TOOKIMA 2.0 - An Interactive Animation System

Alberto Barbosa Raposo[1], Léo Pini Magalhães[1,3], Clarisse Sieckenius de Souza[2], José Tarcísio Franco de Camargo[1], Fernando José Lamanna[1]

[1] Image Computing Group, Dept. of Computer Engineering and Industrial Automation (DCA)
School of Electrical and Computing Engineering (FEEC) – UNICAMP
C.P. 6101, 13083–970 Campinas, SP, Brasil
alberto,leopini,tarcisio,lamanna@dca.fee.unicamp.br

[2] Dept. of Informatics – PUC-Rio
Rua Marquês de São Vicente, 225, 22543–900 Rio de Janeiro, RJ, Brasil
clarisse@inf.puc-rio.br

[3] Computer Science Dept. – CGL, University of Waterloo
200 University Ave. W – Waterloo, Ontario, N2L 3G1, Canada
lpini@cgl.uwaterloo.ca

The production of computer animation without the presence of a programmer requires unobtrusive task-centered interaction between user and system. This paper presents TOOKIMA 2.0 (a **Too**l **Ki**t for scripting computer **M**odelled **A**nimation), an interactive scripting environment for describing kinematic computer-modelled animations. It supports the production of animation at two distinct levels of abstraction. In the lower level, users create animations by writing scripts as a specialized language. In the upper level, users can generate scripts interactively, without knowing the underlying scripting language.The environment is designed so as to introduce basic notions of programming in the upper level, and thus to lead users into acquiring basic programming skills needed for operation in the lower level.

Key words: Computer Modelled Animation - End-User Programming - Graphical User Interfaces

1 Introduction

Computer animation is much more than a combination of sophisticated hardware and software; it involves artistic talents. Computer scientists have been said to be able to create good animation systems, but not as many good animations, for which special skills and artistic training are much needed (Booth 1983). Moreover, the fundamental principles of traditional animation should be applied to computer-modelled animations, since the production of bad animations is often due to lack of familiarity with such principles (Lasseter 1987). It thus becomes apparent that successful animation systems should combine the skills of computer scientists and artists who usually do not have further programming skills.

In order to shorten the gap between these two communities, TOOKIMA 2.0 (a **Too**l **Ki**t for scripting computer **M**odelled **A**nimation) was developed with the involvement of both computer scientists and fine-arts specialists who have used the environment to produce animations and have provided feedback for the software design team. TOOKIMA 2.0 evolved from a set of algorithmic descriptions of computer-modelled animation tools - TOOKIMA 1.0 (Hounsell 1992). It was designed to overcome TOOKIMA 1.0's usability barriers; the former version was in fact a library of programs, and users were required to write full-fledged C programs in order to produce animation.

TOOKIMA 2.0 is the animation part of ProSIm[1] (**Pro**totipação e **S**íntese de **Im**agens foto-realistas e animação), a prototyped graphical environment for synthesizing photo-realistic images and computer animation, developed at DCA - FEEC - UNICAMP. Its development involved two major steps: the design of a scripting language (SL) and that of a graphical user interface (GUI) that conveyed the major constructs of the underlying SL. The user who is interacting with TOOKIMA 2.0 through the GUI maybe totally unaware of the SL, although an evolutionary path from interaction to programming has been conceived of.

In the next two sections, computer animation and user interface design will be briefly discussed to highlight the issues dealt with in TOOKIMA 2.0. Then, the global environment will be presented with an emphasis on its scripting language and user interface. Finally, section 5 will illustrate how animation is actually produced by TOOKIMA 2.0.

[1] See http:// www.dca.fee.unicamp.br / projects / prosim/ prosim.html

2 Computer animation

Animation is "the process of dinamically generating a series of frames of a set of objects, in which each frame is an alteration of the previous frame" (Thalmann 1985). It follows from this definition that "animation" and "motion" are not synonyms, since animation can be characterized by changes in a number of parameters like illumination, color of objects, and the like, which are not cognitively perceived by humans as related to object movement.

The fact that computer animation covers a large range of different techniques has caused some confusion in terminology. "Computer Animation Systems" can refer to a variety of items that have very little in common. A number of initiatives have been taken to classify systems according to differing criteria (Thalmann 1985; Zeltzer 1985; Pueyo 1988), including some of our own (Camargo 1995; Raposo 1996). However, independently of the classification criteria employed in analysis, all computer-modelled animation environments must include three fundamental kinds of tools: (a) a *geometric modeller*, (b) a *renderer* and (c) a *control mechanism*.

The *geometric modeller* is used for the specification of geometric characteristics of all objects in a scene. Modelling techniques include surface modelling (by polygons or functions), solid modelling (by using constructive solid geometry or boundary representation – B-Rep), and generative processes (like fractal geometry) (Wyvill 1990). In ProSIm, a graphical user interface (Malheiros 1994) supports the creation of B-Rep models of solid objects, and that of the camera. Other tools available in the GUI help modelling object texture and scene illumination.

