# Using an Eye-Tracking System to Improve Camera Motions and Depth-of-Field Blur Effects in Virtual Environments

Sébastien Hillaire* Univ. of Rennes 1 / IRISA
Anatole Lécuyer† INRIA / LPPA
Rémi Cozot‡ Univ. of Rennes 1 / IRISA
Géry Casiez§ Univ. of Lille 1 / INRIA

**ABSTRACT**

This paper describes the use of user's focus point to improve some visual effects in virtual environments (VE).

First, we describe how to retrieve user's focus point in the 3D VE using an eye-tracking system. Then, we propose the adaptation of two rendering techniques which aim at improving users' sensations during first-person navigation in VE using his/her focus point: (1) a camera motion which simulates eyes movement when walking, i.e., corresponding to vestibulo-ocular and vestibulocollic reflexes when the eyes compensate body and head movements in order to maintain gaze on a specific target, and (2) a Depth-of-Field (DoF) blur effect which simulates the fact that humans perceive sharp objects only within some range of distances around the focal distance.

Second, we describe the results of an experiment conducted to study users' subjective preferences concerning these visual effects during first-person navigation in VE. It showed that participants globally preferred the use of these effects when they are dynamically adapted to the focus point in the VE. Taken together, our results suggest that the use of visual effects exploiting users' focus point could be used in several VR applications involving first-person navigation such as the visit of architectural site, training simulations, video games, etc.

**Keywords:** eye-tracking, visual feedback, depth-of-field blur, camera motion, focus point, first-person-navigation

**Index Terms:** H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities H.5.2 [Information Interfaces and Presentation]: User Interfaces—Interaction styles, User-centered design

## 1 INTRODUCTION

The point the user's eyes are looking at on a screen, called the focus point, is an important human feature widely used in several domains. As an example, in neurology, Iijima et al. [7] diagnose Parkinson's disease by analyzing the movement of the focus point: healthy patients generate a soft and continuous movement whereas movements of diseased patients are unstable.

In the field of Human-Computer Interaction, new interaction techniques using users' focus point as input have emerged. Sibert and Jacob [9] described them as a "convenient and natural addition to user-computer dialogues". As an example, they used the user's focus point as a pointer instead of a computer mouse. It resulted in fast entity selection [9].

In Virtual Reality, the focus point can first be used to improve navigation in the virtual world. In first-person navigation mode, Smith et al. [10] changed the camera orientation to match users' viewing direction in the VE in order to turn the camera when walking. In a third-person navigation mode, they proposed to automatically move the user's avatar at the position he/she is looking at.

Furthermore, the focus point can also be used in the rendering process. Humans perceive more details close to the focus point on the screen than at its periphery. Some rendering algorithms take into account this property to reduce the complexity and decimate some parts of the virtual scene depending on the distance to the focus point [6][8]. Luebke et al. [8] proposed an algorithm which simplifies the geometry of a scene and ensures that the user does not perceive any visual difference. Levoy [6] described a ray-casting algorithm, also based on the focus point, for volumetric data visualization. This algorithm uses distance to the focus point on screen to define the number of rays thrown per pixel and select the adapted volumetric texture mipmap level.

In this paper, we propose to improve the visual feedback of VE by using user's focus point as input. We focus on first-person navigation in VE, i.e., the camera used to display the VE is placed at the level of the eyes of the user's avatar. In this situation, we propose two additional effects: (1) a camera motion which simulates movements of head and eyes when walking and (2) a Depth-of-Field blur effect which simulates the fact that humans perceive sharp objects only within some range of distances around the focal distance.

In the remainder of this paper, we first detail the algorithm and the architecture we propose to compute in real time the focus point and the focal distance in the VE. Then we describe two techniques to improve the visual feedback of VE using focus point and focal distance information: a compensated camera motion and a DoF blur effect. Finally, we report on an experiment conducted to study subjective preference regarding these effects during first-person navigation in VE. The paper ends with a discussion and a general conclusion.

## 2 COMPUTATION OF FOCAL DISTANCE AND 3D FOCUS POINT

In this section, we describe an algorithm in four steps to compute both the focal distance $f_d$ and focus point $\mathbf{fp_{ve}}$ in the 3D VE, using the output of an eye-tracking system, i.e., the focus point on the screen $\mathbf{fp_{scr}}$. The architecture of our algorithm is illustrated in Figure 1.

