Towards Adaptive Virtual Camera Control In Computer Games

Paolo Burelli and Georgios N. Yannakakis

Center For Computer Games Research
IT University Of Copenhagen
Rued Langgaards Vej 7 2300 Copenhagen, Denmark
pabu@itu.dk, yannakakis@itu.dk

Abstract. Automatic camera control aims to define a framework to control virtual camera movements in dynamic and unpredictable virtual environments while ensuring a set of desired visual properties. We investigate the relationship between camera placement and playing behaviour in games and build a user model of the camera behaviour that can be used to control camera movements based on player preferences. For this purpose, we collect eye gaze, camera and game-play data from subjects playing a 3D platform game, we cluster gaze and camera information to identify camera behaviour profiles and we employ machine learning to build predictive models of the virtual camera behaviour. The performance of the models on unseen data reveals accuracies above 70% for all the player behaviour types identified. The characteristics of the generated models, their limits and their use for creating adaptive automatic camera control in games is discussed.

1 Introduction

In 3D computer games, a virtual camera defines the point of view of the player on the virtual environment and it mediates her visual perceptions. Therefore, a wide range of aspects of the player experience, such as interaction and storytelling are heavily affected by the virtual camera placement and motion [13]. In games and other 3D applications the camera is either manually controlled by the player during her interaction or placed and animated a priori by a designer. However, manual control of the camera often proves challenging for the player as it increases the complexity of the interaction and, on the other hand, statically predefined cameras fail to cope with dynamic virtual environments.

These limits have driven the research towards the identification of a new camera control paradigm: automatic camera control. Within this framework the camera is controlled using high-level and environment-independent requirements, such as the visibility of a particular object or the size of that object's projected image on the screen. A software module, commonly referred as camera controller, dynamically and efficiently infers the ideal camera position and motion from these requirements and the current game state.

The process of finding the virtual camera configuration that best fit a set of requirements has been widely investigated [8]. On the contrary, the requirements themselves have received little attention despite their impact on player experience [11]. Virtual camera parameters are commonly hand-crafted by game designers and do not consider player preferences. Including the player in the definition of these parameters requires the construction of a model of the relationship between camera motion and player experience [11].

In this paper we investigate player preferences concerning virtual camera placement and animation, we propose a model of the relationship between camera behaviour, player behaviour and game-play and we evaluate the performance of this model. For this purpose, data from player gaze and the virtual camera motion is collected through a game experiment and used to identify and describe the players' camera behaviours. In the data-collection experiment the participants play a three-dimensional platformer game featuring all the stereotypical aspects of the genre's mechanics.

2 Background

Early studies on camera control focused on the definition of the camera properties and investigated the mapping between input devices and camera movement [16]. The main research focus in the field rapidly shifted towards automatic camera control since direct control of the camera has shown to be problematic for the user [9].

Several different techniques have been proposed for automatic camera control, based on a variety of mathematical models; however, the vast majority of the approaches model the problem as a constrained optimisation problem [8]. These approaches allow the designer to define a set of requirements on the frames that the camera should produce and on the camera motion. Depending on the approach, the controller positions and animates one or more virtual cameras in the attempt to generate a shot or a sequence of shots that satisfy the predefined requirements.

Requirements for the camera include constraints on camera motion (such as speed limit), constraints on camera position (such as maximum height), or constraints on the rendered frames. The last type of camera requirements, introduced by Bares et al. [2], defines required properties on the frames rendered using the camera such as subject inclusion or subject position within the frame.

2.1 Camera profiles

A large volume of research studies on automatic camera control is dedicated to the analysis of robust and time-efficient techniques to place and move the camera to satisfy a set of given requirements. These sets of requirements, usually referred as *camera profiles* [5], define the set of constraints and the objective function that needs to be optimised by the automatic camera control system.

