

Autonomous Real-Time Camera Agents in Interactive Narratives and Games

Diplomarbeit im Studiengang Informatik von

Alexander Hornung

Erstgutachter:

Prof. G. Lakemeyer, Ph.D.

Lehr- und Forschungsgebiet Informatik V

Rheinisch-Westfälische Technische Hochschule Aachen

Zweitgutachter:

Prof. G. Trogemann

Laboratory for Mixed Realities

Kunsthochschule für Medien Köln

6th March 2003

Contents

1	Introduction	1
1.1	Abstract	1
1.2	Thesis Structure	1
1.3	Motivation	2
1.3.1	Artificial Intelligence in Games and Narratives	2
1.3.2	Camera Control in the Context of Storytelling	3
1.3.3	Target Applications	6
1.4	Related Work	7
1.4.1	Narratives	7
1.4.2	Camera Control	7
1.4.3	AI and Agents	8
2	Cinematographic Concepts	9
2.1	A Single Shot	10
2.1.1	Shot Characterisations	11
2.1.2	Basic Means of Expression	21
2.1.3	Scenery Variants	23
2.2	Shot Sequences	24
2.2.1	Continuity Style	24
2.2.2	Shot Connections	25
2.2.3	Spatial Connections	26
2.2.4	Temporal and Logical Connections	31
2.2.5	Visual Rhythm	32
2.3	Discussion	33
3	Agents	35
3.1	General Introduction	35
3.2	Definition of Agents	36
3.3	Environmental Properties	37
3.4	Agent-Based vs. Object-Oriented	40
3.5	Symbolic vs. Non-Symbolic	42

3.6	Agent Architectures	42
4	The Camera Agent	45
4.1	General System Structure	46
4.2	Communication	47
4.2.1	Narrative Events	47
4.2.2	Camera Position and Orientation	59
4.3	Neuron-based Decision-Making	60
4.3.1	A Single Neuron	61
4.3.2	Neurons Classifying Actions	63
4.4	Architecture	66
4.4.1	Knowledge Subsystem	68
4.4.2	Decision-Making Subsystem	68
4.4.3	Action-Realisation Subsystem	68
4.5	Knowledge Subsystem	69
4.5.1	Narrative Event Analyser	70
4.5.2	Camera Behaviours	72
4.5.3	Update	78
4.6	Decision-Making Subsystem	78
4.6.1	Update	78
4.7	Action-Realisation Subsystem	79
4.7.1	Update	79
5	The Integration into Half-Life	81
5.1	Motivation	81
5.2	Sending Narrative Events	83
5.3	Setting the Camera	86
5.4	Shot Library	86
5.5	Discussion	88
6	Discussion	91
6.1	Critical Assessment	91
6.2	Future Work	92
6.3	Summary	93
A	Geometrical Issues	95
A.1	Camera Distance	95
A.2	On-Screen Position	98
A.3	Spherical Coordinates	101
A.4	Bounding-Spheres	102

B Half-Life Screenshots	105
C Example Configuration	113

Chapter 1

Introduction

“If I could tell the story in words, I wouldn’t need to lug around a camera.”

– Lewis Hine

1.1 Abstract

This work is about pictures telling, changing, and destroying stories. Visual storytelling, be it in films or interactive forms of narratives like computer-games, is a complex and very sensitive field. Images and their composition play an important role in our perception and interpretation of a visually presented story. While classical cinematography has a long history, and a lot of theoretical research has already been done to understand the correlation between a story, its film version, and the visual interpretation by the audience, research in the field of interactive narratives and the application of cinematographic concepts to this field is still rather young. In this thesis, we will investigate cinematographic concepts applicable to this field, with a strong focus on dramaturgical means of expression of cameras and the visual emphasis of narrative content. We will then present a model for a real-time camera agent implementing this knowledge. The prototype of this virtual camera system will be integrated into an interactive narrative environment to give an example for the practicality of the system.

1.2 Thesis Structure

Chapter 1 presents the motivation for this work, describes possible target applications and environments, and puts the work into the context of related research projects.

Chapter 2 is an introduction to cinematography. This chapter presents our analysis of cinematographic concepts in order to find a formalisation of the basic means of expression of cameras.

Chapter 3 introduces the theory of agents and motivates our choice to use agent-based software development for the structure and implementation of our work.

Chapter 4 describes the camera agent and its components as it was developed during this thesis. Concepts found in the preceding chapters are applied to create a communication interface and the prototype of a camera agent implementing cinematographic knowledge.

Chapter 5 gives a short introduction to the integration of the camera agent into the computer-game Half-Life[46].

Chapter 6 discusses the results of this work and presents future work and fields of research.

Appendix A present some of the geometrical mathematics and algorithms used in this thesis.

Appendix B extends chapter 5 and shows some images of the camera agent, taken in the game Half-Life.

Appendix C gives a few examples of shots from our shot library used for Half-Life.

1.3 Motivation

This section introduces the motivation behind our work. Possible benefits of a camera agent in games and other forms of narratives are discussed, and a few examples of target applications for our system are described.

1.3.1 Artificial Intelligence in Games and Narratives

Methods of Artificial Intelligence gain increasing attention in the field of computer games and interactive narratives. While the progress of techniques in computer graphics, be it photorealistic rendering [1] or stylised methods [44], creates more and more immersive environments, unconvincing behaviour of artificial, computer-controlled characters can counteract this immersion into the narrative.

The current goal of AI in games is to produce *believable* virtual characters or objects, which create the illusion of acting intelligently with respect to the underlying story. So far, characteristic methods used to model the behaviour of autonomous objects in virtual worlds were commonly restricted to predefined finite-state machines, production systems and all kinds of pathfinding [36]. But today, as more and more computational power is available and the demand for convincing AI rises, these techniques often are not expressive enough. Thus advanced theoretical models for symbolic planning and machine learning are being investigated. One of the main problems still lies in the complexity of these models, which often consume too much CPU-power for real-time applications or are difficult to design and verify like learning methods based on neural networks. But there exist successful examples of integrating complex AI to enhance game-play or the experience of a virtual story [11], [37].

The more complex the abilities of computer-controlled elements in interactive narratives get, the more difficult it becomes to create and verify these behaviours. It is important to note that such elements like autonomous characters in a game do not only have to (inter-)act intelligently, but that it is necessary for them to consider the narrative context to preserve the intended dramaturgy or even the progress of the overall story.

1.3.2 Camera Control in the Context of Storytelling

This work addresses a part of this problem: the visual interface between the narrative and the player or spectator. The visual interface can be thought of as a virtual camera filming the ongoing narrative and presenting it to the audience. So far, most of the existing camera systems found in games and narratives fail to convey or emphasise dramaturgy or emotions like mood or tension of a situation. The main goal of the visualisation process is the objective presentation of the scenery. This has important reasons, since the less influence the player of a game has over the camera and the more the camera is focused on creating a movie-like visualisation, the more difficult it is to ensure good and intuitive game-play. But on the other hand, this approach also narrows the possibilities for active visual emphasis of the story. Elaborate cinematographic techniques like subtle changes in camera movement, suppression or explicit presentation of hidden visual information like approaching dangers, intensification of feelings like fear, happiness or other emotionally affected moments by choosing appropriate camera shots are widely ignored. But obviously these techniques constitute a very important part of visual storytelling.

In spite of that, existing automatic cameras in games or narratives in

virtual environments are implemented using rather simple methods, that can be roughly classified as follows: a commonly used concept to visualise first-person games having the player taking over control of a virtual character is to show the world from virtual creature's eyes, so that the player has complete control over the view by delegating the creature through the environment. Clearly there is no deeper underlying concept necessary for this kind of camera, and it is especially not possible to convey dramaturgic information because the player has complete control over the camera (which actually is an important immersive concept, but not in the context of this work). Examples of these kinds of systems can be found in [22] or [46] and other first-person shooters. A more elaborate control is found in third-person views having a camera following the player at some distance. This concept bears more possibilities to influence the visual perception of a situation. But common systems like [10] still do not implement more than relatively simple follow-up cameras attached to the main characters body. Such cameras are moving on damped or predefined paths around the player, implementing collision detection and some simple visibility rules at most with the focus on geometric constraint satisfaction. A third type of more sophisticated camera control can be found in games using scripted, determined camera behaviours or fixed camera positions like adventure or sport games. Different views are predefined by the author and cannot be changed by the game or the player. Thus one cannot talk of a real-time autonomous camera control system. Another method of filling the gap between cinematography and games is the use of non-interactive cut scenes between the interactive parts of the game to advance the story and to present narrative content to the viewer. The state-of-the-art combination of these techniques has been realised in [30].

Nevertheless all these concepts found in current games lack the idea of an autonomous camera agent trying to visually communicate complex intentions and dramaturgic aspects of narrative situations to the audience based on real-time decisions. Elaborate cinematographic methods can only be found in games, where the authors explicitly specified the camera behaviour for each important game state.

In academic research this problem is investigated and approached in multiple ways. The main unsolved problem still is to find formalisations and models for cinematography. Though there are certain rules and standard concepts to film dialogues, actions, or emotionally affected situations, other interesting concepts emerge from breaking these established rules of cinematography, and it is doubtful that one can find a complete theory of camera control. As for art in common, there is no universal concept to create a good film or story.

Despite the fact that we are dealing with highly subjective concepts, cin-

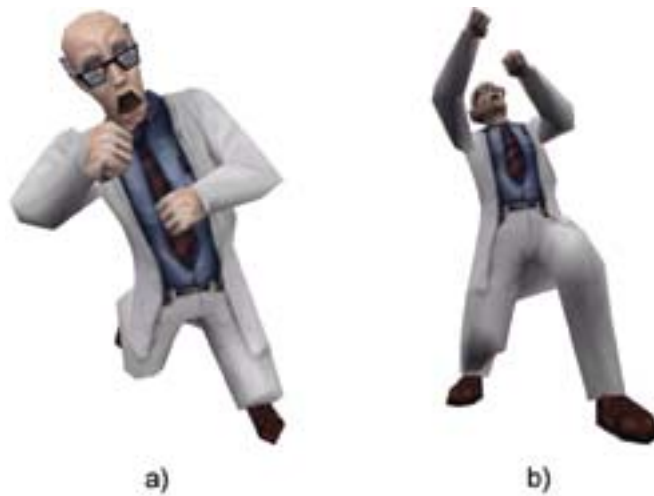


Figure 1.1: An example of two shots of an identical scene conveying completely different intentions. In image a) the character seems anxious. Image b) creates the impression of an evil character raising both fists into the air.

ematographic theory and experience shows, that there are methods for conveying intentions and psychological effects using images. Different research groups are dealing with this topic, but the diversity of approaches also shows that this problem is still not solved. The larger part of research is dedicated to the creation of systems that implement rules for ‘objective’ visualisation of 3D-environments by solving occlusion and visibility problems.

We tried to set our focus on the integration of the camera into the storytelling itself, solving questions like:

- What are the basic means of expression of cameras?
- How does this influence the experience and interpretation of a narrative?

One simple example showing the complexity and influence of the visual presentation of a scene is shown in Figure 1.1. Here we see two images of a character from the computer-game Half-Life [46]. This character is one of the scientist inhabiting the game world. We actually chose a 3D-model of the scientist trembling in fear because of an approaching danger. The left image shows the scientist from an elevated point of view, the right picture shows the same model from a lower point of view. Although the model used in both images is identical, the left picture creates the impression of

an anxious scientist, while the right picture is easily interpreted as an evil scientist, raising both fists into the air, screaming in anger. This shows that two pictures of the same character can create complementary impressions in a spectator. Thus it is necessary to consider these effects in order to create *intended* visual impressions.

An additional aspect of virtual cameras worth mentioning, which creates even more freedom and new possibilities for story-telling, is the fact that virtual cameras do not have physical constraints like mass or size and can potentially make any type of movement. Modern computer generated special effects in films like “The Matrix” [48] or “Fight Club” [13] show to some degree the fascinating visual effects emerging from this unconstrained freedom in camera handling.

A virtual camera system considering the dramaturgic and narrative importance of cinematographic concepts will be able to create a new kind of immersion and visual intensity for interactive narrative applications.

1.3.3 Target Applications

Possible target applications for a camera system following cinematographic concepts could be:

Computer-Games Computer-games are currently the most common form of interactive narrative applications. If one thinks of adventure games for instance, the player takes over the role of some character within a virtual game world and has to solve quests. Current games of this type are based on complex 3D-environments where the player can move freely and participate in a story. As we have already seen in Figure 1.1, the interpretation of visually presented stories depends strongly on the subliminal impression a picture creates in the spectator.

Authoring Tools Tools like aVRed [23] provide new means for authoring complex, interactive, and non-linear stories. The authoring of a story is often accomplished via abstract representations like story-graphs. The next step in the production process would be to verify the created story using some kind of preview-tool. A autonomous camera system considering dramaturgically relevant information for such a previewer helps the author to concentrate on the story-creation itself.

Virtual Reality Environments In virtual reality chat systems for example, where information like the emotions of a chat partner should be expressed, a camera must be able to convey this information. Other

kinds of VR-applications or even e-learning systems supporting narrative content of some kind can be easily found. For example, a walk through a virtual museum, where a mighty, terrifying historic person is introduced or where one watches a historically important dialogue are only a few fields for possible applications.

1.4 Related Work

1.4.1 Narratives

A huge number of attempts to find formalisations for literature and narratives exist, but this field has always been rather diverging lacking comprehensive theories. Beginning with ‘The Poetics of Aristotle’ [2], or ‘The Dialogues of Plato’ [35] for example, there exists a variety of research from different fields like literature or computer science. Some recent work on interactive narratives can be found for example on the AAAI 1999 Fall Symposium on Narrative Intelligence [12] dealing with models for narratives [25], agent-based views [31], logical consistency [39], structure and design [40], or interactive, nonlinear storytelling [23], [42].

1.4.2 Camera Control

Commonly based on [3] and [29] there are a number of publications dealing with models or implementations of virtual cameras. In [4], a system based on constraint satisfaction is described, [21] creates a camera working with film idioms implemented as hierarchically organised finite state machines. Another approach for declarative camera control is presented in [8]. [9] presents a method of encapsulating camera tasks into well-defined ‘camera modules’, [19] deals with a camera system designed for games, focusing on predictive camera planning and frame coherence. A camera-planning agent based on genetic algorithms is presented in [18]. [6] describes a system for guided exploration using potential fields.

Most of these works deal with established, and more or less clearly defined rules for camera control, but often ignore hard to define dramaturgical aspects of narratives based on the given story, situation, or emotional states of characters. Instead they focus mainly on geometrical constraint-satisfaction. [45] represents a behaviour-based autonomous cinematographic system dealing with emotional content of digital scenes, where the camera can change and influence the environment to achieve desired visual results.

The mathematical aspects of camera movement and positioning in 3D

environments are well understood and found in most books on 3D computer graphics like [1], [15]. Some important topics of camera control used in this work can be found in [7].

1.4.3 AI and Agents

Multi-agent systems and architectures are a field of research in Artificial Intelligence. These models play an important role in the design, formulation and verification of rational agents and their process of decision-making. We will give an introduction to the ideas of agent-based models and software design used in this thesis in Chapter 3. An introduction to agent-models and related research is given in [49]. A broader introduction to Artificial Intelligence and techniques like neural networks is given in [38].

[37] present a model for virtual actors as a part of the OZ project, another architecture in this context that supports action, emotion, and social behaviour is [5]. [16] introduces CML, a cognitive modelling language, and presents a partial solution based on CML for the virtual cinematographer as found in [21]. Some important general requirements that have to be addressed by architectures for believable (social) agents are examined in [43]. Recently, the gaming industry has begun to explore agent architectures. For example, the virtual creatures in the computer game 'Black and White' [11], [26] are based on ideas from agent-theory. Perceptrons are used for the expression of desires, and decision tree learning for the adaption of new behaviours. In this game, autonomous creatures inhabiting the game-world can be trained with a technique similar to classic conditioning.

Chapter 2

Cinematographic Concepts

“A film is - or should be - more like music than like fiction. It should be a progression of moods and feelings. The theme, what’s behind the emotion, the meaning, all that comes later.”

– Stanley Kubrick

In this chapter we analyse the narrative potential of cameras for storytelling. The characterisation of cinematographic concepts presented here will have a strong focus on constraints emerging from real-time decision-making (in some sense comparable to live tv-shows) as we find them in the context of games and interactive narrative applications. For example, the final editing of the filmed material such as the choice of transitions is very important for the overall visual impression and even the meaning of single shots (see example in the following section). But since our agent has only little knowledge about future events in strongly interactive environments and thus has to decide on-line about shots, cuts, and their consequences, the possibilities are of course limited. Ideas, which cannot be transferred from classical cinematography to our domain, will not be mentioned here. A complete and consistent analysis of the art of cinematography is unlikely to be found. Many good films are considered good because they break with common rules [13]. Creativity and intuition of a cinematographic genius are difficult to formalise, thus our goal is to find basic formalisms for camera placement and the expression of narrative intentions. Despite the difficulties arising from the artistic background of cinematography, there is still room for interesting on-the-fly storytelling by an autonomous camera agent.

This section will show, that the camera is *not* a means of objective reproduction of the real world but of identification of the audience with characters,

visual manipulation, and storytelling.

2.1 A Single Shot

‘Shots’ are the atomic entities of a movie. A shot, be it static or dynamic, represents a specific camera configuration of a certain length, that is not broken by cuts. It creates a continuous visualisation of a situation characterised for instance by position, orientation, and movement (if any) of the camera and the participating objects of a scene. Thus film semiotics often compares shots with words or sentences of a language.

A sequence of shots establishes relationships between the different shots. Single shots are put into a wider context and convey new meanings that can even change the interpretation an isolated shot into the opposite¹. Thus interdependencies between shots contribute a lot to the impression the audience gets of a single shot within a sequence. Preceding shots create a kind of interpretative threshold that has to be overcome by following shots, if they want to change the visual impression to a different meaning. This is part of the montage of a movie and will be discussed later.

Cinematographic terminology often uses characters as reference objects for shot descriptions, like close-up shots of faces. But since most of the following classifications can be applied to general objects², we will talk of *objects* and their respective *point-of-interest* (PoI). This will become more important in later sections when we introduce the communication and the shot models of our camera agent.

Katz [29] presents four general categories where graphic qualities of a single shot are compared to narrative qualities. These categories can be used as a first attempt to find a definition of the basic means of expression of cameras. The inherent idea is that the visual interpretation of a picture by the audience is based on a process of identification of the spectator with the camera standpoint and view. The four categories introduced by Katz are:

¹L. Kuleschow presented the same neutral face after an image of a bowl of soup and a dead man. The audience interpreted the face in the first case as having a hungry expression, in the second case as being sad.

²With restrictions of course, if it comes to the visualisation of emotional states for instance.

	Graphic	Narrative
1)	Where is the camera positioned?	Whose point of view is being expressed?
2)	What is the size of the shot?	What distance are we from the subject of the scene?
3)	What is our angle of view?	What is our relationship to the subject?
4)	Are we cutting or moving the camera?	Are we comparing points of view?

By analysing the different means of expression of cameras one finds that these categories do not suffice to cover the narrative possibilities of cameras. However they are a good starting point for the following discussion and we will go deeper into these four questions by describing in detail aspects of shots and their narrative meaning.

2.1.1 Shot Characterisations

Shots can be classified and characterised in many different ways. Classifications found in literature are partly diverging or prefer different names for shots. In the following we present the most common types of shots, starting with *field sizes* and their applications (see Figure 2.1).

Long Shot The long shot establishes a scene or situation by providing an overview and allows the audience to put the scene into a context of participating objects and the location. It helps to create an personal representation of the situation and simplifies orientation for later shots. On the other hand, details of a scene are missed. Compared to closer shots, the beholder needs more time to understand the presented shot because of a higher degree of contained information. Long shots of wide landscapes together with slow camera movements for instance can be used to emphasise lyrical moods or epic scenes. In some cases *extreme long shots* as the superlative of long shots are used.

Medium Long Shot In contrast to a long shot which plays the role of a distant beholder (also from an emotional point of view), the medium long shot shows characters or objects and their direct environment in more detail. The audience can immediately find the intended focus of the scene without being visually ‘lost’ like in the above mentioned shot.

Full Shot Using this shot size, a filmed object extends over the complete field-of-view of the camera. Depending on the type of the target object,

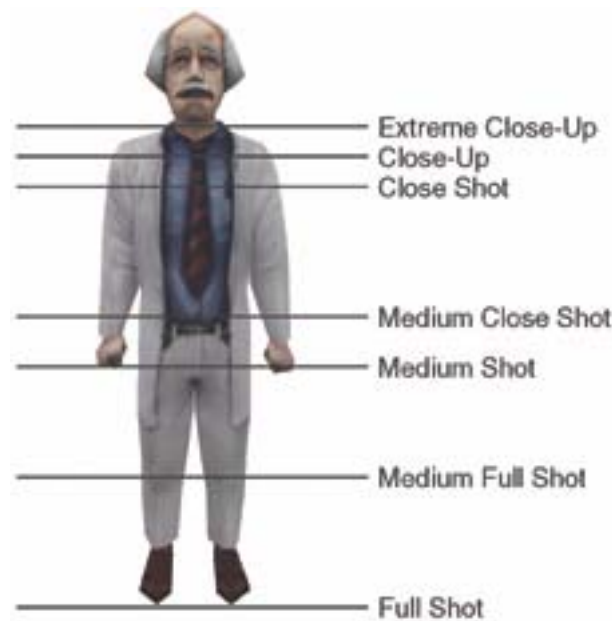


Figure 2.1: Reference framing heights for the human body.

the environment is still partly visible and allows to put the object into a context. This shot can be used to show a talking character using body language for example.

Plan Américain or Medium Full Shot The medium full shot shows objects with about two thirds of their overall size. Characters for example are shown from their knees upwards. This kind of shot was extensively used in old western movies, where the cowboy's colts played an important role and had to be emphasised.

Medium Shot This shot size is similar to the previous one, showing half of the object's size. Characters are shown from their waist upwards. Terminology in cinematographic literature concerning this shot and the previous one is not always clear and consistent. However we will stick to these naming conventions.

Medium Close Shot The camera shows one third of the objects size, characters are framed from the chest upwards. The camera loses its neutral position and creates a more subjective, emotional view than the previous shots. The beholder can identify himself with an additional person, standing at the position of the camera and taking part in the action.



Figure 2.2: Close-Up of Charlie Chaplin in the movie ‘City Lights’, emphasising his facial expression and emotions.

Close Shot This size is similar to the above ones lying in between medium close shot and close-up shots.

Close-Up A close-up shows a small part of the overall object. For characters this would be a shot of the head or face, which covers the whole field-of-view. In general we will speak of a shot focusing at the point-of-interest of an object. It visualises the main motive of the situation. Details, which would otherwise not be visible, are presented in a denser form. Close-ups represent climaxes of the action, and characterise and clarify the filmed object (see Figure 2.2).

Extreme Close-Up As an even stronger and more detailed form of the previous shot type, an extreme close-up emphasises the emotional content of the situation. Even the smallest detail or information is revealed and thus this shot type plays a role of ‘unmasking’.

The different shot sizes are strongly related to each other with regard to visual continuity between shots or scenes. Used together in a ‘correct’ manner they help to create a consistent spatial and temporal order. On the other hand they can of course be used to destroy this order on purpose. The problem of ‘correct usage of shots’ is difficult to define and applies to all of the following characterisations as well. This however is part of the editing and combination of shot-sequences which we will deal with in a later section.



Figure 2.3: Low angle shot of the vampire Nosferatu emphasising his evil, terrifying nature.

A further characterisation of shots is given by the *camera position* and the *camera angle*. The position of the camera is the position of the audience. Thus it plays an important psychological role in conveying subliminal information or intentions. The audience gets subjectively involved or experiences a situation from a more neutral distance. Relationships between characters and their character traits can be emphasised using low or high shots of a person. Different camera angles are one means of conveying perspective to the audience. They can even be used to create visual exaggeration or confusion. We will now give a more detailed description of possibilities given by specific camera positions and angles and their meaning from a narrative standpoint.

Eye Level / Normal Angle The camera is positioned at eye-height, or, more generally, the height of the point-of-interest. This angle represents a neutral, normal view for the audience and carries no subjective information or judging. The spectator is placed at an equal level with the filmed character or object.

Overhead / High Angle In this position the camera floats slightly above the filmed object and creates the impression of isolation and humiliation. This view puts the audience into an elevated, powerful position. It is often used in combination with long shots to facilitate orientation in large, complex scenes.

Bird's-Eye View This view emphasises aspects of the overhead shot and intensifies the impression of distance, the power of the audience, and loneliness of focused objects.

Below Shot / Low Angle A shot from a low angle creates the opposite impression of a high angle shot. It displays might and power of the filmed object in contrast to the audience or other participating objects. Further attributes are heroic and self-conscious, arrogant and dictatorial, or even demonic and terrifying (Figure 2.3).

Worms's-Eye View As above this view is the exaggerated form of a low angle shot and emphasises it's characteristics.

Oblique / Canted Angle A canted view often creates an unreal or uncomfortable impression. It emphasises the abstract or surreal aspects of a situation.

How much an inappropriate camera position can counteract the desired visual impression can be seen in Figure 2.4. While the left shot from a low angle emphasises the desired evil appearance of the scientist by putting the spectator in a lower position, the right shot of the same model from an high angle creates a discrepancy between the meaning of the scene and the perceived impression (see also Figure 1.1).

