

Encapsulated Models: Procedural Representations for
Computer Animation

DISSERTATION

Presented in Partial Fulfillment of the Requirements for
the Degree Doctor of Philosophy in the
Graduate School of The Ohio State University

By

Stephen Forrest May, B.S., M.S.

* * * * *

The Ohio State University

1998

Dissertation Committee:

Prof. Wayne E. Carlson, Adviser

Prof. Richard E. Parent

Prof. Neelam Soundarajan

Prof. Charles A. Csur

Approved by

Adviser

Department of Computer and
Information Science

M A N Y P A G E S D I S C A R D E D

1.1.2 Historical Development of Computer Animation Systems

Traditionally, animation systems have neatly filed themselves into one of two distinctly different classifications: scripting and interactive. Modern systems still can be classified in this manner, although increasingly the distinction between the two classifications is being blurred.

1.1.2.1 Scripting Systems

The first computer animations were produced by writing programs to generate the animation using general purpose programming languages (e.g., Fortran). General purpose languages are flexible and inherently allow programmers to take advantage of conventional programming mechanisms (e.g., variables, subroutines, conditional statement execution, iteration) and techniques (e.g., structured programming, object-oriented design, recursion) to produce animation. The main disadvantage of this approach is that many elements needed to generate animations (e.g., specifying geometric shapes, building hierarchies, generating output for renderers, applying transformations to objects) have to be reimplemented for each animation. The amount of work required to provide the underlying framework can easily outweigh the programming needed that is unique to the production of the animation task at hand.

In response to the problems of using general purpose programming languages, researchers developed specialized programming languages, *procedural animation languages*, designed for the task of computer animation [30, 47, 60, 64, 65, 80, 104, 111, 113]. These animation languages are either new programming languages entirely (often with similarities to existing languages) or extensions of existing general purpose languages (*extension languages*). In the case of an extension language, the underlying, general purpose language is called the *base language*. The base language provides conventional programming functionality; the extensions to the base language provide the functionality needed to produce computer animation. Animation languages that are built from scratch (i.e., without an underlying base language) tend to lack the wide range of functionality that extension languages have (particularly with regards to the richness of data types and device I/O), but are

often easier to learn, use, and implement. A detailed discussion of procedural animation languages is given in Chapter 2.

As the field of computer animation began to mature, more and more non-technical users started working with animation systems. Researchers quickly realized that there was a need for systems that did not require complicated programming in order to specify animation [27, 30, 54]. The *Anima II* system [54] was the first animation system to use a high-level *script* — a list of instructions or commands that specify how the objects in the scene are to be animated. Scripts are specified using a *command language*. Command languages are simpler than procedural animation languages and thus easier for non-programmers to use. Command languages are typified by a lack of one or more of the features considered essential for general programming languages like variables, function calls, data structures, and control-flow constructs. Most command languages employ a simple syntax where one command is specified per line and each command has a similar syntax. Often a verb-noun syntax was used as shown below.

⟨verb⟩ ⟨noun⟩ ⟨parameters⟩

The ⟨verb⟩ specifies the action. The ⟨noun⟩ specifies the object that the action will affect. The ⟨parameters⟩ specify additional values required to fully specify the action. The *Anima II* commands shown below are adapted from [54].

```
SET POSITION ball 0, 0, 0, AT frame 1  
CHANGE POSITION ball TO, 1, 0, 0, FROM frame 1 TO frame 100
```

The commands in *Anima II* are executed in parallel so that simultaneous actions can be expressed. In the example above, an object named “ball” is positioned at coordinates (0, 0, 0) at frame 1 and simultaneously moved to coordinates (1, 0, 0) over frames 1 to 100.

The use of parallel commands such as these avoids the need for iterative programming constructs.

The disadvantage of command languages is that they are not procedural (i.e., they do not have the features of general programming languages). By eliminating the language constructs that make learning animation languages difficult, command languages give up the mechanisms that make animation languages powerful. Complex objects that can be represented using mathematical constraints, procedural rules, and behavioral processes cannot be effectively represented using command languages. Authors of systems which employ command languages have to resort to a general programming language to procedurally generate complex scripts as is illustrated in [26, 96]. Newell states “Many systems dictate a data format based on their particular primitive form. As these systems are developed, the need for more and more generality in the facilities provided results in escalations in the complexity of the data format until it begins to resemble a programming language. At this point, the arguments for and against special purpose programming languages become relevant [94].”

1.1.2.2 Interactive Systems

In the late 1970’s and early 1980’s, the emphasis in computer animation turned towards the development of interactive animation systems [26, 43, 48, 53, 81, 96, 129]. Interactive animation systems allow the animator to make continuous modifications to the shape, transformations, and attributes of three-dimensional objects. Modifications are specified using an input device, such as a mouse, and the results are computed and shown (rendered) quickly enough to the user so that the user feels as if he or she is directly manipulating the object on the screen.

