Contents lists available at ScienceDirect

Entertainment Computing

journal homepage: ees.elsevier.com/entcom

Implementing a low-cost CAVE system using the CryEngine2

Alex Juarez [*], Willem Schonenberg, Christoph Bartneck

Department of Industrial Design, Eindhoven University of Technology, Den Dolech 2, 5600 MB Eindhoven, The Netherlands

ARTICLE INFO

Article history:
Received 17 March 2010
Revised 5 October 2010
Accepted 6 October 2010

Keywords:
CAVE
Low-cost
Interactive environments
CryEngine2

ABSTRACT

In this paper we present the design and implementation of a low-cost CAVE system based on the state of the art game engine CryEngine2. We show the physical construction and preliminary results of such implementation and explore the possibilities of its application to interactive setups, e.g. a virtual museum tour.

1. Introduction

Immersive virtual environments have been an active topic of research for the past decade in a broad range of application areas, such as high-end massive data visualization tools [1], driving simulation [2], and psychological evaluations [3].

A Cave Automatic Virtual Environment (CAVE), is an implementation of a virtual reality system consisting of a physical installation for image or video projection, and the software that controls such installation. The physical installation usually consists of a closed room where video projectors are directed to two or more of its walls, creating an impression of being "inside" the virtual environment. The software component of a CAVE is in charge of controlling the projection of images and video, the rendering of 2D and 3D models and animations, and the overall control of the interactions between the installation users and the virtual worlds.

There is a gap in qualitative and quantitative terms between commercial and open source CAVE systems. This gap is the difference between systems that provide stunning visuals, state of the art modeling, rendering and visualization, and continuous support at high costs; and systems that provide cost effective solutions, with more limited visuals, development capabilities, and overall user experience. As a rule of thumb, commercial solutions like the i-Space [4] or Apollo [5] CAVE systems offer accurate and realistic experience customized to the needs of the client, with extensive application and installation support, at the expense of prohibitive costs. On the other hand, open source, non-commercial solutions like AGAVE [6] and CAVEUT [7] offer cost-effective setups that are in many cases, implemented only at education institutions, using outdated technology, with limited development and scarce support to potential users. In general, CAVE implementations (both commercial and open-source) face the challenge of achieving a high level of realism while keeping the overall costs at an acceptable level.

A low-cost implementation is needed to fill in the gap between commercial and open source systems. The main characteristics of such implementation would be:

- *A low-cost, full-size physical construction*. While there are CAVE systems of reduced dimensions at affordable prices, the truly immersive experience comes from a physical setup that can accommodate at least one person standing comfortably, allowing for some space to move and interact with the system.
- *Easy to setup, maintain and extend*. A system that requires the intervention of experts for even the simplest tasks are unlikely to be more efficient than those where a dedicated enthusiast can get satisfying results. This means that the use of standard and simple components, as well as uncomplicated methods and mechanisms to modify and extend the CAVE are preferred. Furthermore, the existence of an active development community that provides support and continuously improve and extend the software, is also an important factor.
- *Realistic immersive experience*. A successful CAVE system must provide a believable experience to a spectator, presenting a visually rich environment. The CAVE system should also feature state of the art 3D modeling and physics that resemble those of real life.

In this paper we introduce an implementation of a low-cost, full-size CAVE that uses a state of the art game engine as the underlying software system. In the following sections we present the

[*] Corresponding author.
E-mail addresses: acordova@tue.nl (A. Juarez), w.a.s.schonenberg@student.tue.nl (W. Schonenberg), cbartneck@tue.nl (C. Bartneck).

1875-9521/$ - see front matter © 2010 International Federation for Information Processing All rights reserved.
doi:10.1016/j.entcom.2010.10.001

CAVE physical design, address implementation details concerning the software used, and show a prototype implementation in the form of a virtual museum tour. The tour is set in a medieval virtual world.

2. Related work

CAVE systems first appeared at the beginning of 1990s as a visualization tool for virtual reality environments that utilized an array of large projection screens, arranged to resemble a room whose walls, ceiling and floor surrounded a viewer with projected images. The installation was designed to experience an "immersion" in the virtual world [8]. Since then, several variations of this setup have been developed, examples include asymmetric screens, portable CAVEs with only two walls [9], arrangements in "U" [10], and spherical screens [11].

Software for CAVE systems has also evolved from research-oriented, custom-made applications that modified the image aspect ratio and display quality, into powerful software tools. Existing solutions are capable of performing 3D modeling and rendering, incorporating multiprocessing and clustering, and supporting high-performance computing and data intensive systems coupled to collaborative environments [6,12]. Other devices can also be incorporated to the system, like head-mounted displays, 3D glasses, pressure and temperature sensors, etc. [13].

Earlier attempts to produce low-cost CAVEs created solutions based on distributing the processing to networked computers that use commodity hardware acceleration and open source software components to achieve the required performance [14,15]. Unfortunately, the cut in hardware and software costs came at the expense of the expertise required to setup the system: extensive knowledge on graphics and hardware acceleration techniques was required, and made it a hard task for the non-expert.

