Serrano, and Juan Flaquer. Solving the collision detection problem. *IEEE Computer Graphics and Applications*, 14(3):36–43, May 1994.

[Gibbs 92] Simon Gibbs. Video nodes and video webs: Uses of video in hypermedia. In *Proceedings of the ACM ECHT '92 Conference on Hypertext*, 1992.

[Hahn 88] James K. Hahn. Realistic animation of rigid bodies. *ACM Computer Graphics*, 22(4):299–308, August 1988.

[Haug 84] Edward J. Haug. Elements and methods of computational dynamics. *Computer Aided*

Analysis and Optimization of Mechanical System Dynamics, f9:3–38, 1984.

[Haumann 88] D. R. Haumann and R.E. Parent. Mixed methods for complex kinematic constraints in dynamic figure animation. *The Visual Computer*, 4(6):332–347, 1988.

[Herr 90] Charles Herr. Integrating Kinematic and Dynamic Animation. Master's thesis, University of Calgary, Dept. of Computer Science, November 1990.

[Hopcroft 83] J. E. Hopcroft, J. T. Schwartz, and M. Sharir. Efficient detection of intersections among spheres. *The International Journal of Robotics Research*, 2(4):77–80, 1983.

[Isaacs 87] Paul M. Isaacs and Michael F. Cohen. Controlling Dynamic Simulation with Kinematic Constraints, Behavior Functions and Inverse Dynamics. *Computer Graphics (Proc. SIGGRAPH 87)*, 21(4), July 1987.

[Isaacs 88] Paul M. Isaacs and Michael F. Cohen. Mixed methods for complex kinematic constraints in dynamic figure animation. *The Visual Computer*, 4(6):296–305, 1988.

[Liestol 94] Gunnar Liestol. Aesthetic and rhetorical aspects of linking video in hypermedia. In *Proc. ACM European Conference on Hypermedia Technology ECHT 94*, pages 217–223, 1994.

[Mayoh 94] Brian Mayoh, Enn Tougu, and Jaan Penjam, editors. *Constraint Programming*, volume 131 of *NATO ASI Series F: Computer and Systems Sciences*. Springer-Verlag, 1994. Proceedings NATO Advanced Study Institute on Constraint Programming, Pärnu, Estonia, August 1993.

[Metaxas 92] Dimitri Metaxas and Demetri Terzopoulos. Dynamic deformation of solid primitives with constraints. *Computer Graphics (Proc. SIGGRAPH 92)*, 26(2):309–312, July 1992.

[Moore 88] Matthew Moore and Jane Wilhelms. Collision detection and response for computer animation. *ACM Computer Graphics*, 22(4):289–298, August 1988.

[Overveld 92] C.W.A.M van Overveld and Erik van Loon. Hanging Cloth and Dangling Rods: A Unified Approach to Constraints in Computer Animation. *The Visual Computer*, 3:45–79, June 1992.

[Overveld 94] C.W.A.M van Overveld and Hyeongseok Ko. Small steps for mankind: toward a kinematically driven dynamic simulation for curved path walking. *The Journal of Visualisation and Computer Animation*, 5:143–165, 1994.

[Platt 92] J. Platt. A Generalization of Dynamic Constraints. *Graphical Models and Image Processing*, 54(6):516–525, November 1992.

[Preparata 85] Franco P. Preparata and Michael Ian Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, 1985.

[Samet 90] Hanan Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, Reading, Mass, 1990.

[Strauss 92] Paul S. Strauss and Rikk Carey. An object-oriented 3d graphics toolkit. In *SIGGRAPH 92*, pages 341–349, 1992.

[Tyugu 94] Enn Tyugu and Tarmo Uustalu. Higher-order functional constraint networks. In Mayoh et al. [Mayoh 94]. Proceedings NATO Advanced Study Institute on Constraint Programming, Pärnu, Estonia, August 1993.

[Uchiki 83] Tetsuya Uchiki, Toshiaki Ohashi, and Mario Tokoro. Collision detection in motion simulation. *Comput. and Graphics*, 7(3–4):285–293, 1983.

[Veltkamp 93] Remco C. Veltkamp, Edwin H. Blake, Paul J. W. ten Hagen, and Cornelius W. A. M. van Overveld. Constraints in object-oriented interactive graphics. *SION Research Project, Grant 612-322-212*, 1993.

[Veltkamp 94] Remco C. Veltkamp, Farhad Arbab, and Cornelius W. A. M. van Overveld. Constraint-based graphics. *SION Research Project, Grant 612-31-001*, 1994.

[Veltkamp 95] Remco C. Veltkamp and Edwin H. Blake. *Event-based.constraints: coordinate.satisfaction -> object.state*. Focus on Computer Graphics. Springer, 1995.

[vO89] C.W.A.M van Overveld. Application of a perspective cursor in interactive 3-d wire frame modelling. *Computer Aided Design*, 21(10):619–630, December 1989.

[vO93a] C.W.A.M. van Overveld. 30 Years after Sketchpad: relaxation of geometric constraints revisited . *CWI Quarterly*, 6(4):363–383, December 1993.

[vO93b] C.W.A.M van Overveld. Building Blocks for Goal Directed Motion. *The Journal of Visualisation and Computer Animation*, 4:233–250, 1993.

[vO94] C.W.A.M van Overveld. A simple approximation to Rigid Body Dynamics for Computer Animation . *The Journal of Visualisation and Computer Animation*, 5:17–36, 1994.

[Wilhelms 88] Jane Wilhelms, Matthew Moore, and Robert Skinner. Dynamic animation: interaction and control. *The Visual Computer*, 4(6):283–295, 1988.

[Witkin 90] A. Witkin, M Gleicher, and W Welch. Interactive dynamics. *Computer Graphics ACM SIGGRAPH; special issue on 1990 symposium on interactive 3D graphics*, 24(2):11–21, March 1990.

[Zyda 93] Michael J. Zyda, David R. Pratt, William D. Osborne, and James G. Monahan. Npsnet: Real-time collision detection and response. *J. Visualisation and Computer Animation*, 4(1):13–24, 1993.