Gesture-Based, Touch-Free Multi-User Gaming on Wall-Sized, High-Resolution Tiled Displays

Daniel Stødle, Tor-Magne Stien Hagen, John Markus Bjørndalen, Otto J. Anshus
Dept. of Computer Science
University of Tromsø
N-9037 Tromsø, Norway
phone, e-mail: +47 77 64 42 22
{daniels, tormsh, jmb, otto}@cs.uit.no

Abstract

Having to carry input devices can be inconvenient when interacting with wall-sized, high-resolution tiled displays. Such displays are typically driven by a cluster of computers. Running existing games on a cluster is non-trivial, and the performance attained using software solutions like Chromium is not good enough.

This paper presents a touch-free, multi-user, human-computer interface for wall-sized displays that enables completely device-free interaction. The interface is built using 16 cameras and a cluster of computers, and is integrated with the games Quake 3 Arena (Q3A) and Homeworld. The two games were parallelized using two different approaches in order to run on a 7x4 tile, 21 megapixel display wall with good performance.

The touch-free interface enables interaction with a latency of 116 ms, where 81 ms are due to the camera hardware. The rendering performance of the games is compared to their sequential counterparts running on the display wall using Chromium. Parallel Q3A's framerate is an order of magnitude higher compared to using Chromium. The parallel version of Homeworld performed on par with the sequential, which did not run at all using Chromium. Informal use of the touch-free interface indicates that it works better for controlling Q3A than Homeworld.

First presented at the 4th Intl. Symposium on Pervasive Gaming Applications, PerGames 2007. Revised for JVRB.

Keywords: Display wall, multi-touch, device-free, parallelized games

1 Introduction

Wall-sized, high-resolution tiled displays are becoming increasingly common in locations ranging from visualization labs to public spaces. Often, having to carry input devices around in order to interact with applications running on a display wall can be inconvenient. Devices like mice or Nintendo Wiimotes are easily misplaced, and for public installations there is the risk of theft. Asking users to wear optical or electronic markers raises the bar for casual users. Instead, a completely device-free approach to interacting with wall-sized displays is necessary.

Display walls provide high resolution by tiling a set of independent displays in a grid. Each display is usually driven by a computer in a display cluster[LCC+00]. The resolution of a typical desktop display is about 2-3 megapixels, while the resolution of a display wall ranges from 10 to 100 megapixels [LCC+00, SW06] and beyond. The display wall used in this paper is comprised of 28 projectors, each driven by one computer and arranged in a 7x4 grid, for a total resolution of 7168x3072 pixels.

For games, high framerates are important [CCD06]. Maintaining high framerates becomes increasingly difficult as the resolution goes up. Further, the cluster-based architecture of display walls makes running ex-

isting games difficult, as very few, if any, games are written to run on a cluster of computers or use more than a few displays. Chromium [HHN+02] can be used to make the display wall appear as a single display to OpenGL-based applications, but at the cost of sub-optimal performance. In addition, not all software works with Chromium.

This paper presents a device- and touch-free multi-user human-computer interface for display walls. Users standing in front of such a display wall can interact with applications directly using hand- and arm-gestures without the need for any devices and without having to wear markers of any kind. The interface has been integrated with two commercial, but now open-source games[1]: Quake 3 Arena (Q3A) [iS08] and Homeworld [Ent08], respectively a first-person shooter (FPS) and a real-time strategy (RTS) game. Games were chosen because they generally require low-latency input to be playable. If the touch-free interface does not provide sufficiently good accuracy or low latency, the games will become unplayable. The two games were parallelized in order to run on the display wall with good performance, and modified to accept position data from the touch-free interface. Figure 1 shows two persons playing Q3A against each other on a display wall. The person in the middle is playing Homeworld.

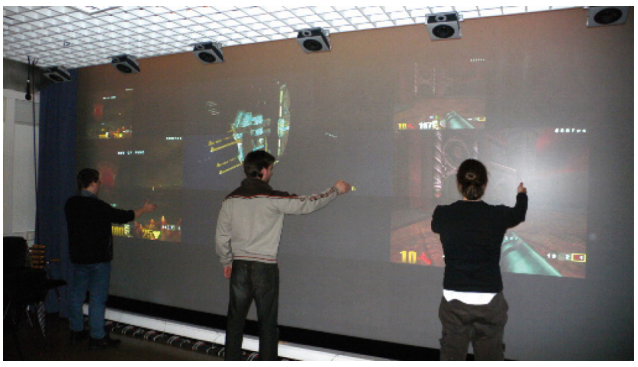


Figure 1: Two persons playing Q3A and one person playing Homeworld simultaneously on a 7x4 tile display wall. Q3A runs on 2x2 tiles to the left and right, and Homeworld on 3x3 tiles in the middle.

The interface uses 16 cameras and 9 computers to detect objects in front of the display wall, and is able to detect multiple objects simultaneously at a rate of 30 Hz. When three or more cameras see the same object, triangulation can be used to determine the object's position. The interface is referred to as touch-free, as users can interact with the display wall without actually touching its canvas. This is an important advantage over existing solutions that require touch to work [Han05], as the canvas used for our display wall is flexible and thus prone to perturbation when users touch it. The interface's main advantage over other approaches, like the IS-900 tracking system [Int08], is that it is completely device-free. Users need not wear markers to accommodate the interface, but can instead walk directly up to the display wall and start interacting. This is particularly important for public installations, where markers or other input devices might easily get lost or stolen. Even in a lab setting it is easy to misplace input devices, or confuse the different input devices with each other ("Which mouse/Wiimote is the correct one? Where did I leave it?").

Two different approaches were used when parallelizing Q3A and Homeworld. For both Q3A and Homeworld, a copy of the game runs on each tile. Each copy's OpenGL view frustum is modified in accordance with the tile it runs on to create a coherent, multi-tile view. For Q3A, the existing client-server based architecture combined with the concept of spectators was exploited. The server keeps all the clients in sync, and the spectator-concept enables different clients to be configured so as to constantly follow a given player. For Homeworld, a state-synchronizing, master-slave approach was taken. Each copy shares a global clock and random number generator seed. The master distributes all input to the different slaves, with the purpose of having all copies compute the exact same game state for each new frame.

