PHYSICALLY-BASED INTERACTIVE CAMERA MOTION CONTROL USING 3D INPUT DEVICES

Russell Turner, Francis Balaguer, Enrico Gobbetti, Daniel Thalmann

ABSTRACT

The newest three-dimensional input devices, together with high speed graphics workstations, make it possible to interactively specify virtual camera motions for animation in real time. In this paper, we describe how naturalistic interaction and realistic-looking motion can be achieved by using a physically-based model of the camera's behavior. Our approach is to create an abstract physical model of the camera, using the laws of classical mechanics, which is used to simulate the virtual camera motion in real time in response to force data from the various 3D input devices (e.g. the Spaceball, Polhemus and DataGlove). The behavior of the model is determined by several physical parameters such as mass, moment of inertia, and various friction coefficients which can all be varied interactively, and by constraints on the camera's degrees of freedom which can be simulated by setting certain friction parameters to very high values. This allows us to explore a continuous range of physically-based metaphors for controlling the camera motion. We present the results of experiments with several of these metaphors and contrast them with existing ones.

Keywords: 3D Interaction, Motion Control, Dynamics, Virtual Cameras

1. INTRODUCTION

Specifying virtual camera motion is an important problem in a number of different computer graphics areas. Animation, scientific visualization, CAD and virtual environments all make considerable use of virtual camera motion in a three-dimensional environment (Brooks et al (1986), Baum et al (1990), Magnenat-Thalmann and Thalmann (1986), Shinagawa et al (1990)). Computer animation in particular, was quick to take advantage of the visual impact (and ease of programming) of complicated camera motions through relatively static scenes. This was, and still is, the basis for much of the commercial computer animations produced. In scientific visualization, CAD, and virtual environments applications, virtual camera motion is often the most important form of three-dimensional interaction (Watson (1989)).

Now, with the existence of graphics workstations able to display complex scenes containing several thousands of polygons at interactive speed, and with the advent of such new interactive devices as the Spaceball, Polhemus 3Space, and DataGlove, it is possible to create applications based on a full 3D interaction metaphor in which the specification of the camera motion is given in real-time. For example, in a virtual environment application the camera becomes the virtual "eyeball" with which the user inspects the virtual reality; in architectural CAD applications, the user has the ability to walk through virtual buildings and inspect them from any angle; in computer animation systems, the animator can specify the camera motion for a scene interactively in real time; for scientific visualization, large multi-dimensional data sets can be inspected by walking through 3D projections.

In such systems, camera control is a key technique which must be as natural and intuitive as possible so that the user is no longer conscious of it and can concentrate on the application task. In fact, interactive camera control might be thought of as a critical component in establishing a new 3D user-interface metaphor which will do for the 3D workstation what the desktop metaphor did for the 2D workstation.

However, the relationship between device input and virtual camera motion is not as straightforward as one might think. Usually, some sort of mathematical function or "filter" has to be placed between the raw 3D input device data and the virtual camera viewing parameters. This filter is usually associated with some sort of real-world metaphor such as "flying vehicle" metaphor, or "eyeball-in-hand" metaphor (Ware and Osborne 1990). Several recent papers have proposed and compared different metaphors for virtual camera motion control in virtual environments using input devices with six degrees of freedom (Ware and Osborne 1990, Mackinlay et al 1990). These metaphors are usually based on a kinematic model of control, where the virtual camera position, orientation, or velocity is set as a direct function of an input device coordinate.

One way to obtain more appealing motions in computer animation is to shift from a kinematic to a dynamic model. Recent publications describe how very naturalistic motions for animation can been obtained using physics-based models of rigid and flexible bodies (Terzopoulos and Platt (1989), Hahn (1988)). Although these models are usually extremely CPU-intensive, it is possible to use certain simplified models which can be calculated in real-time for 3D interaction. This can lead not only to improved interaction techniques, but may also shed light on the intuitiveness of physically based models. One such physical model that we propose is an interactive camera control metaphor based on physical modeling of the virtual camera, using forward dynamics for motion specification. This model is motivated by several hypotheses:

- the interaction is natural because humans are used to physical "Newtonian" behavior
- the movements have a natural look for the same reason
- it is a general parametric model so that a continuously variable set of behaviors can be obtained by varying the parameters
- the parameters have physical meaning and are easy to understand

In this paper we present a physical description and mathematical derivation of the physical camera model. We then give our subjective impressions of the interactive "look and feel" of the model with different parameter settings using the Spaceball as an input device. We then give some examples of how this model could be used to aid interactive camera motion in a specific domain. Finally, a description of the numerical technique and implementation on a Silicon Graphics Iris workstation is given.

