StyleCam:
Interactive, Stylized 3D Experiences using Integrated Spatial and Temporal Controls.

by

Nicolas A. Burtnyk

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Abstract

StyleCam: Interactive, Stylized 3D Experiences using Integrated Spatial and Temporal Controls.

Nicolas A. Burtnyk

Master of Science

Department of Computer Science

University of Toronto

2003

This thesis describes StyleCam, an approach for authoring 3D viewing experiences that incorporate stylistic elements which are not available in typical 3D viewers. StyleCam allows the author to tailor what the user sees and when they see it. The resulting viewing experience can approach the visual richness and pacing of highly authored visual content such as television commercials or feature films. StyleCam allows for a satisfying level of interactivity while avoiding the problems inherent in using unconstrained camera models. StyleCam's main components are camera surfaces which spatially constrain the viewing camera; animation clips that allow for appealing transitions between camera surfaces; and a simple, unified, interaction technique that permits the user to seamlessly and continuously move between spatial-control of the camera and temporal-control of the animated transitions. In addition to describing the conceptual model of StyleCam and its current implementation, we present the results of an evaluation involving real users.

Acknowledgements

I would like to thank a number of people for their help and support throughout the writing of this thesis: My supervisor, Bill Buxton, for his truly invaluable advice, support and commitment to helping me finish on time. Gord Kurtenbach, George Fitzmaurice, Azam Khan and Ravin Balakrishnan for all their ideas that helped flesh out the basis for my thesis, for the time and effort they put in during the writing of our StyleCam paper, for their feedback, encouragement and guidance, for their general enthusiasm about the StyleCam research, and for all the laughs alpng the way. I would especially like to thank George for taking the time to be my second reader in the eleventh hour. Also thank you to Scott Guy for all the help with Maya.

Without the resources of the research lab at Alias, my research would have not been possible. I would like to thank them for their support. I also appreciate the support I received from the province of Ontario which provided me with funding throughout my research.

I would finally like to thank all my friends and family, especially my girlfriend Lori-Ann and my parents for their persistence in pushing me to complete my thesis and for their love and support along the way. Thank you also to my brother Mathieu for reminding me to always take things in stride.

Table of contents

List of Figures

Chapter 1. Introduction

The last decade has seen the internet be embraced by companies and manufacturers as a medium for delivering information and advertising to consumers. It is uncommon now for a company *not* to have a presence on the web. Indeed product overviews, technical specs, photographs, user manuals, pricing information and more are readily available online for all but the most obscure products. The internet has allowed companies to deliver traditional media (specification sheets, brochures, advertising videos, etc.) to people in the comfort of their home quickly and cheaply. In some case, the information and media available online is the same as what one could perhaps have mailed to them. In general, however, the web defines a new medium, the web page, which has no traditional equivalent. A web page is interactive. A web page integrates and augments text, images, audio and video.

At the same time, computers are getting more powerful every day. Most people are familiar with Moore's law of processor speed and memory capacity doubling every 18 months. Computer graphics has been a significant recipient of this rapidly increasing computing power, and as a result we have reached a stage where it is possible to create and render 3D models in real time on off-the-shelf, inexpensive hardware at near photorealistic levels of fidelity.

This trend is in turn making it feasible to use interactive 3D models instead of 2D imagery to represent or showcase various physical artefacts (products). Typical examples are online retailers who are now starting to provide ways of interactively viewing products in 3D over the web, in addition to providing 2D imagery. This is a tremendous leap forward. Potential customers can interactively explore the virtual product, giving them a better "feel" for the product than a 2D image. According to Andrew Jackson of MGI Software, an Internet software and services company, there is a 30 percent increase in sales and a 50 percent reduction in returned goods with the use of 3D imaging in online retailing. Moreover, interactive 3D on a web site has the potential to realize longer visitor retention due to the interactive nature of the experience. Car manufacturers have similarly adopted the use of interactive 3D viewing for product showcasing both on and off the web, in addition to the traditional professionally produced 2D car brochures. At a press conference in Toronto, Torben Ullmann of Pixelconcept GmbH describes the deployment of "… a virtual experience that is similar to being in an auto showroom - but with the convenience of browsing online" for the German automaker Porsche AG.

At first glance it may appear that viewing interactive 3D models will soon dominate as the most effective product seller's medium, but a closer look reveals that fundamental problems with today's interactive 3D viewing will severely impede such a trend. Indeed, interactivity aside, in terms of visual experience the current 3D viewers fall quite short of the slick, professionally produced 2D images of the same items. Again looking at the car manufacturer example, a glossy automobile sales brochure provides a far richer and more compelling representation of the car to the potential customer than the 3D experiences offered on the manufacturer's website. If these 3D viewers are to replace, or at the very least be at par with, the 2D imagery, eliminating this difference in quality is critical.

Figure 1.1. Screenshots of QuicktimeVR (left) and Cult3D (right).

What are we referring to by "today's interactive 3D viewers"? What are today's standards in interactive 3D viewing? Some popular examples that are prevalent on the web are QuickTimeVR (QTVR) and Cult3D (see Figure 1.1). Both of these specifications allow users to explore 3D scenes in some constrained way. Both viewers have very limited camera control metaphors. QTVR, in object-centric mode, allows the user to tumble around a central object of interest and to zoom in and out. In panorama mode, QTVR maps 2D mouse movement to the viewing direction giving the effect of a stationary observer who can move their gaze up, down and around. Cult3D is slightly more sophisticated. It supports the "standard" camera controls of 2D tumble, 2D pan and dolly, as well as pre-defined key views (e.g. "top" or "front"). Cult3D also has support for changing features based on user interaction. For example, the user can click on one of several color swatches to change the car color. The viewers use drastically different source data each with its own advantages and disadvantages (see chapter 2, detailing image-based versus geometry-based viewing). The bottom line is that although QTVR, Cult3D and other current interactive viewers allow 3D content to be viewed interactively, which is entirely new and exciting, the experience felt by the user during an interactive session is not exactly compelling. Using Cult3D to awkwardly spin a 3D car at low resolution and to "experiment" with various

rims, colors and spoiler styles does not compel the user to buy the car nearly to the same extent as say the same car's brochure or television ad. As such, given the current state-of-the-art, interactive 3D can only play a minor role in product advertising or showcasing.

The reasons for the poor quality of these 3D viewers fall roughly into three categories. First, 2D imagery is typically produced by professional advertisers, artists, and photographers who are skilled at using this well-established art form to convey messages to a viewer such as information, feelings, or experiences, whereas creators of 3D models do not necessarily have the same well-established skills and are working in an evolving medium. This problem, however, will work itself out as the medium matures.

The second problem is that of visual quality. Although advances in computer graphics hardware and the development of advanced algorithms are occurring at a high rate, rendering photoreal 3D scenes with realistic lighting and reflections, in realtime, at high resolution and on consumer hardware is still impossible. The importance of visual fidelity is significant. To take an example again from the automotive domain, in many ways, automobile stylists do not simply design cars, they design reflections. The car body is a means of obtaining these reflections. Clearly, to fully appreciate the beautiful reflections of an automobile requires faithful rendering of reflections and other lighting effects beyond the capabilities of today's realtime rendering solutions. This problem, however, is solvable with better, more powerful graphics hardware. If the current trends continue, it will not be long before such a feat is possible on a $2000 PC.

Figure 1.2. A comparison between a traditional advertising media and interactive 3D media.
Top row: Scenes from a car commercial shoot vs. the creation of a interactive 3D experience.
Middle row: Real, high-impact reflections and lighting vs. real-time rendered lighting effects.
Bottom row: Beautiful, expressive, action-packed wide-angle close-up with a slight right roll vs. an attempt to get same angle in Cult3D (roll and zoom not possible).

The third problem with today's interactive 3D is more troublesome. Consider, for example, a photographer taking pictures of a car for a brochure. In this scenario, the photographer can carefully control all of the elements that make up the shot, including the content, lighting, viewpoint and other camera properties (e.g. aperture). Similarly, the director of an automobile television commercial has full control over these same elements, with the additional variable of time. Given this level of control, skilled authors in these media can manipulate the stylistic elements of a shot in order to guarantee or at least increase the likelihood that the viewer receives the intended message. In contrast, when designing content for interactive 3D viewing, authors must relinquish much of this control to the "audience". How can we expect a typical user to achieve the results of skilled professionals working in a production environment? Skimming through car brochures, one can quickly note the consistency in the angles from which the cars are photographed. In fact, according to Barry Fogarty of Diginiche, there are fewer than half a dozen photographers in Detroit who know how to take photographs of automobiles and only one or two people in a car company who are capable of producing renderings of automobiles that meet the stringent demands of this type of presentation (personal communication, Barry Fogarty). It is no surprise that the average user using a 3D viewer will not experience the vehicle in the way the designer or marketer intended.

Figure 1.3. Examples of some poor angles that are obtainable in Cult3D: the underside of a vehicle (left), and the unrecognizable interior of a camcorder model (right).

In interactive 3D viewing the author can only control the content, the lighting and sometimes the camera properties. In Cult3D, or any other interactive viewer that uses the tumble/pan/dolly camera control metaphor, the user is allowed to interactively move the viewpoint within the scene to view any part of the 3D model, effectively allowing any arbitrary camera position and orientation. This results in a host of problems: a user may "get lost" in the scene, view the model from awkward angles that present it in poor light, miss seeing important features, experience frustration at controlling their navigation, etc (see Figure 1.3). As such, given that the author of the 3D experience does not have control over all aspects of what the user eventually sees, they cannot ensure that 3D viewing conveys the intended messages. In the worse case, the problems in 3D viewing produce an experience completely opposite to the author's intentions!

The primary goal of our present research is to introduce the same level of author control and user experience that we see in print ads and television commercials to the world of interactive 3D viewing. In hopes of achieving this goal, we have developed a system which we call StyleCam. StyleCam aims to guarantee a certain level of quality for the user, in terms of visual and interactive experience. Further, we intend that the system should not only avoid the problems suggested earlier, but also have the capability to make the interactive experience adhere to particular visual styles. For example, with StyleCam one should be able to produce an interactive viewing experience for a 3D model of an automobile "in the style of" the television commercial for that same automobile. Ultimately, a high-level goal of our research is to produce interactive 3D viewing experiences where, to use an old saying from the film industry, "every frame is a Rembrandt".

Figure 1.4. Images from automobile brochures illustrating successful, compelling visual experiences.

Central to our research is differentiating between the concept of authoring an interactive 3D experience versus authoring a 3D model which the user subsequently views using general controls. If we look at the case of a typical 3D viewer on the web, in terms of interaction, the original author of the 3D scene is limited to providing somewhat standard camera controls such as pan, tumble and zoom. Essentially, control of the viewpoint is left up to the user and the author has limited influence on the overall experience. Any narrative structure that comes out of the experience is purely due to the actions of the user, not the author who is presumably much more skilful at such narrative visual experiences.

From an author's perspective this is a significant imbalance. If we view an interactive experience by cinematic standards, an author (or director) of a movie has control over several major elements: content/art direction, shading/lighting, viewpoint, and pacing. It is these elements that determine the overall visual style of a movie. However, in the interactive experience provided by current 3D viewers, by placing control of the viewpoint completely in the hands of the user, the author has surrendered control of two major elements of visual style: viewpoint and pacing.

Thus we desire a method for creating 3D interactive experiences where an author cannot only determine the content and shading but also the viewpoints and pacing. However, intrinsic in any interactive system is some degree of user control and therefore, more accurately, our desire is to allow the author to have methods to significantly influence the viewpoints and pacing in order to create particular visual styles. Thus, we hope to strike a better balance between author and user control. In order to achieve this end, StyleCam incorporates an innovative interaction technique that seamlessly integrates spatial camera control with the temporal control of animation playback. Figure 1.5 depicts camera surfaces and various types of transitions between them; elements of StyleCam which an author can use to influence what the user sees and experiences.

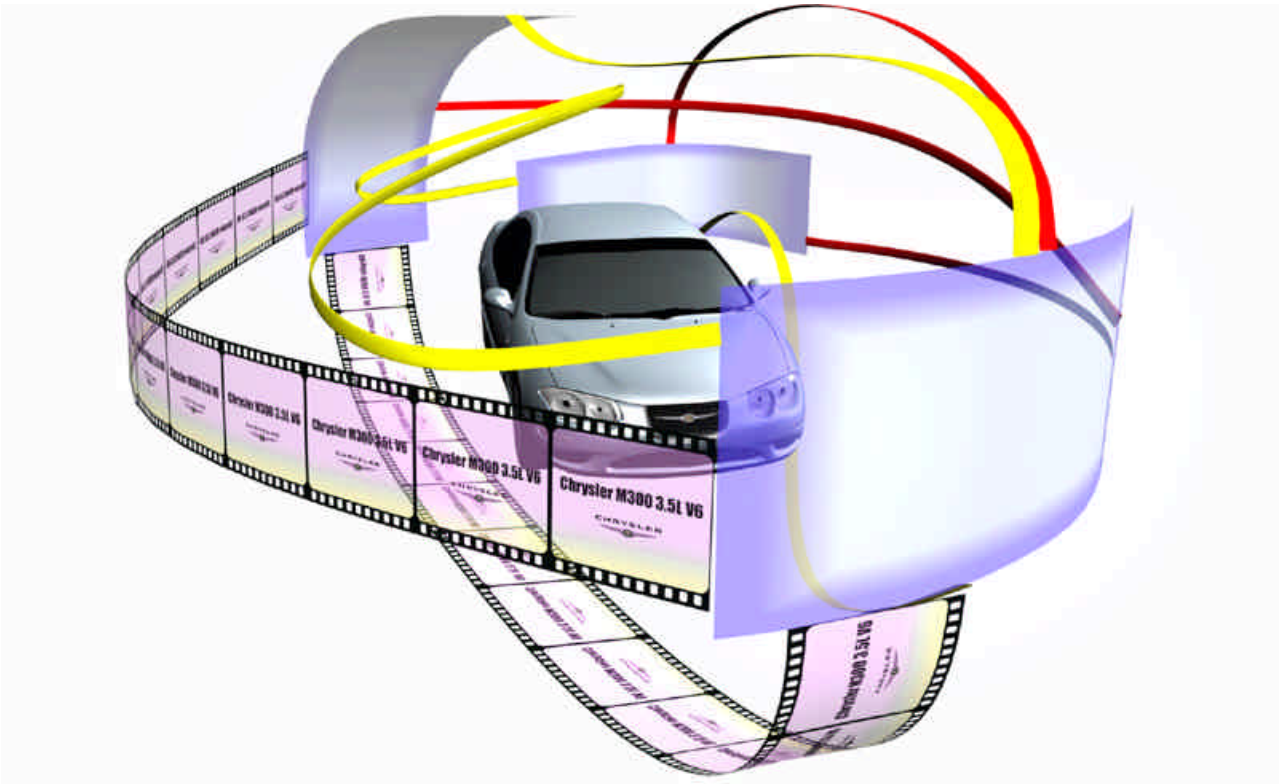

Figure 1.5. StyleCam authored elements.

This thesis is a summary of the body of research that is StyleCam. The current chapter introduced the problems StyleCam aims to solve. Chapter 2 is a primer on the fundamentals of virtual camera movement and its associated user interactions. Chapter 3 is a literature review of past work in virtual camera movement in interactive 3D, that sets the standards by which we will compare StyleCam. In chapter 4, we introduce the conceptual model of StyleCam, showing how the vision of an author can be recorded to be subsequently experienced by an audience. Chapter 5 explores the previous prototypes that lead to the current StyleCam, describing how each iteration improved on the previous. In chapter 6, we consider StyleCam in the context of the other tools that would eventually surround it in the production pipeline. Included in chapter 6 is an example experience, intended to give the reader a feeling of the type of experiences StyleCam can produce. Chapter 8 describes in detail how the StyleCam system was implemented. An informal user evaluation is presented in chapter 9. Finally, chapter 10 discusses possible future directions that we plan to explore with StyleCam.

Chapter 2. Basic Concepts

2.1 Camera & Optical Moves

This thesis deals primarily with camera movement and user interaction in a computer usage context. It is therefore useful to begin this discussion with a review of some basic camera and optical moves used in cinematography. Figure 2.1 illustrates 6 basic moving camera shots which form the building blocks for more complex cinematic shots:

Dolly – camera movement along the viewing direction, either forward or back.

