TASK-BASED AUTOMATIC CAMERA PLACEMENT

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BİLKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Mustafa Kabak

August, 2010

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Tolga K. Çapın (Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Uğur Güdükbay

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Sinan Gezici

Approved for the Institute of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Institute

ABSTRACT

TASK-BASED AUTOMATIC CAMERA PLACEMENT

Mustafa Kabak
M.S. in Computer Engineering
Supervisor: Asst. Prof. Dr. Tolga K. Çapın
August, 2010

Placing cameras to view an animation that takes place in a virtual 3D environment is a difficult task. Correctly placing an object in space and orienting it, and furthermore, animating it to follow the action in the scene is an activity that requires considerable expertise.

Approaches to automating this activity to various degrees have been proposed in the literature. Some of these approaches have constricted assumptions about the nature of the animation and the scene they visualize, therefore they can be used only under limited conditions. While some approaches require a lot of attention from the user, others fail to give the user sufficient means to affect the camera placement.

We propose a novel abstraction called *Task* for implementing camera placement functionality. Tasks strike a balance between ease of use and ability to control the output by enabling users to easily guide camera placement without dealing with low-level geometric constructs. Users can utilize tasks to control camera placement in terms of high-level, understandable notions like objects, their relations, and impressions on viewers while designing video presentations of 3D animations.

Our framework of camera placement automation reconciles the demands brought by different tasks, and provides tasks with common low-level geometric foundations. The flexibility and extensibility of the framework facilitates its use with diverse 3D scenes and visual variety in its output.

Keywords: Camera planning, autonomous cinematography, task-level interaction.

ÖZET

GÖREV YÖNELİMLİ OTOMATİK KAMERA YERLEŞİMİ

Mustafa Kabak
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Y. Doç. Dr. Tolga K. Çapın
Ağustos, 2010

Üç boyutlu bir sanal ortamda gerçekleşen animasyonları görüntülemek üzere kamera yerleştirmek zor bir iştir. Uzayda bir nesnenin konumunu ve yönelimini doğru bir şekilde ayarlamak, ve dahası bu nesneyi sahnedeki olayları takip edecek şekilde hareket ettirmek uzmanlık gerektirir.

Bu eylemi farklı derecelerde otomatik hale getirmeye yönelik öneriler getirilmiştir. Bazı yaklaşımlar kullanıcının sürekli kamera yerleşimiyle ilgilenmesini gerektirirken, bazıları da kullanıcının kamera yerleşimini yönlendirmesine hiç imkan tanımamaktadır.

Bu çalışmada otomatik kamera yerleştirme işlevini gerçekleştirmek için *Görev* (Task) adını verdiğimiz yeni bir soyutlama öneriyoruz. Görevler kullanıcının geometrik detaylarla ilgilenmeden kolayca kamera yerleşimini yönlendirmelerini sağlayarak kullanım kolaylığı ve çıktı üzerinde kontrol imkânı arasında bir denge yakalamaktadırlar. Kullanıcılar üç boyutlu animasyonların görselleştirildiği videolar hazırlarken görevleri kullanarak nesneler, nesneler arasındaki ilişkiler ve izleyici üzerinde bırakılan izlenimler gibi anlaşılır kavramlar üzerinden kamera yerleşimini yönetebilirler.

Burada önerilen otomatik kamera yerleşimi altyapısı farklı görevlerden gelen talepleri bağdaştırır ve görevlerin ortak kullanabileceği geometrik hesaplama işlevleri sunar. Altyapının esnekliği ve genişletilebilirliği sistemin çok çeşitli üç boyutlu sahnelerde kullanılmasını ve çıktı olarak elde edilen videoların da görsel olarak çeşitlilik sunmasını mümkün kılar.

Anahtar sözcükler: Kamera planlama, otomatik sinematografi, görev düzeyinde etkileşim.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Due to the limitations of current technology, computer-generated three dimensional visualizations need to be viewed through two dimensional displays. Our natural methods to explore a real three dimensional object or environment do not work while viewing a projected image on a screen. These methods include walking around in an environment, moving our heads and holding an object to try to get a better viewpoint.

The difference between the available means for exploring a real environment and a computer generated one has consequences for designers of computer applications that use virtual environments. This difference brings both the unavailability of the natural means to explore virtual environments, and an entirely new realm of possibilities. 3D applications have realized a lot of these possibilities by not limiting the usage of a display as a stationary window to a tabletop or to a distant scene: Almost all applications have the concept of a *virtual camera* which can be freely positioned in the virtual environment. The view from this virtual camera is presented to the user of the application.

A virtual camera shares almost none of the movement limitations a human being or a real camera has. It does not have any mass or volume and does not need to stand on or hang from a supporting structure. It can move at speeds impractical for physical objects or jump from a position to another in an instant.

Users of 3D applications have almost unlimited possibilities to view and explore their virtual environments.

This abundance of possibilities introduces a problem of its own: How does the user choose from the unlimited possible positions, orientations and movements of a virtual camera? In the real world, explorers of an environment are constrained by the mobility capacity of their bodies. They are also equipped with the *intuition* they have been developing since they were born. Users of virtual environments lack both the constraints and the intuition, therefore they have both more options and less tools to make a choice.

This *camera placement problem*, besides being a challenge to non-expert users of virtual environments, is seldom considered important by them. Most of the time, the foremost concern of a user is to understand an event or object, or to be immersed in a visceral experience. Such users will expect the 3D application to place the virtual camera to aid them in whatever task they set out to perform and do so in a transparent manner. This expectation of users is our motivation to design and implement a system that places and moves a virtual camera in a way that helps users accomplish their tasks.

Our camera placement system takes the description of a scene and the records of movements of objects in the scene as input and computes a series of camera positions and paths that span the duration of the given animation. The resulting camera behavior is expected to provide a generally understandable and aesthetically pleasing presentation of the events to users. Additionally, the system takes into account a list of *tasks* that users explicitly state and constructs the camera placement plan so that the presentation helps users achieve those tasks.

Tasks enable users to affect the camera placement without knowing about or spending time on the low-level geometric camera parameters. Users simply communicate to the system their objectives in viewing the presentation. This task-based approach gives non-expert users the means to prepare a video out of the recordings of an interaction, results of a simulation, or a piece of scripted 3D animation.

In addition to being easy to use, our camera placement system is easy to extend and configure. New task types can be incorporated to the system and use the underlying fundamentals of the system shared by other task types. The system is structured in a strictly layered manner and the functionality of each layer is accessed through well-defined interfaces, making extension possible at every level.

Summary of Contributions

- An in-depth survey of existing automatic camera control techniques,
- An automatic camera placement system, which
  - can be used with minimal attention and effort,
  - can be used with minimal knowledge about geometry, cinematography and video editing,
  - allows users to express their expectations from camera placement in high-level, understandable terms; and honors those expectations, and
  - can incorporate a certain amount of randomness, therefore producing videos with variety in appearance.
- An effective way to break down camera placement functionality into layers, which facilitates extensibility and versatility.

Organization of the Thesis

- Chapter 2 presents the previous work on automatically placing virtual cameras in 3D scenes.
- Chapter 3 describes our camera placement system.
- Chapter 4 demonstrates a sample output of our system, and compares our approach of implementing and providing automatic camera placement functionality to previous approaches.

- Chapter 5 concludes the thesis.

Chapter 2

Background

Controlling virtual cameras on behalf of users and finding good viewpoints to look at objects and scenes have interested researchers of computer graphics and other disciplines. Automatic camera placement techniques have been utilized in several application areas. The particular application area dictates the scope of functionality of automatic camera placement and provides the researchers with basic assumptions.

One such application area is viewing a static object or a few objects to gain information about their shapes. A technique that is designed to be used in this type of application generally lacks the facilities to follow animated objects. Also such a technique may constrain its solution space of camera positions to a spherical surface around the center of the viewed objects. Since an individual object does not necessarily imply a ground level or up direction, such a technique may freely rotate the camera (or the object) to obtain favorable views.

Another application area that can benefit from automatic camera placement is generating video presentations from recorded or authored animations. A technique that tries to generate videos from animations needs to take into account the concepts of time and movement. Such a technique will most likely need to move the virtual camera in some way during the animation. This kind of technique must take measures to preserve the viewer's orientation about the scene

while moving the camera. It may also be expected from the camera placement technique to produce entertaining or pleasant as well as informative views. Since the input animation is available in its entirety before the placement of cameras in this type of application, the camera placement technique may perform a variety of analyses on the data without a strict time limitation.

Automatic camera placement can also be used in interactive applications. Requirements of interactive applications from an automatic camera placement technique may include the requirements of a video generation application mentioned above. On top of these requirements, an interactive application needs the camera placement to be timely. Automatic camera placement component of an interactive application cannot pause the interaction to carry out elaborate calculations; cannot peek into the future and probably cannot request supporting information from the user. A technique that places cameras in interactive applications may need to constantly anticipate the next movements of the participants.

Camera placement techniques differ in their degree of autonomy. While some techniques completely take over the control of the camera, others only aid the users to control the camera themselves. Another class of techniques fall between these two extremes: be it interactive or off-line, these camera placement techniques accept a limited amount of input from the user. Some of the input is crucial for the resulting presentation. The input mostly asked from users by camera placement techniques is the current *object of interest* in a complex scene. In interactive applications, camera placement techniques may expect the user to indicate the object they want to view through a graphical user interface. In off-line applications, a list of time intervals and the object of interest during each interval may be needed as input. In fact very few techniques claim to find which object(s) to view without user input.

Besides the user input that is crucial, some camera placement techniques accept optional input from users that affects the end result in subtle ways. Input of this kind may include the emotions that are desired to be evoked in the viewers, the pacing of the video or the degree of "artistic" freedom the technique is allowed.

