CINEMA: A System for Procedural Camera Movements

Steven M. Drucker, Tinsley A. Galyean, and David Zeltzer

Computer Graphics and Animation Group
MIT Media Lab
Cambridge, MA. 02139
(smd | tag | dz)@media-lab.media.mit.edu

Abstract

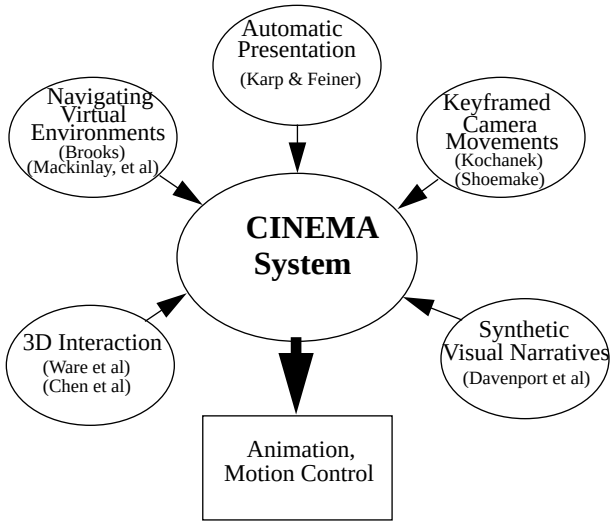
This paper presents a general system for camera movement upon which a wide variety of higher-level methods and applications can be built. In addition to the basic commands for camera placement, a key attribute of the CINEMA system is the ability to inquire information directly about the 3D world through which the camera is moving. With this information high-level procedures can be written that closely correspond to more natural camera specifications. Examples of some high-level procedures are presented. In addition, methods for overcoming deficiencies of this procedural approach are proposed.

1. Introduction

Camera control is an integral part of any 3D interface. In recent years a number of techniques for interactively specifying camera movement have been implemented or proposed. Each of these techniques has provided an interface for solving a problem for a particular domain, but all of them have remained independent making it impossible to use them across domains. These domains include keyframe based computer graphic animation techniques [8, 11], navigation of virtual environments [1, 2, 9, 12, 13], general 3D interaction [3, 12], automatic presentation [6] (in which computers generate a presentation), and synthetic visual narratives [4] (in which users author presentations). The CINEMA system described in this paper is a camera protocol that supports camera interface paradigms useful for all these domains, and provides a framework on which new interfaces can be developed.

The CINEMA system has a procedural interface for specifying camera movements relative to objects, events, and the general state of an environment. This task level approach enables the implementation of many common interactive metaphors and provides the ability to build higher level parameterized procedures that are reusable.

After a brief introduction to the problem, we will review related work in camera control, and then describe the CINEMA system, including the underlying support structure, the implementation, and several examples that demonstrate the system. Finally, we will discuss problems with this approach and suggest alternatives based upon our findings. We will work under the assumption that the actions in the environment are occurring independently from the observer. By making this assumption, the specification of the camera is independent from the 3D world, or can be treated as a window into the world that does not impact on it. This simplification is made by many of the existing camera interfaces reviewed in this paper, and although limiting, it is appropriate for a variety of situations.



An effective camera protocol must support interfaces that investigate/explore and interfaces that present/illustrate the 3D world. Although we have only begun to explore the uses of this system, there are many applications in which it could be used. In both scientific and architectural visualization there is the need to explore the virtual environment interactively and then to later author a set of illustrative camera movements to be shown to clients or colleagues. In electronic books there will be the need for a designer or knowledge based system to generate an interface through which a reader can view the information. In the entertainment industry an animator could use it to direct or specify camera movements. Live action film makers may use it to create interactive story boards of their scenes, plan camera movements, or even to generate commands for motion controlled cameras. Telerobotic or virtual environment applications require a task level camera

protocol in order to allow a human operator to efficiently and intuitively control the view while performing or directing some remote operation. All of these interfaces can be supported on top of the camera protocol described in this paper.

2. Previous Work

Early work in animation is devoted to making the movement of the camera continuous and to developing the proper representation for camera movements along a path [8, 11]. These works are devoted to giving the animator greater control in creating smooth movements and to finding ways to interpolate between user specified keyframes. Although generating spline curves for camera movement can produce smooth paths, it can be difficult to relate the movements of the camera to objects in the environment.

With the advent of virtual environments and related 3D interactive worlds, a great deal of effort has been spent on presenting convenient metaphors through which to change the user's view of an object or the world. A metaphor, as discussed in Ware et al [12] provides the user with a model that enables the prediction of system behavior given different kinds of input actions. A good metaphor is both appropriate and easy to learn. Some examples of metaphors are the 'eyeball in hand' metaphor, the 'scene in hand' or 'dollhouse' metaphor, and 'flying vehicle control.'