Track – lateral camera movement perpendicular to viewing direction, either left or right.

Crane – vertical camera movement perpendicular to viewing direction, either up or down.

Pan – rotation of camera about its vertical axis, either left or right.

Tilt – rotation of camera about its horizontal axis, either up or down.

Locked-Off – camera remains stationary during shot.

Zoom – focal length of camera lens is smoothly changed.

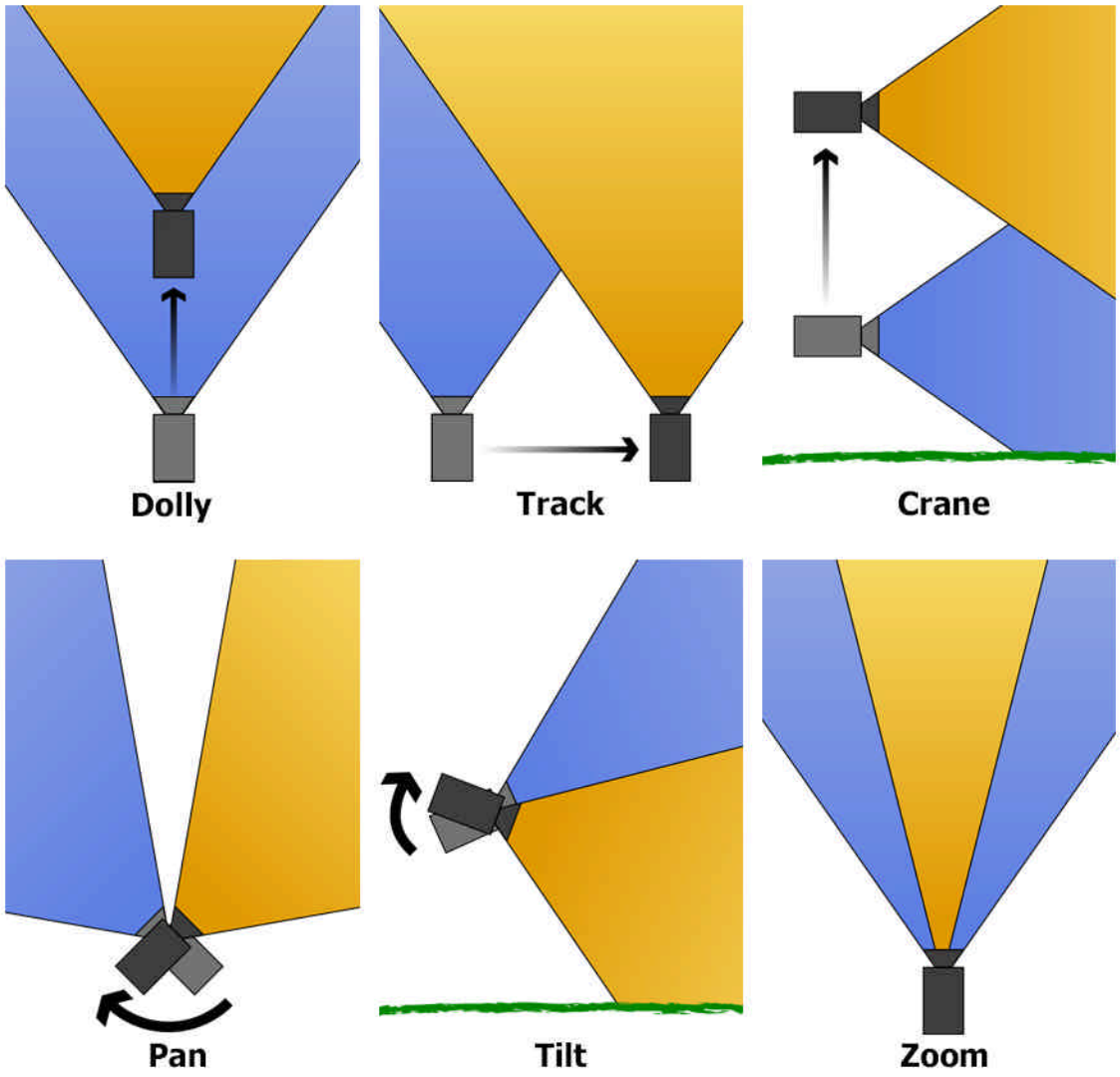

Figure 2.1. Basic cinematic camera moves.

Though these basic camera moves can and often are used by themselves, they can just as easily be combined either in sequence (e.g. dolly out then track right) or in unison (e.g. the cliché Hollywood ending where the camera pulls back and "flies" up about 35 feet is composed of a vertical crane, a dolly and tilt move, all occurring at the same time). A common type of camera move not listed above is the *follow* shot where the camera is hand-held and follows the action (usually with the help of a Steadicam to reduce camera shake and for remote focus and exposure control by assistants). Another common camera "move", the *rack focus* uses rapid changes in

camera lens focus to bring attention to certain subjects. Since the camera itself does not move (only glass elements within the lens), this type of shot is more correctly referred to as an optical move.

Some camera moves are rarely used in film and television but ubiquitous in 3D viewing. For example, a camera *tumble*, which refers to orbiting the camera around a central point in any direction as if sliding it along the surface of a sphere, is the primary camera movement used in almost every 3D viewing application. It is a natural way to inspect the surface of an object, and works very well with a pointing device such as a mouse since it is a 2-dimensional operation.

2.2 Turntables

The concept of *turntables* has been used extensively in the context of design visualization for both physical and virtual models. The term turntable refers to the technique of having the user rotate the object in front of the camera much like as if it was placed on a record player and spun. This is in contrast to the standard technique of orbiting the camera around the object. When the model being examined is a blob-like object like a car or a person or household appliance, the turntable technique actually works very well – the experience is similar to examining an object held in ones hand. Turntables are thus typically most useful when evaluating the form of a single object. The technique is however not well suited for examining more complex objects, groups of objects, landscapes, interiors or any scene with more than one subject. With such scenes, the object-held-in-hand metaphor breaks down. In addition, the turntable metaphor is too restrictive, in that it may not be possible to view all the details of the scene since the camera cannot take on an arbitrary position and orientation. More fundamentally, the turntable metaphor breaks Newton's law of relative motion, which can be easily observed by comparing the reflections produced while examining a virtual model containing static lights with and without the turntable technique.

2.3 Viewing Image- vs. Geometry-based Models

In the context of viewing computer generated imagery, there are two distinct "families" of approaches. The traditional approach is to render an image from a 3D representation of a scene. The 3D geometry data can come from any number of sources including CAD modeling tools or 3D digitizing tools. The 3D representations are commonly in the form of polygonal meshes or parametric "patches", such as nurbs surfaces. A 3D model of a car is shown in Figure 2.2, where the underlying geometry is clearly visible in the wireframe view. 3D models are also typically enhanced by applying texture maps. In contrast with viewing geometry-based models, image-based viewing techniques use a collection of stored images to generate novel views of the scene. There exist several varieties of image-based viewing techniques typically characterized as non-physically based image mapping, mosaicking, interpolation from dense samples, and geometrically-valid pixel reprojection. Each technique uses only source images to produce novel views. For a review of image-based viewing techniques see Kang (1997).

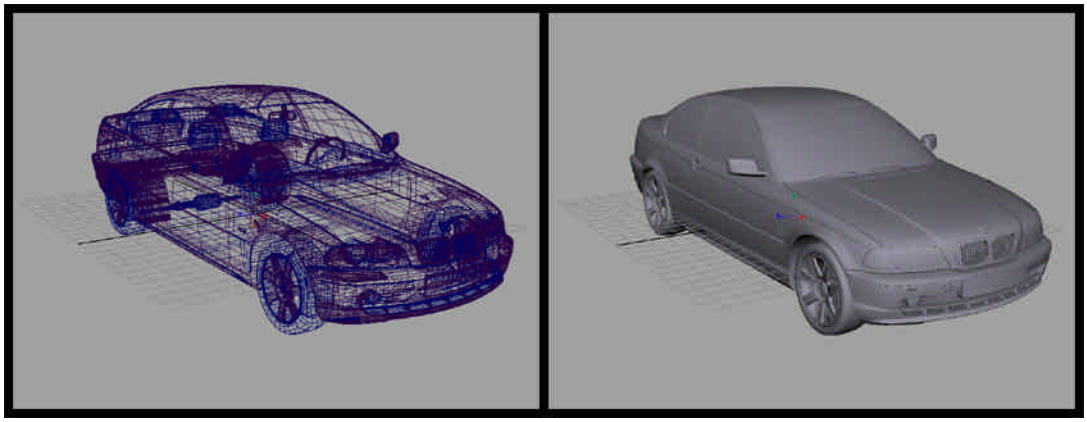

Figure 2.2. Wireframe (left) and shaded (right) views of a 3D model of a car.

Geometry-based and image-based viewing differ in the types of camera moves they support and in their support for dynamic lighting and moving scene objects. The following table shows a simple comparison between these two approaches in the context of moving or manipulating the camera, the lights in the scene or the objects in the scene.

	Geometry-based	Image-based
Main difference	Arbitrary views computed on demand from 3D geometry.	Views are precomputed or determined from nearby, precomputed views.
Camera	No difficulties in computing an unconstrained, user controlled camera. Arbitrary views can be generated at equal cost.	Restrictive. In the best case, any novel view requires a sample image from a similar point of view. In the worst case the camera movement is highly constrained (e.g. QTVR).
Lighting	No difficulties in computing multiple moving and changing light sources. Highly realistic lighting is however computationally expensive and is typically not feasible in a real-time rendering context.	Essentially impossible to support dynamic lighting, especially when using photographs. Potential of higher quality and even photorealistic lighting.
Object	No difficulties in handling a reasonable number of moving objects. Object geometry can also be animated. Increased complexity in object geometry or movements increases the computational cost of rendering.	Can have animated sequences associated with a viewpoint (e.g. QTVR) but impossible to support unconstrained arbitrary movement of scene objects.

Geometry-based and image-based viewing differ fundamentally in the relative trade-offs between speed of rendering, photorealism of output images, compactness of representation (size of input dataset), types of interactions afforded and the relative difficulty of doing so.

The strengths of image-based viewing lie primarily in the quality of the renderings. Indeed the quality of image-based renderings is directly related to the source images used. By using actual photographs, photorealism is easily achievable. On the other hand, the quality of geometry-based renderings is dependant on the quality of the 3D geometry, the textures applied to the geometry,

the rendering algorithms used and more. Achieving photorealism is generally difficult using geometry-based rendering since it means acquiring and using highly complex 3D models, photorealistic texture maps and expensive rendering algorithms (to support diffuse, reflected lighting, depth-of-field, motion blur, etc…).

Geometry-based viewing makes use of the traditional graphics rendering pipeline. The cost of rendering a view therefore is dependant on the complexity of the scene and the type of rendering algorithms and special effects used. High speed rendering is possible however since the geometry rendering pipeline is commonly implemented in hardware in today's personal computers. In contrast, image-based rendering speed is independent of scene complexity. The cost of rendering is not small however and no specialized hardware acceleration exists to help.

The amount of data required to describe a scene is generally greater with image-based viewing. This may not be strictly true for all scenes in all cases, but geometry data is considered to be a more compact representation of a scene than a collection of images. The more the viewpoint is allowed to vary in image-based viewing, the more data will be required, whereas the amount of data describing the scene is independent of the number of constraints on the camera.

Finally, as summarized in the previous table, image-based and geometry-based viewing differ in the types of interactions supported. Geometry-based rendering can handle arbitrary viewpoints with ease, since the conventional rendering pipeline places no limitations on the virtual camera. Furthermore, it is trivial to support such user interactions with the scene as opening a door or changing the colour of an object since the virtual objects which compose the scene are representations of actual objects. This is in contrast to the situation with image-based viewing where the scene data consists of a collection of images and there is no specific correspondence between scene objects and data. While it is possible in theory to support, for example, the changing of an object's color, this would be a highly involved process which would include

identifying the object in each source image (to be able to detect a user interaction with the object) and storing multiple copies of the image, each with the object drawn in a different color. Given the fact that the images used in image-based viewing are typically photographs, this process would be even more difficult. Thus geometry-based viewing supports a much wider range of user interactions and supports them with much greater ease than does image-based viewing.

2.4 Camera Moves and Interaction

So far we have discussed camera moves and dynamic scenes, exploring how these relate to the internal representation of the scene (geometry-based vs. image-based). The following section describes how these camera moves and scene interactions take place – that is the ways in which a user can manipulate the camera and objects in the scene. Examples will be drawn from existing 3D viewing applications.

At the highest level, we can distinguish between graphical and keyboard interaction methods. Keyboard interaction is typically performed by using the arrow keys and additional command keys, and is therefore limited in the number of controllable degrees of freedom (DOF). For many manipulations, 2 or 3 DOF is enough. For example, moving an object relative to some plane in virtual space (e.g. the ground plane, or the image plane) is easily accomplished with keyboard arrow controls. Another example is camera manipulation in QTVR. Since the camera is constrained to a 2D tumble + zoom[1] in object-centric mode and pan + zoom in panorama mode, the combination of keyboard arrows and two extra keys (Shift and Control) provide all the necessary degrees of freedom. Interaction with the keyboard quickly breaks down with anything beyond such simple manipulations, and the number of keys the user must think about and remember becomes unwieldy. In most image-based viewing systems, this is not an issue since

[1] The QTVR zoom is not equivalent to a true optical zoom (changing focal length of camera lens). In QTVR, 'zoom' enlarges and reduces the image by scaling the image pixels. In real cameras, the depth-of-field (range of distances at which objects stay in focus) also changes with the focal length for a given lens aperture.

these viewers almost always place substantial constraints on the types of camera or object manipulation they allow. This is not the case however in the majority of geometry-based viewers which naturally allow more flexibility in camera and object manipulations (because they can!) and thus carry the burden of facilitating control of more degrees of freedom than can be reasonably handled with the keyboard alone. The *fbx* Quicktime extensions from Kaydara, which transform Quicktime from an image-based viewer into a 3D geometry-based viewer, illustrate this breakdown perfectly. The fbx extensions use no less than fifteen keyboard command or modifier keys (e.g. grid on/off – G, Lights on/off – L) in addition to the arrow keys. The keyboard interaction is at best awkward and clumsy. More information about the fbx Quicktime extension can be found on the WWW at *www.kaydara.com/products/fbx_for_QuickTime/.*

The limitations of keyboard control make graphical control methods the popular choice in the vast majority of viewer implementations. Indeed, we have not been able to identify any viewer, image-based or otherwise, which did not make use of graphical controls in some way (for example as an alternative to the keyboard controls). We distinguish three classes of graphical interaction methods: invisible widgets or *indirect* controls, manipulators or *in-scene* controls, and visible widgets or *on-screen* controls.

Indirect control methods are so named because they lack any direct interaction with a visible widget or object. An example of indirect control which is used in many commercial 3D modeling and animation software is viewpoint (camera) manipulation using the mouse and modifiers (keys, menu items, physical buttons or graphical buttons). To tumble the active view in MAYA, for example, the user simply holds down the ALT key while clicking and dragging the mouse in the view. Indirect control is perhaps the best suited method for the user to manipulate their viewpoint. On the other hand, on-screen control methods are those where the user interacts directly with a visible widget. A typical example is a control panel with buttons sliders and other widgets that are linked to manipulations of the scene. In-scene controls (often referred to as

manipulators) make use of visual in-scene widgets (superimposed over the object to be manipulated) to assist with and possibly constrain the manipulation. This differs from on-screen control panel type controls in that the widget is actually in the 3D scene. MAYA makes extensive use of manipulators for object transformation (translation, rotation and scale). When an object is selected and a transformation tool is active, a manipulator corresponding to the transformation type is shown superimposed over the object. The translation manipulator, for example, shows three perpendicular axes and a central box. Clicking the mouse over one of the axes and dragging causes the object to be translated along that axis. Translation in multiple dimensions is possible by dragging the central box of the manipulator. See Figure 2.3 below for screenshots of MAYA showing the translation, rotation and scale manipulators. For another example of in-scene controls, the virtual trackball, see Chen (1988).