Interactive systems have become the predominant method for creating computer animation today because they provide immediate, visual, user feedback. As such, the features of such systems are well known and are typified by the powerful commercial systems such as *Alias PowerAnimator* [2], *SoftImage|3D* [126], and *3D Studio Max* [70].

1.1.2.3 Systems That Combine Scripting and Interaction

Over the years, developers and users have debated about what type of animation system, scripting or interactive, was superior. More recently, researchers agree that both types of systems are important [60, 119]. Three types of systems that incorporate both scripting (procedural) and interactive techniques now exist: procedural animation languages that have integrated interactive capabilities, visual programming systems, and interactive systems that incorporate procedural capabilities.

The most successful system to date that adds interactive capabilities to a procedural animation language is Pixar's *Menv* system [111]. The key technology in *Menv* is a language-level mechanism called the *articulated variable* that permits the procedural representation of objects to be modified interactively. The articulated variable is an extension of previous work by Hanrahan and Sturman [60] and Reynolds [113]. The *Menv* system and articulated variables are discussed further in Section 2.5.

Visual programming systems, such as Haeberli's *ConMan* [55] and the commercial *Houdini* system [120], use a graphical representation of dataflow networks [92, 108]. In a visual programming system, the scene is represented as an acyclic graph (See Figure 1.12). Nodes in the graph represent functions that produce shapes and functions that apply transformations to shapes. Each node, like a function, has a set of inputs and outputs that can be used to pass data between nodes. Outputs of nodes are connected to inputs of other

nodes by interactively connecting the nodes with edges. In addition to inputs (data) from other nodes, nodes typically have other parameters that can be modified interactively and animated over time using conventional animation channels or tracks. Nodes are usually implemented as independent, modular components. New nodes can be implemented in a general purpose programming language (e.g., C or C++) by the user using an API (application programmer's interface) specific to the system. Visual programming systems provide facilities for procedural animation and are easier for non-programmers to use because they are interactive and the syntax of a programming language is avoided. However, more complex types of procedural animation, such as flocking [114], can be difficult to express when communication channels between components are complex or implicit, when centralized data structures are needed, or when an object-oriented design style is preferred.

Interactive animation systems add procedural animation capabilities through three main mechanisms: command languages for scripting (described earlier), *expressions*, and *plug-ins*. Expressions allow parameters that are usually specified interactively to be specified using simple mathematical expressions that are typed in by the user. Plug-ins are modular software units that can be used to extend the functionality of the system. Like the nodes of visual programming systems, plug-ins are implemented using a general purpose programming language and an API specific to the animation system.

The problem with existing systems that use plug-ins and visual programming is portability. It is difficult, if not impossible, for a procedural object implemented in either type of system to be transferred intact (i.e., with the same procedural capabilities) to a different animation system because the representations used to implement plug-ins and store network graphs are system specific.