So far, we have described characteristic shot properties influenced by position and distance of the camera to filmed objects. Apart from still shots, camera movements are a third class of important means for cinematographic expression. On the one hand, they can simplify orientation and perspective view of the audience by leading through the scene and environment. On the other hand, specific camera movements can also intensify dramaturgic elements of a scene by confusing the audience for example. The primary parameters influencing the narrative meaning of camera movements in combination with the above mentioned attributes are motion speed and direction. Due to the fact that a 'real-life' camera is a physical, often heavy object, it is bound by the laws of physics. This implies a huge number of restrictions concerning the possibilities for camera movements and is the reason why common terminology is oriented at *how* a camera is moved in real-life cinematography. Examples are terms like *crane shots* or *dolly shots*. We will therefore use these terms only during the introduction of different shot classifications, because none of these physical constraints apply to our camera agent. It is part of a virtual reality, where gravity, mass, and other physical problems can be ignored. The absence of physical restrictions clearly has a

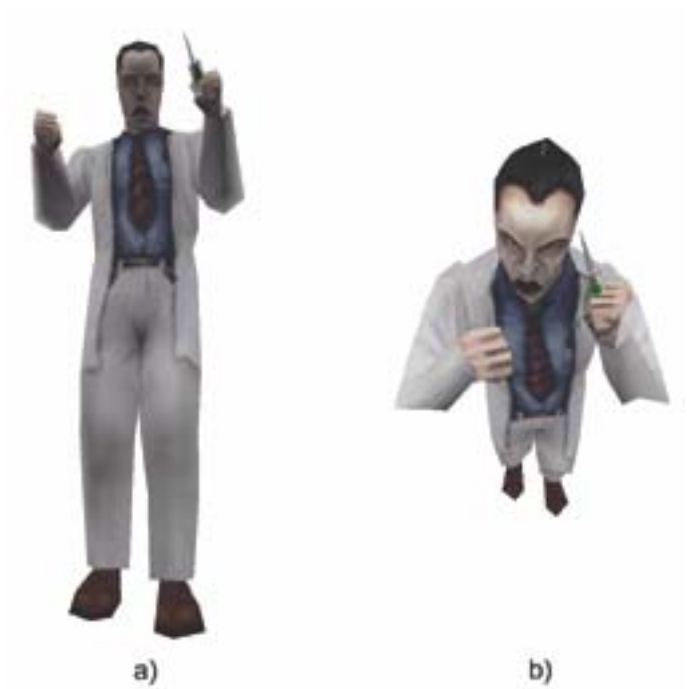


Figure 2.4: Comparison of two shots from different angles of the same model. a) shows a low angle shot creating the intended evil impression. In b), the spectator is irritated by the high angle perspective.

lot of advantages, but it also means that some effects of real cameras in specific shots like *hand-held camera* shots and the visual impression they create have to be approximated by simulation. This can be quite a complex task if one considers hectic, anxious movements of a fleeing person for instance. We will discuss this problem in the section on the implementation of camera behaviours.

In the following we present common ways of camera movements from a cinematographic point of view and their applications.

Pan Tilting the camera in horizontal or vertical direction can have the following functions:

- Create / simplify overview
- Lead the view of the audience
- Follow moving objects
- Create a transition between two shots instead of cuts
- Generate a kind of underlying visual rhythm

Each of these different effects (this holds in fact for all presented shot types) requires a strong embedding of camera shots into the dramaturgic flow of the narrative. The different types of pans and their effects are mostly characterised by speed:

Slow Pan A slow pan creates a calm mood by taking away the audience's feeling for the progress of time. The narrative flow is delayed. This kind of pan can be used to intensify the impression of a beautiful landscape for example.

Follow-Up Pan This type has the function of searching, selecting or analysing objects. It follows moving objects with the goal of keeping them in the field-of-view and strongly depends on the type and speed of the object being tracked.

Fast Pan A fast pan creates moments of surprise, reveals sudden (re-)actions of characters, confronts enemies, emphasises sudden turning-points of the story, or intensifies the effect of dramatic dialogues or actions.

Teared Pan Sudden, abrupt changes prevent the perception of details.

A combination of different pan types producing visual rhythm can lead to comforting or disconcerting feelings.

Camera Moves A shot with a moving camera can have many functions. If it is of moderate speed, orientation is for example easier than with still shots, since the audience can get a better impression of perspective, depth, and the spatial relationships between objects. It feels like personally moving through the scene. On the other hand, camera movement can have a strong dramaturgic meaning, e.g. if the camera moves backwards or forward or at extremely high or low speeds. Approaching camera movements can introduce new objects at the beginning of scenes, whereas backward moves distance the audience from the object, also in an emotional sense. Fast moves can confuse the audience. Theoretically, moves can be of any type, on straight or curved paths with changing focus. As said before some of the following cinematographic terminology is closely related to real world constraints and realisations of shots.

Slow Move Similar to slow pans, lyrical moods can be intensified by this type of move.

Circumnavigation If the camera circles around an object (a combination of panning and moving), back- and foreground are strongly moved against each other. This creates the illusion of space and depth in the picture. Filmed objects are shown from almost every direction providing no possibility to hide something from the audience. One gets the impression of lurking around the object.

Crane Shot It follows objects or leads the audience through the scene. Irritating effects can be sudden upwards movements for instance.

Free Cameras This term stands for cameras that can move freely through the scene. Examples are cameras fixed at the hand or head of a character, at the bottom of a car, etc.. Generally speaking, the camera is fixed or mounted to a part of an object participating in the scene and thus it creates a very subjective point of view.

Criteria for movements found in literature are often diverging. But obviously such movements are justified as long as the camera movement is dramaturgically motivated and emphasises the intended impression of a situation.

The next important aspect of shots is the composition of objects in the scene and on the screen. In cinematographic literature image composition is known as the most difficult to formalise and ‘mysterious’ aspect of creating films. All the previous means of a camera join together here and have to

be integrated. Some catchwords are arrangement, perspective, proportions, colours, focus, etc.. A bad composition of a picture cannot be undone during the montage of a film.

Since our agent has to deal with given scenes in narratives and games in real-time rather than interactive scene composition, and because it has no influence on the position of characters for example, we will concentrate on the second point, namely the different ways of positioning objects on the screen. Many concepts of positioning characters during dialogues like so called A- or L-patterns exist to facilitate a good understanding of the spacial relationships or the facial expressions of characters for instance. In the context of intelligent cameras without the possibility to change their environment, this problem should be an important part of future research. The goal should be to detect such patterns and chose appropriate visualisations rather than reposition objects or characters in a scene.

We already have talked about shot sizes and possible interpretations. The position of objects on the screen can imply a variety of further meanings. If for instance the dominant part of objects is positioned at the border of the picture it seems more dynamical. In contrast to this, picture-centered objects give a static impression. People or objects positioned at the lower border of the screen with a lot of free space between them and the upper border seem to be relatively small in comparison with the rest of the image. In the opposite case, objects touching the upper border create the impression of being rather large. There are numerous aspects of composition based for instance on metaphors like an archway framing a character symbolising safety. These however would require a much more complex world representation in our agent and other knowledge which is beyond the scope and the goals of this work. Thus we will only consider means of expressions suited for our context. The different aspects of composition and their relevant interpretation are:

Static Composition A static composition stands for equilibrium, stability, calmness or safety. Frontally, horizontally oriented shots with straight lines and less perspective dominance stand for a static composition.

Dynamic Composition This type of composition creates a more unstable impression. Increased perspective, irregular lines and proportions increase dynamics in a picture. A frontal view of a car for example seems more static than in a diagonal shot.

High Object Position The relative position of objects on the screen gives further important clues for the visual balance. If objects are arranged in the upper half of the screen, tension and excitement are created.

Low Object Position Objects arranged along the lower border of the frame convey gravity, seriousness, contentedness.

Background Relation Showing the sky as the primary background element implies feelings of (mental) freedom and vastness. In contrast, the earth framing our objects stands for intimacy, closeness and earthbound characters. An equal relation between earth and sky in the background creates and intensifies conflicts. As described above, a horizontal borderline between sky and earth conveys very static impression whereas canted shots of this borderline emphasise dynamic aspects of the scene.

The symmetry of a picture gives further clues and possible interpretations, but depends very much on scene composition. This makes it very difficult to find approaches for our agent to deal with this topic only by screen composition. The same problem arises for geometrical patterns generated by objects. Normally, one wants to break these visual patterns, since they over-emphasise possibly unimportant aspects of the picture and thus confuse or bore the audience unnecessarily. Another important criterium for picture composition is the type of depicted object and the question whether it is ‘photogenic’ at all (or which side of it). Light plays a very important role during shot creation and composition for the visual interpretation. In movies, light is used for a variety of tasks and goals. It can intensify the demonic appearance of a character for example, or emphasise certain parts of an object. This is usually part of the scene composition, since one needs to plan the position of lights before shots are taken to optimally use light as a means of expression. Our camera agent has to visualise a given situation without rearranging its elements and thus use the current lighting conditions for shot creation. This requires a deeper analysis of means of expression of lighting but due to the arising complexity of the overall problem these points should also be addressed as part of future research.

The length of shots and the realisation of cuts is the last point in this section. Fast sequences of short shots with hard cuts increase the subjective speed of the progress of time, the scene seems to be hectic. Long shots with less cuts convey the opposite, the scene seems calm and the flow of time is stretched. The same impression is intensified by slow fade-outs instead of cuts. Such fades additionally create the feeling of melancholy, farewell and transitoriness. Fades generally stretch the perceived flow of time.

These characterisations of different shot types represent material from theoretical and applied literature [3], [29], [28], [32] on cinematographic concepts. Of course cinematography is not restricted to the usage of these shot types, and the breaking of these rules, the combination and variation of shots

can reveal interesting new dramaturgic means. However this characterisation is a summary of basic means of expression of single shots relevant in the context of our agent.

2.1.2 Basic Means of Expression

Having now an overview of possible subliminal meanings of different shot-types and the basic expressiveness of cameras we will try to extract a more formalised version of this information, that can serve as a basis for communication between a narrative application and the camera.

The following descriptions were extracted as a minimal set characterising the basic expressiveness of our cameras. However they are not complete and in some cases simplified. We combined for example the description for ‘calmness’ and ‘dynamics’ of a situation, since they are linked together in many of the cases, though not all. But to keep the complexity on a bearable level for this work we had to assume simplified models. Though some of the descriptions are partly overlapping, they still emphasise different aspects with respect to a camera and its narrative capabilities. Together with the descriptions we give a few examples of the realisation in order to provide the necessary context to our above analysis of shots.

The first characterisation is rather straightforward and follows from the type and range of action in a scene. This reflects the fact that some shots are especially suited for actions with a small radius and others for long range actions. An example of such long range action shots could be a camera movement covering a lot of space or following the direction of action from the subject to the object.

DISTANCE OF INTERACTION	
DISTANT	long shots
MEDIUM	medium shots
CLOSE	close shots

Relationships between objects, display of power and social situation can generally be represented by the might of an object:

OBJECT MIGHT

ISOLATED, HUMILIATED	high angle, long shots
NEUTRAL	eye-level angle
MIGHTY, DEMONIC	low angle, medium shots

The intensesness of a situation, general emotional involvement of the audience, or climaxes are expressed by the emotional involvement of the audience into a situation:

EMOTIONAL INVOLVEMENT

LOW, DISTANT	long shots
NEUTRAL	medium size/speed shots
HIGH, SUBJECTIVE	close shots, point-of-view, hand-held camera

Dynamics, calmness, irritation, confusion, surprises, sudden actions, and disorientation of the audience are expressed by the calmness within a scene:

CALMNESS

CALM	slow moves, long shot length
NEUTRAL	medium speed moves
HECTIC, CONFUSED	fast moves and pans, sudden upwards movements, hand-held camera, fast cuts

The excitement, tension, gravity, seriousness, or contentedness of a narrative situation is summarised in the excitement within a scene:

EXCITEMENT

UNEXCITED, SERIOUS	low object screen position
NEUTRAL	centered positions
EXCITED, TENSION	higher object screen position

These definitions will be used to classify situations and events of a story and to communicate them to the agent. They give us the possibility to represent fuzzy, weighted descriptions of a scene or action using the above classes.

2.1.3 Scenery Variants

In addition to the above characterisation of shots we need some more strict classes to distinguish shots suited for a situation from shot types that are completely unsuited. In cinematographic literature one finds several different classifications of shot types, of which we will present (and expand) the most important ones with respect to our work. We will use these classes as a set of preconditions to select suitable shots from a given library of shots.

The following three different ‘scenery variants’ from [3] have to be handled differently and imply specific shot types.

Dialogue only Cinematographic literature describes established and specific rules for the visualisation of dialogues between characters. Specific rules for camera positioning and shot types have to be applied that can be ignored for other types of actions. We will introduce the so called triangle-system in one of the following sections as an example. Further examples are the above-mentioned A- or L-patterns for scene composition.

Action only Here the focus lies on non-dialogue action. This implies a different camera handling than in dialogue scenes. More complex types of camera movements can be applied, since the audience does not have to follow a conversation, and thus can concentrate completely on the visualisation.

Dialogue & Action This scenery variant stands for a mixture of dialogues and actions. Shots for this type of scenery have to provide an overview of the flow of the dialogue as well as additionally occurring actions.

These variants are focused on scenes and shots with characters as the main objects. For sceneries setting the focus on non-character objects we will use three additional action types taken from [27], providing a more detailed distinction of actions:

Physical actions Actions involving physical interaction with the environment. Shooting a gun, driving a car or running are examples of such actions.

Mental actions These action types can be used to emphasise that a character is thinking about another character or object. For example, facial expressions of the thinking character or distorted views of the object being thought of would be possible shots to emphasise mental actions.

Predicates This type of action can be used to describe the ‘existence’ of an object, which is not described appropriately using mental or physical actions. Examples could be a house upon the top of a hill or a tree existing as part of a landscape.

Further classifications of actions are of course possible but the above-mentioned succeed in describing events of a story in enough detail to provide a good base for communication between the camera agent and the narrative application. The more detailed the description gets the better our agent can react to them but also the more complex the overall problem of defining shots gets. Considering the trade-off between quality and complexity the approaches presented so far constitute an expandable base for cinematography.

2.2 Shot Sequences

Sequences of shots play an important role in the creation of visually aesthetical cinematography and in conveying consistent visual interpretations that do not contradict the intended narrative meaning. Obviously the meaning of ‘aesthetical cinematography’ is difficult to formalise. However there are certain common rules for shot sequences. A new scene for example should generally be introduced with an establishment shot (e.g. long shot) to set the context for the following scenes. During the progress of the action however, there should be no further need for establishment shots. Instead, medium long shots or close-up shots should be used to emphasise the main motives of the current situation. Simple jumps on one axis from long shots to close-up shots should be avoided since they create uninteresting sequences. On the other hand this technique can be found to convey the impression of astonishment of a character.

The general style concerning sequences of shots found in narrative cinematography is described by the term ‘continuity style’.

2.2.1 Continuity Style

This cinematographic term stands for a photographic and editorial style creating a spatial and/or temporal continuum during a sequence of shots. It is mainly used in narrative films, also called ‘Hollywood classical style’ or

‘*découpage classique*’. The strategy of the continuity style consists mainly of two aspects, namely to provide means of visual recognition between shots and to create a relation between the shots by implication or inference. Visual recognition can for example be based on reappearing objects or other parts of the environment. An example of inference found in [29] is the following sequence: A man shown in a long shot approaches a door, the next extreme close-up shows a hand turning a doorknob, which was formerly not visible. The connection is created by the spectator’s expectation that the man wants to go through the door and therefore has to open it by turning the doorknob, even though the close-up could be from a different place and time or a different person.

As we want to create a camera agent for interactive narratives the idea of a visual continuum plays of course an important role. But on the other hand, some concepts of classical cinematography are difficult to apply to interactive storytelling. The fact that a human person is interacting in real-time with the narrative creates a new domain of problems that is not covered by classical ideas of film making and directing.

The idea of continuity has several implications especially for sequences of shot which we will discuss in the next sections.

2.2.2 Shot Connections

Under the assumption that we want to preserve a visual continuum as found in the idea of continuity style, the primary connection of a series of shots can be described by one of the following three classes as found in [29]:

Spatial Connections Shots of different types (e.g. size, position) are taken of the same scenery and thus have a spatial connection. A cut from a long shot of the White House to a recognizable detail of the same building in a closer shot would be an example of simple spatial connections. These connections have to be considered during all shots so that no (unintendedly) irritating jumps or unmotivated changes in shot size do occur. One concept helping to preserve the spatial connection is the concept of the line of action, discussed in the next section.

Temporal Connections These connections emphasise temporal interdependencies between different events occurring during scenes. An example would be a cut from a man dropping a glass in one shot to the glass braking on the floor in the next shot.

Logical Connections Logical connections stand for story-events or shots, that are implicitly connected to each other. If one cuts from a long shot

of the White House to a shot of the President seated in an office the dominating connection between these shots is causal without obvious temporal or spatial connections.

These connections of shots and events of a story are strongly connected to the question of optimal story-representation which still is an open research problem. There are different approaches ranging from graph-based representations [23] to symbolic representations [12], [25]. Especially in the context of interactive narratives, where there is no completely predefined storyboard or script and where future events are potentially unknown until the moment they occur, we have to make several restrictions and focus on aspects solvable by real-time decision-making without complex planning capabilities about the development and progress of the story.

In the following sections we will present cinematographic concepts, which can be considered under the constraints of real-time camera control and which help to create or emphasize the above mentioned connections in sequences of shots.

2.2.3 Spatial Connections

To preserve and emphasise spatial connections between shots we will present two general concepts.

Line Of Action

The ‘Line Of Action’³ is one of the most important concepts for consistent continuous camera placement. From a 3-dimensional, mathematical standpoint it has to be considered as a ‘Plane Of Action’ separating space into two distinct halfspaces. Thus we will alternatively use this term instead if appropriate.

The basic idea behind this concept is the following: every action usually has a specific direction in space. During a dialogue for example, this direction is given by the line between the involved characters (Figure 2.5). Figure 2.6 shows an example where the camera films both characters from the same side of the line of action. A consistent orientation of both characters on screen is preserved here. If the camera crosses this line during the transition between the two different shots, the relative positions of the talking characters on the screen are reversed and they suddenly are both facing in the same screen direction, as shown in Figure 2.7. This has the effect of irritating the audience and making orientation difficult. With more than two characters or objects,

³Alternatively: ‘180 degree rule’

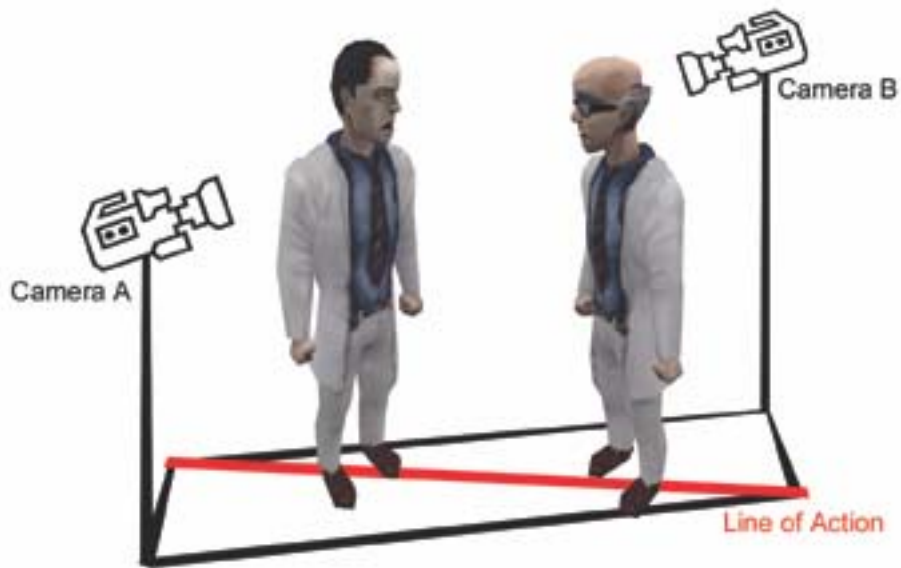


Figure 2.5: The line of action between two characters during a dialogue.



Figure 2.6: The camera stays in the halfspace of Camera A during the consecutive shots a) and b).



Figure 2.7: During the transition from a) to b) the camera jumps over the line of action into the halfspace of Camera B.

the problem of disorientation gets even worse. Thus the concept of the line of action is the most basic rule for camera placement found under the continuity assumption between shots. It preserves consistent screen direction and space.

In Literature like [29] this problem is examined using a large number of case studies to explain special cases like characters crossing the line of action, or the introduction of new characters. Additionally, the problem of consistent camera placement often only comes up during the final editing or cutting of a movie, since no temporal connections need to be preserved during the actual shooting and thus the director can freely choose the order of shots. Finding a consistent sequence of shots that do not hurt the line of action rule is then the task of the cutter of the movie. But since our camera agent is not making movies but has to respect these constraints instantly during a real-time narrative we need specific rules assuring that the principle of the line of action is not violated. Furthermore, bridge shots like extreme close-ups of objects are often used in film editing to establish a new line of action. These shots should of course be narratively motivated and not only bridges to help out with errors made during filming, but for a camera agent in real-time stories without knowledge of future events this possibility will be rarely usable.

From the informal descriptions found in literature we extracted the following rules concerning the relationship between camera placement and the line of action:

1. Generally, the camera must not cross the line during a cut between two related shots⁴. Mathematically speaking the camera has to stay in the same halfspace of the line of action as in previous shots.
2. The situation is different, if the camera crosses the line during a continuous camera movement. This way, the orientation of the audience is preserved and the currently active halfspace gets redefined by the camera position at the end of the movement.
3. If the objects are not static but moving on continuous paths the Line Of Action is defined by the direction of the objects' action (a running character, two talking persons, etc.).
4. If the line of action suddenly changes because the acting objects changed there are two different methods to proceed. In each case however, the line of action gets redefined by the acting objects.

⁴Of course such a jump over the line can be dramaturgically motivated to disorient and confuse the audience by intention

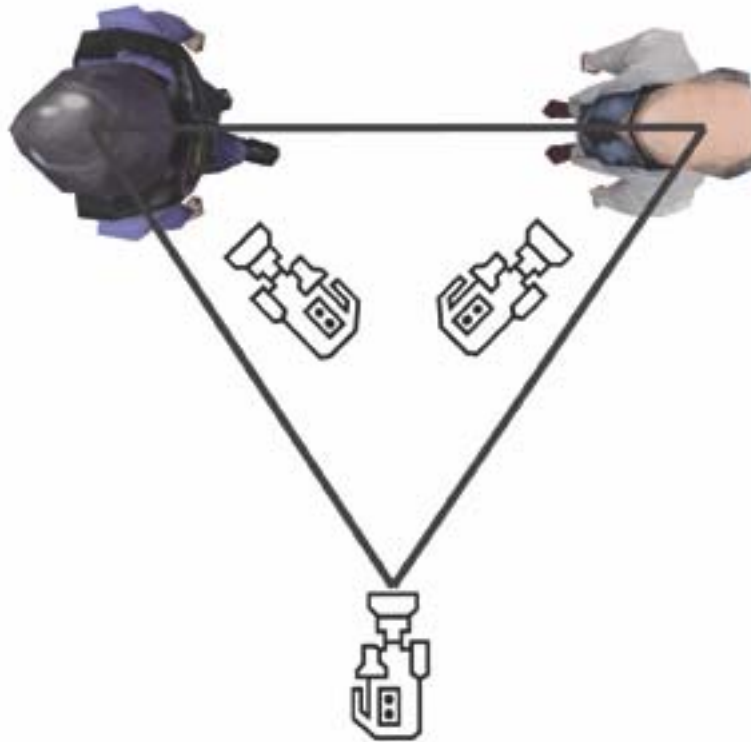


Figure 2.8: A camera setup using the triangle system for angular singles and a master two shot.

- (a) If the new action is closely related to the previous one, for instance, if there is a group of more than two characters having a conversation, the line of action gets redefined as said before and the new active halfspace is defined by the last camera position during the previous shot.
- (b) If the action or objects are not related to the previous shot (a completely new scene for instance) the new active halfspace can be chosen randomly or motivated by other criteria like visibility of the point-of-interest.

Triangle System

Like the line of action, the triangle system represents one of the basic concepts for camera placement. It can be seen as the second point to consider after the line of action has been determined. It is of special importance for the

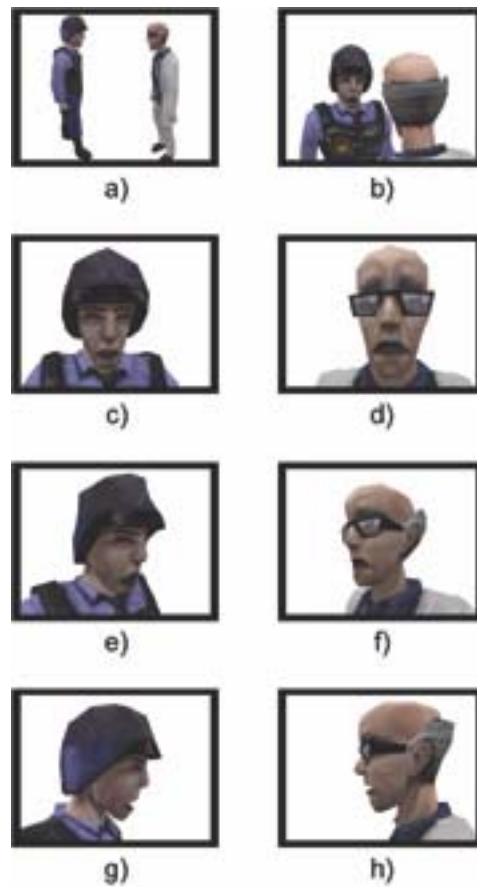


Figure 2.9: a) Master two shot. b) Over-the-shoulder shot. c)-d) Point of view shots. e)-f) Angular singles. g)-h) Profile shots.

visualisation of characters during dialogues, but can also be applied to more general kinds of actions. The triangle system proposes that all basic shots possible for any subject can be taken from three points forming a triangle within the currently chosen halfspace of the plane of action (Figure 2.8).

[3], [29] present five basic camera setups within this system. We will use characters as reference objects here, but they apply as well to other objects:

Master Two Shots Shot of two objects from an orthogonal position under consideration of the line of action. Often used during the establishment of a scene. (Figure 2.9 a))

Over-The-Shoulder Shots Camera views one character from over the shoulder of another character. (Figure 2.9 b))

Point-Of-View Singles Show one character from the position of the other character. (Figure 2.9, c)-d))

Angular Singles Medium shots or close-ups of a single character (Figure 2.9 e)-f)). Using this shot type the spectator can see the facial expressions of the acting character from a third-person position without being identified with the second character like in point-of-view singles.

Profiles Close-up shot showing a character's profile. (Figure 2.9 g)-h))

The exact angles of the different basic setups, shot sizes or sequence compositions are variable and do often follow personal aesthetical preferences or other aspects of a scene that are difficult to formalise.