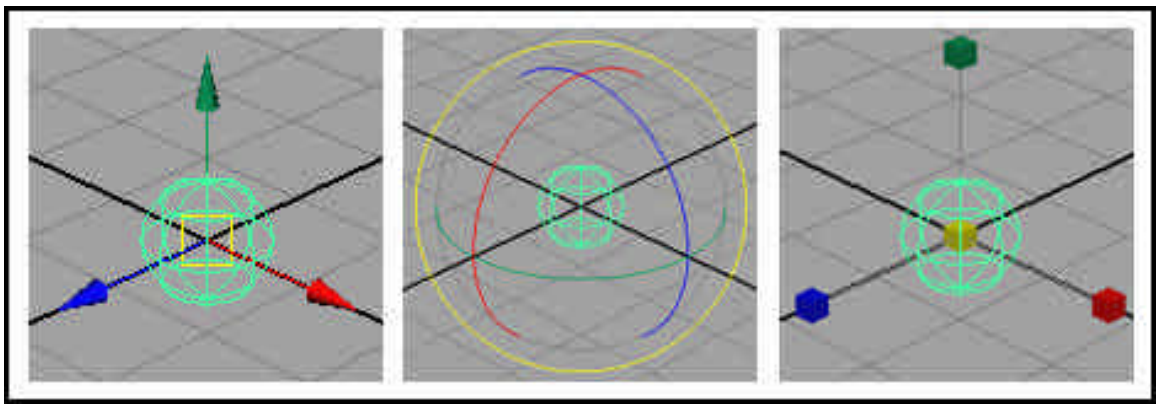

Figure 2.3. MAYA manipulators.

In this chapter we have looked at the kinds of camera movements used in cinematography and cross over to 3D viewing. We also discussed the differences in the forms of input data used in 3D viewing, geometry and images and explored the advantages and disadvantages of each. Finally we covered how the user interaction takes place, in the context of 3D viewing on a computer. These topics were covered to facilitate the review of the current state-of-the-art in 3D viewing which we cover in the following chapter, and to lay the groundwork for a proper comparison of the current state-of-the-art to StyleCam.

Chapter 3. Related Work

There are numerous interactive 3D authoring and viewing systems in existence today. In the following we will list the features of widely available, leading interactive 3D systems and comment on why these systems, though highly sophisticated, do not solve the problem we have described. We will continue by taking a look at current research on the more general problem of camera control and see if the techniques described in that research provide a suitable solution to the problem.

Viewpoint Corporation produces a suite of tools for the creation and viewing of interactive 3D content (Viewpoint Products). Experiences created with Viewpoint's tools are of reasonably high visual fidelity, but support only simple, constrained camera controls, specifically tumble/pan/zoom and bookmarked views. The tumble/pan/zoom camera control technique employed in the Viewpoint system is slightly better than the standard in that it has author-defined maximal and minimal extents, with which an author can restrict the camera movement within certain limits. This can perhaps reduce the frequency of user errors in controlling the camera. Bookmarked views allow the author to define a set of viewpoints which will be accessible to the user via some UI control (e.g. a button). When a user selects a bookmarked view, the camera is smoothly animated from its current position/orientation to the bookmarked view's

position/orientation. The animation is automatic, determined by the system, and can presumably give undesirable results in many cases since the author has no real control over the camera's path through space. The Viewpoint system also supports the animation of scene elements, a very nice feature that can be used to affect changes in the scene based on certain actions on the part of the user. For example, an animated sequence explaining a vehicle's suspension can be played when a UI tab labelled "suspension" is clicked or if the user clicks on or near the vehicle's suspension assembly. It is of interest to note that these animations can include camera animations that move the user's view through the scene.

Cycore's Cult3D product line is very similar to the Viewpoint products described above (Cycore Cult3D). Cult3D boasts reasonably high visual fidelity by using geometry-based rendering and antialiasing techniques. With respect to the interaction methods, authored bookmarked views are supported in addition to the standard tumble/pan/zoom controls. Scene objects, and even the camera itself, can be animated by the author. These animations are triggered by on screen UI widgets.

VRML is an open standard for interactive 3D on the web (VRML Specification). The VRML specification does not support any type of author-influenced user interaction other than bookmarked views. Since VRML is an open specification, a large number of tools exist for creating and viewing VRML worlds. We will focus on the capabilities of one of the most popular VRML viewers, Blaxxun Interactive's Contact (Blaxxun Contact). Blaxxun Contact supports several navigation modes. In "walk" mode, horizontal displacement of the mouse causes left and right panning while vertical mouse displacement cause the view to dolly in and out. Horizontal mouse movement in "slide" mode causes the virtual camera to move perpendicular to its gaze vector and up vector. Vertical mouse movement causes the virtual camera to move up and down along its up vector. "Examine" mode provides 2D tumble around a fixed point of interest and "Pan" mode provides 2D panning mapped to the two mouse axes. Contact also implements a

mode called “Jump” which will respond to the user’s mouse clicks by focusing in on the closest point in the scene under the cursor. Other features of Contact include simple physics approximations of collisions and gravity. Despite the wealth of features and navigation modes available in Contact, there is no mechanism for the author of a VRML world to introduce style by influencing the viewpoint and the pacing felt by the user. As such, experiencing VRML worlds, even with a state-of-the-art VRML viewer, is far from compelling.

The most advanced system for creating and viewing interactive 3D content today is probably the software suite known as Virtools. Interactive 3D experiences are authored with Virtools Dev, published using Virtools Behavior Server and experienced using the Virtools Player (Virtools Products). Virtools Dev is unique in that authors can introduce rich dynamic behaviour to the virtual world without any scripting, simply by assigning pre-defined behaviours to scene elements. Using the Virtools SDK, new behaviours can also be implemented. Virtools exposes so much functionality that fairly sophisticated games can be created and played using this system. Given this rich functionality, it is likely that it is possible to author experiences with significant control over the viewpoint and pacing as well as content and lighting with Virtools Dev and the SDK. The drawback is that such experiences will be very difficult to author well since in addition to the usual authoring difficulties, the author must reckon with programming behaviours. Indeed there is no clear conceptual model of how an author should constrain the viewpoint and how various constraints will relate to each other. Moreover, there is no established user interaction metaphor that the author can design his or her experiences to work with. Nevertheless, experiences created with Virtools Dev are better and more impressive than any experiences we were able to find authored using other systems. Also, we found that Virtools was the only product available that supported author influenced viewing at all. Given this fact, we consider Virtools experiences to be the state-of-the-art in interactive 3D authoring and viewing.

A large amount of academic research has been done on the general topics of 3D scene navigation and camera control techniques. The following section will explore the techniques developed in past research to determine how they deal with author control versus user control and what level of support, if at all, these techniques have for author control over what the user sees and when they see it.

Perhaps the most ubiquitous mouse-based virtual camera control metaphor, the cinematic camera metaphor, enables users to tumble, track and dolly the viewpoint with a regular mouse and modifier keys. This metaphor is used as the primary view control in most 3D modeling and animation software packages and works very well in the majority of cases. Various other metaphors have been explored by researchers, including orbiting and flying (Tan et al., 2001), through-the-lens control (Gliecher & Witkin, 1992), points and areas of interests (Jul & Furnas, 1998), using constraints (Mackinlay et al., 1990), drawing a path (Igarashi et al., 1998), two-handed techniques (Balakrishnan & Kurtenbach, 1999; Zeleznik et al., 1997), and combinations of techniques (Steed, 1997; Zeleznik & Forsberg, 1999).

Tan et al. (2001) integrate speed-coupled flying and object-centric orbiting to allow the user to not only alternate between local views or "walking around" and global overviews of the world, but to also orbit around a central object of interest in order to examine it. The intention of the user is assumed when they first start to drag the mouse. If the mouse drag begins on an object, the system assumes the user is trying to examine the object, whereas if the drag begins in empty space (i.e. the sky or the ground), the system assumes the user wants to freely navigate. When in navigation mode, the user is given standard controls to navigate. An interesting aspect of this scheme is that the height of the camera and its tilt are determined by the speed of navigation so that the faster the user's view is moving, the more the user experiences a zoomed-out or overview effect. The authors showed that that this technique was generally superior in performance and preferred by users in comparison to several other techniques.

Gleicher and Witkin (1992) propose a series of camera controls that let a user control the virtual camera by manipulating and applying constraints to features in the image as seen through the virtual camera's lens. For example, points on objects in the scene can be pinned to a certain screen location, constraining the virtual camera to always keep those two 3D points at the same location on the screen. Image features can also be used as virtual camera controllers, for example dragging a point in the scene causing the camera to move such that this point always stays under the cursor. Multiple constraints and controls can be used simultaneously to operate the virtual camera.

Mackinlay et al. (1990) describe a technique that supports rapid and controlled movement of the virtual camera through space. The user simply chooses a target "point of interest" (POI) on an object in the scene, and the virtual camera moves towards this target logarithmically in distance to target. The POI is dynamic and is recalculated every frame, after which the virtual camera is adjusted to face the POI. Due of the logarithmic movement of the viewpoint, rapid motion is achieved when the distance to the target object is large and controlled movement is achieved as this distance shrinks.

Igarashi et al. (1998) introduce a technique for navigating virtual worlds that uses user-drawn strokes to control the virtual camera. A stroke is projected onto the ground surface in the scene and used as a path for the virtual camera. The stroke can be updated at any time by drawing a new stroke. The authors showed that the technique was actually slightly slower on average than other techniques such as "driving" where the user controls the camera with keyboard arrow keys, and "flying" where the user clicks on a point on the ground causing the virtual camera to fly to that location. The same study also showed, however, that the drawing technique is preferred by most users.

Some research has also attempted to categorize navigation metaphors in various ways. Bowman et. al. (1997, 1999) present action- or motor-based taxonomies and evaluations of various schemes in the context of virtual immersive environment. Tan et al. (2001) propose an alternative, task-based navigation metaphor taxonomy. Other techniques involve automatic framing of the areas of interest as typically found in game console based adventure games which use a ”chase airplane” metaphor for a third person perspective. Systems that utilize higher degree-of-freedom input devices offer additional control and alternative metaphors have been investigated, including flying (Chapman & Ware, 1992; Ware & Fleet, 1997), eyeball-in-hand (Ware & Osborne, 1990), and worlds in miniature (Stoakley et al., 1995).

The major difference between this body of prior research and our work is that we attempt to give the author substantially more influence over the types of views and transitions between them as the user navigates in the virtual space. Indeed none of the previously mentioned camera control metaphors support author designed, substantial viewpoint and pacing influence, and hence an author is scarcely able to introduce stylistic elements to the experience in the traditional, well-established ways.

Beyond techniques for navigating the scene, extra information can also be provided to aid navigation. These include global maps in addition to local views (Elvins et al, 1998; Fukatsu et al., 1998), and various landmarks (Darken & Sibert, 1996; Vinson, 1999). Others have investigated integrating global and local views, using various distorted spaces including “fisheye” views (Carpendale & Montagnese, 2001; Furnas, 1986).

Approaches that give the author more influence include guided tours where camera paths are pre-specified for the end user to travel along. Galyean (1995) proposes a “river analogy” where a user, on a metaphorical boat, can deviate from the guided path, the river, by steering a conceptual “rudder”. A great strength of this technique is that the metaphor is both very simple to grasp

conceptually and fairly powerful. Since the boat flows continuously down the river with or without user input, the river metaphor guarantees an uninterrupted flow for the user. The fact that the river is like a path in the scene with some varying width means that the author can design a path that gives visually pleasing shots of the virtual world. The author also has control over the rate of flow of the river at any point along the river and can use this to influence the pacing of the experience. The virtual river can have multiple branches that can optionally rejoin, allowing for branching visual narratives, which could be more interesting to the user. The river analogy is not without its drawbacks. The types of experiences that can be authored using this metaphor are limited to ones that follow a branching path. Also, the amount of author influence of the viewpoint is not as substantial as we would like to see. Although the author can strictly determine the path the viewpoint will roughly follow, the viewing direction is only partially and vaguely influenced through parameters, such as spring force, boat speed and view direction force, which can be specified along the river. It is not immediately clear as an author what viewpoints will be experienced as a user, a fact that could be too limiting for certain applications.

Fundamental work by Hanson and Wernert (1997, 1999) proposes "virtual sidewalks" which are authored by constructing virtual surfaces and specifying gaze direction, vistas, and procedural events (e.g., fog and spotlights) along the sidewalk. Our system builds upon the guided tour and virtual sidewalk ideas but differs by providing authoring elements that enable a much more stylized experience. Specifically, we offer a means of presenting 3D, 2D, and temporal media experiences through a simple, unified, singular user interaction technique that supports both spatial and temporal navigation. While the system presented by Hanson and Wernert focuses on constrained navigation, our system addresses authored interactive experiences. Although StyleCam's camera surfaces borrow heavily from Hanson and Wernert's "constraint surfaces", they differ on several levels. In Hanson and Wernert's system, one large, often complex constraint surface made up of one or more rectangular patches is used to constrain the camera.

Camera parameters are defined at key points along this surface to influence the view. Although the constraint surface is most often used to control camera position (i.e. map 2D controller to 3D spatial position), it can be used to control other camera parameters (e.g. view direction). In addition to deriving camera parameters from the constraint surface and a set of predefined key values, camera parameters can be derived from other sources such as scene interest points or terrain gradients. In contrast, our system uses multiple simple camera surfaces to constrain the camera position only, and only one money-shot (equivalent to Hanson and Wernert's "constant key vertices") influences the other camera parameters at any given time.

Robotic planning algorithms have been used to assist or automatically create a guided tour of a 3D scene, in some case resulting in specific behaviours trying to satisfy goals and constraints (Drucker et al., 1992; Drucker & Zeltzer, 1994). Individual camera framing of a scene has been used to assist in viewing or manipulation tasks (Phillips et al., 1992). Cinematic principles such as keeping the virtual actors visible in the scene; or following the lead actor can be applied for the camera to automatically frame the scene (He et al., 1996). Yet another system (Bares et al., 2000) allows authors to define storyboard frames and the system defines a set of virtual cameras in the 3D scene to support the visual composition. This previous work assists in the authoring aspects by ceding some control to the system. Our work too involves some automatic system control, but we emphasize author control.

Image based virtual reality environments such as QuicktimeVR (Chen, 1995) utilize camera panning and zooming and allow users to move to defined vista points. The driving metaphor has also been used for navigating interactive video, as seen in the Movie-Maps system (Lippman, 1980). More recently, the Steerable Media project (Marrin et al., 2001) for interactive television aims to retain the visual aesthetic of existing television but increase the level of user interactivity. The user is given the ability to control the content progression by seamlessly integrating video with augmented 2D and 3D graphics. While our goals are similar in that we hope to enhance the

aesthetics of the visual experience, we differ in that our dominant media type is 3D graphics with augmented temporal media (i.e. animations and visual effects) and traditional 2D media (i.e. video, still images).

Chapter 4. Conceptual Model

In this chapter, we describe the concepts of StyleCam that make the authoring and viewing of interactive, stylized viewing experiences possible. The core of this innovation is in establishing a flexible and easily understood conceptual model for both the author and the end-user. Designing and implementing a usable and highly functional authoring GUI was not a goal in the development of StyleCam. We will therefore continue with a description of the StyleCam conceptual model and how it empowers an author to create good interactive 3D experiences.

StyleCam is a conceptual model that allows an author to build interactive experiences by being flexible enough to be highly expressive, yet simple enough so that the experience the user will get is predictable and true to the author's intentions. StyleCam also defines how the user will interact with the system, and how the system will interpret the actions of the user to move the camera.

In order to provide author control or influence over viewpoints and pacing, mechanisms must exist for an author to express the viewpoints and the types of pacing they are interested in. These are:

1. Camera surfaces – an author-created surface used to constrain the users' movement of the viewpoint.

2. Animation clips – an author-created set of visual sequences and effects whose playback may be controlled by the user. These animation clips can include:

    - sophisticated camera movements,
    - Slates – 2D media such as images, movies, documents, or web pages,
    - visual effects such as fades, wipes, and edits,
    - animation of elements in the scene.

3. Unified UI technique – The user utilizes a single method of interaction (dragging) to control the viewpoint, animation clips, and the transitions between camera surfaces.

