Centrum voor Wiskunde en Informatica
REPORT*RAPPORT*

Parametrizable Cameras for 3D Computational Steering

J.D. Mulder, J.J. van Wijk

Software Engineering (SEN)

SEN-R9717 August 31, 1997

Report SEN-R9717
ISSN 1386-369X

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

CWI is the National Research Institute for Mathematics and Computer Science. CWI is part of the Stichting Mathematisch Centrum (SMC), the Dutch foundation for promotion of mathematics and computer science and their applications.
SMC is sponsored by the Netherlands Organization for Scientific Research (NWO). CWI is a member of ERCIM, the European Research Consortium for Informatics and Mathematics.

Copyright © Stichting Mathematisch Centrum
P.O. Box 94079, 1090 GB Amsterdam (NL)
Kruislaan 413, 1098 SJ Amsterdam (NL)
Telephone +31 20 592 9333
Telefax +31 20 592 4199

Parametrizable Cameras for 3D Computational Steering

Jurriaan D. Mulder, Jarke J. van Wijk[1]
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

ABSTRACT

We present a method for the definition of multiple views in 3D interfaces for computational steering. The method uses the concept of a point-based parametrizable camera object. This concept enables a user to create and configure multiple views on his custom 3D interface in an intuitive graphical manner. Each view can be coupled to objects present in the interface, parametrized to (simulation) data, or adjusted through direct manipulation or user defined camera controls. Although our focus is on 3D interfaces for computational steering, we think that the concept is valuable for many other 3D graphics applications as well.

1991 Computing Reviews Classification System: I.3.2, I.3.4, I.6.6, I.6.7
Keywords and Phrases: scientific visualization, viewing, interactive 3D computer graphics.
Note: Presented at the Eighth Eurographics Workshop on Visualization in Scientific Computing, Boulogne sur Mer, France, 28-30 April, 1997.
Work carried out under CWI project SEN-1.3, Interactive Visualization Environments.

1. INTRODUCTION

Computational steering allows a researcher to change parameters of a running simulation and immediately receive feedback on the effect of these changes. As a result, the researcher can gain a much better insight in the behavior of the simulation, he can correct erroneous values for input, and he can steer the simulation into a desired direction. At CWI an environment for computational steering has been developed that allows a researcher to construct and use customized 2D and 3D user interfaces for his simulations. These interfaces consist of both the visualization of the simulation output, and the input widgets that allow the researcher to manipulate the input parameters of the simulation.

The definition of the view, i.e. the transformation of 3D scenes to a 2D screen, is a crucial aspect in 3D graphics applications. This is particularly the case in 3D user interfaces for computational steering. First of all, improper viewing of a 3D scene can prevent a good understanding of the 3D scene; important information can be withheld from the user which leads to an incomplete or, even worse, an incorrect interpretation of the scene. Secondly, in computational steering the 3D scene is dynamic. It is not known in advance how the simulation will evolve and therefore it is not known in advance what the 3D scene will look like. This implies that the viewing also must be dynamic and easily reconfigurable. Thirdly, in computational steering applications the user must be able to interact with the scene. Therefore, the user must have an adequate view of the objects to manipulate and be enabled to get good visual feedback of the changes in the scene which result from his actions. And finally, because he creates his own custom 3D user interface to a simulation, the user has control over the layout of the interface and therefore he must be enabled to specify the view on this interface tailored to his needs.

In this paper, we present a method for the definition of views in custom 3D user interfaces for 3D computational steering applications. The method is based on point-based parametrizable cameras. A view is created by placing a camera in the 3D scene. The characteristics of the camera such as its position and orientation define the view, which is presented to the user in a separate window. These characteristics are defined by the position of the camera's control points. The control points can be moved by the user, parametrized to simulation data,

[1] Netherlands Energy Research Foundation ECN, P.O. Box 1, 1755 ZG Petten, The Netherlands.

or coupled to objects present in the scene. Although we have developed this method to be used within our computational steering environment we believe that it can easily be adapted for usage in many other 3D graphics applications and software packages.

