Camera Control through Cinematography for Virtual Environments: A State of the Art Report

James Kneafsey & Hugh McCabe

School of Informatics and Engineering, Institute of Technology, Blanchardstown, Dublin 15, Ireland.

james.kneafsey@itb.ie, hugh.mccabe@itb.ie

Abstract

Cinematography has evolved as a set of guidelines for filming motion pictures that ensures a standard is adhered to and the events are presented in a coherent manner. Principles of cinematography can be employed such that supplementary information is communicated when necessary and the viewer does not become disoriented due to the camera work. We propose that these and other principles of cinematography can be applied to virtual environments, in particular 3D computer games where disorientation of the user is often an issue. This paper looks at what has already been implemented in the domain of camera control through cinematography for virtual environments and outlines our future work.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Viewing algorithms

1 Introduction

We propose to implement a camera control system for 3D computer games that is informed by cinematography. Cinematography provides film-makers with a language to use when controlling the camera. Certain styles of camera control can be used to convey supplementary information regarding the action to the audience. We believe this power of communication can be adapted for computer games. Camera control for computer games may, therefore, evolve from simple configurations to more elaborate ones and perhaps evoke an emotional response from the user.

Three-dimensional computer games usually require the user to move a character around a virtual world. The virtual camera that films the action is often in a fixed position relative to the character so that the user sees the environment through its eyes or perhaps from behind and above its head. As the user turns the character, the viewpoint turns also. Camera angles like this represent only a subset of the options available from cinematography (see following section). A richer visual experience would be granted to the user if more principles from cinematography were considered in the design of the virtual camera.

Before considering camera control through cinematography, camera control fundamentals must be taken into account. Thus we examine a range of techniques that attempt to solve a number of problems relating to camera control in a virtual environment in this paper. Although our focus is on 3D computer games, it is important to consider camera control in other virtual environments because the techniques employed are still relevant. Fundamental techniques seek to simply position and orient the camera within the virtual world to generate still images. Others provide shots with a moving camera as may be used for museum walkthroughs. More elaborate approaches consider enabling the virtual camera to follow a moving subject and solving common problems associated with this task. Building on all of these methods are approaches to camera control that are informed by cinematography some of which attempt to invoke an emotional response from the user.

The rest of this paper is structured as follows: Section 2 introduces the principles of cinematography relative to camera control in a virtual environment, section 3 discusses camera control fundamentals in a virtual environment, section 4 outlines methods for controlling the virtual

camera through cinematography, and section 5 provides a conclusion and outlines our future work.

2 Cinematography

Cinematography is defined as the art of film-making. It encapsulates the techniques and principles that govern how filming should be carried out for motion pictures. More specifically, it provides guidelines for how the camera in particular should be used in order to accomplish tasks such as engaging the interest of the viewer, enhancing and clarifying the narrative, and presenting the content in an interesting and coherent manner. For a full treatment of this subject we direct the reader to classic texts such as [Mas65] and [Bro02]. For the purposes of this paper we will simply outline some basic principles relevant to camera control in a virtual environment.

A motion picture is broken down into sequences of *scenes* each of which is a single setting where action takes place during a particular time period. Each scene is composed of a number of *shots*, a shot being a continuous view filmed by one camera without interruption. The transition from one shot to the next is known as a *cut*. During a shot the camera may be stationary or it may use simple or more complex movements depending on the content of the scene and the mood that is to be established. Cinematography provides guidelines for deciding how to position and move the camera.

The type of camera angle used dictates how much the camera, and therefore the viewer, becomes part of the action. Camera angles that are closer to and move with the action are more *subjective*; angles that are more detached and seem to move independently of the action are more *objective*. The most subjective angle is achieved when the camera takes the place of one of the characters in the scene. The viewer now sees through the character's eyes as they move around the scene.

Common types of shots used in cinema are the *long shot*, which depicts the main characters and the setting, *medium shot*, which depicts characters from the thighs to above the head, and *close-up*, which depicts them from the chest to above the head. A long shot establishes the layout of a setting, a medium shot may be used to introduce the main characters and a close-up isolates and adds dramatic emphasis to a particular event.

There are a number of different types of camera movements, all of which must be executed such that the viewer does not become disoriented. Two of the main ones are tracking and panning. Tracking is when the camera moves alongside a character while filming them and panning is when it rotates on its vertical axis. Tracking can give the viewer the impression that they are walking alongside a character; panning is often used to take in the expanse of a setting.