Further criteria for shot-settings can be found in [3] for example. Questions mentioned there and influencing camera placement are for example: Do the actors face each other? Are they standing side by side? Are they turning their back to each other? Is the character standing, sitting, lying, etc.? While all these questions can have an influence on camera placement they cannot all be considered in this work. In addition they do not contribute in solving the question about basic means of expression which we are most concerned with.

Character placement is indeed very critical to 'optimal' camera placement. Dialogues between three or more characters are difficult to handle and often the characters are positioned in predefined patterns like already mentioned A- or L-patterns. This is also true if characters should convey a certain impression based on their appearance or position in the scene. Placing a character at a distance from a group in order to emphasise social isolation is just one example. The camera agent on the other hand does not have any influence on character placement but should be able to deal with arbitrary, fixed settings.

2.2.4 Temporal and Logical Connections

These types of connections strongly depend on the representation of a story and are difficult to deal with under real-time constraints where the story is built just-in-time as a sequence of events, for which a logical connection can only be found afterwards.

A lot of shot sequences concerning temporal or logical connections are based on 'question & answer' patterns. The goal is to set up expectations in the audience and then give solutions (or not). Revisiting the example with the man dropping his glass, the interpretation of the situation depends strongly on other pieces of information presented earlier or long after this

event. If in previous or following shots a party was presented to the audience and a man pouring poison into the glass, the short moment of dropping the glass gets a completely new meaning. Without the poison, the question and answer pattern serves only to describe the action of dropping a glass and what happens to it, namely breaking on the floor.

We cannot assume that interactive narratives are necessarily set up using question & answer patterns. It should be clear, that temporal and logical connections need a very complex representation of narrative events and also insight into future events, which cannot be predicted in interactive stories, where everything depends on the behaviour of the interactor.

We will bring up the issue of *coherence* between different story-events later during the introduction of narrative events.

Some special cutting concepts for movements, exits of persons from a scene, or the clearing of frames are also extremely difficult to deal with. This is due to the uncertainty introduced by player movement for example and one would need highly sophisticated planning abilities to predict future player or other object positions which is beyond the scope of this work. Some approaches to motion prediction can be found in [19].

2.2.5 Visual Rhythm

The visual rhythm of shot sequences is concerned with shot lengths and the change of focus points between shots. The duration of single shots within a sequence is an important means of expression. Calm scenes cannot be visualised appropriately using shot lengths of under a second. On the other hand hectic or dangerous scenes are emphasised through fast cuts.

Furthermore the eyes of the audience should not stay focused on the same screen position during a sequence. Instead objects should be positioned such that one has to adapt his view and focused point continually to prevent a tiring or even boring visual impression. This is often done by moving objects or characters to the left or right on the screen. Although we did not find a common, consistent rule for adding such an offset to the object position, interestingly in most of the examples that we studied the object was moved in the opposite direction of the current action direction, so that an open space is created in the direction of the object's movement or the view of a character.

2.3 Discussion

The cinematographic concepts we have discussed so far represent basic means of expression for cameras in a narrative context on which we will base the model and implementation of our camera agent. However these ideas cover only a subset of the expressive possibilities of cinematography. Further problems and concepts like the consideration of the lighting situation, patterns for optimal object visualisation, geometrical constraints and occlusion problems, and complex, predictive camera movement- and cut-planning have all to be considered within a complete camera system. Scene composition, visual metaphors and other psychological effects caused by the choice of colours, or film material for instance are even more difficult to formalise. In addition, there exist a lot of fundamental differences between the fields of interactive narratives and classical cinematography. We already mentioned the physical constraints of real cameras, which can be ignored in virtual realities. The transfer of existing knowledge from classical cinematography only does not satisfy the demands and possibilities of interactive storytelling. On the other hand, some important cinematographic concepts like transitions between narrative situations are difficult to transfer because of the unpredictable nature of interactive stories. Considering the interactivity and non-linearity of narratives based on virtual reality applications in contrast to linear, predefined narratives like movies, the finding of new stylistic devices has to be a further field of future research. Nevertheless the analysis presented so far provides important means to improve the visual emphasis of narrative content.

Chapter 3

Agents

“The real danger is not that computers will begin to think like men, but that men will begin to think like computers.”

– Sydney J. Harris

In the following sections we will motivate the agent-based approach that we based our camera system on.

3.1 General Introduction

There exist a variety of approaches to designing, structuring and implementing complex software systems. All these different software design techniques, be it functional, declarative, object-oriented, agent-based, etc., have been developed in order to subdivide complex problems into smaller parts, which are easier to understand and verify. Each of these approaches is suited more or less for specific tasks and has advantages as well as disadvantages for certain problem domains. Different criteria like the possibilities to model a domain or a problem, efficiency, reusability, platform- or software- independency, etc., can influence the choice of the appropriate design and language for a given problem. This work is situated in the context of highly interactive software like interactive narratives, virtual reality applications, or games. Accordingly we have strong demands on real-time capabilities for instance. But one of the main questions for the design of our camera system is the question: *“How is the camera system embedded in a narrative application?”* In fact, we want our camera to act as an autonomous system in a given environment, without the necessity to be controlled by a user for instance. Using an agent-based approach is one possible solution.

A lot of software applications are not based on direct interaction and feedback by a user, but run in a certain environment to permanently fulfill specific tasks without the necessity of supervision by humans. Examples range from unix software demons for the printer queue over the control software of space probes billions of kilometers away from earth (obviously out of range of direct human control) and bidding agents for online auctions to the software controlling the safety of nuclear power stations. These applications are designed to do some sort of decision-making and/or acting in highly dynamic, unpredictable or open environments. In particular these decisions or actions can be wrong and may result in conflicting, problematic or even dangerous situations. Such applications should be able to resolve problems without human intervention. So in general we are dealing with systems that are able to perceive their environment, make decisions based on their knowledge and state and can act upon the environment. Such computer systems are commonly called agents.

3.2 Definition of Agents

The first problem one encounters when talking about agents is that there is currently no universally accepted definition of the term agent [49]. The debate about agents, agent architectures, implementations and fields of applications is as diverse as the possible domains in which agent technology is investigated. The variety of different demands on agents, the ability to learn in a given environment, the possibility of reasoning about a domain, or purely reactive behavior are the main reason for this lack of a general definition. Terms like rational agents, intelligent agents, reflex agents, utility-based agents, etc. are each used to emphasise different aspects of agent architectures, but they do not necessarily represent distinct classes of agents.

However there is a consensus about at least some aspects of agency which are of importance for this work and which will help to understand the role and behaviour of a camera agent in its environment.

A central notion of agency is the term *autonomy*. Considering autonomy as one as the central ideas of agents [49], [50] one can define an agent as follows: “An *agent* is a computer system that is *situated* in some *environment*, and that is capable of *autonomous action* in this environment in order to meet its design objectives”. Environments can be of very different types as seen in the examples above. This raises an important question: what does the term *autonomy* mean? We again refer to [49] and describe autonomy of agents as the possibility to act without the intervention of humans or other systems and to have control both over their own internal state and

their behaviour. [38] uses a stronger definition of autonomy: “A system is autonomous to the extent that its behaviour is determined by its own experience”. This definition implies the possibility of learning and adaptation to an environment. Our agent has a restricted learning mechanism via perceptrons to classify situations in narratives, but this capability is not adaptive. As stated above, the terminology concerning agents is not completely consistent. We will use the term *autonomy* as defined by [49].

To influence its environment an agent requires *effectoric capabilities* by applying *actions* to the environment. The possibility of acting depends on the current state of the environment and the agent itself, thus actions are associated with *pre-conditions*. Since agents often only have partial control over their environment, the possibility of *failure* has to be considered, which is another important notion of agency: Agent-based software has to consider the possibility that actions applied to the environment may not have the desired effect and thus has to provide solutions for such cases. In contrast to this, one normally assumes that if a procedure of a software system is called with given input parameters, the resulting output is correct for the given environment. This is not necessarily true for agents and even an inherent idea of agent technology. We will return to this topic when we compare object-oriented and agent-based software design. Most of the discussed aspects of agency arise due to the environmental properties the agent is situated in. In the following section we will present classifications of environments and explain their importance for our agent. For a more detailed introduction to agents see [38], [49].

3.3 Environmental Properties

The difficulty of designing agents suited for a given environment strongly depends on the amount of environmental knowledge and control the agent has. In [38] the following classifications to describe the most important environmental properties can be found:

Accessible vs. inaccessible This class stands for the accessibility of information for an agent within the environment. If an agent can obtain complete information about its environment we talk of accessible environments. Otherwise, information is (at least partial) inaccessible.

Deterministic vs. non-deterministic Environments are deterministic if the next state of the environment is completely determined by the current state and actions selected by agents. Thus an action always has a single, guaranteed effect.

Episodic vs. non-episodic Episodic environments represent repeating scenarios or problems of the same kind. Each episode is independent of previous episodes, especially of the agent’s interaction with the environment.

Static vs. dynamic Static environments do not change except due to the agent’s actions on that environment. In particular this means that during the agent’s decision-making it can be assumed that the state of the world after decision-making is the same as before and thus the resulting action and its effect to the world will be as predicted. Dynamic environments can change during decision making and beyond the influence of the agent, creating room for failure.

Discrete vs. continuous Discrete environments have a fixed, finite number of actions and percepts like a game of chess. Continuous environments represent infinite spaces of possible percepts or actions.

These central classifications strongly influence the complexity of designing and implementing agents for a given environment. The “real world” corresponds of course to inaccessible, non-deterministic, non-episodic, dynamic, continuous environments (which is the worst case scenario from an agent point of view). In the context of this work about camera agents, one could find different classifications for target environments. But for the case of interactive narratives and games like [46], which are commonly based on continuous, 3-dimensional environments, the following environmental properties can be assumed:

Inaccessible We want to provide all necessary information for our camera agent to find a proper visualisation for a situation within the environment. However depending on the environment this is not always possible so that the camera agent has to be designed in a robust way. During the integration of the agent into the computer game Half-Life (see Chapter 6) it became apparent that not all game-engine data accessible by the agent is consistent. Just to name an example, the animated character models within the game have no permanently consistent representation of the character midpoint or its extends, which is critical information for a correct visualisation. During the first attempts this resulted in framing the chest of a character instead of taking a close-up shot of the facial expressions during a dialogue. We partially have to deal with wrong or inaccessible information but it will not always be possible to perfectly resolve such conflicts.

Non-deterministic Since we are dealing with interactive environments like games where the human player has a certain amount of control and where other software agents like artificial characters are part of the environment, the camera has to deal with a strongly non-deterministic world.

Non-episodic As we have argued in our analysis of cinematography, the visual aesthetics, means of expression and experienced “soundness” of some filmed situation can depend very much on previous actions and filmed sequences. Thus our agent has to deal to some extent with temporal interdependencies between events within a story and consider the history of shots so far.

Dynamic For similar reasons as for the non-determinism of the environment, our agent has to cope with highly dynamic changes of the environment. Even worse, our agent has absolutely no influence on its environment since it is only presenting it to the audience and should not interact with objects or characters to fulfill its goals.

Continuous If one considers current games (which are so far the only really accessible and elaborated form of digital interactive stories providing the possibility to integrate a camera system) it becomes obvious that they usually take place within a continuous 3D-space with continuous time. Of course there exist discrete kinds of games or interactive narratives but an interesting autonomous camera system does not make sense for board games and the possibility elaborated visual storytelling is rather limited within discrete domains.

Being inaccessible, non-deterministic, non-episodic, dynamic and continuous our agent actually has to be designed for a worst case scenario, considering the difficulties arising in such environments. This of course already follows from the fact that our goal is to create a *camera* agent partially implementing real-world cinematography.

Before we go into more detail about our specific agent architecture, the next section discusses why it makes sense to distinguish agent-based approaches from other related methods for software design such as object-oriented programming. We will present the advantages of using ideas from agent-theory to design our system and will show that an object-oriented approach only would possibly result in designs that are more difficult to control and verify.

3.4 Agent-Based vs. Object-Oriented

As we wanted to create a working prototype during this thesis which can be integrated into some commercial software in order to prove the practicality of our approach we had to use a programming environment and language that meets the strong demands of current real-time applications in the context of interactive storytelling. Most of the current software in this field, especially 3D-games, is implemented using an efficient object-oriented language like C++ [41]. Additionally, ideas originating from agent-theory are increasingly used to realise autonomous objects like non-player characters in games.

The question that arises is what advantages an agent-based approach provides in comparison to common object-oriented programming patterns and designs [33]. Wooldridge writes in [49] that if one considers the properties of agents and objects one finds that *objects are defined as computational entities that encapsulate some state, are able to perform actions (or methods) on this state, and communicate by message parsing*. The similarities to agents as entities are obvious, yet there are some very important differences.

The first is related to the notion of autonomy. One of the central principles of object-orientation is *encapsulation*, which means, that an object has control over its own internal state. In languages like C++ or Java one has the possibility to declare internal variables as *private* so that they only can be accessed via provided *public* methods. This way it is not possible to change the state of an object without using a publicly available method that can control the change of state. Therefore an object has control over its state and is in this way autonomous. Yet in contrast to agents, control over an object's *behaviour* is not part of this autonomy. If a method is made available to other objects, they are free to use this method at any time and there is no way to restrict the execution of it. In particular other objects make the assumption that they can rely on the behaviour of this object once to be deterministic. This is normally not a problem since such designs are build under the premise that objects share a common goal. In most multi-agent systems however this assumption is on longer valid. Such control over the own behaviour by others can lead to problems or even dangerous situations for a software or, even worse, for the people using it. If one thinks of software from competing companies sharing the same environment, this problem should become rather obvious.

So if we are talking about agents, it is not necessarily guaranteed that an agent executes an action that was requested by another agent. The question if an action is performed should depend on the agent's decision-making and it should only be executed if the action is appropriate in the given situation. Thus in contrast to the object-oriented view we cannot talk about the *invo-*

cation of methods but of *requested actions*. This difference is important since one cannot rely on the other agent to execute a requested action.

This does not mean, that agents cannot be implemented using object-oriented techniques. The point is that autonomy, one of the central notions of agent-theory, is not part of the basic object-oriented model.

[49] presents further differences related to the idea of flexible (divided into reactive, pro-active and social) autonomous behaviour and contingency. These points are not important for this work, since our agent does not meet this definition of flexibility nor do we use techniques for multi-threading. Thus these distinctions between object-orientation and agents does not contribute to this work.

But there is another important reason to use agent-theory for the model of our camera system. One of the motivations for object-orientation was to find a representation and language that makes the modelling and understanding of complex software systems easier. Using object-orientation, some real-world problems or objects can be modelled much easier than using other approaches like functional programming. Our agent will be situated in environments which are often comparable to multi-agent systems in some sense. For instance consider an adventure game where other computer-controlled characters are part of the environment trying to compete with the player. To solve the problem of ‘fairness’ concerning the up-to-dateness of relevant information available to each of the characters it is a common approach to separate the perception and decision-making of the characters from the process of acting. Thus no character has more recent information than others and can draw advantage from that fact. This can be easily implemented if the separation has been considered during the design of the software structure. Using other object-oriented techniques not taking care of this necessity one could later run into severe problems, possibly having to restructure the whole software system. Designing our software for camera control from an agent point of view helped to provide an intuitive understanding of a complex system. Many different decisions concerning the behaviour of our software are related to each other in a way that is difficult to model intuitively by using means of object-orientation only. Terminology and architectures from agent-theory provide these means.

Furthermore our first prototypical model of a camera agent could be the basis for a formalisation of this model using a symbolic representation for techniques of logical reasoning. While a symbolic formalisation of our agent is beyond the scope of this thesis, designing the software based on agent-theory could provide a good starting point for such an endeavour.

3.5 Symbolic vs. Non-Symbolic

Before going into detail concerning the architecture, the question whether to use a symbolic representation of the agent to provide symbolic reasoning or to use the agent architecture for software design using a non-symbolic language has to be answered.

As stated before the goal for our camera agent is to work in interactive real-time environments. This raises strong demands concerning internal efficiency and also concerning the possibility of exchanging data structures between the camera and the target application. Symbolic approaches have the obvious advantage of logical deduction and thus complex reasoning abilities about the behaviour of the agent. For most planners however, this is only true in static worlds where there is enough time to reason and where the world does not change in strongly non-deterministic ways. For a camera agent working in a very dynamic, inaccessible, non-deterministic and continuous environment, the possibilities of logical deduction provide no advantages. In highly interactive environments planning ahead for some time does make little sense since the environment can change very fast and prior plans get easily obsolete. Instead the agent has to make instant decisions that can be put into action in a very short time. Additionally we want to create a camera agent which can be easily integrated into existing interactive narrative applications. Current software in this field is commonly based on object-oriented languages like C++ [22], [46]. Having a symbolic representation of our camera agent, it would be very difficult to integrate it into such a system if one considers the exchange of data and real-time performance.

Thus we decided to use an agent-based architecture for software design using a common object-oriented language and not to formalise every aspect of our agent using a symbolic representation. We do not get symbolic reasoning but we still manage to provide the necessary planning abilities needed for a camera agent to work appropriately in its environment.

3.6 Agent Architectures

There exist various more or less abstract architectures for agents describing the different aspects and specialisations of the diverse agent types. We will not go into the details of all different architectures and of logic-based representations of those architectures since this would not yield advantages for our non-logic based approach and because the number of different architectures is far to large. Introductions to abstract agent architectures can be found in [38], [49]. Instead we will describe basic architectures we considered for

the design of our software and show how they helped to provide intuitive understanding of our system.

The most abstract description for *standard agents* is that of a function $action : S^* \rightarrow A$, which maps a sequence of environment states from a set of states S to some action $a \in A$. Intuitively the agent does some action on the basis of its history. The behaviour of the environment can then be described by a function $env : S \times A \rightarrow \wp(S)$.

While this abstract view of agents provides means for analysis, it does not help in designing agent-based software. Thus we will refine this model into sub-systems (based on [49]) to provide a basis for our camera agent architecture. It will help us to understand data structures and the internal flow of control. The first idea is to split the agent into a subsystem providing means for the perception of the environment and one for decision-making and acting upon the environment. The agent perceives its environment via sensors which can be formalised by a function $see : S \rightarrow P$ from a set of states to a set of percepts P . Decision-making and acting upon the environment is then provided by a function $action : P^* \rightarrow A$ from sequences of percepts to some set of actions. To better understand and model the behaviour of agents it is useful to add an internal state to the architecture which expresses that decision-making is based on external environmental percepts and the current knowledge or state the agent has. This can be formalised via a new action function $action : I \rightarrow A$ from a set of internal states I to a set of actions. Additionally we have a function $next : I \times P \rightarrow I$ which updates the internal state based on the current state and percepts. The function see remains unchanged.

This basic architecture of agents with an internal state is one way of splitting the very abstract architecture of standard agents into more intuitive descriptions providing clues for software design. Based on this architecture there exist a variety of different modifications and extensions like reactive architectures, belief-desire-intention architectures, layered architectures and so on. Each of these focuses on different aspects of a problem domain and offers a different understanding of control and data structures.

In this thesis we will use an architectural approach splitting the agent architecture into three main subsystems or modules, each dealing with different processes. Our architecture is motivated by conceptual ideas, that separate perception and knowledge from higher order decision-making and the realisation of committed actions or behaviours, and considerations about fairness and up-to-dateness of relevant information in multi-agent systems.

Chapter 4

The Camera Agent

“The camera is my tool. Through it I give a reason to everything around me.”

– André Kertész

The following sections will present the structure and the different components of the camera agent system from an general overview about the agent and its environment to the details of every subsystem and their role within the overall architecture. The structure of the presentation also represents the general flow of information within the agent and thus helps to understand how the system works. Since we used object-oriented techniques throughout the development of the camera agent, we will stick to these structures in our figures, too. All methods, algorithms, and data structures were implemented in C++.

Furthermore, camera handling in virtual 3D environments naturally involves complex vector mathematics and geometrical computations. However a complete introduction into the fundamentals of computer graphics would be far beyond the scope of this work. For the understanding of the concepts behind our camera agent and its decision-making it is not necessary to know about the mathematical details of real-time 3D computer graphics generation. Basically, current 3D-environments for games or virtual reality applications are normally based on a global cartesian world coordinate system representing the absolute 3-dimensional coordinate space used in the application. In such a system the position and orientation of all objects can be computed and thus communicated to the camera agent for its decision-making and shot calculation. Correspondingly, an application can evaluate information about the position, orientation, and field-of-view of a virtual camera in this coordinate space to create the corresponding view of the 3D-scenery to a spectator. Please see [1], [15], [51] for a detailed introduction

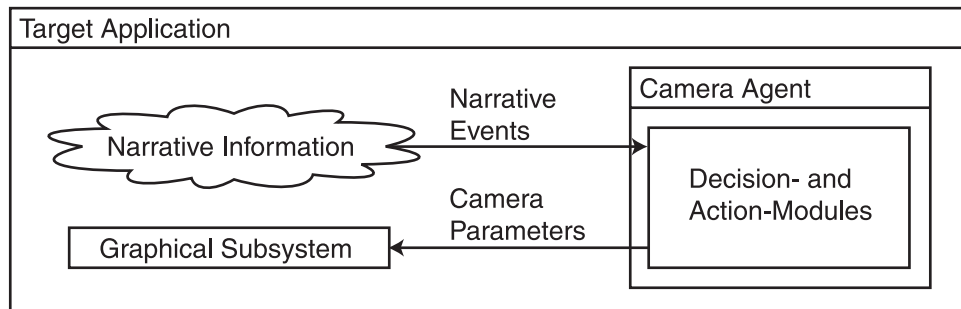


Figure 4.1: General system structure and flow of information.

into the principles of computer graphics and the underlying mathematics. In Appendix A we will present some of the more advanced geometrical algorithms used in this work which allow easy configuration of camera shots in 3D-environments.

4.1 General System Structure

One of the main ideas behind this work was to develop a camera system that can be used in almost any kind of environment providing narrative or dramaturgical information that can be expressed or emphasised by images. Motivated by ideas from autonomous software agents, we created an agent-based system that can easily be integrated into existing narrative applications. To create a camera agent that is independent of specific knowledge of a target application we use an interface for communication based on the analysis of means of expression of cameras, which is expressive enough to communicate all dramaturgically relevant information from the narrative application to the camera agent. This is achieved by *narrative events* as described in the following section about communication.

A sketch of the overall system and the flow of information between the camera agent and the narrative application is shown in Figure 4.1. The narrative application sends relevant information about the current or future state of the narrative in the form of narrative events to the camera agent. Based on cinematographic knowledge the agent decides which events should be considered and how they should be visualised. It then sends information about camera position and orientation back to the graphical subsystem of the narrative application. Current applications based on 3D-graphics are normally able to use the information directly for visualisation. The advantage of this system is the possibility to integrate the camera agent into existing narrative

software and keep the necessity for modifications limited to the translation of relevant information within the target application into narrative events. The integration of the agent into the computer game Half-Life as presented in a later section shows one example of the practicality of this concept.

Before going into the details of the agent architecture, we present the basis for communication between the agent and the application to provide a better understanding of the agent's decision-making process.

4.2 Communication

The most important aspect of the communication between narrative application and the agent was to find a way to reduce the necessity for a complete world representation with all its objects and their internal states in the camera agent, while still providing enough information to enable the agent to choose dramaturgically appropriate visualisations. The representational complexity had to be reduced to a language between the narrative application and the agent that is able to communicate relevant information only. The idea to solve this problem was to let the target application communicate only the 'script' of the story to the agent. We assume here that an application representing a, possibly interactive, narrative of some form has access to all relevant data. For example if there exists an explicit story-path or sequence of story-events which the application presents to a user, the application should be able to send information about these events to the camera. A camera system using a different approach like regarding the camera as an agent that has to gather and analyse *all* information within the environment itself would result in difficulties considering a consistent story-analysis, target-application independence, data representation, and real-time performance. Instead it is the task of the narrative system to 'tell' the camera agent via a specified language or interface about events happening during the progress of the story. The camera agent then analyses these information and chooses the best possible camera shot to appropriately present this event to an audience. The basis for communication, *narrative events*, are explained in more detail in the following section.

4.2.1 Narrative Events

A *narrative event* (Figure 4.2) represents the basic framework for communication between the narrative system and the agent. We know about the means of expression of cameras from the section about cinematographic concepts. Now we have to formalise this knowledge using a fixed structure

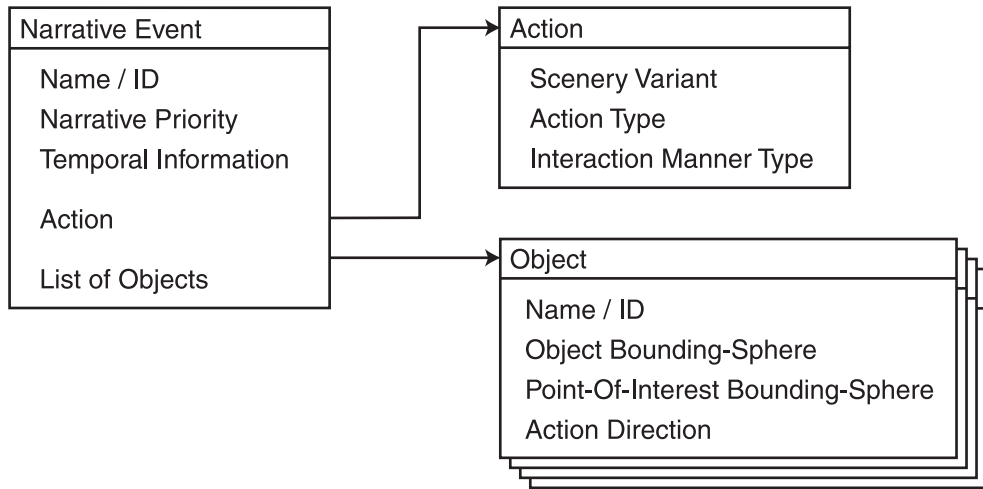


Figure 4.2: Structure of narrative events.

or language. The overall problem was to find a way to abolish the necessity for implementation-specific or application-dependent knowledge like the ‘emotional state engines’ of the participating characters in a narrative situation for example. In other works like [45], the agent needs to know about platform-specific data structures to be able to extract dramatically relevant information. We found that relevant narrative information about events and situations in a story can be formalised by a sentence-like structure consisting of subject, action or verb, and objects. Furthermore, the only ‘carrier’ of dramatically relevant information is the verb itself. So by providing a kind of additional ‘adverbs of manner’, necessary dramatic information can be integrated within the action and thus no further object- or implementation-specific knowledge is needed. This is one of the main reasons for the independency and portability of our system. All of the following classifications of narrative events are based on our analysis of cinematographic concepts.