Experiments were conducted to measure the latency of the touch-free interface, as well as the framerate of the two games. The experiments show that the time before an object's position is available to the games averages 116.7 ms, with the majority of this latency incurred by the FireWire-based cameras. Game-side gesture-processing did not incur significant latencies, due to the simple gestures involved. For Q3A, the framerate is shown to be as much as an order of magnitude better than using Chromium. Homeworld's framerate remains high in the parallel configuration, and outperforms the single-display configuration when running on both 2x2 and 3x3 tiles. Homeworld did not work with Chromium at all.

The main contributions of this paper are (i) a distributed, device- and touch-free multi-user interface,

[1]Only the game engines are open source. The data files still require a license.

(ii) two approaches to parallelizing games for a display wall environment, demonstrating how different aspects of the two games' existing architectures can be exploited, (iii) a prototype system for gesture-based input to games in the FPS and RTS genres, (iv) an evaluation of the interface's responsiveness when used to interact with two games, and (v) evaluation of three different approaches for making existing games run on display walls.

2 Related Work

The Quake-series of games have been popular targets for modification and extension, both in terms of input devices and display surfaces. Some examples include playing Quake using Nintendo's Wiimote, using eye-tracking to play Quake, or controlling Quake from a PDA[2]. CaveQuake is a limited re-implementation of Quake II and Q3A for use in a CAVE[3], but does not support all the features of the full games, and for the Q3A case does not even support playing. In [KLJ04], the authors present a gesture-based interface to Quake 2. The interface is limited in that only one person may use it at a time, and differs from the touch-free interface presented in this paper by the use of whole upper-body gestures. The touch-free interface only enables hand- and arm-gestures. We are not aware of any work to integrate new input devices or new display surfaces for Homeworld.

In [BBH05], a gaming interface based on a commercially available stool, "The Swopper," is presented. The stool and a light gun is used to produce joystick input events to control an FPS game. By shifting the body weight and rotating on the stool in combination with aiming and firing the gun, the user can navigate and interact with the world. The touch-free interface does not require the use of any external devices, and the large display wall makes it possible to have multiple players playing side-by-side simultaneously. The stool-and-light-gun approach is more expressive compared to the gestures recognized by the touch-free interface.

Gesture VR [SK98] is a video-based, hand-gesture recognition system. The system recognizes three gestures which are used to provide applications with different input events, as demonstrated by controlling Doom, an FPS game developed by id Software. Their solution is centralized, using two synchronized cameras connected to a single computer. The touch-free interface comprises 16 cameras connected to 8 computers, enabling it to cover a larger area at the cost of a more elaborate hardware setup. The touch-free interface only recognizes simple gestures (2D position and radius of detected objects), while Gesture VR allows for detection of 3D position and three different gestures.

In [TGSF06], the authors argue that a digital table is a conductive form factor for general co-located home gaming. By combining speech and hand gestures as input to two commercial games, The Sims and Warcraft III, several persons can interact with the games running on the tabletop. The touch-free interface is based on hand- and arm-gestures alone on wall-sized displays. The physical dimensions of the display wall enables more than a couple of people to play simultaneously, against each other or co-operatively. Further, we have modified the source of the two games, enabling more flexible multi-point interaction. The games used in [TGSF06] are not open source, requiring that custom wrappers are built that translate touch- and speech input to mouse and keyboard events. In [SZP+00], the authors demonstrate a bimodal speech- and gesture-based interface for interacting with a 3D-visualization. Apart from the speech-aspect, this system differs from the touch-free interface in that it supports only one user at a time and has a far more limited area in which interaction can take place.

The authors of [TGSF06] use the Diamondtouch [DL01] tabletop for multi-touch interaction. Other technologies for multi-touch interaction include [Han05], where infrared light is projected into a canvas and internally reflected. The internal reflection of the light is frustrated at points where the user touches the canvas. The escaping light can be detected using a camera mounted behind the canvas. The touch-free interface is based on detecting the presence of objects directly, and does not require the user to actually touch the display wall's canvas. In [Mor05], the author presents a camera-based solution to detecting and positioning objects in front of a whiteboard called the "SmartBoard." The approach is similar to the touch-free interface, except that the touch-free interface utilizes a distributed approach with 16 commodity FireWire cameras connected to a set of computers, whereas the SmartBoard uses custom cameras with on-chip processing to perform object recognition.

[2] http://www.youtube.com/watch?v=n1tsXc2RoeM
http://www.youtube.com/watch?v=3pRWYE2LRhk
http://www.youtube.com/watch?v=tNJXjNBgmLs

[3] http://www.visbox.com/cq3a/

Chromium [HHN+02] is a system for distributing streams of rendering commands, allowing many existing OpenGL applications to run on tiled display walls without modifications. Chromium works by conceptually making the individual tiles of a display wall appear as a single, logical display to the application. By making applications use Chromium's OpenGL library, Chromium can intercept rendering commands and forward them to remote rendering nodes. Homeworld did not run using Chromium, and Chromium's rendering performance running Q3A did not scale well beyond 2x2 tiles.

In CaveUT [JH02], a set of modifications to Unreal Tournament is presented that allows it to display in panoramic theaters. The same principle of using spectators to support multi-tile rendering is applied as employed by the parallel version of Q3A. However, no measurements of the resulting performance are presented. This paper presents measurements of the Q3A's framerate and documents the latency incurred by using spectators.