In addition to that, the term "low-cost" did not automatically mean that it was affordable for any medium-sized institution: Green and White [16] estimated in the year 2000 that a low-cost CAVE system with acceptable performance that could be accommodated in a laboratory space, was around 92,500 Euros ($100,000).

A step forward in terms of cost and performance was the appearance of game-engine-based virtual reality systems. These systems used commercially available software packages – *game engines* – that provided advanced simulation and graphics that were in many cases superior to those offered by professional CAVEs. The most widely known example of this kind of system is CAVEUT [7,17], a CAVE based on an extension of the Unreal Engine. This system allowed people with limited programming experience to build a fully functional system, without having to modify the internal workings of the game engine itself. The figures presented in the year 2005 for a system developed with this software were in between $25,000 and $30,000 [17]. To this date, CAVEUT still uses the Unreal Tournament 2004 engine which has become outdated, and has been replaced by the Unreal 3 Engine [18]. There is no information on future plans to port this system to the new, more modern game engine.

A similar, newer system based on the Half-Life Engine [19] seemed to offer an alternative to CAVEUT, however, it is not clear what the capabilities, implementation requirements, and costs of such system were, as well as how it could compete with the graphics, physics and 3D model quality of the Unreal Engine.

The active and fast-paced development of games ensures that the most accurate and realistic, state of the art simulation and rendering is achieved in commodity hardware. Furthermore, the ability to easily modify and extend the content and abilities of the game engine with "in-game" editors indicates that game engine-based CAVEs can be a cost-effective solution.

3. CryVE CAVE system

We developed a CAVE system named *Cryengine automatic Virtual Environment (CryVE)* based on the game engine CryEngine2. The engine is a multiplatform game development middleware that provides state of the art, photorealistic real-time visuals and enhanced physics, handling both interior spaces and vast landscapes in single- or multi-player settings [20].

The CryVE system is based on an arrangement of screens resembling a cubic room, with the projection done from the outside of the room. This allows viewers to move around inside without creating undesired shadows on the projection screens. Each projector is controlled by a single computer, which in turn is connected to a computer network (see Fig. 1 for a side view and Fig. 2 for a top view).

The software architecture is similar to that of CAVEUT: multiplayer instances of a CryEngine2 game are started on all computers in the system. Computers are connected to each other through a network hub, with one of them acting as a server (master) while the rest are game clients (slaves). The server can control the in-game action (walking, jumping, shooting, etc.) while the clients provide the extra "cameras" that complete the peripheral view required by the CAVE, by aligning and synchronizing themselves to the pose and motion of the master, effectively enlarging its field of view. Finally each computer renders its piece of the virtual world to the corresponding projection screen as shown in Fig. 3.

CryVE is a software extension (generally known as *mod*) that sits on top of CryEngine2 without modifying its internal workings. Therefore it can be redistributed independently from the game that is being used in the system, and it can be easily extended and mod-

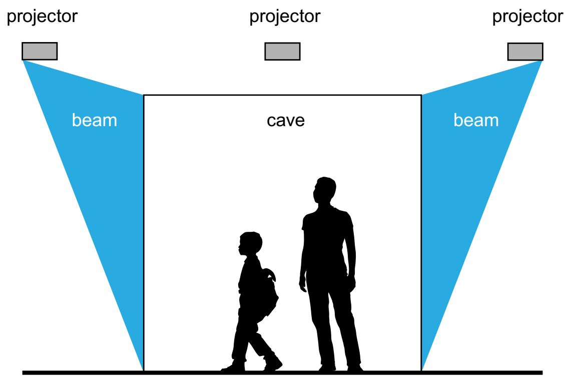


Fig. 1. Side view of physical configuration of the CAVE system.

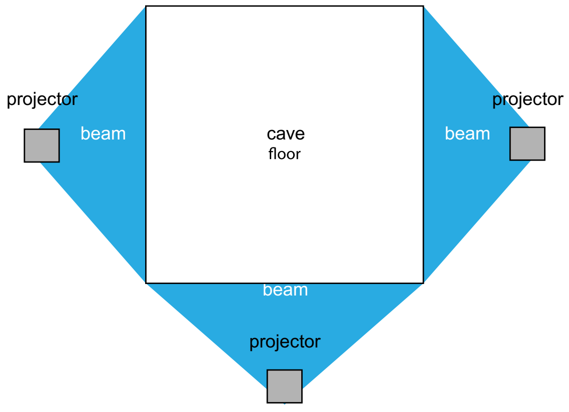


Fig. 2. Top view of physical configuration of the CAVE system.

