Possession Techniques for Interaction in Real-time Strategy Augmented Reality Games

Keith Phillips and Wayne Piekarski

Wearable Computer Lab
University of South Australia
Mawson Lakes Campus, Mawson Lakes, SA 5095
Australia, +61 8 8302 5070

keith@tinmith.net, wayne@cs.unisa.edu.au

ABSTRACT

There have been a number of interactive games created for Augmented Reality (AR) environments. In this paper, interaction techniques to support Real-Time Strategy (RTS) games in AR environments are investigated. One limiting factor is that the user's position within the virtual environment must correspond to their position in the physical world. If it is obvious to the user there is no correspondence between the two, the illusion of a consistent environment will be broken. The primary problem with adapting an RTS game for an AR environment is that the player will need to manage a large force of life-sized military units, which cannot be done effectively if the user is confined to what they can see and how fast they can move in the physical world. In this paper, we introduce the use of AR-VR transitions and a new technique that is called possession, which attempts to address these problems. Possession essentially gives the player the ability to move inside the head of any of their units. This allows the user to see everything that is visible to that unit, and manage their forces with the usual interface even though they are detached from their own body. The possession technique allows control of units over large ranges, makes micromanagement of distant groups possible, and implements realistic views of the world that match what a user would expect in the physical world. Our user interface supports a more realistic interface than is possible in traditional desktop games. Our new techniques were implemented in an operational AR-RTS game that we have named ARBattleCommander.

Categories and Subject Descriptors

H.5.1 [**Multimedia Information Systems**]: User Interfaces – *Artificial, augmented, and virtual realities.* H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *input devices and strategies, interaction styles.* K.8.0 [**Personal Computing**]: General – *games*.

General Terms

Design, Experimentation, Human Factors.

Keywords

Gaming, Augmented Reality, Interaction.

1. INTRODUCTION

Games that are played in an AR environment have a number of advantages over traditional desktop games. Players can interact with the game in an intuitive way using their body, and in previous studies, users have reported that the use of physical movement to navigate virtual environments makes the experience more enjoyable [8]. What games should exploit AR technology? First is the need for the game to be real-time, so that the problem of needing to negotiate large areas is not negated by the game pausing at the end of every turn, giving the player ample time to move or rest as they require. Most AR games created until now have been based on styles that are normally played from a first person perspective, be it on a desktop computer or in real life. This paper describes our investigations into the development of an AR Real-time Strategy (RTS) game, ARBattleCommander, for use in outdoor environments, which differ from previous work in the field.

1.1 Desktop Gaming

Most current AR games are based around first-person interactions that are essentially simulating real world situations. First-person Shooter (FPS) games, like the iD software game Quake [12], place the player in the viewpoint of the main character, and they must navigate the environment and fight enemies from this perspective. In contrast with this, RTS games such as the original Dune 2 [32] along with others such as Warcraft or Command and Conquer, give the player a disembodied god-like viewpoint that floats high above the environment. The player uses indirect commands to guide their forces in the game area. Interactions with their collection of units are performed using simple orders such as move and attack. An entire army can be seen on screen at once in many cases.

When a player starts a RTS game, only areas immediately visible to their units are visible. The rest of the world is covered by a dark fog, referred to as "fog of war". Once a friendly unit has seen the area, it is uncovered for the player to see for the duration of the game. In the context of the game, this allows players privacy while they are working within their guarded regions and enables sneak attacks from enemies. To control units, the player uses a mouse based interface. Clicking on a unit selects it, and a menu containing possible commands appears on the left of the screen. The player then selects a command for the unit(s) to execute, such as move or attack, and then selects a target in the game field. The unit selected will then carry this order out until its task is complete or it has been destroyed.

The Battlezone [2] desktop RTS game successfully mixes elements of both FPS and RTS games. A player can interact with the game as an RTS, by commanding units around and constructing a base, but can also move and shoot on foot or in a vehicle from a first-person perspective. Battlezone does not use an obvious form of occlusion like the fog of war. The player's view of the world is sufficiently limited by their perspective and features of the game to prevent them seeing things that are far away or concealed.

RTS games also require that users have an exceptional level of situational awareness. In the instance of the game Dune 2, the player must always be aware of what is happening throughout all of the area that they have a controlling interest in. Desktop games afford the player good situational awareness and precise control by using top-down or isometric views. These place the player's viewpoint far above ground level and give them the ability to rapidly move from one place to another simply by positioning a mouse cursor at the edge of the screen.