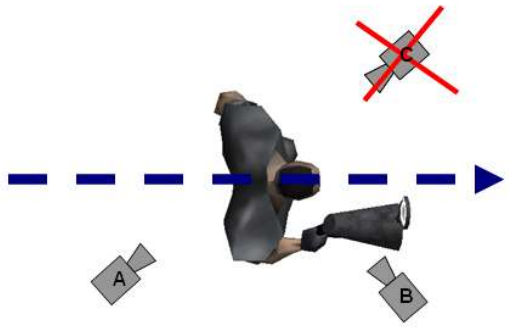

Figure 1: *The action axis: If the subject is filmed with camera A, subsequent shots must keep to this side of the action axis dictated by the subject's motion. Therefore position B is valid while C is not.*

To ensure that the action is presented in a continuous way the *action axis* (figure 1), also called *the line-of-interest*, is used. This is a line, or in a 3D environment, a vertical plane, dictated by the action being performed. If two actors are talking the line will be drawn between them; if one actor is walking the line will be drawn in the direction of their motion. In order to keep the viewer oriented, the camera must keep to one side of the action axis from one shot to the next. It may move across the action axis during a shot but once there is a cut to the next shot it must be positioned on the same side as before the cut.

3 Camera Control Fundamentals

Virtual cameras share many of the characteristics of their real-world counterparts and much of the basic positioning and movement of the camera required by cinematography can be implemented for virtual cameras also. Automating these tasks does introduce some problems however and in this section we will identify these problems and look at how they can be solved.

3.1 Still Shots

The simplest type of shot is a *still* shot, meaning that the camera is fixed in position and orientation, usually pointing at some subject. The position and orientation of the camera is controlled by a set of parameters. These can be expressed as a *lookFrom* vector which defines the camera's position, a *lookAt* vector which defines the location the camera is pointing at, and an *up* vector which defines the camera's vertical axis. In addition to these parameters there is a *field-of-view* parameter. The field-of-view of a camera is the region of the visual field that is filmed. The greater the field-of-view the greater the proportion of the setting that is included and the more it seems that objects are farther away from each other.

These parameters are sufficient to set the camera's position, location and field-of-view but for a camera system to determine them automatically some factors must be taken into account. The system should avoid situations where the required view is partially or wholly occluded by

some scene geometry. This can be accounted for by casting a ray from the camera to the subject that is to be filmed and testing for intersections with scene geometry along the way [TZY04, TBN00]. If an intersection does occur measures may be taken to find a new placement.

We are usually left with a large number of possible occlusion-free viewpoints to choose from so some authors have addressed the problem of how to select optimal ones. Halper et al. [HO00] use a genetic algorithm to automatically generate the optimal camera placement within a virtual environment given some requirements by the user. The user defines *shot properties*, i.e. requirements on how scene elements are to appear in the view, and tolerances on those shot properties. The tolerances are used to ascertain which ones are more important in calculating the view. The user can dictate, for example, that a scene element is located between two points in the viewport (the rectangular area onto which the scene is projected) or that only a certain proportion of it is visible. The required properties of scene elements can also be defined relative to those of other scene elements as they appear in the view. For example, a requirement may be made that one scene element appears twice as large as another or to the left of another. A genetic algorithm will attempt to find a camera placement that conforms to the requirements.

Genetic algorithms are a branch of artificial intelligence that uses concepts from the Theory of Evolution to find an optimal solution to a problem given an initial state. The seven elements of the camera state vector, that is the three elements of its position vector, the three elements of its orientation vector and its field-of-view, are encoded in the chromosomes of a number of genes. These are randomly distributed in a bounded search space. A combination of selection by fitness value (i.e. how well the views conform to the shot properties imposed by the user) and random regeneration evolves the population. This means that the most appropriate views survive into the next generation and the others are regenerated. The process continues until the user aborts or the optimal view is found.

3.2 Moving Camera without Subject

A camera that moves through a virtual environment, providing the user with a changing view is applicable to situations such as virtual museum walkthroughs. The user supplies a goal destination for the camera to travel to and it travels along a path to that destination. Problems that should be anticipated are occlusion of the view and collisions with scene geometry. Occlusion of the view may occur if the camera becomes positioned so close to some scene geometry (some foliage perhaps) that it takes up most of the view. The camera system may cast a ray, as outlined in the previous section, to account for occlusion. Once a clear view is found the system may switch to using a bounding volume to avoid future occlusions. At each cycle it can test whether the bounding volume intersects any scene geometry and find a new placement if it does [MC00].