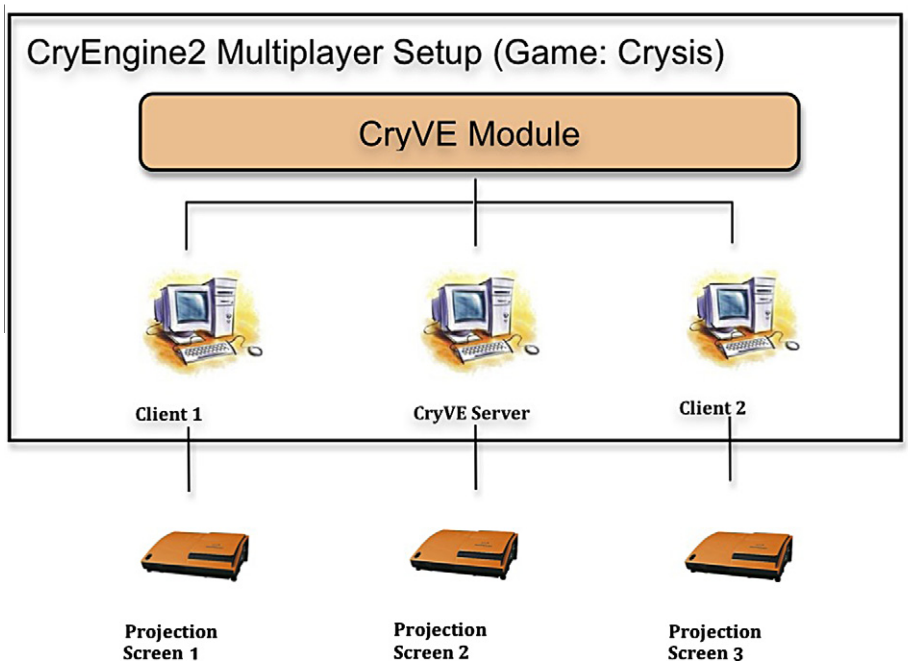


Fig. 3. Proposed CAVE system architecture.

ified without fear of affecting the original game software. Note that, in principle, this means that any computer game that is based on the CryEngine2 engine can be used to run the CryVE mod. Examples of games using CryEngine2 that can run the CryVE software are Crysis, and Crysis Warhead. Other games already available that run the CryEngine2 are Entropia Universe and Blue Mars.

Content creation is easily done by using an in-game editor called *CryEngine Sandbox*[1] that is available freely to download. Also available are extensive documentation and tutorials, as well as a large amount of content created by the gaming community [21]. The editor allows to modify and extend any game map, characters, animations, and behaviors. It is also possible to create totally new environments from scratch and add them to the game.

4. Implementation

As outlined in Section 3, the CryVE system consists of two parts: the physical installation and the software that controls it. In both we faced several challenges that were addressed with pragmatic solutions.

4.1. Physical setup

The CryVE physical installation was built around an aluminum frame that held a plastic white translucent canvas as seen in Fig. 4a. Each side of the room measures 3 × 3 m and five of the six walls are projecting screens, leaving out only the floor. Our first prototype consisted of only three faces of the cube used as projection screens, resulting in a "U" CAVE configuration. The remaining two side projections are currently under construction.

The room enclosing the CAVE installation offered a maximum of 2.2 m of space on each projecting side. Due to this limitation in the physical space available, we used Hitachi ED-A100 XGA projectors, configured to project on XGA resolution (1024 × 768 pixels). These projectors offer the advantage of short throwing distance and easy image adjustment to cover the square canvas of the projection screen. Furthermore, the back projection nature of these devices allowed us to mount them at 4.0 m above ground outside of the room, enabling free transit around the CAVE without undesired shadows appearing on the canvas (see Fig. 4b).

Each projector was then connected to a computer that controlled one of the projected faces of the room. Table 1 details the hardware specifications or the machines used in the setup (all computers were identical). Each station was then connected to each other through a 1 Gbit network, using Gigabit Ethernet adapters for each computer, twisted pair cabling (Cat-5E), and a Gigabit switch to ensure enough bandwidth for a smooth game experience.

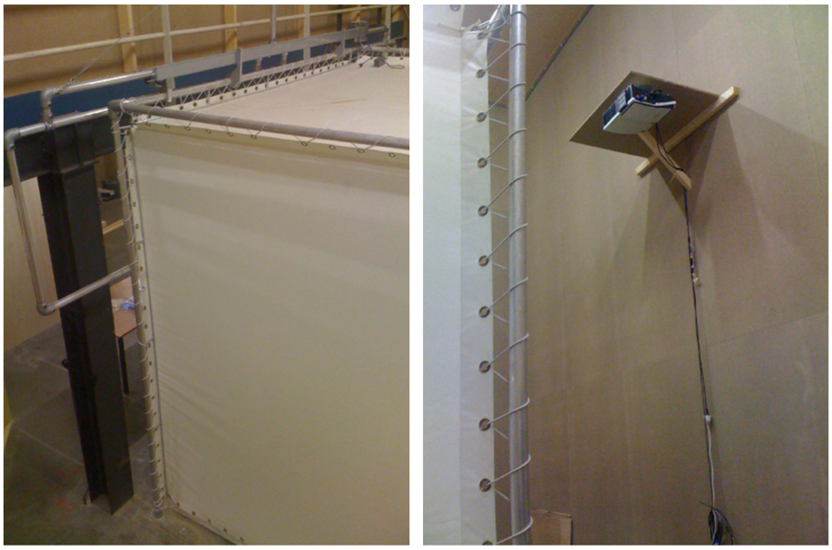
4.2. Software setup

The software developed for the prototype was based on the game Crysis Wars [22], a first person shooter game with multiplayer support that was released as an extension of the original title of the Crysis series. It was also successfully tested with the original Crysis game.