1.2 AR Versions of RTS

The player's situational awareness in an AR-RTS game is constrained by their physical environment. If the physical area the user is playing in is large and flat they are restricted to movement at ground level. Playing these games at ground level is difficult for a number of reasons. In RTS games, the player moves their units by first selecting one and then specifying a target location on the ground. Because the player's viewpoint is only approximately 2 metres from the ground, they have very little visual clarity as the ground converges into a singularity at the horizon. This makes precise selections of points on the ground extremely difficult when they are not in the immediate vicinity of the player. An AR-RTS game provides an automatic fog-of-war for the player. They have a limited field of view because of their ground-level perspective, and are unable to see any parts of the game that they were not standing near.

To overcome this limited field of view, the user must be able to move their virtual position independently of their physical position somehow, so they are capable of seeing areas other than those they are physically near to. This will require the ability to de-couple the user's views of the physical and virtual environments, so that changing their virtual position will not make any inconsistencies between the relative locations of the two environments visible. We believe that once the user is able to detach from their own AR body into a VR view independent of their physical location it will be possible to introduce a range of techniques that are otherwise inappropriate for AR systems.

1.3 Aim

The aim of this investigation is to develop a set of interaction techniques for our AR-RTS game ARBattleCommander, to overcome the user's physical limitations, without disrupting their perception of a consistent environment. We have extended action-at-a-distance interaction techniques (similar to those developed for VR) to overcome these problems in an AR environment.

1.4 Overview

This first section has outlined the research problem and identified a course of action. The next section will provide a detailed overview of some of the previous research performed in relevant areas. Section 3 discusses a way to implement VR interaction techniques in an AR-RTS game, and a new AR/VR interaction technique we have developed termed *possession* is presented. Section 4 is an overview of how we have implemented our AR-RTS game ARBattleCommander to evaluate these techniques. Section 5 provides details of the informal evaluations that were performed with the game. Finally the conclusion summarises the work described in this paper.

2. RELATED WORK

This section contains an overview of related work in the field of AR games and virtual environment interaction techniques, and is divided into the following four sub-sections: outdoor augmented reality, interaction techniques for VR environments, AR interaction techniques, and current AR games.

2.1 Outdoor Augmented Reality

Azuma [4] defines augmented reality systems as those that contain the properties: 1) combines real and virtual objects in a real environment, 2) runs interactively, and in real-time; and 3) registers (aligns) real and virtual objects with each other. Our research has focused on visual displays as the primary format, where the user wears a head-mounted display (HMD) with an orientation sensor attached. Outdoor AR systems offer some additional problems over their indoor counterparts, largely in their hardware requirements. These include power consumption and weight, but another issue faced in outdoor AR systems is how to effectively interact with the workspace available. This issue is critical when the system is potentially using a much larger workspace, like the large playing area used in ARQuake [29], where walking from one end of the game zone to the other might take several minutes, or even hours.

2.2 VR Interaction

Three fundamental operations that are typically performed in interactive virtual environments are flying, scaling, and grabbing, as described by Robinett [25]. Flying is simply translation of the user's viewpoint within the virtual environment, and in Robinett's description is movement in the direction that the tracked input device, like a glove or pushbutton device, is oriented. Scaling is an effective technique for both navigation and viewing of a virtual world. The user can scale the environment down to a convenient size to get an overview, and can then move the centre of scaling and scale the world back up to easily move to another location. Grabbing is the ability to pick up and move objects in the virtual world, and also to rotate it while it is in the user's hand. Grabbing can be performed either within arms reach, or at a distance.

There have been many contributions in the area of VR interaction. Clark developed a surface editor [10] for direct manipulation of splines using a HMD and a wand; Sachs' 3-Draw [26] performs the creation of arbitrary models using direct manipulation of a stylus and tablet; Liang's JDCAD [14] pioneered many new techniques such as lasers and spot lights for action at a distance using 3D input devices; Butterworth's 3DM [6] developed new user interfaces for immersive VR modelling; Forsberg's work with apertures [11] extended Liang's spot lights to use a circular cursor on the hand projected from the head into the scene; Pierce's image plane techniques [24] extended Forsberg's aperture projection concept to introduce a series of selection methods based on the projection of the user's hands and fingers; Mine's CHIMP [15] implemented within arms reach techniques based on proprioception and scaled world operations; Stoakley's Worlds-In-Miniature [28] demonstrated remote manipulation using small copies of the world held within the hands; Koller implemented orbital view techniques [13] where the user's head rotation maps to the position of the viewpoint on a sphere, constantly aiming toward the centre. Commercial systems such as MultiGen's SmartScene [17] implement many of the previously mentioned techniques. For techniques where the user operates on an object at a distance at a normal scale, large or close up objects will typically be easy to operate on. However, when an object is far away or small, it will be difficult to select due to its projected size.