Nieuwenhuisen & Overmars [NO03] address the problem of generating an occlusion-free path through a virtual environment using the probabilistic roadmap method. Given a goal position the system generates a path between the current location and the goal and the camera follows that path. Their method is suitable for architectural walkthroughs.

In order for the system to find a suitable path in a reasonable amount of time a probabilistic roadmap is created in a pre-processing stage. The roadmap is made up of a number of nodes where each node represents a valid location in the virtual world and edges between nodes represent valid paths. When the camera travels from one location to another there must be sufficient clearance from objects in the scene so there are no collisions or near collisions. Nodes in the roadmap therefore correspond to spheres in the virtual world and edges to cylinders of the same radius. This radius defines the clearance. Nodes and cylinders are only added to the roadmap if they do not intersect with any objects.

To generate a path from a starting point to a goal position in the scene, two nodes are created for both locations and a search algorithm finds the shortest path. A penalty function discourages paths that are too long or have sharp corners. After this the path is made smooth by applying circular blends to any corners that are left. When the camera travels along the path it will need to slow down at bends in the path and speed up for straight sections so a speed function analyses the path and returns a speed diagram for the optimal speed for each point on the path. The camera's orientation is set so that it looks at the location where it will be a unit of time in the future (usually one second) rather than always looking along the direction of the path. This has the effect that as it moves into a bend it will look further along the bend which gives the user better orientation.

3.3 Camera Following Moving Subject

When a user moves an avatar around a virtual environment the camera usually follows its movements. There are different models for filming the scene relative to a moving subject depending on whether or not the subject is in the view and whether or not the camera is fixed in position relative to the subject [Bro02].

- First-person perspective: The camera films subjectively so the user sees what the avatar "sees". Thus the user will not see the avatar in the view except for its hands perhaps. This model for filming is easier to implement than the ones discussed below.

- Third-person perspective over-the-shoulder camera: The camera is positioned above and behind the avatar's head. The avatar appears towards the bottom of the screen in the user's view of the scene. This configuration is more difficult to realise than the first-person camera because a clear view of the scene is required while keeping the avatar in view.

- Autonomous camera: The camera continually tries to find the best angle based on the activity the avatar is

engaged in: it might be exploring the scene, having a conversation or perhaps picking up an object. To depict the relative parts of the scene for each activity, the camera system must be informed about the objects in the scene and the activities being performed.

One way of implementing a camera that automatically follows a subject in a virtual environment is with *constraints* [HHS01, AK01, TZY04]. Constraints such as that on a specified height, distance or orientation relative to the subject can be imposed on the camera and as the subject moves around the scene it will conform to the specified constraints.

A first-person camera is trivial to implement and less problems in presenting a suitable view arise. The over-the-shoulder and autonomous views must be continually analysed, however, so that a minimum standard of camera work and the requirements imposed by the designer are adhered to. A significant issue to account for is occlusion of the view [CAH*96, HHS01, TZY04, TBN00]. This can occur due to the combined motions of the avatar and the camera. Sometimes the avatar itself may fill the entire screen and occlude most of the scene. It is important to define how much occlusion acceptable relative to the avatar and to the view of the scene. It may be the case that no occlusion of the avatar is permitted but a small amount of the scene is allowed. Another issue to consider is the movement of the camera through scene geometry. This has the ugly effect of allowing the user to see through walls and depending on the implementation some occlusion may occur also. With the over-the-shoulder and autonomous cameras rather than the first-person camera, it is important that the viewpoint moves smoothly as it follows the avatar around the scene [CAH*96, HHS01, TZY04]; otherwise the user may become disoriented and the visual experience spoiled. Measures can be added to make the camera's movements more flexible relative to those of the avatar so that even if the user directs the avatar in an erratic way the camera will move smoothly.

4 Camera Control through Cinematography

As already mentioned our focus within the area of virtual environments is on 3D computer games. In order to adapt the principles of cinematography for computer games a major issue must be addressed: Computer games are interactive but motion pictures are not. In terms of filming the action, the director of a motion picture is at liberty to reposition actors, rearrange the scene and to film multiple takes until the required shot is achieved. In computer games these options are not available and so the filming of a game may be more comparable to that of a documentary or sporting events since in these situations the camera operator must attempt to predict the action in real-time. This leads to a reduction in the number of principles from cinematography that can be employed but this is acceptable considering the lack of determinism in how the action will play out.