The first classification for separating different types of events is derived from the idea of *scenery variants* as introduced in chapter three. In cinematographic literature we find the following three basic variants:

- Dialogue only
- Dialogue & Action
- Action only

The second classification separates different types of actions into the following three classes:

- Physical actions
- Mental actions
- Predicates

Interestingly we did not need more strict classes to describe actions appropriately. One could expect that the action of fighting against a character would have to be separated from a character drinking a glass of milk for instance. But in our experience it is not necessary to differentiate between these two actions by separate classes¹. It is obvious that it would be difficult to divide all known actions in a language into distinct classes. Instead, in addition to the above general classification, we will introduce a kind of fuzzy description of actions based on the means of expression of cameras. Since this description is based on the actual expressiveness of cinematography² it is expressive enough to handle all dramaturgically relevant information for a camera on the one hand and does not allow cinematographically useless information on the other hand.

The descriptions in the sense of adverbs of manner associated with a narrative action are:

Distance of Interaction expresses the scope of an action, ranging from distant (−1.0) to close (1.0).

Object Might expresses the might associated with the action or the acting object, from isolated (−1.0) and humiliated to mighty or demonic (1.0).

Emotional Involvement expresses how much the spectator should get involved into the action, from low (−1.0) and objective to high or subjective (1.0).

Calmness expresses the dynamics of a scene, from calm (−1.0) to hectic or confused (1.0).

Excitement expresses the tension, gravity or seriousness of an action, from unexcited (−1.0) to excited (1.0).

¹Like a class for ‘fights’ or ‘violent action’ and a class for ‘harmless actions’

²...at least the part of it open to formalisation and not ruled by the mysticisms of the creative genius.

These five categories are communicated as real-valued variables, ranging from -1.0 for low values to 1.0 for high values as shown in the above description. 0.0 stands for medium or neutral values. If the narrative application sends information about a narrative event to the camera agent, it creates a classification based on the scenery, the action type, and on these fuzzy descriptions of the manner of interaction. Some examples of different narrative events are given in the following section.

The agent uses a hybrid approach of rule-based preconditions and neurons in order to classify the action and find a matching shot. We will describe the details of this in the discussion of the agent's knowledge submodule.

The second core data contained in a narrative event is the list of participating objects. They only have to provide some physical parameters for correct shot creation. The object parameters communicated with narrative events are:

Object Bounding-Sphere The bounding-sphere (BS) enclosing the character or object, given by a 3D position in the world coordinate system of the application and its radius.

Point-Of-Interest Bounding-Sphere The BS enclosing the part of the object where the current action originates. This can be the face of a character during a conversation, or the weapon during a fight for instance.

Action Direction A direction vector representing the main direction of the action originating from the Point-Of-Interest (PoI). If the PoI represents the face for instance this vector points away in the viewing direction of the character. For a weapon this would be the direction it is aiming at. This information is given by a 3D-vector in global world coordinates.

Object Identification We added this information to allow the narrative application to easily update previous narrative events. If an object moves for instance, the application can request all active objects of the currently visualised narrative event from the camera agent and then update their information based on its own data structures. This way object tracking by the agent can be easily realised.

See Figure 4.3 for two examples of objects and their respective bounding-spheres and action directions. Using these structures, all necessary geometrical information is found within the objects.



Figure 4.3: Two examples of bounding-spheres and action directions.

The subject of an action is specified by the first object added to the list of objects in our narrative event. All further objects added are targets of the action.

An action provides all necessary narrative information. Alternative approaches would have been to include the direction of the action with the action itself. But since we need an orientation for each object for correct shot creation and since the acting subject already provides this directional information we regard the action as describing a relation of how the objects interact in a dramaturgical sense. In addition we currently assume a global world coordinate system which the camera orientation is based on. Thus we are using the global world up-vector as a reference for the standard orientation of objects in 3D-space. This is correct for most 3D-environments with one coherent up-direction like narrative applications dealing with characters walking on ground or being in a room for instance. In a space-combat game where each object (like spaceships, planets, etc.) can have completely different local up-directions, the information about objects has to be extended by the local coordinate system of the object. We did not include this in our analysis and system because of the inherently much more complex camera shot evaluation. Standard film literature does not describe the topic of filming randomly oriented aliens floating in 3D-space and having a conversation while still expressing relationships between them like might or fear as discussed in the section about cinematographic concepts. These concepts cannot be transferred directly to scenes without a consistent up-direction. The above-mentioned concepts for influencing the visual interpretation need

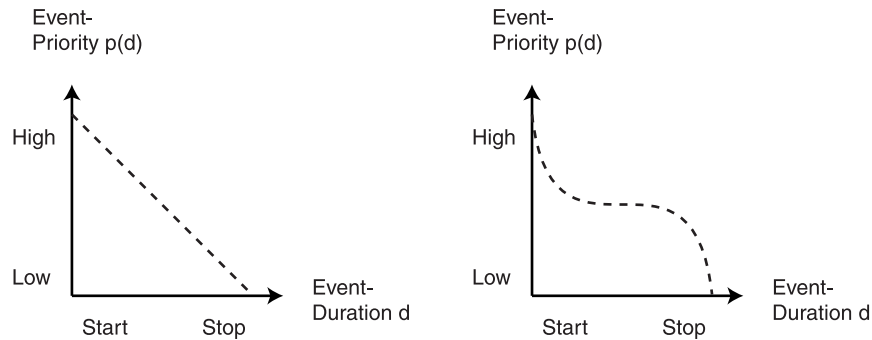


Figure 4.4: Simple event-priority decay functions.

clear means of orientation for the audience as found in everyday-situations to create the intended meaning.

Of course a change of parameters like the position of objects could also be realised by some kind of ‘change’-action like *object A walks to position B*. But this would mean that the camera has to keep track of all object parameters during the whole narrative even if those objects are of no relevance for the current situation and would therefore result in more complex knowledge structures for the agent. A narrative event should represent the information necessary for camera shots in a maximally reduced way.

Further information provided in a narrative event is:

Temporal Information The start-time of an event and, if available, its end-time. This information is needed for future shot preparation. Shots can be planned ahead and temporal dependencies between events can be considered during shot planning. Of course, such information is often not available for future events because of the interactive and unpredictable nature of such applications. But even in systems sending events about instantly occurring events, the agent has to be informed about the termination of an event. For example, associated events running in parallel could be chosen for a new shot, if the current event ended.

Narrative Priority A priority value describing the narrative importance of an event. In interactive applications, where there is no real control about all occurring events because of user interaction, one needs a mechanism for distinguishing possibly unimportant events like a conversation of some extra characters near the user from direct interaction of the user with some other character for instance. If such events occur in parallel, the agent has to pick the more important events for the

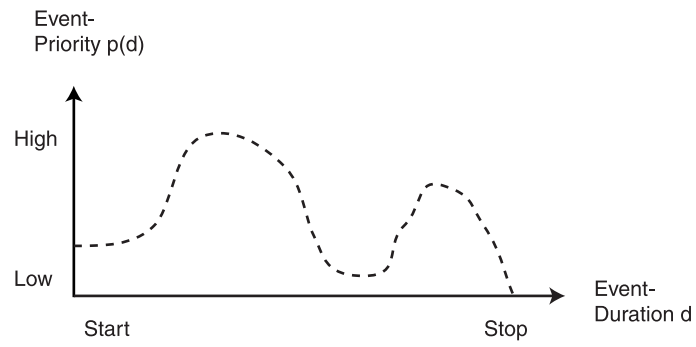


Figure 4.5: A complex priority decay function.

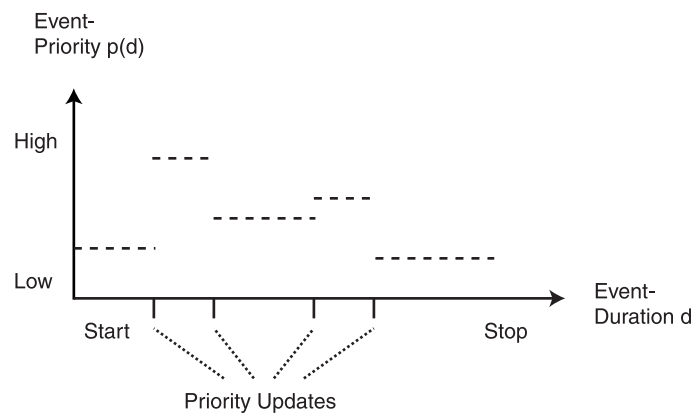


Figure 4.6: Representing complex priority changes by updating event-priorities.

progress of the narrative. This idea of event priority is used in several works. [20] for example proposes a kind of decay-function describing the development of the priority over time. The problem with these approaches is that they use only rather simple functions, which are not motivated by the actual action or scene of the narrative (Figure 4.4). One needs a complex function, which not only expresses decay but also the possibility of increasing priority, because one can easily create narrative situations like a small-talk dialogue for instance, which is of low narrative importance at the beginning and which gets more and more important over time. One solution would be to use complex higher order functions in order to represent an arbitrary development of the priority (Figure 4.5). Here the problem again is that the future development of such an event is not necessarily known due to the interactivity of the plot. Instead we chose a different approach which makes use of *coherence* between events. If for example the narrative importance of an event changes, the narrative application can easily re-send the previous event with updated narrative priority and temporal information to the agent. Due to the similarity of these events given by participating objects, the type of action, etc., the agent can detect coherence. This way we do not need a complex system to represent the priority of events but instead we inform the agent of important changes of the priority (Figure 4.6). If for example the currently filmed event becomes too unimportant in comparison to other running events, the camera agent can decide to switch over to these more important ones.

The above-mentioned coherence or degree of similarity between two narrative events is computed by a the following algorithm which is currently based on a comparison of the objects contained in both events. The decision-making is based on the following rules:

```
IF both events contain no objects
THEN RETURN 1.0;
ELSE IF only one event contains no objects
THEN RETURN 0.0;
ELSE IF the subject of both events is the same
THEN RETURN 1.0;
ELSE IF the subject of one event is an object of the other
THEN RETURN 0.8
ELSE RETURN the ratio of identical objects to the overall
sum of distinct objects occurring in both events;
```

Objects in Event 1	Objects in Event 2	Coherence of both events
-	-	1.0
A	-	0.0
A	B	0.0
B	B	1.0
A,B	B	0.8
A,C	B,C,D	0.25
A,C,D	B,C,D	0.5
A,C,D,E,G	B,C,D,F,G	0.42
A,C,D,E,F	B,C,D,E,F	0.66

Figure 4.7: Examples of the coherence of two events.

These numbers provided reasonable results, examples are given in Figure 4.7. Of course, other criteria like the type of action could be used additionally to compute the value for event coherence. But the idea behind the coherence of events is to distinguish different sequences of connected events. If a new event uses the same objects like its predecessor, we have a connected sequence of actions within the same scenery. If the coherence is low, new objects are introduced or a completely new scene has to be established by the camera.

Of course, other very important information like geometrical data for occlusion detection, the lighting situation, or even sound have to be dealt with in a complete camera system. Because of the complexity of all these issues we focused on the basic means of expression of cinematography. Work on geometrical aspects of camera control is given in [19]. In [45] lights which can be placed by the camera system are used to emphasise dramaturgy. Since our agent is not allowed to change the environment, future work on the integration of light should be focused on considering the given lighting situation in a scene during shot creation instead of rearranging lights.

These topics are beyond the scope of this thesis and possible goals of future research, but the system design of the camera agent should be open enough to integrate additional interfaces or more complex internal knowledge representations similar to the ones presented to solve the above problems.

Inherent in the idea of communication between the narrative system and the camera agent lies a mechanism to control the camera agent's freedom of choice and visual interpretation by the story author. The more events with less detailed information like action classifications are 'told' to the camera agent, the more it is free to decide which events to choose and how to dramaturgically interpret the situation. The integration of the agent into

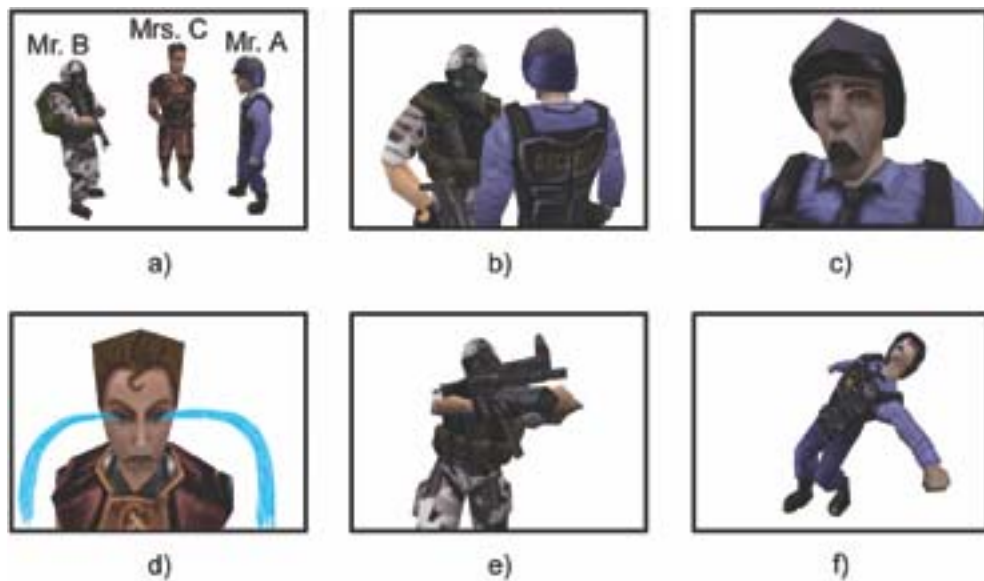


Figure 4.8: Output of the camera agent for the example of a conversation between three characters and some action. a) Establishment shot: Mr. A talking to Mr. B, Mrs. C is listening. b) Over-the-shoulder shot: Mr. B talking a little upset to Mr. A. c) Close-up shot: Mr. A talking very angrily to Mr. B. d) Overhead shot: Mrs. C crying. e) Mr. B shoots at Mr. A. f) Mr. A falls backwards.

the game *Half-Life* is such a case, where the underlying application does not have an explicit script or representation of narratively important story-elements but sends a lot of currently happening events like state-changes of objects in the game to the agent. Missing knowledge could be deduced from other information like the speed and position of objects for example. The less in number and better specified the events are, the more the author has control over the camera's behaviour. Furthermore the system could be easily adapted to work in more interactive, decentralised applications like distributed virtual environments, where one would not want the narrative application to tell the camera agent about each ongoing action for reasons such as data-encapsulation, but where other agents like characters themselves create narrative events and communicate with the camera.

Examples of Narrative Events

We are now going to show a few examples of creating and sending narrative events from the application. The pseudocode is based on the actual C++

implementation but leaves out some implementation-specific detail for simplification and readability. We will use a test case where three persons, Mr. A, Mr. B, and Mrs. C, are having a conversation which is getting increasingly aggressive, ending with Mrs. C crying and Mr. B shooting Mr. A.

This first code fragment shows how to communicate the beginning of the situation to the agent. First we set parameters for the action, for the participating objects, and then for the narrative event which encapsulates this information. At first the dialogue is neutral, thus we do not add any information about the manner of interaction.

```

Action->SetSceneryType("Dialogue");
Action->SetActionType("Physical");

Object1->SetObject(3D-Vector, BS-Radius);
Object1->SetPointOfInterest(3D-Vector, BS-Radius);
Object1->SetActionDirection(3D-Vector);
Object1->SetObjectID(1);

Object2->...
Object3->...

NarrativeEvent->SetName("Mr. A talking to Mr. B,
                        Mrs. C is listening.");
NarrativeEvent->SetNarrativePriority(1.0);
NarrativeEvent->SetStartTime(Current time);
NarrativeEvent->SetStopTime(Current time + duration);
NarrativeEvent->SetAction(Action);
NarrativeEvent->AddObject(Object1);
NarrativeEvent->AddObject(Object2);
NarrativeEvent->AddObject(Object3);

CameraAgent->RegisterNewNarrativeEvent(NarrativeEvent);

```

The camera agent should decide to create an establishment shot of the scene showing the three characters like in Figure 4.8 a). We will leave out the information about objects in the following examples to provide a better understanding of what information changes between these different narrative events. The conversation gets more hectic thus we provide information about the ‘Calmness’ of the situation and we express emotional involvement of the audience.

```

Action->SetSceneryType("Dialogue");
Action->SetActionType("Physical");
Action->SetInteractionMannerType("Calmness", 0.5);
Action->SetInteractionMannerType("EmotionalInvolvement", 0.5);

Object1->...
Object2->...

NarrativeEvent->SetName("Mr. B talking upset to Mr. A");
NarrativeEvent->...

CameraAgent->RegisterNewNarrativeEvent(NarrativeEvent);

```

The agent would now create an over-the-shoulder shot of Mr. B's face as shown in Figure 4.8 b). Now Mr. A gets really angry and Mr. C starts to cry so we send two events with the same temporal information to the camera agent.

```

Action->SetSceneryType("Dialogue");
Action->SetActionType("Physical");
Action->SetInteractionMannerType("Calmness", 1.0);
Action->SetInteractionMannerType("EmotionalInvolvement", 1.0);

Object1->...
Object2->...

NarrativeEvent->SetName("Mr. A talking very angrily to Mr. B");
NarrativeEvent->SetStartTime(Current time);
NarrativeEvent->SetStopTime(Current time + duration);

CameraAgent->RegisterNewNarrativeEvent(NarrativeEvent);

```

The second event:

```

Action->SetSceneryType("Action");
Action->SetActionType("Physical");
Action->SetInteractionMannerType("ObjectMight", -1.0);
Action->SetInteractionMannerType("EmotionalInvolvement", 1.0);

Object1->...

```

```

NarrativeEvent->SetName("Mrs. C starts crying.");
NarrativeEvent->SetStartTime(Current time);
NarrativeEvent->SetStopTime(Current time + duration);

CameraAgent->RegisterNewNarrativeEvent(NarrativeEvent);

```

The agent should show both the angry conversation using a close-up shot because of the high emotional involvement (Figure 4.8 c)), and Mrs. C crying, using an overhead shot to express her weakness and loneliness (Figure 4.8 d)). The situation escalates, Mr. B shoots Mr. A:

```

Action->SetSceneryType("Action");
Action->SetActionType("Physical");
Action->SetInteractionMannerType("Calmness", 1.0);
Action->SetInteractionMannerType("ObjectMight", 1.0);
Action->SetInteractionMannerType("EmotionalInvolvement", 1.0);

Object1->...
Object2->...

NarrativeEvent->SetName("Mr. B shoots at Mr. A.");
NarrativeEvent->...

CameraAgent->RegisterNewNarrativeEvent(NarrativeEvent);

```

The fast and hectic action expressed by the ‘Calmness’-value would imply short, fast cuts, focusing on the weapon first (Figure 4.8 e)) and then on the target Mr. A (Figure 4.8 f)) for instance.

This example shows, how different types of actions and the associated dramaturgical information can be expressed using narrative events. How these events are evaluated is discussed in the section about decision-making and shot-creation.

4.2.2 Camera Position and Orientation

The output of the calculation of a camera shot for the currently active narrative event is a 3D world space camera position and orientation. As said above it can be assumed that 3D-based applications or environments can work with coordinates given in their global world coordinate system. The minimum information describing all necessary parameters to render a scene in a 3D-environment is the following:

I_j	Value of input j
W_j	Weight associated with input j
V	Vector of inputs
In	Weighted sum of all inputs
g	Activation function
O	Activation or output value
T	Target activation value
Err	Error, difference between output and target value
n	Number of inputs

Figure 4.9: Notation.

Camera Position The proposed 3D world space position for the camera.

Camera Orientation Given by three vectors in world space coordinates describing the local coordinate system of the camera, or alternatively by a target point of the camera and one additional vector describing the up-direction for the camera.

Field Of View The field of view describes the viewable area covered by the view-frustum originating at the camera position.

This information is provided to the application by the camera agent and can be accessed via interfaces.

4.3 Neuron-based Decision-Making

While a lot of the involved decisions can be represented using if-then-else rules, especially the fuzzy representation of the manner of interaction by real-valued parameters needs another approach. We decided to use trainable neurons as deciders. The agent uses a neuron model with a sigmoidal output function to calculate a motivation for each known behaviour, given a narrative event. This motivation will then be used to decide the best matching behaviour or camera shot for this event. We will give a short introduction into the theories of single neurons based on [38] and provide informal arguments, that such a neuron can be trained with our representation of the manner of interaction in narrative events.

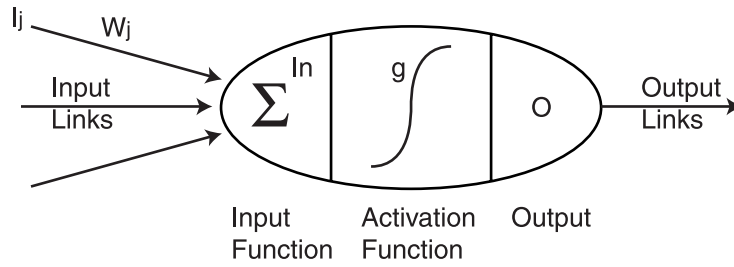


Figure 4.10: A single neuron.

4.3.1 A Single Neuron

A single neuron, also called a node or unit in neural network terminology, provides input- and output-links to its environment. Each input-link has an associated numeric weight, which represents how much the link's value is taken into consideration during the output calculation. By modifying these weights one can change the unit's input/output behaviour according to the expected behaviour. Thus weights are the primary means of long-term storage. Learning is realised by updating weights. A unit computes its activation level and output by building the weighted sum of all input links and then applying a non-linear function to this sum.

$$In = \sum_{j=1}^n W_j I_j$$

$$O = g(In) = g\left(\sum_{j=1}^n W_j I_j\right)$$

See Figure 4.9 and 4.10 for notation. By choosing different functions g , one can produce different neuron models. Common choices are step, sign, and sigmoidal functions. Since we want to use the output value of different neurons as a measure of how much different agent behaviours are suited for a given event, we use a sigmoidal output function. This allows us to create a list of behaviours sorted by this activation value. We currently use:

$$g(x) = \textit{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$

Instead of an explicit threshold representing the minimum total weighted input for the unit to 'fire', or in our case, to return a value $O > 0.5$, we

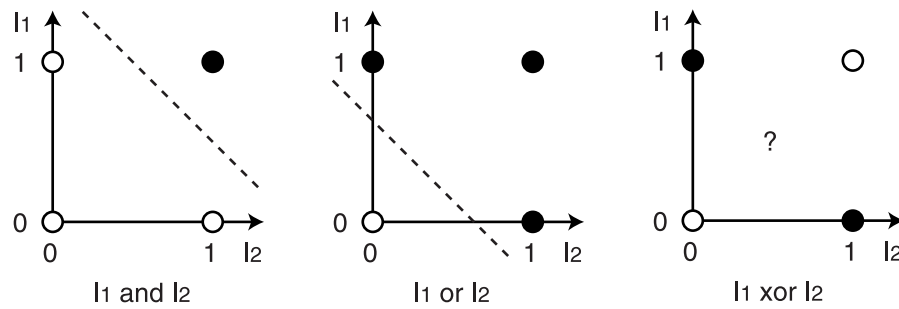


Figure 4.11: Linear separability of boolean functions having two inputs.

equivalently add an fixed input $I_0 = -1$ and a weight $W_0 = t$, where t is the desired threshold, to provide a simpler learning mechanism.

$$O = \text{sigmoid} \left(\sum_{j=0}^n W_j I_j \right)$$

It was proven that such neurons, or single-layer, feed-forward networks³ in general with more than one neuron (see [38]) can represent linearly separable functions only. Figure 4.11 shows examples for linear separability of boolean functions *and*, *or*, and *xor* with two inputs I_1 and I_2 as given in [38]. To be linearly separable the n -dimensional input space of the target function has to be divided in two by a hyper-plane defined by $\sum_{j=0}^n W_j I_j = 0$. If this precondition is met there exists an algorithm to learn any linearly separable function given enough training examples. Each example consists of a vector V of input values and the desired output value T . The algorithm computes the output O of the neuron for V and then updates all weights according to the error they introduced to the result. The error is the difference between T and O :

$$Err = T - O$$

Then the necessary update of each weight is given by

$$W_j = W_j + \alpha I_j Err$$

where α is a numerical value representing the learning rate of the neuron. The algorithm for learning a set of examples by a single neuron is given by the following pseudo-code:

³Also known as perceptrons.

```

REPEAT
  FOR EACH e IN examples DO
    O = output of neuron given input values of e;
    T = desired output for e;
    update all weights of neuron based on T, O and e;
  UNTIL all examples are correctly predicted OR
    stopping criterion is reached;

```

In the following we will argue that a single neuron can be used to learn different activation values for actions described by weighted adverbs of manners, as introduced in our section about narrative events.

4.3.2 Neurons Classifying Actions

We introduced a fuzzy description for actions in narrative events based on the manner of interaction, given by: *Distance of Interaction*, *Object Might*, *Emotional Involvement*, *Calmness*, and *Excitement*. These descriptions are variables ranging from -1.0 for low values, 0.0 for medium values, to 1.0 for high values. The idea is to use these values as inputs for a decider, that returns a high activation for some specific interval within the above range of values. This is interpreted as a match of the behaviour associated with the decider and the tested action. Otherwise the decider should return a low activation. If each behaviour or shot of our agent is associated with such a decider, we can chose a matching behaviour for a given action based on the activation value.

Instead of a possibly large set of different hand-tuned, rule-based deciders, which would be difficult to maintain especially for non-technical persons like artists working with the camera agent, we wanted to provide a more intuitive tool. Using the above neuron model gives us the possibility to train different neurons with examples of different action types and narrative events to create such deciders. To do this, we have to find a representation of the above variables which is linearly separable.

The problem is that intervals are of course defined by two interval borders. If we would map one of the above mentioned fuzzy action descriptions to an input of a single neuron directly, we could not represent a high activation of the neuron for values inside the interval and a low activation for values outside the interval, since the two interval borders divide the input space into three sections instead of two (Figure 4.12) and would therefore not be linearly separable.