In the following sections, each of these components are described in more detail.

4.1 Camera Surfaces

In the motion picture industry a *money-shot* is a shot with a particular viewpoint that a director has deemed "important" in portraying a story or in setting the visual style of a movie. Similarly, in advertising, money-shots are those which are the most effective in conveying the intended message. We borrow these concepts of a money-shot for our StyleCam system. Our money-shots are viewpoints that an author can use to broadly determine what a user will see. Viewpoint parameters include position, orientation (view direction), lens length (zoom) and others.

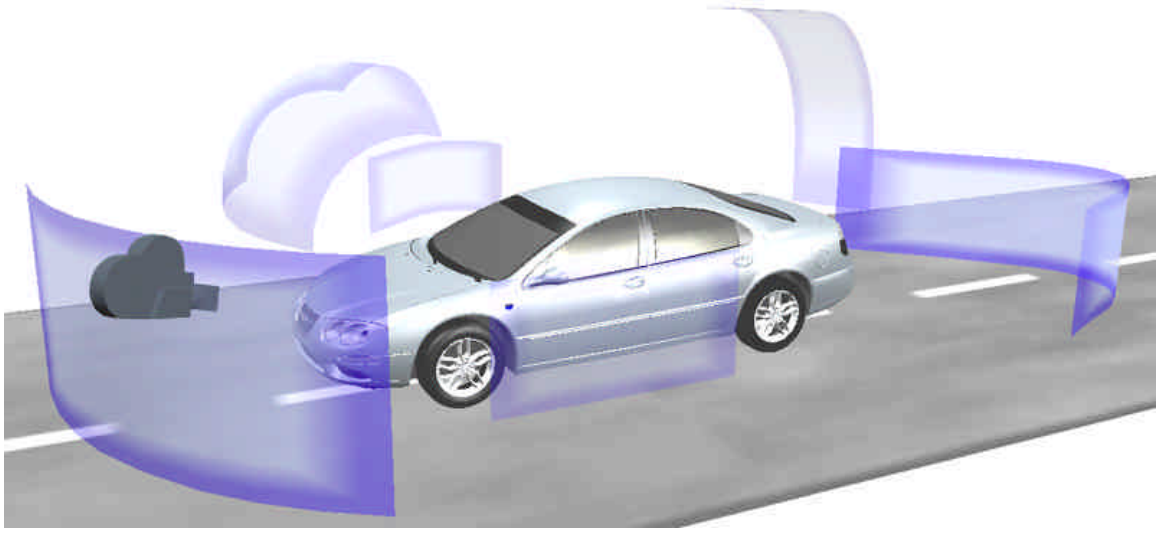

Figure 4.1. Camera surfaces. The active camera is at the money-shot viewpoint on the first camera surface.

Further, we use the concept of a *camera surface* as introduced by Hanson and Wernert (1997, 1999). When on a camera surface, the virtual camera's spatial movement is constrained to that surface. Further, each camera surface is defined such that they incorporate a single money-shot. Figure 4.1 illustrates the concept.

Camera surfaces can be used for various purposes. A small camera surface can be thought of as an enhanced money-shot where the user is allowed to move their viewpoint a bit in order to get a sense of the 3-dimensionality of what they are looking at. Alternatively, the shape of the surface could be used to provide some dramatic camera movements, for example, sweeping across the front grill of a car. The key idea is that camera surfaces allow authors to conceptualize, visualize, and express particular ranges of viewpoints they deem important.

Intrinsic in our authored interactions is the notion that multiple camera surfaces can be used to capture multiple money-shots. Thus authors have the ability to influence a user's viewpoint broadly, by adding different camera surfaces, or locally by adjusting the shape of a camera surface to allow a user to navigate through a range of viewpoints which are similar to a single

particular money-shot. For example, as shown in Figure 4.1, camera surfaces at the front and rear of the car provide two authored viewpoints of these parts of the car in which a user can "move around a bit" to get a better sense of the shape of the front grille and rear tail design.

The rate at which a user moves around on a camera surface (Control-Display gain) can dramatically affect the style of the experience. In order to allow an author some control over visual pacing, we provide the author with the ability to control the rate at which dragging the mouse changes the camera position as it moves across a camera surface. The intention is that increasing/decreasing this gain ratio results in slower/faster camera movement and this will influence how fast a user moves in the scene, which contributes to a sense of pacing and visual style. For example, if small mouse movements cause large changes in viewpoint this may produce a feeling of fast action while large mouse movement and slow changes in movement produce a slow, flowing quality. Figure 4.2 illustrates an example of variable control-display gain, where the gain increases as the camera gets closer to the right edge of the camera surface.

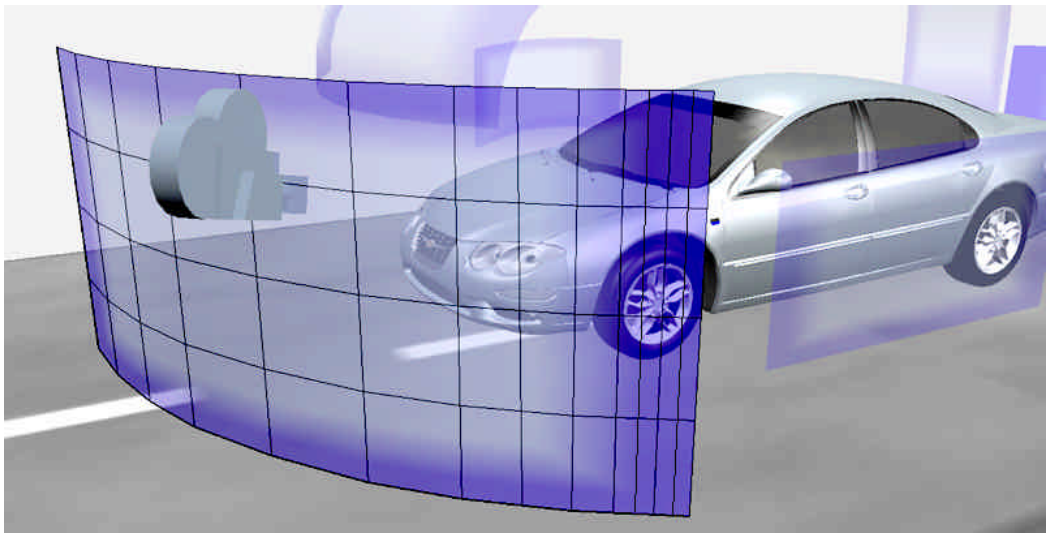

Figure 4.2. Variable control-display gain on a camera surface.

4.2 Animation Clips

To support transitions between two camera surfaces, we use animation clips as illustrated in Figure 4.3. An animation clip can be thought of as a "path" between the edges of camera surfaces. When a user navigates to the edge of a camera surface, this triggers an animation. When the animation ends, they resume navigating at the destination camera surface. One obvious type of animation between the camera surfaces would simply be an automatic interpolation of the camera moving from its start location on the first camera surface to its end location on the second camera surface (Figure 4.3a). This is similar to what systems such as VRML do. While our system supports these automatic interpolated animations, we also allow for authored, stylized, animations. These authored animations can be any visual sequence and pacing, and are therefore opportunities for introducing visual style. For example, in transitioning from one side of the car to the other, the author may create a stylized camera animation which pans across the front of the car, while closing in on a styling detail like a front grille emblem (Figure 4.3b).

The generality of using animation clips allows the author the stylistic freedom of completely abandoning the camera-movement metaphor for transitions between surfaces and expressing other types of visual sequences. Thus animation clips are effective mechanisms for introducing *slates* — 2D visuals which are not part of the 3D scene but are momentarily placed in from of the viewing camera as it moves from one camera surface to another (Figure 4.3c). For example, moving from a view of the front of the car to the back of the car may be accomplished using a 2D image showing the name of the car. This mechanism allows the use of visual elements commonly found in advertising such as real action video clips and rich 2D imagery. In the computer realm, slates may also contain elements such as documents or web pages.

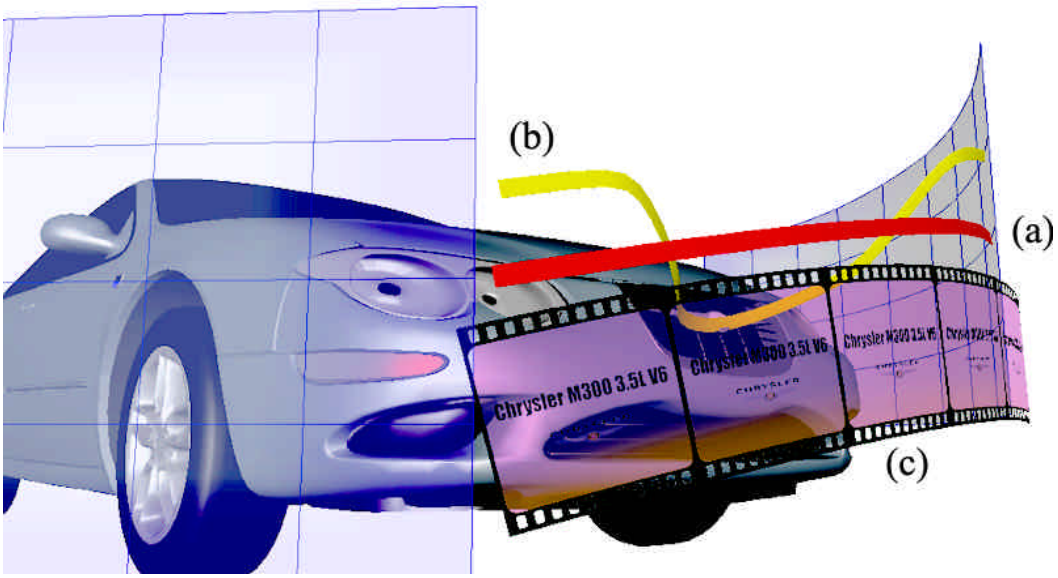

Figure 4.3. Three example animated transitions between camera surfaces. (a) automatic transition, (b) authored stylized transition, (c) slate transition.

The use of animation clips also allows for typical visual transitions effects such as cross fades, wipes etc.

In addition to using animation clips for transitions between camera surfaces, StyleCam also supports the animation of elements in the 3D scene. These scene element animations can occur separately or concurrently with transition animations. For example, while the animation clip for the visual transition may have the camera sweeping down the side of the car, an auxiliary animation may open the trunk to reveal cargo space.

The animation of scene elements can also be used to affect extremely broad changes. For example, entire scene transitions (similar to level changes in video games) may occur when a user hits the edge of particular camera surface.

At the author's discretion, temporal control of animation clips can either be under user control or uninterruptible. The ability to designate an animation clip as uninterruptible is useful for many

reasons. It shifts the balance of control in favour of the author. An uninterruptible animation clip is essentially a movie sequence and therefore offers the ultimate in author control. Furthermore, an uninterruptible clip may be preferred when the distance covered is small or the duration very short, in order to ensure that no important features are missed. Finally certain types of animation clips are required to be uninterruptible to deliver the intended message. For example an animation sequence involving multiple frames of text may only be intelligible if played in the forward direction.

Overall, in terms of visual expression, these varying types of animation clips allow an author to provide rich visual experiences and therefore significantly influence the pacing and style of a user's interaction.

4.3 Unified User Interaction Technique

While animation clips are effective for providing a means to move between camera surfaces and introduce visual styling elements, they also highlight the fundamental issue of arbitrating between user control and system control. At the heart of our system are two distinct types of behaviour: 1) user control of the viewpoint, and 2) playback of animation clips. In other systems these two types of behaviour are treated as distinct interactions. Specifically, the user must stop dragging the camera viewpoint, then click on something in the interface to trigger the animation, dividing their attention and interrupting the visual flow. In our system we wanted to use animations as a seamless way of facilitating movement between camera surfaces. Thus we needed a mechanism for engaging these animations that did not require an explicit mouse click to trigger animation. Ideally we wanted to leave the user with the impression that they "dragged" from one camera surface to another even though the transition between the surfaces was implemented as an authored animation.

These two behaviours are fundamentally different in that viewpoint control is spatial navigation and animation control is temporal navigation. From a user interaction standpoint, spatial behaviour can be thought of as “dragging the camera” while temporal control is “dragging a time slider” or “scrubbing”. Given this we required an interaction model which allowed these two types of drags to be combined together in a way that was well defined, controllable, and corresponded to user’s expectations.

Figure 4.4, which uses the finite-state-machine model to describe interaction as introduced by Buxton (1990) and Newman (1968), shows the interaction model we developed. The key feature of this model is *the ability to transition back and forth from spatial to temporal control during a contiguous drag.* As a user drags the camera across a camera surface (State 1, Spatial Navigation) and hits the edge of the surface, a transition is made to dragging an invisible time slider (State 2, Temporal Navigation). As the user continues to drag, the drag controls the location in the animation clip, assuming that the author has specified the clip to be under user control. Upon reaching the end of the animation, a transition is made back to dragging the camera, however, on a different, destination camera surface (State 1).

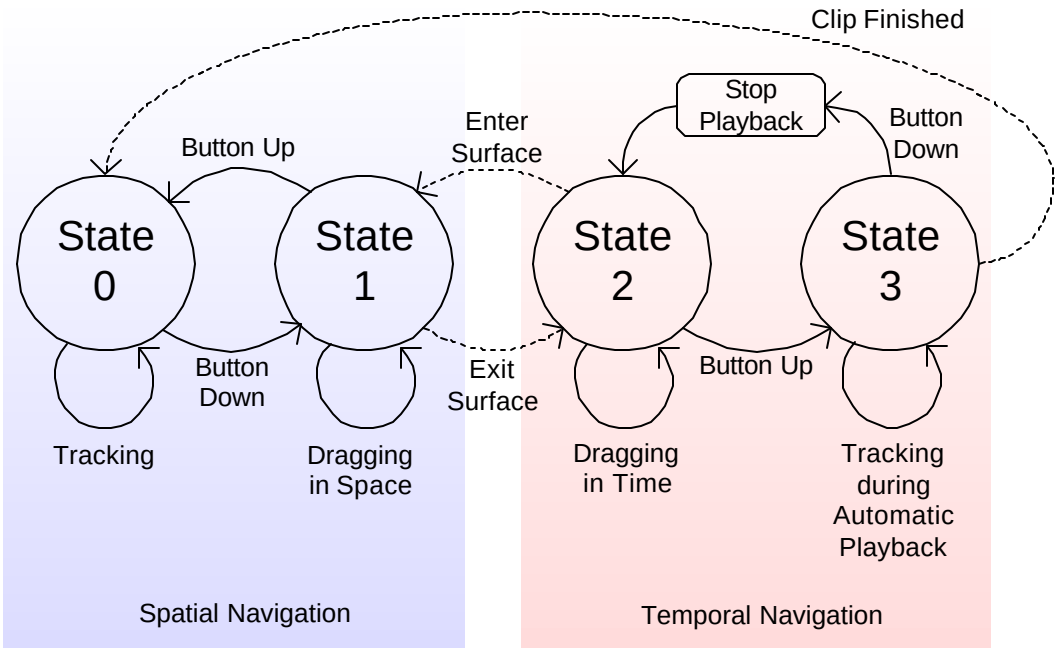

Figure 4.4. StyleCam interaction model.

The interaction model also handles a variety of reasonable variations on this type of dragging behaviour. A user may stop moving when dragging an animation clip, thus pausing the animation. If, however, when in State 2 the user releases the mouse button during a drag, automatic playback is invoked to carry the user to the next camera surface (State 3). Should the user press the mouse button during this automatic playback, playback is stopped and temporal control by the user is resumed (return to State 2). We found in practice that this interaction design enhanced the user's feeling of being in control throughout the entire experience.

Chapter 5. Prototypes

In this chapter, we describe the previous incarnations of StyleCam to illustrate the iterative process that was used in the development of StyleCam. Furthermore, describing earlier StyleCam prototypes will demonstrate why certain features or concepts were added. For example, at first glance, it may appear that the incorporation of animation clips into StyleCam unnecessarily complicates its authoring and use. After all, without animated transitions, we would not have had to develop an interaction technique that blended between spatial and temporal control. Indeed, when we first began our research, our hope was to create a system that simply involved spatial control of a constrained camera.

