Unity Cluster Package – Dragging and Dropping Components for Multi-projection Virtual Reality Applications Based on PC Clusters

Mário Popolin Neto[1,4], Diego Roberto Colombo Dias[2], Luis Carlos Trevelin[2], Marcelo de Paiva Guimarães[3], and José Remo Ferreira Brega[4]

[1] Federal Institute of São Paulo – IFSP Registro, SP, Brazil
[2] Computer Science Department – Federal University of São Carlos – UFSCAR São Carlos, SP, Brazil
[3] Open University of Brazil – Federal University of São Paulo – UNIFESP / Computer Science Master Program – FACCAMP São Paulo, SP, Brazil
[4] Computer Science Department – São Paulo State University – UNESP Bauru, SP, Brazil
mariopopolin@ifsp.edu.br,{diegocolombo.dias,trevelin}@dc.ufscar.br
marcelodepaiva@gmail.com,remo@fc.unesp.br

Abstract. Virtual Reality applications created using game engines allow developers to quickly come up with a prototype that runs on a wide variety of systems, achieve high quality graphics, and support multiple devices easily. This paper aims to present a component set (Unity Cluster Package) for the Unity game engine that facilitates the development of immersive and interactive Virtual Reality applications. This drag-and-drop component set allows Unity applications to run on a commodity PC cluster with passive support for stereoscopy, perspective correction according to the user's viewpoint and access to special servers to provide device-independent features. We present two examples of Unity multi-projection applications running in a mini CAVE (Cave Automatic Virtual Environment)-like (three-screens) system ported using this component set.

Keywords: Virtual Reality, Unity Game Engine, Multi-projection, PC cluster.

1 Introduction

Game technology can be used to create Virtual Reality (VR) applications, since both areas exploit technological advances in diverse fields, such as image synthesis and interaction devices [6]. Game engines have emerged providing greater interactivity and graphics performance [16], which has led researchers to explore these tools for game creation in applications for CAVE™ [8] (Cave Automatic Virtual Environment)-like systems [19]. These multi-projection applications created with game engines typically are found in arts [7,18], simulators [4,14] and visualization systems [17].

The exponential growth of processor capabilities and the emergence of dedicated graphics hardware have enabled the use of PC clusters to run immersive and interactive applications. There are some libraries that support the development of immersive and interactive applications to run on PC clusters (for example, libGlass [5] and FreeVR [21]). However, they require integration with external rendering solutions. Current game engines integrate a set of reusable modules, such as Graphics and Network, but do not have original support to create VR applications to run on PC clusters; they are adapted to meet this demand by means of multi-projection frameworks.

The main contribution of the present research project is to simplify the design of immersive and interactive VR applications based on PC clusters, freeing the developer from having to know all the details of the individual technologies. We present the Unity Cluster Package, a drag-and-drop component set for the Unity game engine that allows the development of immersive and interactive VR applications based on PC clusters in an easy way. We also show two Unity demos that were ported to run in a 3-screens multi-projection system. The package extends these applications to support multi-projection, passive stereoscopy, perspective correction according to the user's viewpoint and access to device servers. We developed this component set for Unity because it is one of the most popular game engines with a custom rendering engine, an extensive documentation and an easy-to-use editor, and it is multi-platform [11].

The remainder of the paper is organized as follows. Section 2 contains the background work related to this research. Section 3 describes the Unity Cluster Package and the main components to use the Unity game engine in the development of immersive and interactive VR applications. Section 4 explains the port of two Unity demos for a 3-screens multi-projection system. Section 5 discusses final considerations. Section 6 presents conclusions and future works.

2 Related Work

Although Paul Rajlich's CAVE Quake II was probably the first implementation with a game engine (Quake Engine) for an immersive system [19], the framework CaveUT™ [12] became more popular allowing the development of Unreal™ Engine applications [7,18] for CAVE™-like systems. The use of game engines to develop multi-projection applications depends on the level of access to their internal modules, since these tools are not only a set of reusable modules, but also a layer that connects and manages them [3]. Multi-projection frameworks for game engines, such as CaveUT™ and its successor CaveUDK [19] (Unreal® Engine), and CryVE [13] (CryEngine® 2), use the internal modules to develop immersive and interactive virtual environments for multi-projection systems.

