Hot Topics

LNAI 2160

Ruqian Lu
Songmao Zhang

Automatic Generation of Computer Animation

Using AI for Movie Animation

Springer

Ruqian Lu   Songmao Zhang

Automatic Generation of Computer Animation

Using AI for Movie Animation

Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Authors

Ruqian Lu
Songmao Zhang
Academia Sinica, Institute of Mathematics
Beijing 100080, P.R. China
E-mail: {rqlu/smzhang}@math08.math.ac.cn

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Lu, Ruqian:
Automatic generation of computer animation : using AI for movie animation / Ruqian Lu ; Songmao Zhang. - Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ; London ; Milan ; Paris ; Tokyo : Springer, 2002
(Lecture notes in computer science ; 2160 : Lecture notes in artificial intelligence)
ISBN 3-540-43114-4

CR Subject Classification (1998): I.2, I.4, I.3, J.5

ISSN 0302-9743
ISBN 3-540-43114-4 Springer-Verlag Berlin Heidelberg New York

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

http://www.springer.de

© Springer-Verlag Berlin Heidelberg 2002
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Boller Mediendesign
Printed on acid-free paper SPIN: 10840290 06/3142 5 4 3 2 1 0

Preface

We are both fans of watching animated stories. Every evening, before or after dinner, we always sit in front of the television and watch the animation program, which is originally produced and shown for children. We find ourselves becoming younger while immerged in the interesting plot of the animation: how the princess is first killed and then rescued, how the little rat defeats the big cat, etc. But what we have found in those animation programs are not only interesting plots, but also a big chance for the application of computer science and artificial intelligence techniques. As is well known, the cost of producing animated movies is very high, even with the use of computer graphics techniques. Turning a story in text form into an animated movie is a long and complicated procedure. We came to the conclusion that many parts of this process could be automated by using artificial intelligence techniques. It is actually a challenge and test for machine intelligence. So we decided to explore the possibility of a full life cycle automation of computer animation generation. By full life cycle we mean the generation process of computer animation from a children's story in natural language text form to the final animated movie. It is of course a task of immense difficulty. However, we decided to try our best and to see how far we could go.

After nearly ten years of effort, we have been successful in developing a rough prototype of this technique which resulted in a software package called SWAN. A cartoon produced by SWAN, called "the three brothers", was shown on CCTV (Chinese Central Television Station) in October 1995. We are aware of the fact that this prototype is rough, because our technique is still far from being mature enough to produce delicate animated movies of commercial value. Nevertheless, we have been encouraged by the conclusion of an appraisal committee chaired by Academician, Prof. Yunmei Dong: the industrialization of the SWAN technique would hold great potential for economical and social benefits.

It is appropriate to make a note here about the contents of this book. Following the completion of our SWAN project, we carried out a review and assessment of the techniques developed in it. Based on the review, we have improved and redesigned many parts of its methodology, partly in the interest of scientific research, partly for the sake of preparing the implementation of the second version of SWAN. In this sense, the contents of this book conforms to the new design, not quite to the originally implemented one.

The SWAN project has received a grant from a special foundation of the President of the Chinese Academy of Sciences. We owe special thanks to the former President of the Chinese Academy of Sciences, Current President of the Chinese Association of Science and Technology, Academician, Prof. Guangzhao Zhou; the former Vice-President of the Chinese Academy of Sciences, Current Vice-President of the Chinese Association of Science and Technology, Academician, Prof. Qiheng Hu; and the former head of department of the Bureau for High-Tech, current Vice-Director of the Institute for Software, Mr. Wenzeng Yang. We are grateful to the Institute of Mathematics, Institute of Computing Technology, Lab for Intelligent Information Processing, Lab for Management, Decision, and Information Sciences, Center for Brain and Mind Science of the Chinese Academy of Sciences for their financial support. SWAN has also been supported by the Chinese Natural Science Foundation (especially project 69733020), by Key Projects of the Ministry of Science and Technology and by 863 High-Tech Projects.

More than 30 graduate students have participated in the SWAN project and made their contributions. Following is a list of some of their names: Qing Geng, Wenhong Zhu, Xiaobin Li, Lixin Yu, Zhenghao Jin, Dejie Yang, Yue Xu, Haihu Shi, Ronglin Wan, Zhaobin Chen, Ying Zhao, Yuan Li, Yi Luo, Bo Gao, Yuejiao Zhou, Yinghao Ma, Yizhen Li, Yujie Wang, Haixia Du, Lu Li, Ping Yang, Lu Fan, Guangfeng Ji, Xuesheng Zhang, Lengning Liu, Fan Yang, Xiaolong Jin, Chengmin Sun, Xiangliang Meng, Jingwei Cai.