2. THE PHYSICAL MODEL

Like all physically-based modelling in computer graphics, the physical camera model is motivated by the assumption that human beings are best-equipped to deal with environments that resemble the natural world. In our case, since we are using video display terminals for visual input, the "natural world" is, we propose, the world of film and video imagery. The natural virtual camera model is therefore most appropriately a real movie camera held by hand or mounted on some type of device for controlling camera motion.

Although there exist a variety of machines and vehicles used by the film industry for controlling camera movement, and it would obviously be possible to physically model all of these, the large number of camera movements in the "real" film/video world are created by one or more cameramen moving a mounted camera by hand. Therefore, we have chosen to model an idealized real-world camera which is manipulated by a human being who exerts forces and torques on it. In this case, the behavior of the camera--the virtual camera metaphor--is determined by the mechanical properties of the camera. These properties can then be considered the parameters of our parametric virtual camera model.

2.1 The Parametric Camera Model

Although real cameras and mounts are complicated mechanical devices, their motions are determined for the most part by a few simple gross physical properties. Therefore, we have constructed an idealized physical model consisting of a single rigid body attached to a massless camera mount. The camera mount consists of three gimbals and three rails, one in each of the x, y, and z local coordinates, resulting in three Cartesian and three rotational degrees of freedom. Each of the gimbals and rails exerts friction and elastic forces on the camera.

The important mechanical properties of this model which affect its motion are its mass, its moments of inertia, and the coefficients of friction and elastic forces imposed by the camera mount. The mass parameter specifies the amount by which the various forces will change the camera's linear velocity over time. The moment of inertia parameters affect, in an analogous way, the camera's response to the various torques. Usually a real camera's degrees of freedom are constrained in some geometrical way. For example, it may be placed on a dolly or a railway or tripod, constraining its linear motion to two, one and zero dimensions respectively. Likewise, the angular degrees of freedom may be restricted in one, two, or three axes by locking the gimbal bearings on the camera mount. Friction forces tend to reduce the linear and angular velocity of the camera over time and to oppose the applied forces and torques. There are several types of phenomenological friction forces used by physicists to model the dissipation of kinetic energy. We have found two, viscous friction and static friction, to be useful in our model. Viscous friction is proportional and opposite to the direction of motion, bringing the camera eventually to rest. Static friction is a constant force opposing the applied force, active only when the camera is at rest or below a threshold velocity. Although real camera mounts usually try to minimize vibrations, we have found it useful to add an elastic parameter, in the form of a Hookian spring, to each degree of freedom. This elasticity is only important when the camera is at rest (i.e. static friction is active) and results in a more naturalistic transition from the static state to the dynamic state and back.

2.2 Motion Under External Driving Forces and Torques

Fortunately, the simplified camera model is quite easy to analyze physically and, from the point of view of classical mechanics, is a well understood problem (Feynman et al (1963)). The general motion of a rigid body such as a camera can be decomposed into a linear motion of its center of mass under the control of an external net force and a rotational motion about the center of mass under the control of an external net torque.

Fig 1: The virtual camera driven by a force and a torque

2.2.1 Linear Motion

The linear motion under a total net force **F** can be computed by solving the differential equation

$$\ddot{\mathbf{X}} = \frac{\mathbf{F}}{\mathrm{m}} \qquad (1)$$

where m is the mass of the camera and **X** the position vector of its center of mass.

To compute the total net force **F** we have to take into account the driving force $\mathbf{F_d}$ and the forces that are generated by all the various frictions $\mathbf{F_f}$.