The CryVE software implementation consists of three components: a game modification (*mod*), a game *flowgraph* and a modified multiplayer map. A CryEngine2 game mod is a piece of code (usually written in C++) that can access the low-level data structures and API of the game engine, and extends its functionality by modifying the behavior and appearance of characters, and even the gameplay itself. A *flowgraph* is a graphical representation of a script that allows different *mods* to be called upon, and be connected to other components/mods available in the game system. A game *map* is a representation of the virtual world that includes *flowgraphs*, characters, assets, textures, 3D models, player behaviors and all other elements that will be loaded by the game engine upon game start.

The CryVE mod component is in charge of deciding if a specific instance of a game is a server or client in the CryVE setup. If the current instance is a master, the mod sends a signal to potential clients (other slaves) that it is available to connect. If the instance is a client, the mod obtains the gaze of the master and aligns the

[1] The CryEngine Sandbox can be freely downloaded from the official Crytek modding website <http://crymod.com>.



(a) Steel frame holding the canvas. (b) Projectors mounted 4.0 meters above ground.

Fig. 4. Physical installation of the CryVE system.

Table 1
Hardware specification of the computers used in the CryVE setup.

Computer specifications	
Operating system	Windows XP Professional SP3
CPU	Intel Core 2 Duo E4800 3 GHz
Memory	3.25 GB
Graphics	Nvidia Quadro FX 3700 512 MB DDR3 dedicated memory
Networking	PCI-based 10/100/1000 Mbps Gigabit Ethernet Network Adapter Card

camera view of the client accordingly. Once the cameras are aligned, CryVE reads a configuration file for the required image transformation (plane translation, rotation and frustum shape transformation in 3D space). This process is called *system calibration*.

In order to calibrate the system appropriately, there are some parameters that must be calculated depending on the geometry of the CAVE and the desired viewing position inside it. These parameters are the field of view (FOV), yaw, pitch, and roll of the projection. Penna [23] showed that the yaw, pitch and roll parameters for the perspective projection of a quadrilateral can be calculated using

$$\alpha = \tan^{-1}(r_{12}/r_{11})$$
$$\beta = \tan^{-1}(-r_{13}/(r_{11}^2 + r_{12}^2)^{1/2})$$
$$\gamma = \tan^{-1}(r_{23}/r_{33})$$

where r_x are the components of a 3×3 matrix that defines the desired rigid motion rotation of the projection plane.

As the prototype implementation is a cubic CAVE, the desired point of view was fixed at the center of the cube (see Fig. 5). This simplifies the calculation of the camera look view parameters, resulting in

$$\alpha = \pi/2$$
$$\beta = 0$$
$$\gamma = 0$$

The calculation of the vertical and horizontal FOV is done by applying the formulae

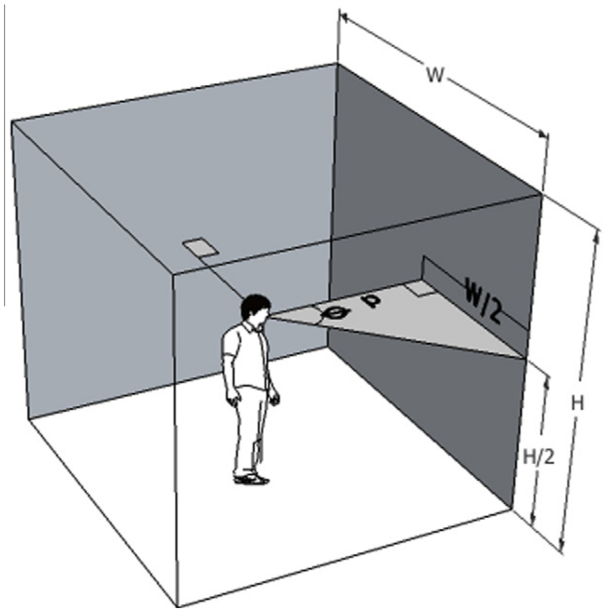


Fig. 5. Dimensions used to calculate the configuration parameters in the CryVE prototype.

$$\mathrm{FOV}_{\mathrm{vertical}} = 2\theta = 2(\tan^{-1}(H/2p))$$
$$\mathrm{FOV}_{\mathrm{horizontal}} = 2\phi = 2(\tan^{-1}(W/2p))$$

However, given the cubic shape of the CAVE, we know that $\mathrm{FOV}_{\mathrm{vertical}} = \mathrm{FOV}_{\mathrm{horizontal}}$. Furthermore, we observe that $p = W/2 = H/2$, therefore, the formula for FOV (both horizontal and vertical) can be simplified to

$$\mathrm{FOV} = 2(\tan^{-1}(H/2p)) = 2(\tan^{-1}(W/2p)$$
$$\mathrm{FOV} = 2(\tan^{-1}(1))$$
$$\mathrm{FOV} = \pi/2$$

The calculation of the parameters for each face of the cube is done in a similar way. After this, the process of translation and rotation is applied to each image frame before is rendered by the clients. Fig. 6 shows the process diagram for the CryVE mod.