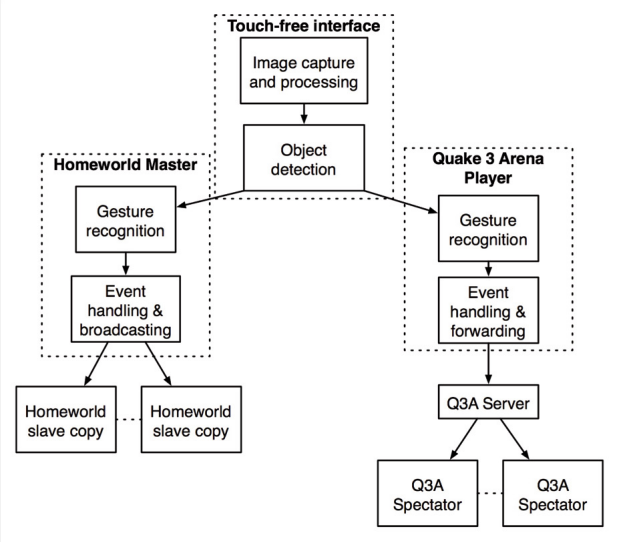


Figure 2: The design of the touch-free interface, and its use with Q3A and Homeworld.

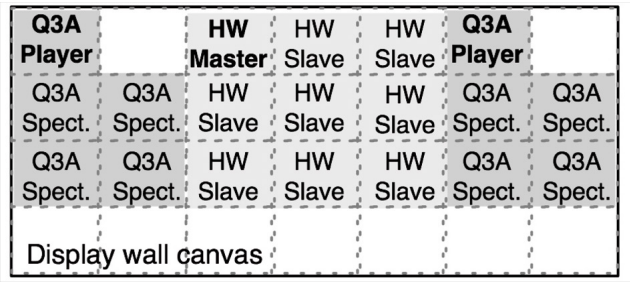


Figure 3: Running Q3A and Homeworld on a 7x4 display wall. To the left and right, two Q3A players control a set of Q3A spectators. In the middle, a single Homeworld master synchronizes the rendering and game simulations of 8 Homeworld slave copies.

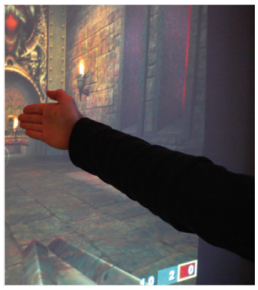
3 Design

Quake 3 Arena [iS08], developed by id Software, is an open-source first-person shooter designed for multiplayer gaming. It is based on a client-server architecture where the server maintains the state of the game. At a fixed rate, independent of the connected clients, the server updates its game state, before broadcasting state changes to connected clients. Clients use this to update their view of the game. A client in Q3A is either a player or a spectator. A player is a client that participates in the game. A spectator is a client that instead of participating, follows one of the players around and displays that player's view of the game.

Homeworld [Ent08] is a 3D real-time strategy game developed by Relic Entertainment. In September 2003, the Homeworld engine was made open source. Although the Linux version still lacks some of the features of the complete game, including software rendering, cut-scene playback and networked multiplayer support, the game itself is fully playable in single-player mode. In contrast to Q3A, Homeworld has a monolithic design, with all code running inside a single process.

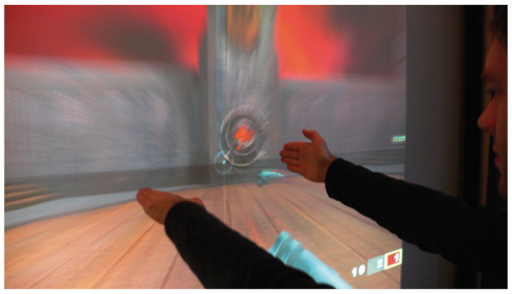
Figure 2 shows the overall design of the touch-free interface, and its use with Q3A and Homeworld. Images are captured and then analyzed to locate objects in a plane parallel to the display wall's canvas. The positions of these objects are then processed by an object detector that yields the object's 2D position and radius, before the resulting information is sent to the two games. The two games process the data individually, using object positions and radii to detect gestures and handle them in game-specific ways.

The design of the parallelized Q3A uses a modified player that receives input from the touch-free interface. The player uses the positions received to recognize gestures, and converts them to keyboard and mouse events suitable for the game. The player relays its actions to the Q3A server, which then updates all clients with the new game state. This causes the spectators following a given player to update their view.



(a) Using a "vertical" hand to control the player's aim.

(b) Controlling aim and firing by making the hand flat.



(c) Moving and aiming simultaneously using both hands.

Figure 4: Gestures for controlling Q3A .

For Homeworld, a single copy is elected as a master. The master becomes responsible for accepting and interpreting input from the touch-free interface. After recognizing gestures, the resulting input is handled and broadcast to the slave copies. Figure 3 shows one configuration of a 7x4 tiled display wall where two users can play Q3A against each other, while a third user simultaneously plays Homeworld. This configuration is identical to the one pictured in Figure 1. Several other configurations are also possible.

3.1 Hand- and arm-gestures

When playing an FPS using a mouse and keyboard, the mouse is used to aim and fire, and the keyboard is used for movement. In addition, the mouse's scroll wheel is often used to switch weapons, and the keyboard to control other actions the player can take (ducking, jumping, etc.). The following gestures, summarized in Table 1, were used for controlling Q3A. When only one hand is detected by the input system, its position is used for controlling the player's aim. When the hand is tilted (making it flat), it will additionally fire the player's weapon. When two hands are detected, the right hand controls aim and firing, and the left hand is used to move the player forwards or backwards. Figure 4 illustrates the gestures.

Action	Gesture
Aim	Move right/only hand
Fire weapon	Flat right/only hand
Move forward	Vertical left hand
Move backward	Flat left hand

Table 1: The gestures in Q3A that the game recognizes and maps to actions.

Homeworld uses a different control scheme. When using a keyboard and mouse, the main controls can all be accessed with the mouse, and the keyboard is mostly used for shortcuts for different menu selections and buttons. When no mouse buttons are pressed, the mouse simply controls an on-screen cursor. Holding down different mouse buttons, the user can pan and zoom the camera, as well as select entities and manipulate them from a contextual menu.

Action	Gesture
Control cursor position	Move right/only hand
Select/click entities	Flat right/only hand
Pan view/contextual menu	Flat left hand
Toggle tactical view	Vertical left hand
Zoom	Flat left and right hand, distance between hands control zoom factor

Table 2: Actions in Homeworld and their corresponding gestures.