In the next section we describe related work on viewing methods for 3D scenes (Sect. 2.1) along with a brief description of the computational steering environment developed at CWI, including the tool developed to create custom 3D interfaces (Sect. 2.2). Section 3 describes the requirements of view definitions in 3D interfaces for computational steering applications followed by a description of the point-based parametrizable camera object (Sect. 4). In Sect. 5 some example applications of the camera object are shown. Finally, Sect. 6 gives some concluding remarks and indicates areas of future research.

2. Related Work

2.1 Viewing Control

Several projection methods exist for displaying a 3D scene on a 2D display device [4]. Most applications however, use the perspective projection model where the view is defined by specifying the center of projection, the view plane, and the clipping volume. On top of this model, different view manipulation metaphors have been developed for both user controlled and automated view manipulations.

Ware et al. defined the *eyeball in hand*, *scene in hand*, and *flying vehicle control* metaphors for exploration and virtual camera control in virtual environments using a six degree of freedom input device [16]. They found that the different metaphors each have their advantages and disadvantages depending on the particular task that is to be performed. The flying vehicle metaphor for instance was more useful in navigating through an interior while the scene in hand metaphor was useful for manipulating closed objects.

Phillips et al. presented a method for automatic viewing control to support 3D direct manipulation techniques of objects in a scene [12]. Through automatic viewing adjustments their system tries to avoid viewing obstructions and reduce the problems with degenerate axes common to most direct manipulation techniques.

Gleicher et al. presented the concept of *through-the-lens camera control* [5] where the user can manipulate a virtual camera by controlling and constraining features in the image seen through its lens. Constrained optimization is used to compute the time derivates of the actual viewing parameters which allows for controls to be defined independently of the underlying view parametrization.

Many other researchers have addressed research issues associated with view definitions and in particular viewpoint movements [7, 14, 8, 2, 3, 6, 11]. However, most of the developed techniques are heavily application dependent. They do not allow a user to interactively define multiple views on a 3D scene in an intuitive, graphical manner such that each view can be configured and parametrized according to the user's needs.

2.2 The CSE and PGO editor

The Computational Steering Environment (CSE) which is being developed at CWI [15], is based on two major concepts: A central *Data Manager* surrounded by processes called *satellites*. The Data Manager takes care of centralized data storage and event notification, and the satellites can connect to and communicate with the Data Manager by the use of a 'publish and subscribe' paradigm, see Fig. 1. Typically, one of these satellite processes is a simulation. Via a few simple function calls the simulation can open a connection to the Data Manager, connect the appropriate input and output variables, and update and retrieve their values when needed. Once the input and output variables of the simulation have been connected to the Data Manager, additional visualization and data manipulation satellites can be used to visualize the simulation results and steer the simulation by altering the input parameters present in the Data Manager. Several general satellites have been developed for this purpose, such as a logging satellite, a calculator satellite, and a slicing satellite. The most important satellite however is the *PGO editor* [9, 10].

The PGO editor is a tool that enables the user to create 2D and 3D user interfaces for the visualization and manipulation of the (simulation) data present in the Data Manager. These interfaces are constructed out of *point-based Parametrized Geometric Objects*. A set of simple basic objects (such as a sphere, a line, and a box) is provided to the user out of which he can construct complex, composite input/output widgets. The geometry of these basic objects is defined with control points. The control points indicate important perceptual characteristics of the objects such as 'tip', 'bottom', 'corner', etc.. Changes of the positions of these control points

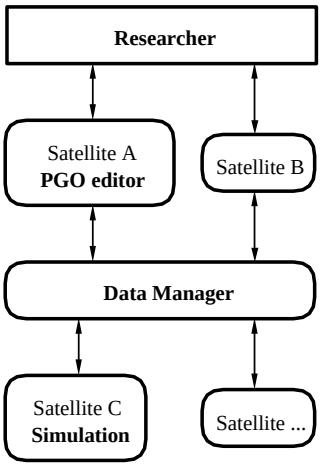

Figure 1: Architecture of the Computational Steering Environment.

change the geometry of the objects. Therefore, the geometry of the objects can be parametrized to data in the Data Manager by parametrizing the point positions to the data. This is accomplished by assigning a domain of allowed positions to a point and using Cartesian or spherical coordinates to describe the point's position inside this domain. These x, y, and z, or *azimuth*, *elevation*, and *radius* parameters are then linked to variables in the Data Manager. As a result, the user can manipulate the simulation input parameters present in the Data Manager by dragging the control points, i.e. the objects can be used for user input through direct manipulation, while changes in the simulations output data cause the objects in the interface to transform, i.e. the objects can be used to visualize the output data. Further, to construct complex, composite input/output widgets hierarchical inter-point connections can be established to propagate control point transformations from one point to another and create inter-object relations.