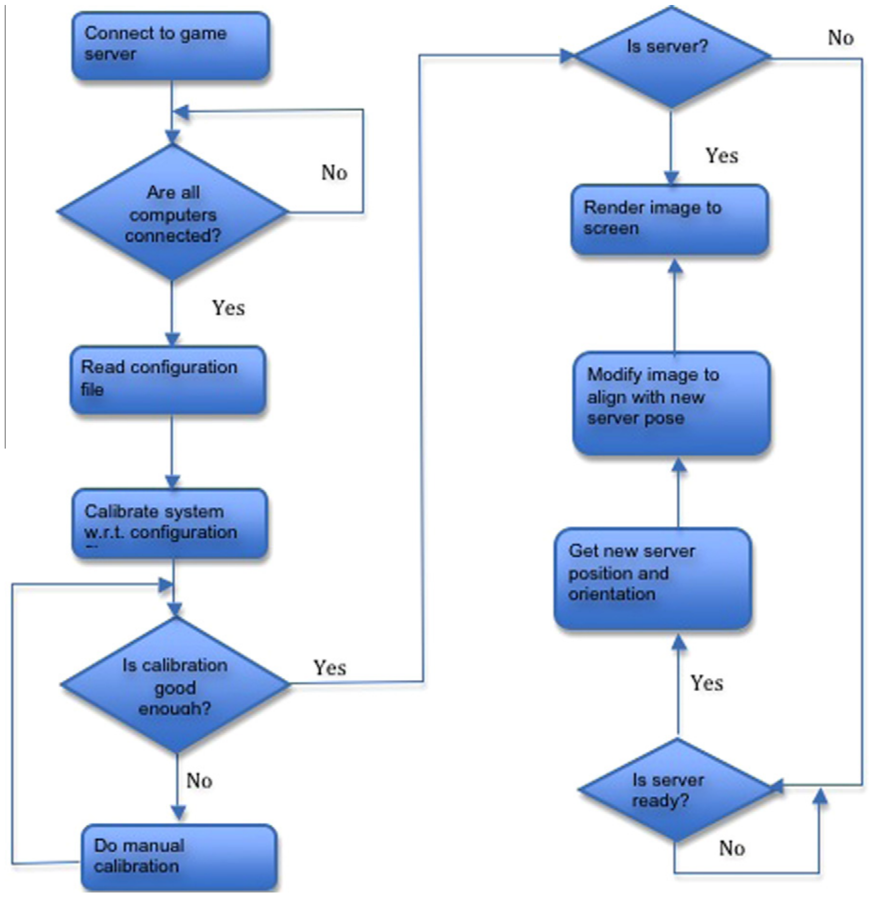

Connect to game server
Are all computers connected?
No
Yes
Read configuration file
Calibrate system w.r.t. configuration
Is calibration good enough?
Yes
No
Do manual calibration
Is server?
No
Yes
Render image to screen
Modify image to align with new server pose
Get new server position and orientation
Yes
Is server ready?
No

Fig. 6. CryVE mod process diagram.

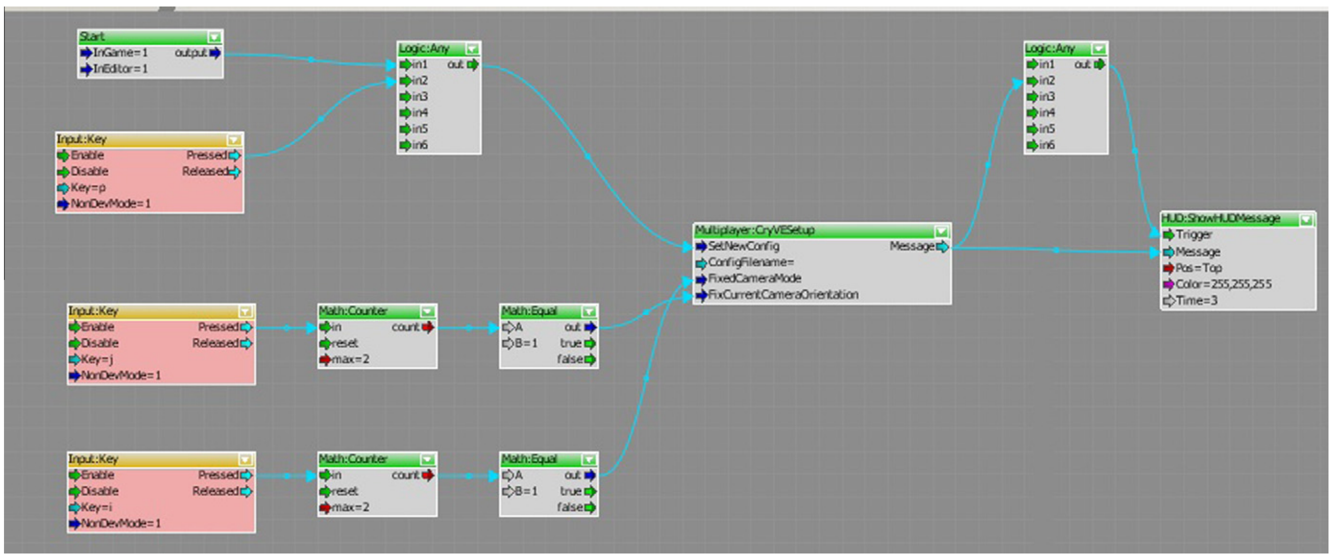


Fig. 7. CryVE prototype flowgraph.