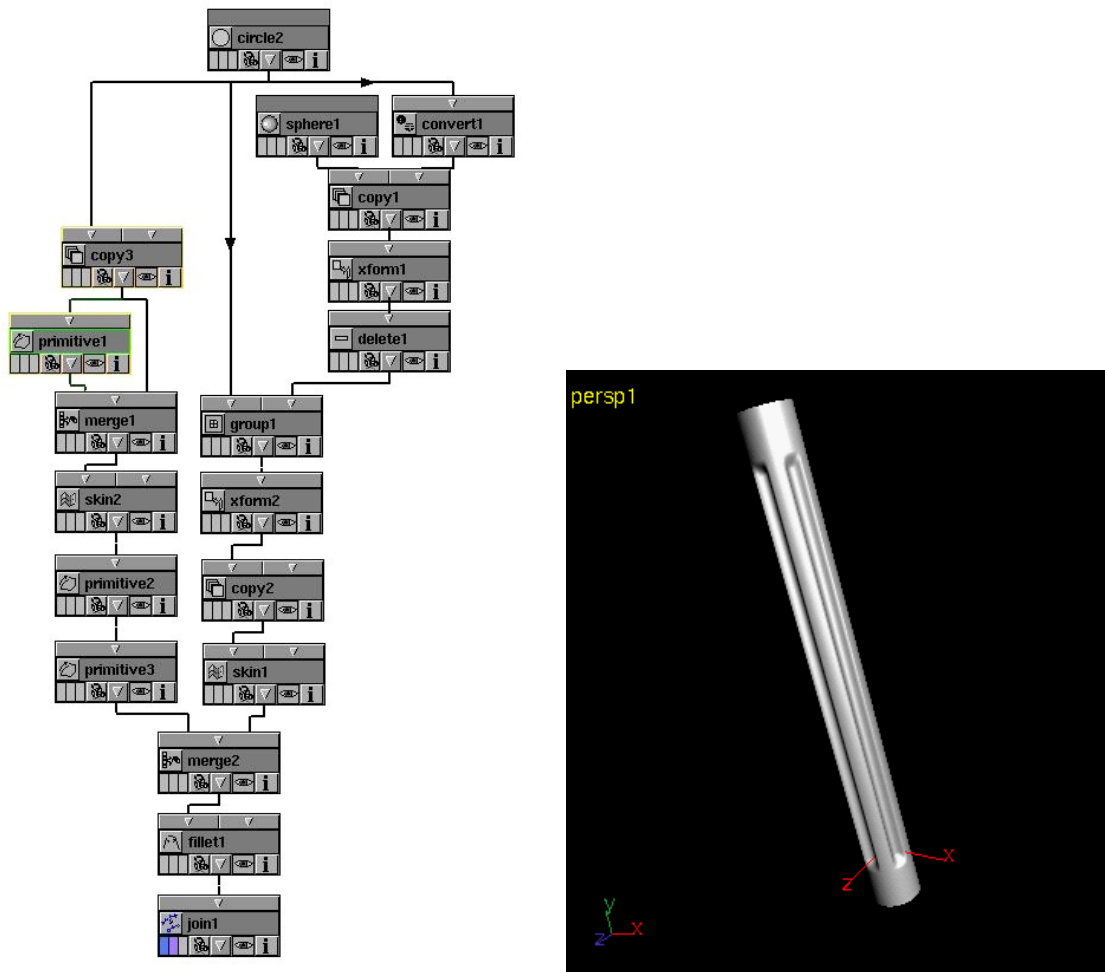


Figure 1.12: A *Houdini* network (left) for a fluted cylinder (right).

1.2 Motivation

This chapter began by introducing the term “encapsulated models.” Recall that an encapsulated model is an object that contains an integrated set of dynamic attributes using a procedural data format (i.e., a procedural animation language). The encapsulated attributes may include some or all of the following: geometric shape, time-varying motion, surface materials, light sources, cameras, sound, and even special effects like smoke and fire.

The concept of encapsulated models is an attempt to improve the representation (i.e., data format) used to describe graphical objects for photorealistic computer animation. An improved representation will yield two major benefits:

1. The complexity and sophistication of objects that can be created will be increased.
2. The cost of producing complex, sophisticated computer animation will be reduced.

Nearly all animate objects exhibit complex, dynamic interactions between motion, form, and other attributes. A procedural representation can model these dynamic interactions effectively. Consider the work of Scheepers *et al.* in anatomically-based modeling of human muscles [117]. Using the AL language, visually realistic, procedural models of the anatomy of humans were developed. The procedural models include different types of muscles that vary in shape, size, placement, and behavior. Complex interactions between the different muscle types, tendons, bones, and skin are modeled. Animation of the skeletal structure (represented as an articulated hierarchy) by the animator results in realistic changes in the musculature and skin. The detailed models of skeletal and muscle layers can be manipulated in real-time. The detail and computational efficiency of the complex musculature is achieved because the underlying procedural data format allows for multiple, highly specialized models of joints, muscles, and tendons to be developed. Models of this complexity (and more) where shape and animation are integrally linked will become increasingly common as artistic demands and computing technology advance.

To argue reduced costs, consider the state of affairs with commercial companies that sell three-dimensional objects for producers of computer animated films. These companies

sell data for thousands of objects including all types of cars, planes, people, furniture, architecture, appliances, etc. Unfortunately, the objects only contain static geometric information (usually polygonal or NURB-based) and in some cases, surface material descriptions. Why? Because the data formats used by animation systems for the description of individual objects lacks the expressive power needed to describe other components. Even simple objects (e.g., an alarm clock, See Section 4.2.5) would be much easier to use if animators could purchase models of objects, “off-the-shelf,” that have predefined movable parts, built-in animation controls, built-in sound effects, and parameters to vary the shape, style, or functionality of the model. In order to provide this type of object, more sophisticated data representations and animation systems that manipulate those representations have to be developed. Currently, the price of three-dimensional data objects is generally based on the number of polygons. With more advanced data representations, the price could be dictated by the sophistication of the self-contained animation, the quality of built-in sound effects, and the range of parameterization.

In order to develop the concept of encapsulated models, the following problems have to be solved:

1. The properties of encapsulated models have to be defined in order to reason about representations and systems for them.
2. The requirements for procedural animation languages as a representation for encapsulated models have to be specified. These requirements can be derived directly from the properties in item 1. The requirements should be independent of any specific programming language so that they can have general applicability.

3. Mechanisms for interactive manipulation of encapsulated models have to be developed. Historically, procedurally based animation systems are difficult to use (even by technicians) because they lack interactivity and require programming. One approach, to make such systems effective, is to develop programming language mechanisms and software tools to support interactive manipulation and construction of procedural representations.
4. Prototype procedural animation languages and interactive animation systems to support encapsulated models must be developed.