Table 2 lists the different actions in Homeworld, and their mapping to gestures. The cursor is controlled using a one-to-one mapping from hand location to the display wall. When the right/only hand is flat (like the fire-gesture in Q3A), the user can select or click items. The user can enter or leave Homeworld's tactical view using a vertical left hand. With a flat left hand, the user can either invoke Homeworld's contextual menu (for moving ships, creating formations, and so on), or panning the camera (by simultaneously moving the right hand). Finally, the user can zoom the camera in and out using a flat left and right hand, varying the distance between them to control the amount of zoom.

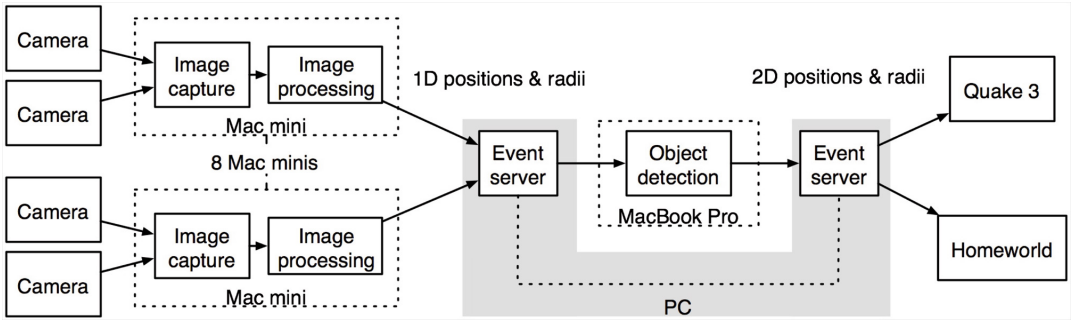


Figure 5: The architecture of the touch-free interface.

4 Implementation

Figure 5 shows the architecture of the touch-free interface. The interface makes use of 16 FireWire cameras, connected in pairs to 8 Mac minis. The cameras are mounted along the floor, enabling the detection of objects in a plane parallel to the display wall's canvas. The cameras have a 42-degree field-of-view. Images are captured at 30 FPS with a resolution of 640x480 pixels in 8-bit grayscale. Each image is processed by subtracting the background, removing noise and thresholding the result to identify objects (which are typically hands or arms). This yields zero or more pairs of 1D position and radius.

Each Mac mini sends its identified positions and radii via an event server to a MacBook Pro that determines the position of each object in 2D space using triangulation (Figure 6). The resulting 2D positions and radii are sent via the event server to either Homeworld or Q3A. The event server's role is to distribute events of different kinds to software used with the display wall. The software for capturing images, detecting and positioning objects was implemented for Mac OS X in Objective-C and C, using libdc1394[4] to communicate with the FireWire cameras; more details on the design and implementation appear in [SHBA08].

Q3A and Homeworld were modified to receive object position events from the touch-free interface, and then interpret them according to the gestures outlined in the previous section. When a gesture is recognized, events corresponding to the action associated with the gesture is injected into the game's input event stream. Depending on the relative amount of movement detected, mouse events can be generated, and the object's radius is used to determine whether it is interpreted as a flat hand or a vertical hand.

[4]http://libdc1394.sourceforge.net/

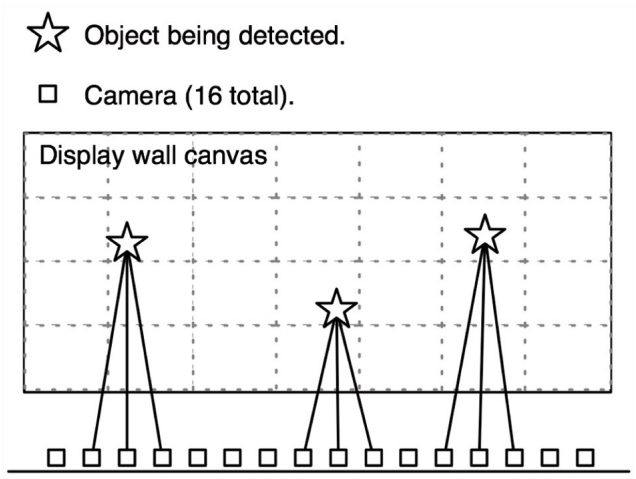


Figure 6: 16 cameras positioned below the display wall's canvas are used to triangulate the position of different objects.

4.1 Parallelizing Q3A and Homeworld

Running Q3A and Homeworld on a tiled display wall requires that each tile displays a part of the total view for each game. To achieve this, the view frustum used by OpenGL for both Q3A and Homeworld must be modified in relation to the tile on which the game runs.

The parallel version of Q3A is controlled by configuring a set of environment variables, and then reading them from within the game. The variables control how the view frustum is configured, as well as whether or not a client is designated as a player or a spectator, and which player a given spectator follows. Due to the client-server architecture of Q3A, this is sufficient to create a parallel version that will run on the display wall. Figure 7 shows a player in the upper-left corner, with four spectators following that player, as it would appear on a tiled display wall.

Homeworld was parallelized by running several

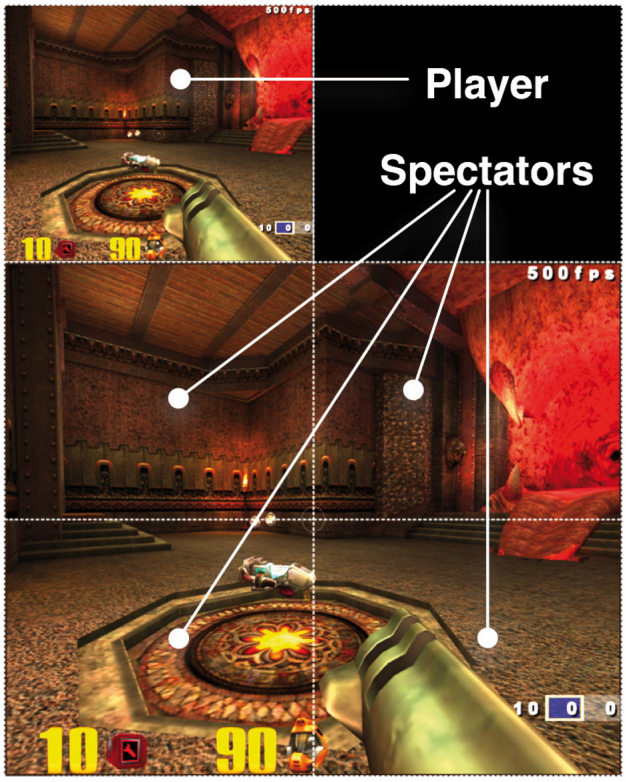


Figure 7: Example Q3A configuration on a display wall. The upper left corner shows the player, while the remaining four clients are spectators following that player, with modified view frustums to match the tiles on which they run.