Christianson et al. [7] introduced a language that permits the definition of shot sequences (idioms) with the desired timings and subjects. Other researchers extended Christianson's work by connecting shot plans with camera constraints [1], or by introducing advanced planning techniques to support interactive storytelling [6, 10]. While the aforementioned approaches address primarily the issues related to the manual design of camera behaviours for dynamic and interactive environments, other researchers have investigated approaches which does not require the contribution of a designer [3].

Yannakakis at al. [17] studied the impact of camera viewpoints on player experience and built a model to predict this impact. That research study demonstrates the existence of a relationship between player emotions, physiological signals and camera parameters. Therefore, in the light of these results, Picardi et al. [12] investigated the relationship camera and player behaviour in a 3D platform game. The article demonstrates the existence of a significant relationship between the player's preferences on camera, measured using a gaze tracker, and some game-play features such as number of collected items or number of jumps performed.

In this paper we extend this work by analysing the interplay between past player behaviour and camera control to automate the generation and selection of the virtual camera parameters.

2.2 Gaze interaction in games

Eye movements can be recognised and categorised according to speed, duration and direction [18]. In this paper, we focus on *fixations*, *saccades* and *smooth pursuits*. A fixation is an eye movement that occurs when a subject focuses at a static element on the screen; a saccade occurs when a subject is rapidly switching her attention from one point to another and a smooth pursuit is a movement that takes place when a subject is looking at a dynamic scene and she is following a moving object.

Sundstedt et al. [15] conducted an experimental study to analyse players' gaze behaviour during a maze puzzle solving game. The results of their experiment show that gaze movements, such as fixations, are mainly influenced by the game task. They conclude that the direct use of eye tracking during the design phase of a game can be extremely valuable to understand where players focus their attention, in relation to the goal of the game. Bernhard et al. [4] performed a similar experiment using a three-dimensional first-person shooting game in which the objects observed by the players were analysed to infer the player's level of attention. We are inspired by the experiment of Bernhard at al. [4]; unlike that study however, we analyse the player's gaze patterns to model the player's camera movements and we model the relationship between camera behaviour, game-play characteristics and player-behaviour.

(a) Avatar (b) Platform (c) Fuel Cell (d) Copper (e) Re-spawn Point (f) Jump Pad

Fig. 1: Main components of the game.

3 The Game

A three-dimensional platform game has been designed and developed as a testbed for this study[1]. The game features an alien-like avatar (see Fig. 1a) in a futuristic environment floating in the open space. The player controls the avatar and the camera using keyboard and mouse. Avatar movements, defined in camera-relative space, are controlled using the arrow keys, and jump and hit actions are activated by pressing the left and right mouse buttons respectively. The camera orbits around the avatar at a fixed distance; the player can change the distance using the mouse wheel and can rotate the camera around the avatar by moving the mouse.

The environment thought which the avatar moves is composed by floating platforms . Each platform can be connected to another platform directly, forming a cluster of platforms, through a bridge or it can be completely isolated, in which case the avatar is required to jump to move from one platform to the other. Four main object types may appear on the platforms: *fuel cells*, *coppers*, *re-spawn points* and *jump pads*. Fuel cells (see Fig. 1c) are collectable items increasing the score of the player. Coppers (see Fig. 1d) are opponent non player characters. Re-spawn points (see Fig. 1e) are small stands placed on some platforms that, when activated, act as the avatar's spawn point after he falls from a platform. Finally, the jump pads (see Fig. 1f) are black and yellow striped areas which allow the player to perform a longer jump.

The aim of the player is to cross the virtual world until the last platform while collecting fuel cells and killing coppers to achieve the highest possible score. However, the player needs also to avoid falling from the platforms and loosing too much time as this will negatively affect the final score.