The *renderer* is used to generate the frames of animated sequences. It combines information about form, color and texture of all objects with information about their position (the kind of notion passed by control mechanisms). The renderer uses an algorithm (usually scanline, ray-tracing or radiosity) or some compound system for the actual generation of images. In ProSIm, a public-domain library that includes a powerful scanline Z-buffer algorithm is used (SIPP). It allows for flat, Gouraud and Phong shading, shadow generation, oversampling, and some predefined texture maps (Yngvesson 1994). The whole ProSim environment is designed to support compatibility with other rederers such as POV-Ray (Young 1996) and Renderman (Upstill 1989).

The *control mechanism* accounts for the motion of objects in the animation, as well as for interactions both among objects themselves and between animator and objects. Unlike the geometric modeller and the renderer, for which a number of alternative tech-

niques are available, control mechanisms do not have generic solutions which have been turned into techniques and tools. In fact, most items in the literature refer to case studies where very specific control techniques and models are used (Armstrong 1985; Barzel 1988; Thalmann 1991; Camargo 1994; Koga 1994; Costa 1996; Rose 1996).
The control mechanism used in ProSIm was the original TOOKIMA, which also integrated the geometric modelling and the rendering components of the environment. It also provided a scripting language of its own for scene description and temporal synchronization. However, the language was actually an extension of C (Hounsell 1992) which was rather unwielding for nonprogrammers. Thus, in order to improve the usability of ProSim, a redesign of interface and scripting languages made available for users was clearly called for.

3 User interfaces

The user interface is a critical component of any computer system, since it directly impacts the usability of all software resources embedded in an application. Usability can be defined as a combination of the following factors (Shneiderman 1992) : ease of learning, high speed of user tasks' performance, low user error rate, user retention of constructs over time, and subjective user satisfaction. Moreover, usability can be also related to the ability of users in applying known software to novel problem situations (Adler 1993). This requires not only that user interfaces support learning and understanding of computer applications' conceptual models, but also creative usage encouraged by analogies and generalizations motivated by interface design features.
Recently, some researchers have advocated that the interface language can or should become a programming language for users (Dertouzos 1992; Eisenberg 1995; Gentner 1996; de Souza 1996). In this new scenario, user interfaces should provide people not only with a means of fully exploring the resources embedded in computer applications, but also with a means to customize, extend, and/or combine them into other contexts of needs and usage.
Following this approach, M. Eisenberg claims that well-designed applications should have graphical user interfaces, suitable for naive users, and should be so designed as to "keep an eye toward leading the user gently into programming" (Eisenberg 1995). In other words, applications should contain both an extensive, learnable user interface, and an interpreter for a corresponding programming language. These end-user programming (EUP) languages offer a wealth of resources, like reference mechanisms for domain objects and conditional structures that can be used to express novel processing instructions, which empower users to design their own software.
TOOKIMA 2.0 reflects a design effort to harmonize GUI and EUP languages into a communicative environment that supports computer animation production. As a first step (Raposo 1995), two distinct levels of abstraction have been conceived of: a scripting level and an interactive level (Fig. 1).

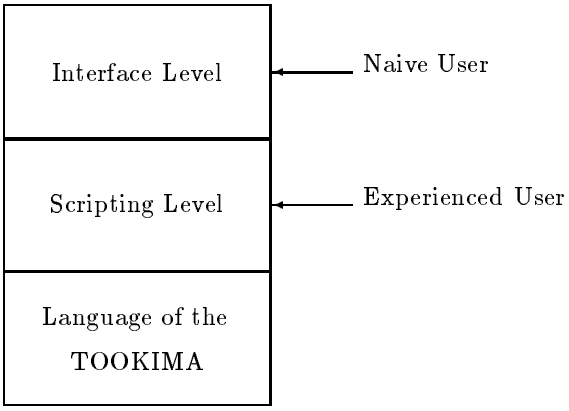

Figure 1: Three abstraction levels for the definition of an animation.

The interactive level maintains strict correspondence with the scripting level, so that interactions can be thought of as being interpreted into TOOKIMA scripts. As in any other GUI, the focus of abstraction in both levels is different. At the interface level, users are allowed to concentrate more closely on task components; required parameters and structures present in scripts can be either provided by default values or prompted for by unobtrusive interactive means (like dialogue boxes, buttons, and the like). At the scripting level, however, task-related focal signs present in the interface are repeated and integrated into extended linguistic constructs that give evidence of explicit control mechanisms and structural articulation of TOOKIMA objects.
Both the interface and the scripting layers are added on top of the original TOOKIMA 1.0 library. This is actually an extensible component of ProSim, but it requires a highly specialized programming knowledge not expected from end users either naive or experienced. The thread leading from concepts of one level to the next deeper level is more continuous from the top level into the middle one, than from this to the deepest library. However, an arbitrary number of layers can be introduced in the environment if needed. Our focus is placed solely on the end-user languages.