There are many ways including the above to outline the literature on automatic camera placement. We saw fit to classify the existing camera placement techniques by two criteria. In Section 2.1, camera placement techniques are classified by the way they model and treat the objects that will be viewed. In Section 2.2, techniques are classified by the dynamic behavior abilities of the virtual cameras that they control.

The place of a technique in these two classifications generally determines the functionality available to the designer of an application or presentation who has decided to use that technique. These classifications also hint at the primarily intended application areas of each technique.

2.1 Scene Structure

A scene, in the most general definition, is a collection of three dimensional objects that occupy a common three dimensional space. For the sake of classifying camera placement techniques, in this section, we add to the above definition the ability to place a virtual camera among the objects, "inside" the scene. Therefore the techniques that place the camera strictly outside a bounding convex volume of the set of objects they view are not considered capable of viewing a scene. These techniques are classified under the heading "Object Based" (Section 2.1.1).

2.1.1 Object Based

Vázquez et al. [13] define a good view as one that conveys a high amount of information to the viewer. Since their approach involves quantifying the amount of information, they make use of information theory. They define *viewpoint entropy* as a measure of information a viewpoint provides to the viewer about a set of objects. This measure basically favors the viewpoints that see great numbers of faces of the polygonal object models, see the faces at shorter distances and better angles. Their technique to find viewpoints with high entropy is to sample

a number of viewpoints on a sphere around the objects and to pick the one with the greatest viewpoint entropy. They also have developed methods for finding a set of viewpoints that cover the entire scene, and for moving around a scene while viewing all the faces of objects.

Sokolov et al. [11] approach the same problem of viewing a stationary set of objects from outside their bounding volume, but they claim to have developed a higher-level method than the one proposed by Vázquez et al. [13]. Their high-level method benefits from considering the scene not as a flat collection of polygonal faces but as a structure of individual objects. While calculating the quality of a viewpoint, they take into account the *importance* and *predictability* of the objects. If the only information about a scene is its geometry, they assign importance and predictability values by applying a default function. Designers can supply higher importance values for the objects they particularly want the viewers to see; and higher predictability values for the objects which are likely to be recognized even when a small part of them is visible. Similarly to Vázquez et al. [13], they have a method to find good paths around the objects for the camera to follow.

Kwon and Lee [10] aim to find viewpoints that communicate the movements of articulated character models. They quantify a viewpoint's ability to capture the dynamism of a character as *motion area*, the area that is swept by the bones of the model projected onto the view plane. To obtain fixed camera positions with high motion area values, they use the eigenvalue decomposition optimization technique. They also extend this technique to find favorable camera paths for off-line and interactive character animation sequences.

2.1.2 Scene Based

In this section existing techniques that can place the camera inside a scene are explained. The techniques are classified by the dynamic characteristics of the scenes they can view. Techniques mentioned in Section 2.1.2.1 can only view scenes in which objects do not move. In Section 2.1.2.2 camera placement techniques can take an animation record as input and place and move the camera to

appropriately view the moving objects. In Section 2.1.2.3 techniques can react to movements in the scene as they occur.

2.1.2.1 Stationary Scene

Drucker and Zeltzer [5] acknowledge the difficulty of directly adjusting low-level camera parameters and propose a system that provides the users simpler interfaces to control the camera. They claim that different user interfaces are optimal for different kinds of tasks (not to be confused with our concept of tasks) and present the *camera module* abstraction that encapsulates both the user interface and the underlying mechanism that derives low-level camera parameters from user inputs. The underlying mechanism is a different set of *constraints* for each camera module. Whenever a particular camera module is active, a generic constraint solver solves the constraints of that module for the low-level camera parameters and applies them to the virtual camera accordingly. By virtue of being easy to use, the user interface of each module limits the user's choices for placing the camera. Therefore, we can say that each module contains a separate definition (that is appropriate for the kind of task the module is designed for) of a good view.

Halper and Olivier [7] do not define and impose a desirable view. Instead their system CamPlan presents a great variety of *shot properties* for the users to select and tries to find the camera position that will satisfy those properties for the objects. These properties can be in object space or image space; they can be in absolute terms or relative to other objects; and they can be about position, size, orientation or visibility. By permuting these shot property classes users are equipped with a powerful means to affect camera placement which is not tied to a specific application.

2.1.2.2 Scene with Scripted Animation

Christianson et al. [4] look to the established practices in the field of cinematography to define desirable camera positions and angles to view an animated scene. In order to formulate a cinematographer's experience so that a computer can use it, they designed *Declarative Camera Control Language* (DCCL) and codified 16 *idioms* in this language. Each idiom matches a scene configuration to a series of *shots*. An example idiom gets activated whenever three actors are talking to each other, and alternatingly directs the camera to individual actors and groups of 2 or all 3 actors in a way that is described in the cinematography literature. The variety of both scene configurations and resulting camera configurations is very limited, rendering their technique unsuitable for most 3D applications.

Kennedy and Mercer [9] similarly encode cinematography knowledge into a knowledge base and utilize it when placing the camera. Instead of directly matching actions to camera placements, their system takes into account the *themes* and *moods* the designer wishes to evoke. Therefore different themes and moods, given to the system together with the same animation description, yield different camera placements. In addition to camera placements, their system also determines the lighting and colors in the scene.

2.1.2.3 Interactive Scene

Bares and Lester [2] adopt a probabilistic approach to camera placement. Their system picks camera positions and directions relative to the object of interest (which is selected by the user) randomly from a predetermined set of object-camera distances (close-up, wide, etc.) and horizontal and vertical angles (front, front-left, right etc. and low, medium, high). The system does not take into account the desirability or the information content of candidate viewpoints. The novelty in their approach is that their system honors the viewer's preferences while placing the camera. They present the user understandable alternatives such as informational vs. dramatic style, fast vs. slow pacing and gradual vs. abrupt camera transitions. These preferences are then used to obtain a user

model which contains various probability values that are used in the camera placement algorithm.

Bares et al. [1] present a technique that has access to semantic as well as geometric information about a scene. This technique automatically places cameras to follow an *interactive fiction*. The participants in this kind of fiction can be autonomous agents. Their actions are constrained in a way that will let the system know their meaning and importance. *Cinematographic goals* are obtained from these actions, giving the system the ability to know where the important events are taking place at any moment and show them to the viewer.

These goals are then used to obtain *constraints* to place the camera. These constraints are purely geometrical and include subject inclusion constraints, shot distance constraints and vantage angle constraints. These constraints are solved to finally obtain low-level camera parameters. One novel approach here is *partial constraint satisfaction*. If the constraint solver cannot find a camera position and direction that satisfies all the constraints, it partitions the constraints and satisfies them in more than one view. These multiple views are then displayed either in succession, or at the same time in different frames on the same screen.

In another paper [3], Bares and Lester introduce the ability to increase the clarity of the presentation by showing informative visual cues over the displayed scene. These cues can point to an object which is otherwise not easily noticeable. They drop the interactive fiction aspect of their previous work and let the users specify the goals directly, in real time from the graphical user interface. Also, users can specify different desired viewing angles and distances for different objects in the scene. Users can also select the style and pacing of the presentation, as in their previous work [2].

Drucker and Zeltzer [6] improve on their previous work [5] to handle interactive scenes as well as stationary ones. They still modularize the automatic camera placement functionality into camera modules. However, the transition of control of the camera among camera modules is facilitated by a *filming agent* that can reason about the scene, rather than the users themselves. The filming agent depends on formalized cinematography practices to produce appropriate camera

constraints or to select camera modules, similar to the *idioms* mentioned by Christianson et al. [4] and He et al. [8].

Drucker and Zeltzer also introduce a visual programming environment to build camera modules and module switching behaviors.

He et al. [8] improve the technique proposed by Christianson et al. [4] to use cinematography idioms in interactive environments. To achieve this, they replace the off-line planning components in their previous work with idiom-switching logic built into the idioms themselves. In this system, an idiom can *call* other idioms and return to its caller. Idioms are responsible for detecting when they are no longer appropriate for a scene configuration and yield the control of the camera to another idiom. Most of the time such an idiom gives control to a more general idiom in a hierarchy. This general idiom may then return to an even more general idiom or find a more specific idiom that fits the scene configuration.

Also, idioms are modeled as finite state machines instead of linear shot descriptions as in the previous technique [4]. This way they become the dynamic constructs that are needed to react to unpredictable events in an interactive application. Idioms are allowed to slightly affect the positions of the objects in the scene to obtain better views.

Tomlinson et al. [12] implement an automatic camera control system as a behavioral agent with *emotions* and *motivations*. To adequately function, this system needs to access the emotions and motivations of the actors in the environment. The emotions and motivations of the actors affect the emotions and motivations of the camera. Emotions change more slowly and work more subtly than motivations. Motivations directly determine the target actor(s) and camera angles, distances and movement. They are designed to bring out behavior as would be expected from a cinematographer. For example, the camera has a great motivation to keep its object of interest the same just after it has changed it. This motivation diminishes in time, therefore very short and very long shots are avoided. Their system affects the lighting of the scene as well as camera placement.

2.2 Camera Behavior

Besides the movement capabilities of the viewed objects, movement capabilities of the virtual cameras differentiate automatic camera control approaches. In this section the camera placement techniques mentioned in Section 2.1 are classified by their capability to move the virtual camera. In the following classification, a "moving camera" (Section 2.2.2) is defined as one that can move in a *continuous* manner, like a real camera being pushed on a track or being rotated by a camera operator. A "cutting camera" (Section 2.2.3) is defined as a camera that can move instantly to discontinuous positions and orientations. Conceptually, a camera control technique that can "cut" can be compared to a *vision mixer* who selects a view from many real cameras in real time, or an editor who splices together video clips after all of them have been recorded. In fact, *cutting* is the film editing term for sequencing film fragments taken from different cameras or taken from the same camera at different times, which is exactly the behavior in question.