tightly coupled copies and manually ensuring state consistency between them. Each copy runs on one tile, and the Message Passing Interface (MPI) [BDV94] is used to exchange state information and keep the copies synchronized. One copy is elected as master, and the remaining copies become slaves. For each frame, the master accepts input from the touch-free interface and broadcasts it to the slaves. Before starting a new frame, all the copies synchronize at a barrier. This ensures that each slave receives the same input during the same simulation step in the game, and synchronizes the visual display. To ensure that each copy's game simulation proceeds identically on all nodes, the same value is used to seed each copy's pseudo-random number generator. Finally, a global clock is shared by all the copies and controlled by the master.

5 Experiments

Three experiments were conducted. The first experiment was performed to determine the latency involved in using the touch-free interface, and determine if it is sufficiently low to play games. The next two experiments measured the rendering performance of the two games. For Q3A, the results are compared to Q3A running on the display wall using Chromium; for Homeworld, the results are compared to running Homeworld on a single display.

The hardware used was (i) a display cluster with 28 nodes (Intel Pentium 4 EM64T, 3.2 GHz, 2 GB RAM, HyperThreading enabled, NVIDIA Quadro FX 3400 with 256 MB Video RAM, running the Rocks cluster distribution 4.0) connected to 28 projectors (1024x768, arranged in a 7x4 matrix), (ii) switched, Gigabit Ethernet, (iii) 8 Mac minis (1.66 GHz Intel Core Duo, 512 MB RAM, Mac OS X 10.4.9), (iv) 16 Unibrain Fire-i FireWire cameras, (v) a MacBook Pro (2.33 GHz Intel Core 2 Duo, 3 GB RAM, Mac OS X 10.4.9). Each Mac mini was connected to two cameras. The MacBook Pro was used to run the object detection software.

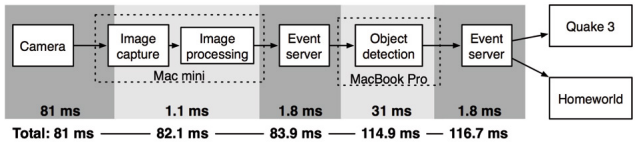
5.1 Latency Measurements

Referring to Figure 5, there are five areas where significant latency may be introduced: (1) The time taken from the camera captures an image, until the image is available to a Mac mini for processing, (2) the time taken by the Mac mini to process the image, (3) the time taken to transfer processed data over the network to the MacBook Pro, (4) time taken by the MacBook Pro to detect objects using information gathered from all the Mac minis, and (5) the time taken to distribute the resulting object positions to the two games.

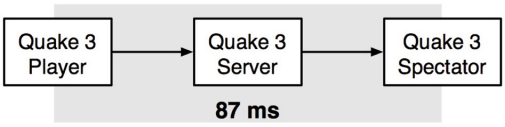
For Q3A, there is one additional, latency-inducing step. This step is the time from a gesture is recognized, until the action caused by the gesture is shown by the spectators. This latency is caused by the required round-trip from a Q3A player via Q3A's server to the spectators.

5.1.1 Methodology

The camera-induced latency (1) is measured by pointing a camera at the screen attached to a computer capturing images from the camera. The computer's screen is initially black, before it is turned white. At this point, a timer starts. The timer stops when the images



(a) The latency from when the cameras grab images, until positions of objects are available for processing by either Q3A or Homeworld. Each measurement represents an average measure of the latency.



(b) The additional latency as input events are delivered to a Q3A player, sent to the server and finally made visible by the spectators.

Figure 8: Latency measurements for (a) the touch-free interface and (b) Quake 3 Arena.

captured by the camera show a white screen, with the resulting latency being the elapsed time since the timer was started.

The processing-sensitive latencies (2 and 4) are measured by measuring typical execution times for the code that respectively performs image processing and object detection. The network latencies (3 and 5) are determined by measuring the time taken to send a message from one computer via an event server to the target, and receiving a reply.

To avoid modifying Q3A's server, the added latency in Q3A is determined as follows. When the player fires his weapon, the Q3A engine will cause a weapon-fire sound to be played. The client-side sound-playing code was modified to start a timer when that sound is played. Each spectator reports back to the player when it plays a weapon-fire sound, yielding an estimate of the latency from when something happens at the controlling player, until it is visible to the spectators.

5.1.2 Results

The results from the latency measurements are summarized in Figure 8(a). The additional latency introduced through Q3A's client-server architecture is shown in Figure 8(b). The average latency before an object's position is available to either game is 116.7 ms. The camera-induced latency is the greatest contributor, at 81 ms. Object detection requires 31 ms. For Q3A, the added latency averaged 87 ms with a standard deviation of 59 ms over 1287 samples gathered from 9 spectators.

5.2 Rendering Performance

The metric used to measure the performance of Q3A and Homeworld is frames per second. For both Q3A and Homeworld, input events are recorded over a period of about 30 seconds. The game is started in a known state, and the recorded input events are played back[5]. During playback, the framerate is logged continuously.