In the 'eyeball in hand' metaphor, a 6 degree of freedom device is used to position and orient a camera by directly translating and rotating the input device. Ware et al found this method somewhat awkward to use but easy to learn. The 'scene in hand' metaphor allows the user to rotate and translate the scene based on the position of the input device. This was found to be very convenient for hand sized objects, but nearly impossible to use for navigating inside closed spaces. Another scheme discussed by Ware et al was to control a simulated flying vehicle. The user's position and orientation respectively affected the linear and angular velocity of the camera viewpoint and direction of gaze. This metaphor makes it easy to navigate, but difficult to examine a particular object. Although 3D input devices such as a Polhemus Isotrack system or a Spatial Systems Spaceball enable the user to specify 6 degrees of freedom simultaneously, simulations of these devices can be done using only 2D devices [3].

Mackinlay et al [9] discuss the problem of scaling camera movements appropriately. They develop methods to select an object of interest and to move exponentially towards or away from the object. In this way, when the user is close to an object, the viewpoint changes only a little, while when they are far from an object, the viewpoint changes rapidly. By selecting 'point of interest,' the authors can reorient the camera to present a maximal view of the desired object. The degrees of freedom are therefore restricted and the user can concentrate more on the task of navigating through the environment.

Brooks [1, 2] developed several different methods for moving around architectural simulations including steerable treadmills or shopping carts with devices to measure the direction and speed of movement.

The above work shows that different interfaces are appropriate for different application requirements. In our view, no one interface is ideally suited for all tasks, and a common underlying structure on top of which several different metaphors can be implemented would give the user a powerful tool to interact with 3D environments.

An important ability is to allow the user to select an object of interest within the environment. We have expanded on this by allowing the user to make general queries about the visibility and orientation of objects within the environment. This allows the user to manipulate camera motion based on the actions within the environment.

Furthermore, while direct manipulation has certain advantages in interactive systems, there are several deficiencies. It is not necessarily good for repetitive actions, and any action that requires a great deal of accuracy, such as smooth movement for cameras, is not necessarily suited to input using one of the metaphors suggested in the preceding paragraphs. Some of the problems inherent in using 6 DOF input devices presently available are noise which is inherent in user movements and the number of degrees of freedom which must be simultaneously controlled. Textual systems, with interaction built on top of them, allow both a high level input device interface, and an underlying language through which commands can be specified directly or generated through other rule bases.

An expert system for presentation, including the selection of proper camera movements, is discussed in some detail by Karp and Feiner [6]. In their Esplanade system (Expert System for PLANning Animation Design and Editing), they emphasize the ability to incorporate cinematic knowledge for the construction of coherent descriptions of a scene. To do so, they need to have representations of a database of objects and explicit events, along with a notion of how frames, shots, scenes and sequences can be put together to make an effective narrative. Their work emphasizes using a knowledge based system for automatically selecting camera placement and for choosing appropriate camera movements based on cinematic considerations. Currently, they do not concentrate on the movements themselves, but more on the initial placement of the camera for shots and how to make transitions to other shots.

3. The CINEMA System

We have developed the CINEMA system to address the problem of combining different paradigms for controlling camera movements into one system. The CINEMA system is extensible, permitting the user to build higher level procedures from simpler primitives. It also provides the very important ability to make inquiries into a database which contains information describing the state of objects within a 3D environment. After the system was developed, it was used by a dozen students in a course entitled "Synthetic Cinematic and Cinematic Knowledge.[1]" In this course the students used this system to explore alternative ways of animating one of several scenes. Although this system has mainly been used for a synthetic narrative application, we feel that what was learned is applicable to the other domains such as the applications mentioned above.

The CINEMA system is divided up into two major parts. The first is a database which contains information about objects, their positions over time, and events over time. The second part is a parser that accepts and interprets user commands. The user commands are restricted to inquiries about the state of the database and commands which query or affect the state of the camera.

---

1 The course has been taught at the MIT Media Lab by Professors David Zeltzer and Glorianna Davenport – two short versions in January 1989 and 1990, and two full semester courses in the Spring of 1991 and 1992.

3.1 Support structure for CINEMA

To produce CINEMA's procedural interface it was necessary to develop a set of primitive functions. There are three parts to this support structure. First, there is a set of commands for moving the camera or inquiring about the current camera state. Second is a set of commands for inquiring about the state of the 3D world. Last is a set of mathematical routines for manipulating the values returned from the other functions.

