Immersion: The Challenge for Commodity Gaming

Paul Bourke
iVEC@UWA
The University of Western Australia
Perth, West Australia 6009, Australia
Email: paul.bourke@uwa.edu.au

***Abstract*: Presented here is a case for a higher level of engagement and immersion in gaming by proposing gaming software support for displays that engage the players peripheral vision. It is widely accepted that the sense of "being there" or "presence" is heightened by "removing the frame", that is, filling the viewers field of view with a virtual environment [1]. Except for a few special cases this is almost entirely unsupported in current commodity games, and yet the importance is readily accepted as is illustrated by the widespread use in commercial simulators and virtual reality environments. This paper will present the general requirements necessary to support a range of seamless immersive displays, that is, displays that engage a significant portion, if not all, of the users field of view. It will further outline a software pipeline split into two parts, a generic section that generates the necessary visual information followed by a section that handles the device/display specific details.**

Keywords: Immersion, virtual reality, peripheral vision, surround display, gaming, engagement, presence.

I. INTRODUCTION

A wide range of game genres place the player within a 3D environment. Not always but often it is an environment that is familiar to us, that is, modeled after our real 3D world. FPS (First Person Shooters), flight simulators, and racing games for example bear a strong similarity to training simulators in industry or the military [2]. Simulators in general take their lead from what is traditionally called virtual reality a key characteristic of which is the user does not perceive any of the real world, the virtual environment fills their entire field of view (FOV) and is often additionally presented in stereoscopic 3D. In time the visual quality of games is getting closer and closer to the real world as the technology improves, for example the graphical realism, frame rate performance, audio quality, and physics engine. If the gaming experience is constrained to flat displays then this may become the limit to the extent that the gaming experience can approach our real world experience.

A key difference between commodity gaming and commercial simulators and training systems is the degree to which the player is immersed. Many games support stereoscopic presentation on a single planar surface, to dubious benefit [3][4], but very few commercially available games are able to engage the full human field of view. This mismatch between commercial immersive environments and commodity games could be argued to be due to the need for space and display devices outside the budget of gamers. This doesn't explain the almost total absence of support in commodity games, noting in particular that what is spent on hardware by some gamers and the disposable income of a significant population of mid-life gamers [5]. With the increased commoditisation of data projectors for the home theatre market, the price point for creating surround projected environments is reasonably low [6].

One example of an attempt at exploiting the peripheral vision of games is the jDome [7]. The jDome uses unmodified games and as such it is noteworthy that despite the extreme distortion that occurs in the wide peripheral zones, players report a heightened immersive experience.

Head mounted displays (HMDs) might be one opportunity to provide a simultaneous stereoscopic and immersive device for gaming [8]. They additionally offer a tight coupling between head movement and view direction greatly enhancing the sense of viewing interaction in the virtual space. They also block out any of the real world. Unfortunately HMDs, despite their promise over the last 2 decades, have failed to deliver on providing a wide field of view. Limitations in the fabrication process currently prevents small seamless curved displays with sufficient resolution. The solution in the military training space has been to tile multiple small flat displays [8], this unfortunately leads to further issues such as the interface between the panels, calibration of panels, and the increased number of graphics pipes required to drive the tiled grid of displays.

Connecting multiple flat panel displays together has been one attempt by gamers to provide "look around", these panels can be placed such that they wrap around the player. A number of products have arisen to support this approach, including the Matrox dual and triple head splitter units and more recently the multiple head (up to 6) pipes on the AMD FirePro graphics cards. There are a number of limitations of this approach. While the width of the bezels of some products are reducing there is inevitably a distracting gap between the displays. Except in special cases where the gaps can be made to coincide with structure in the game (for example, frame of the plane window in a flight simulator), we don't always view our world through rectangular window frames so doing so in the gaming context is not optimal. More importantly perhaps is that these discrete tiled panels don't fill the players field of view. A common configuration of 3 horizontal displays might achieve 100 degrees horizontally but it doesn't provide a significant vertical FOV, note that studies have identified 100 degrees as a minimum for immersion [1]. It should also be noted that the ad-hoc addition of displays to a game, if all the displays are not

lying in a single plane also requires explicit support in the game for multiple view frustums. Without this explicit support the imagery on multiple non-planar displays is distorted. Figure 1 illustrates the frustums required for three displays angled around the player. The correct frustums shown do not give the same result as a single wide perspective frustum encompassing the extreme corners of the display achieved say by increasing the horizontal field of view accordingly. In addition the frustums depend on the player position, this is critical for displays that surround the viewer but strictly speaking also required for a single display. While support for arbitrary view frustums and careful positioning of the view position with respect to the display is standard with virtual reality applications it is almost never supported in games. This dependence of the players position with respect to the display is clear in figure 2 where the imagery in the iDome appears distorted, this arises because the camera is not located at the position the content was created for, namely at the players position. This effect is well known for stereoscopic 3D content creation but also true for monoscopic immersive displays. The Google Earth Liquid Galaxy [9] while not strictly a game is an example where such support is provided.