These two functionalities do not imply each other, and each of the two involves different considerations.

2.2.1 Stationary Camera

Halper and Olivier [7] only consider finding one camera position and orientation that satisfies users' clearly defined requirements. They provide a powerful means to define the constraints a stationary view should satisfy, but their approach does not extend to finding desirable camera paths or movements.

Vázquez et al. [13] and Sokolov et al. [11] are mainly interested in single non-moving views that give the greatest amount of information about a group of objects. Even though they can move the camera around the objects of interest, their movement functionality is generally built upon finding favorable stationary views and connecting them.

2.2.2 Moving Camera

As mentioned above, Vázquez et al. [13] and Sokolov et al. [11] facilitate camera movement as extensions to their stationary camera placement functionality. Specifically, Vázquez et al. [13] propose an incremental, greedy approach that starts from a random position and continuously selects the best from a number of nearby candidate positions. On the other hand, Sokolov et al. [11] first find a number of favorable viewpoints around the objects, and then connect these viewpoints to obtain a path for the camera to follow.

Kwon and Lee [10] obtain camera paths by calculating a favorable viewpoint for every frame in an animation, and *blending* these viewpoints together. Measures are taken to produce a smooth camera path without any jarring sudden movements.

Bares and Lester [2], [3] and Bares et al. [1] consider camera movement only in a spherical coordinate system around the object of interest at any time. Since the object is always the center of this coordinate system, whenever it moves the camera also moves to track it. Other than this need to track objects, their motivation to move the camera is keeping the user interested by adding variation to the presentation. The camera either moves around the object of interest or moves toward or away from it, all the while continuously being directed to it. Camera movements are triggered by passage of time and they occur in a random fashion. In their latter two papers [1] [3], users can specify the directions they prefer to look at for individual objects.

Christianson et al. [4] and He et al. [8] move the camera only in certain ways that are commonplace in cinematography for filming real world scenes. These movement types (*panning*, *tracking*, etc.) are presented as the primitives of the system, and they can only be used as they are by the *idioms* (the constructs that codify the cinematography principles). Kennedy and Mercer [9] also present a predetermined set of camera movement types.

Tomlinson et al. [12] use a spring-damper system for moving the camera.

The configuration of the spring-damper system reflects the emotions of both the camera and the subject. For example, a sad camera moves slowly while an angry camera moves more abruptly.

Drucker and Zeltzer [5], [6] do not impose movement capability limitations on cameras. Each of their *camera modules* may have different capabilities for moving the camera. One module they describe in detail [5] uses path planning techniques to take the viewer on a tour in a virtual museum. Since their system is designed for interactive use, some of their camera modules directly map user input to movement or position of the camera, carrying out only minimal checks to prevent gross errors like going through objects.

2.2.3 Cutting Camera

There are two major motivations to change the position of the virtual camera instantaneously and discontinuously. One of these motivations is following the interesting actions whenever they occur. Existing techniques depend on users in varying degrees to know their object of interest. A class of interactive camera placement techniques directly depend on users: they provide a graphical user interface for the users to indicate which object they want to view at any time, and always show that object. Bares and Lester [2], [3] and Drucker and Zeltzer [5] describe such techniques. The intelligence in this kind of techniques manifests itself in other aspects than finding which object is the object of interest.

Another motivation to cut is adding variation to the presentation and making it more interesting. Even if the object of interest is the same for a long time, a camera placement technique may choose to look at that same object from a different angle or different distance. Techniques presented by Bares and Lester [2], [3], Bares et al. [1] and Tomlinson et al. [12] exhibit this behavior. These cuts have timing constraints: they need to be apart for the presentation to be clear and understandable, however if they are too apart the presentation may become uninteresting.

The techniques cited above use the cutting behavior to communicate stylistic and emotional aspects to the viewer. Bares and Lester [2], [3] allow the user to indicate the desired pacing of the presentation. If the user wants a fast pacing, the cuts become more frequent. Tomlinson et al. [12] connect the cutting behavior to the emotions of the camera and the actors in the scene.

Christianson et al. [4] take a complete description of an animation and process it to plan the placement of the virtual camera. The cuts take place both when the object of interest changes, and when the idioms dictate. The cutting behavior of the idioms is taken from cinematography practices. Improving this work, He et al. [8] turn idioms to state machines. Most of the state transitions in these state machines involve cutting. These transitions are also modeled according to the cinematography practice. Cuts also take place when an idiom decides that it no longer applies to the scene and yields control.

Kennedy and Mercer [9] also need a complete animation description before planning camera placement. Similar to Bares and Lester [2], [3] and Tomlinson et al. [12] they use cutting behavior to communicate mood.

Bares et al. [1] and Bares and Lester [3] use cuts to satisfy view constraints in more than one view. Specifically, if their technique cannot find a viewpoint that satisfies all the constraints, it partitions the constraints and tries to find a set of viewpoints that collectively do so. Views from these multiple viewpoints are either shown in succession (hence the cutting behavior) or at the same time in different frames on screen.

Chapter 3

Approach

The camera control technique proposed here analyses an animation; takes into account high-level, qualitative directives and hints encoded in *tasks* which are given by users; and produces a video that depicts the input animation. The preparation of the output video focuses on the fulfillment of users' wishes which they communicated to the system in the form of tasks.

The mechanism of this camera control system, its components, underlying principles and limitations are explained in this chapter.

3.1 Context

Our camera control technique has several assumptions about its input data and working environment. The origin of most of these assumptions is the particular application area of the technique which inspired it in the first place: presenting off-line simulations of military engagements. Broadly, these assumptions are that placement of cameras is a secondary concern to the animation itself (which is realistically computed in the case of military simulations, instead of being authored by an artist); and that the users of the system are not expected to be experts in directly controlling cameras and video editing.

3.1.1 Usage

Our camera control system functions in an off-line fashion. In order to place the cameras, our system reads in the complete record of an animation. The input animation may possibly be obtained in a number of ways: It may be the result of a realistic simulation; it may be designed by an artist; or it may be a recorded interaction (as in the *replay* function of a game).

Once the input animation is ready, the user picks the communication tasks that he or she wants to accomplish. These tasks and the animation itself constitute the input to our system.

After receiving the input data, our system computes a camera placement plan that describes the position, orientation and other values the camera should take for every moment of the animation. A rendering component follows both the animation and the camera placement plan, producing the output video.

The user can then show the video to its intended viewers. The viewers' understanding of the animation will be guided by the tasks selected by the user at the beginning of the camera placement procedure.

3.1.2 Users

Our technique is meant to be useful for a particular kind of user. Users outside the target group may find our system either too complicated, or inadequate for their needs.

Camera control skill: Users of our system are not expected to be experienced in low-level camera control. Users may not have any ideas about where to put the cameras exactly, or even if they do they may be unable to articulate the low-level geometric camera parameters to achieve their desired camera placements. Handling those details is precisely the functionality of our system.

Knowledge of the input animation: It is expected from users to be informed about the content of the animation that is given as input to our system. In order to communicate the meaning of the animation to the viewers, our system takes a variety of inputs from the user. These inputs, in a way, augment the *geometric* data that is available from the animation with *semantic* data that can only be provided by a human being who understands the meaning of the animation.

Purpose of preparing a video: Even if the user of our system is expected to know the content of the animation, viewers of the output video are not. Our system can be used to prepare informative videos that help communicate to the viewers the meaning of the animation as it is in the mind of the user. Preparing videos for improving one's own comprehension about the input animation is not precluded, but it is not emphasized either.

3.1.3 Input Scene and Animation

There are several requirements of our technique related to input data.

Completeness: Our camera control technique requires that the scene and animation data are complete before preparing the camera placement plan. Our system cannot react to events as they occur.

Continuity: The animation is expected to be continuous, taking place in one space over an undivided time period.

Nature of objects: Our technique assumes that the objects in the scene have generally unchanging, solid morphologies. Objects with articulated limbs and liquid and gaseous forms are not handled.

Nature of motion: The objects are assumed to be able to move freely in three dimensions over long distances. Our technique is not constrained to a limited planar surface.

3.1.4 Output Video

The imperative of the output video is fulfilling the communication needs as expressed by users in the form of tasks. Therefore the expectations from the output video depend on the tasks selected by the user. These expectations will be explained for every type of task. However, there are also concerns that transcend or encompass the individual tasks.

Comprehensibility: The output video is expected to present the animation in an understandable manner. Watching the video, viewers should be able to keep a correct sense of orientation of the scene at all times.

Interestingness: The output video should ideally grab the attention of the viewers. An interesting presentation is an invaluable asset for the users of our system to communicate the animation and their interpretation of it.

The rendering of the animation is not the concern of our technique. The camera placement plan our system produces can be given to an appropriate rendering system along with the original input data to obtain the video.

3.2 Concepts

Our system incorporates various components to carry out the automatic camera control functionality. Every component of the system belongs to a category which is formally defined in this section. These categories form a hierarchy in which components that belong to higher level categories use the lower level ones and embody more of the camera control intelligence.

Low level components serve to determine the format of the input and the output of the system. The characteristics of our input and output were briefly mentioned in Section 3.1. In this section, their formats are precisely defined (Sections 3.2.1 and 3.2.3).

High level components are the ones that actually produce the camera positions and movements. High level component categories and their relations present a framework to implement and integrate several individual camera placement methods. Their definitions are also in this section (Sections 3.2.4 and 3.2.5).

The totality of our camera control technique emerges from the collaboration and interaction of various components. This section only defines how they are classified and how they interface with each other. This distributed structure with clearly defined interfaces facilitates the extensibility of our system.