When the camera is moving at a speed below a given threshold velocity $\mathbf{v_s}$ and the driving force is smaller than a specified limit $\mathbf{F_s}$, we consider that the camera is in a static situation and that the frictions are caused by the springs that are present in the camera mount. Therefore we have

$$\mathbf{F_f} = k_{vs}\dot{\mathbf{X}} + k_{ss}(\mathbf{X} - \mathbf{X_o}) \tag{2}$$

where $\mathbf{X_0}$ is the position where the camera first entered the static situation, k_{vs} is the damping factor of the springs and k_{ss} the spring constant.

When we are at speeds higher than $\mathbf{v_s}$ or the driving force is bigger than $\mathbf{F_s}$, we consider the dynamic situation, where the frictions are mainly viscous. In that case we have

$$\mathbf{F_f} = k_{vd}\dot{\mathbf{X}} \tag{3}$$

where k_{vd} is the viscous friction coefficient.

It is useful to control the behavior of the camera separately for each of its principal axes. For this reason, the equations will be solved by projecting them onto the body-fixed reference frame, and a different value of the friction parameters will be specified for each one of the local axes.

2.2.2 Angular Motion

The rotational dynamics is expressed in a body-fixed reference system by the equation

$$\mathbf{I}\ddot{\theta} + \dot{\theta} \times \left(\mathbf{I}\dot{\theta}\right) = \mathbf{T_d} \tag{4}$$

where θ is the orientation of the camera expressed using Euler angles, $\mathbf{I}$ is the moment of inertia tensor of the body and is constant in the body-fixed reference frame, and $\mathbf{T_d}$ is the total net torque applied to the camera.

The modeling of the friction is analogous to the translational case. We can define ω_s to represent the threshold angular velocity and $\mathbf{T_s}$ to represent the threshold torque. In a static situation the friction torque $\mathbf{T_f}$ is given by

$$\mathbf{T_f} = k_{vs\text{-}rot}\dot{\theta} + k_{ss\text{-}rot}\left(\theta - \theta_0\right) \tag{5}$$

where θ_0 is the orientation where the camera first entered the static situation, $k_{vs\text{-}rot}$ is the damping factor of the springs and $k_{ss\text{-}rot}$ the spring constant.

In the dynamic case we have

$$\mathbf{T_f} = k_{vd\text{-}rot}\dot{\theta} \tag{6}$$

where $k_{vd\text{-}rot}$ is the viscous friction coefficient.

As for the translational case, it is useful to control the behavior of the camera separately for each of its principal axes, and a different value of the friction parameters will be specified for each one of the local axes.

3. THE PARAMETERS

The behavior of the virtual camera in response to user actions is completely specified by the parameters of the physical model of the camera. A different value of the friction parameters can be specified for every dimension in the reference frame attached to the camera. By this means we provide an easy and intuitive way for controlling the behavior of the model separately for each of the six degrees of freedom of the camera and several interesting camera motions or "camera metaphors" can be specified.

3.1 Mass and Inertia Tensor

The camera mass parameter determines the degree of acceleration control by the user. If all other parameters are set to zero, the camera maintains a steady velocity and changes its velocity only in response to user input, resulting in a pure acceleration control metaphor. Higher mass results in a smoother, more continuous motions and a higher degree of acceleration control, although it also makes it more difficult to bring the camera to rest at a given location. A high mass parameter is useful in situations where a smooth camera motion is desired, or where continuous motion is wanted with minimal input from the user, for example, tasks such as surveying a large scene or moving along a straight path.

In an analogous way, the inertia tensor determines the degree of torque control by the user over the camera orientation. A large inertia tensor results in smooth panning and tilting motions. This is often desirable because jerky camera rotation can be disorienting. Without a correspondingly high rotational friction parameter, however, it can be difficult to stop the camera from rotating. This is usually much more disorienting than the analogous translational situation. High inertia tensors are useful for the same sorts of tasks that high mass parameters are useful for: slow steady examination of a large scene. For more precise control up close to objects, a lower inertia tensor is preferable.