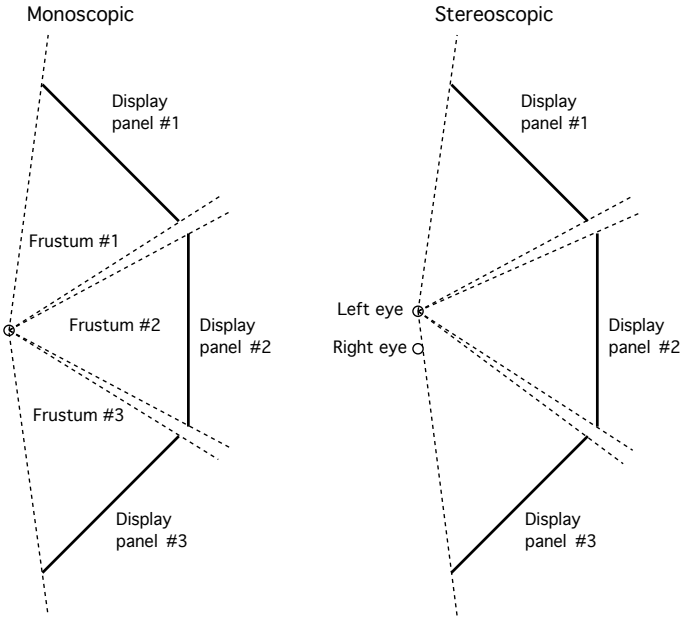

Figure 1. Three distinct view frustums required for three discrete monoscopic display panels. Three frustums required for each eye for three stereoscopic displays (only frustums for the left eye are shown).

In what follows, the requirements for games that can exploit the full human visual field will be discussed. The requirements will be illustrated with two case studies: the iDome [10], a single projector based hemispherical display and a multiple projector cylindrical display [11]. These are chosen because they provide for a seamless display and are based upon standardised geometry for surround displays [12] in virtual reality installations.

I. Example: Dome

The iDome is a small personal display based upon a hemisphere, similar to a traditional planetarium dome but orientated in such a way as to provide a view orientation more suited to a forward looking gaming experience. The iDome is presented here primarily as an example of a range of hemispherical display configurations for which a fisheye projection is the natural image format that encapsulates the visual information required. The challenge is that fisheye projections are not directly supported in the two real time graphics APIs, namely DirectX and OpenGL. The standard perspective projection cannot represent a field of view of 180 degrees, indeed it becomes increasingly inefficient above 120 degrees. Since these realtime APIs only support orthographic and perspective projections, they are not immediately suited to single pass renderings that require more than 120 degrees of FOV, the iDome requires 180 degrees horizontally and 140 degree vertically.