As well as the use of game engines that have multi-projection frameworks, VR integration libraries can be combined with game engines with accessible modules, such as the use of FreeVR with the open source Delta3D, for the development of

[5] http://sourceforge.net/projects/libglass/

an immersive simulator [14]. Although open source game engines are an option, they do not exhibit some properties provided by commercial game engines, such as the possibility of development for multiple platforms and rendering quality. Hardware adapters that split and distribute the video signal also can be used, such as the Matrox DualHead2Go [6] used in the Virtual Reality Theatre [20]. This approach is based on hardware and it does not require any type of software, but increases the area occupied by a single pixel resulting in loss of image resolution.

PC clusters based on the Master-Slave [22] rendering model are the most frequently used when dealing with game engines for multi-projection applications development [4,7,14,17,18,23]. The multi-projection frameworks for game engines, such as CaveUT™ [12], CaveUDK [19] and CryVE [13], provide support to PC clusters. These frameworks are similar regarding the usage of the game engine network module for communication and synchronization among cluster nodes applications. The utilization of the network internal module for these purposes has become a trend, and it is also being used for local solutions [17]. The Unity Cluster Package presented in this paper was developed based on the Unity network module.

CaveUDK [19] is a natural successor of CaveUT™ (Unreal® Engine 2, extending the Unreal® Development Kit (Unreal® Engine 3) for CAVE™-like systems. CaveUDK supports stereo images by active stereoscopy and interaction device from VRPN (*Virtual Reality Peripheral Network* [25]) servers via an integrated module (C++ DLL). This framework is an open source initiative, but requires Unreal® Development Kit purchase. CryVE [13] aims for a low-cost implementation that sits on top of CryEngine® 2; however, despite offering an extensible solution (open source), it does not provide stereo images or VRPN access, and it uses an out-of-date game engine, since there is no record from code recompilation of this solution for the CryEngine® 3.

CaveUDK and CryVE are game engine extensions in open source compiled code, usually C++ DLLs, providing code reusability for developers. The game engine purchase is also a characteristic of both frameworks. The MiddleVR [7] plugin can be used with the popular Unity game engine to develop applications for different VR systems, such as HMD (Head-Mounted Display) [5] and CAVE™ [23]. This plugin supports PC clusters, active and passive stereoscopy, and VRPN servers, but it is a closed commercial solution and depends on Unity Pro to provide the PC cluster feature. The Unity Cluster Package is free of charge, and it can be used in both Unity game engine versions (Pro and Free). This package provides reusable drag-and-drop components to develop Unity applications for PC clusters, allowing exploitation of the Unity qualities in multi-projection systems.

[6] http://www.matrox.com/graphics/en/products/gxm/
[7] http://www.imin-vr.com/middlevr/

3 Unity Cluster Package

Unity Cluster Package is a Unity package (file extension .unitypackage) that was designed to overcome the difficulties faced by developers in creating immersive and interactive VR applications. Figure 1 shows the layers of components an immersive and interactive virtual reality PC cluster application created with Unity, consisting of: (1) Virtual Reality Applications, immersive and interactive VR applications; (2) Unity Cluster Package, drag-and-drop components to support VR applications development; and (3) Unity, the game engine Unity.

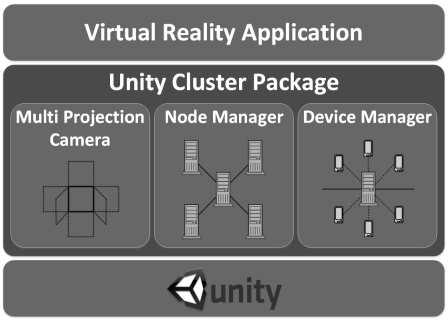

Fig. 1. Unity Cluster Package.

3.1 Multi-projection System

The Unity Cluster Package can be used in various multi-projection systems, and a MiniCAVE [9,10] is our target system here. Figure 2 presents the MiniCAVE, which consists of three screens (2.5 m x 1.5 m each) arranged at an angle of 30 degrees to each other. For each screen, there are two projectors (BenQ W1000 HD) connected to two dedicated computers (Intel Core i7 8GB RAM), with NVIDIA FX 1800 as the graphics card. Polarized lenses are used to provide passive stereoscopic projection. A total of six computers are used for the projections, and they are connected to a server computer by a gigabit switch. The Unity Cluster Package provides a coherent, seamless and contiguous view from the PC cluster (six rendering nodes and one server node) applications in the MiniCAVE.