3.2 Viscous Friction Coefficient

The viscous friction parameter specifies the degree of velocity control. If the other parameters are set to zero, the camera metaphor becomes a pure velocity control one. Velocity control is useful for tasks where quick stopping and changes of direction are necessary, such as avoiding obstacles or inspecting objects up close. A typical application where a high viscous friction coefficient would be useful is a three-dimensional modeler. One problem with high viscous friction camera metaphors is that the camera motions are not usually very smooth and the user must give continuous input while the camera is moving. For many interactive tasks, this is not a problem. For other tasks, for example an architecture walk-through application, a suitable combination of velocity and acceleration metaphors can be formed by adjusting various amounts of the mass and viscous friction parameters.

The rotational analog to the viscous friction coefficient is the rotational viscous friction coefficient, which determines the amount of angular velocity control. Angular velocity control is particularly useful because of the need to stop camera rotation quickly. The balance between the moment of inertia and the rotational viscous friction coefficient determines the smoothness of camera panning motions, and many real-world camera mounts have adjustable angular viscous friction controls.

3.3 Threshold Force and Velocity

A small static friction parameter establishes a threshold force below which the camera will stay stationary. A small threshold velocity provides a breaking force that brings the camera to rest more rapidly than the viscous friction forces. This can be useful for tasks in which the user wants to alternate between motion along different degrees of freedom. For example, moving the camera along only one axis at a time is easy as long as the threshold force is not exceeded in the other axes. Likewise, a task that involves hopping to a fixed location, looking around in different directions without moving, then hopping to the next location, is much easier with a small amount of static friction.

By setting the static friction parameter extremely high, its possible to, in effect, lock a particular degree of freedom, resulting in a constrained motion. For example, by locking one of the rotational degrees of freedom, the camera is forced to always maintain the same "up" direction, and by locking one of the translational degrees of freedom, the camera is forced to move in a plane. This could be useful in an architectural walk-through application to simulate a walking person's point of view. Locking the vertical and horizontal translational degrees of freedom results in a flying-vehicle camera metaphor.

3.4 Static Behavior: Damping Factor and Spring Constant

The spring constant and damping factor parameters control the vibrational behavior of the camera mount when it is in the static state. A small amount of damped vibration smooths out the jerkiness in the transition between dynamic and static states. It also provides a small degree of position control feedback while the camera is in the static state. In this way, a small applied force will move the camera slightly, but it will pop back to its rest position. A larger force, above the static friction threshold, will set the camera in motion. This allows the user to get an idea of what direction an applied force will act in before actually moving the camera's position. If the static friction parameter is set extremely high, then the camera becomes locked in the static state and a position control camera metaphor results. Finally, if the damping factor is set low, a genuinely bouncy camera motion can be created. This can be used to simulate a hand-held camera motion.

4. CONTINUOUS VARIATION BETWEEN DIFFERENT BEHAVIORS

It is generally accepted that no one particular type of camera motion or camera control is appropriate for all tasks (Ware and Osborne 1990). For example, in a scene editor application, the user might require acceleration control for moving rapidly and smoothly as he surveys the overall organization of the scene, while the task of inspecting and moving an individual object in detail, where the camera viewpoint is close to the object, might require more precise velocity control.

Unfortunately, genuinely useful tasks often require a combination of metaphors or a sequence of alternating metaphors. A common way to deal with this is to allow the user to continually change camera control metaphors by swapping through the different interaction modes. Modes can be changed by selecting menus, striking keys or mouse buttons. There are two problems with this technique. First it is obviously inconvenient and unnatural to continually change modes. The user has to stop performing his task to swap computer and mental modes. After a while, this can be distracting and will often inhibit the user from changing metaphors until the current one becomes really impracticable. Secondly, it is not necessarily true that there are only a finite number of discrete useful metaphors. There are, in fact, a lot of inbetween situations where no pure metaphor suits the task best (Mackinlay et al 1990).

The parameterized physical camera model provides a solution to this problem by giving us a way to control the camera behavior through its parameter values. For a given task, the user can experiment with the camera metaphor and tune the parameters interactively, through valuators, until a subjectively "best" set of parameters is found. These parameters can then be saved and restored, either automatically or by the user, whenever that particular task is encountered. Alternately, the user can interactively control certain camera parameters while he is performing his task. For example, a mouse button or foot pedal can be set to momentarily increase the amount of viscous friction, acting as a kind of break.