1. **eye-tracking**: we simply recover the position of the users' focus point on screen $\mathbf{fp_{scr}}$ corresponding to the eye-tracker's 2D output.

2. **low-pass filtering**: we compute $\overline{\mathbf{fp_{scr}}}$ which corresponds to the low-pass filtered $\mathbf{fp_{scr}}$ using a cut-off frequency of 15Hz to avoid high frequency jerks.

3. **auto-focus system**: the accuracy of the eye-tracker's output is unfortunately superior to one pixel. Thus, in some cases, the user might look at other pixels located near $\overline{\mathbf{fp_{scr}}}$, or to the shape of an object close to $\overline{\mathbf{fp_{scr}}}$. The depth of a single pixel does not seem sufficient to estimate $f_d$. Thus, we use the same auto-focus system, as described previously in [2], with a square focus zone centered on $\overline{\mathbf{fp_{scr}}}$, as shown in Figure 2. The computation of $f_d$, using the pixels located inside

*e-mail: shillair@irisa.fr
†e-mail: anatole.lecuyer@irisa.fr
‡e-mail: cozot@irisa.fr
§e-mail: gery.casiez@lifl.fr

IEEE Virtual Reality 2008
8-12 March, Reno, Nevada, USA
978-1-4244-1971-5/08/$25.00 ©2008 IEEE

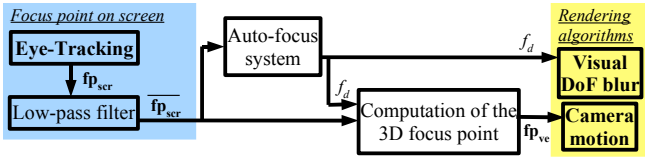


Figure 1: Software architecture.

the focus zone, is achieved by averaging the weighted depth of each pixel in the focus zone. The weight of each pixel is computed using the pixel semantic value and the distance to $\overline{\mathbf{fp_{scr}}}$. The closer the pixel is to $\overline{\mathbf{fp_{scr}}}$, the higher the weight is. If the pixel corresponds to a semantically important object (e.g., a target) its weight is also increased. For further details, see Hillaire et al. [2].

4. **focus point in VE**: first, we compute user's viewing direction $\mathbf{V_{cam}}$ expressed in the camera reference frame using $\overline{\mathbf{fp_{scr}}}$. Then, we transform it into the reference frame of the VE to obtain $\mathbf{V_{ve}}$. Finally, we can compute user's 3D focus point in VE $\mathbf{fp_{ve}}$ using Equation 1, in which $\mathbf{C}$ is the camera position and $f_d$ the focal distance computed in the previous step.

$$\mathbf{fp_{ve}} = \mathbf{C} + f_d \times \mathbf{V_{ve}} \tag{1}$$

Finally, the computed focus point and focal distance can be sent to the two rendering algorithms described in the following sections: a camera motion and a DoF blur effect.

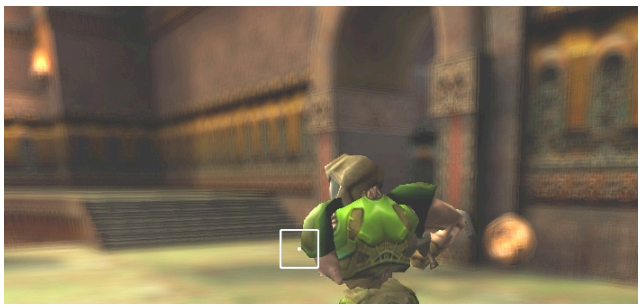

Figure 2: Quake III video game with Depth-of-Field blur effect implemented. The white square corresponds to the auto-focus zone that follows the user's focus point measured by eye-tracking.

## 3 Camera Motion for First-Person Navigation Based on the Focus Point

Camera motion is sometimes used in first in First-Person-Shooter games. It generally consists in applying a sinusoidal motion to the virtual camera to simulate the visual flow corresponding to a walking motion. Lécuyer et al. [5] suggested to improve the classical techniques by changing the camera orientation in addition to the change in position. This change in camera orientation is supposed to simulate eye movements and ocular compensation [4]. However, they did not use an eye-tracking system and the camera was always focusing on an object located at the center of the screen. Moreover, they did not provide further details on how to implement such camera motion with an eye-tracking system.