3.2 Unity

Unity is a commercial game engine created and maintained by Unity Technologies that has a Pro and Free version [1]. This popular game engine is composed of a custom rendering engine (OpenGL or DirectX) and an editor with intuitive workflows. Unity belongs to the Modular Framework Engine group [3], which

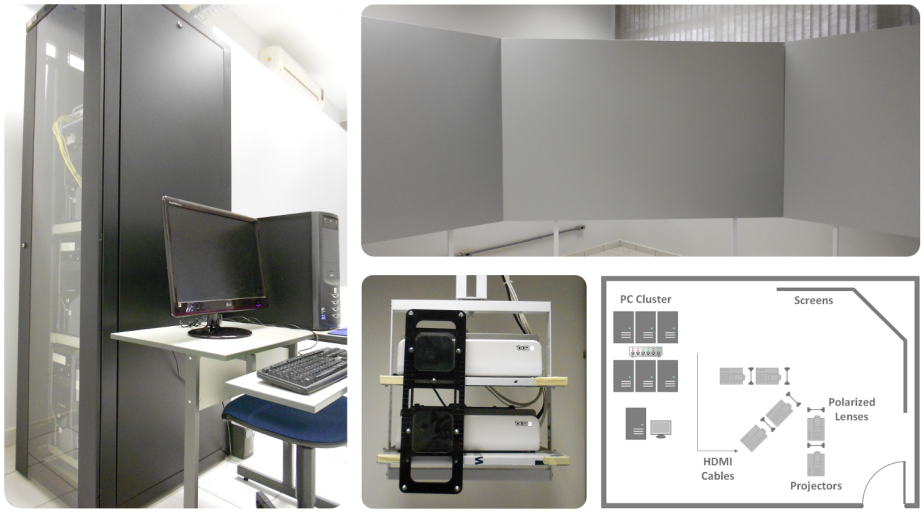

Fig. 2. MiniCAVE: PC cluster (seven computers and gigabit switch), three screens, projectors and polarized lenses.

provides a good relation between customization and complexity. Developers can implement scripts in JavaScript, C# and Boo to build the game logic, and these scripts can be attached to scene objects using prefabs. Prefab is a Unity mechanism that allows storage of objects with their properties and components, working as a template which can be shared and reused. Unity is also a multi-platform game engine, which enables the development of games for Windows, Linux, Mac OS, iOS, Android, web browsers, and consoles [1].

3.3 Unity Cluster Package: drag-and-drop components

The Unity Cluster Package is easily imported into the Unity Editor and it allows the developer to build immersive and interactive VR applications, dragging and dropping the available components as needed. This package is user-friendly, and not only suitable for new applications, but also for legacy code. This is an important feature, as most available solutions for these applications require a substantial amount of code; some of them require a complete change of the application design.

The Unity Cluster Package's main goal is to allow easy development of immersive and interactive VR applications based on PC clusters through well-designed and high configurable components, which are Unity prefabs that arrange scene objects and C# scripts according to their functionality. There are three main components: the *Multi Projection Camera*, the *Node Manager*, and the *Device Manager*.

Multi Projection Camera The *Multi Projection Camera* is a customized virtual camera for the development of multi-projection applications. It arranges a PreRender script, and the scene objects *ProjectionPlane* (plane) and *UserHead* (sphere) to calculate the projection matrix [15], where the points Pa, Pb, and Pc (Figure 3) encode the size, aspect ratio, position and orientation of the screen. This projection matrix allows perspective correction according to point Pe.

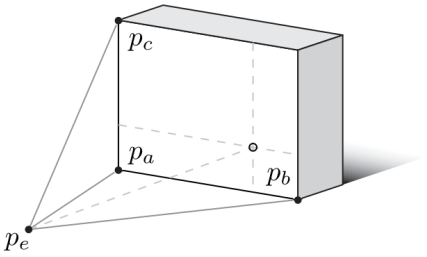

Fig. 3. Points used to calculate the projection matrix: Pa, Pb, Pc, and Pe - Adapted [15].

The PreRender script calculates the projection matrix obtaining the points Pa, Pb, and Pc in the virtual environment through the *ProjectionPlane*. The *UserHead* position is taken as the point Pe, providing a motion parallax metaphor. By moving the object *UserHead* according to any head-tracker device, the motion parallax is achieved, offering a better perception of the virtual environment.