The design challenges faced by EUP can be summarized into that of designing a chain of intended high-level interpretations of computer constructs that can lead from algorithms to tasks, from data structures to domain objects, from applications to resources. In it, some concepts provide the interpretive guideline along which extensions, analogies, and semantic transformations are performed by users as they increase their insight about computation.
In the following section we will track how TOOKIMA 2.0 has been elaborated on top of some basic domain concepts that appear first in TOOKIMA's GUI and then in its scripting texts.

4 TOOKIMA 2.0

The purpose of the present phase of the project is to make ProSIm accessible to a larger range of users, specially artists, when developing animations. This is the origin of TOOKIMA 2.0, which changed the structure of the previous tool and provides a scripting language and a user interface, both directed to the animation paradigm of the TOOKIMA.
In TOOKIMA's version 2.0, XView (Heller 1990) was chosen as the development tool. XView is an object-oriented toolkit[2], which was choosen to maintain consistency among the modules of the ProSIm (the modeller had already been developed using this toolkit). To reinforce consistency, the designed interface maintained the layout defined for the interface of the modeller.
In order to achieve compatibility with the other modules of the system, TOOKIMA 2.0 works with the same set of data produced by them (such as the B-Rep files from the modeller). TOOKIMA 2.0 was designed to be modular and relatively independent of a specific geometrical modeller or renderer.
Features of TOOKIMA 2.0 include the generation of animation frames in PPM or TGA formats, the generation of wireframe previsualizations and MPEG animations. To achieve all this features, TOOKIMA 2.0 works in cooperation with other tools, such as the Berkeley MPEG-1 Video Encoder (Gong 1994), the Berkeley MPEG-Player and the XV (Bradley 1994), besides the modeller (Malheiros 1994) and the renderer (Yngvesson 1994), defining ProSIm's environment for modelling, animating and rendering scenes.

[2] A *toolkit* for the development of user interfaces is a library of widgets (menus, buttons, dialog boxes, etc) that can be called by the application (Linton 1989).

4.1 The scripting level

The use of scripts is an usual and powerful method to control the animation process. A script[3] is an ordered sequence of commands, interpreted unambiguously by the animation system.
The scripting language developed for TOOKIMA 2.0 is intermediate between the C-like language of the "old" TOOKIMA and the graphic interface. This new scripting language can be seen as "scene-oriented" (i.e. oriented towards the creation of elements such as actors, cameras, lights, etc.) since the script is divided into modules, each of them dealing with a specific object of the scene (for instance, everything related to the camera goes in the module CAMERA of the script, everything related to the illumination goes in the module LIGHTS, and so on).
In this approach, the animator is not driven to think in a time line and to determine which objects are moving at a given instant. He or she treats the movements of each object (or group of objects) using movement primitives directly (translate, rotate, scale) or defining kinematics rules.
There are 8 possible modules in the script (these modules do not have to appear in this order in the script; the only restriction is that the module GENERAL must be the first):

GENERAL has global parameters of the animation (such as total time and frame rate).

ACTORS has the characteristics (geometry and texture) of the objects that compound the scene and the definitions of their movements .

GROUPS is a language constructor. It creates groups of actors to be moved together.

CAMERA specifies the parameters of the camera and its movements.

LIGHTS defines the light sources of the environment and their movements. It also accounts for the specification of the background color.

RENDER defines the interval of the animation that is going to be rendered, allowing that the animation be partially rendered.

OUTPUT specifies the type of the output (MPEG animation, wireframe, etc) and the name of the output file(s).

[3] This paper will not consider other kinds of interaction (e.g., direct manipulation, which could also be used in animation tasks that are neither very complex, nor repetitive).

TRACKS defines trajectories by the interpolation of curves defined by control points. This trajectories can be followed by actors, groups of actors, light sources or the camera.

The reuse of script components is emphasized by allowing their definition in external files, creating reusable libraries. This applies also to the definitions of the geometry and texture of actors, previously defined using other facilities of ProSIm.
In addition, it is also possible to create point files to define the trajectories to be followed by actors, groups of actors, light sources and/or the camera, as indicated in Fig. 2 by the arrows from TRACKS module .
The schema of the relationship between the script and the libraries is shown in Fig. 2.