We are also very thankful to Prof. Zichu Wei, who has made an important contribution to the SWAN project during discussions about the design and implementation of the SWAN software, and through his involvement in the management of the project.

We thank Prof. Christian Freksa of the University of Hamburg, Prof Bernd Krieg-Brueckner and Prof. Manfred Wischnewski of the University of Bremen for their invitation to give a series of seminar talks about SWAN at these universities. We also thank Prof. Manfred Broy of the Technical University of Munich and Prof. Jörg Siekmann of the University of the Saarland for their invitation to introduce SWAN at these universities. We owe special thanks to Prof. Jörg Siekmann. Without his encouragement and support the publishing of this manuscript would have been impossible. While preparing the talks, the first author of this book had the opportunity to further improve the manuscript.

Last, but not least, we would like to thank Ms. Wenyan Zhang and Dr. Cungen Cao for their valuable help in the final editing of the manuscript. We also thank the editors of the LNAI series and Springer-Verlag, and Ms. Karin Henzold for their kind support and help in the editing and publishing of this manuscript.

June, 2001

Ruqian Lu
Songmao Zhang

Contents

1 Overview of Research on Computer Animation and Related Topics

Although commercially available animation packages provide sophisticated facilities for generating computer animation interactively, although the products of these packages often seem quite attractive for users, the real use of these tools is nevertheless not so easy and requires a lot of tedious work.

Lots of papers have been published on the techniques developed in this field. In this chapter, we summarize shortly the research work on computer animation and related topics reported in the literature. Note that our focus of attention is the automatic generation of computer animation, not the animation itself. Furthermore, we have been focusing on the automatic generation of plots and motions of characters. Therefore, we have paid relatively little attention to the graphics aspects. For example, numerical models for generating physical phenomena, e.g. fire, water, fog, rain, etc. are not included in this survey. Because of the limited access to the current and past literature, this survey is by no means complete. Just as one Chinese idiom says: "for one thing cited, ten thousand may have been left out". The source of the material includes journal papers, proceedings, technical reports and also survey papers of other authors, for example, [Arnold, 1989], [Platt and Barr, 1988] and [Jin, Bao and Peng, 1997] contain quite comprehensive and rich material. Apart from that, we cite also material contained in the survey part of other published papers. We owe many thanks to all of these authors.

1.1 Key Frame Based Computer Animation

1.1.1 Basic Ideas and Numerical Techniques

The most successful technique may be that of key frame based animation generation which is probably the most mature technique of computer aided animation generation. With this technique, human painters design and draw a few animation frames as key frames, then the computer performs interpolation to supplement the missing intermediate frames.

Since interpolation is the key technique used in this approach, the numerical mathematics plays an important role. Steketee et al. proposed to use both position splines and motion splines as interpolation parameters. He called this method double interpolation [Steketee and Badler, 1985]. Brotman proposed to use differential equations of classical mechanics to describe the motion of objects and to consider the key frames as constraint conditions [Brotman, 1988]. In order to calculate the position of an object in some frame, which moves along a trajectory, one needs methods for numerical integration. The traditional technique for that is the Simpson algorithm. Guenter et al. proposed to use instead a combination of a Gauss type numerical integration and a Newton-Raphson iteration to solve the problem [Guenter, 1990]. More important was the work of Shoemake who for the first time introduced the quaternion space as a mathematical tool for Bezier spline interpolations [Shoemake, 1985]. This quaternion method has been widely used by animators since then.

The key frame based method has its own disadvantage. As pointed out by [Bruderlin and Calvert, 1989], in this approach, the quality of a motion is directly proportional to the number of key frames specified. If the desired movements are complicated, then it is the animator, not the system, who controls the motion. On the other hand, though this approach is effective in some cases, it is not suitable for a system for automatic generation of animation, like SWAN [Lu, Zhang and Wei, 1999].