The first step of the computation of the camera motion consists in changing the camera position to simulate walking motion. We apply three sinusoidal offsets to the original camera position (see Figure 3A) on three axes: up vector of the VE $\mathbf{z_{ve}}$, left vector $\mathbf{y_c}$ and forward vector $\mathbf{x_c}$ of the camera. We compute the amplitude of each offset on each axis using Equation 2:

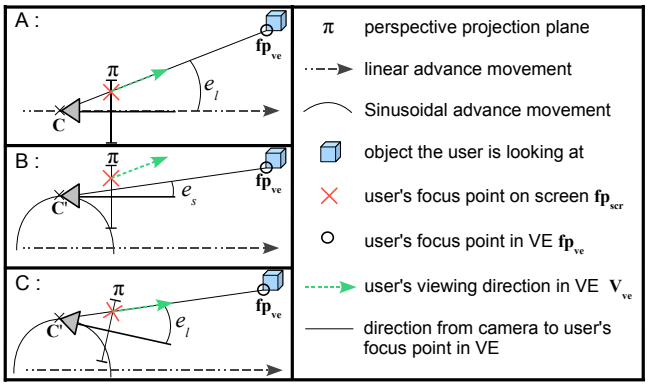


Figure 3: computation of the camera orientation using user's focus point.

$$l_a = K_a \times \sin(2\pi \times \frac{t}{T_a} + O_a), \forall a \in \{\mathbf{x_c}, \mathbf{y_c}, \mathbf{z_{ve}}\} \tag{2}$$

where $t$ is the time and $a$ one of the three axis. For each axis $a$, $T_a$ is the period of walking motion, $K_a$ a constant amplitude, $O_a$ the initial phase and $l_a$ is the resulting amplitude of the offset for the current frame. The final camera position $\mathbf{C}'$ (see Figure 3B) is obtained using Equation 3.

$$\mathbf{C}' = \mathbf{C} + l_{x_c} \times \mathbf{x_c} + l_{y_c} \times \mathbf{y_c} + l_{z_{ve}} \times \mathbf{z_{ve}} \tag{3}$$

The second step of our computation consists in changing the orientation of the camera. We compute the change in orientation of the camera so to keep $\mathbf{fp_{ve}}$ projected on $\overline{\mathbf{fp_{scr}}}$ on the projection plane $\pi$.

Let us note $(\alpha, \beta, \gamma)$ the roll, pitch and yaw angles which represent the original camera orientation and $(\alpha', \beta', \gamma')$ the angles of the desired camera orientation as shown in Figure 3C. We compute the final orientation using Equation 4:

$$\begin{pmatrix} \alpha' \\ \beta' \\ \gamma' \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} + \left( \begin{pmatrix} 0 \\ e_s \\ a_s \end{pmatrix} - \begin{pmatrix} 0 \\ e_l \\ a_l \end{pmatrix} \right) \tag{4}$$

where $(e_l, a_l)$ are elevation and azimuth angles of $\mathbf{fp_{ve}}$ in camera reference frame when following the initial linear motion as shown in Figure 3A, i.e., before offsets are applied to camera position $C$. Then, $(e_s, a_s)$ are elevation and azimuth angles of $\mathbf{fp_{ve}}$ in the camera reference frame at position $\mathbf{C}'$ in Figure 3B. In our final implementation, $\alpha$ and $\alpha'$ angles were set to 0. Our other parameters were set using preliminary testing to: $(T_{x_c}, T_{y_c}, T_{z_{ve}}) = (0ms, 780ms, 390ms)$, $(K_{x_c}, K_{y_c}, K_{z_{ve}}) = (0, 0.015, 0.02)$ and $(O_{x_c}, O_{y_c}, O_{z_{ve}}) = (0, \pi/2, 0)$.

The resulting camera motion is thus compensated in real time based on the point the user is looking at in the VE, thanks to the eye-tracking system.

## 4 Depth-of-Field Blur Effect Based on the Focus Point

The second effect we propose to improve visual feedback using users' focus point is a DoF blur effect. Brooker et al. [1] conducted an evaluation of DoF blur effect using a stereoscopic display. However, their results did not show evidences of performance improvement when the DoF effect was computed using an eye-tracker. They concluded that it could be due to the very slow frame rate of their application and they suggested to implement and further evaluate real time DoF blur effect in VE. In a previous study [2], it was

shown that, in absence of an eye-tracking system, only a half of the participants enjoyed a DoF blur effect computed with a focus zone constantly positioned at the center of the screen. It was suggested that an eye-tracking system could improve the results. Thereafter, we describe the implementation of a DoF blur effect adapted in real time to user's focal distance in VE thanks to an eye-tracking system and to the algorithm described in section 2.