The idea of StyleCam was sparked by the observation that tumbling the camera is equivalent to sliding the camera along the surface of a sphere surrounding a central object of interest. The first StyleCam prototype extended this concept to sliding the camera along an arbitrary closed surface surrounding the 3D object of interest. The camera view direction was constrained to remain normal to this single camera surface at all times. While this gives the author more control over what the users sees than using a simple unconstrained camera, we found that it was difficult to author a single camera surface that encompassed all the desirable viewpoints and interesting transitions between those viewpoints. For example, with such a scheme, it is impossible create a

camera surface that allows the camera to move along a straight path while pointing in the direction of movement. The camera is always pointing in the direction of the surface normal which is, by definition, perpendicular to the surface itself (the surface defines the direction of movement). The concept of a camera surface was nevertheless not abandoned since a surface is such a natural way for an author to describe a constrained set of possible camera positions in space. It is trivial to visualize and predict the range of positions in space that a surface represents. Furthermore, a surface provides a simple mapping between two-dimensional mouse movements and a position on the surface.

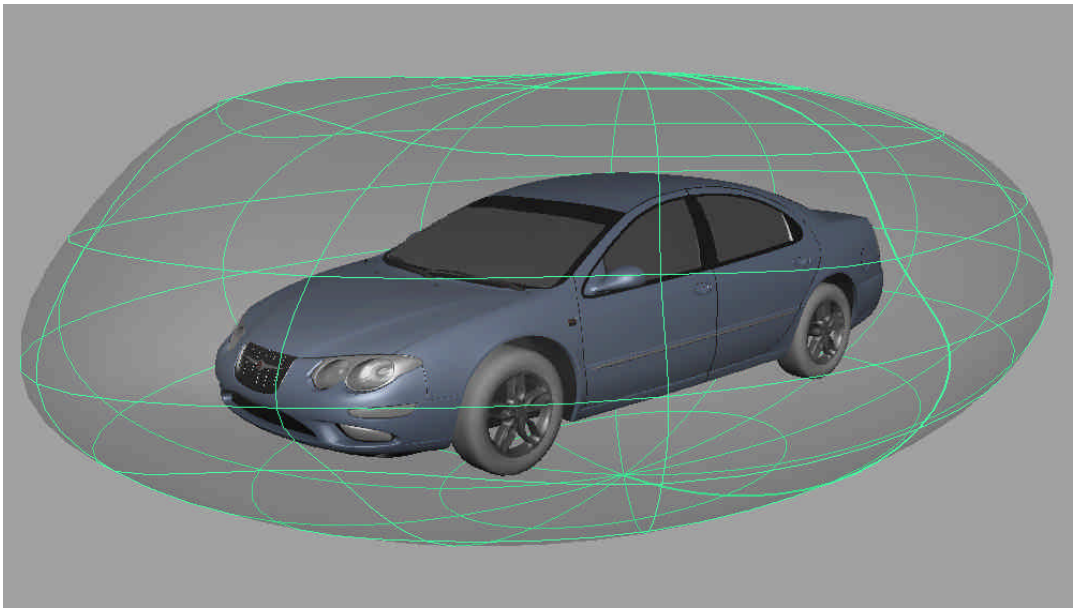

Figure 5.1. Car model surrounded by a single camera surface.

In order to guarantee desirable viewpoints over all parts of the surface, we introduced the concept of authored money-shots that were placed on the single camera surface. Each money-shot defined the camera parameters at a specific point on the camera surface. For points on the camera surface between money-shots, the camera surface itself defined the camera position, while other camera parameters were determined based on a weighted average of the surrounding money-shots. This approach removed the limitation of having the camera always point perpendicular to its direction of movement, yet it was still difficult to for the author to predict what the user would see when

between money-shots. In other words, while money-shots worked well, the transitions between them worked poorly. The problem was compounded by the fact that averaging camera parameters between money-shots necessarily implies that the addition of a money-shot will change the computed average camera parameters at all non money-shot locations on the surface. This second prototype was only marginally more successful than the first.

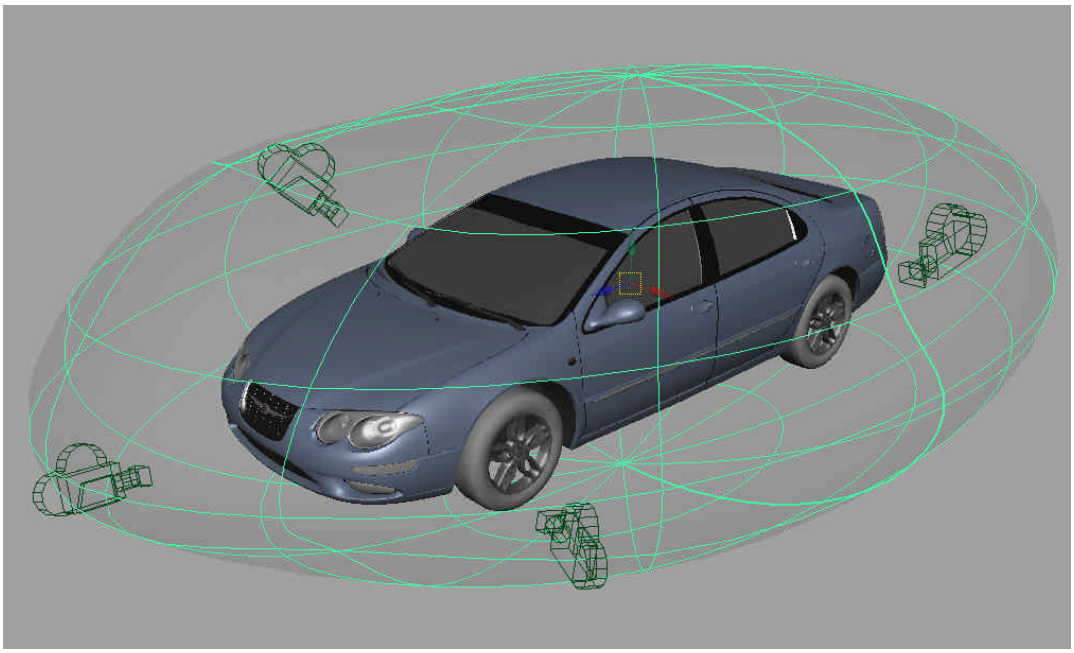

Figure 5.2. Car model surrounded by a single camera surface embedded with four money-shots.

The third prototype was developed after re-examining the basic issues. What is StyleCam trying to accomplish? The simple answer is interactive 3D experiences as compelling as television commercials. The observation was made that in a car commercial, you don't see every single detail on the car, but rather you see a carefully selected set of details from very specific viewpoints. These viewpoints are captured by the concept of money-shots, but money-shots alone offer no room for interactivity. It is the camera surface that adds the interactive component. To address the problem of unsatisfactory transitions, we first replaced the concept of a single global camera surface with separate local camera surfaces for each money-shot. Then, to define

transitions between these local camera surfaces, we introduced the idea of animating the camera with the use of the three types of animation clips described earlier. The user's viewpoint was now being influenced by exactly one money shot at any given time, giving more predictable, and therefore easier to author experiences. Each money-shot and camera surface combination is self contained, with no interactions between adjacent or nearby money-shots, completely eliminating the problems we experienced of having unpredictable results when averaging.

In the third prototype of StyleCam, all animated transitions whether automatic, authored or slate-style, were uninterruptible. Simply playing back the animation clips between camera surfaces, however, made users feel that they had completely lost control during this period. To maintain the feeling of continuous control throughout, we developed our integrated spatial-temporal interaction technique which we described in detail in the previous chapter.

Chapter 6. Example Experience

6.1 StyleCam in Context

In prior chapters, we discussed StyleCam in the context of the tools and concepts it replaces. In this chapter, we discuss how StyleCam fits with the roles of the various people involved in the production and visualization process. In chapter 4, we commented that the development of StyleCam was not meant to produce a useable tool but rather to materialize the concepts needed to produce compelling interactive experiences. In doing so, we implied that a single tool would support all the roles in the production and visualization pipeline, except perhaps the viewer. This is not a realistic picture. The production of a StyleCam experience, in practice, would involve at least the roles of:

Modeller	Generates the 3D models.
Animator	Animates the 3D geometry (generates transitions and other animations).
Director of photography	Directs lighting, framing, pacing of experience.
Graphic designer	Designs and generates the 2D media (slates).
Script writer	Writes the "script" for the StyleCam experience.

Art director	Provides high-level direction of the StyleCam experience.
Layout designer	Designs the environment in which the StyleCam experience is to be presented.
Market researcher	Collects data from users, gives feedback to all other levels.
Viewer	Experiences the StyleCam experience.

These roles are, in the most part supported by existing, well established tools. For example a variety of sophisticated 3D modeling software packages (MAYA, 3DS Max, etc..) are in use today by thousands of professional 3D modellers. Similarly, powerful graphic design tools have been around for many years. StyleCam should play along with these tools, augmenting and complimenting their features to make the creation of StyleCam experiences simple. To illustrate how this might be done, we step through an envisioned production pipeline for a StyleCam experience of an automobile.

Marketing requests the development of a StyleCam experience for the company's newest model. Marketing will likely have ideas about who the target audience is and how they want to market the car, which they communicate to the production team. Given the input from the marketing team, the Art director and script writer will begin to work on the script and storyboard (likely with the help of a storyboard writer) for the experience - that is the sequence of money-shots and transitions including some rough framing. The tools used so far may include no more than paper, pencils, tape and a large wall.

Two-dimensional content creation is assigned to the graphic designer who uses, among other things, a digital camera, stock photos of the car and Photoshop. Three-dimensional content is created in MAYA, likely adapted from existing 3D models of the car (e.g. reduce complexity to optimize rendering). The director of photography "shoots" the StyleCam experience, working with a 3D lighting and rendering artist and an animator and using the script as a reference. It is

during this step that the shots set up during the scripting and storyboarding are realized. Here, a suite of plug-ins, geared to assist in StyleCam-specific activities is of great use. For example, a camera surface tool could generate camera surfaces automatically based on the money-shot and instructions from the director (pacing, suitable camera movements. etc..). A transition tool would automate the creation of the appropriate transition data based again on instructions from the director (e.g. pull-out while flying quickly from camera surface A to camera surface B. A StyleCam preview tool would be used to "try out" the camera surfaces and transitions.

If this were an example of the production of an animated commercial, the output at this point would be a large number of rendered frames which would be subsequently sent to post-production for final compositing. In our example, the output is a StyleCam experience which will be played or viewed by a customer. Specialized StyleCam tools are therefore necessary to get from several lit, framed (including camera surfaces, transitions and animations) MAYA scenes and a collection of 2D media to a workable StyleCam experience. This includes the sequencing of money-shots and transitions, and the assignment of animations to events. This step can be viewed as the editing phase of the experience.

The layout designer is responsible for setting the environment in which the experience will be viewed. This includes choosing the size and type of display device, the environmental styling, layout and lighting, and the input device among other things.

Finally, the user views the StyleCam experience using specialized StyleCam viewing software running on a powerful computer with an advanced 3D graphics processor. The StyleCam viewer plays the experience based on the input of the user. In addition, the viewer collects data about the experience such as the paths the user chose and the time spent inspecting various features. This data is collected by the Marketing team which feeds back to all the other players.

To summarize, StyleCam is right now a set of concepts and some rough prototype-quality tools to test these concepts. The use of StyleCam in practice will required the development of a variety of specialized StyleCam tools to complement the existing tools used by professionals. The combination of these new tools with the established tools will simplify the workflow of creating a StyleCam experience in practice.

6.2 An Example Experience

In the previous section we described an example authoring production pipeline for StyleCam experiences. In this section, we illustrate how StyleCam operates by an example from the point of view of a user. This example is taken directly from an experience we authored using the prototype StyleCam authoring system, and tested in our informal user study of StyleCam. The purpose of this example is to show, to the extent that it is possible with still images, what the user experiences. The experience in this example is meant to be an advertisement for an automobile. It was authored to have a particular look and feel of being mysterious and elegant.

Figure 6.1 illustrates the system components and how they react to user input, as well as screen shots of what the user actually sees. The user starts by dragging on a camera surface (position A). The path A-B shows the camera being dragged on the surface (spatial navigation). This camera surface allows the user to explore the front-end of the car including the grille and the headlights. This particular camera surface not only constrains the user's viewpoint to specific angles of the car's front-end, it also enforces a particular focal length for the shot, giving the evocative wide-angle perspective.

At B, the user reaches the edge of the camera surface and this launches an animation that will transition the user from B to E. The zigzag path from B to D indicates that the user is scrubbing time on the animation (temporal navigation). It should be noted that the transition from spatial to temporal navigation is completely seamless. If the user never releases the mouse button during

the transition, they will likely not even notice that they have made the transition. Position C simply illustrates an intermediate point in the animation that gets seen three times during the interaction.

At position D, the user releases the mouse button, whereupon the system automatically completes playing back the remainder of the animation at the authored pacing. The change from user control and system control is smoothly blended such that the playback does not abruptly change speeds to match the authored pacing. Once the animation clip is complete, at position E, the user enters another camera surface and resumes spatial navigation of the camera. It should be noted that the end-point of the animation clip is framed very precisely to give the shot shown. By taking advantage of this, an author can end transition on important viewpoints for emphasis.

The path E-F shows the movement of the camera along the camera surface corresponding to the input from the user. When the user's camera exits this camera surface at position F, another animation is launched that will transition the user to position J. Since the user releases the mouse button at position F, the animation from F to J is played back at the authored pacing. Since this animation is a slate animation, the intermediate shots at positions G, H, and I along the path from F to J are of slates containing information on the car fading in and out as the camera pans over the top of the car.

Though this example is only a fraction of the whole authored experience, it is representative of the level of author control made available by StyleCam, and the quality of the experiences that can be produced. The net result of this StyleCam experience is a view of the car that is far more visually rich and influenced by an author who intends to convey a certain message, rather than using simple camera controls as is typical in current 3D viewers. Indeed, simply tumbling around the same automobile model in MAYA was much less impressive.

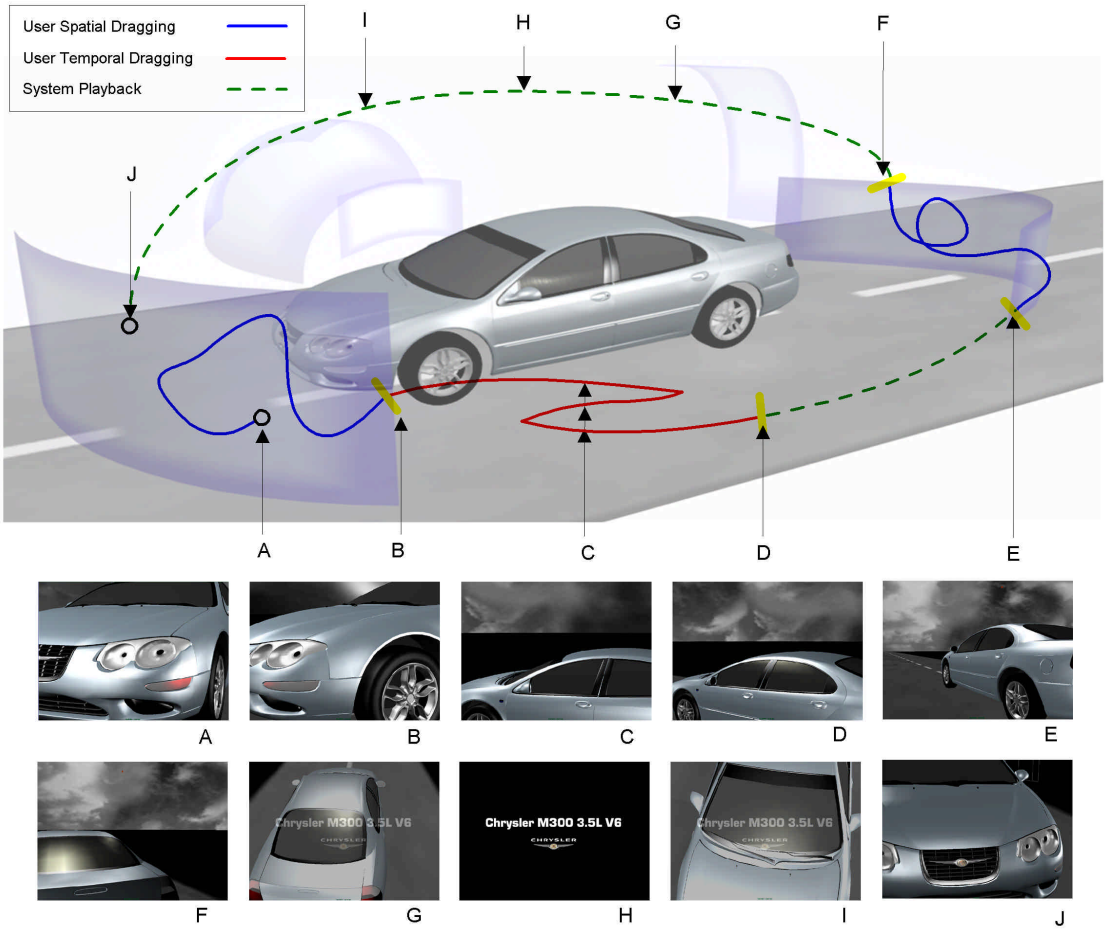

Figure 6.1. Example StyleCam experience.
Top: system components and their reaction to user input. Bottom: what the user sees.