Node Manager The *Node Manager* uses the Unity network module to implement the Master-Slave [22] rendering model, in which each cluster node runs the same application with different virtual camera configurations, and all user input interactions are treated by the master node. This component is a scene object with a special script attached to start the application according to the cluster node type, where each slave node (Unity Client) is connected to the master node (Unity Server), as shown in Figure 4.

The script also instantiates the *Multi Projection Camera* in the master node application using the Unity network module; in this way, all slave node applications have an instance of this custom virtual camera. To achieve image coherence, the *Multi Projection Camera*'s position and rotation must be synchronized. The Unity network module provides the NetworkView component to perform scene objects synchronization. The *Multi Projection Camera*'s position and rotation are changed on the master node and synchronized through the StateSynchronization functionality provided by the NetworkView.

Slave node applications are assigned to the screen setting up the *Multi Projection Camera* by means of the points Pa, Pb, and Pc in a coordinate system with the origin in the center of the front screen. The *Node Manager* obtains the cluster node specification through an XML configuration file that contains the

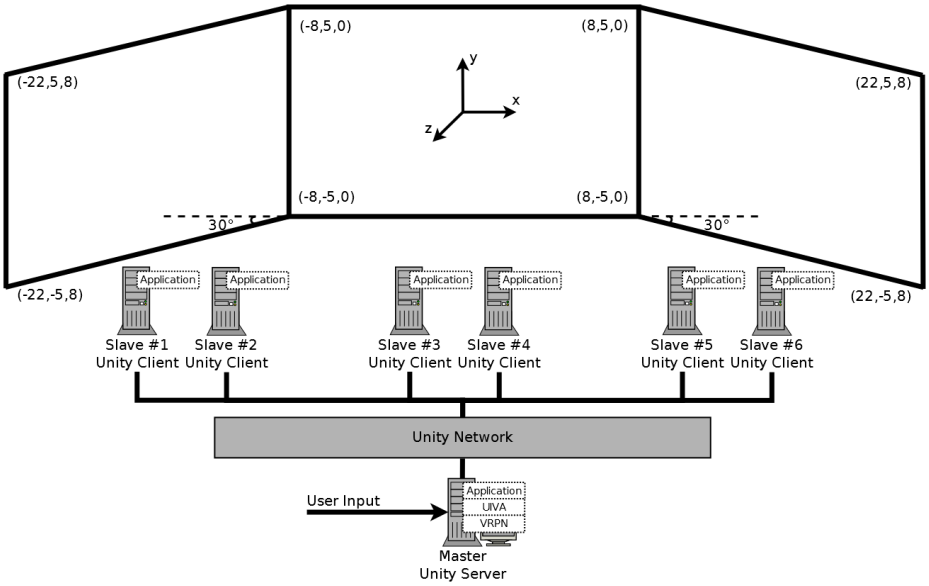

Fig. 4. Unity Cluster Package for the MiniCAVE.

points Pa, Pb, and Pc, the node type (master or slave), and the IP address and connection port to the Unity Server.

Figure 4 shows the points Pa, Pb, and Pc for the MiniCAVE, where the Slave #3 application is assigned to the front screen with Pa = (-8,-5,0), Pb = (8,-5,0), and Pc = (-8,5,0). The Slave #3 configuration file is shown in Figure 5. The *screen* tag parameters *stereo* (true or false) and *eye* (right or left) specify for which eye the application addresses the projection. In order to achieve passive stereoscopic projection, the Slave #3 and Slave #4 address the projection for the same screen but for different eyes: Slave #3 for the left eye (eye = "left") and Slave #4 for the right eye (eye = "right").

```
<node type="slave">
  <server ip="192.168.1.13" port="25000"/>
  <screen stereo="true" eye="left">
    <pa x="-8" y="-5" z="0"/>
    <pb x="8"  y="-5" z="0"/>
    <pc x="-8" y="5"  z="0"/>
    <pe x="0"  y="0"  z="5"/>
  </screen>
</node>
```

Fig. 5. Slave #3 configuration file.

Device Manager The Unity game engine provides the Input class to access data from the supported interaction devices (mouse, keyboard, joystick, and mobile devices) through static variables and functions. All input variables are reset on each rendered frame.