The computation of the DoF blur effect is achieved using the classical techniques described in [2]. We simulate a part of the human visual system: focal distance accommodation using a low-pass filter and DoF blur. To compute the amount of blur per pixel we use a lens model that takes into account the focal distance. Then, by applying a gathering blur technique, we obtain the final blurred image without the problem of color leaking, thanks to depth comparisons. The implementation parameters are the same as in [2].

Figure 2 illustrates our algorithm implemented in real time in the Quake III video game engine [3]. Thanks to the focus zone centered on $\mathbf{fp_{scr}}$, the focus is done on the semantically important character instead of the background, even if the character covers few pixels in the auto-focus zone.

## 5 Preliminary Conclusion

We have enhanced two existing techniques suitable for first-person navigation in VE to make them work with an eye-tracking system: (1) a compensated camera motion and (2) a DoF blur effect. They have been successfully implemented in the real time open-source engine of Quake III video game [3][1]. On a PC with an Intel PentiumD CPU at 3.4Ghz, 2.0Gb of RAM and a nVidia Quadro FX 3450/4000 SDI, performance was of 70 frames per second with the both effects activated, instead of 260 without, at a resolution of $1280 \times 1024$ pixels. The video game with our techniques implemented and the hardware configuration aforementioned were used in the experiment described in the next section.

## 6 Experimental evaluation

An experiment has been conducted to study the subjective preference of users regarding the DoF blur effect and camera motion when eye-tracking is enabled or not.

### 6.1 Apparatus

We used our camera motion and DoF blur effect implemented in the video game Quake III as described in section 3 and section 4. In all test sessions, we used Q3DM7 map. The player was alone (no targets) and could move freely. No information was displayed on the screen but the classical central visor. The initial altitude of the camera was set to 50 units.

We used an immersive room with a cylindrical screen (3.8m radius, 2.35m height and 45 degree of arc) with a display resolution of $1280 \times 1024$ pixels. The participants were positioned at the center of the cylindrical screen and no sound was played during sessions.

We computed participant's focus point position on screen by using ASL 6000 eye-tracker system with head-mounted optics. We used a chin-rest to maintain participant's head at the same position.

### 6.2 Participants

Eight men with a mean age of 25,8 (SD=4,3) participated in our experiment. All subjects were familiar with first-person navigation and had normal or corrected vision with lens. Five participants were right handed. All subjects were unaware of our work.

### 6.3 Experimental plan

First, each participant navigated in the VE during 3 minutes as a training session. Then, the experiment was divided into two parts.

[1] Our sources are available at www.irisa.fr/bunraku/eye/

The first part consisted in testing four conditions of camera motion: (1) Control (basic linear motion as if the user was driving a car) ; (2) Mv (sinusoidal movement without motion compensation) ; (3) Mv_comp, (sinusoidal movement with motion compensation computed using a focus zone constantly located at the center of the screen) ; (4) Mv_comp_eyeT (sinusoidal movement with motion compensation computed using a focus zone centered on user's focus point).

The second part consisted in testing three conditions of DoF blur effect: (1) Control (normal scene rendering without DoF blur) ; (2) DOF (scene rendered with DoF blur effect computed using a focus zone constantly located at the center of the screen) ; (3) DOF_eyeT (scene rendered with DoF blur effect computed using a focus zone centered on user's focus point on screen).

For each part and each participant, the presentation order of each condition was randomized. At the beginning of each part, the eye-tracker was calibrated and participants were informed of what is going on in each condition: the type of camera motion or DoF blur effect used, if we used eye-tracking or not, etc. Each condition was tested during 5 minutes and the experiment lasted 45 minutes. The eye-tracker was worn during the whole experiment.

At the end of each part, participants were asked to fill up a global appreciation questionnaire during which they were free to test each condition at runtime by selecting them using keyboard keys.

### 6.4 Results

The data were analyzed using the Friedman test followed by Wilcoxon post-hoc analyses.

#### 6.4.1 Camera Motion

Participants were asked to grade the four conditions of camera motion using a seven-point Likert scale according to four subjective criteria: (1) impression of walking in the virtual world, (2) "fun", (3) depth perception and (4) immersion in the virtual world. The results are displayed in Figure 4.