Chapter 7. Implementation

In this chapter, we describe the implementation details of a prototype of the StyleCam concepts. Ultimately, for production use, a specialized authoring environment is desired, likely comprised of a suite of tools that includes existing 2D and 3D applications and new customized StyleCam specific tools. For the purposes of our research, however, we built the StyleCam prototype using Alias' MAYA 3D modeling and animation package as the foundation. MAYA was used to author the 3D content to be experienced, the required camera surfaces, animation clips, and required associations between them. A custom written plug-in allows the user to control their view of the 3D content based on their mouse input and the authored camera surfaces, animation clips, and associations within the MAYA environment.

The following description of our implementation assumes some knowledge of MAYA, although we have endeavoured to be as general as possible without sacrificing accuracy.

7.1 Authoring

7.1.1 Money-Shots and Camera Surfaces

Money-shots are created by defining a MAYA camera with a specific position, orientation, and other standard camera parameters. Then, a camera surface intersecting the position of the money-

shot camera is defined by creating an appropriate non-trimmed NURBS surface within MAYA (see Figure 7.1). This camera surface defines the set of possible user viewpoint positions in space. To include an optional camera look-at point, the author simply defines a point in 3D space (using a MAYA locator). If defined, the user's view direction will always point to this look-at point (Figure 7.2a). To obtain the effect of a constant view direction at all points on the camera surface, the look-at point can be placed at a great distance from the camera surface (Figure 7.2b). If no camera look-at point is defined, the user's view will remain normal to the camera surface at all times (Figure 7.2c). Finally, to make these components easily locatable by the plug-in, they are grouped under a named MAYA node within its dependency graph as shown in Figure 7.3.

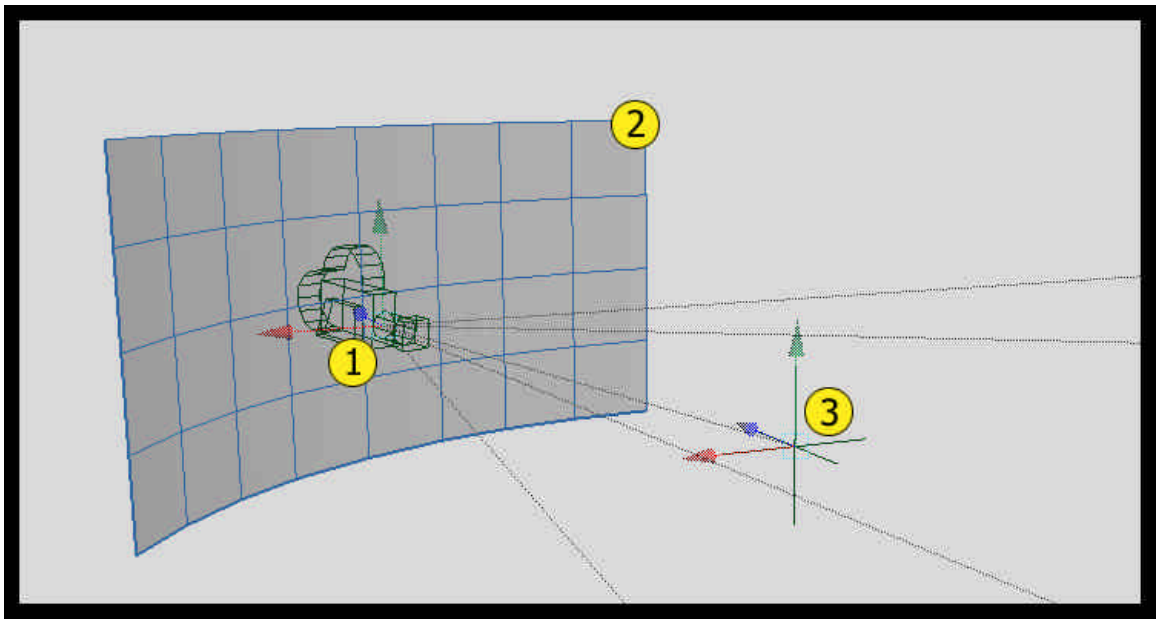

Figure 7.1. A money-shot (1), camera surface (2) and look-at point (3).

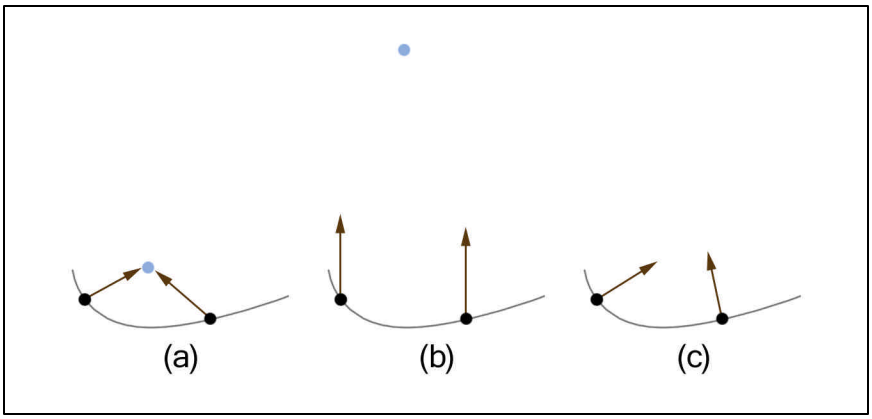

Figure 7.2. Three types of view direction constraints.
(a) local look-at point; (b) distant look-at point giving constant view direction effect; (c) view direction constrained to normal of camera surface

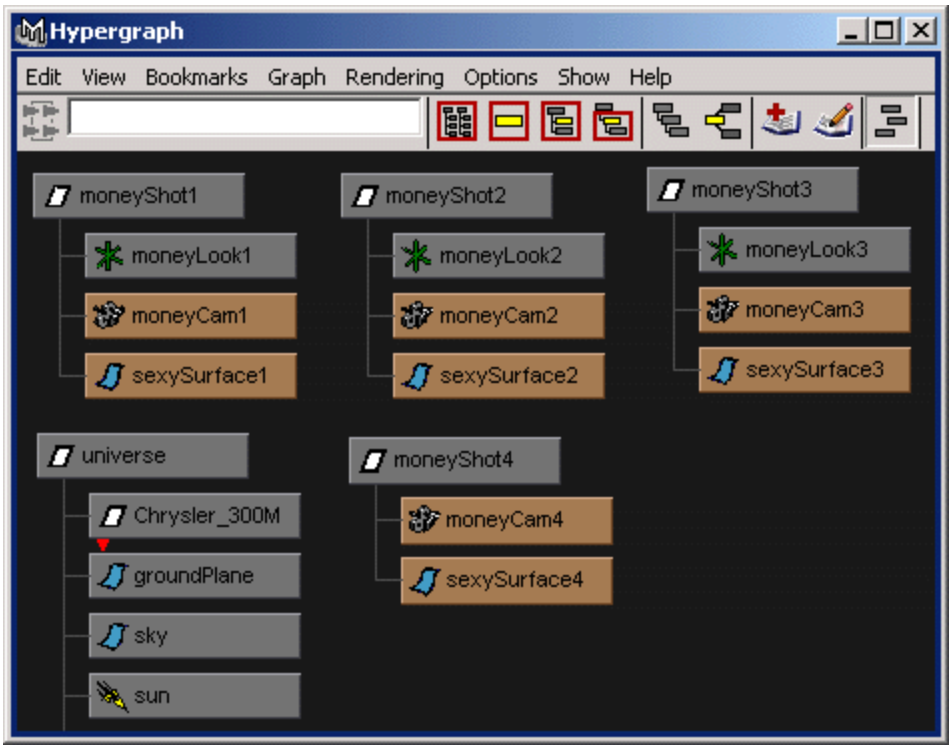

Figure 7.3. Example MAYA dependency graph for a StyleCam scene.

7.1.2 *Animation Clips*

StyleCam animation clips are created by defining sets of animation curves which describe the value of a parameter over time. This is done as one would normally create animations in MAYA, using the TRAX non-linear animation editor (Figure 7.4) and the "Graph Editor" window (Figure 7.5). As a result, the author has the flexibility to use any of MAYA's animation creation tools in the process. For example, animations can be generated by key-framing, physics simulations or expressions.

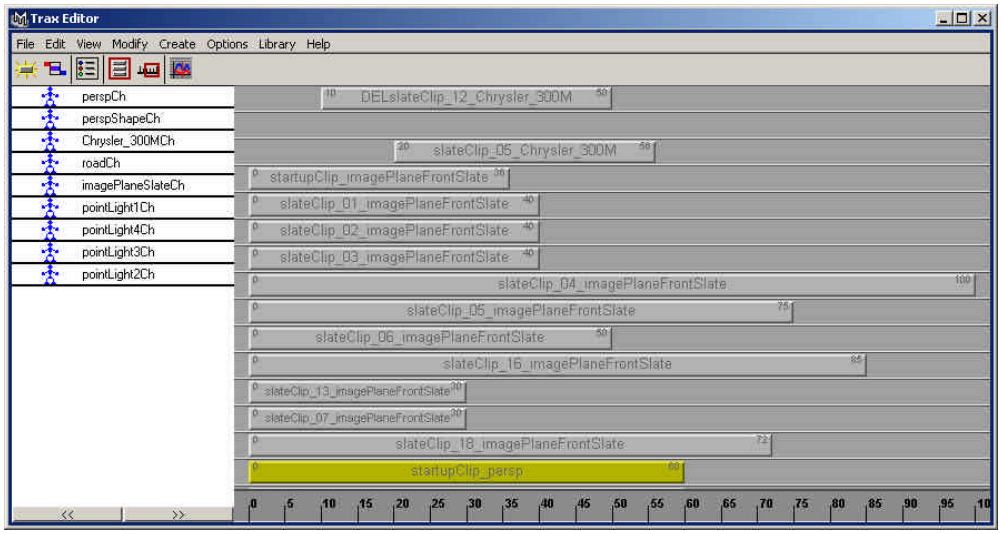

Figure 7.4. MAYA's TRAX non-linear animation editor with several TRAX clips.

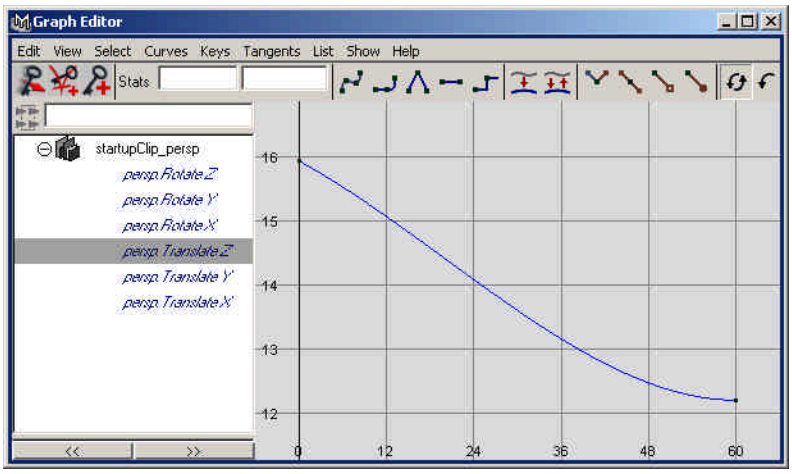

Figure 7.5. MAYA's "Graph Editor" window displaying a curve which describes the camera's displacement over time in the z dimension.

StyleCam animations clips are logically composed of one or more object animations. Each object animation (called *clips* in TRAX) is composed of one or more parameter animation curves. Because TRAX has no inherent support for named groups of object animations, we group related object animations into StyleCam animation clips by naming the object animations with common prefixes.

7.1.3 Events and Scripts

StyleCam allows the author to create scripts and associate them with events. Supported events are session startup, camera surface entry, camera surface exit, and camera surface timeout (Figure 7.6).

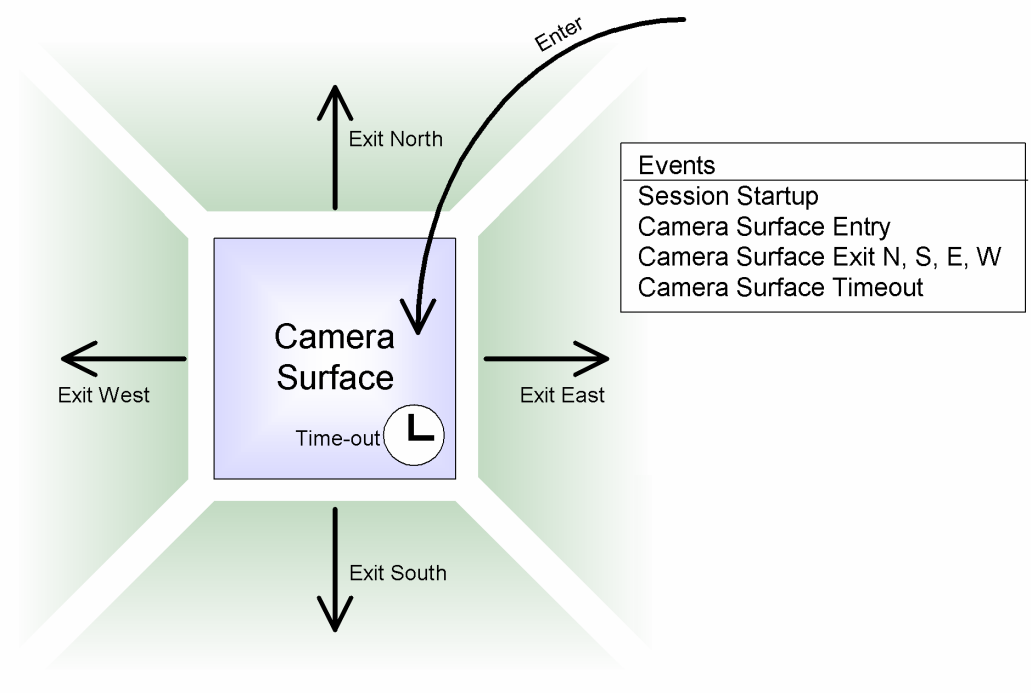

Figure 7.6. StyleCam events.

The session startup event is triggered only once when the user initially begins using StyleCam to view a scene. The session startup script can be used to, for example, trigger an introductory animation clip, or specify which money-shot to start at. Exit events are triggered when the user leaves a camera surface from one of four directions. The associated scripts can specify destination camera surfaces and whether to perform an automatic, slate or authored transition. In the case of authored and slate transitions, the script also identifies the animation clip to be played. Time-out events are triggered when the mouse is idle for a given duration while on a particular camera surface, and can be used, for example, to launch an automatic presentation. StyleCam's event and script mechanism provides for the use of logic to dynamically alter the presentation. For example, scripts can ensure that some surfaces are only visited once, while others are shown only after certain surfaces have already been visited.