The CryVE flowgraph allows to encapsulate the libraries produced by the mod into a component that can be reused in any CryEngine2 game. The flowgraph defines input and output ports for the mod and connects them to other ingame components like player_hud, player_id, player_position, and player_stance. Fig. 7 shows the resulting flowgraph in the CryVE plugin, along with other components and flowgraphs already available in the game engine.

Every flowgraph must be encapsulated in a map for it to be used within a game. The CryVE prototype modified a single player medieval map called Niveus [24] into a multiplayer map containing the flowgraph and mod implementations. The map was installed in all the computers in the network. Fig. 8 shows a snapshot of the map used in the prototype, Fig. 9 shows a user inside the CAVE where the map is being rendered.

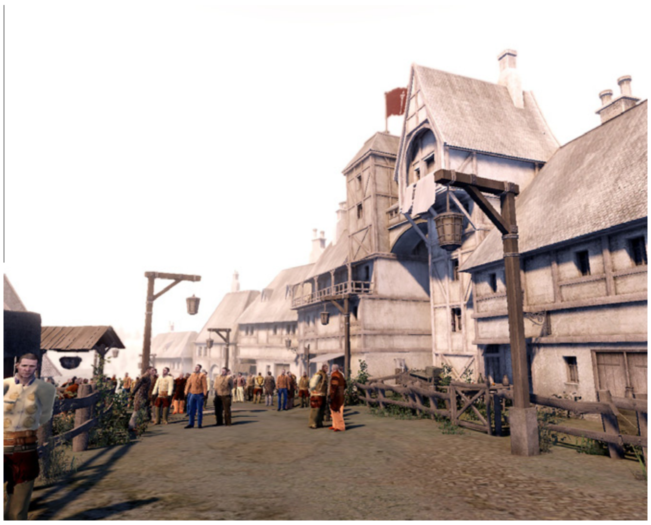


Fig. 8. Snapshot of the medieval map used to test the CryVE setup.

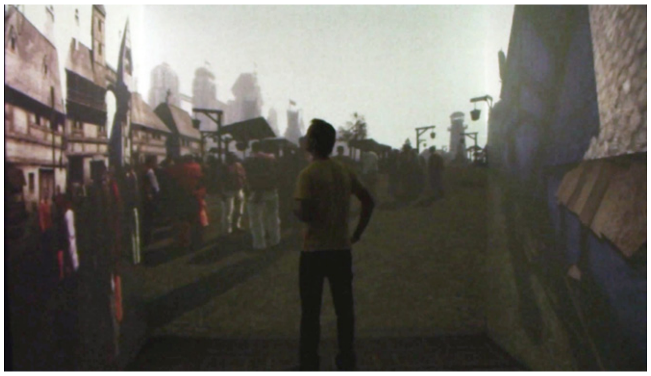


Fig. 9. A visitor inside the CAVE.

All the software for this implementation, as well as additional information can be found at the website <http://cryve.id.tue.nl>.

5. Use case: a virtual museum tour

Based on the prototype implementation of the CAVE, we developed a conceptual design of an exhibition for the Nationaal Historisch Museum in the Netherlands. This museum is an initiative that aims to engage the general public into discovering and appreciating their cultural heritage, by combining latest media and technology with more traditional museum approaches. The exhibition consists of a virtual museum tour that reproduces events of the history of the Netherlands.

Visitors of the museum can use a portable controller in combination with CryVE to have an immersive experience. As a visitor enters the CAVE, the handheld device switches into a controller, enabling the navigation through, and interaction with the virtual world. Using the handheld device, the visitor walks through the landscape and meets people (avatars) with whom he or she can interact. The handheld device is used as interface for interaction. It presents a selection of possible questions that can be asked to the avatars. The avatars will respond to the question, adjusting dynamically to the visitor's selection. In a subtle way it will introduce the visitor to the historic event that is about to happen in the virtual world.

Other museum visitors can also join the exhibition. Each visitor has its own audio, and is able to select it's own interaction with people in the virtual world through the hand held device. Having separate audio and shared visuals in the virtual world, allow visitors to have both individual and group experiences. When a group of visitors is present, the control of the virtual world is taken from the visitors and transferred back to the system, eliminating conflictive situations with visitors trying to achieve different goals. The handheld devices, however, are still able to individually interact with people and objects in the virtual environment.

At one point during the exhibition, the visitors are asked to join a shared event, where every handheld device will have the same audio. This is the start of the historic event. If one or more visitors decide not to enter the shared event, they can step out of the immersive environment. When the shared event is over, the visitor can continue with other exhibitions in the museum.

The interactive exhibition based on the CAVE system allows visitors to become participants instead of mere spectators. It enables them to walk around and interact in a dynamic virtual world that recreates 16th century Holland. They can actively experience an event that took place in real life both individually and collectively.

6. Results

The physical setup of the CryVE system showed that it is possible to build a full-size, high-performance CAVE with commodity hardware. Up to five people are able to stand comfortably inside the closed projection room, enjoying an immersive experience.