Ideally, however, the user should only be required to use a minimum of input and should not have to be overly aware of the camera metaphor at all. This can be achieved by having the application adjust the camera control parameters algorithmically as a function of position or some aspect of the task at hand. This is potentially the most powerful use of the parametric camera model.

For example, we have experimented with creating a scalar viscosity field within a scene such that the camera viscous friction parameter increases in the vicinity of objects. This results in a camera metaphor that continuously varies from mainly acceleration control when the camera is far away from objects, to mainly velocity control when the camera is close to an object. In this way, the camera's behavior varies as a function of its distance from to objects.

Fig 2: Exploring a scene

Fig. 3: Inspecting an object

5. USE OF 3D DEVICES

Our dynamic camera control system is implemented on a Silicon Graphics Iris workstation in C using an object-oriented style of programming, on top of the Fifth Dimension 3D interaction toolkit (Turner et al 1990). In addition to the typical sorts of 3D classes, such as lights, hierarchical models and cameras, the toolkit abstracts every input device as an instance of an input device class. These input device objects communicate their data through a uniform event message protocol.

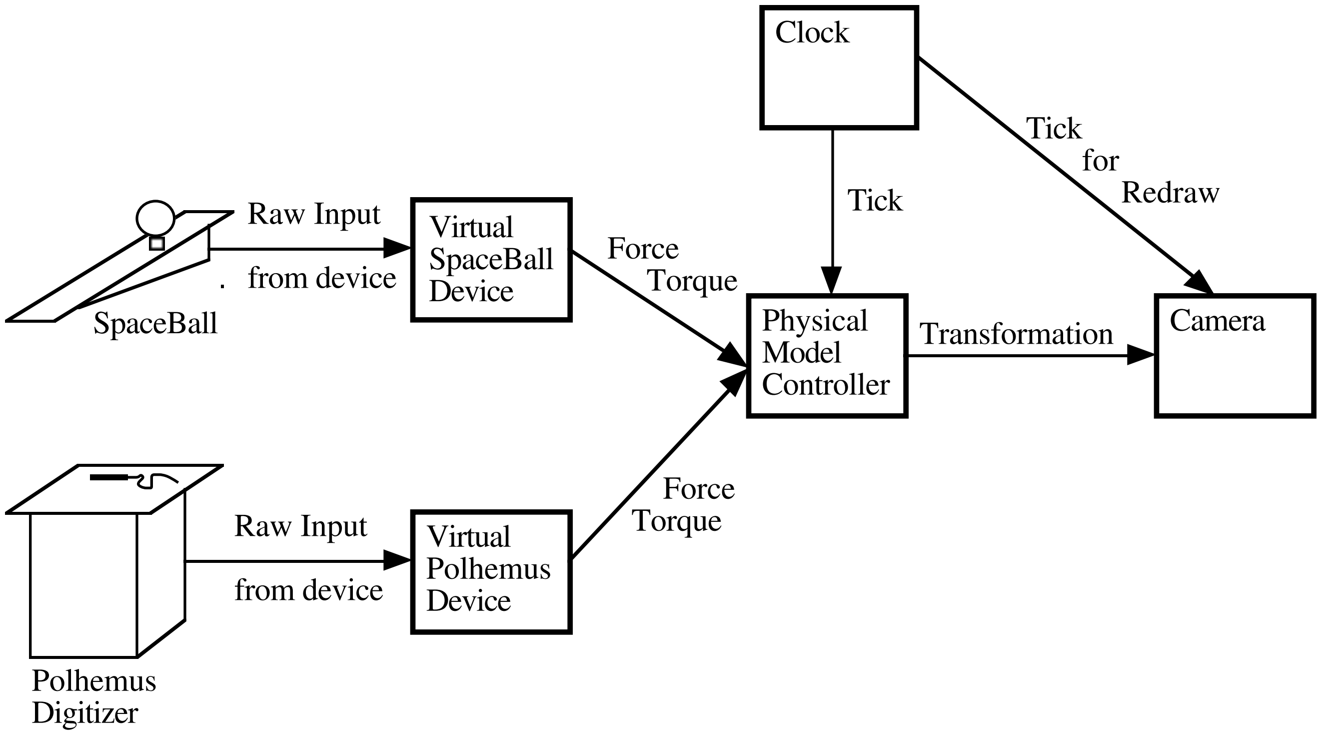

Fig. 4: Event communication diagram for camera controller

The interface between the Input Device objects and the Camera object is implemented by a Camera Controller object. This object receives events from input devices, interprets the data according to a particular camera metaphor, and updates the camera object's position and orientation accordingly. Since our Camera Controller involves a dynamic simulation, it also receives tick events from a Clock object at regular time-step intervals.

One of the advantages of this kind of software architecture is that various other devices, such as the Polhemus 3D, can be interchanged with the Spaceball as input to the Camera Controller. The use of this kind of devices is essential for providing an intuitive way of controlling the dynamic camera. Obviously, pressure-sensitive input devices are usually more appropriate because they provide a passive form of "force-feedback". In our case, the device that gave the best results is the Spaceball. Also, different types of Camera controllers with different behaviors can be swapped in and out, and the same controllers can be used to control Light objects or hierarchical models.

6. CONCLUSIONS AND FUTURE WORK