The *Device Manager* features device-independent interaction supporting devices via VRPN servers through the extensible UIVA (*Unity Indie VRPN Adapter* [2]). This component is a scene object that allows use of a Wii Remote as the interaction device and a Microsoft Kinect as the user's head tracker, providing scripts based on the Unity Input class, where static variables containing Wii Remote and Kinect data are updated on each rendered frame and can be accessed by other scripts. As well as the possibility of using other supported devices, such as the SpacePoint Fusion sensor and PhaseSpace's optical motion capture system, UIVA is open source, which enables addition of support for new devices.

4 Case Studies

We present two case studies, showing how Unity applications can be extended to run in a full immersive and interactive environment based on PC clusters using the Unity Cluster Package. The environment can range from modest (Portable CAVE) to high-quality (High-end CAVE), while keeping the same procedures. We adapted the Unity demos StarTrooper and Bootcamp available on the Unity Asset Store [8] to run in the MiniCAVE system. The cluster node applications were arranged as presented in Figure 4, and then imported the Unity Cluster Package; its components are used by dragging and dropping into the scene creation window in the Unity Editor. The Unity demos adaptation involved three basic steps:

1. Add the *Node Manager*, which starts the cluster node application and instantiates the *Multi Projection Camera* according to the cluster node type.
2. Add the *Device Manager*, which enables access to the Wii Remote and Kinect data updated on each rendered frame.
3. Define the scene objects that need to be synchronized among the cluster node applications through the Unity network module, where only the master node application must perform state changes on these objects.

StarTrooper in the MiniCAVE (Figure 6–(a)) is a circular arena where the aircraft flight and missile launch systems are provided by a Wii Remote. Regarding the third step, the StarTrooper game objects 'aircraft', 'missile', and 'ring' are synchronized by the NetworkView component from the Unity network module. Bootcamp Explore (Figure 6-(b)) enables first person navigation in the Bootcamp scenario using a Wii Remote and Nunchuk in the MiniCAVE. The Unity standard scripts for first person controller were modified and attached to the *Multi Projection Camera*, enabling the Wii Remote and Nunchuk as controller devices, and using the NetworkView component for synchronization (third step). Motion parallax was implemented in the Bootcamp Explore, assigning the

[8] https://www.assetstore.unity3d.com/en/

UserHead position from the *Multi Projection Camera* to the user's head position obtained from the Kinect.

Fig. 6. Unity demos adapted for the MiniCAVE.

The master node is responsible for running the UIVA, the VRPN server for the Wii Remote [2], and the VRPN server FAAST (*Flexible Action and Articulated Skeleton Toolkit* [24]) for the Kinect. Some changes were made in the UIVA in order to support the Wii Remote additional peripheral controller Nunchuk. The Unity demos adaptation can be viewed at this video [9].

The performance of these demos adapted for the MiniCAVE is presented in Table 1. It shows the frames per second (FPS) average in relation to the number of rendered triangles. The performance is satisfactory, considering that VR applications should have more than 30 FPS. Unity Cluster Package provides a good relationship between frame rate and rendered triangles. CaveUDK gives a frame rate of 50 FPS for 200 K to 600 K rendered triangles [19]. CryVE, with its medieval museum virtual tour, provides a frame rate less than 20 FPS [13].

Table 1. Unity demos adaptations – performance.

Unity Demo Adaptation	**FPS Average**	**Rendered Triangles**
StarTrooper	566	12K
Bootcamp Explore	58	237K

[9] https://www.youtube.com/watch?v=nvVDAvdw_k8

5 Final Considerations

Some comparisons can be made between the Unity Cluster Package and other solutions (Table 2). The reusability of our solution resides in its components, which can be used by dragging and dropping into the scene creation window in the Unity Editor. Multi-projection frameworks for game engines, such as CaveUDK and CryVE, are extensions as open source compiled code (DLLs), requiring programming knowledge from developers, and game engine purchase (Pro), similar to MiddleVR with the Unity game engine. The Unity package presented in this paper is easily imported into the Unity Editor, and can be used in both Unity versions (Free and Pro).

Table 2. Solutions comparison.