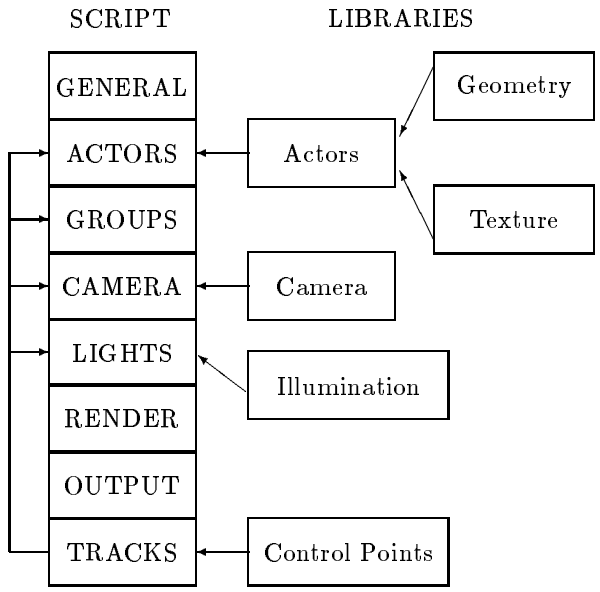

Figure 2: Reusable libraries in the script.

Section 5 will show a detailed example, aiming to demonstrate the main facilities of TOOKIMA 2.0. The complete manual of the scripting language and its formal specification can be found in (Raposo 1995).

4.2 The interface level

TOOKIMA 2.0's interface has the traditional graphic elements (menus, buttons, etc), offering all the necessary functions for the construction of a script. In addition, it has a text area, where the script can be directly manipulated (similarly to text editing). While the script is being constructed by the graphic elements of the interface, it is being updated in the text area, introducing a naive user "gently into programming", and not establishing a cut between naive (presently) and skilled users.
The work area of the interface is shown in Fig. 3, with four distinct sub-areas.
On top of the main window, there is a button line accounting for the construction of the script and the general control of the system (file manipulation, general parameters, etc). Some buttons of this line generate a button column, on the left-hand side. This column functions as a submenu of the button in the top line and is divided into three parts: editing, creation of movements and their cancellation. In the editing part, the element can be created and altered. In the movement specification part, both the position of the element and its physical characteristics can be altered. The option "metamorphosis" offers the possibility to link a list of B-Rep files (describing some geometrical modification) to an actor, within a frame interval. In the cancellation area, the movements of the element can be deleted. In particular, the ACTORS module (as shown in Fig. 3) has a fourth area, in order to access the geometric and the texture modellers of ProSIm.
Most of the main window surface is occupied by the text area, where the script is written (by the system itself, according to the user interaction with the graphical interface, or by the user, who directly edits the script).
The third part of the main window is the message area, at the bottom. In this area are shown the name of the script being edited (on the right-hand side) and messages regarding the current command (on the left-hand side). This messages provide online help tips.
There are two levels for movement control at the interface. On the upper level, actors, camera and light sources are associated with previously created trajectories[4]. On the lower level, movements are directly defined, by the use of commands such as *translate* and *rotate* (for actors); *zoom*, *travelling* and *pan* (for the camera).
When the script (or the script prototype) is completed, the frames can be rendered. The rendering process is executed in background, allowing the use of the interface or of other programs meanwhile. While the frames are being rendered, a button appears in the message area, allowing the user to abort this process. In addition, a preview tool is also available to help the animator in the process of modelling an animation.
Previewing an animation is a very powerful design

[4]The interface also allows the creation of these trajectories, using a "trajectory editor".