To implement a camera control system that is informed by cinematographic principles two main issues must be resolved. Firstly, the camera system must know the layout of the scene, the principal characters in the scene and any important objects in the scene. Secondly, the principles must be encoded in the system so it can shoot the scene in a suitable way. One way of encoding them is as film idioms. These are standard ways to shoot common scenes, e.g. a scene of two people having a conversation. One way of acquiring information about state of the scene is directly from the real-time application at regular intervals. A finite state machine may be used to decide what idioms to use. Idioms usually consist of a number of shots where a shot may be something simple like a pan or a track. Low level camera positioning and orientation for each shot is often achieved with constraints.

Christianson et al. [CAH*96] use film idioms in an off-line method to adapt cinematography for portraying 3D animations. They formalise film idioms with their Declarative Camera Control Language (DCCL). Each idiom contains a number of shots for filming each type of scene. Shots are further decomposed into fragments, where a fragment represents a simple camera movement (e.g. a pan or a track) or a static camera. An animation trace representing action that has already occurred and a number of film sequences are supplied to the camera system. The film sequences indicate which actors to film over each time period. The animation trace is broken down into scenes and the relevant idiom applied to each scene. Suitable idioms must maintain smooth camera movements and not cross the line-of-interest. Fragments that are too short or in which the camera pans backwards will be discarded. Although idioms are a suitable way of encoding principles of cinematography this particular approach is unsuitable for 3D computer games since it is not real-time.

Ting-Chieh et al. [TZY04] use a similar approach for *Cinematographic Camera Control for Computer Games.* They employ various principles of cinematography such as the action axis and establishing shots for shooting action in their real-time computer game system. The *real-time application* informs the *cinematic camera control system* about actor's positions, orientation, equipment and activities and about user input. Cinematographic camera control methods are encoded in *camera modules* and *descriptions of shots*. Descriptions of shots, which would be equivalent to Christianson's idioms [CAH*96], are arranged in a hierarchy with more general descriptions closer to the root. For example, one of the first-level nodes in the hierarchy is a shot description for a conversation. Its children nodes at the next level are *two talk* and *three talk* for two-person and three-person conversations respectively. Each shot description is modelled as a finite state machine (FSM) with each state in the FSM representing a camera module. Camera modules encode the information for filming individual shots. Specific events will trigger a transition between states in the FSM and therefore a cut to a new shot.

Low-level camera control is implemented through constraints. The available constraints are:

- *Level at:* The height of the camera relative to the subject.

- *Angle to line-of-action:* The camera's orientation relative to the line-of-action.

- *Facing:* The camera's orientation relative to the direction a character is facing.

- *Size:* The apparent size of the subject in the view. This is dependent on the actual size of the subject, its distance to the camera and the camera's field-of-view.

- *Height angle:* The angle above or below the subject to film from.

- *View at angle:* The position of the subject relative to the screen (towards the left, top etc.).

- *Visibility:* The visibility of the subject.

Discontinuity of motion is dealt with by ensuring frame coherence, i.e. the smoothness of transitions between shots. This means that rather than adhering rigidly to each constraint which may result in erratic movements, the camera configuration changes gradually producing a more pleasing view. Any jump cuts (cuts to positions or angles that are so similar to the previous ones that it looks like a mistake has occurred) will be converted to a gradual change in the camera specification such as a smooth movement to the new location. It is also ensured that the camera does not cross the line-of-action. Occlusions are prevented by testing if a cylinder drawn between the camera and subject intersects with scene geometry and if it does moving the camera back along the way it came.

Idioms are also used in *The Virtual Cinematographer* [HCS96], a camera control system for a real-time virtual party setting. The virtual cinematographer (the camera control system in this case) gains information about the state of the scene directly from the real-time application in order to choose what idioms to use. Each idiom consists of a number of camera modules, where a camera module represents a simple camera placement or movement. The virtual cinematographer may require subtle repositioning of actors before the scene is rendered. Some of the principle modules are:

- *Apex:* A shot of two actors on opposite sides of the screen with each centralised on their side of the screen.

- *Close apex:* Similar to an apex but the actors are framed in close-ups. They may be repositioned to be framed this way.

- *External:* A shot of one character over the shoulder of another.

- *Internal:* The same line of sight as the external camera module but with the camera positioned closer so that the second character is off-screen and the first character appears larger.

- *Ext1to2:* A shot of two characters over the shoulder of a third. Characters may be repositioned if necessary to achieve the required framing.