5.2.1 Methodology

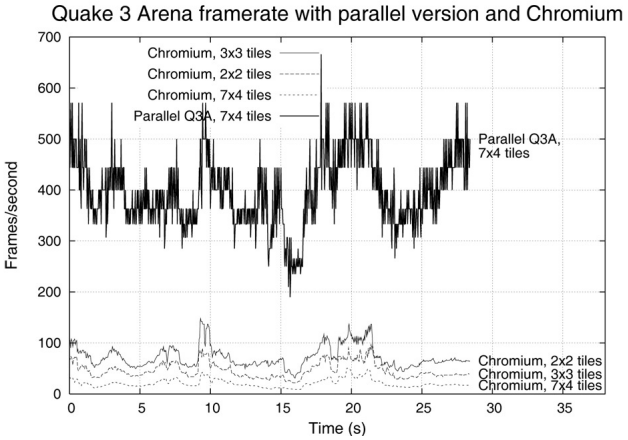
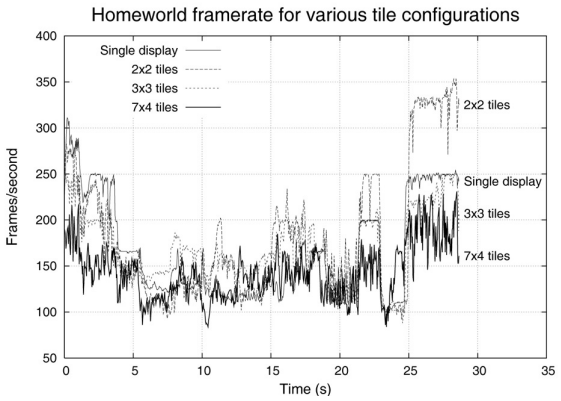


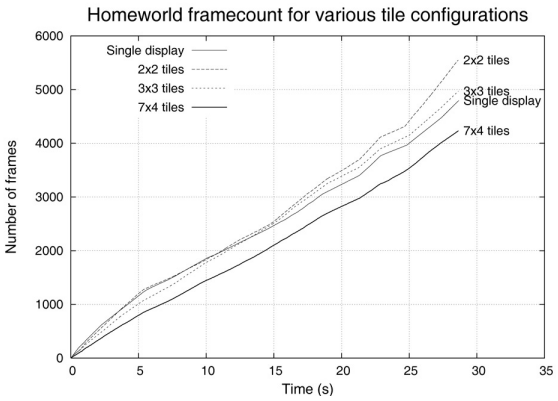
Figure 9: The framerate when running Q3A on 2x2, 3x3 and 7x4 tiles using Chromium, compared to the parallel version's framerate running on 7x4 tiles.

The performance of both Homeworld and Q3A was measured for four different configurations, with 1, 4, 9 and 28 rendering nodes. For Q3A, the framerate was limited to 500, and the performance measured both when using Chromium to distribute the rendering, and when running the parallel version. The Q3A server ran locally on the same network. For Homeworld, which did not work with Chromium, the parallel version's framerate was measured, and compared to running Homeworld on a single display.

[5]This is similar to measuring Quake performance by running a timedemo. The timedemo mechanism already in Quake does not work for the parallel version, as it is designed to run on a single computer only.



(a) The framerate when running Homeworld on a single display, compared to running it on 2x2, 3x3 and 7x4 tiles.



(b) The total number of frames drawn when running Homeworld on a single display, compared to 2x2, 3x3 and 7x4 tiles.

Figure 10: Homeworld performance measurements.

5.2.2 Results

Figure 9 shows the results from measuring Q3A's rendering performance. The peak performance with Chromium on 4 rendering nodes (2x2 tiles) is 148 FPS, and the average at 73. For 3x3 tiles, the peak FPS is 97 and the average is 47, and for all 7x4 tiles, the peak is 51 and the average 21 FPS. The figure only lists the results from the parallel version running on all 28 tiles, as there were no significant difference in performance when varying the number of rendering nodes for the parallel version. The maximum framerate for the parallel version was 666, and the average framerate was 398.

Figure 10(a) shows the results from measuring Homeworld's framerate, while Figure 10(b) shows the cumulative number of frames drawn by the game during the experiment. The framerate varies much more compared to the Q3A measurements. The maximum framerate for Homeworld running on a single tile, 2x2, 3x3 and 7x4 tiles were respectively 311, 353, 250 and 231. The respective average framerates were 168, 183, 169 and 143. Figure 10(b) shows that running Homeworld on both 2x2 and 3x3 tiles performs better than running it on a single display. The framerate was never lower than 80 for any of the configurations.

6 Discussion

Our expectations prior to implementing touch-free, multi-user support in Q3A and Homeworld were that using gestures to control Q3A would be awkward and difficult, while gestures for controlling Homeworld would be more natural as the pace of the game is slower and the gestures similar to emulating a mouse. Although we haven't conducted any formal user studies, our initial, subjective experiences indicate that the touch-free interface was more natural when controlling Q3A than controlling Homeworld. There are several potential explanations, including the characteristics of the touch-free interface and the intrinsics of the games. For instance, since Homeworld uses a one-to-one mapping between hand position and cursor position, a user might not be able to reach all points on the display wall. Another observation is that as one plays the games for extended periods of time, one's arms become fatigued.

6.1 Latency

In [MW93], the authors investigate the effect of lag (i.e, latency) on human performance in interactive systems. As latency goes up, accuracy deteriorates and time to perform tasks increases. For this reason, it is important for the touch-free interface to provide input with as low latency as possible. In [Arm03], the authors show that Q3A players prefer using Q3A servers where their average ping[6] is no more than 150-180 ms. The touch-free interface has a latency of 116.7 ms, and the average latency from the parallelized Q3A implementation is 87 ms. This gives a total latency of 203.7 ms, 23.7 ms more than the maximum preferred latency. The latency for Q3A fluctuated with a standard deviation of 59 ms, which may be an artifact of the latency measuring experiments, or a result of the Q3A server experiencing varying loads. Even though

[6]The latency from a player takes an action until it becomes observable by other players.

the average latency using the touch-free interface is slightly higher than the maximum preferred latency, the touch-free interface can be improved sufficiently to perform below the limit.