2.3 AR Interaction

As noted by Azuma [3], because the virtual and physical worlds are aligned, to move through the virtual world requires the user to also move through the physical world. This method is used by almost every outdoor AR system to date, in the form of direct mappings from GPS coordinates to virtual world location. The user's movement is tracked via GPS, and their position in the virtual world is continually updated to reflect their position in the physical world.

The limitation of linking the physical and virtual worlds together is that it is difficult to work beyond the scale of the user. If an object is many kilometres away or very large, the user is not able to physically reach out and grab it. While previously described techniques are able to overcome this, they require the link between the physical and virtual world to be detached through scaling or flying. Previous AR systems such as Tinmith [23] support the ability to perform modelling operations at a distance using AR working planes techniques and tracked pinch gloves. However, even Tinmith relies on having an object to be selectable from the current viewpoint without any enhancements, and so restricts the possible range of operation.

2.4 AR Gaming

A number of different AR gaming systems have been created to date, the first is AR2Hockey [18]. In AR2Hockey the players wear HMDs, and the game is played with tracked physical hockey mallets on a physical table. The only aspect of the game that is simulated is the movement of the puck.

AquaGauntlet, created by the MR Systems Lab [16], is based on the older game RV-Border Guards with similar objectives. The game is played in an indoor environment using HMDs and toy guns. The aim is to destroy all of the enemies who appear, and this is achieved by using the gun in a combination of gestures.

The MIND-WARPING system [27] is an AR gaming system that allows different users to interact with the system in different ways, rather than having everybody experience the same actions from different perspectives. This game involves two players, one who wears a HMD and colour-segmented gloves, and the other who uses a workbench with an integrated display. The user wearing the HMD must then fight off the enemies using a combination of hand gestures and a kung fu yell.

Touch-Space is the part of the Game-City system [7] that is played in an indoor environment. The first part of Game-City is essentially just an AR scavenger hunt in an outdoor environment. Touch-Space is the more complex part of the gaming system. It is a truly mixed reality experience with virtual, tangible and augmented reality gaming experiences throughout several distinct and separate parts.

The first system to explore outdoor AR gaming was ARQuake [29]. This game is based upon the original source code for the id Software game Quake [12]. Users in this game don a portable computer system with HMD and toy gun, and walk around outdoors to play the game. The user's head is used to control the aim of the weapon, and the trigger of the toy gun is used to fire. Monsters were added to the environment, and the user's goal is to eliminate all the monsters in the area.

Another system that explores large-scale multiplayer games and providing different roles for players is Human Pacman [9]. This is an AR version of the vintage Pacman game, in which ghosts chase Pacman around a maze as it tries to collect pellets. Much the same approach is taken in Human Pacman, and the players are divided into two teams, Ghosts and Pacmen. Pacmen must try to collect all the cookies in the environment, and the Ghosts must try to devour all of the Pacmen before they achieve their goal.

3. GAME INTERACTION TECHNIQUES

As described previously, we have developed an AR-RTS game ARBattleCommander to investigate new user interface technologies. This game is played in an outdoor augmented reality (AR) environment, and has techniques implemented to overcome the physical limitations people encounter when using an AR system over a large area. This section describes the different problems that have been considered in the interface and also the decisions made when developing our techniques.

3.1 Real-Virtual Registration

AR applications cannot use existing VR interaction techniques to overcome problems with distant interactions because the physical world must remain aligned with the virtual. If it becomes obvious to the user that the virtual world has no relation to the physical, the illusion of a consistent AR world is broken, and the display of the physical world becomes nothing more than a backdrop to the virtual environment [31].

This problem can be neatly sidestepped by switching the user's viewpoint to an immersive VR view of the game, in a similar manner to the Magicbook application [5]. In a pure VR view, the user can no longer see the physical world and is unable to see any discrepancies between the positions of physical and virtual objects. This shields the user from any perceptions that registration between the two environments has been broken. Therefore it is now possible to apply a number of VR interaction techniques by switching to immersive VR views while they are in use. When the player is using an immersive VR view, there needs to be some differences in the way the game is drawn. When the virtual world is displayed in AR mode, only the important objects need to be drawn, as the real world is used as a landscape. If the video was blanked out in an immersive VR view, the user would have difficulty gaining a reference to where the ground is, determine distance, or find a horizontal level. Therefore, while in VR views it is necessary to fabricate at least a ground plane, and for visual appeal we have implemented a textured ground plane and surrounding cloud box.