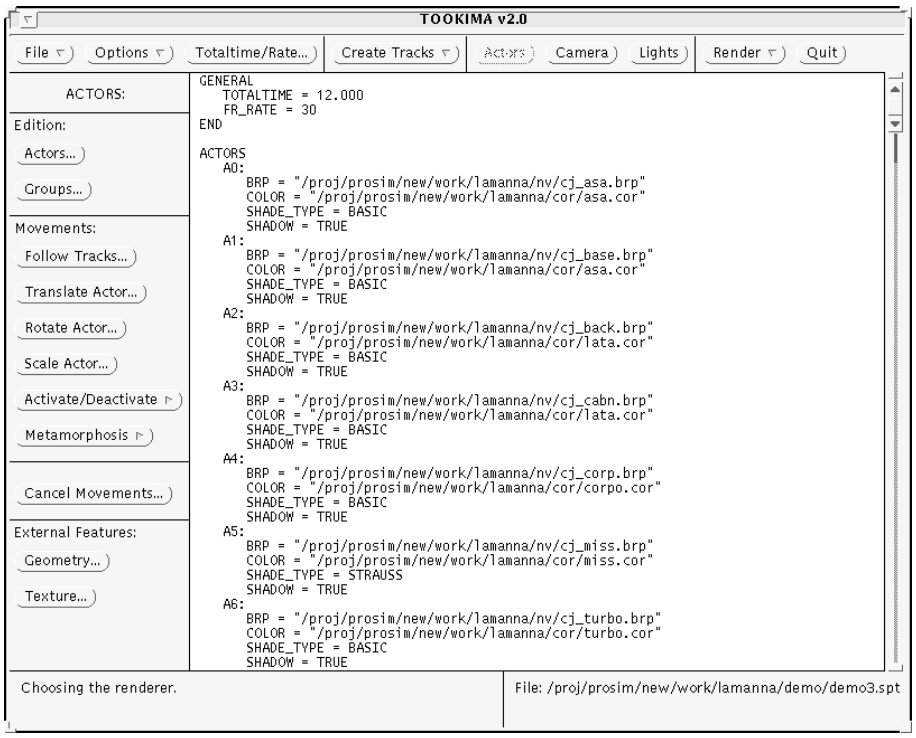

Figure 3: The main window of the TOOKIMA 2.0.

strategy which is fully supported by the defined environment, by means of wireframe and MPEG previsualizations of the animation or only parts of it.

4.3 Information flow

The environment of the TOOKIMA 2.0 is a script-driven animation environment. TOOKIMA 2.0's script can be interactively and/or directly edited, by means of its graphical or text-editing tools. The defined script (a script will always be generated) will be then translated into a C program, compiled and executed. The output of execution is a set of n files (where n is the number of frames to be generated), which are then interpreted by the renderer, that generates the corresponding n frames. If needed, the MPEG encoder is used to encode these images into a MPEG animation. This information flow is schematically shown in Fig. 4.
It is important to note that all the intermediate processes are transparent to users, who only work on the script and see the results (the images or the animation).
The quality of the images is independent of the animation defined by the user, since different renderers can be associated with a script. Currently, SIPP is being used, and two other renderers (POV-Ray and Renderman) are planned to be used.

5 An Example

In Fig. 5, frames of an animation developed using the TOOKIMA 2.0 can be seen. The script of this animation is partially given below (the lines starting with ";" are comments).
The animation shows a spaceship flying over a space station. The GENERAL module determines it has 900 frames (30 seconds at a rate of 30 frames/s). The objects that constitute the spaceship and the station are defined in the ACTORS module (actually, the animation has more than 100 actors, but only two are going to be shown below). In the GROUPS module, some actors are grouped to compose the spaceship. These groups represent the mobile objetcs of the animation, whose movements are defined in the same module. The CAMERA module defines not only the camera parameters, but also its movements: that of changing its position and that of aiming at the spaceship (i.e., the observer follows a predefined trajectory and is always looking in the direction of the spaceship). The background color and the light sources of the animation are defined in the module LIGHTS. The module RENDER determines that all the frames will be rendered, and the module OUTPUT defines their quality and names.