Solution	**Type**	**Game Engine**	**Stereoscopy**	**VRPN**	**Version**
Our Package	Open Source	Unity	Passive	Yes	Free/Pro
MiddleVR	Commercial	Unity	Active/Passive	Yes	Pro
CaveUDK [19]	Open Source	Unreal® Engine	Active	Yes	Pro
CryVE [13]	Open Source	CryEngine® 2	–	No	Pro

Through polarized lenses (passive stereoscopy), such as in the MiniCAVE, stereo images can be achieved with common graphic cards and projectors; on the other hand, active stereoscopy requires specific hardware, such as NVIDIA Quadro 5000 and the shutter glasses used in the CaveUDK. Access to interaction devices from VRPN servers can be considered as a standard, since this is found in both commercial (MiddleVR) and open source (CaveUDK and Unity Cluster Package) solutions.

6 Conclusion and Future Works

This paper presented the Unity Cluster Package, a flexible and extensible solution for creating immersive and interactive VR applications using the Unity game engine. This package allows extension of Unity applications to support multi-projection systems based on PC clusters, passive stereoscopic projection and perspective correction according to the user's viewpoint. The integration with the extensible UIVA enables the use of the wide range of interaction devices supported by VRPN servers. In the case studies we used a MiniCAVE to test our solution; however, this is not a restriction since the package is suitable for other multi-projection systems. Unity Cluster Package is open source and it is available via the Unity Cluster Package Sourceforge project page [10].

As future work, we will exploit the Unity game engine in the development of immersive and interactive VR applications, and also investigate the use of mobile devices (tablets and smartphones) as cluster nodes and interaction devices.

[10] https://sourceforge.net/projects/unityclusterpackage/

7 Acknowledgements

The authors would like to thank to CAPES Foundation, a body of the Brazilian Ministry of Education, for financial support. Mário Popolin Neto and Diego Roberto Colombo Dias were recipient of scholarships from CAPES.

The adaptations in this paper used the Unity demos StarTrooper and Bootcamp available on the Unity Asset Store. All rights to the graphical and audio assets belong to Unity Technologies and are not used here for commercial purposes.

References

1. Create the games you love with Unity. http://unity3d.com/unity, [Online; accessed March-2015]
2. Unity Indie VRPN Adapter – UIVA. http://web.cs.wpi.edu/~gogo/hive/UIVA/, [Online; accessed March-2015]
3. Anderson, E.F., McLoughlin, L., Watson, J., Holmes, S., Jones, P., Pallett, H., Smith, B.: Choosing the infrastructure for entertainment and serious computer games - a whiteroom benchmark for game engine selection. In: Games and Virtual Worlds for Serious Applications (VS-GAMES), 2013 5th International Conference on. pp. 1–8 (Sept 2013)
4. Backlund, P., Engstrom, H., Hammar, C., Johannesson, M., Lebram, M.: Sidh - a game based firefighter training simulation. In: Information Visualization, 2007. IV '07. 11th International Conference. pp. 899–907 (2007)
5. Beimler, R., Bruder, G., Steinicke, F.: Smurvebox: A smart multi-user real-time virtual environment for generating character animations. In: Proceedings of the Virtual Reality International Conference: Laval Virtual. pp. 1:1–1:7. VRIC '13, ACM, New York, NY, USA (2013), http://doi.acm.org/10.1145/2466816.2466818
6. Bouvier, P., De Sorbier, F., Chaudeyrac, P., Biri, V.: Cross benefits between virtual reality and games. In: Computer Games and Allied Technology 08, CGAT 08 - Animation, Multimedia, IPTV and Edutainment, Proceedings. pp. 186–193 (2008), www.scopus.com
7. Cavazza, M., Lugrin, J.L., Pizzi, D., Charles, F.: Madame bovary on the holodeck: immersive interactive storytelling. In: Proceedings of the 15th international conference on Multimedia. pp. 651–660. MULTIMEDIA '07, ACM, New York, NY, USA (2007), http://doi.acm.org/10.1145/1291233.1291387
8. Cruz-Neira, C., Sandin, D.J., DeFanti, T.A.: Surround-screen projection-based virtual reality: the design and implementation of the cave. In: Proceedings of the 20th annual conference on Computer graphics and interactive techniques. pp. 135–142. SIGGRAPH '93, ACM, New York, NY, USA (1993), http://doi.acm.org/10.1145/166117.166134
9. Dias, D.R.C., Brega, J.R.F., Trevelin, L.C., Popolin Neto, M., Gnecco, B.B., de Paiva Guimaraes, M.: Design and evaluation of an advanced virtual reality system for visualization of dentistry structures. In: Virtual Systems and Multimedia (VSMM), 2012 18th International Conference on. pp. 429–435 (Sept 2012)
10. Dias, D.R.C., Brega, J.R.F., Lamarca, A.F., Popolin Neto, M., Suguimoto, D.J., Agostinho, I., Gouveia, A.F.: Chemcave3d: Sistema de visualização imersivo e interativo de moléculas 3d. In: Workshop de Realidade Virtual e Aumentada. WRVA, Uberaba, MG, Brasil (2011)