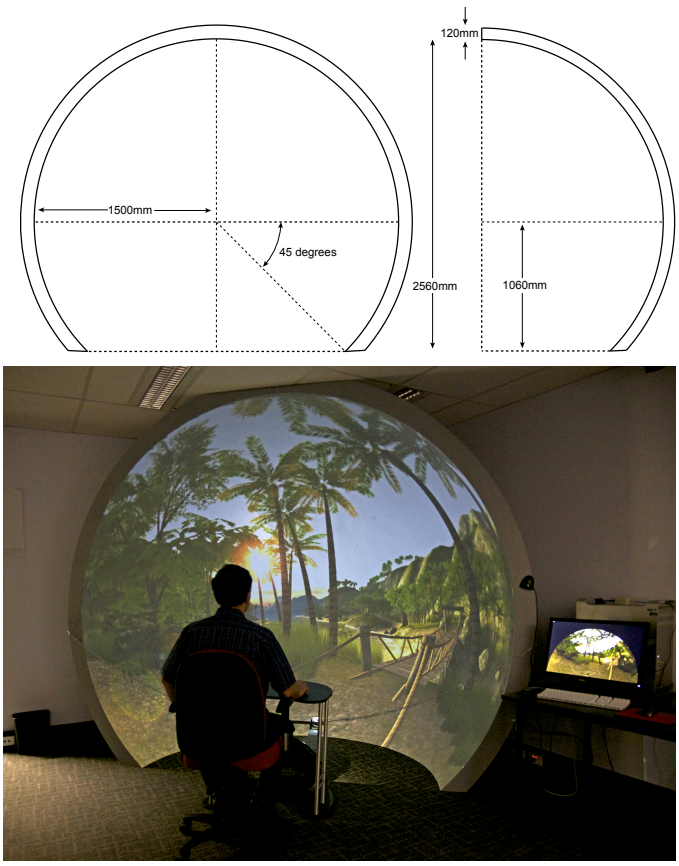

Figure 2. Top: Plans for the iDome. Bottom: The Unity3D demonstration game in the iDome. Note the apparent distortion arises because the camera is offset from the front of the dome, from the players perspective there is no distortion.

The solution at the fundamental level requires that sufficient visual information is captured, only once that is achieved can the content for the display be correctly formed. The approach proposed here requires that the 3D scene is rendered multiple times each with a camera pointing in different directions. There are an infinite number of ways one might do this but there is a standard and historical approach that offers a trade-off between the number of render passes used and the image resolution acquired from each pass. The solution uses four perspective cameras, each located at the center of a unit cube surrounding the camera and with frustum edges through each vertex of the faces of the cube. Once these four passes have been rendered the four images can be remapped to create a seamless and

artifact free fisheye projection, the standard image projection used that has the visual information required for a dome display, see figure 3.

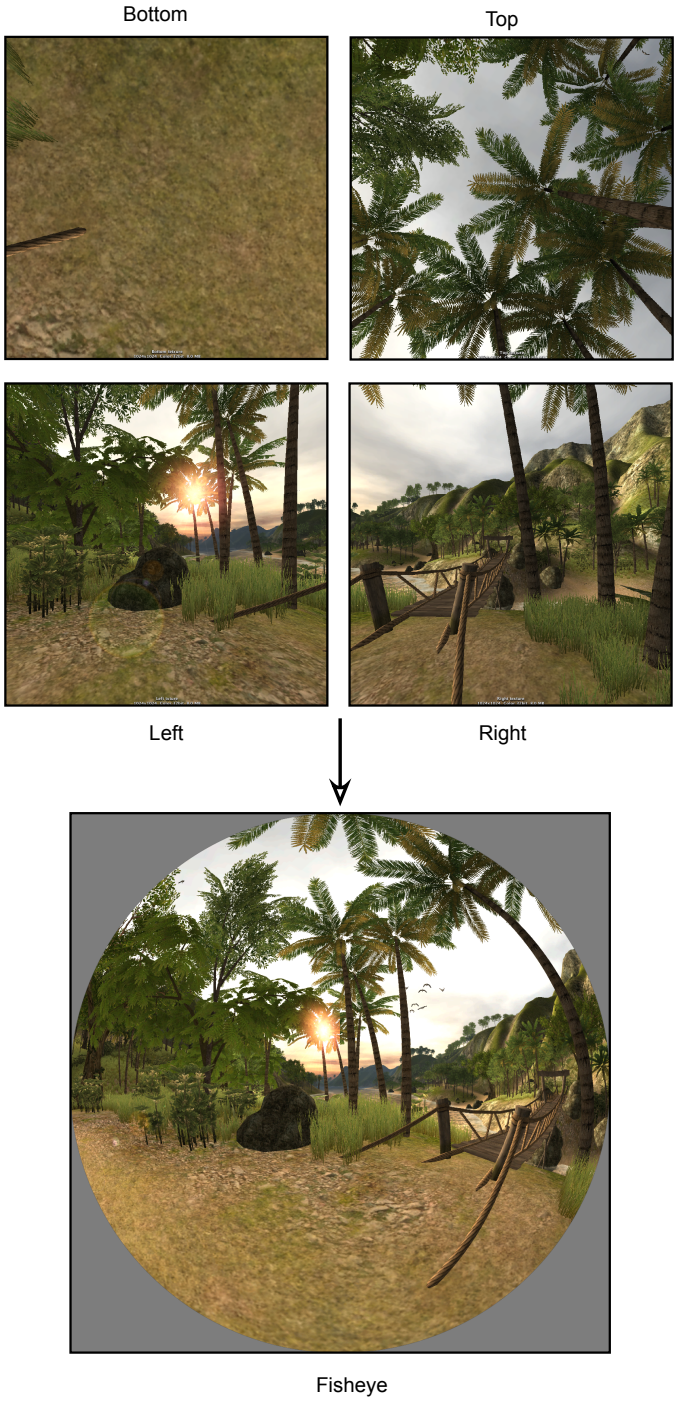

Figure 3. Four pass rendering to create the visual field necessary for a fisheye projection. Note the grey region is not visible in the display.