MANY PAGES DELETED

BIBLIOGRAPHY

- [1] Adobe Systems Incorporated. *PostScript Language Reference Manual*, second edition, 1994.
- [2] Alias|Wavefront, a division of Silicon Graphics Limited, Toronto, Ontario. *Alias Studio, V8*, 1997.
- [3] Phil Amburn, Eric Grant, and Turner Whitted. Managing geometric complexity with enhanced procedural models. In David C. Evans and Russell J. Athay, editors, *Computer Graphics (SIGGRAPH 86 Conference Proceedings)*, volume 20, pages 189–195. ACM SIGGRAPH, August 1986.
- [4] Susan Amkraut and Michael Girard. Eurhythm: Concept and process. *Journal of Visualization and Computer Animation*, 1(1):15–17, August 1990.
- [5] David Baraff. Coping with friction for non-penetrating rigid body simulation. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH 91 Proceedings)*, volume 25, pages 31–40, July 1991.
- [6] Bren Bataclan. Documentation of a pilipino folk song animation utilizing computer graphics technology and visual communication principles. Master's thesis, The Ohio State University, June 1995.
- [7] Bren Bataclan, Steve May, and Ferdi Scheepers. Bahay kubo: A pilipino folk song. The Ohio State University, August 1995.
- [8] Bruce G. Baumgart. Geometric modeling for computer vision. AIM-249, STA CS-74-463, CS Dept, Stanford University, October 1974.
- [9] D. Scott Birney. *Observational Astronomy*. Cambridge University Press, 1991.
- [10] Preston Blair. *Cartoon Animation*. Walter Foster Publishing, Inc., Tustin, California, 1994.
- [11] James F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, July 1982.

- [12] Beth Blostein. Procedural generation of alternative formal and spatial configurations for use in architecture and design. Master's thesis, The Ohio State University, 1995.
- [13] Beth Blostein and Terry Monnett. Tectonic evolution. The Ohio State University, August 1995.
- [14] David E. Breen, Donald H. House, and Michael J. Wozny. Predicting the drape of woven cloth using interacting particles. In Andrew Glassner, editor, *SIGGRAPH 94 Conference Proceedings*, Annual Conference Series, pages 365–372. ACM SIGGRAPH, Addison Wesley, July 1994.
- [15] C. Wayne Brown and Barry J. Shepherd. *Graphics File Formats*. Manning Publications Co., 1995.
- [16] Wayne E. Carlson. An advanced data generation system for use in complex object synthesis for computer display. In *Proceedings of Graphics Interface '82*, pages 197–204, 1982.
- [17] Wayne E. Carlson. *Techniques for the generation of three dimensional data for use in complex image synthesis*. PhD thesis, The Ohio State University, 1982.
- [18] Wayne E. Carlson. Computer and Information Science 783: Geometric modeling. The Ohio State University, 1996.
- [19] John E. Chadwick, David R. Haumann, and Richard E. Parent. Layered construction for deformable animated characters. In Jeffrey Lane, editor, *Computer Graphics (SIGGRAPH 89 Conference Proceedings)*, volume 23, pages 243–252, July 1989.
- [20] A.H.J. Christensen. “blocked puzzle,” approximation to a doughnut. In James J. Thomas, editor, *Computer Graphics (SIGGRAPH 80 Conference Proceedings)*, volume 16. ACM SIGGRAPH, July 1980.
- [21] Sharon Rose Clay and Jane Wilhelms. Put: Language-based interactive manipulation of objects. *IEEE Computer Graphics and Applications*, 16(2):31–39, March 1996.
- [22] William Clinger, Jonathan Rees, et al. The revised⁴ report on the algorithmic language Scheme. *LISP Pointers*, 4(3), 1991.
- [23] Alison Colman. Deadly mister misty. The Ohio State University, August 1996.
- [24] Robert L. Cook. Stochastic sampling in computer graphics. *ACM Transactions on Graphics*, 5(1):51–72, January 1986.
- [25] Franklin C. Crow. The aliasing problem in computer-generated shaded images. *Communications of the ACM*, 20(11):799–805, November 1977.