1.1.2 Example of a Desktop System for Animation Generation: The Life Forms

This is a product of the Simon Fraser University, developed by Tom Calvert, Armin Bruderlin, John Dill, Thecla Schiphorst and Chris Welman [Calvert, Bruderlin, Dill, Schiphorst and Welman, 1993]. The work started in 1986 as a project to develop computer-based tools to assist in choreographic for composing dance. Then, they found that this prototype, which was originally aimed at assisting working choreographers, could be further developed into a front-end of general-purpose systems for animation generation and a storyboard style tool for cartoon producers. Initially developed on Silicon Graphics workstations, Life Forms has been now transplanted to Apple Macintosh. Noticeable is the fact that this Macintosh version of Life Forms now interacts with Macromedia Director, Electric Image and Hypercard. This work combined key frame based method with articulate planning. Therefore we cite it in this section.

As a general-purpose desktop animation system Life Forms should have the following characteristics, as proposed by the authors, a relative complete platform for animation manufacturing, a flexible and easy to use tool for non-professional animators and a powerful and less expensive desktop workstation for personal use. In fact, the functions of Life Forms include full modeling, rendering and composition capabilities.

In the following, we summarize the functions of Life Forms in form of an algorithm:

Algorithm 1.1.1

Part One: Make a concept plan about the animation one wants to have

1. Call for the screen windows of animation generation.
2. Select bodies for characters (which can also be simple tokens at this moment) from menus and place them on the stage in appropriate positions and facing appropriate directions.
3. Consider the result of the last step as a key frame and store it in the computer.
4. Repeat this process to get a series of key frames
5. Add some background if needed.
6. Adjust the key frames with rotations, translations and zooms if needed.

Part Two: Implement single character movements

7. Design postures of characters in key frames by selecting and mounting limbs of characters and adjusting their directions, where constraints for limb positions and directions are posed to limit the various possibilities.
8. Using "forward kinematics" to build up body instances by successively adjusting the distal segments.
9. Using "constrained inverse kinematics" to calculate the interpolated positions and angles of articulates.
10. Develop new stances or modify old stances to get a series of body stances.
11. Edit this sequence to get the optimized one with controlled tempo of joint movements (by stretching or squeezing its sub-sequences, i.e. by changing the parameters).
12. For those parts where there is a locomotion (e.g. walk here and there), use procedural methods.

Part Three: Implement multiple character movements

13. Assemble the movements of characters together by using a stage window with a time axis, where the time intervals of character movements are registered and are subject to adjustments.

Part Four: Postproduction

14. Design body models for animation
15. Take human skeletons with 22 to 26 segments for building up characters.
16. Select one from the four possibilities listed in the menu: stick figures, outline figures, contour figures and solid models, for body representation.
17. Run the produced movement sequence of characters to get the animation.

End of Algorithm

1.2 Articulate Planning: Kinematics and Dynamics

Models of kinematics and dynamics are widely studied in the literature. There are two kinds of kinematical models. In the forward kinematical model, motion, rotation and scaling of objects and their articulates are described as functions of time. The success of this approach depends largely on the knowledge, skill and experience of the animator. Another kind of the kinematical approach is the inverse kinematics. In this approach, the animator may draw trajectories of movements as a set of constraints and let the computer calculate the motion of articulates. The number of constraints one can impose on the articulates depends on the number of degrees of freedom (DOF) of the objects. Generally speaking, each constraint "consumes" a degree of freedom. The complexity of the corresponding algorithms may be very high and may increase drastically with the increase of articulate numbers. Therefore, one tends to use numerical methods to obtain approximate solutions. An advantage of kinematical models is the possibility of defining constraints for the motions. For example, the system may let a character pick up an apple from the table while keeping a fixed distance off the table and fixing the feet on the ground. Both kinematical models have the disadvantage that the produced articulate movements are often not natural and smooth enough.

Pure (forward or inverse) kinematics ignore any influence from the existence of inertia. Mathematically, it was Denavit and Hartenberg [Denavit and Hartenberg, 1955] who was the first to propose a matrix method for describing the relative positions of articulates based on a relative Cartesian system. This method was then widely used by animators [Strurmann, 1986]. Korein proposed a method whose principle was to minimize the articulate translation. This principle brought the problem that the solutions thus obtained are not necessary natural. Girard and Maciejewski [Girard and Maciejewski, 1985] [Girard, 1987] used a pseudo inverse Jacobian matrix to find solutions in their inverse kinematics method and obtained better methods. Jack [Badler, 1991] is a multi-faceted system which was strongly based on pure kinematics. It was aimed at the description and generation of articulated body movements by interactively modeling, manipulating, and animating geometrically accurate human bodies. This human body model can be used in compliance with anthropometrical statistical data.