There are two sets of primitive functions for changing the camera position and orientation. The lower level of these are the commands that directly set the *x*, *y*, and *z* positions and the *from*, *up*, and *at* vectors that are so commonly used in computer graphics. The slightly higher level primitives (but still part of the support structure) perform simple camera moves like *pan*, *tilt*, *roll*, *truck*, *dolly*, and *crane* [7]. In the film industry terms such as dolly and truck are loosely used. For example, truck may be used to mean a move in or out, or a move from side to side. In this implementation we have chosen one of the possible definitions for these terms to avoid confusion. The conversion between the computer graphics vectors and the film standards is straightforward.

Many descriptions of how to film, frame, and navigate the scene (by both screenwriter and layperson) are with respect to the objects in the world. For example one might ask for the camera to move alongside object A while looking at object B. An interface that supports these descriptions must provide information about events, geometric and spatial relationships such as position, relative occlusion, direction of glance, and distances. For example, functions like `obj_visibility(), obj_obj_visibility()` find visibility information between the camera and an object or between two particular objects. Currently this is implemented by using simple ray casting with bounding box intersection. More sophisticated techniques can be used to provide a more precise notion of visibility. However, this implementation has proven adequate for this preliminary research. Other functions (like `frame_events()`) are provided to support inquiry into discrete events which might take place during an animation.

In addition to the commands described above, the system provides a set of supporting mathematical commands, including both scalar and vector calculations. These commands are needed to manipulate the output of the inquiry commands. With these functions, an inquiry about the state of the scene can be manipulated to calculate new camera parameters (such as position, from, at and up vectors). With combinations of these basic tools higher level procedures can be built.

3.2 Implementation

The entire system is currently implemented on 2 platforms: an HP9000-835 turbo SRX in C using a public domain front end language called Tcl [10], and on an Apple Macintosh. The Macintosh platform can not provide interactive update rates for rendered images, but is successfully used for wireframe images.

3.3 Examples

The following examples are representative of how the CINEMA System was used in several different situations. The first example shows how the CINEMA system is interfaced to a 3D environment, and implements one of Ware et al's movement metaphors. The second example shows how higher level camera movements can be built from lower level primitives and inquiry functions. Finally, example 3 shows the cinematic power of the system in filming a simple animation.

Example 1: CAMERA MOVEMENT METAPHOR: This example shows how a 3D input device such as the Isotrack Polhemus or Spatial Systems Spaceball can be used to change the view in a scene. In the accompanying video, we use an Ascension Technologies Bird to control the *x*, *y*, and *z* position of the camera while always looking at the object called "joe." This is very similar to the "eyeball in hand" movement metaphor discussed by Ware et al.

The following pseudocode shows how this function is implemented using the CINEMA system. The function consists of an inquiry to the 6 DOF input device and then translating the camera based upon the translation returned by the input device.

```
proc eyeball_in_hand(object) {
 (x,y,z) := get_input_from_device();
 cam_set_point(x,y,z);
 lookat(object);
}
```

Example 2: EXTENSIBLE LANGUAGE: The procedure "vertigo shot" simulates Hitchcock's classic shot in the film "Vertigo" where the camera moves outwards while the field of view grows narrower keeping the object a constant size at the center of the frame. This effect makes viewers feel as if they are moving closer and closer to an unattainable goal. In only a few minutes we constructed the following procedure to make a vertigo shot.

```
proc vertigo_shot(obj, rate, no_frames) {
 /* get the angle subtended by the object */
 angle := get_angle_height(obj);
 /* get the camera's field of view */
 fov := cam_fov();
 /* compute the percentage of the fov */
 /* which the object subtends */
 percent := angle/fov;
 for (i=0;i<no_frames;i++) {
        /* truck in the specified direction */
        /*    at the specified rate */
        cam_truck(rate);
        /* set the field of view so that */
        /*    the object subtends the same */
        /*    percentage as before */
        frame_it(obj, percent);
 }
}
```

Example 3: SYNTHETIC NARRATIVE: The last example shows that the system can be used for simple cinematic teaching purposes. An animation of a figure sitting down is filmed. A cut in the middle of the animation changes the viewpoint from an oblique view to a head on view. The views are selected so that a "match" cut [5] is achieved. See sequence of frames at the end of the paper.

4. Future Work

The CINEMA system needs to be extended to provide a mechanism to easily combine and constrain multiple procedures. For example, suppose a user would like to track the motion of a walking figure while preventing the camera from moving through walls. Ideally, one would like to have these procedures (one for tracking and one for avoidance) automatically combined to

achieve the desired performance. Currently, it would be necessary to construct a new procedure meeting both constraints.