- [26] C. Csuri, R. Hackathorn, R. Parent, W. Carlson, and M. Howard. Towards an interactive high visual complexity animation system. In Bary W. Pollack, editor, *Computer Graphics (SIGGRAPH 79 Conference Proceedings)*, volume 13, pages 289–299. ACM SIGGRAPH, August 1979.
- [27] Charles Csuri. Computer graphics and art. *Proceedings of the IEEE*, pages 558–570, April 1974.
- [28] Charles Csuri and Stephen F. May. Playground. <http://www.horizonsmedia.com/csuri/Playground.html>, October 1995.
- [29] Charles Csuri, Stephen F. May, Peter G. Carswell, and Lawson Wade. Queenly gestures. ACM SIGGRAPH 97 Conference, August 1997.
- [30] Tom DeFanti. The digital component of the circle graphics habitat. In *AFIPS Conference Proceedings (National Computer Conference)*, volume 45, pages 195–203, 1976.
- [31] Julie Dorsey and Pat Hanrahan. Modeling and rendering of metallic patinas. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 387–396. ACM SIGGRAPH, Addison Wesley, August 1996.
- [32] Steven M. Drucker, Tinsley A. Galyean, and David Zeltzer. CINEMA: A system for procedural camera movements. In David Zeltzer, editor, *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, volume 25, pages 67–70, March 1992.
- [33] R. Kent Dybvig. *The Scheme Programming Language*. Prentice Hall, second edition, 1996.
- [34] David S. Ebert. Advanced geometric modeling. In Jr. Allen Tucker, editor, *The Computer Science and Engineering Handbook*, chapter 56. CRC Press, 1997.
- [35] David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steven Worley. *Texturing and Modeling: A Procedural Approach*. Academic Press, October 1994. ISBN 0-12-228760-6.
- [36] David S. Ebert and Richard E. Parent. Rendering and animation of gaseous phenomena by combining fast volume and scanline A-buffer techniques. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH 90 Conference Proceedings)*, volume 24, pages 357–366, August 1990.
- [37] Gerald Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*. Academic Press, Inc., San Diego, CA, 1988.
- [38] David Flanagan. *Java in a Nutshell: A Desktop Quick Reference for Java Programmers*. O’Reilly & Associates, Inc., 1996.

- [39] David Flanagan. *JavaScript: The Definitive Guide*. O'Reilly & Associates, Inc., second edition, 1997.
- [40] Kurt Fleischer and Andrew Witkin. A modeling testbed. In *Proceedings of Graphics Interface '88*, volume 21, pages 127–137. Canadian Inf. Process. Society, 1988.
- [41] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics, Principles and Practice, Second Edition*. Addison Wesley, Reading, Massachusetts, 1990.
- [42] Mark Fontana, Kirk Bowers, and Steve May. Butterflies in the rain. The Ohio State University, 1997.
- [43] D. Fortin, J. F. Lamy, and Daniel Thalmann. A multiple track animator system for motion synchronization. In *Proceedings ACM SIGGRAPH/SIGART Interdisciplinary Workshop: Motion: Representation and Perception*, pages 180–186, Toronto, April 1983.
- [44] Nick Foster and Dimitris Metaxas. Modeling the motion of a hot, turbulent gas. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 181–188. ACM SIGGRAPH, Addison Wesley, August 1997.
- [45] Alain Fournier and William T. Reeves. A simple model of ocean waves. In David C. Evans and Russell J. Athay, editors, *Computer Graphics (SIGGRAPH 86 Proceedings)*, volume 20, pages 75–84, August 1986.
- [46] Curtis F. Gerald and Patrick O. Wheatley. *Applied Numerical Analysis*. Addison Wesley, Reading, MA, 1985.
- [47] G. B. Goates, M. L. Griss, and G. J. Herron. PICTUREBALM: A LISP-based graphics language system with flexible syntax and hierarchical data structure. In James J. Thomas, editor, *Computer Graphics (SIGGRAPH 80 Conference Proceedings)*, volume 14, pages 93–99. ACM SIGGRAPH, July 1980.
- [48] Julian E. Gomez. TWIXT: A 3D animation system. In K. Bo and H. A. Tucker, editors, *Eurographics '84*, pages 121–133. North-Holland, 1984.
- [49] Mark Green and Hanqin Sun. A language and system for procedural modeling and motion. *IEEE Computer Graphics and Applications*, 8:52–64, November 1988.
- [50] Robin M. Green. *Spherical Astronomy*. Cambridge University Press, 1985.
- [51] L. Gritz and J. K. Hahn. Genetic programming for articulated figure motion. *Journal of Visualization and Computer Animation*, 6(3):129–142, July 1995.