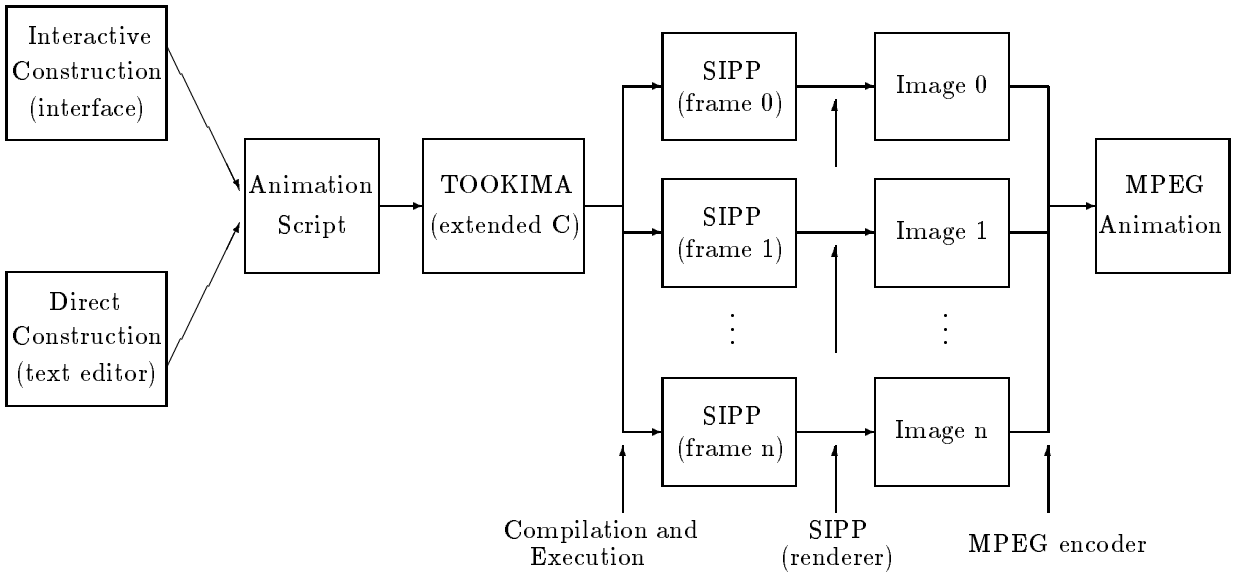

Figure 4: The information flow of the TOOKIMA 2.0.

```
GENERAL
   ;Animation with 30s
   TOTALTIME = 30.
   ;Rate of 30 frames/s
   FR_RATE = 30
END

ACTORS
   ;First actor
   A0:
      ;B-Rep file defining the geometry
      BRP = "/proj/prosim/brp/cj_asa.brp"
      ;Texture file
      COLOR = "/proj/prosim/cor/asa.cor"
      ;The type of the texture
      SHADE_TYPE = BASIC
      ;The actor generates shadows
      SHADOW = TRUE
   ;Second actor
   A1:
      BRP = "/proj/prosim/brp/cj_base.brp"
      COLOR = "/proj/prosim/cor/asa.cor"
      SHADE_TYPE = BASIC
      SHADOW = TRUE
   ;(...)
   ;Other actors are defined similarly
END

GROUPS
   ;Actors are grouped to move together
   G0:
      COMPONENTS(A0, A1, A2, A3, A4, A5,
                 A6, A7 )
      ;Gravity center
      GC = 51, 43, -31
      ;Movements
      Between(FRAME(1), FRAME(120))
       rotate_actor_actor (ABSOLUTE, A8,
                           A9, 360) TOTAL
      Between(FRAME(241), FRAME(360))
       rotate_actor_actor (ABSOLUTE, A8,
                           A9, -270) TOTAL
      Between(FRAME(481), FRAME(540))
       rotate_actor_actor (ABSOLUTE, A8,
                           A9, -300) TOTAL
   ;(...)
   ;Other groups are defined similarly
END

CAMERA
   ;Parameters
   OBS = 94.7, 58., -4.3
   COI = 55.6, 66.3, -4.4
   VUP = 0, 0, 1
   D = 1
   APERTURE = 15.000, 15.000
   ;Movements
   Between(FRAME(1), FRAME(60))
           spin_clock(66) TOTAL
   Between(FRAME(1), FRAME(60))
           go_forward(38) TOTAL
   Between(FRAME(1), FRAME(60))
           go_north(42.2) TOTAL
   ;Aiming actor A3
   Between(FRAME(61), FRAME(420))
           aim_actor(A3)
```

```
   ;New viewpoint
   At(FRAME(95)) set_obs(48.3, 28.3, -11)
   At(FRAME(95)) set_coi(49.1, 30.1, -11)
   ;There are other camera movements
   ;(...)
END

LIGHTS
   ;Background color
   BACK = 0, 0, 25700
   ;Light sources
   L0:
      TYPE = SPOT
      POSITION = 37, 20, -33.5
      DIRECTION = 14, 23.2, 16.5
      COLOR = 65535, 65535, 65535
      SOLID_ANGLE = 150
      SPOT_TYPE = SOFT
   L1:
      TYPE = SPOT
      POSITION = 68, 10, 27
      DIRECTION = -29, 33.2, -44
      COLOR = 65535, 65535, 65535
      SOLID_ANGLE = 180
      SPOT_TYPE = SOFT
   ;(...)
END

RENDER
   ;Render interval
   FROM FRAME(0)
   TO FRAME(900)
   ;Render all the frames
   ONE_FR_IN 1
END

OUTPUT
   ;Path for the output images
   OUTPUT_PATH = "/proj/prosim/imgs/space"
   ;Output: MPEG animation
   QUALITY = SIPPMPEG
   ;Phong Shading
   SIPP_SHADE = PHONG_SIPP
   ;Shadows will be generated
   SIPP_SHADOW = TRUE
   ;Oversampling
   SIPP_SAMPLE = 1
   ;Dimensions of the generated frames
   VIEWPORT = 320, 280
END
```

This animation was produced in order to demonstrate some of the resources offered by TOOKIMA 2.0, especially those related to movement control, e.g. grouping of actors (module GROUPS), and definition of compound movements (movement of the camera, rotation and translation of a group of actors).
The animation showed in this section can be accessed in MPEG format at http://www.dca.fee.unicamp.br /projects/prosim/galeryPS.html at "Animations" (spacecraft).

6 Concluding Remarks

TOOKIMA 2.0 has been developed to improve robustness and usability of the animation production module of ProSIm. As a result, the global environment has now gained both a friendlier user interface and a scripting language of its own. The interface has been so designed as to lead users into progressively deeper understanding and familiarity with programming in TOOKIMA 2.0's embedded language. Eventually, users are expected to be able to generate scripts directly, if they so wish.
TOOKIMA 2.0 is being currently used by members of the ProSIm project group, including artists. They can create general-purpose animations of the kind illustrated in Section 5. Other animations generated by TOOKIMA2.0 can be found on the Web (http://www.dca.fee.unicamp.br/ projects/ prosim/ galeryPS.html).
Further improvements on the tool are currently being studied. Among them are extensions that accommodate adaptations to other geometric modellers and renderers, as well as other animation techniques (such as dynamics, inverse kinematics, and the like). Efforts are being made in the field of distributed and parallel computation, in order to improve rendering processes.