We believe that the physically-based camera control model provides a powerful, general-purpose metaphor for controlling virtual cameras in interactive 3D environments. Because it is based on a real camera model, it is natural for the user to control. Its parameters are physically-based and, therefore, easy to understand and intuitive for the user to manipulate. Its generality and control parameters make it configurable to emulate a continuum of camera behaviors ranging from pure position control to pure acceleration control. As it is fully described by its physical parameters, it is possible to construct more sophisticated virtual camera control metaphors by varying the parameters as a function of space, time, application data or other user input. Also, when used with force-calibrated input devices, the camera metaphor can be reproduced exactly on different hardware and software platforms, providing a predictable standard interactive "feel".

We are currently working on extending our model to specify camera paths for computer animation. Currently, the interactive camera metaphor permits the generation of position, orientation and acceleration information simultaneously along a path. For animation purposes, it is possible to record this camera motion and then edit it interactively by replaying and selectively re-recording parts of the input data. For example, on the first pass we might be happy with the recorded position path, but not with the camera orientation. Therefore, on the second pass we can re-record only the orientation information.

We are also continuing to explore different kinds of algorithmic control of the camera parameters by creating various types of parameter fields within the scene space, and we are planning to use the dynamic model approach for other kinds of interactive tasks such as modeling, assembling, and specifying the animation of different kinds of objects.

ACKNOWLEDGEMENTS

The authors are grateful to Angelo Mangili for his technical help.
The research was partly sponsored by "Le Fond National pour la Recherche Scientifique".

REFERENCES

Baum R., Wingel J.W (1990) Real Time Radiosity Through Parallel Processing and Hardware Acceleration *Proceedings 1990 Workshop on Interactive 3D Graphics* ACM: 67-75

Brooks F.P. Jr (1986) Walkthrough - A Dynamic Graphics System for Simulating Virtual Buildings *Proceedings 1986 Workshop on Interactive 3D Graphics* ACM : 9-22

Feynman R.P., Leighton R.B., Sands M. (1963) The Feynman Lectures on Physics Addison-Wesley Reading, Massachusetts.

Hahn J.K. (1988) Realistic Animation of Rigid Bodies *Computer Graphics* 22(4) : 299-308

Mackinlay J.D., Card S.K, Robertson G. (1990) Rapid Controlled Movement Through a Virtual 3D Workspace, *Computer Graphics* 24(4) : 171-176

Magnenat-Thalmann N., Thalmann D. (1986) Special Cinematographic Effects Using Multiple Virtual Movie Cameras, *IEEE Computer Graphics & Applications* 6(4): 43-50

Shinogawa Y., Kunii T.L., Nomura Y., Okuno T., Young Y. (1990) Automating View Function Generation for Walkthrough Animation Using a Reeb Graph, *Proceedings Computer Animation 90* Springer, Tokyo: 227-238