Figure 2 shows a simple example of an input/output widget that is constructed in the PGO editor. The arrow consists of two basic objects (a cylinder and a cone) that are each defined with three control points. One control point is parametrized to variables in the Data Manager such that the length and orientation of the arrow can be altered.

3. Viewing in Computational Steering Applications

The viewing operations to be provided in 3D computational steering interfaces must enable the user to

- Define multiple views on the 3D scene.
  The benefits of multiple views are obvious: different viewing configurations that can be displayed simultaneously can significantly improve the insight in a 3D scene and relieves the user from having to redefine or switch between different views.

- Parametrize a view to (simulation) data.
  For automated view control the parametrization of the view to data is required. This allows for automated walk-throughs in complex visualizations calculated by an external process (a satellite in the CSE), or coupling of a view configuration to simulation output data.

- Couple a view to entities in the interface.
  This allows the user to automatically trace objects as they move through the 3D scene. This is important if the user wants to focus on one particular aspect of the object or manipulate (part of) the object while it is moving, which is a task that can be quite difficult to perform if the object is moving fast or is located in complex, crowded scenes.

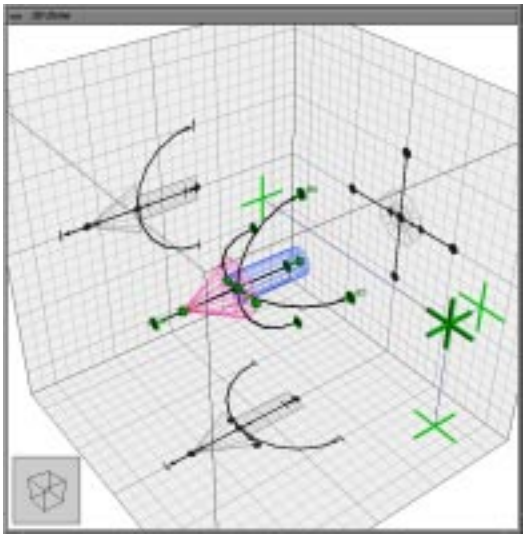

Figure 2: An arrow defined in the PGO editor.

- Construct user defined controls over the view and change a view by direct manipulation.
  The user must also be able to configure a view that he can control and manipulate himself while using the interface to steer a simulation. The manipulation mode however, can differ from application to application and from view to view. Sometimes the user may want to only translate a view along a particular path while on other occasions he might want to be able to rotate the view about a particular axis. Therefore, the user must be enabled to define different view manipulation modes that control the view according to his needs.

To meet these requirements we have developed the point-based parametrizable camera object. This object is provided in the PGO editor and allows for easy definition of different view configurations in the user defined 3D interface.

4. Point-Based Camera Object

The PGO editor provides one main view on the scene. This view is used as an overview of the entire scene. The user can change this overview by direct manipulation of a small box icon present in the main view. To define additional views in the PGO editor a camera metaphor is used. This metaphor allows for easy and intuitive definitions of different view configurations. To create a view a camera object is positioned in the 3D scene. The geometric properties of the camera object are defined by three control points. The camera's position and direction of view can be defined in a very natural way by the use of these control points. One control point is used to define the camera's position and the second control point defines the direction of view; the camera is pointed towards this control point. The third control point is used to define the camera's roll, i.e. the rotation about the line of sight. The plane that is defined by the three camera control points is the median plane of the camera. An important attribute of the camera is the diameter of the lens opening: this defines the zoom factor. The diameter of the lens is derived from the distance between the third control point and the line of sight.