The touch-free interface's architecture is currently bound latency-wise by existing camera-technology, which are the biggest contributors to the overall system latency. As camera technology improves, the intrinsic latency of cameras can be reduced, which will directly affect the latency of the touch-free interface. Improvements in the I/O bus and OS will reduce this latency. In earlier work [SHBA07], the latency due to the cameras was found to be 102 ms. More recent experiments puts the latency at 81 ms, as shown in Section 5. We speculate that this reduction in latency is due to an operating system update, as neither the computers or cameras changed in between the experiments. The first set of experiments were conducted using Mac OS X 10.4.8, while the results presented in this paper were obtained on Mac OS X 10.4.9.

The next-biggest contributor to latency is the object detector. The detector waits for all the cameras to provide data before triangulating object positions. This synchronizes the cameras, and ensures that only fresh data from each camera is used for the triangulation. The result is improved accuracy. The cameras all run at 30 FPS, which corresponds well with the 31 ms average latency from the object detector. Improvements in camera technology will also help bring the object detector latency down. As the image capture rate of a camera goes up, the resulting latency incurred by the object detector will go down, as less waiting must be done in order to ensure that fresh data is in use from all cameras. For instance, doubling the camera framerate to 60 FPS, will result in an upper bound on the object detector latency of 16 ms. The architecture of the touch-free interface is scalable, as all image processing is done locally by each computer capturing image data. This reduces the amount of data required to be processed by the object detector by several orders of magnitude.

One problem with the touch-free interface is that its accuracy for positioning objects decreases as the objects move faster. This is caused by the use of many different cameras to capture images. Although each camera operates at the same framerate, they capture images at slightly different points in time. For a moving object, this results in the object appearing at different positions for different cameras. When these positions are used to triangulate an object's 2D position, the result can be inaccurate. These inaccuracies appear as jitter in the object's vertical position. The horizontal position is also affected, although not as much as the vertical position. This problem can be alleviated by using cameras with higher image capture rates, or cameras where the image capture can be synchronized.

6.2 Parallelizing games

Q3A's existing architecture made it possible to rapidly parallelize the game and make it run on the display wall's cluster. In particular, the spectator-concept, which can be viewed as a single data, multiple view model, was useful. This model is absent from Homeworld, making the process of parallelizing Homeworld more laborious. Applications that support this model should be simpler to parallelize for tiled display wall environments. The performance penalty from using spectators in this way is an 87 ms increase in the latency from when a player performs an action until it is visible on the display wall. This latency is independent of the input system used (keyboard/mouse or touch-free interface). Even better results may be achieved by parallelizing the game from scratch, but at the cost of a much greater effort.

Homeworld's architecture made it possible to parallelize it by running synchronized copies on the tiles. However, to determine where to synchronize, the game engine had to be analyzed to identify all places where data is used that could impact the game simulation. At these places the copies must synchronize in order to use identical data. Finding all these synchronization points is difficult, and verifying that all places have been identified requires exercising all possible code-paths of the engine. One way of doing this would be to play the entire game from start to finish; to date only the first level has been completed. Minor bugs and timing issues can also potentially skew the copies out of sync. For these reasons, parallelizing Homeworld required more effort than parallelizing Q3A.

When running Q3A on the entire display wall, the framerate for the parallel version was an order of magnitude higher than the framerate achievable using Chromium. Homeworld outperformed the sequential version when running on 2x2 and 3x3 tiles. This is somewhat unexpected, as the simulation itself was not parallelized. In principle, each copy runs the same code on the same data, with the addition of synchronization overhead for the parallel version. The fact that a higher framerate is still achieved for these tile

configurations, is because the tiles share the rendering workload. For the 7x4 configuration, the framerate is lower than for a single display. We hypothesize that this is due to increased synchronization overhead, mainly from the MPI barriers used.

7 Conclusion

This paper has introduced a touch-free, multi-user interface for controlling applications on wall-sized, high-resolution tiled displays. The interface uses 16 cameras and 9 computers to triangulate the position of objects in a plane parallel to the display wall's canvas. Input from the touch-free interface is converted to hand- and arm gestures, which are then interpreted and injected into Quake 3 Arena and Homeworld as regular mouse and keyboard events. To run on the display wall, the two games were parallelized by exploiting different aspects of the two games' architectures. For Q3A, the spectator-concept was utilized to follow each player on several tiles of the display wall. For Homeworld, a master-slave approach was taken, synchronizing all game state and input.

Players control the games by using one or both hands. Users do not need to use external devices, wear gloves or optical markers in order to interact. In this regard, the interface is not only touch-free, but also completely device-free. This enables the interface to work in a public setting where other input devices might get lost, misplaced or stolen. It also makes interaction more direct, as users no longer must interact through devices like mice or keyboards.

The responsiveness of the touch-free interface was measured by determining its end-to-end latency. The parallel versions of the two games were evaluated by measuring their framerates in both parallel and sequential (unmodified) versions running on the display wall. The touch-free interface's latency was 116.7 ms, with the majority of this latency due to the cameras used. The parallel version of Q3A consistently outperformed the sequential version running on the entire display wall, averaging 398 FPS vs sequential's 21 FPS. The average framerate for Homeworld on a single display was 168 FPS, while running Homeworld on the entire display wall yielded an average framerate of 143 FPS. The high framerates indicate that the parallelized games will scale to more tiles and higher resolutions. The framerates are well beyond what is displayable by a typical LCD panel or projector with a 60 Hz refresh rate.

Acknowledgments

The authors wish to thank Espen S. Johnsen, Tore Larsen and Ken-Arne Jensen for their discussions. Supported by the Norwegian Research Council, projects No. 159936/V30, SHARE - A Distributed Shared Virtual Desktop for Simple, Scalable and Robust Resource Sharing across Computer, Storage and Display Devices, and No. 155550/420 - Display Wall with Compute Cluster.

References

[Arm03] Grenville Armitage. *An experimental estimation of latency sensitivity in multiplayer Quake 3*. In *ICON 2003: Proceedings of the 11th IEEE International Conference on Networks*, pages 137–141. 2003. ISSN 1531-2216.