The game is preceded by a tutorial level that explains the game controls and gives an overview of the contribution of the game actions to the score. Moreover, during the tutorial, the player is walked through all the challenges she will come across in the game. The game is divided in a series of areas classified into three categories according to the game-play experience they offer: *jump*, *fight* and

[1] The game is based on Lerpz, a tutorial game by Unity Technologies — http://www.unity3d.com

(a) Fight area (b) Jump area (c) Collection area

Fig. 2: The three different area types met in the game.

collection areas. Figure 2a shows a fight area where the main threat is given by the opponent copper at the centre of the platform. The jump area depicted in Fig. 2b is composed by several small floating platforms; the player needs to make the avatar jump across all the platforms to complete the area. Figure 2c shows an area where the main task of the player is to collect the fuel cells placed around the platform. In total, the game comprises 34 areas containing 14 collection areas, 11 fight areas and 9 jump areas.

4 Experimental Methodology

Our experimental hypothesis is that the way a player controls the virtual camera depends on what actions she performs in the game and on how she performs them. We represent the virtual camera behaviour as the amount of time the player spends framing and observing different objects in the game environment while playing the game. This representation of behaviour is chosen over a direct model of the camera position and motion as it describes the behaviour in terms of the content visualised by the camera and, therefore, it is independent of the absolute position of the avatar, the camera and other objects.

To get information about the objects observed by the player during the experiment, we used the *Eyefollower*[2] gaze tracker which samples the player's gaze at a 120 Hz frequency (60 Hz per eye).

Twenty-nine subjects participated in the experiment. Twenty-four were male, five were female; the age of the participants ranged between 23 and 34 years. Statistical data about game-play behaviour, virtual camera movement and gaze position is collected for each participant. The collection of the data is synchronised to the Eyefollower sampling rate, therefore, both data from the game and from the gaze tracker are sampled 120 times per second. Each data sample contains: information about the game-play, position and orientation of the camera, coordinates of the gaze position on the screen and the number and the type of objects around the avatar. The objects are classified into two categories: close and far. All the objects reachable by the avatar within the next action are labelled as close, otherwise as far.

[2] developed by LC Technologies, Inc. - www.eyegaze.com

The data is logged only during the time the participants play the game; this phase is preceded by for each player by the calibration of the gaze tracking system, a tutorial level and a demographics questionnaire.

5 Extracted Features from Data

The data collected for each game is sorted into three datasets according to the three area types described in Section 3. Each time a player completes an area two types of statistical features are extracted from that area: game-play and camera related features. The features of the first type are the experiment's independent variables and encapsulate elements of the player's behaviour in the area. The features of the second type describe the camera behaviour for each platform, therefore, they define the experiment's dependent variables.

The player's behaviour is defined by the actions the player performs in each area or, more precisely, by the consequences of these actions. Hence, the features extracted describe the interaction between the player and the game though the avatar's actions, rather than the sequences of pressed keys. For each area the following features have been extracted: the numbers of fuel cells collected, damage given, damage received, enemies killed, re-spawn points visited and jumps. The features are normalised to a range between 0 and 1 using a standard min-max uniform normalisation.

To model the camera behaviour, we analyse the content visualised by the camera instead of the camera absolute position and rotation. The presence of a certain object on the screen, however, does not necessarily imply an intentionality of the player; e.g. the object might be on the screen only because it is close to an object the player is interested to. The gaze data available permits to overcome this limitation since, using the gaze position, it is possible to understand which object is currently observed among the ones framed by the player. Therefore, we combine camera movements and gaze coordinates to identify the objects observed by the player at each frame and we extract the following statistical features: the relative camera speed as well as the time spent observing the avatar, the fuel cells close to the avatar, the enemies close to the avatar, the re-spawn points close to the avatar, the jump pads close to the avatar, the platforms close to the avatar and the far objects.

The seven features related to the time spent observing objects are calculated as the sum of the durations of the smooth pursuit and fixation movements of the eyes [18] during which the gaze position falls within an object's projected image. These values are normalised to $[0, 1]$ via the completion time of each area. The first feature is the average speed S of the camera relative to the avatar and it is defined as $S = (D_c - D_a)/T$, where D_c is the distance covered by the camera during the completion of an area, D_a is the distance covered by the avatar and T is the time spent to complete the area.