The mechanisms individual components use to obtain better camera positions and movements will be described later (Sections 3.3.1 and 3.3.3).

3.2.1 Object and Scene

Objects are three dimensional geometrical entities that populate a *scene*. A scene is the subject of our camera placement problem. Objects are the primitive elements of a scene in our system, meaning that parts of an object are not taken into consideration while dealing with the scene.

An object O is defined as follows:

$$O = \langle \vec{c}, r, \vec{p}, o \rangle$$

$\vec{c}$: Center of the bounding sphere of the object

r : Radius of the bounding sphere of the object

$\vec{p}$: Position of the object

o : Orientation of the object represented as a quaternion

The bounding spheres of objects are used in the definition instead of their precise shapes. We have found this approximation to be generally adequate for the purpose of placing cameras.

Objects in a scene can be animated. An animated object O_A is defined in terms of a bounding sphere and a series of object keyframes K_O:

$$
\begin{aligned}
K_O &= \langle \vec{p}, o, t \rangle \\
O_A &= \langle \vec{c}, r, \{K_{O_1}, K_{O_2}, \ldots, K_{O_n} | K_{O_i}\,.\,t < K_{O_{i+1}}\,.\,t\} \rangle \\
t &: \text{ Time value of the keyframe} \\
K_{O_i}\,.\,t &: \text{ Time value of keyframe } K_{O_i}
\end{aligned}
$$

As it was mentioned in Section 3.1.3, our technique assumes that the objects are solid and their shapes do not change throughout the animation. Therefore, an animated object has an unchanging bounding sphere, but a position and an orientation which vary with time.

Animation of the object position and orientation needs to be expressed in a series of object keyframes. Each keyframe holds a time value, and the position and orientation of the object at that time. To obtain the position and orientation of an object at an arbitrary point in time, keyframes are linearly interpolated.

A scene N is simply a collection of objects:

$$N = \{O_1, O_2, \ldots, O_n\}$$

An animated scene N_A contains animated objects:

$$N_A = \{O_{A_1}, O_{A_2}, \ldots, O_{A_n}\}$$

All the objects in a scene share a world coordinate system. Animated objects in an animated scene also share a time dimension.

3.2.2 Camera

Besides objects and scenes, another fundamental concept in our system is a *camera*. The entire purpose of our system is appropriately placing cameras to view scenes. Here a camera C is defined:

$$C = \langle \vec{p}, o, fov, a, n, f \rangle$$

$\vec{p}$: Position of the camera

o : Orientation of the camera represented as a quaternion

fov : Vertical field of view angle

a : Aspect ratio

n : Near clipping plane distance

f : Far clipping plane distance

Position and orientation constitute the *view parameters* and the remaining are *projection parameters*, according to the common computer graphics terminology. The emphasis in our system is on finding appropriate view parameters. Projection parameters are mostly dictated by concerns external to camera placement. Aspect ratio depends on the medium in which the output video will be presented, and clipping plane distances —aside from the need to satisfy a few requirements— do not affect the viewers' perception and understanding of the scene. Field of view angle, however, in some cases can be used to communicate some aspect of the scene to the viewer.

The aspect ratio and clipping distances are nevertheless included in the camera definition. Even though the camera placement functionality does not have a say in determining these parameters, they are required for evaluating the cameras (see Section 3.2.4).

There is not an animated camera counterpart to animated objects and scenes. The reason is that in our approach cameras are not intended to have identities. Cameras are not taken from an inventory and then placed in the scene, but rather

they are (conceptually) created whenever needed and then destroyed. The next section clarifies this aspect of cameras.

3.2.3 Shot and Shot Sequence

Shots are the mechanism by which a camera is animated. Each shot has a beginning time and end time, and defines completely the parameters a camera will take between these two points in time. Time values are required to be in synchronicity with the time values of the scene animation.

A shot as defined in our technique is comparable to the concept of shot in cinematography. A real world shot is an uninterrupted recording of a camera. During an uninterrupted recording, the camera may have stayed stationary, or it may have moved in some way. An instantaneous change of camera position or orientation in a film marks the end of a shot and beginning of another. See the difference between a moving camera and a cutting camera in Section 2.2.

Similar to the animated object definition, a shot S is defined in terms of camera keyframes K_C:

$$K_C = \langle \vec{p}, o, fov, t \rangle$$

$$S = \langle a, n, f, t_b, t_e, \{K_{C_1}, K_{C_2}, \ldots, K_{C_n} | K_{C_i} \,.\, t < K_{C_{i+1}} \,.\, t \wedge K_{C_i} \,.\, t \in [t_b, t_e]\} \rangle$$

$\vec{p}$: Position of the camera

o : Orientation of the camera represented as a quaternion

fov : Vertical field of view angle

t : Time value of the keyframe

a : Aspect ratio

n : Near clipping plane distance

f : Far clipping plane distance

t_b : Beginning time of the shot

t_e : End time of the shot

A complete video is made up of a series of shots that are subsequent in time. A *shot sequence* Q, a series of shots which may constitute the entirety of the output or a part of it, is defined as follows:

$$Q = \langle t_b, t_e, \{S_1, S_2, \ldots, S_n | S_i \,.\, t_e = S_{i+1} \,.\, t_b \wedge S_i \,.\, t_b \geq t_b \wedge S_i \,.\, t_e \leq t_e\} \rangle$$

t_b : Beginning time of the shot sequence

t_e : End time of the shot sequence

$S_i \,.\, t_b$: Beginning time of shot S_i

$S_i \,.\, t_e$: End time of shot S_i

This definition basically ensures that the shots in a shot sequence do not overlap in time; and there are not any gaps between the beginning and end times of the shot sequence in which no shot is responsible for supplying the camera

parameters. The final output of our system can be considered a shot sequence whose time interval covers the entire input animation.

The concepts introduced up to this point can be useful in any kind of application in which objects and cameras are animated. The following concepts, on the other hand, explicitly involve automatic camera placement.

3.2.4 Objective Function

Objective functions evaluate cameras and shots by various criteria. An objective function F is defined as follows:

$$F : \{\langle C, N\rangle\} \rightarrow \mathbb{R}$$
$$C : \text{ Camera to be evaluated}$$
$$N : \text{ The scene which is the subject of the camera}$$

The output of an objective function indicates the *score* of the camera according to the evaluation criterion of that particular objective function. Higher scores indicate more favorable camera placements for viewing the scene.

An animation objective function F_A evaluates shots for viewing animated scenes:

$$F_A : \{\langle S, N_A\rangle\} \rightarrow \mathbb{R}$$
$$S : \text{ Shot to be evaluated}$$
$$N_A : \text{ The animated scene which is the subject of the shot}$$

Objective functions are primarily used for obtaining favorable shots. Since the adequacy of shots can be compared, optimization techniques that aim to produce better shots can be devised. A straightforward optimization technique is getting

a large number of shots which are constructed randomly and selecting the one with the highest score.

As mentioned above, each objective function evaluates its inputs by a single criterion. However, most of the time the shots that make up an output video need to satisfy several such criteria at the same time. In order to evaluate a camera or a shot by multiple criteria, scores from several objective functions can be weighted, summed and normalized to obtain a single score (see Section 3.3.1.6).

3.2.5 Task

It has been mentioned that the final output of our system is a series of shots. *Tasks* are the components that are responsible for producing these shots. A task T is defined as the combination of a shot generation function G and a relevance function R (which will be explained later):

$$
\begin{aligned}
T &= \langle G, R \rangle \\
G &: \{\langle t_b, t_e, N_A \rangle\} \rightarrow \{Q | Q \,.\, t_b = t_b \wedge Q \,.\, t_e = t_e\} \\
R &: \{\langle t_b, t_e, N_A \rangle\} \rightarrow \mathbb{R} \\
N_A &: \text{ The animated scene to be viewed} \\
t_b &: \text{ Beginning of the input time interval} \\
t_e &: \text{ End of the input time interval} \\
Q \,.\, t_b &: \text{ Beginning time of shot sequence Q} \\
Q \,.\, t_e &: \text{ End time of shot sequence Q}
\end{aligned}
$$

The shot generation function of a task, when given a time interval, returns a shot sequence that spans that time interval.

Tasks are the most important components of our system. The semantic information that comes from the user is interpreted and converted to geometric information by the tasks. Neither shots, nor objective functions have any notion

about the meaning and significance of the objects and their movements.

In addition to reasoning about the semantic information about a scene, tasks themselves are the way the users express that semantic information: Each task has a definition users can understand, and users select a task to be active if they want to achieve that task by showing the output video to viewers.

Tasks mostly need to be associated with a particular object in the scene or a group of objects. The user, when selecting such a task to be active, must indicate which object(s) the task applies to.

Since tasks cannot manipulate the scene or the animation, their means to communicate various aspects of the scene is the placement of cameras and the sequencing of shots. Even though tasks cannot manipulate the scene configuration, they can detect the scene configurations where they are able to be more effective. Tasks carry out this evaluation through their relevance functions R:

$$R : \{\langle t_b, t_e, N_A \rangle\} \rightarrow \mathbb{R}$$

The relevance function of a task gives higher relevance scores to time intervals in which the task will be more successful in communicating the particular aspect of the scene it was designed to. This relevance score is used by the presentation planner (Section 3.3.4) to efficiently assign time intervals to tasks.

To illustrate the task concept, consider a task which emphasizes and exaggerates the size difference between two objects. The user may decide to use this task to point out the huge size of an object in the scene. The user then selects that object, and selects another object to make the size comparison. When the task is asked to produce a shot sequence, it can return a shot in which the camera looks at the big object with a low angle (i.e. from below), while ensuring that the small object falls in the frame too. If the viewers are familiar with the size of the small object, the resulting video can successfully communicate the great size of the other, possibly unfamiliar object. Furthermore, this task can indicate that it is more relevant when the two objects are close to each other; since in that case

it can place the camera at a lower angle.