The solution we chose was to split every variable describing one of the above parameters into two variables: one representing the range from -1.0

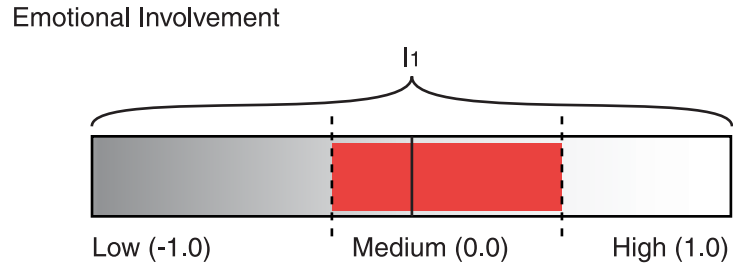


Figure 4.12: An interval separating one input into three sections.

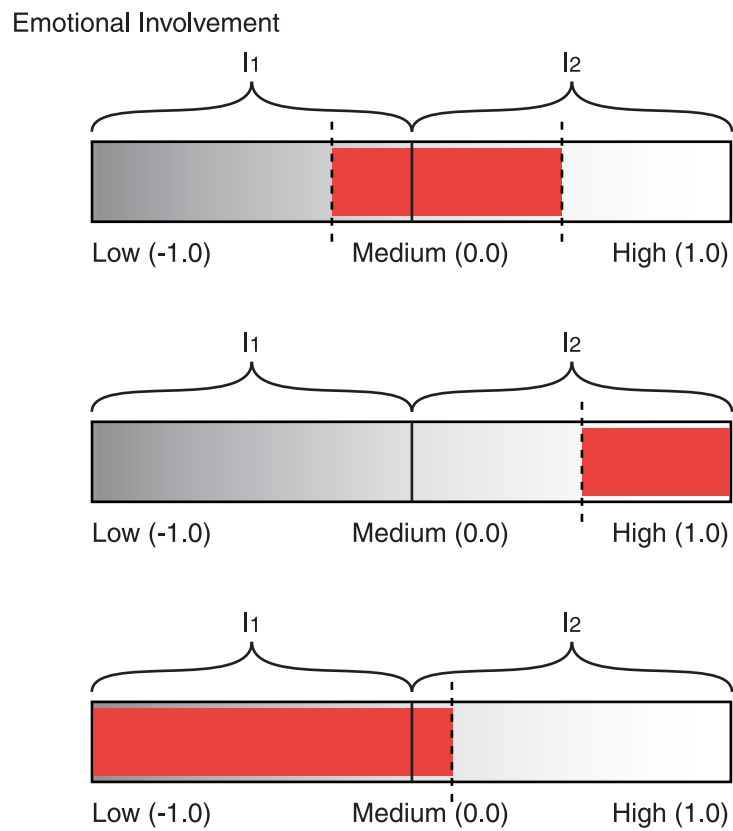


Figure 4.13: Examples of representable intervals using two inputs.

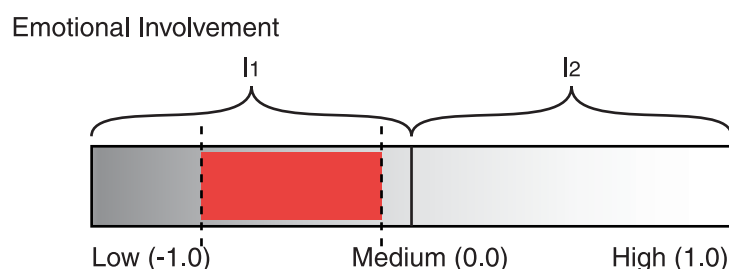


Figure 4.14: Example of non-representable intervals.

to 0 and one for 0.0 to 1.0. Each variable is then mapped to a separate input of the neuron. We still cannot represent any interval since if we want for example a high activation for values from 0.3 to 0.5 only, both interval borders would have to be recognised by one neuron input, which violates the constraint of linear separability. But we can represent intervals, where only one interval border lies in the range of one input variables. This however is expressive enough to represent most of the necessary standard applications for our decider. Figure 4.13 gives three examples for representable intervals. Using our method we can create up to three neurons recognising distinct intervals for each manner of interaction. This way we can represent rather unspecific shots suited for any type of action description to very specific ones.

Figure 4.14 gives an example where both interval borders lie within one of the inputs and where input space is not linearly separable. To represent arbitrary input intervals, networks of neurons with several layers have to be considered. Extending our basic neuron model to more complex networks, which are capable of learning any type of specialisation should be a goal of future research.

Additionally we assume a single connected interval for high neural activity. If one considers the representation of *Object Might* for instance, shots from a higher angle would emphasise the loneliness of a character, eye-level shots would provide a more neutral visualisation, and shots from below create the impression of a mighty, dangerous character. During our analysis of shots we did not find shots capable of visualising contradicting intentions. This means, by our findings it is not possible to create a single shot emphasising loneliness and might at the same time, which is on the other hand not suited for neutral characters. Thus we can assume a single connected interval within our parameters which causes a high neural activity.

The linear separability for the overall input vector consisting of our 5-dimensional action description is based on the following assumption about

shots. In a single neuron each input of the input vector can only influence the final output of the neuron in one direction, no matter what the other input values are. Especially, there is no way to represent an exclusive or-function. But for similar reasons as above, we do not have a necessity for shots, which are suitable if and only if one of two parameters has a high value. Instead, a shot, which can be used for short distances and which can be used to emphasise loneliness, can also be used for short distances in combination with lonely characters. A shot for specific intervals of our five parameters will also be suited for any combination of these parameters. Intervals for a high neural activation for one parameter will not change depending on other parameter's intervals. Thus we have no interdependencies between our different manners of interaction.

Accordingly we can use the above neuron model to learn practically relevant, consistent examples based on our 5-dimensional action classification. Two additional neurons will be used for decision-making within the narrative event analyser. These neurons work with only three simple input parameters and linear separability should be obvious there.

By using a sigmoidal activation function for our different neurons and by interpreting their output as the 'motivation' to apply the associated behaviour to the tested action, we can generate a sorted list of matching behaviours for further decision-making.

4.4 Architecture

The camera agent's architecture is divided into three subsystems, motivated by agents with state and some considerations in order to separate decision-making and acting. In narrative applications like games one wants to assure that each agent interacting with the environment and especially with other agents is treated in a 'fair' way concerning the up-to-date-ness and relevance of the available information. If one looks at round-based applications for example, where all agents are sequentially called to perceive and act during each iteration of the main program-loop, the last agent has an advantage, since it knows the position of all other agents at the end of the current iteration and thus can act based on more recent information. In a fight between two agents this would result in a serious disadvantage for the agent activated first. Though our camera agent should not get involved in fights or other forms of interaction with the environment, it is of advantage to use a coherent underlying agent architecture for all types of agents in a multi-agent system to simplify control-processes. Updates of agents can be simplified and one does not have to deal with special considerations concerning the different

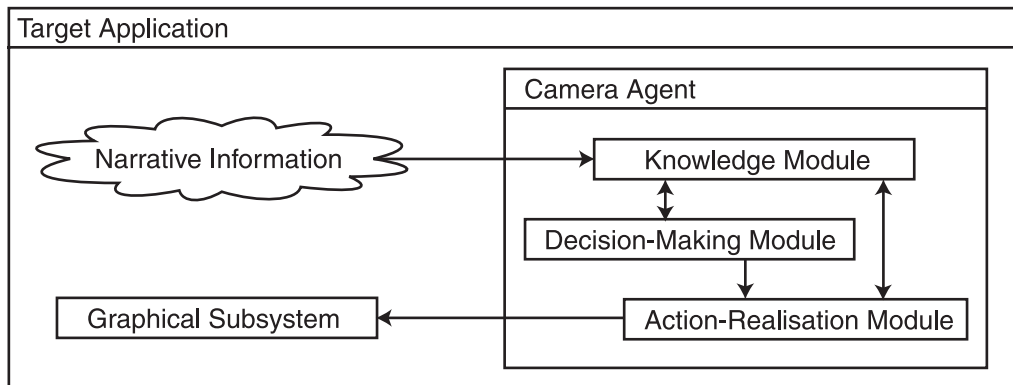


Figure 4.15: The three main modules of the camera agent and the flow of information.

agents.

We divide the process of acting as found in basic agent architectures into two subsystems, one for decision-making and one for acting. Additionally, instead of using a subsystem dealing with perception only, we use a module controlling the knowledge of the agent. ‘Perception’ is a further subsystem this knowledge module. This results in the general architecture as shown in Figure 4.15. The agent is structured into the three main modules *Knowledge*, *Decision-Making* and *Action-Realisation*. The arrows in the figure depict the flow of information between the different subsystems of the camera agent and the narrative target application. We will go into the details of the integration into narrative environments and the flow of control in the following sections. The solution to the above problem of up-to-date-ness is to activate and update the different modules as shown in the following pseudo-code:

```

LOOP
  FOR EACH Agent DO
    Agent->Knowledge->Update();
    Agent->DecisionMaking->Update();
  FOR EACH Agent DO
    Agent->ActionRealisation->Update();
ENDLOOP

```

Each submodule has a central update method, which is called during each iteration and which handles all subsequent functions. This of course assumes that a central structure has control about the agents and that the different agents do not have separated control threads. In a multi-threaded

environment the problem of consistency and fairness of available information has to be dealt with using much more complex mechanisms based on semaphores and other techniques for concurrency problems. But even in the case of multi-threading this distinction of decision-making and acting into separate modules provides better possibilities to assure fairness concerning up-to-date-ness of the individual agents. The problems of multi-threading are not examined further in this thesis. Most current applications dealing with agents or objects of some sense provide central control mechanisms to synchronise agent activity for example. This synchronisation can be performed by synchronising each of the three modules as shown above.

We will now give a short overview over each of the three subsystems and then go into their details.

4.4.1 Knowledge Subsystem

The knowledge module represents and encapsulates the agent's information about its environment. This information is not necessarily right or permanently up-to-date but represents what the agent believes to be right. This knowledge mainly consists of information about the camera agent's current state, narrative situations provided by the target application, a library of camera shots, and cinematographic knowledge based on our analysis of cinematographic concepts.

4.4.2 Decision-Making Subsystem

This module represents the central decision-making and commitment of the agent to an action. Based on the current knowledge the agent determines which narrative situation is currently most important to be presented to the audience, and what the available behavioural options for this situation are. The camera agent decides, whether or not certain camera shots are suited and considers additional aspects like interdependencies between shots and narrative events. The main question answered in this subsystem is: *What goals does the agent want to achieve?*

4.4.3 Action-Realisation Subsystem

After focusing on a narrative situation and on the respective behavioral options, this module tries to put one of these options into action. The camera agent proposes a shot for the given situation to the target application. If this

shot cannot be realised for some reasons⁴, it will examine alternative options. The main question answered here is: *How* is the agent going to achieve its goals?

We will now go into the details of each of these modules and describe their subcomponents.

4.5 Knowledge Subsystem

This subsystem is currently the most complex of the three modules, since it encapsulates all relevant information that can be considered as the ‘knowledge’ of the agent. This knowledge is realised as object-oriented subsystems implementing different aspects of scene- and shot-analysis as well as modules for the realisation of camera shots. Thus most of the agent’s main-components like camera behaviours and the narrative event analyser will be discussed in this section. The behaviour modules for example provide two methods. One is focused on decision-making, the other one on the realisation of the behaviour. Though we will present each of the subsystems completely in this section, some of the respective methods are invoked within the other two main modules of the agent.

The main components are⁵:

Narrative Event Analyser This is the submodule, which is storing and analysing narrative events. It is discussed in a separate section.

Camera Behaviours A library of the different shots available to the agent. Each element of the library contains information about the preconditions and the realisation of the associated shot. These behaviours are implemented as submodules and discussed in a separate section.

Current Behaviour The currently active shot.

Plane Of Action A data structure holding all necessary information and interfaces associated with the cinematographic concept of the ‘plane of action’ between the objects of a narrative event.

Shot Sizes A table of different shot sizes and the associated interpolation ratios between the different objects’ bounding-spheres and their respective on-screen sizes (Figure A.3, Appendix A). This data is based on [20], [29], and is used during the generation of shots.

⁴Like geometrical constraints.

⁵We will not go into detail here on additionally necessary submodules like ‘timers’.

Camera Parameters This data structure holds the camera position, target of the camera, the up-vector for orientation, and the field-of-view.

Two of the central and most important components of the agent, the narrative event analyser and the behaviour modules, are presented in the following two sections. They are part of the knowledge module since they represent and encapsulate the agent's cinematographic 'knowledge'. Both modules however have functions for decision-making and action-realisation to manipulate and apply the encapsulated knowledge. We will present both modules in this section, their respective manipulation-methods are called during the update of the decision and action module.

4.5.1 Narrative Event Analyser

The narrative event analyser is the main module for deciding about narrative events. It analyses for example, which events sent by the target application should be considered further, which events can be dropped, and which events should be visualised by the agent. The four main methods to interact with the analyser are the following:

Register Narrative Event This method represents the main interface between the target application and the camera agent⁶. By calling this method the application informs the agent about future or currently starting events.

Update This function updates the internal data structures of the analyser and is called by the knowledge module. For example past events, that are not active anymore and thus do not have to be considered further, are dropped.

Calculate Candidate Event In this method the analyser examines all currently known narrative events and decides based on criteria like event-coherence or the current shot length if the camera should switch to a new event or if it has to keep the current shot. This method is called during the decision-making process of the agent.

Activate Candidate Event This method is used to commit the agent to the narrative event chosen during the calculation of candidate events.

⁶We already used this method in our examples of narratives events.

The Calculation of Candidate Events

The calculation of candidate events represents the main routine of the narrative event analyser. Since it is one of the most important components of the decision-making process, we will present the used algorithm here in a simplified form omitting implementation specific details:

```

candidateEvent = NULL;
FOR EACH event in event-queue DO
  IF event is currently running OR
     event starts within n seconds THEN
    use neuron to compare the priority,
    coherence, and temporal information of event with
    the currently active event and candidateEvent;
    IF neuron fires THEN candidateEvent = event;

```

This is the first step of the algorithm. We trained a neuron with inputs for the priority difference, the coherence difference, and the temporal difference between events. Then we use the following differences as inputs for the neuron:

```

priorityInput = candidateEvent->priority - event->priority;
coherenceInput = candidateEvent->coherenceWith(currentEvent) -
                 event->coherenceWith(currentEvent);
IF candidateEvent starts earlier than event THEN
  temporalInput = 1.0;
ELSE
  temporalInput = 0.0;

```

If the neuron fires, the respective event becomes the new candidate event. The important cinematographic concept of *continuity style* is preserved by considering the coherence between events.

The second part of the algorithm decides, if the agent should switch to the new candidate event or if it should keep the currently active event. The algorithm considers the shot duration for the currently visualised event and decides, whether the camera can make a cut and switch to the new event or not. We compute a minimum shot duration based on the ‘Calmness’ parameter of the current event’s action. The currently used formula is:

```

durationFactor = (1.0 - currentEvent->action->calmness)/2.0;
minimumDuration = 1.0 + durationFactor * 3.0;

```

By this we get minimum shot lengths of one second for hectic actions like fights, and shot lengths of four seconds for very calm actions. The pseudo-code for deciding if the new candidate event should be kept as waiting candidate for a new shot is then as follows:

```

IF current shot duration < minimumDuration THEN
    candidateEvent = NULL;
ELSE IF current event is finished THEN keep candidateEvent;
ELSE use neuron to compare the priority and
    coherence of the candidateEvent and
    the currently active event;
    IF neuron fires THEN candidateEvent = NULL;

```

While the meaning of the first two conditions should be clear, the third one is used to decide, if the candidate event is important enough to drop the currently active event, though it did not terminate yet. We use a simple, trained neuron here to make this decision.

4.5.2 Camera Behaviours

‘Behaviours’ encapsulate all data structures, routines for decision-making, and methods for action-realisation needed by the camera agent to generate shots. Each behaviour represents a specific camera shot. Different shots can be configured in an initialisation-file, which is processed at the start-up of the agent. Behaviours are divided into two main components: one for the computation of a motivation used for decision-making, and one for the realisation of a behaviour.

Motivation Given a narrative event, a numeric value between 0.0 and 1.0 is calculated by a neural decider associated with this behaviour. This value is interpreted as the ‘motivation’ to apply a behaviour or shot to the tested narrative event. By querying all behaviours, the agent can create a priority list of matching behaviours and use this list for further decision-making.

Realisation This method computes all steps to create the associated behaviour. The current narrative event is analysed in detail, and based on the objects’ positions, the plane of action, action directions, etc., the camera position and orientation is calculated.

Motivation

The current motivation for a given narrative event is computed using a hybrid approach of rules and a neural decider. Every behaviour is designed to correspond to a specific action defined by its scenery variant, action type, and the manner of interaction. Additionally, shots can be assigned to a *shot class*. Interdependencies between shots can be evaluated by comparing the class of the predecessor shot with the current shot class. Given a list of positive and negative predecessor shot classes, the motivation value for a shot is lowered, if the predecessor shot was in the set of negative predecessor shots, and is raised if it was in the list of positive predecessors. Fixed sequences of shots for dialogues or special effects can be generated similar to other approaches based on state-machines (e.g. [21]). By including the class name of the current shot into its own list of negative predecessors, we can also omit the repetitive usage of the same shot.

For every shot the following parameters can be set within an initialisation file describing the shot library available to our camera agent. They are then evaluated during the computation of the motivation:

Scenery Variant This can be *Dialogue*, *Action* or *Dialogue & Action*.

Action Type *Physical*, *Mental* or *Predicate*.

Neuron The name of a trained neuron for action classification.

Shot Class The name of the associated shot class.

Positive Pred. Shot Classes A list of predecessor shot classes increasing the motivation for the current shot.

Negative Pred. Shot Classes Predecessor shot classes decreasing the motivation.

Number Of Necessary Objects The minimum number of objects needed for the realisation of this shot.

The input of the algorithm computing the motivation is a narrative event. It returns a value expressing how suited the shot is for the given event. The neuron evaluates the manner of interaction provided with the action of the narrative event and the event coherence between the currently visualised event and the new candidate event. By explicitly considering the coherence between events, we can configure shots that are suited for the establishment of new scenes like master-two-shots as well as shots for already established

scenes like close-ups. See Appendix C for some examples of shot configurations.

The motivation is computed as follows:

```
IF action scenery variant != shot scenery variant THEN
  RETURN motivation = 0.0;
ELSE IF action type != shot action type THEN
  RETURN motivation = 0.0;
```

```
motivation = Neural activation (0.0 to 1.0) of the neuron
associated with the currently tested shot for the given
interaction manner types and event coherence;
```

```
IF predecessor shot IS a negative predecessor THEN
  RETURN motivation = motivation * 0.8;
ELSE IF predecessor shot IS a positive predecessor THEN
  RETURN motivation = motivation * 1.2;
ELSE RETURN motivation;
```

We additionally add a minimal random disturbance to the motivation returned by the neuron to avoid shots with equal values. Otherwise, this would result in shots tested first being preferred.

Realisation

One of the main goals was to provide a simple but effective interface to configure camera shots. We used two approaches based on spherical coordinate systems and object on-screen positions. Since the involved geometrical computations are complex and our agent even provides basic techniques for keyframe-animation to create camera movements, we will not go into every algorithmic detail. Appendix A discusses some of the important geometrical formulae and algorithms that were developed and used for the implementation.

The realisation-procedure of a behaviour analyses the geometrical information of the current narrative event, and computes camera parameters like position, orientation and the field-of-view. The following parameters are used to describe shots and can be set in an initialisation-file (see Appendix C):

Focused Object A number defining the focused object. This parameter can also be set to ‘all’ to show all participating objects of a narrative event. This way we can describe if the camera should focus on a certain object for close-up shots, or if it should show every object of an event like in establishing shots .



Figure 4.16: An example to generate an over-the-shoulder shot for dialogues using virtual screen positions. The left part of the image shows both characters standing in 3D-space, talking to each other. The camera and the screen on the right show the approximate camera position and the resulting picture by specifying on-screen coordinates (illustrated by the red and green lines).

Shot Size The size of the focused object based on Figure A.3. The interpolation ratios between the object's bounding-sphere, the point-of-interest bounding-sphere, and their respective on-screen sizes are used to compute the distance between the camera and the focused object.

Minimum Shot Length The minimum duration of the shot. This is useful during certain camera movements, which have to last a minimum time to be completed. If the camera is circling around a character's head to emphasise confusion for instance, this shot should at least last a number of seconds without being interrupted.

Shot Definition A shot can be defined by spherical coordinates or object on-screen positions.

Object on-screen position This shot description is defined by target positions for the objects on the screen. One can define the final on-screen position of objects by providing object coordinates (x_P, y_P) for the primary, and (x_S, y_S) for the secondary object on a virtual screen ranging from -1.0 to 1.0 on the x - and y -axis. This way, one has good control over shots, where it is necessary to present an object in a specific relation to another object. The

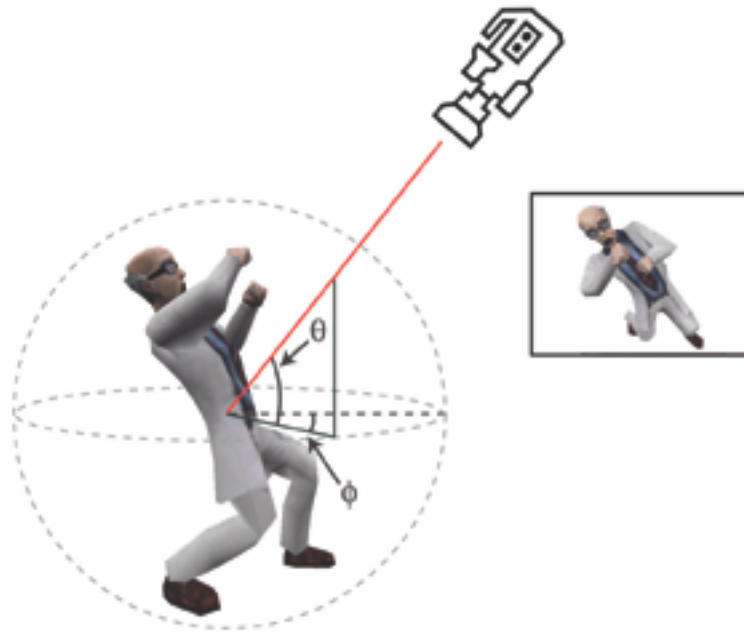


Figure 4.17: Shot definition using spherical coordinates. The camera position is defined by two angles and a distance.

current version of the algorithm is able to compute a camera position and orientation given two reference objects, their respective 3D-coordinates, and on-screen positions. Figure 4.16 shows an example to create an over-the-shoulder shot for a dialogue using this technique.

Spherical coordinates The camera position is defined by spherical coordinates in the local coordinate system of the focused object. The distance of the camera to the object is computed by the given shot size, the elevation by the vertical angle θ , and the azimuth by the horizontal angle ϕ . Figure 4.17 shows an example for the realisation a full-shot from a high angle. Additionally, one can provide an offset of the focused object on the screen similar to the above method. This way, one can generate a kind of visual rhythm as described in our cinematographic analysis. Static shots for instance can be created by putting the object in the center of the screen. More dynamic shots, tension, or gravity for instance are emphasised by positioning the object towards the respective borders of the screen.

Appendix A discusses the involved algorithms and formulae in more detail.

FOV This parameter defines a change in the field-of-view during a shot. For instance, interesting dramaturgical effects can be produced by constantly narrowing the field-of-view, while filming the astonished facial expression of a character.

Damping Factor A damping factor used for interpolation between the last camera position and the currently calculated goal position. This parameter can be used in order to create smoother camera movements, or to simulate inertia.

Key Frames Key frame times used during keyframe-animation. For each of the parameters described so far, a list of values can be given, which are then interpolated linearly using these key frame times. The current solution supports linear interpolation only. More sophisticated methods based on splines for instance have to be considered for further development. By providing key frame times for each shot parameter in combination with the damping factor, we are already able to generate complex camera movements (see Figures 5.3, B.3, B.4).

During the computation of the camera position and orientation, every shot considers the rules associated with the plane of action. If a shot based on spherical coordinates results in a camera position, which lies in the wrong half-space of the current plane of action for example, and which is not reached by a continuous, smooth movement, the camera is repositioned by inverting the horizontal angle ϕ . In the case of on-screen coordinates, x_P and x_S are multiplied by -1.

Both methods for shot description already provide the possibilities to realise a broad range of the most common shot types. But obviously this list of mathematical tools to describe shots can be extended endlessly. For example, it should be possible for a target application to request certain shots that are based on fixed way-points within the current scenery. This bears however the necessity to include cinematographic knowledge into the target application itself. Our goal was to focus on a consistent basis for the dramaturgical emphasis of narratives by an autonomously deciding camera system. The above extensions are important as well as geometrical considerations, but have to be part of future research. We will come back to this topic during the discussion of this work.

At the beginning of this work, we also investigated a completely different approach to create shots. It was based on the idea, that one could provide

only a few basic shot types, which are then modified by the agent. Instead of providing a library of fixed shots, one could for example define a standard shot for neutral close-ups. Depending on the action classification of a narrative event, the camera agent could automatically lower the camera position, if the focused character was evil for instance. This approach however would have constrained the number and diversity of shots too much. Additionally, this idea is difficult to transfer from static shots to arbitrary camera movements, because it would be difficult to define a ‘standard’ camera movement and the appropriate modifiers. Thus we decided to use the classification of actions only for decision-making, and not for the realisation of shots.

4.5.3 Update

During each update iteration of the knowledge module the following steps are computed:

1. The update method of the narrative event analyzer is called and updates internal data structures.
2. Based on the currently activated narrative event and the respective objects, the plane of action is calculated. If the event has exactly one associated object, the plane of action is given by the object position and its action direction. For more objects it is defined by the primary and secondary object position.

4.6 Decision-Making Subsystem

This module consists mainly of a central update routine and handles all higher-order decision-making related to narrative events and camera shots for the current situation. It invokes the respective functions of the narrative event analyser and the camera behaviours in order to calculate a motivation for each shot.

4.6.1 Update

One call of the update-function of the decision-making subsystem processes the following steps:

1. Check, if the minimum shot duration for the currently active shot has been reached yet. If not, the agent does not chose a new shot or a new narrative event for visualisation. This shot duration should only

be given for shots, where a minimum duration is explicitly needed to create the intended visual impression.