The goal-oriented technique is a special form of inverse kinematics method. Zeltzer [Zeltzer, 1982] developed a task-oriented system to animate human body motions such as walking and jumping. He used methods of kinematics and measured data interpolation to calculate joint rotation angles. These methods have difficulty in changing motion speed and step lengths. An improvement of his work was made by Bruderlin and Calvert [Bruderlin and Calvert, 1989], who implemented the KLAW (Keyframe-Less Animation of Walking) system. This is a hybrid approach which combines goal-directed and dynamic motion control. The overall architecture is a hierarchical control process. The user first specifies the wanted locomotion at the top level as tasks which are then decomposed in subtasks. These subtasks are solved by dynamic models of legs and calculated by numerical approximation.

On the other hand, a pure dynamics method does not make use of kinematics principles. The dynamics simulation is an much more difficult task, it includes complicated calculation of force and torques. This has the advantage that the animation designer does not have to care much about the trajectories of object motion in four-dimensional space (three space dimensions plus one time dimension). For example, to describe the trajectory of an object thrown forwards by some person, we only need to let the computer solve a differential equation. Another advantage of this approach is the naturalness and smoothness of the resulted animation. Generally speaking, in case of full passive systems (no autonomous internal force or joint torque) the dynamics approach is a good choice. But the computational complexity may be very high. And the resulted motion pattern may not be a very well controlled one.

Many authors tried to combine the kinematics approach other methods. One example was Van Overveld and Hyeongseok Ko who proposed feasible way of combination [Overveld and Ko, 1994]. They studied the simulation of bipedal locomotion. As input, they used a 14 DOF simplified human body model, together with walk directions and gait lengths. Using kinematics they calculated the trajectories of feet over time. Then trajectories of some other body parts (head, shoulder, etc.) may be derived from the known ones. All these trajectories form motion constraints for the dynamics algorithms. Thus the motion of all components is driven by their trajectories (if any) plus additional forces and torques (if any). All components without trajectory specifications will be driven by inertia effects. Recently, a software package called IKAN has been developed by the center for Human Modeling and Simulation. It is a complete set of inverse kinematics algorithms combined with analytical methods suitable for an anthropomorphic arm or leg. [IKAN, 2000]

We think that the idea of Zeltzer is closer to that of ours. He even proposed a motto: "Knowledge-based animation" [Zeltzer, 1983] which is similar in some aspects to our knowledge based method of SWAN. Another approach which we think is useful for us was adopted by Maiocchi [Maiocchi, 1990] who has developed an animation system called Pinocchio which records articulated body movements and builds a library of them for use during animation generation. His approach is similar to the one used by our system in the last stage of cartoon generation.

1.3 Path Planning

While planning the motion of a character in a movie, an obvious desire of the animator is to give only commands at task levels. For example, you may wish to let a cat run from under the window up to the top of a tree. Then it is comfortable to write down the command

Run (cat, under (window), top (tree))

Is this command still effective when there are obstacles between the window and the tree? Will it happen that the cat goes through the wall of a house or even goes through the body of its master until it arrives outdoors? We all want to avoid such circumstances. But to avoid obstacles of different sizes, forms and distances requires too many details in motion description such as "turn right with an angle of 90 degrees" and "go forwards 10 meters" etc. What we want to have is to keep the simplicity of descriptions like "run from under the window up to the top of a tree" and avoid the cumbersome task of writing detailed path descriptions. This is the task of path planning.

There has been a lot of research on this topic. Most of them simplified the problem in that they assumed the obstacles having simple forms, e.g. polygons or circles. Various results have been obtained. Most of them are related to two-dimensional cases, because the two-dimensional case is much simpler than the three-dimensional one. A straightforward way of dividing the two-dimensional space in several sub-spaces solves this problem with a time complexity of O (n^2 log n). A well known result was due to T. Lozano-Perez and M.A.Wesley [Lozano-Perez and Wesley, 1979] who developed a method called visible graphs, using which the authors proved a better result for finding a shortest path with the time complexity O (n^2).

This result was then improved. Two sub-problems are often addressed. One is to find a shortest path between two points within a polygon. Another one is to find a shortest path between two points outside of a polygon. In both cases the wanted path should not penetrate the border of the polygon. For the first sub-problem, [Lee and Preparata, 1984] proposed a method of constructing a tree of shortest paths based on a triangulation of the polygon. The time complexity for tree construction is O (n), and that for the triangulation is O (n log n). The total complexity is O (n log n). Thus the complexity of triangulation plays the decisive role. The complexity for solving the second sub-problem is similar. Shi improved the two results to a lower complexity of O (n) [Shi, 1996].