The task in the above example can make use of an objective function that gives higher scores to views with low angles (See Section 3.3.1.2), together with another objective function that gives higher scores to views in which the small object is visible (See Section 3.3.1.4). Or it can decide not to use objective functions and produce its outputs by evaluating a closed-form equation. Even though objective functions are a helpful mechanism for coming up with better shots, their use by tasks is not mandatory.

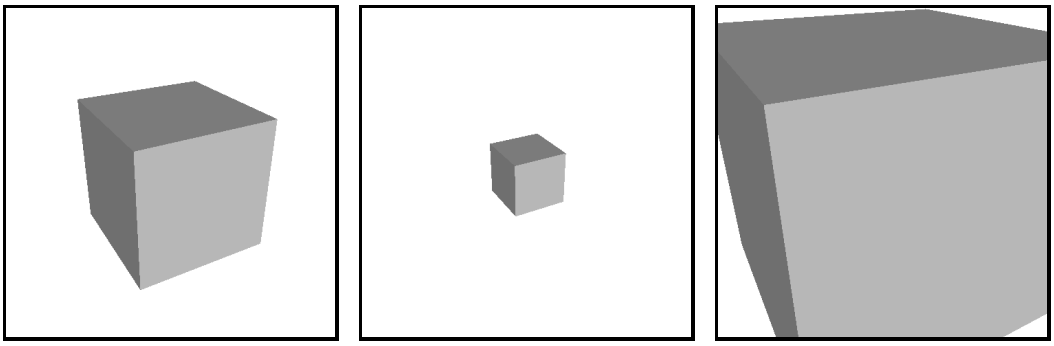
3.3 System Components

So far it is established that the automatic camera placement behavior of our system arises from the collaboration of several components, each of which carry out a well-defined function. After the explanation of categories that the components can belong to (Section 3.2), now several sample components from each category is illustrated in this section. The following discussion also touches on implementation-related considerations.

3.3.1 Objective Functions

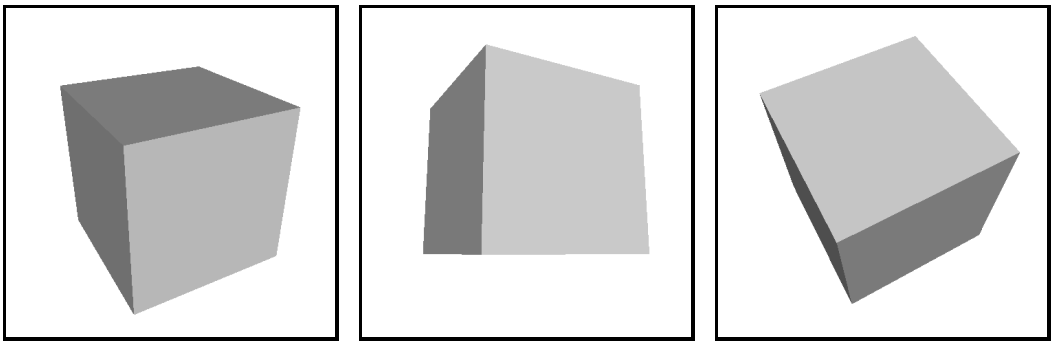
3.3.1.1 Visible Size

Visible size objective function is used to ensure that the objects that are the target of a shot cover a reasonable portion of the screen. The "reasonable portion" value is configurable. Given the camera and an object, visible size objective function gives higher scores when the ratio of the screen covered by the object is close to the indicated ideal value. Figure 3.1 demonstrates the scores given to some sample views.



(a) Score: 0.95 (b) Score: 0.53 (c) Score: 0.36

Figure 3.1: Scores given to sample camera-object combinations by visible size objective function. The ideal ratio for the object to cover on the screen is set to 0.75.

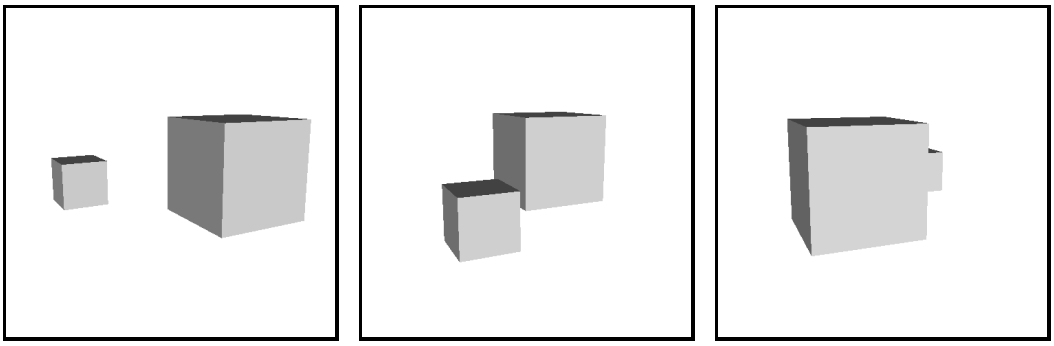


(a) Score: 0.77 (b) Score: 0.18 (c) Score: 0.02

Figure 3.2: Scores given to sample camera-object combinations by elevation objective function. The ideal elevation angle of the camera is set to 20 degrees.

3.3.1.2 Elevation

Elevation objective function evaluates the vertical angle between the view direction and the horizontal plane. The ideal vertical angle (elevation angle) of the camera is configurable. Using this objective function, it is possible to obtain shots that look downwards or upwards. Scores for cameras at different vertical angles can be seen in Figure 3.2.



(a) Score: 1 (b) Score: 0.41 (c) Score: -0.11

Figure 3.3: Scores given to sample camera-object combinations by occlusion objective function.

3.3.1.3 Occlusion

Occlusion objective function measures the amount of occlusion between the objects in a scene, and gives higher scores to views with less occlusion. This objective function is used to ensure that when several objects fall in the frame, they are all reasonably visible. Figure 3.3 shows scores given to several views with different occlusion amounts.

3.3.1.4 Object Inclusion

Object inclusion objective function evaluates a view based on the visibility of a particular object. If the object is not in the frame at all, object inclusion objective function gives a very large negative score. Since most of the shot types (See Section 3.3.2) have the concept of a "target object" which is unconditionally visible, this objective function is usually used to obtain views in which another object is visible together with the target object.

Note that occlusion objective function and object inclusion objective function serve different purposes and use different mechanisms, even though both of them involve visibility.

3.3.1.5 Closing

Closing objective function is an animated objective function that favors shots during which the camera comes close to, or moves away from, or keeps a constant distance to a particular object. The desired movement of the camera is indicated through an ideal closing amount value.

3.3.1.6 Aggregate Objective Function and Animation Adapter

Aggregate objective function and *objective function animation adapter* do not analyze geometric properties of the camera and the scene like the other objective functions. Rather, these two objective functions serve as helpers to ease the use of other objective functions.

Aggregate objective function holds several objective functions, and passes the scene and the camera given to it to each of its objective functions. It gathers the scores from the objective functions and aggregates them to a single score by calculating a linear combination of them.

Objective function animation adapter, on the other hand, facilitates the use of static objective functions as animation objective functions (See Section 3.2.4 for definitions of these two kinds of objective functions). This adaptation is accomplished by taking several samples from the camera and scene animation during a shot; and aggregating the scores given by a static objective function for each of the samples. Whether this adaptation technique is appropriate or not should be judged for each objective function separately.

3.3.2 Shot Types

3.3.2.1 General Considerations

Shot types need to be distinguished from *Shots*(Section 3.2.3). Shots are the building blocks of a camera placement plan. They supply the low-level camera parameters during their assigned time interval. It is possible, albeit tedious, for a user to write down the low-level parameters of the camera by hand for several consecutive frames of the animation to come up with a shot. Shot types, on the other hand, are software components that are responsible for calculating this long string of low-level parameters from much fewer high-level —but still geometric— parameters, whose nature depend on the particular shot type.

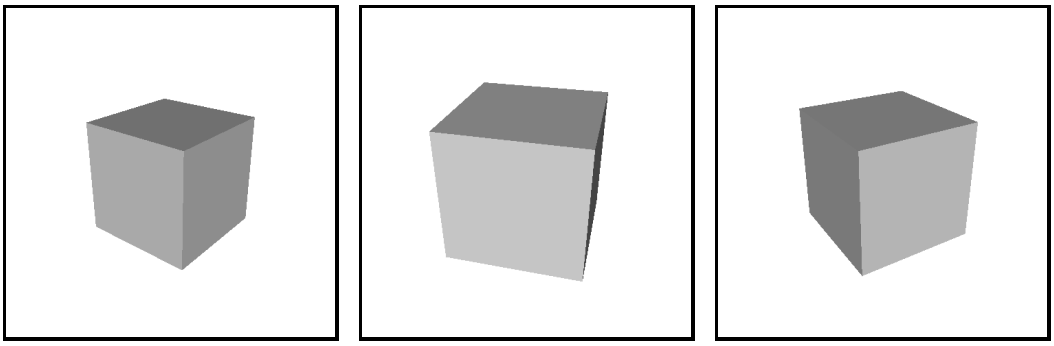
Tasks use shot types to decouple themselves from the low-level geometric calculations. After analyzing the animation, tasks only need to decide which type of shot to use, and provide that shot type with a few required parameters. This separation of geometric calculations also facilitates reuse: Many different tasks may utilize the same shot type.