Terzopoulos D., Platt J. (1989) Physically-Based Modeling: Past, Present and Future *SIGGRAPH 1989 Panel Proceedings* ACM : 191-209

Turner R., Gobbetti E., Balaguer F., Mangili A., Thalmann D., Magnenat-Thalmann N. (1990) An Object Oriented Methodology Using Dynamic Variables for Animation and Scientific Visualization *Proceedings Computer Graphics International 90* Springer-Verlag : 317-328

Ware C., Osborne S. (1990) Exploration and Virtual Camera Control in Virtual Three Dimensional Environments *Proceedings 1990 Workshop on Interactive 3D Graphics* ACM : 175-183

Watson V. (1989) A Breakthrough for Experiencing and Understanding Simulated Physics *ACM SIGGRAPH Course Notes on* State of the Art in Data Visualization, IV-26 - IV-32

APPENDICES

A. MOMENT OF INERTIA TENSOR

The rotational equivalent of the mass is the moment of inertia tensor **I**. It can be represented by a three-dimensional symmetric matrix whose elements are given by (Feynman, 1963):

$$\mathbf{I}_{xx} = \int \left(y^2 + z^2\right) dm$$

$$\mathbf{I}_{yy} = \int \left(x^2 + z^2\right) dm$$

$$\mathbf{I}_{zz} = \int \left(x^2 + y^2\right) dm$$

$$\mathbf{I}_{xy} = \mathbf{I}_{yx} = -\int xy \, dm$$

$$\mathbf{I}_{yz} = \mathbf{I}_{zy} = -\int yz \, dm$$

$$\mathbf{I}_{xz} = \mathbf{I}_{zx} = -\int xz \, dm \qquad \text{(A1)}$$

The diagonal elements of this matrix represent the moment of inertia of the body, and the non-diagonal elements represent the products of inertia of the axes. When the reference frame where the moment of inertia is specified corresponds with the principal inertia frame, all the products of inertia are null, and **I** becomes a diagonal matrix. In such conditions the equation of rotational motion (number (4) in the text) simplifies to the well known Euler equations (Feynman, 1963):

$$\begin{aligned}
\mathbf{I}_{xx}\ddot{\theta}_x + (\mathbf{I}_{zz} - \mathbf{I}_{yy})\dot{\theta}_z\dot{\theta}_y &= \mathbf{T}_x \\
\mathbf{I}_{yy}\ddot{\theta}_y + (\mathbf{I}_{xx} - \mathbf{I}_{zz})\dot{\theta}_x\dot{\theta}_z &= \mathbf{T}_y \\
\mathbf{I}_{zz}\ddot{\theta}_z + (\mathbf{I}_{yy} - \mathbf{I}_{xx})\dot{\theta}_y\dot{\theta}_x &= \mathbf{T}_z
\end{aligned} \qquad \text{(A2)}$$

We approximate the camera model with a rectangular box of homogeneous distribution with dimensions $\mathbf{d}_x$, $\mathbf{d}_y$, and $\mathbf{d}_z$ to obtain the values of the moments of inertia:

$$\begin{aligned}
\mathbf{I}_{xx} &= \frac{m}{12}\left(\mathbf{d}_y^{\,2} + \mathbf{d}_z^{\,2}\right) \\
\mathbf{I}_{yy} &= \frac{m}{12}\left(\mathbf{d}_x^{\,2} + \mathbf{d}_z^{\,2}\right) \\
\mathbf{I}_{zz} &= \frac{m}{12}\left(\mathbf{d}_x^{\,2} + \mathbf{d}_y^{\,2}\right)
\end{aligned} \qquad \text{(A3)}$$

If we consider that the box has nearly the same dimension on the three axes, then equation (A2) can be linearized.

B. NUMERICAL INTEGRATION OF THE EQUATIONS OF MOTION

To simulate the behavior of our virtual camera, the equations describing its motion in response to the external driving forces and torques have to be integrated through time. If we project these equations on the local axis, we obtain six ordinary scalar differential equations of the second order, under the assumption that equations A2 can be linearized by considering their second term piecewise constant. These equations are of the form:

$$a\ddot{x} + b\dot{x} + cx + d = f(t) \tag{B1}$$

Although an analytical solution exists for many of the forms of this equation, the driving function in this case is not analytic but rather strictly data-driven, being the instantaneous user force or torque input over time. Therefore we must use a numerical method to find the solution.