Acknowledgements. The authors wish to thank the following institutions for the expressive support granted to this research: UNICAMP (Universidade Estadual de Campinas), FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo) and CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico). Thanks also to all the members of the ProSIm project, specially to Dr. Wu Shin-Ting and Marcelo Malheiros.

References

Adler P, Winograd T (1993) Usability: Turning Technology into Tools. Oxford University Press

Armstrong WW, Green MW (1985) The dynamics of articulated rigid bodies for purposes of animation. The Visual Computer 1(4): 231-240

Barzel R, Barr AH (1988) A Modeling System Based on Dynamic Constraints. Computer Graphics

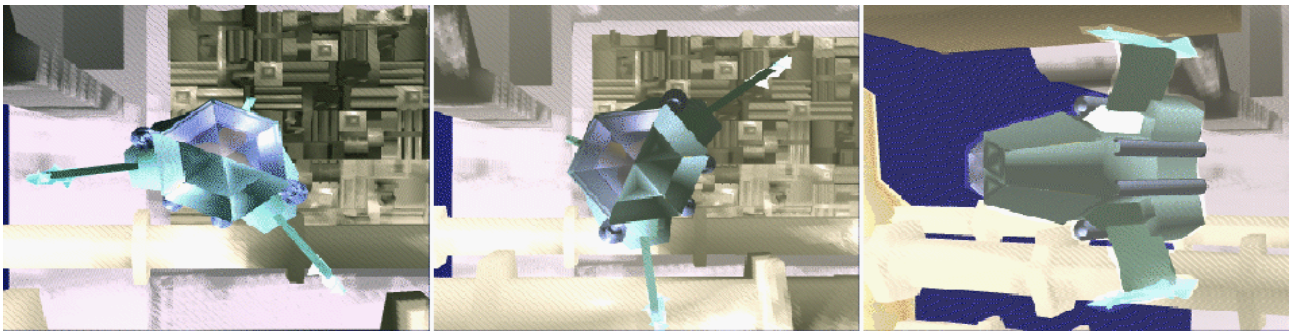

Figure 5: Frames of an animation created with the TOOKIMA 2.0.

22(4): 179-188

Booth KS, Kochanek DH (1983) Computers animate films and video. IEEE Spectrum 20(2): 44-51

Bradley J (1994) XV - Interactive Image Display for the X Window System - Version 3.10. User's Manual. (http://is.rice.edu/~shel/xv-3.10a/)

Camargo JTF, Magalhães LP, Raposo AB (1994) Modeling motion simulation with DEDS. Proc. of 13^{th} IFIP World Computer Congress-94: 162-167 (http://www.dca.fee.unicamp.br/projects/prosim/publiPS.html)

Camargo JTF, Magalhães LP, Raposo AB (1995) Fundamentos da animação modelada por computador (Fundaments of computer modeled animation). Tutorial presented at VIII SIBGRAPI (Brazilian Symposium on Computer Graphics and Image Processing). In Portuguese. (http://www.dca.fee.unicamp.br/projects/prosim/publiPS.html)

Costa M, Feijó B (1996) Agents with Emotions in Behavioral Animation. Computer & Graphics 20(3): 377-384

Dertouzos M (1992) The user interface is the language. In Myers B (ed) Languages for Developing User Interfaces. Jones and Bartlett, pp 21-30

Eisenberg M (1995) Programmable Applications: Interpreter meets Interface. SIGCHI Bulletin 27(2): 68-93

Gentner D, Nielsen J (1996) The Anti-Mac Interface. Communications of the ACM 39(8): 70-82

Gong KL (1994) Berkeley MPEG-1 Video Encoder - User's Guide. Computer Science Division, University of California, Berkeley, CA

Heller D (1990) XView Programming Manual - X Window System Series - Vol. Seven. O'Reilly & Associates

Hounsell MS, Magalhães LP (1992) TOOKIMA - animação cinemática (TOOKIMA - kinematics animation). Proc. of V SIBGRAPI (Brazilian Symposium on Computer Graphics and Image Processing): 269-273. In Portuguese

Koga Y, Kondo K et alli (1994) Planning Motions with Intentions. Proc. of SIGGRAPH 94: 395-408

Lasseter J (1987) Principles of Traditional Animation Applied to 3D Computer Animation. Computer Graphics 21(4): 35-44

Linton MA, Vlissides JM, Calder PR (1989) Composing User Interfaces with Interviews. IEEE Computer 22(2): 8-22