Other results about two-dimensional path planning have been published. [Lozano-Perez and Wesley, 1979] also proved an O (n^2log n) complexity for finding a shortest path in an area with a set of n obstacles all of which are convex polygons. This complexity is also valid for the problem of a moving circle with polygon obstacles [Sharir, 1985].

However, in three-dimensional space, the path planning problem becomes much more difficult. To our best knowledge, until present, only exponential results are obtained [John and James, 1994]. It was not until recently that researchers have proved that the problem of finding a shortest path between two points in a three-dimensional space with a finite set of polyhedral obstacles is a NP complete problem [Canny and Reif, 1987]. Nevertheless, some limited results have been obtained. For the case of an arbitrary number of polyhedral obstacles, the exponential complexity can not be further improved [John and James, 1994]. If the number of polyhedral obstacles, k, is equal to one, [Sharir and Schorr, 1986] has proved a complexity of O (n^3 log n), which was later improved to O (n^2) [Joseph, 1987]. For $k = 2$ one obtained O (n^3 log n) [Baltsan and Sharir, 1988], for $k = 3$ O (n^{4k}) [Sharir and Baltsan, 1986] [Sharir, 1987]. Shi reported an O (n^2) result for $k = 2$ [Shi, 1996].

The results mentioned above have been used to solve a variety of different motion planning problems, including the problem of moving a rectangle object (piano movement, [Schwartz and Sharir, 1983]), the problem of moving rigid obstacle objects in two-dimensional space [Lozano-Perez and Wesley, 1979], the problem of cooperative movement of several circle objects in a two-dimensional space with polygon obstacles [Sharir, 1985], and a generalization of the piano moving problem to the three-dimensional space [Sifrony and Sharir, 1987], etc. For the two-dimensional stick moving problem, Leven and Sharir [Leven and Sharir, 1987] have found a solution with time complexity O (n^2 log n) which improved the result of [Schwartz and Sharir, 1983] (time complexity O (n^5)) a lot. Later, the result of Leven and Sharir was proved to be almost optimal.

In case that the obstacles are movable, the complexity is significantly high. Some researchers assume that a robot may move away obstacles he meets in his way. An analysis of this problem proved that the complexity is NP if the goal place is not fixed, otherwise the complexity is PSAPCE [HuangYK and Ahuja, 1992]. In case that there is only one obstacle, the complexity reduces to O (n^3 $\log^2$ n) [HuangYK and Ahuja, 1992].

In our SWAN system, we did not use these algorithms with high complexity. We have adopted a series of heuristic methods to reduce the problem to a two-dimensional one. The only useful algorithm for us is that of T. Lozano-Perez and M.A.Wesley. We have implemented it which solves our problem most of the time. Besides, we have developed another principle of path finding. Namely, we have found that in many cases it is not the shortest path which is the most suitable, but the “most safe path”, a concept we have developed ourselves.

1.4 Camera Planning

One of the most salient features of a movie as compared with a drama is the multiplicity of view angles and viewpoints of the same plot. The audience can observe the progress of the plot from all possible directions and all possible distances, of course guided by the director’s camera. This fact reminds us of the necessity of an automatic camera planning as a component of automatic animation generation.

In SWAN, we have designed and implemented our camera planning module. As main sources of reference we have taken “Dictionary of Movie Arts”, edited by Xu Nanming, etc. [XuN, 1986] (in Chinese), “Grammar of the Film Language” written by D. Arijon [Arijon, 1976], “Grammar of the Edit” written by Roy Thomson [Thompson, 1993], and several others. It was only after we have finished our work and started to write this monograph that we found and read the paper “The Virtual Cinematographer: A Paradigm of Automatic Real-Time Camera Control and Directing”, written by He Liwei, Michael Cohen and David Salesin [He, Cohen and Salesin, 1996]. Much to our surprise that there are a lot of similarities between their work and that of ours. For example, we share the opinion that camera planning needs a formalized computer language to describe it, that one should summarize thumb rules used by cameramen in form of routines of this language, that a film should be considered as a structure consisting of scenes and shots (rather than just a linear sequence), that one should design a large set of film idioms to be called by camera planning modules, that these program modules have an additional task of reposition the characters in a scene to avoid people to be occluded, etc. Notwithstanding that such similarity exists, there are also a great lot of differences between their work and that of ours, which will be explained in more details below and in later chapters.