- *Track/pan/follow:* Tracking and panning shots as described in section 2. A follow shot is a combination of a pan and a track. The camera first pans to follow the subject as they pass by and then tracks to follow them from behind.

- *Fixed:* A fixed camera.

Transitions from one shot to another are again facilitated by encoding the camera modules in each idiom in a finite state machine. Specific events lead to a transition to a different state and therefore a different camera placement. The camera modules enforce adherence to the line-of-interest and detect occlusion. The idioms can change to a different shot if occlusion occurs. The Virtual Cinematographer would be unsuitable for current 3D computer game systems because the requirement that characters be repositioned for certain shots could compromise the precision of such a real-time application.

Cognitive modelling can be used to tell the camera control system about the state of the virtual world and how to film the action. Funge [Fun99] uses a *cognitive modelling language* (CML) for this purpose. The world is modelled as a sequence of situations, i.e. snapshots of the state of the world and any property of the world that may change over time is known as a *fluent*. The fluent Talking(*A*,*B*), for example, is true whenever characters *A* and *B* are having a conversation. The user may impose restrictions on what actions are carried out given the state of the scene. In the case of the fluent Talking(*A*,*B*) the camera controller can only film an external shot (over-the-shoulder shot) of character *A* if already filming *A* and it has not becoming boring yet or if not filming *A*, *A* is talking and the current shot has lasted for long enough. The user must also specify the results of actions, such as a new position in the world being taken up by the camera. The action *external*(*A*,*B*) results in the camera being directed at character *A* and being located above the shoulder of character *B*.

The required camera behaviour is encoded as complex actions written in CML by the user. A complex action is one constructed of other actions by using regular programming constructs, such as loops. A complex action can be written to direct a camera controller in shooting a conversation:

```
setCount;
while(0<silenceCount){
        pick(A,B)external(A,B);
        tick;
}
```

This directs the camera controller to pick a new external shot (over-the-shoulder shot) of either character A or B when a certain amount of time has elapsed.

One of the principal aims of cinematography is to communicate the emotional state of characters in the scene. Tomlinson et al. [TBN00] attempt to achieve this with their automatic cinematography system for interactive virtual environments. The camera control mechanism in this case is their *CameraCreature*, an artificially intelligent entity, which also positions lights in the scene. The virtual environment is populated with artificially intelligent characters, each one of which is modelled as a creature with a behaviour system. The CameraCreature can read the emotions of the characters in order to determine its own emotional state. This in turn determines the style of filming used. For example, a happy camera may cut more frequently and use oscillating movements whereas a sad camera may use long swooping movements. The camera's movements are based on a dynamical system of springs and dampers the settings of which can be altered to establish different styles of motion.

Motivations also influence filming but in a more localised way. The CameraCreature has motivations, such as, *DesireForTwoShot* which makes it look for two characters having a conversation to film. Characters in the scene also have motivations, such as, *DesireForCloseUp*. In this way they can request specific shots from the CameraCreature. Lighting is used to imply different emotional states in the scene. There is interplay between the camera, global lights and personal lights. For example, when the camera moves in for a close-up the global lights dim and the personal light of the character, which moves with them, brightens. Occlusion detection is carried out by casting a ray from the camera to the subject. In one implementation the camera moves up until the line of sight is clear. Although it would be desirable to represent the emotional state of characters in a 3D computer game by using cinematography this information is not present in current computer game systems.

Amerson and Kime [AK01] use idioms to present real-time interactive narratives in virtual worlds. Idioms are encoded in a data structure called a *scene tree*. At the root of the tree is a generic type of idiom; idioms become more specific towards the leaves of the tree. During the interactive narrative the suitable idiom is determined by performing a search on the scene tree. The idioms are encoded using Amerson and Kime's FILM (Film idiom language and model) system. Each idiom contains a number of shots where each shot represents a simple camera movement, e.g. tracking. Shots contain a list of constraints to be adhered to in order to film the shots and each constraint has a weight associated with it to define its importance to the shot.

E.g. lensType(NORMAL_LENS) WEIGHT=.9

This defines a constraint on using a normal lens to have an importance of 90%. Some of the other constraints used in the FILM system are:

- *lookAt:* A constraint on the object the camera is to film.
- *relativeLocation:* The location of the camera relative to an object.
- *maintainLensType:* To maintain the lens type used in the previous shot.
- *maintainRotation:* To maintain the orientation from the previous shot.
- *maintainAbsoluteLocation:* To maintain the camera's absolute position.