Malheiros MG (1994) Manual da Interface Gráfica do Sistema ProSIm (Manual of the Graphical Interface of the ProSIm). Internal Report, DCA - FEEC - UNICAMP. In Portuguese

Pueyo X, Tost D (1988) A Survey of Computer Animation. Computer Graphics Forum 7(4): 281-300

Raposo AB (1995) Uma Linguagem para Desenvolvimento de Roteiros de Animação (A Language for the Development of Animation Scripts). Internal Report, 002/95 - DCA - FEEC - UNICAMP. In Portuguese (http://www.dca.fee.unicamp.br/projects/prosim/publiPS.html)

Raposo AB (1996) Um Sistema Interativo de Animação no Contexto ProSIm (An Interactive Animation System in the Context of the ProSIm). M.Sc. Thesis, DCA - FEEC - UNICAMP. In Portuguese (http://www.dca.fee.unicamp.br/projects/prosim/publiPS.html)

Rose C, Guenter M, Bodenheimer B, Cohen MF (1996) Efficient Generation of Transitions using Spacetime Constraints. Proc. of SIGGRAPH 96: 147-154

Shneiderman B (1992) Designing the User Interface: Strategies for Effective Human-Computer Interaction. 2^{nd} Ed. Addison-Wesley

de Souza CS (1996) The Semiotic Engineering of Concreteness and Abstractness: From User Interface Languages to End User Programming

Languages. Dagstuhl Seminar on Informatics and Semiotics, Schloss Dagstuhl

Thalmann NM, Thalmann D (1985) Computer Animation - Theory and Practice. Springer-Verlag

Thalmann NM, Thalmann D (1991) Complex Models for Animating Synthetic Actors. IEEE Computer Graphics and Applications 11(5): 32-44

Upstill S (1989) The Renderman Companion: A Programmer's Guide to Realistic Computer Graphics. Addison-Wesley

Wyvill B (1990) A Computer Animation Tutorial. In Rogers DF, and Earnshaw RA (ed) Computer Graphics Techniques: Theory and Practice. Springer-Verlag, pp 235-282

Yngvesson J, Wallin I (1994) User's Guide to SIPP - a 3D Rendering Library - Version 3.1 (http: // www.ai–lab.fh–furtwangen.de / ai-lab / Documentations/sipp/)

Young C et alli. (1996) Persistence of Vision Ray-Tracer (POV-Ray) - User's Documentation 3.0.10. POV-Ray Team (http://www.povray.org)

Zeltzer D (1985) Towards an integrated view of 3-D computer animation. The Visual Computer 1(4): 249-259

Alberto Barbosa Raposo is currently working towards a Ph.D. degree at the Dept. of Computer Engineering and Industrial Automation (DCA) in the School of Electrical and Computing Engineering (FEEC) - UNICAMP (State University of Campinas). He received the B.S. and M.Sc. in Electrical Engineering from the same School, in 1994 and 1996, respectively. His current research interests include computer animation, virtual reality and distributed systems.

Léo Pini Magalhães received the B.S. and M.Sc. in Electrical Engineering at the School of Electrical Engineering of Campinas - UNICAMP (State University of Campinas) in 1974 and 1977, respectively, and the Dr.-Ing. degree at the Informatics Department of the Technische Hochschule Darmstadt - Germany in 1981. Since 1977 he teaches at the School of Electrical and Computer Engineering of Campinas, University of Campinas, Brazil, where he is presently an associate professor. He has worked in the areas of adaptive control, database systems, and graphics. His current research interests are in animation, with emphasis in the control and interface aspects. Presently, Dr. Magalhães is spending a sabbatical year at the Dept. of Computer Science, Computer Graphics Lab., of the University of Waterloo. Dr. Magalhães is a member of ACM, Eurographics, and a senior member of IEEE.

Clarisse Sieckenius de Souza is an Associate Professor of the Informatics Department at Rio de Janeiro Catholic University (DI/PUC-Rio). In 1987 she received her PhD in Applied Linguistics and joined DI/PUC-Rio. She has since been doing research and teaching in the fields of Artificial Intelligence, Computational Linguistics, User Interface Language Design, and Computer Semiotics. Among her most recent topics in research is that of End-User Programming.

José Tarcísio Franco de Camargo received his B.S., M.Sc. and Ph.D. in Electrical Engineering at the School of Electrical Engineering of Campinas - UNICAMP (State University of Campinas) in 1989, 1992 and 1995, respectively. Currently, he is a Visitor Professor at the School of Electrical Engineering of Campinas, Brazil. His current research interest area is computer animation/simulation.

Fernando José Lamanna is a plastic artist graduated by the Arts Institute of the UNICAMP (State University of Campinas) in 1996. He currently works with Computer Animation in LCA - FEEC - UNICAMP. His research interest areas include screenplay and storyboard development, computer geometric modeling and motion control.