2. The narrative event analyser is called to compute a candidate event for the current time. If no candidate event for a new shot can be found because of missing events or other constraints like shot duration, event coherence, etc., the agent keeps the current event.
3. If a new candidate event was found, the agent iterates through all shots in its shot library and computes their respective motivation to visualise the given candidate event. A priority list is generated, which is storing all matching shots and their motivation values. This list is then evaluated during action-realisation module.
4. If one or more matching shots could be found for the candidate event, the currently visualised event is dropped, and the new candidate event activated. Additionally, the update routine of the knowledge subsystem is called to recalculate data like the plane of action.

4.7 Action-Realisation Subsystem

This subsystem puts shots into action. It evaluates the priority list of shots created during decision-making, and tries to realise one of the best matching shots by calling their respective realisation-function.

4.7.1 Update

The main steps of the central update function of this subsystem:

1. If the decision-making module found a new narrative event and matching shots, this module tests each shot in the order given by the computed motivations, if it can be used for visualisation. Even if a shot was chosen as appropriate for the new event, it could still fail to visualise it because of geometrical constraints. If a shot is defined by distant on-screen coordinates of two objects for instance, which are located very close to each other within the environment, and a very great camera distance is chosen, there could be no camera position fulfilling these constraints and the agent has to choose an alternative shot. This testing is done by calling the realisation routine of every shot. The first shot from the priority list, which returns a proper camera position and orientation, is then used as the new shot type for the current event.

2. If no new event was found during decision-making, the current shot's realisation function is called in order to check, if the current shot can still satisfy all geometrical constraints like on-screen positions, and to update the camera position and orientation. This can be necessary, when objects move and the camera position has to be updated to keep them in the field of view, or during camera movements defined by keyframe-animation.

Chapter 5

The Integration into Half-Life

“Do you know who ate all the donuts?”

– Black Mesa Scientist

This chapter presents the integration of the camera agent into the computer-game Half-Life [46]. It describes the motivation for choosing Half-Life as the test-environment, shows the main steps necessary for the integration, and gives an evaluation of the results.

5.1 Motivation

The game *Half-Life* was published in November 1998, and is considered to be one of the first games combining the elements of pure 3D action shooters with storytelling. It is the story of scientist Gordon Freeman, research associate in the Anomalous Materials Laboratory of the top secret Black Mesa Federal Research Facility. While experimenting with a new crystalline specimen a portal to an alien world is opened, transforming reality into a nightmare filled with hordes of monsters. So far the game is very similar to other games of this genre. The two really important differences to other games making Half-Life a good test-environment for our camera agent are the following:

1. Half-Life has various, interactive story-elements like dialogues, intermissions, and action scenes embedded into the game-play.
2. The complete C++ source code of the game was published, which enabled us to integrate our prototype.

Until the publishing of Half-Life, most action games featured a kind of motivating background story, while the game itself was just another standard

action shooter without any real importance left to the underlying story. In such games, the player is not aware of the story anymore, but concentrates solely on the action component. In Half-Life, the player experiences conversations between other non-player characters like scientist and soldiers, or is introduced into new situations by intermissions. These elements, which are advancing the progress of the story, are integrated into the interactive gameplay and not realised by non-interactive cut-scenes like pre-rendered videos used in other games.

The reason why our camera agent can change and actively contribute to the storytelling in Half-Life is the fact, that the whole game in its original form is based on a first-person camera perspective. First-person views emphasise the immersion of the player into the game world, and the personification with the controlled game character. In addition, and probably the main reason for action shooters like Half-Life, the first-person view is actually a necessary precondition in order to be able to play fast-paced action scenes like gunfights, where it is of importance to aim precisely at targets. On the other hand, this view obviously restricts the possibilities of cinematographic storytelling, since the camera is fixed to the view from the controlled game-character's eyes. A visual emphasis of emotional content like moods or relationships between characters cannot be created. By integrating our camera agent, and by replacing the first-person view, this dramaturgical information within the game can be emphasised, and the narrative aspect of Half-Life is supported by pulling the story into the foreground.

The second reason for choosing Half-Life is the fact, that it is the only commercially released game featuring at least some elements of interesting storytelling, of which the whole C++ source code was published. While there are other game sources available (e.g. [22]), Half-Life is the only game providing enough narrative content like dialogues, intermissions, and action scenes in order to use at least some of the capabilities of our agent. One could criticise, that a pure adventure game without this strong focus on fights would have provided more possibilities to prove our concepts. But since there were no commercial game sources or other reliable, or sophisticated enough game projects available providing the necessary 3D-based environments and storytelling, Half-Life was the best choice available. Additionally, Half-Life's action scenes in connection with dialogues and intermissions already provide a diverse number of different test-scenarios for the camera agent.

Since the game Half-Life and its source code has been published, a very large community of people building modifications of the original game exists, and still keeps growing. There are already some modifications featuring different, independent camera views. The best known and most sophisticated one is HLTV [34] for watching online-competitions between players. However,

these modifications are primarily focused on geometrical issues of camera control.

While there were a lot of reasons for using Half-Life, there were also a lot of problems we had to solve during the integration. First of all, Half-Life consists of approximately 120,000 lines of undocumented, and uncommented source code. Thus the integration of the agent was to a large amount understanding and reengineering of the code. The only help available were a few online coding-tutorials introducing partial aspects of the game-engine. Such tutorials as well as the software development kit and the Half-Life source code can be found at [47].

The second problem was the fact, that a lot of the available data within the game-engine is too inconsistent for our camera agent. For example, the agent needs the exact geometrical properties like objects-sizes or eye-positions of game characters and their respective motion directions in order to be able to compute a correct camera position. But it turned out, that Half-Life assumes the same size for almost all characters in the game, be it standing or sitting humans, or even large monsters. In addition, due to the animation system used for objects and characters, the actual midpoint of these objects can only be accessed at the first frame of a model's cyclic animation-loop. Thus the midpoint of moving objects 'jumps' forward instead of moving on continuous paths. This makes correct camera placement a difficult task. We corrected some of these problems within the Half-Life engine. The camera agent itself was not modified during the integration.

Last but not least, accessing narrative relevant information was not an easy task. Accordingly, also because of the missing documentation, it was often difficult to extract the important narrative elements like intermissions. Thus we focused on three different aspects of the game: dialogues, fights, and the first intermission, when the portal to an alien world is opened at the beginning of the game.

5.2 Sending Narrative Events

The behaviour of most objects, or characters in Half-Life is implemented using state-machines. For instance, characters have different states for attacking, talking to characters, or doing some other action. The game-engine is based on a client-server architecture and is split into two software libraries. The server-side handles the game logics, and the behaviour and interaction of objects. The client-side manages the visualisation and the user front-end. The communication of narrative events from Half-Life to the camera agent is depicted in Figure 5.1. The first step is to send information about specific

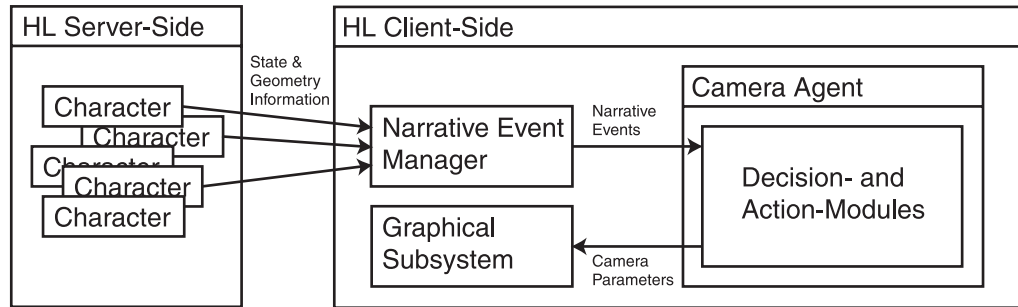


Figure 5.1: Architecture and flow of information between the integrated camera agent and the two main components of Half-Life.

state-changes of an object from the servers-side to the client, into which the camera agent is integrated. On the client-side, a narrative event manager receives such information about objects and creates the according narrative events for the agent.

If a character starts to speak a predefined sentence for example, we send a message using the internal message system of Half-Life from the game-server to the client encoding the following properties:

- Type of Half-Life action (here: ‘Speaking’)
- Duration of the sentence
- Identification of the character
- Character center position
- Head position (representing the point-of-interest)
- Orientation (representing the action-direction)
- Size of the character

Similar information is sent for other types of actions or objects. If an action is addressed to another object like a character during a dialogue, or a target during a fight, we also send the respective information about these objects.

The narrative event manager on the client-side receives and analyses such messages, and transforms them into narrative events for the camera agent.

For example, a neutral dialogue between two characters is represented as follows:

```
Action->SetSceneryType("Dialogue");
Action->SetActionType("Physical");

Object1->SetObject(character center, character size);
Object1->SetPointOfInterest(head position, head size);
Object1->SetActionDirection(character orientation);
Object1->SetObjectID(Half-Life character id);

Object2->...

NarrativeEvent->SetNarrativePriority(0.5);
NarrativeEvent->SetStartTime(current time);
NarrativeEvent->SetStopTime(current time + duration);
NarrativeEvent->SetAction(Action);
NarrativeEvent->AddObject(Object1);
NarrativeEvent->AddObject(Object2);

CameraAgent->RegisterNewNarrativeEvent(NarrativeEvent);
```

A fast attack of a dangerous enemy is transformed into the following narrative event:

```
Action->SetSceneryType("Action");
Action->SetActionType("Physical");
Action->SetInteractionMannerType("Calmness",1.0);
Action->SetInteractionMannerType("Object Might",1.0);

Object1->SetObject(enemy center, enemy size);
Object1->SetPointOfInterest(weapon position, weapon size);
Object1->SetActionDirection(weapon direction);
Object1->SetObjectID(Half-Life enemy id);

Object2->SetObject(target object center, target size);
Object2->SetPointOfInterest(hit position, wound size);
Object2->SetActionDirection(front facing direction of wound);
Object2->SetObjectID(Half-Life target object id);
```

```

NarrativeEvent->SetNarrativePriority(1.0);
NarrativeEvent->SetStartTime(current time);
NarrativeEvent->SetStopTime(current time + duration);
NarrativeEvent->SetAction(Action);
NarrativeEvent->AddObject(Object1);
NarrativeEvent->AddObject(Object2);

CameraAgent->RegisterNewNarrativeEvent(NarrativeEvent);

```

During every update-call, the narrative event manager additionally queries the object-ids of the currently visualised narrative event from the camera agent, and updates their respective geometrical information like position and orientation. This is important for moving characters for instance. If necessary the camera agent can update its position to fulfill constraints of the current shot like object on-screen positions.

5.3 Setting the Camera

The Half-Life client has a central routine within its graphical subsystem, which is called during every visualisation step. It evaluates the current player position and generates the first-person view. We modified this function in order to call the update-routines of our camera agent. The agent computes a camera position and orientation. These information are written into the appropriate data-structures within the Half-Life graphics-engine. Since our agent does not analyse geometrical properties of the surrounding environment, we added a simple occlusion check to prevent intrusion of the camera into solid bodies like walls.

5.4 Shot Library

We built a shot library consisting of fifteen different shots and six different neurons for action classification for the Half-Life test-scenario. The different neurons were trained to identify the following action classifications:

Neutral actions with a low event coherence This neuron type is used for the definition of master two shots for dialogues for example, where a low event coherence is a necessary precondition.

Neutral actions with a high event coherence This neuron is used for all other dialogue shots like angular singles, close-ups, etc.

Hectic actions This neuron is used for the identification of hectic action scenes like fights.

Lonely, isolated actions We use this neuron to detect intermissions, where the main character is transported into the foreign alien world and surrounded by monsters.

Mighty actions This neuron is also used for a shot during one of the intermissions, where the power of the exploding reactor core is shown.

Neutral actions For some scenes of the intermission we need a bridging shot for completely dark scenes. We decided to use a neutral neuron here.

Note that these neurons are only one part of the actual decision-making process regarding the choice of alternative shots for a given situation. The other criteria are given by the respective scenery variant and action type.

The agent's shot library for Half-Life consisted of the following shot types:

Follow-up camera A long shot from behind the character along its view-direction. This shot is used when the player is simply moving within the environment. It is activated by neutral, physical actions with a high event coherence. Figure 5.2 b) shows one picture of the follow-up camera on the upper left. The player-character is wearing an orange suit.

Dialogue shots implementing the triangle system We used five basic camera setups for dialogues as introduced with the triangle system in Chapter 2. These shots are configured for neutral, physical dialogues, and use the respective neurons introduced above to separate dialogue actions by their event coherence from each other. See Figures 5.2, B.1, B.2.

Intermission shots We implemented four different shots for the intermissions within Half-Life. They are activated by mental actions like the main character feeling fear and loneliness during its transportation into the alien world. See Figures 5.3, B.3, B.4.

Attack shots Four different views of an attacking object and its target are implemented for the visualisation of fights. Additionally we created a shot generating an effect known from the movie 'The Matrix' [48], where the progress of time is stopped and the camera circles around the focused object. Attack shots are activated by hectic, physical actions. See Figures 5.4, B.5, B.6.



Figure 5.2: A dialogue between four characters. a) Original first-person view. b) Camera agent: the spectator is guided through the conversation by appropriate camera shots.

5.5 Discussion

Though it is difficult to describe the interactive experience of playing *Half-Life* using the camera agent, Figures 5.2, 5.3, and 5.4 show comparisons of three situations¹ in the game, taken from the original first-person view and from the camera agent.

The first example in Figure 5.2 shows, how the camera agent can ‘guide’ the player through the narrative. In the first-person view (Figure 5.2 a)), which is used permanently within the original game, the player can turn around, run away, or do whatever he likes. The narrative importance of the situation can be completely destroyed. By guiding the player through the dialogue using appropriate camera shots, the story has control over how the player experiences important narrative elements (Figure 5.2 b)).

Figure 5.3 is another example of how the camera agent changes the visual impact of a situation. While the jump through a dimensional breach into an alien world is rather unimpressive using a simple first-person view (Figure 5.3 a)), the situation becomes much more terrifying by having the camera circling around the player, slowly revealing the alien monsters (Figure 5.3 b)).

¹Because of the interactive, unpredictable nature of the game, it is not possible to produce exactly the same setting twice. Thus, character-positions may vary from each other in the different views.

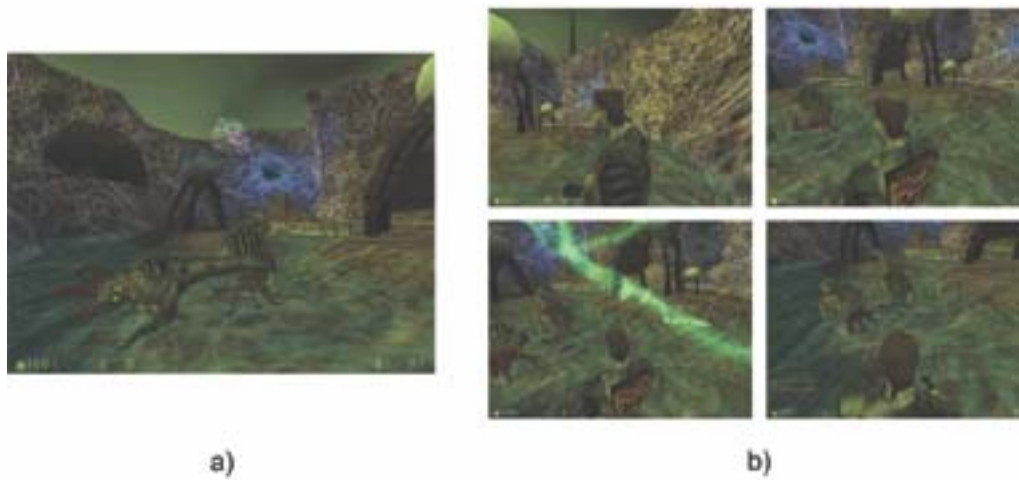


Figure 5.3: A scene from one of the intermissions. The player is transported through a dimensional breach and is surrounded by aliens. a) Original first-person view. b) Camera agent: a sequence of shots. The camera is circling around the player, and the terrifying impression of the intermission is emphasised in contrast to a).



Figure 5.4: A fight between the player and some assassins. a) Original first-person view. b) Camera agent: four shots of attacking enemies (the player being attacked is wearing the orange suit).

In Figure 5.4, one can see the difference between fighting in standard first-person view (Figure 5.4 a)) and using the camera agent (Figure 5.4 b)). The fight becomes much more interesting and is intensified by switching between the different characters, using fast cuts and unusual camera positions.

Note, that the fight-scenes of the game are hardly playable with the agent being activated, because it is not possible to aim exactly, or one does not see the player-character at all. But as mentioned before, the goal of the camera agent is not to make first-person shooters more playable, but to show the narrative capabilities of an autonomous camera following cinematographic principles. The above scenes already show, that the agent can strongly emphasise narrative content by choosing appropriate camera shots.

For example, an interesting scenario for games like Half-Life would be to link several computers by a network, some running the game in standard first-person view, and others running the game using the camera agent. While people are playing on the computers with the first-person version of the game, an audience watching the pictures of the camera agent on the networked computers could experience the same game like a movie. Similar concepts are already being used to watch online competitions [34], but without the explicit consideration of the narrative meaning of pictures. This concept could easily be transferred to games with more interesting narrative content and without this strong focus on the action component. Hybrid approaches like first-person views for action scenes in combination with the camera agent for the ‘narrative’ parts of a game could also be realised easily. However we did not chose this approach here, because the fight scenes were an interesting test-field for the camera agent.

Though the narrative capabilities of Half-Life are limited in comparison to more recent games or other virtual reality based applications, the integration of the camera agent into this game clearly showed interesting results of how the same content can be experienced in completely different ways by changing the visualisation. Appendix B shows some more impressions of the camera agent’s work.

Chapter 6

Discussion

“It’s a little too obscure I think, sorry.”

– Community Site Manager,
`planethalflife.com`

This chapter presents a critical assessment of this work, discusses open problems, and proposes fields of future research.

6.1 Critical Assessment

The work on this thesis consisted of three main parts. The first goal was to analyse and formalise cinematographic concepts in order to transfer them to the domain of an autonomous camera system and interactive narratives. Though we were able to extract at least partial formalisations, it became apparent, that the visual interpretation of narratives is a highly complex field, that cannot be investigated in detail without the additional consideration of psychological or narrative theory. The finding of a consistent, basic formalism of cinematographic means of expression already turned out to be very difficult. The constant breaking of established cinematographic rules by progressive film-makers also shows, that the theoretical basis of film-making is still in a flux. Furthermore, the transfer of classical cinematographic concepts to interactive concepts of storytelling involves the necessity for very different approaches, and often even simple concepts cannot be transferred at all.

The second goal of this thesis, a working implementation of an agent-based camera system implementing the results of our cinematographic analysis turned out to be a rather difficult task because of the diversity of the involved fields, from architectural issues over ai-related questions like the neural deciders to geometrical solutions for camera handling in 3D-based

environments. Though the implemented prototype already shows some satisfying and interesting results, a lot of the concepts are still only basic ideas for solving the involved problems. For example, our neuron-based approach could become problematic if the number of relevant parameters for actions grows due to extensions considering light or geometrical constraints within a scenery.

The last point, which proved to be difficult and extremely time-consuming without direct relation to this work was the problem of inconsistent data within the Half-Life test-environment. While the actual integration of the agent into the game needed only minor efforts, learning to program the Half-Life source code without a proper documentation and correcting errors and inconsistencies relevant to the agent took a lot of time.

6.2 Future Work

We will now present directions for future research based on our work. First, a more detailed investigation of cinematography and related domains like psychological or narrative theory is necessary to understand the complex interplay between these fields. This work tackled only a few basic concepts found in cinematographic literature. The problem of finding a theory for dealing with interactive narratives and the transfer of cinematographic knowledge to this domain still remains unsolved. The fact, that real-world constraints do not apply to virtual narratives provides the possibility and necessity to introduce and evaluate new stylistic devices in addition to classical concepts. New approaches for the representation of interactive, non-linear stories have to be investigated. For example, our chosen representation of important story-elements using narrative events can appropriately represent the progress of a story in our context of highly dynamic and interactive applications without the necessity to plan ahead much. But for the inclusion of a further, more complex scene analysis like the consideration of the lighting situation and geometrical constraint solving for occlusion problems, it will be necessary to extend the representation of a story and of the environment. The basic means of expression we found need to be extended to integrate additional information. For instance, one could want the camera to explicitly ‘hide’ an action like approaching dangers from the audience, and show only an involved character’s reaction to increase the tension. A better representation could allow more intelligent and complex planning abilities of the camera agent. The currently used methods for story-representation and decision-making are not suited for planning ahead shots, transitions, or for making assumptions on the future progress of the story. The learning mechanism for narrative events

based on single neurons has to be extended to layered neural nets or other learning techniques like decision trees to support arbitrary action classifications. A more intuitive trainer based on a real 3D-environment could be used to train the agent in using different shot types. Training the camera agent with examples instead of coding behaviours would facilitate the usage of such a system by non-researches or non-programmers. Also, the possibilities of describing camera shots are still limited. We implemented two methods, which already suffice to describe a variety of shot types, and we also supplied basic tools to create different camera movements, but additional geometrical tools for shot description like spline-based approaches would allow more complex camera movements.

The interaction and exchange of information between the camera agent and the application is another field for future investigation. In this work we tried to separate camera specific knowledge and application specific knowledge from each other completely. The only form of communication are narrative events and basic camera parameters. But the application has no information about available shot types for instance, though it could sometimes be necessary to ‘request’ specific shots from the agent. Also, the agent needs additional information on the capabilities of the graphical subsystem to decide, if a certain shot, a change in the field-of-view, changing the progress of time, or transitions like blending are possible.

6.3 Summary

In this thesis, we developed an agent-based, autonomous camera system following cinematographic concepts with a strong focus on the demands of interactive narratives and games in real-time environments. We analysed and partially formalised the narrative capabilities and basic means of expression of cameras. In particular, we concentrated on the dramaturgical aspects of camera handling instead of geometrical constraint satisfaction. Based on these findings we presented an architecture and implementation of a camera agent following these cinematographic concepts. To provide an example of the practicality of our system we integrated the camera agent into the computer-game Half-Life. Finally we discussed this work and presented possible fields for future work.

Appendix A

Geometrical Issues

The following four sections introduce some of the geometrical concepts used in order to implement important aspects of 3D camera handling. All geometrical computations assume a standard, 3D-graphical environment based on a so called right-handed cartesian coordinate system (Figure A.1). Please see [1], [15] for a detailed introduction into the principles of computer graphics and the underlying mathematics. Notation: We use capital letters with an arrow \vec{V} for 3D-vectors. Normalised \hat{V} vectors are written with a hat. Matrices are depicted by bold letters **M**.

A.1 Camera Distance

The algorithm computing the necessary distance between the camera and the focused object to achieve the goal shot size is based on the following equations. Figure A.2 explains the used notation, Figure A.3 gives the interpolation values for the requested shot size.

$$\begin{aligned} O_{Distance} &= O_{Radius} \left(\cos(\alpha) + \frac{\sin(\alpha)}{\tan\left(\frac{\psi}{2}\right)} \right) & (A.1) \\ \text{with } O_{Radius} &= I_{BodyPoI} BodyRadius + \\ & \quad (1 - I_{BodyPoI}) POIRadius \\ \psi &= \phi_{min} \cdot O_{FOVRatio} \\ \alpha &= \frac{\pi - \psi}{2} \end{aligned}$$

Derivation (see Figure A.4): Given the current field-of-view (FOV) ϕ_y in y -direction and the x/y -screen aspect ratio *AspectRatio*, the minimal FOV

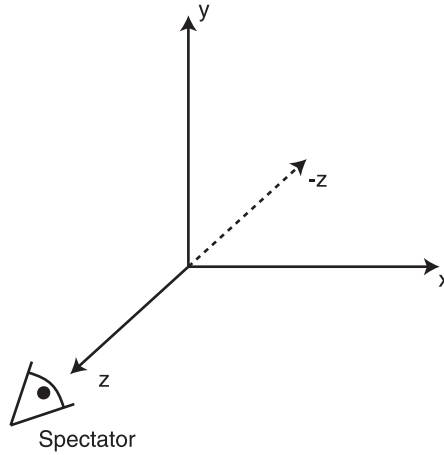


Figure A.1: The underlying right-handed cartesian coordinate system. A spectator looks down the negative z-Axis.

$O_{Distance}$	The necessary distance between the camera and the focused object to fulfill geometrical constraints. This is the result of our equation
$Body_{Radius}$	Radius of the object's bounding-sphere
PoI_{Radius}	Radius of the bounding-sphere of the point of interest
O_{Radius}	Interpolated object radius
$O_{FOV_{Ratio}}$	Ratio of the field-of-view (FOV) occupied by the object. Depends on shot sizes given in Figure A.3
ψ	FOV covered by the object
ϕ_y	FOV in screen y direction
ϕ_x	FOV in screen x direction
ϕ_{min}	Minimal FOV
$VS_{Distance}$	Distance between the camera and a virtual screen ranging from -1 to 1, covering view-frustrum
O_{VSS}	Half size of the focused object on the virtual screen
$T_{Distance}$	The distance between camera and the perpendicular dropped from the tangent of the bounding-sphere

Figure A.2: Notation for camera distance equations.

Shot Name	$I_{BodyPoI}$	$O_{FOVRatio}$
Extreme Close-Up	0.0	1.5
Close-Up	0.0	1.0
Close Shot	0.125	1.0
Medium Close Shot	0.235	1.0
Medium Shot	0.3125	1.0
Medium Full Shot	0.625	1.0
Full Shot	1.0	1.0
Long Shot	1.0	0.5
Extreme Long Shot	1.0	0.25

Figure A.3: Shot sizes $O_{FOVRatio}$ and Body-PoI interpolation values $I_{BodyPoI}$.