Because the camera object is point-based, it is fully integrated in the PGO editor. The same flexibility that is provided for the geometric objects applies to the camera, and the same interface and manipulation techniques are used for both the construction of the 3D scene and the configuration of the cameras. Each instance of the camera object displays the scene from its point of view in a separate window. The user can edit or manipulate the 3D scene in the main view or in one of the camera windows.

Different viewing configurations can be established by connecting the camera's control points to the control

points of other objects, sharing the control points with other objects, or parametrize the control points to variables in the Data Manager. The user can also create custom camera control widgets out of the geometric objects or change the camera configuration by direct manipulation of the camera's control points.

Figure 3 shows the point-based camera object in one of many possible configurations. The camera is displayed in wire-frame mode to reveal the center control point. The camera's position is parametrized to the variables c_x, c_y, and c_z, its direction of view to c_ele and c_azi, and the roll and zoom to c_roll and c_zoom.

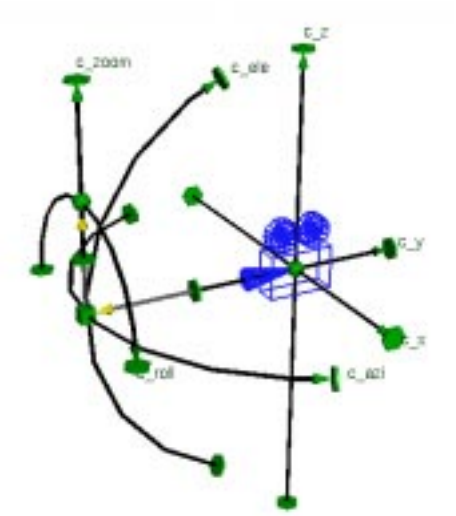

Figure 3: Point-based camera.

5. Examples

Figure 4 depicts a robot arm that was constructed with the PGO editor. The robot arm comprises several different rotational and translational joints. The user can control the robot arm with the sliders, where each slider manipulates one joint, or by manipulation of the robot arm itself.

Two camera objects are used in this scene. One camera is rigidly mounted on the end effector of the arm. Therefore, this camera moves with the end effector and always displays what the end effector 'sees'. In front of this camera a slider has been constructed to control the zoom of the camera. This slider always remains in front of the camera such that the user can adjust the zoom factor by manipulating the slider in the camera window. The other camera has a static position. Its direction of view however, is coupled to the robot arm's end effector. So, this camera traces the end effector from a fixed position. The views of both cameras are depicted in Fig. 5.

The picture on the left in Fig. 6 shows an interface to a path planning application developed by K. Trovato [13] and L. Dorst et al. [1][1]. The interface shows two representations of a car parking problem. One representation visualizes the task space: the street, the car, and the two obstacles in between which the car has to be parked by the path planning program. The other representation shows the configuration space, called c-space. Here the three parameters that describe the configuration of the car are visualized: two position parameters x and y, and the car's orientation ϕ. Each plane represents a particular angle of the car. Because ϕ is cyclic the planes are ordered in a cyclic order. Each plane represents the x and y parameters of the car. The areas in the planes that are not filled represent a parameter configuration for the car such that it does not interfere with the obstacles.

[1] The path planning software is ©by Philips Laboratories, 1988. Philips has four patents pending related to the vehicle planning methods and control.

Figure 4: Robot arm with camera objects.

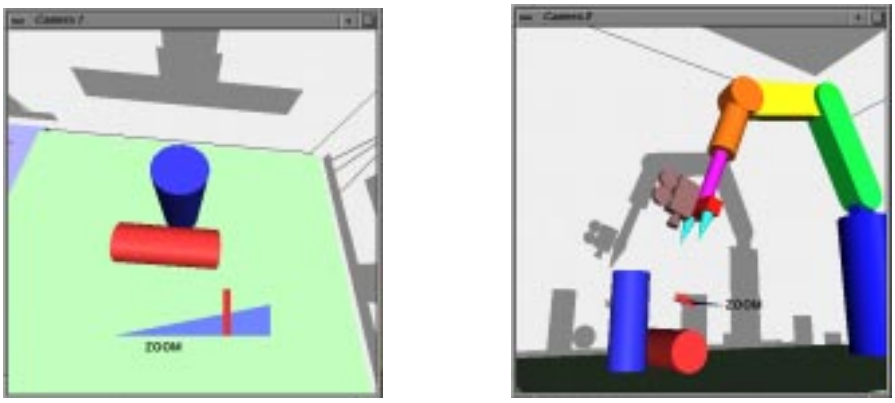

Figure 5: View from the cameras present in the robot arm interface. Left the view from the camera mounted on the end effector and right the view from the camera that traces the end effector.