Such a technique is by no means new, it has been used in a range of algorithms in computer graphics [13][14] for a few decades. The advantage in this context is that the capture of N faces of a cube, where N is the smallest number necessary to capture the required field of view, is straightforward to implement in the rendering pipeline. If all six faces are captured then any camera view (from that position) can be synthesised as no more than an image mapping. Once one has the required visual information then any display specific image can be derived. In the case of the iDome this involves a mapping from the four cube faces image mapped to a fisheye image. This is a single texture warping operation [13] and is generally performed with a negligible performance hit on modern graphics hardware, the two main real time APIs support the required "render to texture" capability.

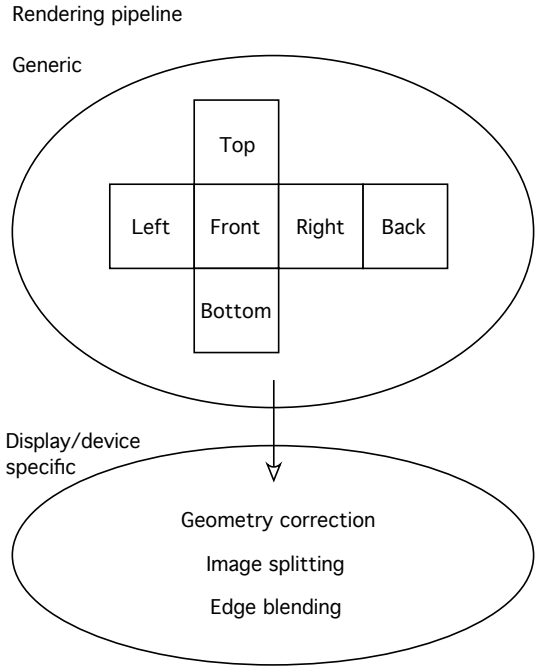

Figure 4. Rendering pipeline separating out the generic cube image map rendering and the device/display specific components.

If the dome employs a fisheye lens attached to the data projector then this image is projected as it stands. In the case of the iDome there is an additional geometry correction to compensate for the distortion introduced by the spherical mirror [10]. If the dome has multiple projectors then there is an additional slicing, geometry correction, and edge blending stage. The necessary support for these is discussed for the example of a cylindrical display in the next section.

The general pipeline and support required in games is shown in figure 4. It has been split into two parts, the first is a generic multipass rendering stage. The number of cube faces rendered depends on the FOV of the final display surface(s). This is a single parameter preference setting and while not enough in itself, it does provide the base image information from which any device specific projections can be formed.

II. Example: Cylindrical Display

Cylindrical displays [11] are categorised by their horizontal and vertical FOV. In contrast to the iDome they generally employ multiple projectors, the consequence of which is that in addition to the geometry correction they also require multiple output pipes and edge blending. The example discussed here is for a 180 degree horizontal FOV and 80 degree vertical FOV. In this case only two of the possible six cube faces are

necessary, as with the iDome the cube is orientated with the camera direction pointing towards the midpoint of the edge of the cube.