The optimal camera placement is chosen based on the constraints defined in the idiom. If a suitable placement cannot be found due to occlusion of the view or a conflict between constraints, those with weaker weights can be relaxed. An ad hoc approach to relaxing constraints is used in which predefined procedures adjust camera parameters.

Principles of cinematography such as the line-of-interest are employed in *A Camera Engine for Computer Games* [HHS01]. In this approach the balance between constraint satisfaction and frame coherence is addressed as a key issue. Different types of shots are encoded as *shot templates*. Shot templates are selected for application to the camera in real-time. Again, constraints are the basis for low level camera control. The constraints used are:

- *Level at:* The height of the camera relative to the subject.
- *Angle to line-of-interest:* The angle to film the subject at relative to the line-of-interest.
- *Facing:* The side of the subject to film.
- *Size:* The apparent size of the subject.
- *Height angle:* The angle with which the camera looks up to or down at the subject.
- *View at angle:* The position of the subject relative to the screen (towards the left, top etc.).
- *Visibility:* Simply that the subject is visible on the screen.

The balance between constraint satisfaction and frame coherence is achieved by relaxing constraints that have weaker weights when necessary.

5 Conclusion

It can be concluded that there are a range of techniques for both encapsulating cinematography in a camera system for virtual environments and acquiring data on the state of the scene. In this way suitable shots can be applied as the

events in an environment unfold. One of the main issues that must be addressed, however, is the fact that computer games are both interactive and occur in real-time. The number of principles of cinematography used will therefore be reduced and their application must be adapted for a real-time scenario.

One requirement discussed that could not be applied to a computer games is the repositioning of characters because this could compromise the accuracy of the real-time action. The ability to portray the emotions of the characters in a game would greatly enhance the visual experience but this information would have to be added to the computer game system. At most it might be possible to guess the emotional state of characters due to the events that are occurring.

Our future work will include:

- Identifying a number of key scenarios in 3D computer games and adapting principles and techniques from cinematography in order to develop an automatic camera control system suitable for filming such scenarios.

- Testing a number of approaches, including our own, to camera control through cinematography on these typical scenarios and comparing them with the knowledge of domain experts, i.e. cinematographers.

- Evaluating our implementation from the point-of-view of playability, i.e. the user's ability to carry out the task required by the game with a cinematographic camera.

References

[AK01] Amerson, D. & Kime, S. (2001). Real-Time Cinematic Camera Control for Interactive Narratives. In The Working Notes of the AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment, Stanford, CA.

[Bro02] Brown, B. (2002). Cinematography: Image Making for Cinematographers, Directors and Videographers. Oxford: Focal.

[CAH*96] Christianson, D. B., Anderson, S. E., He, L., Salesin, D. H., Weld, D. S. & Cohen, M. F. (1996). Declarative Camera Control for Automatic Cinematography. In Proceedings of the Thirteenth National Conference on Artificial Intelligence, 148-155.

[Fun99] Funge, J. (1999). Cognitive modeling for computer games. AAAI Spring Symposium on Artificial Intelligence and Computer Games, Stanford University.

[HO00] Halper, N. & Olivier, P. (2000). CAMPLAN: A Camera Planning Agent. In Smart Graphics: Papers from the 2000 AAAI Symposium, 92-100.

[HHS01] Halper, N., Helbing, R. & Strothotte, T. (2001). A Camera Engine for Computer Games: Managing the Trade-Off Between Constraint Satisfaction and Frame Coherence. In Proceedings of Eurographics, 174–183.

[HCS96] He, L., Cohen, M. F. & Salesin, D. H. (1996). The Virtual Cinematographer: A Paradigm for Real-time Camera Control and Directing. In Proceedings of SIGGRAPH 1996.

[MC00] Marchand, E. & Courty, N. (2000). Image-Based Virtual Camera Motion Strategies. In Proceedings of the Graphics Interface Conference.

[Mas65] Mascelli, J. V. (1965). The Five C's of Cinematography. Los Angeles: Silman-James Press.

[NO03] Nieuwenhuisen, D. & Overmars, M. H. (2003). Motion Planning for Camera Movements in Virtual Environments.

[TZY04] Ting-Chieh, L., Zen-Chung, S. & Yu-Ting, T. (2004). Cinematic Camera Control in 3D Computer Games. In Proceedings WSCG 2004.

[TBN00] Tomlinson, B., Blumberg, B. & Nain, D. (2000). Expressive Autonomous Cinematography for Interactive Virtual Environments. In Proceedings of the Fourth International Conference on Autonomous Agents, 317-324.