Each shot type is made up of two sub-components: A *shot driver* and a *shot constructor*. Shot drivers are responsible for placing the camera when given a time value. Shot constructors, given their high-level input parameters, obtain further *mid-level* parameters to guide the shot drivers (See Table 3.1). Most of the shot constructors carry out optimization procedures to come up with the mid-level parameters. These optimization procedures make heavy use of objective functions.

Currently, the optimizing shot constructors use the rudimentary approach of producing several random sets of parameters and selecting the one with the highest score according to their objective functions. The introduction of randomness to camera placement at this stage has the effect of providing the indispensable factor of variety to the output video.

Table 3.1: The geometric parameter classes involved with shot types

Parameter class	**Description**
High-level	High-level parameters come from the entity which is requesting the shot. They minimally define the shot. For example, if the *target object* for a shot is dictated from outside by a task, the target object is a high-level parameter.
Mid-level	Most of the time, high-level parameters are not enough by themselves to derive the final camera parameters. Shot constructors provide additional mid-level parameters which augment the high-level parameters to arrive at the precise mathematical definition of a shot. Once the mid-level parameters are complete, low-level parameters can be deterministically calculated. For example, if the distance between the object and the camera is not among the high level parameters for a shot type, it is a mid-level parameter.
Low-level	The parameters required to ultimately specify the configuration of a camera are low-level parameters. Camera position and camera direction are among these. For example, camera position and camera direction can be obtained from mid-level parameters such as distance, horizontal angle and vertical angle through trigonometry calculations.



(a) $t = 0.29$ (b) $t = 1.12$ (c) $t = 2.42$

Figure 3.4: Sample stationary following shot follows the cube as it moves to the right.

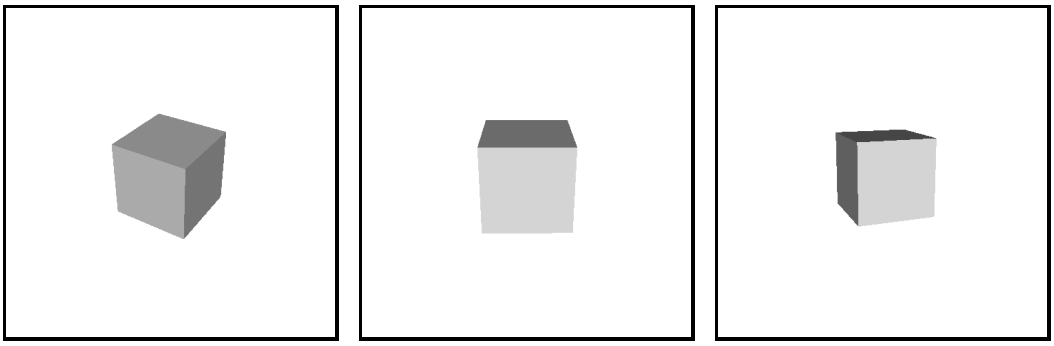
3.3.2.2 Stationary Shot

Stationary shot is the most basic shot type. It does not perform any camera animation. When the stationary shot constructor is given a non-animated target object, it selects a single position and orientation from which that object can be seen.

3.3.2.3 Stationary Following Shot

Stationary following shot keeps its target object in the view while keeping the position of the camera constant. When given a target object and a time interval, stationary following shot selects a point in space to place the camera; and keeps the camera there for the duration of the shot. Figure 3.4 shows frames from a camera controlled by a stationary following shot.

If the target object is desired to come closer to or move away from the camera, the "ideal closing amount" value of the closing objective function (Section 3.3.1.5) used by the shot can be configured accordingly.



(a) $t = 0.32$ (b) $t = 1.5$ (c) $t = 2.4$

Figure 3.5: Sample circling shot moves the camera around the cube.

3.3.2.4 Circling Shot

Circling shot, as its name implies, circles the camera around its target object. For the duration of the shot, the camera is directed to the object, but the camera's relative position to the object changes. Figure 3.5 shows frames from a camera controlled by a circling shot.

3.3.2.5 Restricted Following Shot

Restricted following shot is similar to stationary shot in that it does not vary the orientation of the camera that it controls. However, restricted following shot animates the position of the camera so as to keep its target object in the view.

3.3.2.6 Over-the-Shoulder Shot

Over-the-shoulder shot works with two target objects. One of the target objects is designated as the *near object*, while the other is the *far object*. Over-the-shoulder shot keeps the camera on the imaginary line that connects the two objects, from the side of the near object. Metaphorically, it looks at the far object over the "shoulder" of the near object.

3.3.2.7 Shot Sequence

Shot sequences, defined in Section 3.2.3, are implemented in our system simply as another shot type. This shot type does not produce or use geometric parameters. It holds several other shots, sorts them by their beginning and end times; and when asked to place the camera it delegates the responsibility to the appropriate shot in the sequence.

3.3.3 Tasks

3.3.3.1 General Considerations

As explained in Section 3.2.5, tasks are responsible for producing shots for any time interval assigned to them. When a task is given a time interval, everything about the configuration of the camera is controlled by that task during the interval. Even though tasks are not externally constrained about the way they place the camera, they still observe several rules about using the time interval.

When given a long time interval, tasks try to divide it to shorter intervals and generate a shot for each short interval. This is sometimes necessary for the task to be successful in communicating the desired aspect of the animation to viewers. Even when it is not necessary, this behavior keeps the presentation interesting by providing visual variety (See Section 2.2.3).

Tasks, when they decide to produce a sequence of shots rather than a single one, take measures to prevent the occurrence of two consecutive shots showing the same object. This phenomenon is known as a "jump cut" in cinematography, and it is a jarring experience for the viewer.

Another consideration in producing shots is the need to display the objects at the moments they perform a significant action. As mentioned in Section 3.3.4, the significance of some actions can only be judged by the user preparing the presentation. However, other actions can be decided to be significant by analyzing

the animation data. The tasks in our system take into account the moments where an object changes the speed or direction of its motion and tries to produce shots that show the object at those significant moments.

3.3.3.2 Introduction Task

Introduction task aims to generally present an object to the viewers, without further special considerations. Its name comes from its intended use of *introducing* the objects in a scene to viewers. An object introduced this way receives the attention of the camera, and therefore it is implied to viewers that the object in question carries importance, or it is the protagonist in the story of the animation. In a complex animation with many objects, introduction tasks can be associated with significant objects to visually distinguish them from the others.

Introduction task also allows users to control the order of introduction of objects. When an object is associated with introduction task, a time interval for the introduction can also be specified. This way, if the user wishes to communicate that a particular object "joins" some activity at some point in time, he or she can request from our system for that object to be introduced at the appropriate time (See the discussion about forcing a task to be active at a desired time interval at the end of Section 3.3.4).

Introduction task uses a circling shot (Section 3.3.2.4) to show its target object. Circling shot both emphasizes the target object, and gives a good overview of its shape by moving the camera around it.

3.3.3.3 Approaching Task

Approaching task is used for pointing out that two objects are approaching each other. Approaching task tries to both convey the notion of approaching, and suggest the directions of approaching objects relative to each other. In the shot sequences returned by an approaching task, the two objects are alternatingly

shown from angles where the other object is either directly in front, or is directly behind the camera. Approaching task uses over-the-shoulder shot (Section 3.3.2.6) and restricted following shot (Section 3.3.2.5).

An approaching task associated with two objects returns high relevance scores when the two objects are actually approaching each other. When given a time interval, approaching task checks the rate of decrease of distance between two objects. Relevance score for that interval is proportional to the distance decrease rate.

3.3.3.4 Confrontation Task

Confrontation task focuses on two selected objects and their spatial relations. The word "confrontation" is used in a general sense here: The camera placement does not assume an emotional situation such as hostility or rivalry; the purpose of confrontation task is rather to emphasize that two objects are interacting with each other in some manner.

Similar to approaching task, confrontation task tries to ensure that viewers are aware of the relative directions of the two objects. Confrontation task uses over-the-shoulder shot (Section 3.3.2.6) and stationary following shot (Section 3.3.2.3).

The time period of the confrontation is simply deduced from the proximity of the target objects. Confrontation task returns higher relevance scores when the two target objects are near each other.

3.3.4 Presentation Planner

The *Presentation Planner* is the component of our system that assembles and outputs the final, complete camera placement plan. Since individual tasks are responsible for assessing the situation in the scene and producing shots that cover a given time period, the responsibility left to the presentation planner is assigning

time intervals to tasks, without gaps or overlaps. Once the tasks output their shots or shot sequences for their intervals, the presentation planner concatenates them and passes the complete camera placement plan to the rendering component.

The presentation planner needs to be aware of all the active tasks in order to obtain shots from them. This makes the presentation planner the most directly user-facing component, since it is the user who selects the tasks.

The criteria for deciding which task is responsible for each interval are the values of the relevance functions of tasks. For each interval, tasks are asked for their relevance during that interval. The task with the highest relevance value is assigned to the interval. This ensures that the aspects of the scene that the user wants to convey to the viewers are emphasized at the appropriate moments in the animation.

In fact, the task-interval assignment functionality of the presentation planner can be considered in two parts: dividing the duration of the animation to assignment intervals; and assigning those intervals to tasks. In order to increase the effectiveness of assignment, these two phases are not carried out independently from each other. Just as the partitioning of time affects the assignment of tasks; task assignments affect the way animation time is partitioned. This mechanism is used to avoid assigning consecutive separate time intervals to the same task. In cases where one task is relevant for an extended period of time, that task gets an undivided long interval instead of several short ones. The task can then use the interval more effectively.

Users may wish to override the task-interval assignment decided by the presentation planner. Some fact about the animation that cannot be deduced from its geometric definition, some piece of knowledge that relates to the meaning of the animation may dictate that at a particular time interval a particular object needs to be shown. Our system makes the overriding of task-interval assignment possible without any modifications to the architecture: The tasks that are declared by the user to be responsible for a time period give relevance values which are above the range for computed relevance values during their user-assigned time period.