Each time the avatar leaves an area, the aforementioned features are logged for that area. The data is then, cleaned from experimental noise by removing all the records with a duration inferior to 3 seconds and the ones with no platforms

Collection Areas ($k = 2$)						
Records	Avatar	Fuel Cells	Jump Pads	Re-spawn Points	Far Objects	Speed
150	0.595	0.108	0.034	0.113	0.021	3.338
89	0.361	0.125	0.056	0.072	0.012	8.852

Fight Areas ($k = 3$)							
Records	Avatar	Fuel Cells	Coppers	Jump Pads	Re-spawn Points	Far Objects	Speed
137	0.674	0.042	0.095	0.049	0.034	0.036	3.283
99	0.676	0.032	0.478	0.044	0.056	0.025	5.293
142	0.250	0.029	0.069	0.030	0.052	0.013	5.927

Jump Areas ($k = 3$)					
Records	Avatar	Fuel Cells	Platforms	Far Objects	Speed
33	0.759	0.464	0.795	0.202	2.1293
80	0.736	0.166	0.658	0.059	2.7593
29	0.450	0.113	0.559	0.012	5.5854

Table 1: Average camera behaviour features with the number of records of each cluster. Speed indicates the average camera speed with respect to the avatar. The remaining features express the time spent observing each object of a type in an area divided by the time spent completing the area. Highlighted in dark grey is the feature related to the main task of the area type. The features related to the other objects close to the avatar are highlighted in light grey.

or no enemies and fuel cells left. The remaining records are sorted into three separate groups according to the area type and stored into three datasets, containing 239 records for the collection areas, 378 records for the fight areas and 142 records for the jump areas.

6 Camera Behaviour Modelling

To investigate and create a model of the relationship between camera behaviour and game-play, we analyse the collected data through two steps: identification and prediction. In the first step we use a clustering technique to extract the relevant camera behaviours and analyse their characteristics and then, in the second step, we build a model based on this categorisation able to predict the correct behaviour given a set of game-play data.

6.1 Behaviour Identification

The number of distinct camera behaviours as well as their internal characteristics can only be based, in part, on domain knowledge. One can infer camera behaviour profiles inspired by a theoretical framework of virtual cinematography [10] or alternatively follow an empirical approach — as the one suggested here — to derive camera behaviour profiles directly from data. The few existing frameworks focus primarily on story-driven experiences with little or no interaction, thus are not applicable in our context. Therefore, we adopt a data-driven approach and

we employ the k-means clustering algorithm on the gaze-based extracted features for the purpose of retrieving the number and type of different camera behaviours.

Unsupervised learning allows us to isolate the most significant groups of samples from each dataset. However, k-means requires the number of clusters k existent in the data to minimise the intra-cluster variance. To overcome this limitation, the algorithm runs with a progressively higher k value — from 2 to 10 — and the clusters generated at each run are evaluated using a set of five cluster validity indexes. The algorithm runs 50 times for each k and the run with the smallest within cluster sum of squared errors is picked. Each selected partition is evaluated against 5 validity indexes: Davis-Bouldin, Krzanowski-Lai, Calinski-Harabasz, Dunn and Hubert-Levin. The smallest k value that optimises at least 3 validity measures out of five is used for the clustering; the chosen k value is 2 for the collection type areas and 3 for the fight and jump type areas.

As seen in Table 1, the camera behaviour is described with a different feature set for each area type. The features are selected to match the visual stimuli offered by each area, thus only the features describing observation of objects which are present in the area type are included in the set. Moreover, for each area type the features are sorted into 5 categories: camera speed and time spent observing the avatar, the primary task objects, other close objects and far objects. The primary task objects highlighted in dark grey in Table 1, represent the time spent observing objects relative to the main task of each area type, all the other objects are categorised as secondary. According to this feature categorisation it is possible to observe three main behaviour types: *task focused* including the clusters spending more time observing the task related objects, *avatar focused* including the clusters mostly observing the avatar and *overview* which includes the clusters evenly observing all the objects in each area. For an in-depth description of the clusters the reader is referred to our previous study on camera behaviour modelling through gaze [12]