11. Hu, W., Qu, Z., Zhang, X.: A new approach of mechanics simulation based on game engine. In: Computational Sciences and Optimization (CSO), 2012 Fifth International Joint Conference on. pp. 619–622 (June 2012)
12. Jacobson, J., Hwang, Z.: Unreal tournament for immersive interactive theater. Commun. ACM 45(1), 39–42 (Jan 2002), http://doi.acm.org/10.1145/502269.502292
13. Juarez, A., Schonenberg, W., Bartneck, C.: Implementing a low-cost cave system using the cryengine2. Entertainment Computing 1(3–4), 157 – 164 (2010), http://www.sciencedirect.com/science/article/pii/S1875952110000108
14. Koepnick, S., Norpchen, D., Sherman, W., Coming, D.: Immersive training for two-person radiological surveys. In: Virtual Reality Conference, 2009. VR 2009. IEEE. pp. 171–174 (2009)
15. Kooima, R.: Generalized Perspective Projection. http://aoeu.snth.net/static/gen-perspective.pdf, [Online; accessed March-2015]
16. Lewis, M., Jacobson, J.: Game engines in scientific research. Commun. ACM 45(1), 27–31 (Jan 2002), http://doi.acm.org/10.1145/502269.502288
17. Louloudi, A., Klugl, F.: Visualizing agent-based simulation dynamics in a cave - issues and architectures. In: Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on. pp. 651–658 (Sept 2011)
18. Lugrin, J.L., Cavazza, M., Palmer, M., Crooks, S.: Artificial intelligence-mediated interaction in virtual reality art. IEEE Intelligent Systems 21(5), 54–62 (Sep 2006), http://dx.doi.org/10.1109/MIS.2006.87
19. Lugrin, J.L., Charles, F., Cavazza, M., Le Renard, M., Freeman, J., Lessiter, J.: Caveudk: A vr game engine middleware. In: Proceedings of the 18th ACM Symposium on Virtual Reality Software and Technology. pp. 137–144. VRST '12, ACM, New York, NY, USA (2012), http://doi.acm.org/10.1145/2407336.2407363
20. Schou, T., Gardner, H.J.: A wii remote, a game engine, five sensor bars and a virtual reality theatre. In: Proceedings of the 19th Australasian Conference on Computer-Human Interaction: Entertaining User Interfaces. pp. 231–234. OZCHI '07, ACM, New York, NY, USA (2007), http://doi.acm.org/10.1145/1324892.1324941
21. Sherman, W.R., Coming, D., Su, S.: Freevr: honoring the past, looking to the future. vol. 8649, pp. 864906–864906–15 (2013), http://dx.doi.org/10.1117/12.2008578
22. Staadt, O.G., Walker, J., Nuber, C., Hamann, B.: A survey and performance analysis of software platforms for interactive cluster-based multi-screen rendering. In: Proceedings of the Workshop on Virtual Environments 2003. pp. 261–270. EGVE '03, ACM, New York, NY, USA (2003), http://doi.acm.org/10.1145/769953.769984
23. Steptoe, W., Steed, A., Slater, M.: Human tails: Ownership and control of extended humanoid avatars. Visualization and Computer Graphics, IEEE Transactions on 19(4), 583–590 (2013)
24. Suma, E., Lange, B., Rizzo, A., Krum, D., Bolas, M.: Faast: The flexible action and articulated skeleton toolkit. In: Virtual Reality Conference (VR), 2011 IEEE. pp. 247–248 (March 2011)
25. Taylor, II, R.M., Hudson, T.C., Seeger, A., Weber, H., Juliano, J., Helser, A.T.: Vrpn: A device-independent, network-transparent vr peripheral system. In: Proceedings of the ACM Symposium on Virtual Reality Software and Technology. pp. 55–61. VRST '01, ACM, New York, NY, USA (2001), http://doi.acm.org/10.1145/505008.505019