Table 3.2: Hierarchy of components, from high-level to low-level

Layer	**Functionality**
Presentation Planning Layer *(Presentation Planner, Tasks)*	Prepares the final output of the system, which is a sequence of shots. Tasks, which are in this layer, request the individual shots from the optimization layer.
Optimization Layer *(Shot constructors*, Objective Functions)*	Given the target, time interval and type of shot, shot constructors compute appropriate shot parameters. They make use of objective functions as the criteria to produce better shots.
Camera Animation Layer *(Shot drivers*)*	The output of shot constructors contain the shot type and defining parameters for that type of shot. Shot drivers, on the other hand, are responsible for calculating all the low-level parameters of a camera at any point in their assigned time interval.
Scene Layer *(Object, Scene, Camera)*	The objects to be shown, and the camera for the objects to be shown through are the fundamental entities in our system. The components in this layer both serve the upper layers as their subject matter, and constitute the system's connection to the external rendering components.

* See Section 3.3.2.1 for definitions of shot constructors and shot drivers.

The way the presentation planner and other components interact is summarized in Table 3.2.

Chapter 4

Results and Discussion

In this chapter, a complete input animation and the output video depicting that animation are examined. Moreover, the power of abstractions that are proposed in our approach is compared to those of other automatic camera control techniques in the literature.

4.1 Demonstration

In the following discussion a sample run of our system is demonstrated in detail. This demonstration is intended to give a concrete example of the automatic camera placement procedure in its entirety.

Our sample input animation features a light airplane and an attack helicopter. The helicopter is both faster and more maneuverable than the airplane. The animation involves the two aircraft coming across each other and the helicopter briefly harassing the airplane.

To direct the camera placement for viewing the input animation, three tasks have been selected and given to the presentation planner:

- An introduction task (Section 3.3.3.2) is associated with the airplane for

a time interval at the beginning of the animation (See Section 3.3.4 about explicitly assigning a task to a time interval). This task is responsible for starting the video with a shot showing the airplane in order to establish it as the main participant of the animation.

- An approaching task (Section 3.3.3.3) is associated with the two aircraft. It is responsible for showing that they come closer to each other (They indeed come closer for some time at the beginning of the animation). The helicopter is designated as the object that approaches the airplane.
- A confrontation task (Section 3.3.3.4) is associated with the two aircraft. It is responsible for bringing the engagement of the two aircraft into attention.

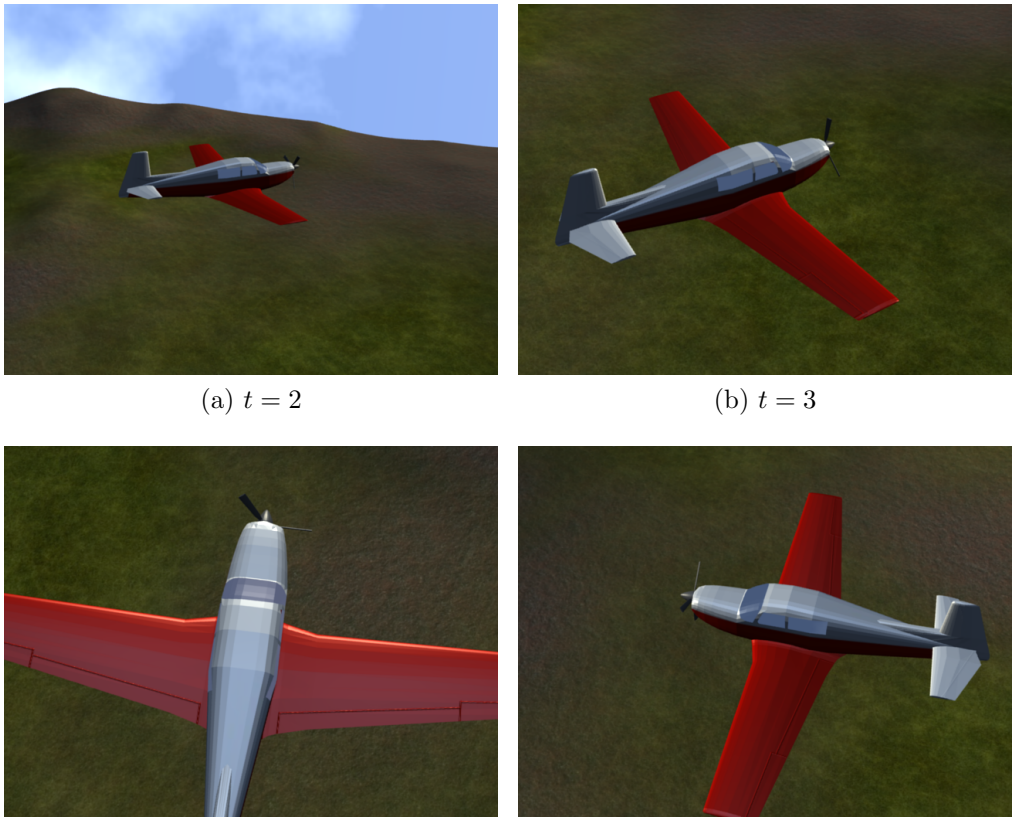
4.1.1 Output Video

The animation and the task set described above constitute a complete piece of input data which can be used for calculating a camera placement plan. This input has been given to our system; the output of our system has been recorded; and this output in turn has been given to an external rendering component. The rendering component has rendered the animation using the camera positions and orientations described by the camera placement plan. Below are frames from the output video, along with comments about the operation of our system as it applies to this particular input.

Since an introduction task was associated with the airplane and this task was determined by the user to be active at the beginning, the video begins with a shot that shows the airplane (Figure 4.1).

Once the assigned time interval of introduction task ends, it begins returning very small relevance values in order to avoid inadvertently being assigned to another time interval. From then on, relevance values of the other two tasks are compared by presentation planner to decide which one of them becomes active.

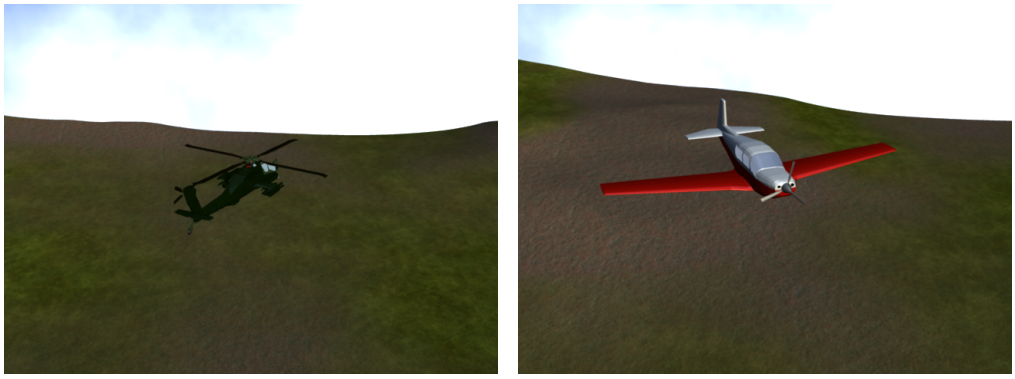
For some time, the two aircraft become closer to each other as they fly in their straight courses. During this period, since the distance between them decreases



(a) $t = 2$ (b) $t = 3$

(c) $t = 4$ (d) $t = 5$

Figure 4.1: Sample frames from the shot produced by introduction task



(a) $t = 7$ (b) $t = 10$

Figure 4.2: Sample frames from shots produced by approaching task

steadily, approaching task returns high relevance values. And since they are not yet close enough to each other to produce shots that can clearly show them together, confrontation task returns low relevance values. Therefore approaching task takes control of the camera during this period. It produces shots that show the two objects individually, while placing the camera so as to suggest the direction of the unseen object relative to the object that is being shown (Figure 4.2).

In Figure 4.2a, the airplane, although not visible, is at the direction the camera is facing. In Figure 4.2b, the helicopter is directly behind the camera.

At one point in the animation, the helicopter intercepts the airplane and begins harassing it. The airplane tries to evade the helicopter. During this part of the animation the aircraft are close enough for confrontation task to take over the control of the camera. Confrontation task produces shots that emphasize the relation between the two objects in space (Figure 4.3).

Confrontation task takes into account the moments at which a significant change occurs in the movement of an object, and tries to show that object at those moments (see Section 3.3.3.1). Figure 4.4 shows such a significant moment where the airplane changes direction.

(a) $t = 20$ (b) $t = 23$

(c) $t = 24$ (d) $t = 35$ (detail)

(e) $t = 37$ (f) $t = 47$ (detail)

Figure 4.3: Sample frames from shots produced by confrontation task

(a) $t = 28$ (b) $t = 29$ (c) $t = 30$

Figure 4.4: Direction change of the airplane captured in the output video

4.1.2 Possible Alternative Task Sets

The selected tasks and the corresponding output demonstrated above represent one of the many possible interpretations of the input animation. The user may prefer to interpret the input in a different way, and select the tasks accordingly.

The user may decide that the helicopter, not the airplane, is the main object in the animation. In that case, he or she simply associates the introduction task with the helicopter instead of the airplane.

The user also may want the encounter of the two aircraft to be a surprise to the viewer. This effect can be accomplished by omitting the approaching task and extending the time interval of the introduction task.

4.2 Comparison

In this section, our approach is compared to selected existing automatic camera placement techniques. Note that the techniques mentioned here might have been developed under different assumptions, so the following comparisons do not necessarily indicate that one technique is more adequate than another. The purpose of this discussion is to evaluate how suited our technique is to the target application area described in Section 3.1, compared to other techniques.