6.2 Behaviour Prediction

Once the relevant camera behaviour types are identified, we proceed by modeling the relationship between game-play and camera behaviour types. More precisely, since the model is intended to select the most appropriate camera behaviour that fits the player's preferences in the game, we attempt to approximate the function that maps the game-play behaviour of the player to the camera behaviour. For this purpose, we use Artificial Neural Networks (ANNs) which are chosen as a function known for its universal approximation capacity. In particular, we employ a different ANN for each area type, instead of one for the whole dataset, to be able to base each model on the best features necessary to describe the game-play in that area.

The three fully connected feed-forward ANNs are trained using Resilient Backpropagation [14] on the game-play data (ANN input) and the camera behavior clusters (ANN output) using early stopping for over-fitting avoidance. The networks employ the logistic (sigmoid) function at all their neurons. The performance of the ANNs is obtained as the best classification accuracy in 100

Area Type	Fuel Cells	Damage Given	Damage Received	Enemies Killed	Re-spawn Points	Jumps
Collection	**5.02**	-	-	-	1.23	0.76
Fight	1.63	**12.42**	**10.89**	**24.03**	0.49	**9.64**
Jump	**11.98**	-	-	-	-	0.53

Table 2: F-distribution values of the inter-cluster ANOVA test on the game-play features. The threshold for a 5% significance level is $F > 3.85$ for the collection areas and $F > 2.99$ for the jump areas. Values above the threshold appear in bold.

independent runs using 3-fold cross-validation. While the inputs of the ANN are selected algorithmically from the set of game-play features the ANN outputs are a set of binary values corresponding to each cluster of the dataset.

The exact ANN input features, the number of hidden neurons and the number of previous areas considered in the ANN input are found empirically through automatic feature selection and exhaustive experimentation. Sequential Forward Selection (SFS) is employed to find the feature subset that yields the most accurate ANN model. SFS is a local-search procedure in which a feature is added at a time to the current feature (ANN input) set until the accuracy of the prediction increases. Once the best feature set is selected, the best ANN topology is calculated through an exhaustive search of all possible combinations of neurons in two hidden layers with a maximum of 30 neurons per layer.

The combination of automatic feature and topology selection is tested on three different feature sets representing different time horizons in the past: input (game-play features) from one (*one step*) previous area visited in the past, input from the previous two areas visited (*two step*) and the combination of one previous area in the past with the average features of the rest of the previous areas visited (*one step + average*).

7 Results and Analysis

In this section we present and discuss the results in terms of prediction accuracy of the camera behaviour models created. First, a statistical analysis of the data is performed to check the existence of a relationship between camera behaviours and game-play features and to identify the significant ones. Then, the prediction accuracy of the models is evaluated with respect to the length of the time-series expressing the past which is considered in the ANN input, the selected feature set and the network topology.

To isolate the significant features among the ones logged, we perform an inter-cluster one-way ANOVA for each game-play feature to identify for which features we can reject the null hypothesis (no statistical difference exists).

As it is possible to see in Table 2, for each area type, at least one feature demonstrates a significant difference revealing the existence of significant linear relationships. In the fight areas dataset there is a significant difference in terms of

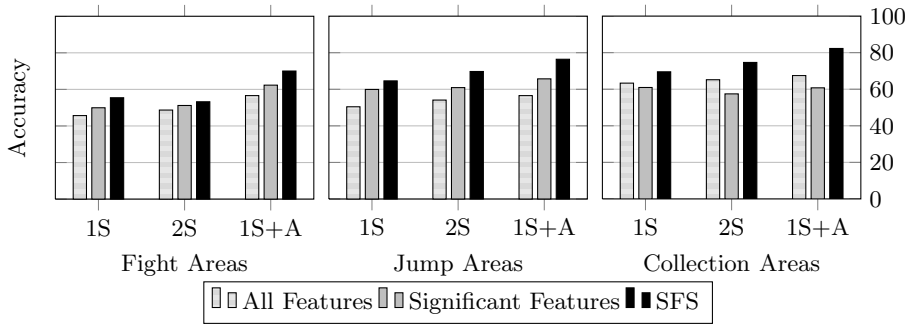

Fig. 3: Best 3-fold cross-validation performance obtain by the three ANNs across different input feature sets and past representations. The bars labelled 1S refer to the one step representation of the past trace, the ones la- belled 2S refer to the two step representation and 1S+A to the representation combining one previous step and the average of the whole past trace.