7.2 Interaction

7.2.1 Session Startup

When the StyleCam plug-in is activated, the session startup event is fired. If a session startup script exists, it is executed to determine if any animation clips should be played and which money-shot to use as the initial view. If no startup script exists, the first money-shot of the first camera surface is used as the initial view and no animation clips are played. If a look-at point is defined for this initial camera surface, the orientation of the user camera is set such that the camera points directly at the look-at point. Otherwise, the orientation is set to the normal of the camera surface at the money-shot viewpoint's position.

7.2.2 Mouse Movements

User's mouse movements and button presses are monitored by the StyleCam viewing plug-in. Mouse drags result in the camera moving along the current camera surface. Specifically, for a

given mouse displacement (dx, dy), the new position of the camera on the camera surface (in *uv*-coordinates local to the camera surface) is given by

$$(u1,v1) = (u0,v0) + c*(\text{dx, dy})$$

where ($u0$, $v0$) is the last position of the camera, and c is the gain constant. If either the u or v coordinate of the resulting position is not within the range [0,1], the camera has left the current camera surface. At this point, the author-scripted logic is executed to determine the next step. First, the destination money-shot is resolved. Next, an appropriate transition is performed to move to the next camera surface.

We implement variable control-display gain on a camera surface by varying the separation between the isoparms on the NURBS surface. Each isoparm represents an equal step in *uv*-coordinates, so the closer they are together, the less sensitive the mouse becomes in that direction. For example, Figure 7.7 below shows a camera surface with increasingly tight isoparms in the u direction near its right edge. Thus as the camera approaches the right edge of the surface, the control-display gain increases (less sensitive) and the pacing slows. Note that in this example, the reduction in sensitivity only occurs in the u direction. In the v direction, the isoparms are evenly spaced indicating a constant control-display gain.

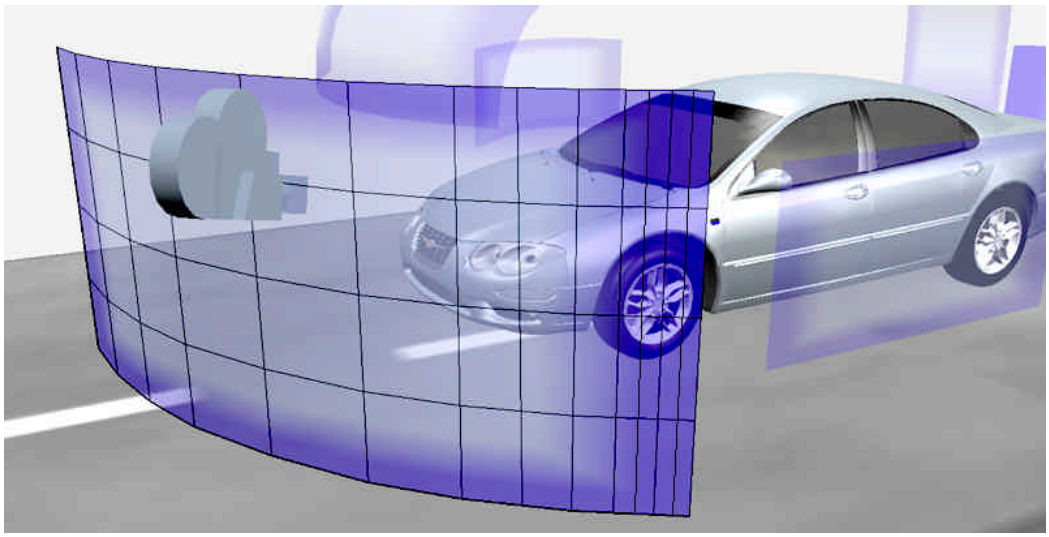

Figure 7.7. Variable control-display gain on a camera surface

7.2.3 Transitions

As shown in Figure 7.8, StyleCam supports three types of transitions: automatic, authored, and slate.

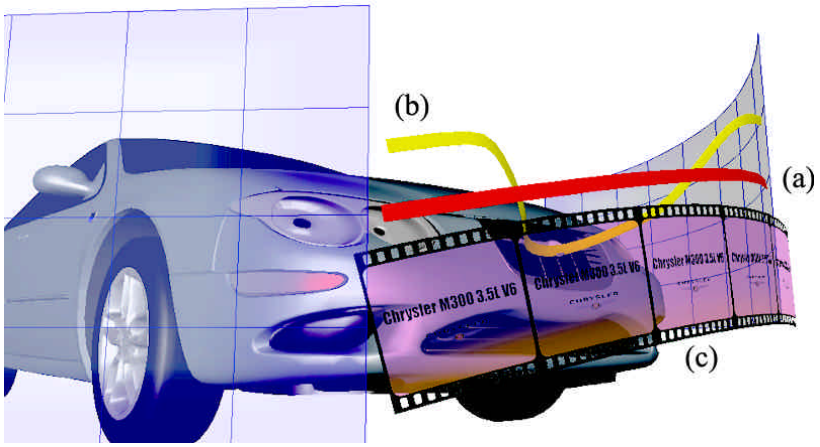

Figure 7.8. Three types of transitions are made available to the author in StyleCam: (a) automatic transitions, (b) authored stylized transitions, (c) slate transitions.

Automatic transitions are those that smoothly move the camera from one camera surface to another without requiring any authored animation clips. Specifically, the user's view is moved from the point of exit from the source camera surface to the position of the money-shot of the

destination camera surface. The system performs quaternion interpolation (Shoemake, 1985) of camera orientation, combined quaternion and linear interpolation of camera position, and linear interpolation of other camera properties such as focal length to complete an automatic transition. Using quaternion interpolation ensures smooth changes in orientation while defining a smooth arcing path for the position. At each time step in the transition, two quaternions representing the required fractional rotations of the position and orientation vectors of the camera are calculated and applied to the source vectors. In addition, the magnitude of the position vector is adjusted by linear interpolation between the source and destination position vector magnitudes. The result is a series of intermediate camera positions and orientations as Figure 7.9 illustrates.

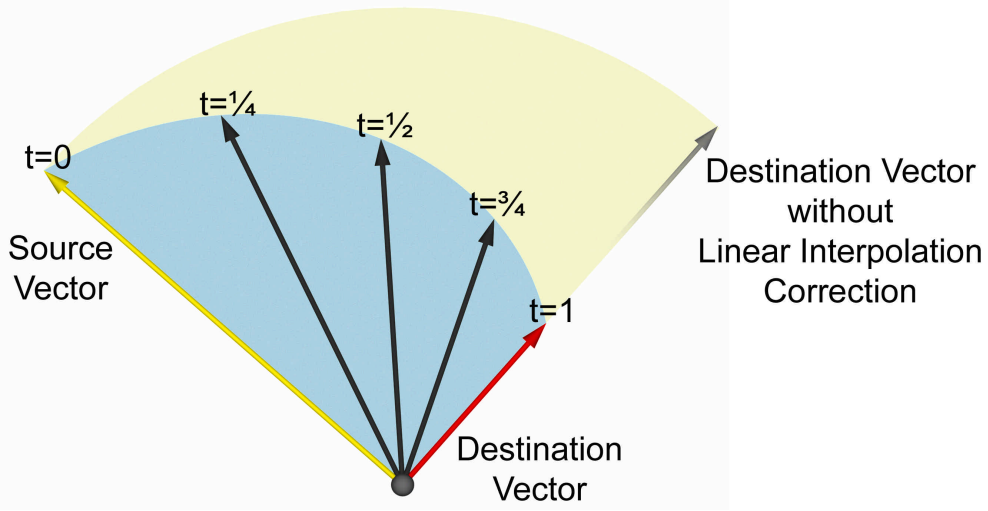

Figure 7.9. Combined quaternion and linear interpolation as applied to the camera position vector.

Authored transitions involve the playback of pre-authored animation clips. Any parameter of any object in the scene can be animated during an authored transition. This includes the user's viewpoint, view direction, other camera parameters, scene elements and special effects helper object parameters such as image plane transparency. When an authored transition is triggered an automatic camera transition is performed in parallel to the playing of the transition's associated animation clips. This allows the author the flexibility to only animate scene elements, letting the

system take care of animating the user's viewpoint. For complete flexibility, clips which animate camera parameters automatically override the automatic transition interpolation, giving the author complete control over the user experience during the transition including the pacing, framing and visual effects.

Slate transitions are a special case of authored transitions. Used to present 2D media, slate transitions are authored by placing an image plane in front of the camera as it transitions between camera surfaces. Various visual effects can be achieved by using multiple image planes simultaneously and by animating the transparency and other parameters of these image planes. While the slate transition is in progress, the camera is simultaneously being smoothly interpolated towards the destination camera surface. This essentially allows for a "soft" fade from a camera view, to a slate, and back, as Figure 7.10 illustrates.

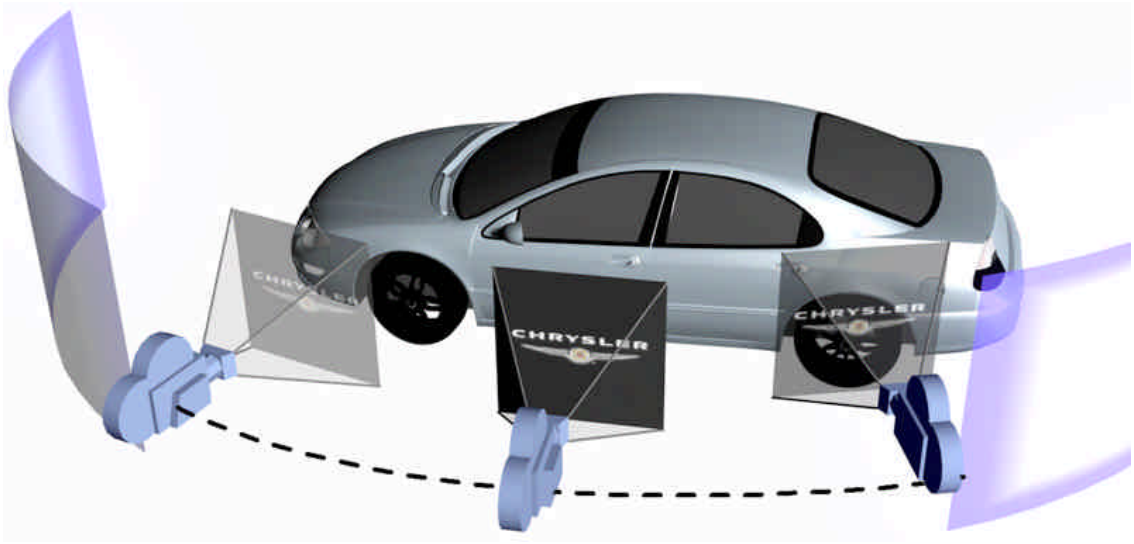

Figure 7.10. Slate transitions

7.2.4 Temporal Control of Animations

StyleCam supports temporal control or "scrubbing" of animations. During navigation mode, the user's mouse drags control the camera's position on the camera surface. However, when the user

moves off a camera surface into an animated transition, mouse drags control the (invisible) time slider of the animation. Time is advanced when the mouse is dragged in the same direction that the camera exited the camera surface and reversed if the directions are also reversed. When the mouse button is released, the system takes over time management and smoothly ramps the time steps towards the animation's original playback rate.

Our present implementation supports scrubbing only for automatic transitions. Authored and slate transitions are currently uninterruptible. There is however no technical reason why *all* transitions cannot support scrubbing. In future versions we intend to give the author the choice of determining whether or not any given transition is scrubable. This is important since in some cases it may be desirable to force the animation to playback uninterrupted and/or at a certain rate. For example, a sequence of informative textual information slates may be nonsensical if played in reverse.

Chapter 8. Evaluation

In this chapter, we describe an informal user study which we conducted to get a sense of users' initial reactions to using StyleCam. The study included seven participants, three of whom had experience with 3D graphics applications and camera control techniques, and four who had never used a 3D application or camera controls. The participants were asked to explore a 3D car model using StyleCam. In order to ensure the study resembled our intended casual usage scenario, we gave participants only minimal instructions. We explained the click-and-drag action required to manipulate the camera, and gave a brief rationale for the study. We asked the participants to imagine they were experiencing an interactive advertisement for that car. We did not identify the various components (camera surfaces, animated transitions, etc) nor give any details on them. This was deliberately done so that the participants could experience these components in action for themselves and give us feedback without knowing in advance of their existence. The study was conducted on a powerful workstation with a high-resolution CRT monitor for maximum graphics performance. Participants were seated at a clear desk, with the keyboard hidden, leaving the mouse as the only input device, as illustrated in Figure 8.1. The StyleCam experience presented to the participants was created by the present author, who is neither a 3D artist nor a professional in the film or advertisement production industry. Notes on user behaviour and

observations were made throughout the each test session. At the end of each session, participants' comments and reactions were recorded. Screenshots from some of the participants' experiences are shown in Figure 8.2.

Figure 8.1. A user interacting with a StyleCam experience.

Figure 8.2. Screenshots of a StyleCam experience.

One very promising result was that none of the participants realized that they were switching between controlling the camera and controlling the time slider on the animations. They felt that

they had the same type of control throughout, indicating that our blending between spatial and temporal control worked remarkably well. Also the simplicity of the interaction technique – essentially a single click and drag action – was immediately understood and usable by all our users.

Another reaction from all the participants was that, to varying degrees, they sometimes felt that they were not in control of the interaction when the uninterruptible animations occurred. This was particularly acute when the information in the animations seemed unrelated to their current view. In these cases, participants indicated that they had no idea what triggered these animations and were often annoyed at the sudden interruptions. However when the information was relevant the interruptions were not as annoying and often actually appreciated. In some cases participants indicated that they would have liked to be able to replay the animation or to have it last longer. This highlights the importance of carefully authoring the intermingling of uninterruptible animations with the rest of the interaction experience.

Participants also indicated that they would have liked the ability to click on individual parts of the car model in order to inspect them more closely. This request is not surprising since we made no effort in our current implement to support pointing. However, we believe that in future research StyleCam could be extended to include pointing.

As we expected, all the participants with prior 3D graphics camera experience stated that they at times would have liked full control of the camera, in addition to the constrained control we provided. Participants without this prior experience, however, did not ask for this directly although they indicated that there were some areas of the car model that they would have liked to see but could not get to. However, this does not necessarily imply full control of the camera is required. We believe that this issue can be largely alleviated at the authoring phase by ascertaining what users want to see for a particular model and ensuring that those features are

accessible via the authored camera surfaces. Interestingly, the participant with the most 3D graphics experience commented that the automatic transitions and smooth camera paths during those transitions were very good and that “for those who don’t know 3D and stuff, this would be very good”!

Central to our StyleCam system is the integration of spatial and temporal controls into a single user interaction model. The implications of this interaction model go far beyond a simple interaction technique. The blending of spatial and temporal control presents a completely new issue that an author needs to understand and consider when creating these interactive visual experiences. As evident from the comments of our users, temporal control can feel very much like spatial control even when scrubbing backwards in an animation when the animation consists of moving the viewing camera around the central object of interest. However, if the animation is not around the central object of interest, for example in some of our slate animations, temporal control can produce very different sensations. These include the feeling of moving backwards in time, interruption of a well-paced animation, jarring or ugly visuals, loss of control and sometimes even nonsensical content.

As a result, the author needs to be extremely cognizant of these artefacts and make design decisions as to when and where to relinquish control - and how much control - to the user. At one extreme, the author can specify that certain animations are completely uninterruptible by the user. In the experience we authored for our user study, we included several of these types of transitions. As discussed earlier, whether users favoured this depended heavily on the content. In other words, in some cases, as authors, we did not make the right decision. Further improvements could include partially interruptible animations. For example, we may not allow movement backwards in time but allow the user to control the forward pacing. This will largely solve the nonsensical content problem but may still result in occasionally jarring visuals.

If we intend to support these various types of control, we must also be able to set the users' expectations of what type of control they have at any given time. It is clear that the current StyleCam switching between spatial and temporal control without any explicit feedback to the user that a switch is happening works in most cases. In the cases where it fails, either the visual content itself should indicate what control is possible, or some explicit mechanism is required to inform the user of the current or upcoming control possibilities. In addition to the obvious solution of using on-screen visual indicators (e.g., changing cursors) to indicate state, future research could include exploring "hint-ahead" mechanisms that indicate upcoming content if the user chooses to stay on their current course of travel. For example, as the user reaches the edge of a camera surface, a "voice-over" could say something like "now we're heading towards the engine of the car". Alternatively, a visual "signpost" could fade-in near the cursor location to convey this information. These ideas coincide with research that states that navigation routes must be discoverable by the user (Furnas, 1997).