Friedman analysis revealed an overall significant difference between the techniques for the **impression of walking** in the VE ($\chi^2(3)=18.8$, $p<0.0001$). Bonferroni-corrected Wilcoxon post-hoc analyses confirmed that all sinusoidal movements are significantly preferred compared to the basic linear motion ($p<0.004$). Marginal significant difference between Mv and Mv_comp_eyeT techniques (z-score=-1.8, p=0.047) suggest that the compensation improves the impression of walking only when the focus point is used. Friedman analysis showed a significant main effect across navigation techniques in term of **fun** ($\chi^2(3)=13.6$, $p<0.004$). Post-hoc analyses revealed that the Mv_comp_eyeT technique is significantly different from the others ($p<0.02$). A significant difference between Mv_comp and Control (z-score=-2.3, p=0.012) shows that the motion compensation improves the navigation in term of fun. Friedman analysis showed a significant main effect across techniques for the **depth perception** ($\chi^2(3)=12.8$, $p<0.005$). Post-hoc analyses revealed that the Mv_comp and Mv_comp_eyeT techniques are significantly different from the Control ($p<0.03$). Friedman analysis showed a significant main effect across techniques for the **immersion** in the virtual world ($\chi^2(3)=15.9$, $p < 0.001$). Post-hoc analysis revealed significant difference between Mv_comp_eyeT and all other techniques ($p<0.02$). Significant difference were also found between Control and all other techniques ($p<0.03$) and between Mv and Mv_comp_eyeT (z-score=-2.2, p=0.016).

#### 6.4.2 Depth-of-Field blur

Participants were asked to grade the three conditions of DoF blur effect using a seven-point Likert scale according to four subjective criteria: (1) rendering realism, (2) "fun", (3) depth perception and (4) immersion in the virtual world. The results are displayed in Figure 5.

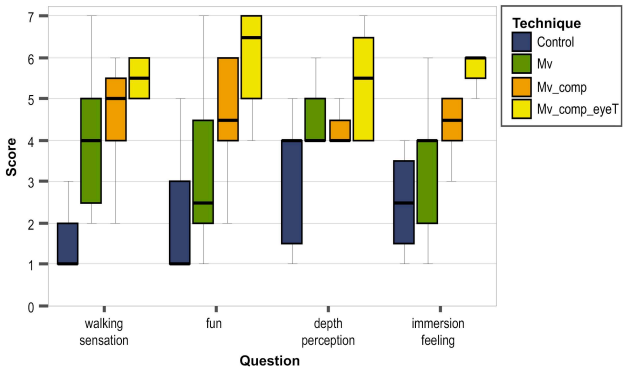


Figure 4: Box plot representation of the camera motion subjective preference for each question and each technique.

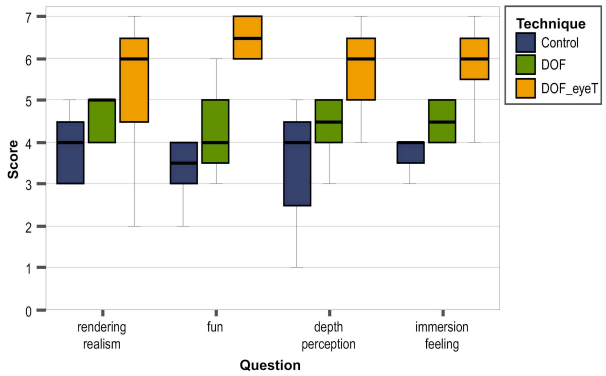


Figure 5: Box plot representation of the Depth-of-Field blur subjective preference for each question and each technique.

Friedman analysis showed no overall significant difference between the techniques for the **rendering realism** ($\chi^2(2)$=5.4, p=0.067). Friedman analysis revealed an overall significant difference between the techniques in term of **fun** ($\chi^2(2)$=12.1, p=0.002). Bonferroni-corrected Wilcoxon post-hoc analyses confirmed a significant difference between DOF_eyeT and the two other techniques ($p<0.01$). Friedman analysis showed a significant main effect across techniques for the **depth perception** ($\chi^2(2)$=8.3, p=0.016). Post-hoc analysis revealed significant difference between DOF_eyeT and the two other techniques ($p<0.05$). Friedman analysis showed a significant main effect across techniques for the **immersion** in the VE ($\chi^2(2)$=13.2, $p<0.001$). Post-hoc analysis revealed significant difference between all techniques ($p<0.03$).