damage (both given and taken), number of killed enemies and number of jumps. In the other two area datasets the clusters differ significantly in the number of fuel cells collected.

The features highlighted by the ANOVA test reveal the existence of a linear relationship between the current camera behaviour and those features. However variable relationships, in general, are most likely more complex given that linearly separable problems are extremely rare. Thus, the aim of the analysis presented below is the construction of non-linear computational models of camera behaviour via the use of ANNs described in Section 6. Figure 3 depicts the best performance (3-fold cross validation) of 100 runs for each feature set on the three representations of the past events described in Section 6.

Each value shown in Figure 3 corresponds to the best topology found. It is noteworthy that all the selected topologies have at least one hidden layer, confirming the non linear nature of the relationship. This aspect is also highlighted, in Figure 3, by the difference in prediction accuracy between the ANNs that use the subset of significant features identified through the ANOVA test and the ANNs using the subset identified through SFS. The latter type of ANNs, yield a better accuracy regardless of the past representation and game areas. The best 3-fold cross-validation performances achieved for the fight, the jump, and the collection areas are, respectively, 70.04%, 76.43% and 82.29%. It is worth noting that in the collection areas, while the first type of ANNs, built solely on the features that are found significant by ANOVA, perform even worse than the ones using the full feature set, indicating that the linear analysis does not capture the relationship between game-play and camera behaviour accurately.

While, in the collection and jump areas, the ANOVA test indicates the number of fuel cells collected as the only significant feature, SFS selects the number of jumps and the number of re-spawn points activated as additional features for the ANN input. On the other hand, in the collection areas, SFS does not

only select features not indicated by ANOVA (the number of re-spawn points activated), but it also discards the number of jumps performed.

The results shown in Figure 3 confirm also our supposition that a more extensive representation of the past events would lead to a better accuracy. In fact, the best accuracies are achieved when the ANNs use the most extensive information about the past game-play events.

8 Conclusion

This article introduced the first step towards adaptive virtual camera control in computer games by proposing a model of camera behaviour and its relationship to game-play. Camera behaviour is modelled using a combination of information about players' gaze and virtual camera position collected during a game experiment. The data collected is sorted into three sets of areas according to the game-play provided to the player. For each group the data is clustered using k-means to identify relevant behaviours and the relationship between the clusters and the game-play experience is modelled using three ANNs. The evaluation of the ANN accuracy in predicting camera-behaviour is analysed with respect to the number ans type of features used as input to the model. The analysis reveals that the best prediction accuracies (i.e. 76.43% for jump, 82.29% for collection and 70.04% for fight) are achieved using a representation of the past events which includes the previous area and the average features of the other previously visited areas. Moreover, sequential feature selection reduces vector size which results in higher accuracies for all ANNs.

While the models constructed from the data show a prediction accuracy way above chance level, the analysis of collected data also suggests that the game is visually very complex and a multitude of objects, competing for the player's attention, generate noise in the data. This aspect could be limited by reducing the visual noise in the game. However, this would require to reduce the game complexity and, thus, also reduce the applicability to other games of the same genre. The actual version of the game incorporates visually all the standard features of modern 3D platformers and, besides the aforementioned limitations, the results give strong indications of the link between camera behaviour models and game-play in such games.