To solve the equation, time is subdivided into equal time steps Δt. We use the second order accurate approximations

$$\begin{aligned} \ddot{x}_t &= \frac{1}{\Delta t^2}\left(x_{t+\Delta t} - 2x_t + x_{t-\Delta t}\right) \\ \dot{x}_t &= \frac{1}{2\Delta t}\left(x_{t+\Delta t} - x_{t-\Delta t}\right) \end{aligned} \tag{B2}$$

that we substitute into equation (1) to find the explicit integrator:

$$x_{t+\Delta t} = \frac{\Delta t^2 f_t - \Delta t^2 d - x_t\left(-2a + c\Delta t^2\right) - x_{t-\Delta t}\left(a - \frac{\Delta t}{2}b\right)}{a + \frac{\Delta t}{2}b} \tag{B3}$$

This explicit procedure evolves the dynamic solution from given initial conditions x_o and x_{-1}. The current and the previous value of x are used to solve for the value at a small time Δt later. No oversampling is necessary, because the precision required is not high (the user interacts with the solver), and Δt represents the time increment between displayed frames. Our current implementation allows us to have an interactive display rate (more than 10 Hz) on a Silicon Graphics Iris 4D/80 with fully shaded scenes containing up to two-thousands of polygons. The time spent for the dynamic computations is negligible with respect to the redraw time.

Russell Turner is a researcher at the Computer Graphics Laboratory of the Swiss Federal Institute of Technology in Lausanne, Switzerland. He received his B.S. in Physics and his M.S. in Computer and Information Science from the University of Massachusetts at Amherst. He has also worked as a software engineer for V.I. Corporation of Amherst, Massachusetts. His research interests include computer animation, physical modeling, user-interfaces and object-oriented programming. He is a member of IEEE and ACM.
E-mail: `turner@elma.epfl.CH`

Francis Balaguer is a researcher at the Computer Graphics Laboratory of the Swiss Federal Institute of Technology in Lausanne, Switzerland. He received his diplôme d'ingénieur informaticien from the Institut National des Sciences Appliquées (INSA) in Lyon, France. His research interests include 3D interaction, computer animation, user-interfaces, and object-oriented programming.
E-mail: `balaguer@ligsg2.epfl.CH`

Enrico Gobbetti is a researcher at the Computer Graphics Laboratory of the Swiss Federal Institute of Technology in Lausanne, Switzerland. He received his diplôme d'ingénieur informaticien from the same institute. His research interests include visualization, computer animation, human-computer interaction and object-oriented programming.
E-mail: `gobbetti@elma.epfl.CH`

Daniel Thalmann is currently full Professor and Director of the Computer Graphics Laboratory at the Swiss Federal Institute of Technology in Lausanne, Switzerland. Since 1977, he was Professor at the University of Montreal and codirector of the MIRALab research laboratory. He received his diploma in nuclear physics and Ph.D in Computer Science from the University of Geneva. He is coeditor-in-chief of the *Journal of Visualization and Computer Animation*, member of the editorial board of the *Visual Computer* and cochairs the EUROGRAPHICS Working Group on Computer Simulation and Animation. Daniel Thalmann's research interests include 3D computer animation, image synthesis, and scientific visualization. He has published more than 100 papers in these areas and is coauthor of several books including: *Computer Animation: Theory and Practice* and *Image Synthesis: Theory and Practice*. He is also codirector of several computer-generated films: *Dream Flight*, *Eglantine*, *Rendez-vous à Montréal*, *Galaxy Sweetheart*, *IAD, Flashback.*
E-mail: `thalmann@eldi.epfl.CH`

The authors may be contacted at:

Computer Graphics Laboratory
Swiss Federal Institute of Technology
CH-1015 Lausanne, Switzerland

Tel: 41.21.693.52.14
Fax: 41.21.693.39.09