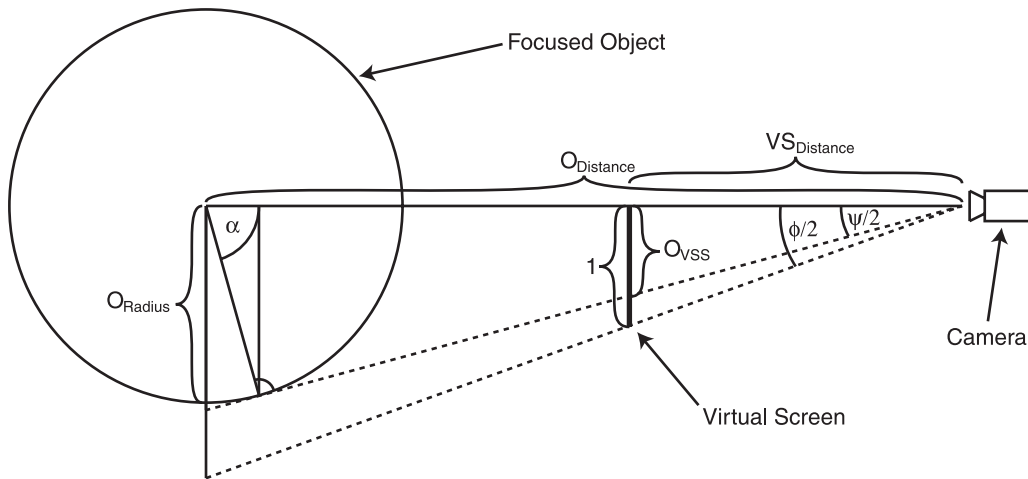


Figure A.4: Geometrical relations for the computation of the necessary distance between the camera and an object to achieve a specific object size on the screen.

ϕ_{min} has to be computed to ensure that the object bounding sphere covers $O_{FOVRatio} \cdot \phi_{min}$ independently of the current screen size. If requesting a ‘Full Shot’ of an object for instance, one wants to have the full object on screen, no matter if ones uses wide screens or a standard computer screen. If $AspectRatio < 1$ the minimal FOV found in x -direction is given by:

$$\phi_{min} = 2 \arctan \left(AspectRatio \cdot \tan \left(\frac{\phi_y}{2} \right) \right)$$

On a virtual screen ranging from -1 to 1 in x and y direction, half the object’s virtual screen size is

$$O_{VSS} = \frac{\tan \left(\frac{\psi}{2} \right)}{\tan \left(\frac{\phi_{min}}{2} \right)}$$

and the distance of our virtual screen is given by

$$VS_{Distance} = \cot \left(\frac{\phi_{min}}{2} \right)$$

To get the desired distance $O_{Distance}$ we first compute the distance $T_{Distance}$:

$$T_{Distance} = O_{Radius} \cdot \sin(\alpha) \cdot \frac{VS_{Distance}}{O_{VSS}}$$

$O_{Distance}$ then is (see equation (A.1)):

$$O_{Distance} = O_{Radius} \cdot \cos(\alpha) + T_{Distance}$$

A.2 On-Screen Position

This section presents the formulae used in order to compute camera shots based on the *ObjectScreenPosition* shot definition for two objects . It is based on a slightly modified version of J. Blinn’s publication ‘Where Am I? What Am I Looking At?’ [7].

Given the positions of the objects’ bounding-spheres, their respective on-screen positions, and the target size of the primary object, we first interpolate the focal point \vec{P}_P for the primary object similar to the above interpolation of

Universe space	Camera space	Definition
\hat{U}	\hat{V}	Up vector
\hat{T}	\hat{L}	View direction
\hat{A}	\hat{H}	Camera to primary object
\vec{Q}	\vec{R}	Primary object to secondary object
\vec{D}	\vec{E}	Primary object to camera

Figure A.5: Used vectors and their meaning.

O_{Radius} by using the values found Figure A.3. The position of the secondary object is \vec{P}_S . We then compute the necessary distance $O_{Distance}$ of the camera to the focused object to achieve the goal shot size (see Appendix A.1).

The next step is to compute the 3D camera position and orientation, given the target on-screen positions of two objects, their respective 3D world coordinates, and the camera distance to the focused object. We will present the necessary steps of the algorithm computing the camera position here. See Blinn's paper for the complete derivations. Figure A.5 denotes the used vector definitions.

We have given the on-screen position of the primary, focused object, and the secondary object by (x_P, y_P) and (x_S, y_S) , and the distance $d = O_{Distance}$. Vectors \vec{E}' and \vec{H}' are then:

$$\vec{E}' = \left(x_P, y_P, -\cot\left(\frac{\phi_y}{2}\right) \right)^T \quad \vec{H}' = \left(x_S, y_S, -\cot\left(\frac{\phi_y}{2}\right) \right)^T$$

The vector from the primary object to the camera position in local camera coordinate space is then:

$$\vec{E} = d \frac{\vec{E}'}{|\vec{E}'|}$$

We also need

$$\hat{H} = \frac{\vec{H}'}{|\vec{H}'|} \quad \text{and} \quad \vec{Q} = \vec{P}_S - \vec{P}_P$$

By applying the law of cosines $q^2 = d^2 + a^2 - 2ad \cos(\theta)$ with θ given in eye space by $\cos(\theta) = \frac{-\vec{E}}{|\vec{E}|} \cdot \hat{H}$, we have:

$$a = \max\left(-(\vec{E} \cdot \hat{H}) \pm \sqrt{(\vec{E} \cdot \hat{H})^2 - d^2 + q^2}\right)$$

$$\vec{R} = \vec{E} + a\hat{H}$$

To calculate the camera up-vector \vec{V} we can use the following equation:

$$\vec{V} = \begin{pmatrix} 0.0 \\ \vec{V}_y \\ \vec{V}_z \end{pmatrix} = \begin{pmatrix} 0.0 \\ \frac{-ac \pm b\sqrt{a^2+b^2-c^2}}{a^2+b^2} \\ \frac{-cb \mp a\sqrt{a^2+b^2-c^2}}{a^2+b^2} \end{pmatrix} \quad \text{with}$$

$$a = \vec{R}_y \quad b = \vec{R}_z \quad c = -\hat{U} \cdot \vec{Q}$$

Occurring singularities for negative square roots indicate unsatisfiable geometrical conditions, which can occur, if the separation distance of the objects on screen is very large, but the camera is too far away from the objects, so that the constraints cannot be fulfilled. \hat{T} is a linear combination:

$$\hat{T} = \alpha\vec{Q} + \beta\hat{U} + \gamma(\vec{Q} \times \hat{U}) \quad \text{with}$$

$$\alpha = (-\vec{R}_z + \hat{V}_z(\hat{U} \cdot \vec{Q})) / \delta$$

$$\beta = (-\hat{V}_z(\vec{Q} \cdot \vec{Q}) + \vec{R}_z(\hat{U} \cdot \vec{Q})) / \delta$$

$$\gamma = (-\vec{R}_x \hat{V}_y) / \delta$$

$$\delta = \vec{Q} \cdot \vec{Q} - (\hat{U} \cdot \vec{Q})^2$$

To transform \vec{E} from the local coordinate system of the camera into world space we create the rotation matrix \mathbf{M} as follows:

$$\hat{C}_Z = -\hat{T}$$

$$\hat{C}_X = \hat{U} \times \hat{C}_Z$$

$$\hat{C}_Y = \hat{C}_Z \times \hat{C}_X$$

$$\mathbf{M} = ((\hat{C}_X) (\hat{C}_Y) (\hat{C}_Z))$$

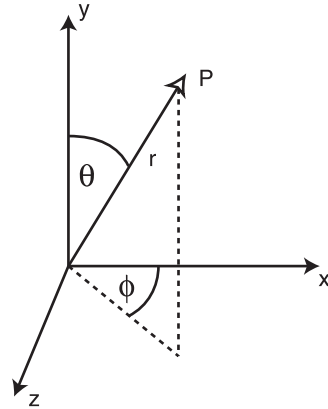


Figure A.6: Spherical coordinates.

\vec{D} is then

$$\vec{D} = d \frac{\mathbf{M} \cdot \vec{E}}{|\mathbf{M} \cdot \vec{E}|}$$

The camera position \vec{P}_{Cam} in world space is then

$$\vec{P}_{Cam} = \vec{P}_P + \vec{D}$$

and the camera orientation is given by $\hat{C}_X, \hat{C}_Y, \hat{C}_Z$. If the camera position is in the wrong half-space of the plane of action, and thus hurts the plane of action rules, we simply multiply the given on-screen coordinates x_P and x_S by -1 .

A.3 Spherical Coordinates

The second approach to position the camera in relation to a given object is to use a shot definition based on spherical coordinates. Having the necessary distance between camera and the focused object, we put the camera into a spherical orbit at a specific distance away from the object. This can be easily realised by describing the camera position via spherical coordinates.

Given the elevation by the vertical angle θ , the azimuth by the horizontal angle ϕ , and radius r , the cartesian coordinates \vec{P} can be generated as follows:

$$\vec{P} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} r \cos(\phi) \sin(\theta) \\ r \cos(\theta) \\ r \sin(\phi) \sin(\theta) \end{pmatrix}$$

In this special case, one wants the camera to look down the positive z-axis of the objects local coordinate system for $\phi = \theta = 0$ so that the ‘front’, based on the point-of-interest and action-direction, of the object (if it has any) is centered for these values. \vec{P} represents the resulting world space coordinates. The transformation from world space into the object’s local coordinate system can be done with the rotation matrix \mathbf{M} . Using the objects local coordinate system vectors $\hat{O}_X, \hat{O}_Y, \hat{O}_Z$ as columns of the rotation matrix, we get:

$$\mathbf{M} = \left(\begin{array}{ccc} (\hat{O}_X) & (\hat{O}_Y) & (\hat{O}_Z) \end{array} \right)$$

The complete formula to convert spherical coordinates from the object’s local system into world space cartesian coordinates \vec{W} is then given by this formula:

$$\vec{W} = \mathbf{M} \begin{pmatrix} r \cos(\phi - \frac{\pi}{2}) \sin(\theta + \frac{\pi}{2}) \\ r \cos(\theta + \frac{\pi}{2}) \\ r \sin(\phi - \frac{\pi}{2}) \sin(\theta + \frac{\pi}{2}) \end{pmatrix} \quad (\text{A.2})$$

A.4 Bounding-Spheres

We are currently using object representations based on bounding-spheres to describe object sizes. These bounding-volumes do not represent the exact geometrical extends of an object, but are sufficient to describe the respective maximal extends. Having this information, the camera can assure that an object like a person is always shown completely during a ‘Full Shot’ for instance. In some cases with very thin and long geometries, more exact representations for objects should be used. But for the scope of this work, bounding-spheres are an appropriate and efficient solution to represent geometrical extends.

For some types of shots¹, the camera has to focus on several objects instead of only one. In such a case we need to calculate a bounding-volume including all objects. The problem of finding an exact bounding-sphere for a number of smaller bounding-spheres is known as a complex mathematical problem (see [14]). For this thesis, we implemented an algorithm computing

¹Master two shots during the establishment of dialogues for instance.

an approximate solution enclosing all given bounding-spheres, which is however not the smallest possible bounding-sphere. The introduced error can result in shot sizes showing objects a little smaller than expected for high numbers of enclosed objects. But since there is already some error introduced by using object representations based on bounding-spheres only and since the number of participating objects of a scene is usually relatively small, the error due to our approximation is not relevant for practicality.

Our algorithm computing an enclosing bounding-sphere for bounding-spheres works as follows:

1. Calculate a world coordinate oriented bounding-box of all bounding-spheres by iterating through the objects of a narrative event and storing the minimum and maximum values for each dimension.
2. Use bounding-box midpoint as midpoint of the enclosing bounding-sphere.
3. Compute the radius of the enclosing bounding-sphere as the maximum distance between the enclosing bounding-sphere's midpoint and the midpoint of each enclosed bounding-sphere plus the respective enclosed bounding-spheres' radii.

By this we get a bounding sphere centered at the midpoint of the enclosing bounding-box with a radius just as big as the maximum distance to the enclosed spheres' outer hulls, which encloses all of the smaller spheres (Figure A.7).

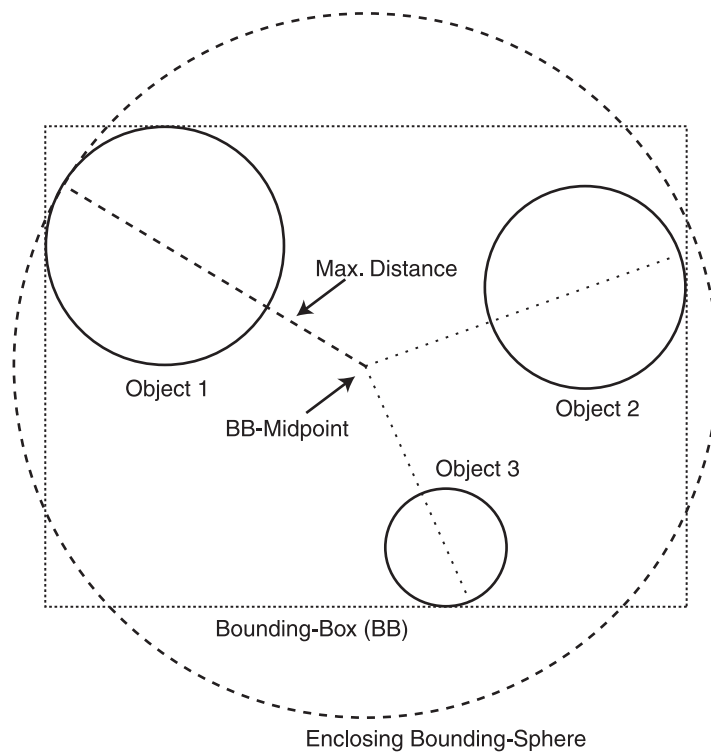


Figure A.7: This Figure shows a 2D-example of three objects, an axis-aligned bounding-box enclosing the objects, the distances from the midpoint of the bounding-box to the outer hull of each object, and the enclosing bounding-sphere.

Appendix B

Half-Life Screenshots

The following images are screenshots taken from the game Half-Life with the camera agent being activated. Figures B.1, B.2 show dialogues between the player and scientists, Figures B.3, B.4 are pictures of an intermission showing an dimensional breach to an alien world, and Figures B.5,B.6 are fights scenes between the player and some enemies.

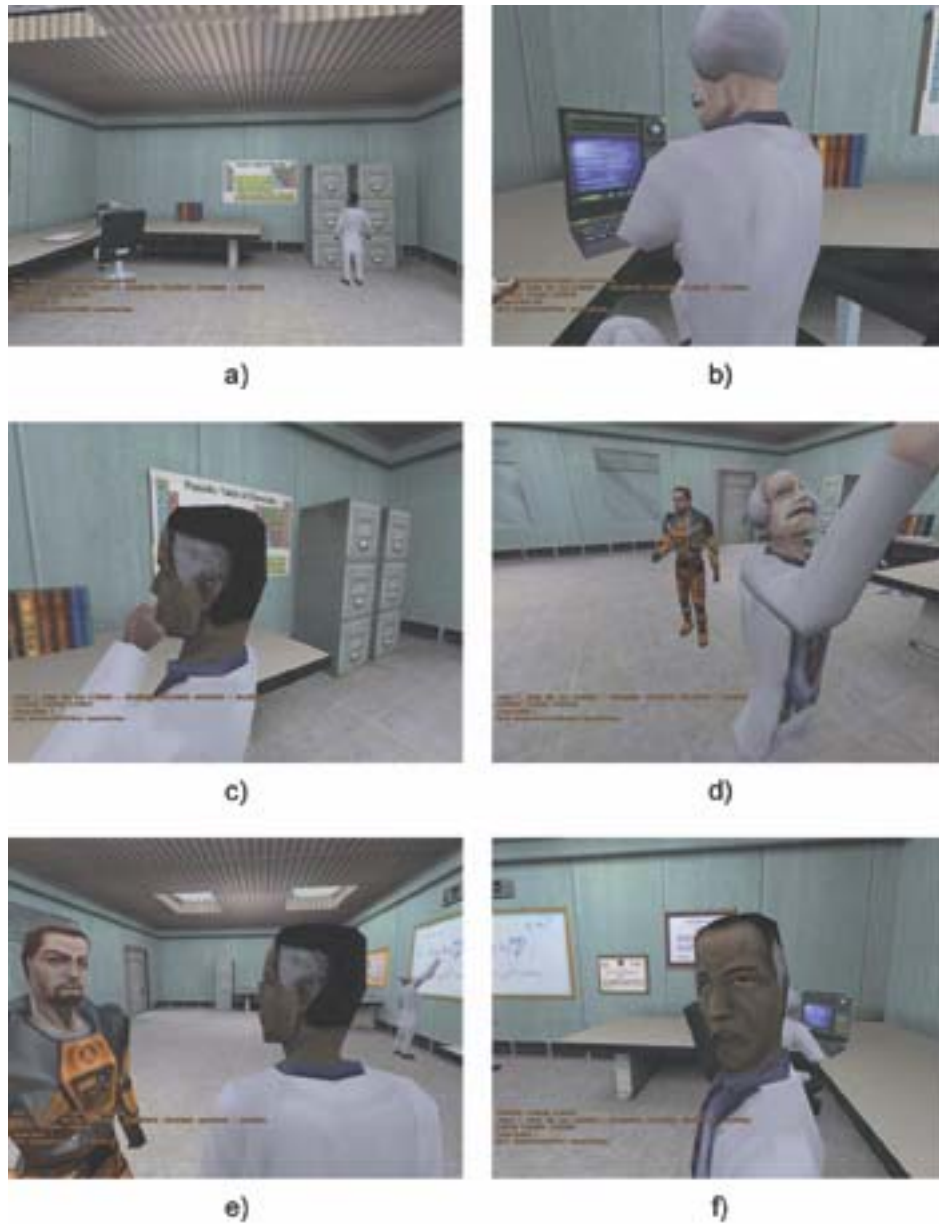


Figure B.1: A dialogue between three scientists and the player showing different shot types like an establishing master two shot, angular singles, or over-the-shoulder shots. a) “Do you know who ate all the donuts?” b) “Theoretically. . .” c) “Are you thinking what I am thinking?” d) “You can’t be serious!” e) “Aren’t you a bit worried about that exponential cascade scenario we discussed?” f) “Hello?”

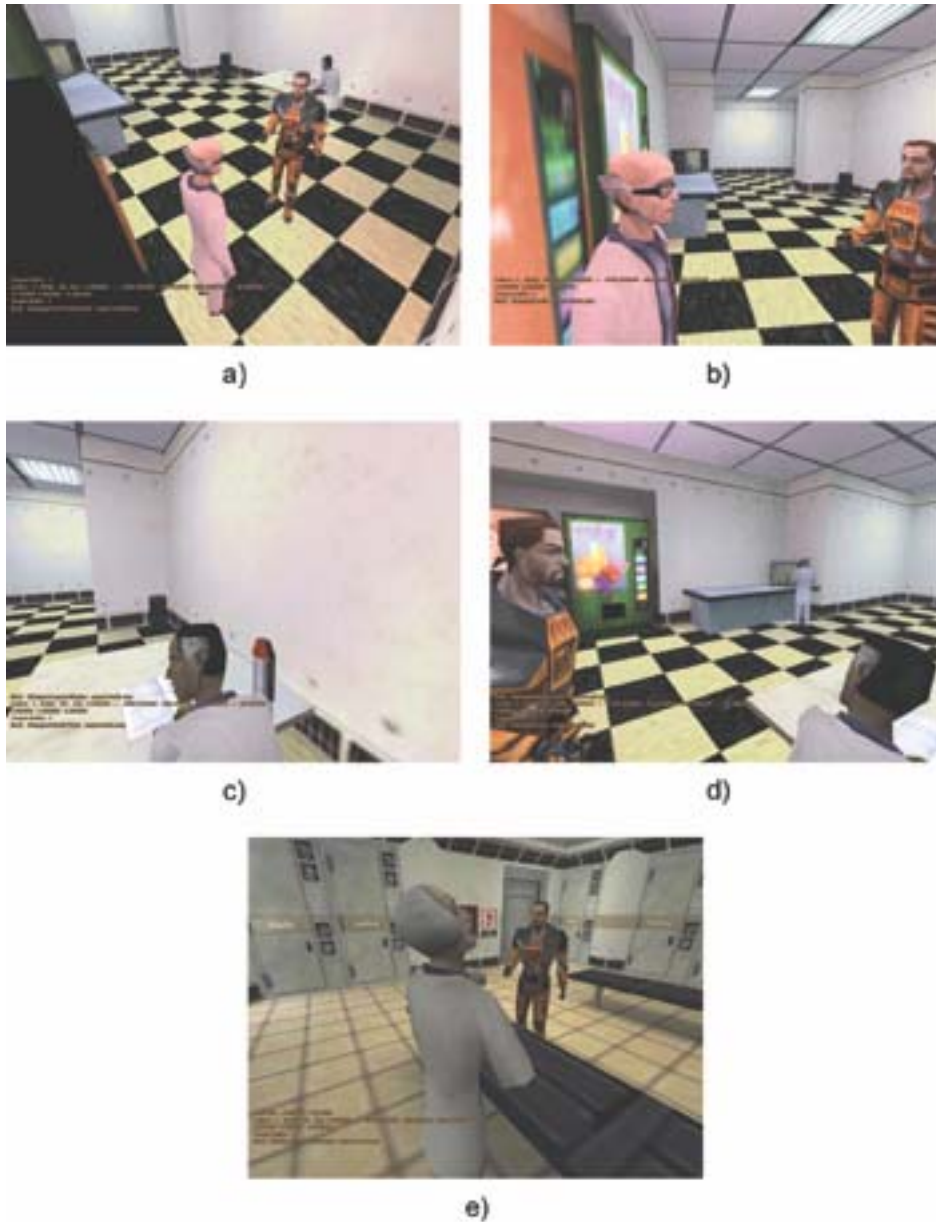


Figure B.2: More dialogue-screenshots. a) “Ah, Freeman!” b) “Have you’ve been able to get the beverage machine to work, yet?” c) “Didn’t you just ask me that?” d) “You know you repeat yourself sometimes!” e) “Why do we all have to wear these ridiculous ties?”

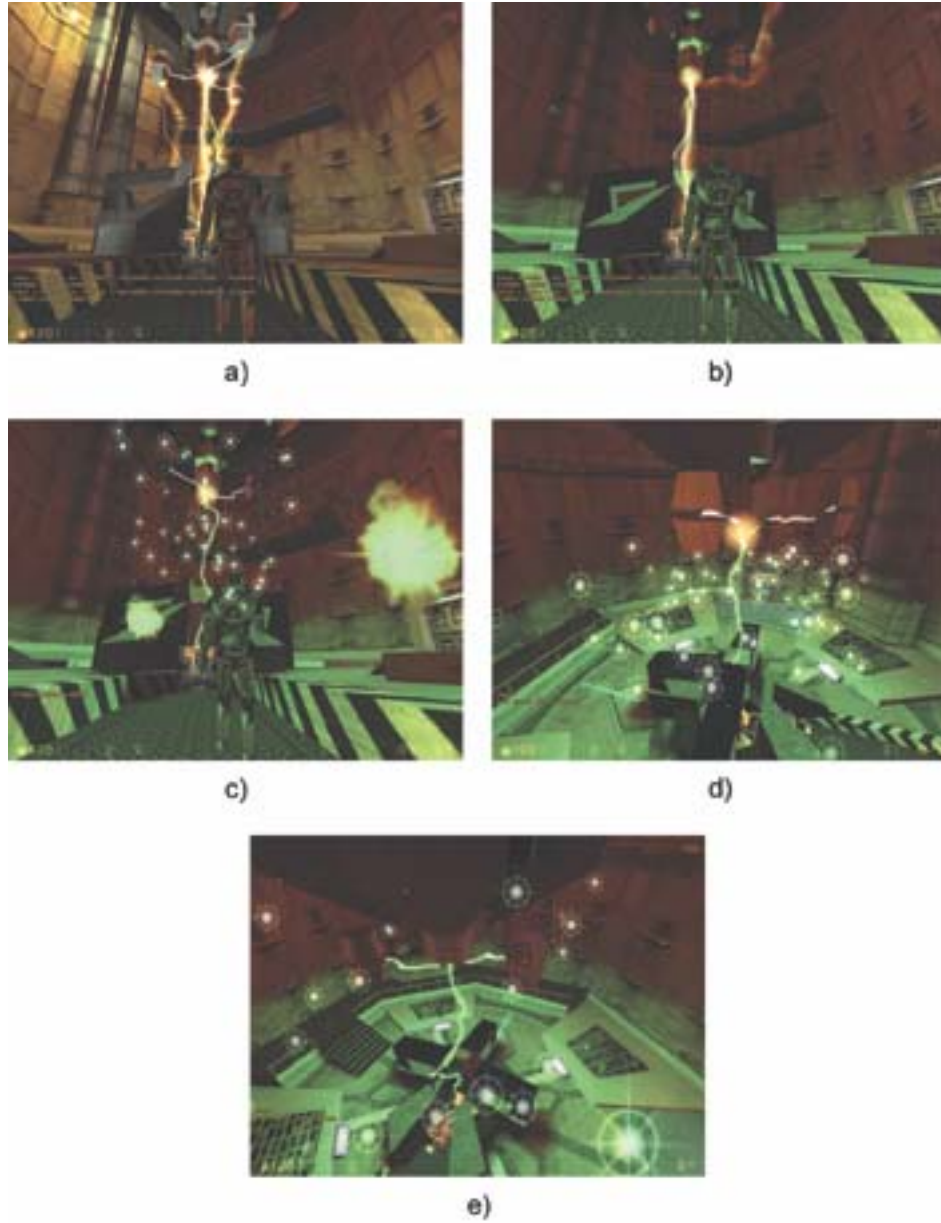


Figure B.3: Images a) - e) show a reactor of the Black Mesa Research Facility getting out of control. The pictures were taken in sequence, while the camera was circling around the reactor.