- [52] Larry Gritz and James K. Hahn. BMRT: A global illumination implementation of the RenderMan standard. *Journal of Graphics Tools*, 1(3), 1996.
- [53] R. Hackathorn, R. Parent, B. Marshall, and M. Howard. An interactive microcomputer based 3-D animation system. In *Proc. of the 7th Canadian Man-Computer Communications Conf.*, pages 181–191, 1981.
- [54] Ronald J. Hackathorn. Anima II: a 3-D color animation system. In James George, editor, *Computer Graphics (SIGGRAPH 77 Conference Proceedings)*, volume 11, pages 54–64. ACM SIGGRAPH, July 1977.
- [55] Paul E. Haeberli. ConMan: A visual programming language for interactive graphics. In John Dill, editor, *Computer Graphics (SIGGRAPH 88 Conference Proceedings)*, volume 22, pages 103–111. ACM SIGGRAPH, Addison Wesley, August 1988.
- [56] Paul E. Haeberli and Kurt Akeley. The accumulation buffer: Hardware support for high-quality rendering. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH 90 Conference Proceedings)*, volume 24, pages 309–318, August 1990.
- [57] J. K. Hahn, J. Geigel, J. W. Lee, L. Gritz, T. Takala, and S. Mishra. An integrated approach to motion and sound. *Journal of Visualization and Computer Animation*, 6(2):109–124, April 1995.
- [58] James K. Hahn. Realistic animation of rigid bodies. In John Dill, editor, *Computer Graphics (SIGGRAPH 88 Proceedings)*, volume 22, pages 299–308, August 1988.
- [59] Mark Hammel and Przemyslaw Prusinkiewicz. Visualization of developmental processes by extrusion in space-time. In Wayne A. Davis and Richard Bartels, editors, *Graphics Interface '96*, pages 246–258. Canadian Information Processing Society, Canadian Human-Computer Communications Society, May 1996.
- [60] Pat Hanrahan and David Sturman. Interactive animation of parametric models. *The Visual Computer*, 1(4):260–266, December 1985.
- [61] David R. Haumann and Richard E. Parent. The behavioral test-bed: obtaining complex behavior from simple rules. *The Visual Computer*, 4(6):332–347, December 1988.
- [62] Li-wei He, Michael F. Cohen, and David H. Salesin. The virtual cinematographer: A paradigm for automatic real-time camera control and directing. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 217–224. ACM SIGGRAPH, Addison Wesley, August 1996.
- [63] C. Hewitt and R. Atkinson. Parallelism and synchronization in actor systems. In *ACM Symposium on Principles of Programming Languages 4*, January 1977.

- [64] Michael B. Johnson. *Wavesworld: A testbed for constructing three-dimensional semi-autonomous animated characters*. PhD thesis, Massachusetts Institute of Technology, 1995.
- [65] Ben Jones. An extended Algol-60 for shaded computer graphics. In Toby Berk, editor, *Computer Graphics (Proceedings ACM Symposium on Graphic Languages)*, volume 10, pages 18–23. ACM SIGGRAPH/SIGPLAN, April 1976.
- [66] K. Kahn. An actor-based computer animation language. In *Proceedings of the ACM-SIGGRAPH Workshop on User-Oriented Design of Computer Graphics Systems*, October 1976.
- [67] Arie Kaufman, Daniel Cohen, and Roni Yagel. Volume graphics. *IEEE Computer*, 26(7):51–64, July 1993.
- [68] Brian W. Kernighan and Dennis M. Ritchie. *The C Programming Language*. Prentice Hall, second edition, 1988.
- [69] Muqem Khan. The platonic problem. The Ohio State University, June 1996.
- [70] Kinetix, a division of Autodesk, Inc., San Francisco, CA. *3D Studio Max*, April 1996.
- [71] J. U. Korein and N. I. Badler. Techniques for generating the goal-directed animation of articulated structures. *IEEE Computer Graphics and Applications*, 2(9):71–81, November 1982.
- [72] John Lasseter. Principles of traditional animation applied to 3D computer animation. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH 87 Conference Proceedings)*, volume 21, pages 35–44, July 1987.
- [73] Oliver Laumann. *The OOPS Package for Elk Scheme*. 1994.
- [74] Oliver Laumann and Carsten Bormann. Elk: The extension language kit. *USENIX Computing Systems*, 7(4), 1994.
- [75] Eli L. Levitan. *Handbook of Animation Techniques*. Van Nostrand Reinhold Company, New York, 1979.
- [76] Marc Levoy. *SIGGRAPH 96 Conference Course Notes: Introduction to Volume Visualization*. 1991.
- [77] Matthew Lewis. Abulafia gallery. The Ohio State University, <http://www.cgrg.ohio-state.edu/~mlewis/Gallery/gallery.html>, June 1995.
- [78] Matthew Lewis. Art 894: Procedural animation. The Ohio State University, 1997.

- [79] Matthew Lewis. Conversations. The Ohio State University, <http://www.cgrg.ohio-state.edu/~mlewis/Converse/converse.html>, July 1997.
- [80] N. Magnenat-Thalmann and D. Thalmann. The use of high-level 3-D graphical types in the Mira animation system. *IEEE Computer Graphics and Applications*, 3:9–16, December 1983.
- [81] N. Magnenat-Thalmann, D. Thalmann, and M. Fortin. MIRANIM: An extensible director-oriented system for the animation of realistic images. *IEEE Computer Graphics and Applications*, 5(3):61–73, March 1985.
- [82] Nadia Magnenat-Thalmann and Daniel Thalmann. *Computer Animation: Theory and Practice*. Springer-Verlag, New York, 1985.
- [83] Martti Mantyla. *An Introduction to Solid Modeling*. Computer Science Press, Rockville, MD, 1988.
- [84] Robert Marshall, Roger Wilson, and Wayne Carlson. Procedure models for generating three-dimensional terrain. In James J. Thomas, editor, *Computer Graphics (SIGGRAPH 80 Conference Proceedings)*, volume 14, pages 154–162. ACM SIGGRAPH, July 1980.
- [85] Stephen F. May. AL: Animation language reference manual. Technical Report OSU-ACCAD-11/94-TR3, ACCAD, The Ohio State University, November 1994. <http://www.cgrg.ohio-state.edu/~smay/AL>.
- [86] Stephen F. May. *LibDm: A Library for Shared Memory IPC Using C++ Objects*. The Ohio State University, August 1995.
- [87] Stephen F. May, Wayne Carlson, Flip Phillips, and Ferdi Scheepers. AL: A language for procedural modeling and animation. Technical Report OSU-ACCAD-12/96-TR5, ACCAD, The Ohio State University, December 1996.
- [88] Neal McDonald and Tonya Ramsey. Context. The Ohio State University, August 1995.
- [89] Jean Meeus. *Astronomical Formulae for Calculators*. Willmann-Bell, 1988.
- [90] Gavin Miller and Andrew Pearce. Globular dynamics: A connected particle system for animating viscous fluids. *Computers and Graphics*, 13(3):305–309, 1989.
- [91] Michael E. Mortenson. *Geometric Modeling*. John Wiley and Sons, 1985.
- [92] Tom Nadas and Alain Fournier. GRAPE: An environment to build display processes. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH 87 Conference Proceedings)*, volume 21, pages 75–84, July 1987.