[BBH05] Steffi Beckhaus, Kristopher J. Blom, and Matthias Haringer. *A new gaming device and interaction method for a First-Person-Shooter*. In *Proceedings of the Computer Science and Magic 2005*. 2005. GC Developer Science Track.

[BDV94] Greg Burns, Raja Daoud, and James Vaigl. *LAM: An Open Cluster Environment for MPI*. In *Proceedings of Supercomputing Symposium*, pages 379–386. 1994.

[CCD06] Mark Claypool, Kajal Claypool, and Feissal Damaa. *The Effects of Frame Rate and Resolution on Users Playing First Person Shooter Games*. In *Proceedings of ACM/SPIE Multimedia Computing and Networking (MMCN)*, volume SPIE-6071, pages 1–11. jan 2006.

[DL01] Paul Dietz and Darren Leigh. *DiamondTouch: a multi-user touch technology*. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 219–226. ACM Press, New York, NY, USA, 2001. ISBN 1-58113-438-X.

[Ent08] Relic Entertainment. *Homeworld*, 2008. www.relic.com/, www.homeworldsdl.org/ and

www.thereisnospork.com/projects/, last visited: April 1st, 2008.

[Han05] Jefferson Y. Han. *Low-cost multi-touch sensing through frustrated total internal reflection*. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 115–118. ACM Press, New York, NY, USA, 2005. ISBN 1-59593-271-2.

[HHN+02] Greg Humphreys, Mike Houston, Ren Ng, Randall Frank, Sean Ahern, Peter D. Kirchner, and James T. Klosowski. *Chromium: a stream-processing framework for interactive rendering on clusters*. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 693–702. ACM Press, New York, NY, USA, 2002. ISBN 1-58113-521-1. ISSN 0730-0301.

[Int08] InterSense. *InterSense IS-900 Systems*, 2008. http://www.isense.com/www.isense.com/products.aspx?id=45, last visited April 1st, 2008.

[iS08] id Software. *Quake 3 Arena*, 2008. www.idsoftware.com/ and ioquake3.org/, last visited: April 1st, 2008.

[JH02] Jeffrey Jacobson and Zimmy Hwang. *Unreal Tournament for Immersive Interactive Theater*. *Commun. ACM*, 45(1):39–42, 2002. ISSN 0001-0782.

[KLJ04] Hyun Kang, Chang Woo Lee, and Keechul Jung. *Recognition-based gesture spotting in video games*. *Pattern Recognition Letters*, 25(15):1701–1714, 2004. ISSN 0167-8655.

[LCC+00] Kai Li, Han Chen, Yuqun Chen, Douglas W. Clark, Perry Cook, Stefanos Damianakis, Georg Essl, Adam Finkelstein, Thomas Funkhouser, Timothy Housel, Allison Klein, Zhiyan Liu, Emil Praun, Rudrajit Samanta, Ben Shedd, Jaswinder Pal Singh, George Tzanetakis, and Jiannan Zheng. *Building and Using A Scalable Display Wall System*. *IEEE Comput. Graph. Appl.*, 20(4):29–37, 2000. ISSN 0272-1716.

[Mor05] Gerald D. Morrison. *A Camera-Based Input Device for Large Interactive Displays*. *IEEE Computer Graphics and Applications*, 25(4):52–57, 2005. ISSN 0272-1716.

[MW93] I. Scott MacKenzie and Colin Ware. *Lag as a determinant of human performance in interactive systems*. In *CHI '93: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 488–493. ACM Press, New York, NY, USA, 1993. ISBN 0-89791-575-5.

[SHBA07] Daniel Stødle, Tor-Magne Stien Hagen, John Markus Bjørndalen, and Otto J. Anshus. *Gesture-Based, Touch-Free Multi-User Gaming on Wall-Sized, High-Resolution Tiled Displays*. In *Proceedings of the 4th Intl. Symposium on Pervasive Gaming Applications, PerGames 2007*, pages 75–83. June 2007.

[SHBA08] Daniel Stødle, Phuong Hoai Ha, John Markus Bjørndalen, and Otto J. Anshus. *Lessons Learned using a Camera Cluster to Detect and Locate Objects*. In *Parallel Computing: Architectures, Algorithms and Applications. Proceedings of the International Conference ParCo 2007.*, volume 15 of *Advances in Parallel Computing*, pages 71–78. IOS Press, 2008. ISBN 978-1-58603-796-3.

[SK98] Jakub Segen and Senthil Kumar. *Gesture VR: Vision-based 3D hand interace for spatial interaction*. In *MULTIMEDIA '98: Proceedings of the sixth ACM international conference on Multimedia*, pages 455–464. ACM Press, New York, NY, USA, 1998. ISBN 0-201-30990-4.

[SW06] Bram Stolk and Paul Wielinga. *Building a 100 Mpixel graphics device for the OptIPuter*. *Future Gener. Comput. Syst.*, 22(8):972–975, 2006. ISSN 0167-739X.

[SZP+00] Rajeev Sharma, Michael Zeller, Vladimir I. Pavlovic, Thomas S. Huang, Zion Lo, Stephen Chu, Yunxin Zhao,

James C. Phillips, and Klaus Schulten. *Speech/Gesture Interface to a Visual-Computing Environment*. *IEEE Computer Graphics and Applications*, 20(2):29–37, 2000. ISSN 0272-1716.

[TGSF06] Edward Tse, Saul Greenberg, Chia Shen, and Clifton Forlines. *Multimodal Multiplayer Tabletop Gaming*. In *PerGames '06: Proceedings of the 3rd International Workshop on Pervasive Gaming Applications*. 2006.

Citation

Daniel Stødle, Tor-Magne Stien Hagen, John Markus Bjørndalen, Otto J. Anshus, *Gesture-Based, Touch-Free Multi-User Gaming on Wall-Sized, High-Resolution Tiled Displays*, Journal of Virtual Reality and Broadcasting, 5(2008), no. 10, August 2008, urn:nbn:de:0009-6-15001, ISSN 1860-2037.