The extension module that enables the calibration and screen alignment of the clients with respect to the server computer, can be easily configured using a file with directives that indicate the desired translation and rotation of the projected screen. The fact that the CryEngine2 includes a free-to-use editor with every game purchased, means that virtually everyone is able to use, modify and extend the module. Furthermore, the availability of a visual process editor (flowgraph) and extensive community support of gamers around the world makes the learning curve of this software much less steep than other similar CAVE applications. The subjective appreciation of the performance of the system in terms of the realism of computer graphics, physics simulation and overall user experience, is much improved when compared with similar applications like CAVEUT.

We measured the performance of the CryVE system in terms of average frames per second (FPS), CPU and memory usage. The CPU performance was measured in percentage of CPU used during system operation and the memory was measured in percentage of the total RAM used in the master computer by the processes that run the game (see Table 2).

From these results it can be observed that the framerate does not suffer excessively as new machines are added to the network, while the CPU and memory usage keep reasonably low. This results can be attributed to the nature of the CryVE setup: taking advantage of the multiplayer game properties of the CryEngine2, the computers in the network only send the necessary information for the slaves to react to events in the master, all the "heavy-duty" processing is done in each slave computer and rendered directly to the corresponding projector.

Due to the different technologies used by other CAVE systems, our performance measurements are not directly comparable to,

Table 2
Performance of CryVE system in average FPS, CPU, and memory usage.

No. of computers	FPS	CPU (%)	Memory usage (MB)
1	19.5	0.54	580
2	18.6	0.59	595
3	18.1	0.61	599

Table 3
Detail of estimated costs of the basic CryVE installation.

Item	Cost (EUR)
Canvas	6700
CAVE structure	3600
PCs	4600
Projectors	4100
Networking and other materials	300
Total	19,300

for example, the performance measurements of ClusterJuggler in [25]. However, we believe that they can give an indication of the kind of performance that can be obtained from commodity hardware running a state-of-the-art game engine-based CAVE.

With respect to the cost of the system, we contacted three commercial companies and inquired the price of their CAVE system that would meet our requirements. The requirements were based on our current implementation of CryVE. We received two responses (160,000 Euros and 630,000 Euros). The price range is considerable and is far above our system.

The low-cost nature of the CryVE system was confirmed by the figures obtained from the prototype: The basic, fully functional CAVE installation described in this paper costs around 19,300 Euros.[2] This makes our system comparable to CAVEUT in terms of overall cost, but with improved graphics, physics, simulation and overall experience that the CryEngine2 engine offers over the version of Unreal Engine used by CAVEUT. Table 3 presents a detail view of the system price.

In our experience, it was rather difficult to acquire the offers from the commercial companies we contacted. Furthermore, these companies do not estimate possible to produce such system for less money than the quote they provided. We believe this strengthens the case of our low-cost CAVE system as a viable option for institutions with reduced budgets.

Finally, we would like to mention that at the moment of writing, CryEngine has released a new version called CryEngine3. According to the CryEngine3 specification brochure [26], the changes with respect to CryEngine2 target the execution of games in multiple platforms like Xbox 360, Sony Playstation3, and PC. As the engine API does not report dramatic changes, it is safe to say that porting CryVE to CryEngine3 should be a matter of recompiling the C++ code of the *game mod*, using the new engine platform. This opens new opportunities for the use of CryVE as a multi-screen, immersive gaming platform that can be run in the most popular gaming consoles.

However, there is a word of caution. In the past, game mods have been developed by (and distributed to) the gaming community using the CryEngine Sandbox. CryEngine3 is set to include a new version of the Sandbox software shipped with every purchased game. This new Sandbox application will, in principle, allow the deployment of mods like CryVE to both PC's and consoles like XBox 360 or PlayStation3. Unfortunately, this can only be tested once the first game that uses the CryEngine3 appears on the market.

7. Limitations

When the master computer (the game that acted as CryVE server) moved in the virtual world (specially with rotational motion), valleys in framerate, and peaks in CPU and memory usage occurred, sometimes accompanied by a "lag" in the expected motion in the slave computers. While this was rarely perceivable from the user point of view, we acknowledge that this behavior must be further studied in order to find a proper explanation and solution.

In addition to this, the calibration step needed to properly align the images in the different faces of the CAVE must be revised: we successfully implemented an automatic configuration mechanism that can read the desired position, orientation and frustum shape of the virtual world view for a client machine in the CryVE system, however, the alignment of the planes sometimes needed to be corrected manually to achieve a perfect result. This suggests that the automatic calibration process needs to be improved in future releases of the CryVE software.

8. Conclusion

We presented an implementation of a low-cost CAVE system that uses the game engine CryEngine2 as the underlying software. This offers several advantages over other existing implementations, e.g. state of the art, photorealistic graphics and physics, easy of use and extensibility.

The physical construction of the CAVE and the use of short throwing distance projectors enables its use in environments with limited space available while, at the same time, maintains life-size projection screens for a better immersion experience. The low-cost nature of the CryVE system was confirmed by the figures obtained from the prototype (19,300 Euros), compared to the quotes obtained for two commercial solutions (160,000 Euros and 630,000 Euros).