- [93] Jackie Neider, Tom Davis, and Mason Woo. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Release 1*. Addison Wesley, Reading, Massachusetts, 1993.
- [94] Martin E. Newell. *The Utilization of Procedure Models in Digital Image Synthesis*. PhD thesis, University of Utah, 1975.
- [95] William M. Newman. Display procedures. *Communications of the ACM*, pages 651–660, October 1971.
- [96] T. J. O’Donnell and Arthur J. Olson. GRAMPS – A graphics language interpreter for real-time, interactive, three-dimensional picture editing and animation. In Henry Fuchs, editor, *Computer Graphics (SIGGRAPH 81 Conference Proceedings)*, volume 15, pages 133–142. ACM SIGGRAPH, August 1981.
- [97] Open Software Foundation, Cambridge, MA. *OSF/Motif Programmer’s Guide*, 1990.
- [98] John K. Ousterhout. *Tcl and the Tk Toolkit*. Addison Wesley, 1994.
- [99] Alan Paeth, Ferdi Scheepers, and Stephen F. May. A survey of extended graphics libraries. In Alan Paeth, editor, *Graphics Gems V*. AP Professional, 1995.
- [100] Richard E. Parent. *A system for generating three-dimensional data for computer graphics*. PhD thesis, Ohio State University, 1977.
- [101] Darwyn R. Peachey. Modeling waves and surf. In David C. Evans and Russell J. Athay, editors, *Computer Graphics (SIGGRAPH 86 Proceedings)*, volume 20, pages 65–74, August 1986.
- [102] Ken Perlin. An image synthesizer. In B. A. Barsky, editor, *Computer Graphics (SIGGRAPH 85 Proceedings)*, volume 19, pages 287–296, July 1985.
- [103] Ken Perlin and Athomas Goldberg. IMPROV: A system for scripting interactive actors in virtual worlds. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 205–216. ACM SIGGRAPH, Addison Wesley, August 1996.
- [104] Gregory F. Pfister. A high level language extension for creating and controlling dynamic pictures. In Toby Berk, editor, *Computer Graphics (Proceedings ACM Symposium on Graphic Languages)*, volume 10, pages 1–9. ACM SIGGRAPH/SIGPLAN, April 1976.
- [105] Pixar. PhotoRealistic Renderman. Richmond, CA, 1987-1997.
- [106] Pixar. Tin toy. By John Lasseter and Eben Ostby and William Reeves, August 1988.

- [107] Pixar. *The RenderMan Interface, Version 3.1*, September 1989.
- [108] Michael Potmesil and Eric M. Hoffert. FRAMES: Software tools for modeling, rendering and animation of 3D scenes. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH 87 Conference Proceedings)*, volume 21, pages 85–93. ACM SIGGRAPH, July 1987.
- [109] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag, 1990.
- [110] William T. Reeves. Particle systems – a technique for modeling a class of fuzzy objects. *ACM Transactions On Graphics*, 2:91–108, April 1983.
- [111] William T. Reeves, Eben F. Ostby, and Samuel J. Leffler. The Menv modelling and animation environment. *Journal of Visualization and Computer Animation*, 1(1):33–40, August 1990.
- [112] William T. Reeves, David H. Salesin, and Robert L. Cook. Rendering antialiased shadows with depth maps. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH 87 Conference Proceedings)*, volume 21, pages 283–291, July 1987.
- [113] Craig W. Reynolds. Computer animation with scripts and actors. In R. Daniel Bergeron, editor, *Computer Graphics (SIGGRAPH 82 Conference Proceedings)*, volume 16, pages 289–296. ACM SIGGRAPH, July 1982.
- [114] Craig W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH 87 Conference Proceedings)*, volume 21, pages 25–34. ACM SIGGRAPH, July 1987.
- [115] Barbara Robertson. Toy story: A triumph of animation. *Computer Graphics World*, pages 28–38, August 1995.
- [116] Ferdi Scheepers. *Anatomy-Based Surface Generation for Articulated Models of Human Figures*. PhD thesis, The Ohio State University, June 1996.
- [117] Ferdi Scheepers, Richard E. Parent, Wayne E. Carlson, and Stephen F. May. Anatomy-based modeling of the human musculature. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series. ACM SIGGRAPH, Addison Wesley, August 1997.
- [118] Robert Scheifler and James Gettys. The X window system. *ACM Transactions On Graphics*, 5:79–109, 1986.
- [119] John F. Schlag. Eliminating the dichotomy between scripting and interaction. In M. Green, editor, *Proceedings of Graphics Interface '86*, pages 202–206, May 1986.