The user can examine the c-space by manipulating two boundary planes in the c-space visualization to select a region to be visualized.

The car can be dragged to a new initial or goal position by manipulating the car itself or its representation in the configuration space: a small sphere. Also, the two obstacles can be resized by direct manipulation. The traveled path of the car is visualized with wire frame projections of the car in the task space, and with a polyline in the configuration space. A camera has been mounted on top of the car to get an impression of the scene from the driver's perspective as shown in the picture on the right in Fig. 6.

Figure 7 illustrates a different use of the camera object. Here, the control points of the camera have been parametrized to variables in the Data Manager. A separate satellite process is used to control these parameters and thereby to steer the camera. The satellite process has a user interface with several buttons that allow the user to move or rotate the camera and zoom in or out. The satellite process interprets the user's actions on the buttons, computes new values for the variables to which the camera's control points are parametrized and stores these in the Data Manager. The control points of the camera then adapt their position according to the new values and thereby alter the view.

An extensive library of such satellite processes could be developed where each satellite controls the camera in

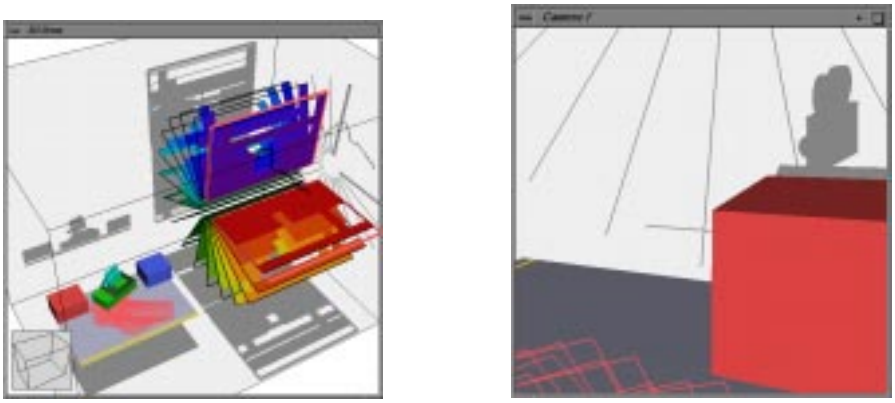

Figure 6: Interface to a path planning application. On the left an overview of the interface, on the right the scene from the driver's perspective.

a different fashion. The user can then select one of these satellites according to which is needed for the particular application. The use of a satellite process to control the camera motions via variables in the Data Manager also enables the usage of special input devices such as a head tracking device or a Spaceball. The satellite process is then used as a device driver that reads out the device's parameters and stores them in the Data Manager.

These examples illustrate some possible applications of point-based parametrizable cameras in the PGO editor. The camera object allows the user to fully customize the viewing of a 3D user interface in the same manner as the user has constructed that interface itself. Many different viewing configurations can be realized yet the camera model remains easy and intuitive to use.

6. Conclusion

3D Graphics applications can greatly benefit from multiple user defined or automated views. This holds particularly for customized 3D interfaces for computational steering where the user is to interact with a dynamic 3D scene. Different viewing configurations will aid the user in understanding the 3D scene and provide support for manipulations on the objects in the scene. The user should be enabled to create and define the views as he also creates the interface itself and therefore will have specific ideas for the views on the interface.

The point-based parametrizable camera object presented in this paper provides an easy and intuitive method to define multiple views with different configurations. The user can easily manipulate the views themselves, link the views to objects in the scene or parametrize the views to (simulation) data for automated camera control. Even though we have developed the camera object for usage in the PGO editor we think that the principle can easily be adapted for usage in other applications.