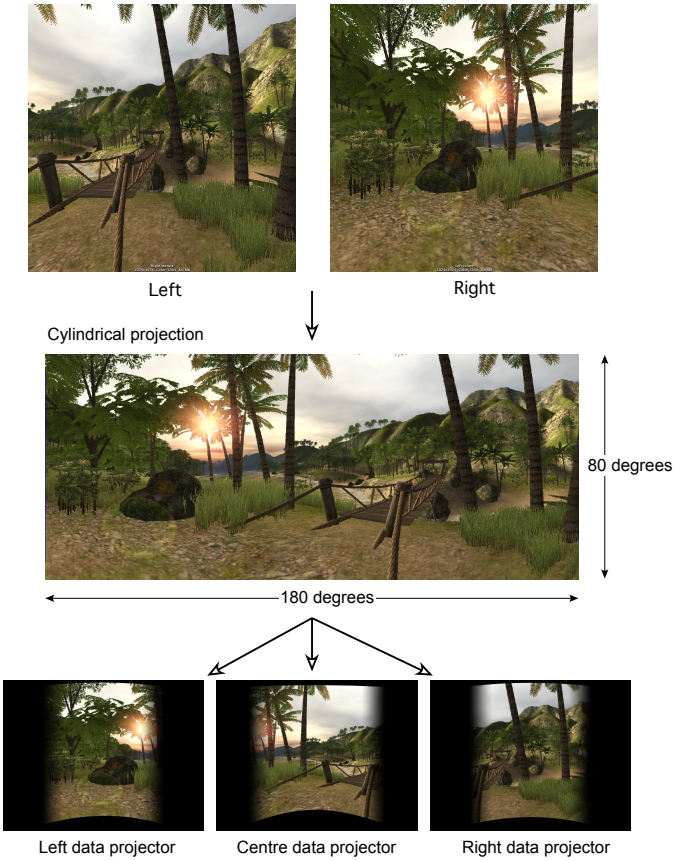

Figure 5. Pipeline illustrating the two generic cube maps needed to create a 180 x 80 degree cylindrical projection followed by display specific geometry correction and edge blending masks.

The display specific mappings include geometry correction and edge blending, the exact details are potentially different for each environment. Every possibility (screen dimensions, number of data projectors, positions and zoom setting of the data projectors, resolution and aspect ratio of the data projectors, width of edge blend zones, and so on) cannot be reasonably supported by each game developer but the mappings can be abstracted if the developer provides a pass that maps the cubic map images or derived combinations onto a mesh [15]. The geometry corrections (image warping) necessary can be supported by each vertex of the mesh having it's own position (x,y) coordinate (in normalised image coordinates) and it's own texture coordinate (in the range of 0 to 1 on both axes). While this does offer a degree of redundancy some mappings are more readily realised by using either vertex coordinates or vertex texture coordinates or both to achieve the desired result. Generalised per gamma corrected edge blending is accompished by each vertex having a multiplicative scale factor (on the range of 0 to 1), se figure 6.

In this model the game developer need not be concerned with the details of the final display, only with providing support for multipass cubic maps and subsequent mappings onto regular texture meshes. The generation of the mesh for this second pass is the responsibility of the display supplier, indeed only they are in a position to implement the alignment and blending necessary for the hardware involved. All that is required is a standard format in which to describe this warping mesh data.