The improved visuals and performance of the CryEngine2 over the game engines used by similar systems like CAVEUT, confirm CryVE system as solid alternative to bring state of the art technology and photorealistic graphics to low-cost CAVE systems. CryVE is a low-cost implementation that offers advantages to the scientific, gaming, and graphics enthusiasts communities. Therefore we invite them to use and extend it.

[2] This estimate does not cover the cost of man-hours of work invested during the installation process.

References

[1] J. Leigh, A. Johnson, T. DeFanti, S. Bailey, R. Grossman, A tele-immersive environment for collaborative exploratory analysis of massive data sets, in: Proceedings of the ASCI, Citeseer, vol. 99, 1999, pp. 3–9.
[2] J. Freeman, G. Watson, Y. Papelis, T. Lin, A. Tayyab, R. Romano, J. Kuhl, The Iowa driving simulator: an implementation and application overview, SAE Transactions 104 (1996) 113–122.
[3] J. Loomis, J. Blascovich, A. Beall, Immersive virtual environment technology as a basic research tool in psychology, Behavior Research Methods Instruments and Computers 31 (4) (1999) 557–564.
[4] B. Solutions, i-Space Cave System. <http://www.barco.com/projection_systems/downloads/VRandimmersive_march06.pdf> (cited January 2010).
[5] HoloVis, Cave Solutions. <http://www.docstoc.com/docs/22604870/HoloVis—CAVE-Solutions_V1> (cited January 2010).
[6] J. Leigh, G. Dawe, J. Talandis, E. He, S. Venkataraman, J. Ge, D. Sandin, T. DeFanti, Agave: access grid augmented virtual environment, in: Proceedings of the AccessGrid Retreat, Argonne, Illinois, 2001.
[7] J. Jacobson, M. LeRenard, J. Lugrin, M. Cavazza, The caveut system: immersive entertainment based on a game engine, in: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology, ACM, 2005, p. 187.
[8] C. Cruz-Neira, D. Sandin, T. DeFanti, R. Kenyon, J. Hart, The Cave: Audio Visual Experience Automatic Virtual Environment.
[9] P.M. Sauter, Vr2go: a new method for virtual reality development, SIGGRAPH Comput. Graph. 37 (1) (2003) 19–24. <http://doi.acm.org/10.1145/763993.763995>.
[10] R. Darken, D. Bernatovich, J. Lawson, B. Peterson, Quantitative measures of presence in virtual environments: the roles of attention and spatial comprehension, CyberPsychology & Behavior 2 (4) (1999) 337–347.
[11] K. Fernandes, V. Raja, J. Eyre, Cybersphere: the fully immersive spherical projection system, Communications of the ACM 46 (9) (2003) 141–146.
[12] K. Park, Y. Cho, N. Krishnaprasad, C. Scharver, M. Lewis, J. Leigh, A. Johnson, CAVERNsoft G2: a toolkit for high performance tele-immersive collaboration, in: Proceedings of the ACM Symposium on Virtual Reality Software and Technology, ACM, 2000, pp. 8–15.
[13] C. Cruz-Neira, A. Bierbaum, P. Hartling, C. Just, K. Meinert, VR Juggler-an open source platform for virtual reality applications, in: 40th AIAA Aerospace Sciences Meeting and Exhibit, Citeseer, 2002.

[14] R. Belleman, B. Stolk, R. deVries, Immersive virtual reality on commodity hardware, in: Proceedings of the 7th Annual Conference of the Advanced School for Computing and Imaging, Citeseer, vol. 7, 2001, pp. 297–304.
[15] P. Lin, Z. Pan, J. Yang, J. Shi, Implementation of a low-cost CAVE system based on networked PCs, in: Proceedings of the VE on PC, St Petersburg, 2002, pp. 33–40.
[16] M. Green, L. White, The Cave-let: a low-cost projective immersive display, Journal of Telemedicine and Telecare 6 (Supplement 2) (2000) 24.
[17] J. Jacobson, M. Lewis, Game engine virtual reality with caveut, IEEE Computer 38 (4) (2005) 79–82.
[18] Unreal Technology. <http://www.unrealtechnology.com/features.php?ref=technology-overview> (cited January 2010).
[19] T. Schou, H. Gardner, 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, ACM, 2007, p. 234.
[20] H. Seeley, Game technology 2007: Cryengine2, in: ACM SIGGRAPH 2007 Computer Animation Festival, ACM, 2007, p. 64.
[21] Crymod Database. <http://www.crymod.com/> (cited January 2010).
[22] E. Arts, Crysis Game. http://www.ea.com/games/crysis> (cited January 2010).
[23] M. Penna, Determining camera parameters from the perspective projection of a quadrilateral, Pattern Recognition 24 (6) (1991) 533–541.
[24] Niveus Crysis Map. <http://crymod.com/thread.php?threadid=36684> (cited February 2010).
[25] P. Morillo, A. Bierbaum, P. Hartling, M. Fernández, C. Cruz-Neira, Analyzing the performance of a cluster-based architecture for immersive visualization systems, Journal of Parallel and Distributed Computing 68 (2) (2008) 221–234.
[26] Crytek, Cryengine3 Specifications. <http://www.crytek.com/technology/cryengine-3/specifications/> (cited March 2010).