The same methodology could be used to construct camera behaviour models for different games such as role-playing games or action games. Models constructed using this methodology could be used to dynamically select the camera behaviour best suiting a certain player and therefore, generating a personalised game experience. For this purpose, it would be necessary to further investigate how to translate the camera behaviours identified through this methodology into camera profiles that could be used by an automatic camera control system. Moreover, in the future it would be interesting to investigate how to connect the proposed models to the detected cognitive and affective state of the player, in order to be able to influence aspects such as fun or attention.

References

1. D. Amerson, S. Kime, and R. M. Young. Real-time cinematic camera control for interactive narratives. In *International Conference on Advances in Computer Entertainment Technology*, pages 369–369, Valencia, Spain, 2005. ACM Press.
2. W. H. Bares, S. McDermott, C. Boudreaux, and S. Thainimit. Virtual 3D camera composition from frame constraints. In *ACM Multimedia*, pages 177–186, Marina del Rey, California, USA, 2000. ACM Press.
3. W. H. Bares, L. S. Zettlemoyer, D. W. Rodriguez, and J. C. Lester. Task-Sensitive Cinematography Interfaces for Interactive 3D Learning Environments. In *International Conference on Intelligent User Interfaces*, pages 81–88, San Francisco, California, USA, 1998. ACM Press.
4. M. Bernhard, E. Stavrakis, and M. Wimmer. An empirical pipeline to derive gaze prediction heuristics for 3D action games. *ACM Transactions on Applied Perception*, 8(1):4:1—-4:30, 2010.
5. O. Bourne and A. Sattar. Applying Constraint Weighting to Autonomous Camera Control. In *AAAI Conference On Artificial Intelligence In Interactive Digitale Entertainment Conference*, pages 3–8, 2005.
6. F. Charles, J.-l. Lugrin, M. Cavazza, and S. J. Mead. Real-time camera control for interactive storytelling. In *International Conference for Intelligent Games and Simulations*, pages 1–4, London, 2002.
7. D. Christianson, S. Anderson, L.-w. He, D. Salesin, D. Weld, and M. Cohen. Declarative Camera Control for Automatic Cinematography. In *AAAI*, pages 148–155. AAI, 1996.
8. M. Christie, P. Olivier, and J.-M. Normand. Camera Control in Computer Graphics. In *Computer Graphics Forum*, volume 27, pages 2197–2218, 2008.
9. S. M. Drucker and D. Zeltzer. Intelligent camera control in a virtual environment. In *Graphics Interface*, pages 190–199, 1994.
10. A. Jhala and R. M. Young. A discourse planning approach to cinematic camera control for narratives in virtual environments. In *AAAI*, number July, pages 307–312, Pittsburgh, Pennsylvania, USA, 2005. AAAI Press.
11. H. P. Martinez, A. Jhala, and G. N. Yannakakis. Analyzing the impact of camera viewpoint on player psychophysiology. In *International Conference on Affective Computing and Intelligent Interaction and Workshops*, pages 1–6. IEEE, Sept. 2009.
12. A. Picardi, P. Burelli, and G. N. Yannakakis. Modelling Virtual Camera Behaviour Through Player Gaze. In *International Conference On The Foundations Of Digital Games*, 2011.
13. D. Pinelle and N. Wong. Heuristic evaluation for games. In *CHI*, CHI '08, page 1453, New York, New York, USA, 2008. ACM Press.
14. M. Riedmiller and H. Braun. *A direct adaptive method for faster backpropagation learning: the RPROP algorithm.* IEEE, 1993.
15. V. Sundstedt, E. Stavrakis, M. Wimmer, and E. Reinhard. A psychophysical study of fixation behavior in a computer game. In *Symposium on Applied perception in graphics and visualization*, pages 43–50. ACM, 2008.
16. C. Ware and S. Osborne. Exploration and virtual camera control in virtual three dimensional environments. *ACM SIGGRAPH*, 24(2):175–183, 1990.
17. G. Yannakakis, H. Martinez, and A. Jhala. Towards Affective Camera Control in Games. *User Modeling and User-Adapted Interaction*, 2010.
18. A. L. Yarbus. *Eye movements and vision.* Plenum press, 1967.