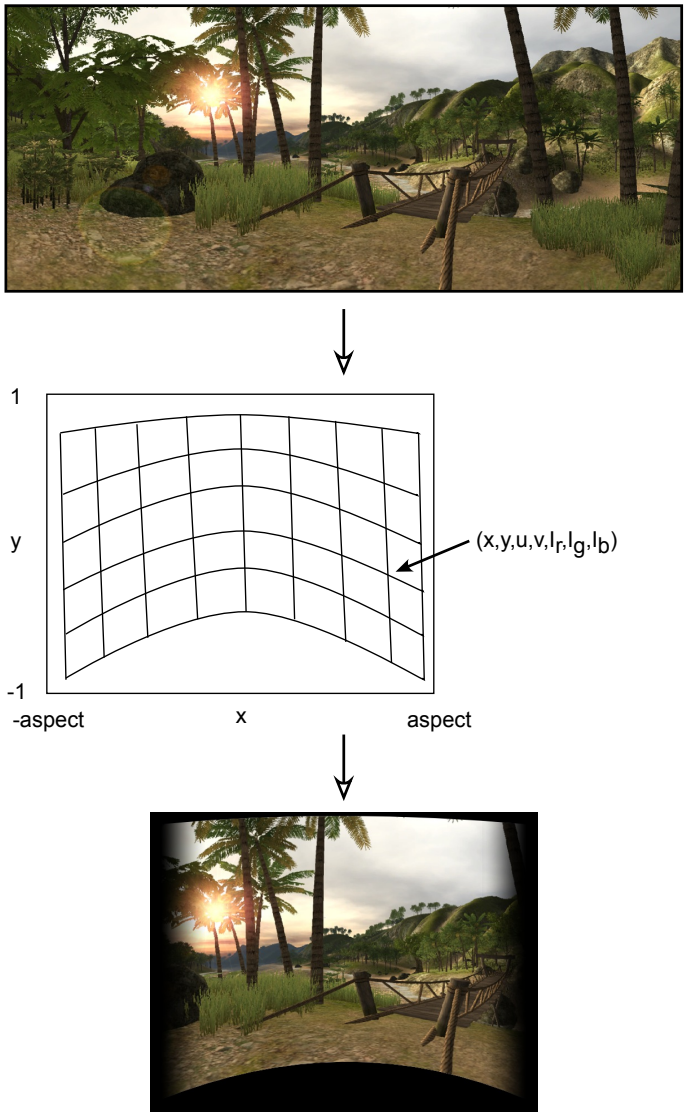

Figure 6. Implementation of display dependent mappings can be handled by meshes, the vertices of which have positions (x,y), texture coordinates (u,v), and multiplicative intensity factors per colour component (I_r,I_g,I_b).

III. SUMAMRY

The benefits of immersion and displays that fill the viewers field of view are well established and exploited in training simulators and virtual reality systems [13][14]. Gamers who are engaging in very similar activities are unable to take advantage of immersion because of the lack of support for surround seamless displays. Support for surround displays cannot readily be retrofitted to an existing game, rather support needs to be explicitly built into the game software. Presented here is an approach implemented in two phases. In the first the goal is to capture the required visual field of view, this is accomplished in a very general way by capturing N frustums (N <= 6) defined by the faces of a cube surrounding the virtual

camera. The second phase deals with the more device specific nature of the final display, implementing geometry correction and edge blending. The exact details of the mesh for this second phase is the responsibility of the display supplier.

ACKNOWLEDGMENT

The work was supported by iVEC through the use of advanced computing resources located at the University of Western Australia. The case study examples and proof of concept demonstrations were created using the Unity3D game engine.

REFERENCES

[1] T. Hatada, H. Sakata, H. Kusaka. Psychophysical Analaysis of the Sensation of Reality Induced by a Visual Wide-Field Display. SMPTE Journal, Vol 89, August 1980, pp. 560-569.

[2] J. I. Thompson. A three dimensional helmet mounted primary flight reference for paratroopers. Master's thesis, Air Force Institute of Technology, 2005.

[3] B. Ball, C. North. The effects of peripheral vision and physical navigation on large scale visualization. ACM International Conference Proceedings Series; Vol 322. Proceedings of graphics interface. ISSN:0713-5424. 2008.

[4] P.D. Bourke, D. Q. Felinto. Blender and Immersive Gaming in a Hemispherical Dome, GSTF International Journal on Computing (JoC). ISSN: 2010-2283, Vol. 1, No. 1, 2010.

[5] P.D. Bourke. Low Cost Projection Environment for Immersive Gaming. JMM (Journal of Multimedia), ISSN: 1796-2048, Volume 3, Issue 1, 2008, pp 41-46.

[6] L. Likarish, O. Brdiczka, N Yee, N. Ducjeneaut, L Nelson. Demographic Profiling from MMOG Gameplay. 11th Privacy Enhancing Technologies Symposium. Waterloo, Canada, 2011.

[7] M. Shapiro. Comparing user experience in a panoramic HMD vs. projection wall virtual reality system. tech. rep., Sensics, Inc., 2006.

[8] J. Nilsson. jDome. Web reference: http://jdome.com/ (January 2012).

[9] Web reference: Google Earth Liquid Galaxy (Janurary 2012) http://www.google.com/earth/explore/showcase/liquidgalaxy.htm

[10] P. D. Bourke. iDome: Immersive gaming with the Unity game engine. Proceedings of the Computer Games & Allied Technology 09 (CGAT09), Research Publishing Services, ISBN: 978-981-08-3165-3, pp 265-272.

[11] Web Reference: Alive Laboratories City University Hong Kong. http://www.cityu.edu.hk/scm/alive/main.htm (January 2012)

[12] C. Cruz-Neira. Surround-screen projection based virtual reality. Proceedings of Siggraph 93.

[13] P. S. Heckbert. Fundamentals of texture mapping and image warping. Master's thesis, University of California, Berkeley, 1989

[14] H.G. Hoffman, T. Richards, B. Coda, A. Richards, S.R. Sharar. The Illusion of Presence in Immersive Virtual Reality during an fMRI Brain Scan. CyberPsychology and Behavior, Vol 6, Number 3, 2003.

[15] W. Monaco, J. Kalb, C. Johnson. 2007. Motion detection in the far peripheral visual field. Army Research Laboratory Report ARL-MR-06