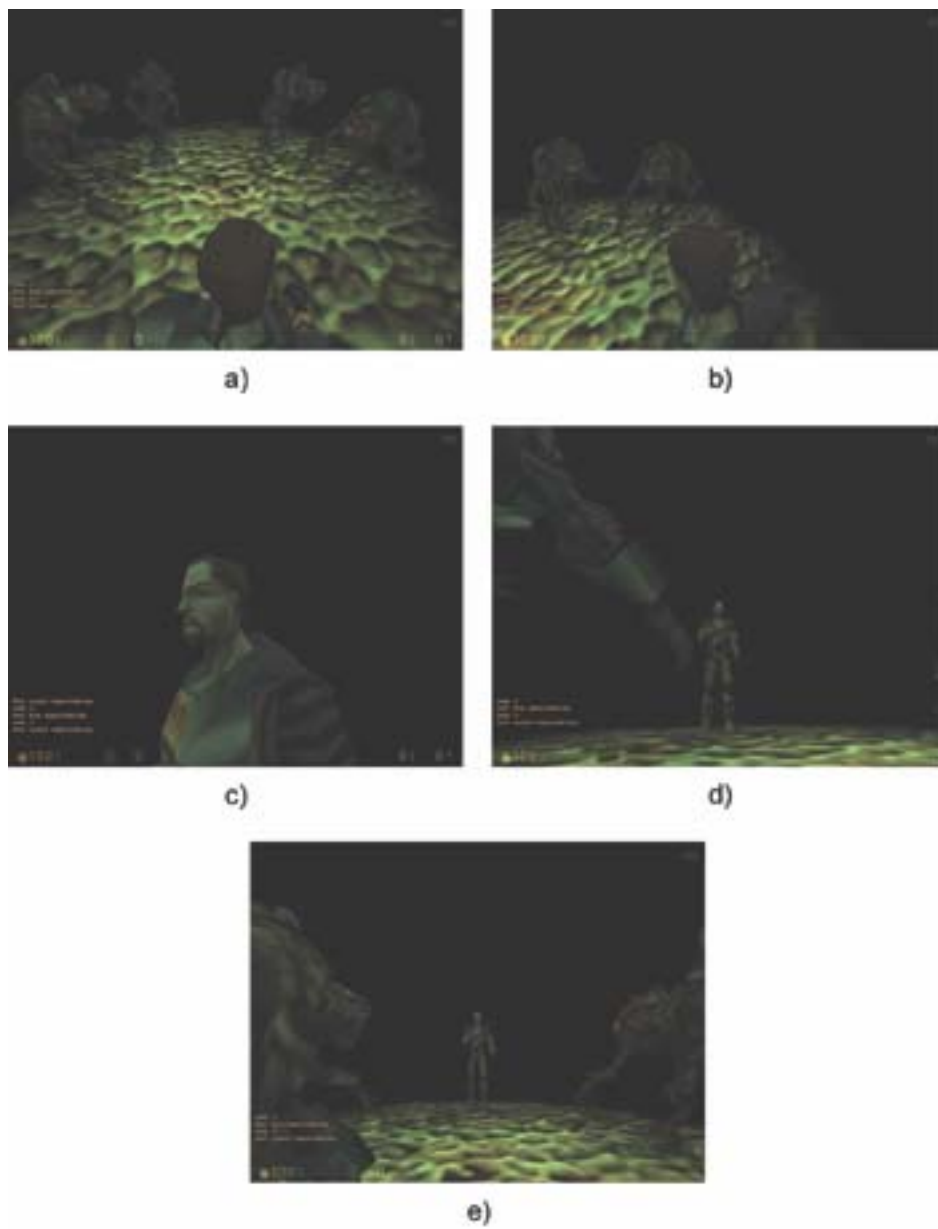


Figure B.4: Images a) - e) present another camera movement during the intermission, showing the player surrounded by aliens.

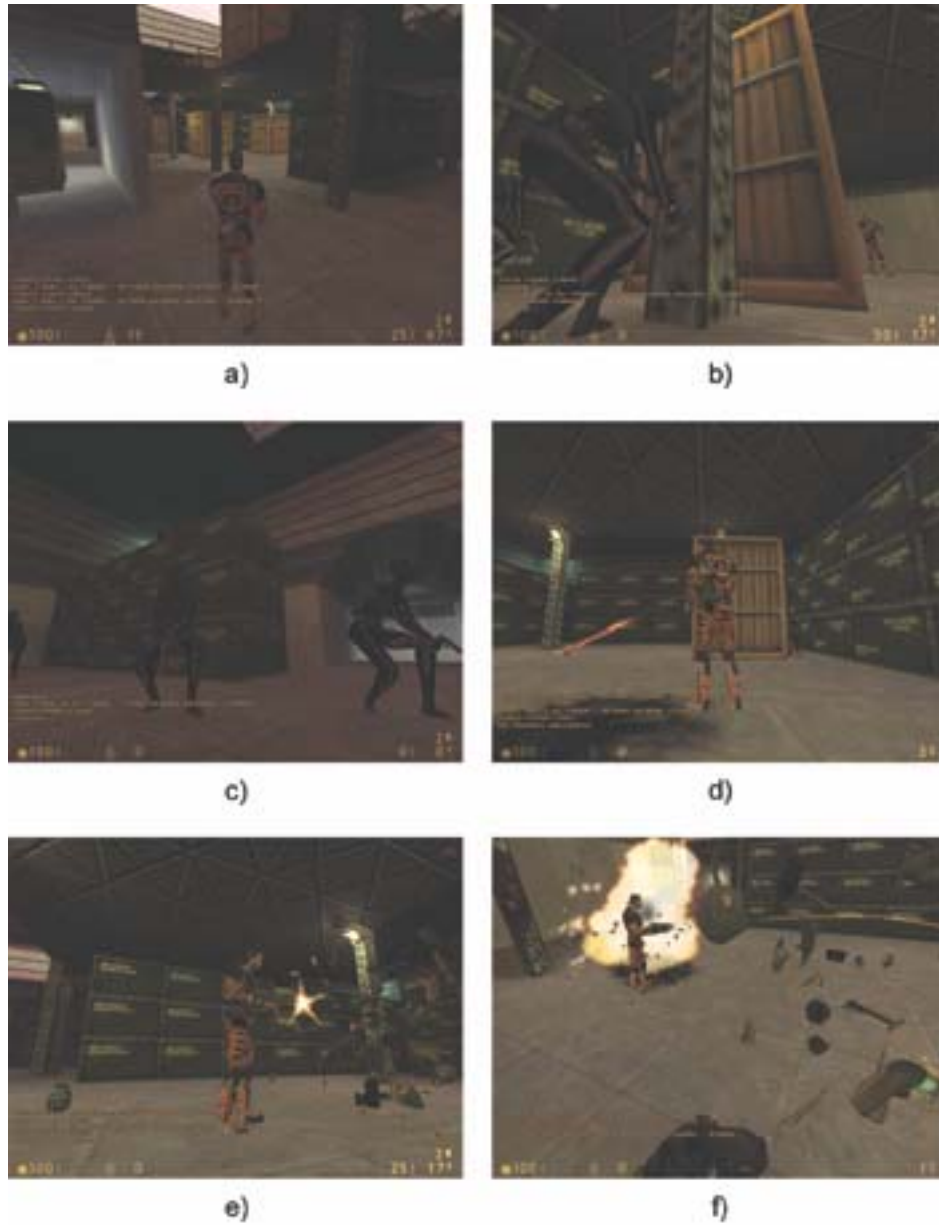


Figure B.5: These images show some fight scenes from the game. a) shows the standard follow-up camera, while the player is moving around in the game-world. Pictures b) and c) show assassins waiting in an ambush. d) - f) show the player fighting against the enemies.

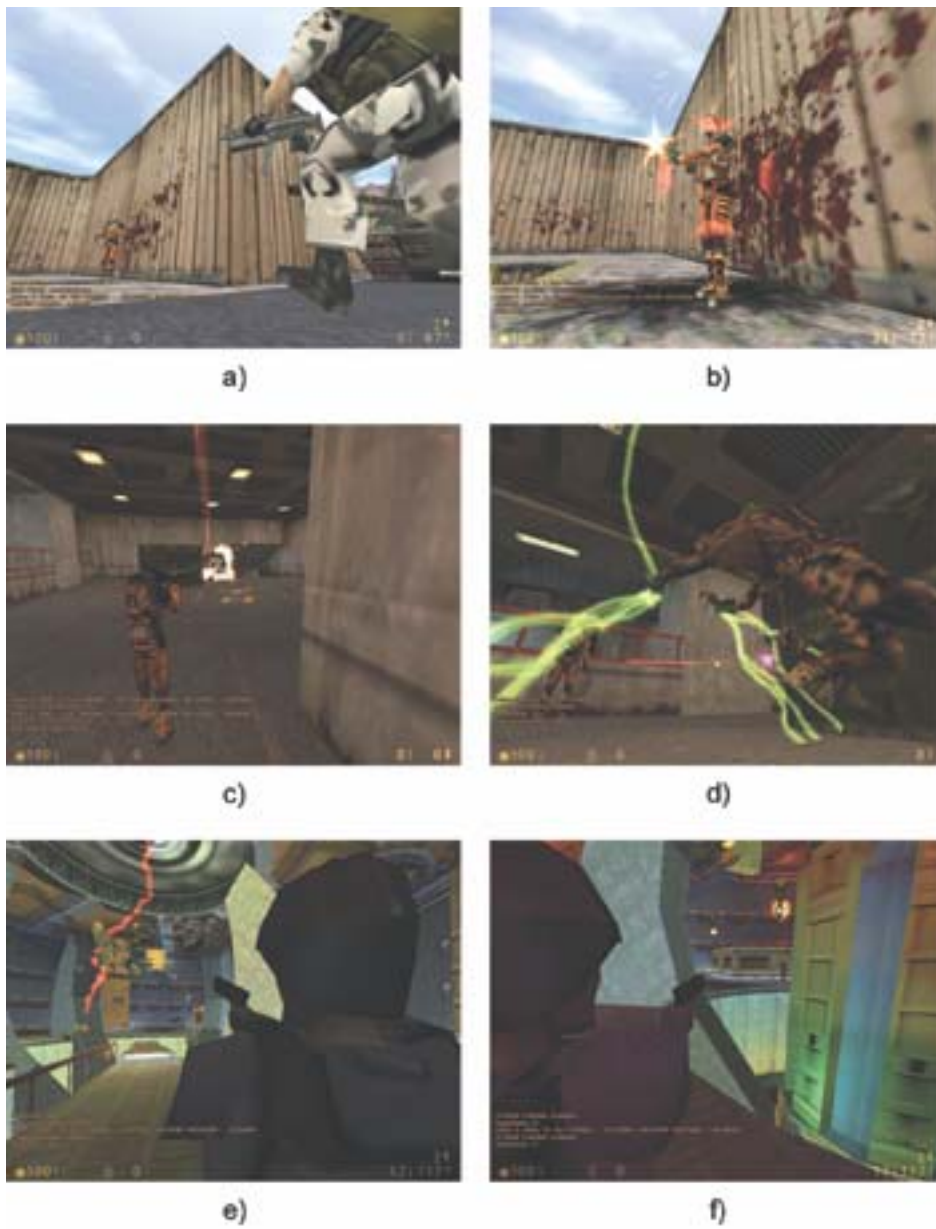


Figure B.6: Pictures a) - f) show some further impressions from different fight scenes.

Appendix C

Example Configuration

The following is an example configuration to set up three basic shots for dialogues (Figures B.1, B.2), a shot for fight scenes (Figures B.5, B.6), and one shot for a camera movement during the intermission (Figure B.4). These shot definitions are taken from our shot library for Half-Life.

```
[Knowledge]
Behaviours = DIALOGUE1, DIALOGUE2, DIALOGUE3, ATTACK1, LONELY1
```

```
#####
```

```
[DIALOGUE1]
Name = DialogueMasterTwoShot

SceneryType = Dialogue
ActionType = Physical
Neuron = NeutralLowCoherence
ShotClass = Dialogue1
NegativePredecessorShotClasses = Dialogue1
NumberOfNecessaryObjects = 1

FocusedObject = all
ShotSize = FullShot
ShotDefinition = SphericalCoordinates
Phi = 90.0
Theta = -10.0
ScreenPositionFocusedObjectX = 0.0
ScreenPositionFocusedObjectY = -0.2
```

```
#####
```

```
[DIALOGUE2]
```

```
Name = DialogueAngularSingles
```

```
SceneryType = Dialogue
```

```
ActionType = Physical
```

```
Neuron = NeutralHighCoherence
```

```
ShotClass = Dialogue2
```

```
NegativePredecessorShotClasses = Dialogue2
```

```
NumberOfNecessaryObjects = 1
```

```
FocusedObject = 0
```

```
ShotSize = CloseShot
```

```
ShotDefinition = SphericalCoordinates
```

```
Phi = 30.0
```

```
Theta = -20.0
```

```
ScreenPositionFocusedObjectX = 0.3
```

```
ScreenPositionFocusedObjectY = -0.2
```

```
#####
```

```
[DIALOGUE3]
```

```
Name = DialogueOverTheShoulder
```

```
SceneryType = Dialogue
```

```
ActionType = Physical
```

```
Neuron = NeutralHighCoherence
```

```
ShotClass = Dialogue3
```

```
NegativePredecessorShotClasses = Dialogue3
```

```
NumberOfNecessaryObjects = 2
```

```
FocusedObject = 0
```

```
ShotSize = MediumFullShot
```

```
ShotDefinition = ObjectScreenPosition
```

```
ScreenPositionFocusedObjectX = 0.1
```

```
ScreenPositionFocusedObjectY = -0.2
```

```
ScreenPositionBackgroundObjectX = -0.2
```

```
ScreenPositionBackgroundObjectY = 0.2
```

```
#####
```

[ATTACK1]

Name = Attack1

SceneryType = Action

ActionType = Physical

Neuron = HecticHighCoherence

ShotClass = Attack1

NegativePredecessorShotClasses = Attack1

NumberOfNecessaryObjects = 2

FocusedObject = 0

ShotSize = FullShot

ShotDefinition = ObjectScreenPosition

ScreenPositionFocusedObjectX = 0.4

ScreenPositionFocusedObjectY = 0.2

ScreenPositionBackgroundObjectX = -0.5

ScreenPositionBackgroundObjectY = -0.3

#####

[LONELY1]

Name = Lonely1

SceneryType = Action

ActionType = Mental

Neuron = Lonely

ShotClass = Lonely1

NegativePredecessorShotClasses = Lonely1

NumberOfNecessaryObjects = 1

FocusedObject = 0

DampingFactor = 0.8

MinimumShotLength = 5.0

KeyFrames = 0.0, 4.0

ShotSize = CloseUp, ExtremeLongShot

ShotDefinition = SphericalCoordinates

Phi = -180.0, 50.0

Theta = -80.0, 5.0

ScreenPositionFocusedObjectX = 0.0, -0.5

ScreenPositionFocusedObjectY = -1.0, -0.4

List of Figures

1.1	An example of two shots of an identical scene conveying completely different intentions. In image a) the character seems anxious. Image b) creates the impression of an evil character raising both fists into the air.	5
2.1	Reference framing heights for the human body.	12
2.2	Close-Up of Charlie Chaplin in the movie ‘City Lights’, emphasising his facial expression and emotions.	13
2.3	Low angle shot of the vampire Nosferatu emphasising his evil, terrifying nature.	14
2.4	Comparison of two shots from different angles of the same model. a) shows a low angle shot creating the intended evil impression. In b), the spectator is irritated by the high angle perspective.	16
2.5	The line of action between two characters during a dialogue.	27
2.6	The camera stays in the halfspace of Camera A during the consecutive shots a) and b).	27
2.7	During the transition from a) to b) the camera jumps over the line of action into the halfspace of Camera B.	27
2.8	A camera setup using the triangle system for angular singles and a master two shot.	29
2.9	a) Master two shot. b) Over-the-shoulder shot. c)-d) Point of view shots. e)-f) Angular singles. g)-h) Profile shots.	30
4.1	General system structure and flow of information.	46
4.2	Structure of narrative events.	48
4.3	Two examples of bounding-spheres and action directions.	51
4.4	Simple event-priority decay functions.	52
4.5	A complex priority decay function.	53
4.6	Representing complex priority changes by updating event-priorities.	53

4.7	Examples of the coherence of two events.	55
4.8	Output of the camera agent for the example of a conversation between three characters and some action. a) Establishment shot: Mr. A talking to Mr. B, Mrs. C is listening. b) Over-the-shoulder shot: Mr. B talking a little upset to Mr. A. c) Close-up shot: Mr. A talking very angrily to Mr. B. d) Overhead shot: Mrs. C crying. e) Mr. B shoots at Mr. A. f) Mr. A falls backwards.	56
4.9	Notation.	60
4.10	A single neuron.	61
4.11	Linear separability of boolean functions having two inputs. . .	62
4.12	An interval separating one input into three sections.	64
4.13	Examples of representable intervals using two inputs.	64
4.14	Example of non-representable intervals.	65
4.15	The three main modules of the camera agent and the flow of information.	67
4.16	An example to generate an over-the-shoulder shot for dialogues using virtual screen positions. The left part of the image shows both characters standing in 3D-space, talking to each other. The camera and the screen on the right show the approximate camera position and the resulting picture by specifying on-screen coordinates (illustrated by the red and green lines).	75
4.17	Shot definition using spherical coordinates. The camera position is defined by two angles and a distance.	76
5.1	Architecture and flow of information between the integrated camera agent and the two main components of Half-Life. . . .	84
5.2	A dialogue between four characters. a) Original first-person view. b) Camera agent: the spectator is guided through the conversation by appropriate camera shots.	88
5.3	A scene from one of the intermissions. The player is transported through a dimensional breach and is surrounded by aliens. a) Original first-person view. b) Camera agent: a sequence of shots. The camera is circling around the player, and the terrifying impression of the intermission is emphasised in contrast to a).	89
5.4	A fight between the player and some assassins. a) Original first-person view. b) Camera agent: four shots of attacking enemies (the player being attacked is wearing the orange suit).	89

A.1	The underlying right-handed cartesian coordinate system. A spectator looks down the negative z-Axis.	96
A.2	Notation for camera distance equations.	96
A.3	Shot sizes $O_{FOVRatio}$ and Body-PoI interpolation values $I_{BodyPoI}$	97
A.4	Geometrical relations for the computation of the necessary distance between the camera and an object to achieve a specific object size on the screen.	97
A.5	Used vectors and their meaning.	99
A.6	Spherical coordinates.	101
A.7	This Figure shows a 2D-example of three objects, an axis-aligned bounding-box enclosing the objects, the distances from the midpoint of the bounding-box to the outer hull of each object, and the enclosing bounding-sphere.	104
B.1	A dialogue between three scientists and the player showing different shot types like an establishing master two shot, angular singles, or over-the-shoulder shots. a) "Do you know who ate all the donuts?" b) "Theoretically..." c) "Are you thinking what I am thinking?" d) "You can't be serious!" e) "Aren't you a bit worried about that exponential cascade scenario we discussed?" f) "Hello?"	106
B.2	More dialogue-screenshots. a) "Ah, Freeman!" b) "Have you've been able to get the beverage machine to work, yet?" c) "Didn't you just ask me that?" d) "You know you repeat yourself sometimes!" e) "Why do we all have to wear these ridiculous ties?"	107
B.3	Images a) - e) show a reactor of the Black Mesa Research Facility getting out of control. The pictures were taken in sequence, while the camera was circling around the reactor.	108
B.4	Images a) - e) present another camera movement during the intermission, showing the player surrounded by aliens.	109
B.5	These images show some fight scenes from the game. a) shows the standard follow-up camera, while the player is moving around in the game-world. Pictures b) and c) show assassins waiting in an ambush. d) - f) show the player fighting against the enemies.	110
B.6	Pictures a) - f) show some further impressions from different fight scenes.	111

Bibliography

- [1] T. Akenine-Möller, E. Haines, *Real-Time Rendering, Second Edition*, A K Peters 2002, ISBN 1-56881-182-9
- [2] *Poetics of Aristotle*, University of North Carolina Press 1967, ISBN 0807840173
- [3] D. Arijon, *Grammar of the Film Language*, Focal Press, Boston, 1976
- [4] W. Bares, J. Gregoire, J. Lester, *Realtime Constraint-Based Cinematography for Complex Interactive 3D Worlds*, AAAI/IAAI 1998, p1101-1106
- [5] J. Bates, A. Loyall, W. Reilly, *An Architecture for Action, Emotion, and Social Behavior*, Proceedings of 4th European Workshop on Modeling Autonomous Agents in a Multi-Agent World 1994, p55-68
- [6] S. Beckhaus, F. Ritter, T. Strothotte, *Guided Exploration with Dynamic Potential Fields: The CubicalPath System*, Computer Graphics Forum 20(4) 2001, p201-210
- [7] J. Blinn, *Where Am I? What Am I Looking At?*, Jim Blinn's Corner: A Trip Down the Graphics Pipeline, Morgan Kaufmann, 1996
- [8] D. Christianson, S. Anderson, L. He, D. Salesin, D. Weld, M. Cohen, *Declarative Camera Control for Automatic Cinematography*, AAAI/IAAI, Vol. 1 1996, p148-155
- [9] S. Drucker, D. Zeltzer, *CamDroid: A System for Implementing Intelligent Camera Control*, Symposium on Interactive 3D Graphics 1995, p139-144
- [10] EIDOS Interactive, *Tombraider*, <http://www.eidos.com/>
- [11] R. Evans, Lionhead Studios, *Varieties of Learning*, AI Game Programming Wisdom, Charles River Media, 2002

- [12] Fall Symposium on Narrative Intelligence, AAAI 1999, <http://www-2.cs.cmu.edu/~michaelm/narrative.html>,
- [13] D. Fincher, *Fight Club*, USA 1999
- [14] K. Fischer, *The Smallest Enclosing Ball of Balls*, Thesis, Institute for Theoretical Computer Science, ETH-Zürich, 2001
- [15] Foley, van Dam, Feiner, Hughes, *Computer Graphics: Principles and Practice*, Addison-Wesley 2nd edition 1995
- [16] J. Funge, X. Tu, Demetri Terzopoulos, *Cognitive Modeling: Knowledge, Reasoning and Planning for Intelligent Characters*, Proceedings of SIGGRAPH 1999, p29–38
- [17] M. Gleicher, A. Witkin, *Through-the-Lens Camera Control*, Proceedings of SIGGRAPH 1992
- [18] N. Halper, P. Olivier, *CamPlan: A Camera Planning Agent*, AAAI Workshop on Smart Graphics 2000
- [19] N. Halper, R. Helbing, T. Strotthotte, *A Camera Engine for Computer Games: Managing the Trade-Off Between Constraint Satisfaction and Frame Coherence*, Computer Graphics Forum: Proceedings EUROGRAPHICS 20(3) 2001, p174-183
- [20] B. Hawkins, *Creating an Event-Driven Cinematic Camera*, Game Developer Magazine 10/11 2002, CMP Media LLC, p34-40(10), p36-40(11)
- [21] L. He, M. Cohen, D. Salesin, *The Virtual Cinematographer: A Paradigm for Automatic Real-Time Camera Control and Directing*, Proceedings of SIGGRAPH 1996, p217-224
- [22] ID-Software, *Quake*, <http://www.id-software.com>
- [23] *alVRed*, Nonlinear Dramaturgy in VR-Environments, Laboratory for Mixed Realities, Cologne, <http://www.alvred.de/>
- [24] J. Lander, “*Lights... Camera... Let’s Have Some Action Already!*”, Game Developer Magazine 4 2000, CMP Media LLC, p15-20
- [25] R. Raymond Lang, *A Declarative Model for Simple Narrative*, AAAI 1999
- [26] Lionhead Studios, *Black & White*, <http://www.lionhead.com/>

- [27] R. Lu, S. Zhang, *Automatic Generation of Computer Animation*, Springer 2002, ISBN 3-540-43114-4
- [28] P. Kandorfer, *Lehrbuch der Filmgestaltung*, Deutscher Ärzte Verlag GmbH 1978
- [29] S. Katz, *Film Directing Shot By Shot: Visualizing From Concept To Screen*, Micheal Wiese Productions 1991, ISBN 0-941188-10-8
- [30] Konami, *Metal Gear Solid 2*, <http://www.konami.com/>
- [31] John F. Meech, *Narrative Theories as Contextual Constraints for Agent Interaction*, AAAI 1999
- [32] J. Monaco, *How to Read a Film*, Oxdord University Press, London / New York, 2000
- [33] T. Muldner, *C++ Programming with Design Patterns Revealed*, Addison Wesley, 2002, ISBN 0-201-72231-3
- [34] M. Otten, *Half-Life TV*, <http://www.slipgate.de/>
- [35] *The Dialogues of Plato*, Bantam Books 1994, ISBN 0553213717
- [36] S. Rabin ed., *AI Game Programming Wisdom*, Charles River Media, 2002
- [37] D. Rousseau, B. Hayes-Roth, *A Social-Psychological Model for Synthetic Actors*, Proceedings of the 2nd International Conference on Autonomous Agents 1998, ACM Press, p165–172, ISBN 0-89791-983-1
- [38] S. Russel, P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, 1995
- [39] M. Schroeder, *How to Tell a Logical Story*, AAAI 1999
- [40] Michael B. Skov, Peter B. Andersen, *Designing Interactive Narratives*, Proceedings of the 1st International Conference on Computational Semiotics in Games and New Media 2001
- [41] B. Stroustrup, *The C++ Programming Language, Special Edition*, Addison Wesley, Boston, 2000, ISBN 0-201-70073-5
- [42] N. Szilas, *Interactive Drama on Computer: Beyond Linear Narrative*, AAAI 1999

- [43] A. Guye-Vuilleme, D. Thalmann, *A high-level Architecture For Believable Social Agents*, VR Journal, Springer, Vol.5 2001, p95-106
- [44] J. Thoma, *Non-Photorealistic Rendering Techniques for Real-Time Character Animation*, Diploma Thesis, Aachen University, Germany, 2002
- [45] B. Tomlinson, B. Blumberg, D. Nain, *Expressive Autonomous Cinematography for Interactive Virtual Environments*, Proceedings of the Fourth International Conference on Autonomous Agents 2000, p317-324
- [46] Valve Software, *Half-Life*, <http://www.sierra.com/games/half-life/>
- [47] Valve Software, *Editing Resource Center*, <http://www.valve-erc.com/>
- [48] A. & L. Wachowski, *The Matrix*, USA 1999
- [49] ed. Gerhard Weiss, *Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence*, The MIT Press 1999, ISBN 0262232030
- [50] M. Wooldridge, N. R. Jennings, *Intelligent Agents: Theory and Practice*, The Knowledge Engineering Review 10(2) 1995
- [51] R. Wright, Jr., M. Sweet, *OpenGL SuperBible, Second Edition*, Waite Group Press, 2000