It is very clear from our experiences with StyleCam that the user's viewing experience is highly dependent on the talent and skill of the author. It is likely that skills from movie making, game authoring, advertising, and theme park design would all assist in authoring compelling experiences. However, we also realize that authoring skills from these other genres do not necessarily directly translate due to the unique interaction aspects of StyleCam.

Chapter 9. Future Directions & Conclusions

During the implementation and evaluation of StyleCam, we identified several avenues that we intend to explore further. This chapter discusses these future directions, and finally draws some important conclusions about the StyleCam research.

Although StyleCam has the appropriate components for creating compelling visual experiences, it is still currently a research prototype that requires substantial skills with MAYA. The workflow is awkward and not well suited to a real production environment. As outlined in chapter 6, we envision a collection of specialized tools working together with existing 2D and 3D tools to be useful and necessary for StyleCam to be adopted. The current implementation of StyleCam provides no custom GUIs for authoring StyleCam experiences, rather relying on naming schemes and grouping and other "hacks" using built-in MAYA tools. We strongly believe that the development of a custom authoring toolkit, will allow authoring teams to produce better StyleCam experiences, which strongly adhere to the authors' stylistic intentions. With such a foundation, it is reasonable to believe that new tools would be designed to work together with the StyleCam and traditional tools to further enhance the workflow or provide additional features to the StyleCam conceptual model.

While convenient for prototyping StyleCam, using MAYA as the viewer for authored experiences imposes many limitations to the visual quality of the experiences. Using a more optimized and advanced real-time rendering engine would allow experiences with more complex scene geometry and special effects (e.g., increased lighting and reflection realism, particle systems, environmental effects, etc..). We intend to explore the feasibility of implementing the StyleCam viewer as an application for Microsoft's X-Box gaming console. Unfortunately, for the time being, it is not possible to get photorealistic rendering.

Currently, StyleCam provides authoring concepts to produce purely visual interactive experiences. But the visual experience is only half the story. StyleCam is glaringly lacking in audio support. Integration of soundtracks and other audio into StyleCam will provide authors with additional ways of setting the pace and mood of the experience. A track could be associated with the experience, like the background music in a television commercial. Additional sound clips could be triggered and mixed-in by the user crossing over an author-defined hot zone on a camera surface.

A behaviour commonly exhibited by StyleCam users was to click on objects or parts of objects in the scene that they wanted to explore more carefully. We observed that this behaviour came very naturally to most users. As a result, we believe that StyleCam might benefit from the addition of a picking system, whereby the author could script "responses" to users picking objects. For example, the author may trigger an animation which reveals the engine when the user clicks on the car's hood. The addition of picking, however, adds a completely new dimension to StyleCam – that of object manipulation. It is not clear whether picking will add to the power of StyleCam without sacrificing its simplicity. Simplicity not only for the viewing user (due to a simplified interface) but also for the authoring team since picking is just another dimension requiring balance between author and user control.

The current StyleCam system does not provide a simple mechanism for authors to directly and precisely map mouse movements to changes in the user's view direction. We intend to explore the idea of adding a new type of money-shot/camera surface combination where the money-shot represents the camera position, and the camera surface (which would no longer necessarily intersect the money-shot position) defines a mapping between mouse movements and camera look-at points. Similarly, the author may wish to vary other camera parameters (e.g. lens length or 'zoom') based on user's mouse movements, perhaps in parallel with the change in camera position or look-at point. Such mappings could be implemented by associating a greyscale texture map for each varying parameter to a camera surface. The grey-value of the texture at a particular point on the camera surface would provide a value for the varying parameter.

Authored transitions are not as flexible as we had planned during the design of StyleCam. Transitions are triggered (via an authored script) when the user's camera reaches an edge of a camera surface. If the authored transition animates the user camera's position, the animation will likely appear very jarring unless the first animated camera position happens to coincide precisely with the point where the camera left the camera surface. This jumping of the camera can be avoided by always fading to black before animating the user camera's position. We nevertheless feel this to be too strict of a limitation. A possible solution to this problem is to treat the animated camera position as a target position and to smoothly move to this position over time.

Some other avenues we may explore include mechanisms for authoring animation paths using alternate techniques such as Chameleon (Fitzmaurice, 1993), and Boom Chameleon (Tsang et al., 2002)

Finally, it is important to note that StyleCam is not limited to product or automobile visualization. Other domains such as visualization of building interiors and medical applications could also utilize the ideas presented in this thesis. Figures 9.1, 9.2, and 9.3 illustrate some examples.

StyleCam experiences in different domains will have highly different production pipelines and associated roles. As a result, the tools used in the creation of an automobile advertising experience will have different requirements than those used in the creation of a molecular visualization experience. We envision the development of different tools for different domains, all implementing the StyleCam concepts in a way suitable for the domain in question. For example, for the visualization of medical or biochemical models, it is likely that only one or two people would be involved in the creation of the experiences. Furthermore, their skill sets will be very different from those of directors, animators and graphic designers. A tool suitable for the medical domain would therefore benefit from more automation, less flexibility and greater simplicity.

In this thesis, we investigated the feasibility of creating authored, stylized, interactive 3D experiences. We reviewed the current state-of-the-art in 3D viewing, determining that the user experience still falls short of that of traditional advertising media. In response, we formed a conceptual model which gives authors the flexibility needed to create experiences of varying visual styles yet which is simple enough to be predictable and usable. The conceptual model consists of camera surfaces and money-shots, animated transitions between these surfaces, and a novel interaction technique which seamlessly blends the spatial and temporal navigation into a single interaction. To test the concepts as they developed, prototypes of both the authoring environment and the viewing application were implemented in an iterative process. We authored an example experience with our final prototype and tested it in an informal user study to evaluate the effectiveness of the conceptual model. Our main findings indicated that although the experiences we authored were certainly not as compelling as a professionally produced automobile commercial, the conceptual model was an effective way for an author to transform their vision into an interactive experience. We concluded that with additional refinements, professional authoring, faster realtime rendering graphics hardware and high quality content, the

StyleCam system could indeed produce experiences rivalling today's best television commercials, while maintaining a high level of user interactivity.

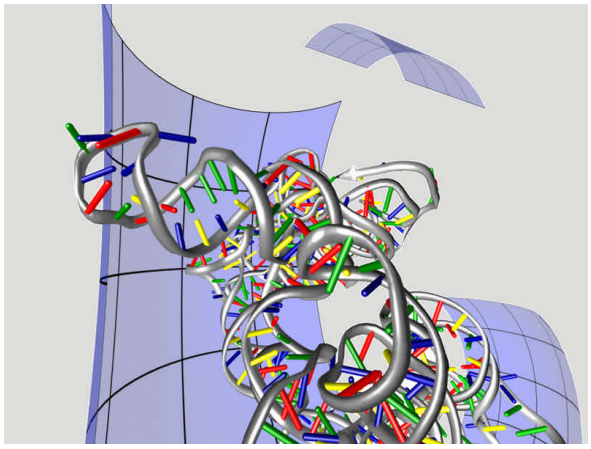

Figure 9.1. Example of StyleCam camera surfaces around a Ribozyme molecule.

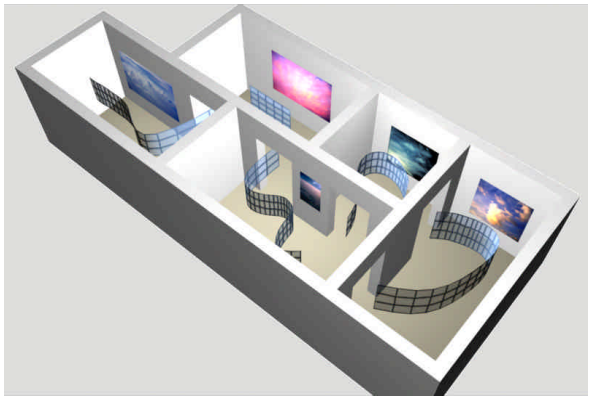

Figure 9.2. Example of StyleCam camera surfaces within the interior of a virtual art gallery.

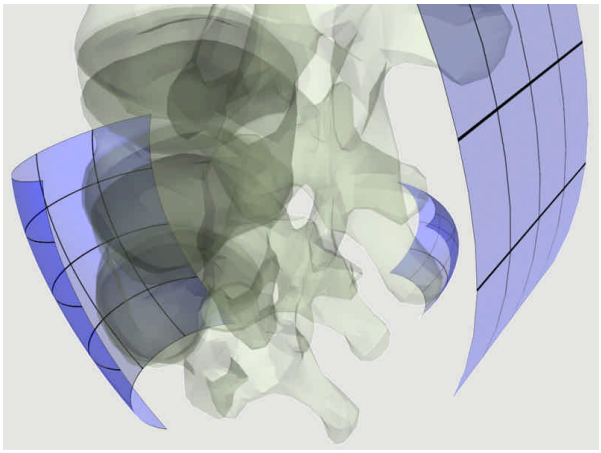

Figure 9.3. Example of StyleCam camera surfaces around a visualization of human vertebrae fragment.

References

Balakrishnan, R., & Kurtenbach, G. (1999). Exploring bimanual camera control and object manipulation in 3D graphics interfaces. *ACM CHI*. p. 56-63.

Bares, W., McDermott, S., Boudreaux, C., & Thainimit, S. (2000). Virtual 3D camera composition from frame constraints. *ACM Multimedia*. p. 177-186.

Blaxxun Contact. Blaxxun Interactive AG. Munich, Germany. (http://www.blaxxun.com).

Bowman, D., Johnson, D., & Hodges, L. (1997). Travel in immersive virtual environments. *IEEE VRAIS*. p. 45-52.

Bowman, D., Johnson, D., & Hodges, L. (1999). Testbed environment of virtual environment interaction. *ACM VRST*. p. 26-33.

Buxton, W. (1990). Three-state model of graphical input. Human-computer interaction - INTERACT'90, ed. D. Diaper. Elsevier Science Publishers B. V. (North-Holland): Amsterdam. p. 449-456.

Carpendale, M.S.T., & Montagnese, C. (2001). A framework for unifying presentation space. *ACM UIST*. p. 61-70.

Chapman, D., & Ware, C. (1992). Manipulating the future: predictor based feedback for velocity control in virtual environment navigation. *ACM Symp. on Interactive 3D Graphics*. p. 63-66.

Chen, M., Mountford, S. J. & Sellen, A. (1988). A study in interactive 3-D rotation using 2-D control devices. *Computer Graphics, 22(4)*. p. 91-97.

Chen, S.E. (1995). QuickTime VR: An image-based approach to virtual environment navigation. *ACM SIGGRAPH*. p. 29-38.

Cycore Cult3D. Cycore AB. Uppsala, Sweden. (http://www.cult3d.com).

Darken, R., & Sibert, J. (1996). Wayfinding strategies and behaviours in large virtual worlds. *ACM CHI*. p. 142-149.

Drucker, S, Galyean, T., & Zeltzer, D. (1992). CINEMA: A system for procedural camera movements. *ACM Symp. on Interactive 3D Graphics*. p. 67-70.

Drucker, S., & Zeltzer, D. (1994). Intelligent camera control in a virtual environment. *Graphics Interface*. p. 190-199.

Elvins, T., Nadeau, D., Schul, R., & Kirsh, D. (1998). Worldlets: 3D thumbnails for 3D browsing. *ACM CHI*. p. 163-170.

Fitzmaurice, G. (1993). Situated information spaces and spatially aware palmtop computers. *Comm. of the ACM, 36*(7). p. 38-49.

Fukatsu, S., Kitamura, Y., Masaki, T., & Kishino, F. (1998). Intuitive control of bird's eye overview images for navigation in an enormous virtual environment. *ACM VRST*. p. 67-76.

Furnas, G. (1986). Generalized fisheye views. *ACM CHI*. p. 16-23.

Furnas, G. (1997). Effective view navigation. *ACM CHI*. p. 367-374.

Galyean, T. (1995). Guided navigation of virtual environments. *ACM Symp. on Interactive 3D Graphics*. p. 103-104.

Gliecher, M., & Witkin, A. (1992). Through-the-lens camera control. *ACM SIGGRAPH*. p. 331-340.

Hanson, A., & Wernet, E. (1997). Constrained 3D navigation with 2D controllers. *IEEE Visulization*. p. 175-182.

He, L., Cohen, M., & Salesin, D. (1996). The virtual cinematographer: a paradigm for automatic real-time camera control and directing. *ACM SIGGRAPH*. p. 217-224.

Igarashi, T., Kadobayashi, R., Mase, K., & Tanaka, H. (1998). Path drawing for 3D walkthrough. *ACM UIST*. p. 173-174.

Jul, S., & Furnas, G. (1998). Critical zones in desert fog: aids to multiscale navigation. *ACM UIST*. p. 97-106.

Kang, S. B. (1997). A survey of image-based rendering techniques. *Cambridge Research Lab Technical Report Series*.

Lippman, A. (1980). Movie-maps: an application of the optical videodisc to computer graphics. *ACM SIGGRAPH*. p. 32-42.

Mackinlay, J., Card, S., & Robertson, G. (1990). Rapid controlled movement through a virtual 3D workspace. *ACM SIGGRAPH*. p. 171-176.

Marrin, C., Myers, R., Kent, J., & Broadwell, P. (2001). Steerable media: interactive television via video synthesis. *ACM Conference on 3D Technologies for the World Wide Web*. p. 7-14.

Newman, W. (1968). A system for interactive graphical programming. *AFIPS Spring Joint Computer Conference*. p. 47-54.

Phillips, C., Badler, N., & Granieri, J. (1992). Automatic viewing control for 3D direct manipulation. *ACM Symp. on Interactive 3D Graphics*. p. 71-74.

Shoemake, K. (1985). Animating rotation with quaternion curves. *ACM SIGGRAPH*. p. 245-254.

Smith, G., Salzman, T., & Stuerzlinger, W. (2001). 3D Scene manipulation with 2D devices and constraints. *Graphics Interface*. p. 135-142.

Steed, A. (1997). Efficient navigation around complex virtual environments. *ACM VRST*. p. 173-180.

Stoakley, R., Conway, M., & Pausch, R. (1995). Virtual reality on a WIM: Interactive worlds in miniature. *ACM CHI*. p. 265-272.

Tan, D., Robertson, G., & Czerwinski, M. (2001). Exploring 3D navigation: combining speed-coupled flying with orbiting. *ACM CHI*. p. 418-425.

Tsang, M., Fitzmaurice, G., Kurtenbach, G., Khan, A. & Buxton, B. (2002). Boom Chameleon: Simultaneous capture of 3D viewpoint, voice and gesture annotations on a spatially-aware display. *ACM UIST*. p. 111-120.

Viewpoint Products. Viewpoint Corporation. New York, NY. (http://www.viewpoint.com).

Vinson, N. (1999). Design guidelines for landmarks to support navigation in virtual environments. *ACM CHI*. p. 278-285.

Virtools Products. Product Information, Virtools SA. Paris, France. (http://www.virtools.com).

VRML Specification. Web 3D Consortium. Orinda, CA. (http://www.web3d.org)

Ware, C., & Fleet, D. (1997). Context sensitive flying interface. *ACM Symp. on Interactive 3D Graphics.* p. 127-130.

Ware, C., & Osborne, S. (1990). Exploration and virtual camera control in virtual three dimensional environments. *ACM Symp. on Interactive 3D Graphics*. p. 175-183.

Wernert, E., & Hanson, A. (1999). A framework for assisted exploration with collaboration. *IEEE Visualization*. p. 241-248.

Zeleznik, R., & Forsberg, A. (1999). UniCam - 2D Gestural Camera Controls for 3D Environments. *ACM Symp. on Interactive 3D Graphics*. p. 169-173.

Zeleznik, R., Forsberg, A., & Strauss, P. (1997). Two pointer input for 3D interaction. *ACM Symp. on Interactive 3D Graphics.* p. 115-120.