- [120] Side Effects Software Inc., Toronto, Ontario. *Houdini 2.0: User Guide*, September 1997.
- [121] David Siegel. *Creating Killer Web Sites: The Art of Third-Generation Site Design*. Hayden Books, 1996.
- [122] Karl Sims. Artificial evolution for computer graphics. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH 91 Proceedings)*, volume 25, pages 319–328, July 1991.
- [123] Karl Sims. Evolving virtual creatures. In Andrew Glassner, editor, *SIGGRAPH 94 Conference Proceedings*, pages 15–22. ACM SIGGRAPH, Addison Wesley, July 1994.
- [124] Alvy Ray Smith. Plants, fractals, and formal languages. In Hank Christiansen, editor, *Computer Graphics (SIGGRAPH 84 Conference Proceedings)*, volume 18, pages 1–10. ACM SIGGRAPH, July 1984.
- [125] John M. Snyder. *Generative Modeling for Computer Graphics and CAD: Symbolic Shape Design Using Interval Analysis*. Academic Press, Inc., 1992.
- [126] Softimage, a subsidiary of Microsoft Corp. *Softimage|3D*, 1997.
- [127] Richard Stallman. *GNU Emacs Manual*. Free Software Foundation, 11th edition, 1994.
- [128] Jos Stam and Eugene Fiume. Depicting fire and other gaseous phenomena using diffusion processes. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 129–136. ACM SIGGRAPH, Addison Wesley, August 1995.
- [129] Garland Stern. Bbop - a program for 3-dimensional animation. In *Nicograph '83*, pages 403–404. December 1983.
- [130] Paul S. Strauss and Rikk Carey. An object-oriented 3D graphics toolkit. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH 92 Conference Proceedings)*, volume 26, pages 341–349. ACM SIGGRAPH, Addison Wesley, July 1992.
- [131] Tapio Takala and James Hahn. Sound rendering. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH 92 Conference Proceedings)*, volume 26, pages 211–220, July 1992.
- [132] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH 87 Proceedings)*, volume 21, pages 205–214, July 1987.

- [133] Frank Thomas and Ollie Johnston. *Disney Animation: The Illusion of Life*. Abbeville Press, New York, fifth edition, 1981.
- [134] T. Towle and T. DeFanti. GAIN: An interactive program for teaching interactive computer graphics programming. In Richard L. Phillips, editor, *Computer Graphics (SIGGRAPH 78 Conference Proceedings)*, volume 12, pages 54–59. ACM SIGGRAPH, August 1978.
- [135] Paul Trachtman. Charles Csuri is an ‘Old Master’ in a new medium. *Smithsonian*, 25(11):56–65, February 1995.
- [136] Steve Upstill. *The RenderMan Companion: A Programmer’s Guide to Realistic Computer Graphics*. Addison Wesley, 1990.
- [137] Lawson Wade and Richard E. Parent. Automatic generation of control skeletons for animation of polyhedral models. Technical Report OSU-ACCAD-3/98-TR1, ACCAD, The Ohio State University, January 1998.
- [138] Graham Walters. *SIGGRAPH 96 Conference Course Notes: The Making of Toy Story*. 1996.
- [139] Alan Watt and Mark Watt. *Advanced Animation and Rendering Techniques: Theory and Practice*. Addison Wesley, 1992.
- [140] Jerry Weil. The synthesis of cloth objects. In David C. Evans and Russell J. Athay, editors, *Computer Graphics (SIGGRAPH 86 Proceedings)*, volume 20, pages 49–54, August 1986.
- [141] Josie Wernecke. *The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor, Release 2*. Addison Wesley, Reading, Massachusetts, 1994.
- [142] Terry Winograd. *Understanding Natural Language*. Academic Press, 1974.
- [143] David Zeltzer. Task-level graphical simulation: Abstraction, representation, and control. In Norm Badler, Brian Barsky, and David Zeltzer, editors, *Making them move: mechanics, control, and animation of articulated figures*, chapter 1, pages 3–33. Morgan Kaufmann, 1991.