One of the research items we want to explore in the near future is the aspect of view selection. If multiple views are defined on a 3D scene, can we develop techniques for automated selection of the best view for a particular task? For instance, if multiple cameras are used to trace different objects through the 3D scene it might be useful to automatically pop up the appropriate camera view upon selection of the object the camera traces. Other future research areas include the use of multiple views in virtual reality applications and the development of techniques for automated camera generation.

References

1. L. Dorst, I. Mandhyan, and K. Trovato. The geometrical representation of path planning problems. *Robotics and Autonomous Systems*, 7:181–195, 1991.
2. S.M. Drucker, T.A. Galyean, and D. Zeltzer. CINEMA: A system for procedural camera movements. In D. Zeltzer, editor, *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, pages 67–70, 1992.
3. S.M. Drucker and D. Zeltzer. CamDroid: A system for implementing intelligent camera control. In P. Han-

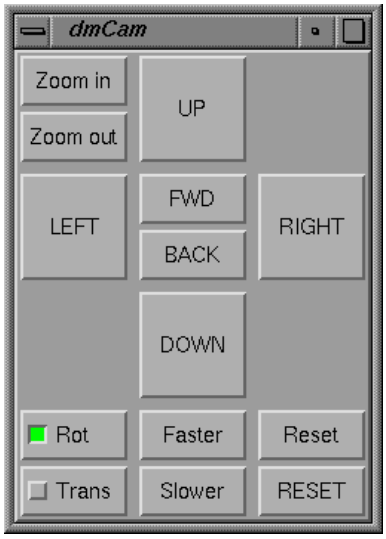

Figure 7: A separate satellite to control camera movements.

rahan and J. Winget, editors, *1995 Symposium on Interactive 3D Graphics*, pages 139–144, 1995.

4. J. Foley, A. van Dam, S. Feiner, and J. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley, second edition, 1990.
5. M. Gleicher and A. Witkin. Through-the-lens camera control. In E.E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 331–340, 1992.
6. L. He, M.F. Cohen, and D.H. Salesin. The virtual cinematographer: A paradigm for automatic real-time camera control and directing. In H. Rushmeier, editor, *Computer Graphics (SIGGRAPH '96 Proceedings)*, pages 217–224, 1996.
7. J.D. Mackinlay, S.K. Card, and G.G. Robertson. Rapid controlled movement through a virtual 3D workspace. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, pages 171–176, 1990.
8. M. McKenna. Interactive viewpoint control and three-dimensional operations. In D. Zeltzer, editor, *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, pages 53–56, 1992.
9. J.D. Mulder and J.J. van Wijk. 3D computational steering with parametrized geometric objects. In G.M. Nielson and D. Silver, editors, *Visualization '95 (Proceedings of the 1995 Visualization Conference)*, pages 304–311, 1995.
10. J.D. Mulder and J.J. van Wijk. Logging in a computational steering environment. In R. Scateni, J. van Wijk, and P Zanarini, editors, *Visualization in Scientific Computing '95, Proceedings of the sixth Eurographics Workshop*, pages 118–125, 1995.
11. P. Palamidese. A camera motion metaphor based on film grammar. *Journal of Visualization and Computer Animation*, 7(2):61–78, 1996.
12. C.B. Phillips, N.I. Badler, and J. Granieri. Automatic viewing control for 3D direct manipulation. In D. Zeltzer, editor, *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, pages 71–74, 1992.
13. K. Trovato. Autonomous vehicle maneuvering. In *Proceedings SPIE Volume 1613*, pages 68–79, November 1991.

14. R. Turner, F. Balaguer, E. Gobetti, and D. Thalmann. Physically-based interactive camera motion control using 3D input devices. In N. M. Patrikalakis, editor, *Scientific Visualization of Physical Phenomena (Proceedings of CG International '91)*, pages 135–145, 1991.

15. J.J. van Wijk and R. van Liere. An environment for computational steering. In G.M. Nielson, H. Müller, and H. Hagen, editors, *Scientific Visualization: Overviews, Methodologies, and Techniques*, pages 89–110. Computer Society Press, 1997.

16. C. Ware and S. Osborne. Exploration and virtual camera control in virtual three dimensional environments. In R. Riesenfeld and C. Sequin, editors, *Computer Graphics (1990 Symposium on Interactive 3D Graphics)*, pages 175–183, 1990.