Table 4.1: Convenience criteria for automatic camera placement techniques

Criterion	**Description**
Lack of need for user attention	Users should not need to select the object of interest for every moment of the animation. Camera placement procedure should be able to complete with minimal interaction with users. Conflicts in camera placement needs should be able to be resolved by the system silently.
Lack of need for complementary input about the animation	Camera placement technique should require only position, orientation and other geometric data about the objects in the animation. It should not depend on the animation being output from a specific system which integrates well with the camera placement system.
Cinematography expertise	Camera placement technique should be aware of accepted cinematography practices and apply them. Users should be able to obtain a result which looks pleasant without having technical information about positioning cameras.
Visual variety	The output of the camera placement system should show sufficient variety. The objects should not be viewed from same angles and distances every time the system is run. Users should not need to modify any parameters to obtain a slightly different camera placement.

4.2.1 Convenience

Camera control techniques require different levels of effort and skill from their users. A technique that provides greater convenience requires less effort and skill. Our technique is intended to be used by unskilled users, with a minimal amount of decision making about the interpretation of the input animation. Furthermore, our technique accepts bare geometric descriptions of the objects and the animation. This ensures that animations from diverse sources can be the subject of our camera placement technique. Several criteria for evaluating the convenience provided by a camera placement technique are presented in Table 4.1.

Two techniques proposed by Bares and Lester [2], [3] require the user to manually determine the object of interest at all times. Another technique of theirs [1] does not have that requirement, however, this technique depends on the input animation being strictly defined through their *interactive fiction* framework. The input to the technique needs to be annotated with semantic information about the events, and the camera placement behavior is strongly coupled to the set of events that are defined in the system.

The CamDroid system proposed by Drucker and Zeltzer [6] does not require the user to continuously indicate the object of interest. However, their mechanism for determining the object of interest depends on semantic information being provided alongside the geometric information about the scene.

The technique described by Christianson et al. [4] also needs the input animation to be in an extended format which both contains semantic information, and the object of interest for every moment. The sequence planner component is dependent on the set of events it can understand. The successor to this technique, proposed by He et al. [8] works in interactive situations but still needs a stream of events with semantic information.

The automatic cinematography system described by Tomlinson et al. [12] needs the participants of the animations to be interactive agents with *emotions* and *motivations*. The system depends on the emotions and motivations of the

Table 4.2: Expressiveness criteria for automatic camera placement techniques

Criterion	Description
High-level facilities to affect camera placement	Users should be able to express their expectations from camera placement through high-level, understandable concepts. Users should not need to manipulate geometric quantities to achieve their objectives.
Means for accommodating local interventions to automatic camera placement	Users should be able to override individual camera placement decisions without disrupting the rest of the output.

participants, hence cannot work with animations from an arbitrary source.

4.2.2 Expressiveness

Relieving the users from the burden of attending to camera placement is a desired characteristic of camera control techniques. On the other hand, allowing the users to do so when they want to is also a strength. Our technique allows users to direct camera placement through tasks, which do not require them to occupy themselves with geometric details. An expressive camera control technique provides the users with such practical means. In general, convenience and expressiveness are at odds with each other. Unless special measures are not taken, an increase in convenience means a decrease in expressiveness. Several criteria for evaluating the expressiveness of a camera placement technique are presented in Table 4.2.

Bares and Lester [2], [3] and Bares et al. [1] provide the users with stylistic choices that globally affect the camera control behavior. Users are not able to associate their preferences with objects, events or time periods; preferences are applied to the whole camera placement behavior. In one of these papers [1] camera placement is sensitive to events in the scene, however the semantic information that is consumed by the camera placement system does not come from user preferences. It is rather an integral quality of the input animation.

Drucker and Zeltzer [5], [6] allow the users to affect camera placement through

simple interfaces. However, the simplified means of camera control still remain in the geometric domain, meaning that users need to keep details like camera position and direction in mind.

Kennedy and Mercer [9] provide powerful means to affect camera placement in the form of *themes* and *moods*. Their system includes a knowledge base that contains cinematography techniques appropriate for various themes and moods. The system is intended to allow the user to affect the feelings of viewers. Enabling users to emphasize desired spatial relations between objects in an animation does not seem to be a priority of the system, but presumably the knowledge base and the collection of themes and moods can be extended to serve that purpose.

Tomlinson et al. [12], similarly to Kennedy and Mercer [9], focus on communicating feelings. They associate cinematography techniques with emotions. However, these emotions are not determined by the user. Emotions are the products of interaction between the autonomous agents in the system. Even though the system is expressive, it is not expressive of user's interpretation of the animation.

4.2.3 Extensibility

Since our system is made up of several components with well-defined interfaces (Section 3.2), it is eminently extensible. For example, a new task can be introduced to the system, and the new task can utilize existing shot types and objective functions. A new objective function can be added and made to be used by an existing shot type. Few of the camera control techniques mentioned here support this level of extensibility.

Drucker and Zeltzer [5], [6] implement their camera control technique in several *camera modules*. These modules are of a common construction, and each module makes use of a single constraint solver. Modules are meant to have their own interfaces to the user. When a new module is desired to be incorporated into the system, the constraints to be satisfied while the module is active need to be

determined, along with the module's user interface and its internal state.

The modular camera control components presented by Christianson et al. [4] and He et al. [8] are *idioms*. Idioms are designed to have a uniform interface and work in collaboration with each other, similar to the components in our system. New idioms introduced to the system can use existing camera motion capabilities.

Even though camera modules and idioms are modular constructs, the systems that they are parts of do not provide more extensibility than their modularity. The components they use, or the components that use them are not similarly modular.

Chapter 5

Conclusion

The motivation for this thesis is helping people prepare videos from 3D animations. Specifically, the system we propose automates the placement of virtual cameras through which the animation is recorded to the output video. This system aims to relieve its users from keeping geometric concepts such as positions and directions in mind. While handling these details, we also want to give users high-level control over the camera placement.

In order to achieve these objectives, we propose encapsulating camera control intelligence in *tasks*. Each task has a distinct, understandable definition about what it communicates to viewers; and it knows how to place cameras to carry out this communication. Users only need to select the tasks they want in order to control camera placement.

Tasks get assigned to time intervals during which they control the camera. They make use of other components of our system to come up with final camera parameters. Tasks are assigned to time intervals in a manner that maximizes the effectiveness of each task. The fact that this assignment is performed by our system means that users do not need to pay attention to time values just as they do not need to pay attention to geometry.

Our approach brings together the ability of users to control camera placement

and the convenience of automation to a level that is not achieved before. Also, the highly modular architecture of our system allows it to be extended more easily than any previous automatic camera control system.

In the future, our system may be improved by implementing more tasks for users to choose from. The tasks may be organized in a structured catalog instead of an ad-hoc collection. Another improvement may be restricting the task definition to allow control over visual style of all tasks, giving users another high-level mechanism to control camera placement. Such a restriction may also make it possible to incorporate cinematographic principles that apply across task-interval assignments.

Bibliography

[1] W. Bares, J. Grégoire, and J. Lester. Realtime constraint-based cinematography for complex interactive 3D worlds. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1101–1106. AAAI, 1998.

[2] W. Bares and J. Lester. Cinematographic user models for automated realtime camera control in dynamic 3D environments. In *Proceedings of the Sixth International Conference on User Modeling (UM 97) Vien, Austria*, pages 215–226. Springer-Verlag, 1997.

[3] W. Bares and J. Lester. Intelligent multi-shot visualization interfaces for dynamic 3D worlds. In *Proceedings of the 4th International Conference on Intelligent User Interfaces*, pages 119–126. ACM, 1998.

[4] D. Christianson, S. Anderson, L. He, D. Salesin, D. Weld, and M. Cohen. Declarative camera control for automatic cinematography. In *Proceedings of the National Conference on Artificial Intelligence*, pages 148–155. AAAI, 1996.

[5] S. Drucker and D. Zeltzer. Intelligent camera control in a virtual environment. In *Proceedings of Graphics Interface*, pages 190–199. Canadian Human Computer Communications Society, 1994.

[6] S. Drucker and D. Zeltzer. CamDroid: A system for implementing intelligent camera control. In *Proceedings of the Symposium on Interactive 3D Graphics*, pages 139–144. ACM, 1995.

[7] N. Halper and P. Olivier. CAMPLAN: A camera planning agent. In *Proceedings of AAAI Spring Symposium on Smart Graphics*, pages 92–100. AAAI Press, 2000.

[8] L. He, M. Cohen, and D. Salesin. The virtual cinematographer: a paradigm for automatic real-time camera control and directing. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, pages 217–224. ACM, 1996.

[9] K. Kennedy and R. Mercer. Planning animation cinematography and shot structure to communicate theme and mood. In *Proceedings of the 2nd International Symposium on Smart Graphics*, pages 1–8. ACM, 2002.

[10] J. Kwon and I. Lee. Determination of camera parameters for character motions using motion area. *The Visual Computer*, 24(7):475–483, 2008.

[11] D. Sokolov, D. Plemenos, and K. Tamine. Viewpoint quality and global scene exploration strategies. In *Proccedings of International Conference on Computer Graphics Theory and Applications (GRAPP 2006)*, pages 184–191, 2006.

[12] B. Tomlinson, B. Blumberg, and D. Nain. Expressive autonomous cinematography for interactive virtual environments. In *Proceedings of the Fourth International Conference on Autonomous Agents*, pages 317–324. ACM, 2000.

[13] P. Vázquez, M. Feixas, M. Sbert, and W. Heidrich. Viewpoint selection using viewpoint entropy. In *Proceedings of Vision, Modeling, and Visualization*, pages 273–280, 2001.