The ability to combine procedures would allow user input to be treated as another procedure that can be combined with other constraints. Camera movements could then be interactively adjusted to achieve a desired result.

To address some of these problems, we have already begun exploration into constraint satisfaction techniques for camera placement and movement. By specifying the camera's relationship to other objects via weighted constraints, the system can find the best position that satisfies certain criterion. These constraints are maintained as the objects, and the camera moves throughout the environment. Additional constraints can be placed on the movement of the camera, so that the camera can have attributes of a simulated physical object such as a fluid head.

5. Conclusion

The CINEMA system provides users with the ability to rapidly experiment with various camera movement paradigms. Users can create new camera metaphors or extend existing ones. The ability to inquire about the state of objects in the environment provides support for more powerful camera movement procedures.

The CINEMA system has already proven quite useful in the teaching domain. Students were able to use the CINEMA system to explore different ways to film and present a simple animation, and to plan a real camera shoot. The constraint satisfaction methodology described above is an ongoing area of research. There are many other areas to explore in camera movement systems including rule based generations systems, codifying stylistic attributes, examining cuts, and interfacing with task oriented applications to name just a few. The hope is that once a strong support base for camera positioning and movement is produced, further research in these areas will be easier.

The CINEMA system makes it possible to experiment with camera paradigms quickly and conveniently. We intend to continue evolving the CINEMA system with an eye toward different application domains including telerobotics/virtual environments and synthetic narratives.

Acknowledgements

The authors wish to thank Glorianna Davenport, David Sturman, Mike McKenna, Steve Pieper, and David Chen for their comments and assistance. Hong Tan was invaluable for her help with the implementation on the Macintosh. The students in the Synthetic Cinema course also provided many useful suggestions. This work was sponsored in part by NHK (Japan Broadcasting Corporation), DARPA/Rome Air Development Center Contract F30602-89-C-022, and equipment grants from Apple Computer and Hewlett-Packard.

References

1 Brooks, F.P. (1986). Walkthrough - A Dynamic Graphics System for Simulating Virtual Buildings. *Proceedings of 1986 Workshop on Interactive 3D Graphics* (Chapel Hill, North Carolina). In *Computer Graphics*, 9-21.

2 Brooks, F.P. (1988). Grasping Reality Through Illusion: Interactive Graphics Serving Science. *SIGCHI* '88 1-11.

3 Chen, M., S.J. Mountford, A. Sellen. (1988). A Study of Interactive 3D Rotation Using 2D Control Devices. *Proceedings of SIGGRAPH '88* (Atlanta, Georgia). In *Computer Graphics* 22(4):121-129.

4 Davenport, G., T.A.Smith., N.Pincever (1991). Cinematic Primitives for Multimedia. IEEE Computer Graphics & Applications. July. 67-74.

5 Hochberg, J. and V. Brook. (1978). The perception of motion pictures. in Handbook of Perception Vol X. eds. Carterette and Friedman. Academic Press: New York. Chapter 11.

6 Karp, P., S.Feiner. (1990). Issues in the Automated Generation of Animated Presentations. *Graphics Interface '90* (Halifax, Nova Scotia). 39-48.

7 Katz, E. (1979). *The Film Encyclopedia.* Perigree Books: New York.

8 Kochanek, D.H.U. (1984). Interpolating Splines with Local Tension, Continuity, and Bias Control. *Proceedings of SIGGRAPH '84* (Minneapolis, Minnesota). *Computer Graphics.*18(3):33-42.

9 Mackinlay, J.D., S. Card, G. Robertson. (1990). Rapid Controlled Movement Through a Virtual 3D Workspace. *Proceedings of SIGGRAPH '90* (Dallas, Texas). In *Computer Graphics* 24(4):171-176.

10 Ousterhout, J. (1990). Tcl: An Embeddable Command Language. Proceedings of USENIX Winter Conference. 133-146.

11 Shoemake, K. (1985). Animating Rotation with Quaternion Curves. *Proceedings of SIGGRAPH '*85 (San Francisco, CA). In *Computer Graphics* 19(3):245-254.

12 Ware, C., S. Osborne. (1990). Exploration and Virtual Camera Control in Virtual Three Dimensional Environments. *Proceedings of the 1990 Symposium on Interactive 3D Graphics* (Snowbird, Utah).In *Computer Graphics* 24(2):175-184.

13 Zeltzer, D. (1991). Task-level Graphical Simulation: Abstraction, Representation, and Control, *Making Them Move*, eds. Badler, Barskey and Zeltzer. Morgan Kaufmann Publishers: California. 3-33.