### 6.5 Discussion

First, concerning the camera motion effect, we found similar results as Lécuyer et al. [5]. Indeed, here also, sinusoidal camera motions are preferred compared to the linear motion. When eye-tracking is not used, the addition of a motion compensation (Mv_comp) is marginally preferred to the sinusoidal camera motions but, as in [5], the result is not significant. Then, we found here that the results are better when the compensated camera motion is computed using eye-tracking. Indeed, with the eye-tracking system in use, the participants significantly felt more fun and more immersed in the virtual world. The preference failed to reach significance concerning depth perception and walking sensation, possibly due to the small number of participants. Besides, some participants noticed that the difference between Mv_comp and Mv_comp_eyeT conditions was difficult to perceive, except when they were close to virtual objects. This is due to the fact that variations of camera orientation are stronger when the focal distance is close to the eyes.

Concerning the DoF blur effect, the analysis shows a strong influence of using the eye-tracking system. As in [2], the DOF condition is not clearly preferred compared to the Control condition. The DoF blur effect seems to be preferred only when eye-tracking is used (DOF_eyeT condition). The eye-tracking condition was significantly preferred in terms of fun and immersion feeling. Furthermore, participants felt that they better perceived the depth of the virtual scene using this condition. One participant even said "Sometimes, it's like looking at an auto-stereoscopic screen". It suggests that the DoF blur effect could be used to convey additional information concerning spatial relations between objects, like shadows do.

## 7 Conclusion

We have described the use of user's focus point on screen to dynamically adapt visual effects during first-person navigation in VE. We have proposed (1) a compensated camera motion and (2) a DoF blur effect which are both adapted in real time using user's focus point in the VE. An experiment was then conducted to study users' preferences regarding these two effects during first-person navigation. It showed that participants globally preferred these effects when they are computed using an eye-tracking system. In this case, these effects were found to improve sensations and perception of the VE and it resulted in better immersion in the VE.

**Acknowledgements**: this work was supported by the French National Agency of Research through the PERF-RV2 project.

## References

[1] J. P. Brooker and P. M. Sharkey. Operator performance evaluation of controlled depth of field in a stereographically displayed virtual environment. *Stereoscopic Displays and Virtual Reality Systems VIII*, A. J. Woods, M. T. Bolas and J. O. Merritt (Eds), 4297(1):408–417, 2001.

[2] S. Hillaire, A. Lécuyer, R. Cozot, and G. Casiez. Depth-of-field blur effects for first-person navigation in virtual environments. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 203–206, 2007.

[3] IdSoftware. Quake 3 arena. source code v1.32b under GPL, www.idsoftware.com, 2006.

[4] T. Imai, S. T. Moore, T. Raphan, and B. Cohen. Interaction of the body, head, and eyes during walking and turning. *Experimental Brain Research*, 136(1):1–18, 2001.

[5] A. Lecuyer, J.-M. Burkhardt, J.-M. Henaff, and S. Donikian. Camera motions improve the sensation of walking in virtual environments. In *Proceedings of the IEEE conference on Virtual Reality*, pages 11–18, 2006.

[6] M. Levoy and R. Whitaker. Gaze-directed volume rendering. In *Proceedings of the ACM symposium on Interactive 3D graphics*, pages 217–223, 1990.

[7] A. Iijima, M. Haida, N. Ishikawa, H. Minamitani, and Y. Shinohara. Head mounted goggle system with liquid crystal display for evaluation of eye tracking functions on neurological disease patients. In *Proceedings of the IEEE conference on Engineering in Medicine and Biology Society*, volume 4, pages 3225–3228, 2003.

[8] D. P. Luebke and B. Hallen. Perceptually-driven simplification for interactive rendering. In *Proceedings of the ACM Eurographics Workshop on Rendering Techniques*, pages 223–234, 2001.

[9] L. E. Sibert and R. J. K. Jacob. Evaluation of eye gaze interaction. In *Proceedings of the ACM SIGCHI conference on Human factors in computing systems*, pages 281–288, 2000.

[10] J. D. Smith and T. C. N. Graham. Use of eye movements for video game control. In *Proceedings of the ACM conference on Advances in Computer Entertainment Technology*, pages 20–28, 2006.