3.2 VR Interaction Techniques

A number of VR techniques could be applied to the game with the use of AR-VR transitions, such as flying, scaled-world techniques, and external views.

3.2.1 Flying

In a desktop RTS game, if a battle is taking place at some point far away from the user's position, they need merely to make a small mouse movement to cover thousands of metres of game area in seconds. An analogous metaphor could be extended to the AR-RTS game, where the user can make use of a VR flying technique. Using this, it would be possible to move great distances in very little time, and the player could theoretically move their viewpoint to any location within the virtual environment. Such a feature would provide unlimited and quick movement to any point in the virtual environment, and because it would be just as easy to use for short distances, the player could resort to using this form of movement in preference of the usual AR interactions.

3.2.2 Scaled-World Techniques

To overcome the issues of distant interactions, one possibility would be to use scaled world techniques. This would essentially give the player the ability to view the game world in the form of a small-scale map; in much the same way is done in desktop RTS games. In this way, the player would be capable of getting an overall strategic view of the playing field, rather than being entirely confined to localised tactical decisions. As the player would be capable of seeing the entire game area at once, and could interact with less effort than in AR mode, they might rely too heavily on this method of interaction. Because this would essentially result in the game being played exactly as if it were a desktop RTS, this would completely defeat the purpose of playing it on a mobile AR-RTS system.

3.2.3 External Views

Orbital views could be implemented so that the player can specify any of their units as the centre of rotation. This would provide a good view of the area surrounding that unit, but would prevent the player from seeing the entire game world at one time, which may discourage them from relying too heavily upon this interaction technique.

Unfortunately, providing spatial input while in an orbital view presents some difficulties. Because the user's head orientation is used solely to move the viewpoint around a centre of rotation, it becomes impossible to look directly at more than one object without needing to change views again. Therefore input for the selection cursor would have to be provided through another means, such as hand tracking, limiting the player's choice of input modes.

A floating viewpoint may be implemented to provide a similar perspective to the top-down view used in desktop RTS games. The player could specify a point on the ground, and then their viewpoint would appear at a location some distance above the point specified. Normal rotational control of the viewpoint would remain so the user could look down upon the game world as though they were flying above it. This would allow the player to interact with the game as usual but using a top-down view as is provided by desktop RTS games, and would make it possible to both get a good overview of what is happening and also help with making commands over distances.

The primary disadvantage of such a technique is that it essentially is a form of flying, and suffers the same problem of providing too much freedom. This could be easily resolved by creating an artificial limitation of only allowing the player to specify points that are within a set distance of units they own.

3.2.4 Comparison of Techniques

While flying and scaled-world techniques would both provide the required functionality, both allow too great a degree of freedom of virtual movement. Players might therefore rely too heavily on such features, which would be to the overall detriment of the system and make it less realistic. The intrinsic use of physical movement is an enjoyable part of the augmented reality gaming experience [8], and if the player does not take advantage of it, there is little benefit in using an AR platform. If the player were to fall into this pattern of consistently using the VR interaction technique, there would be no point to running the game on a mobile system whatsoever.

Enabling orbital views of the virtual environment may work, but as noted it would be more complicated to provide input in such a viewing mode, and a technique which closer resembled the AR interaction method would be desirable for greater consistency.

Providing a floating viewpoint would likely be quite effective, but is avoided primarily because when in use it results in the player interacting with the game in exactly the same way as a desktop RTS. This makes the AR platform pointless if the player resorts to this technique frequently, which is likely to be the case as it would provide a better view of the environment than the player's perspective and make distant interactions somewhat easier.

Therefore it was decided that the freedom these techniques afford must be limited in some natural way, while still maintaining enough functionality to make their use worthwhile. To meet this goal we have developed another VR technique that we term *possession*, which provides a more intuitive and appropriately restrictive method of viewpoint movement.

3.3 Possession

Possession is the AR technique that we have developed to overcome the user's physical limitations while playing an AR-RTS game. Possession allows the user to select any of the game units that they control, and take control of the unit's view of the world. If the player needs to see what is happening far away, they can select any of their units that are near the point of interest to get a close-up view. In effect, it is equivalent to observing the video captured by the remote user's eyes over a wireless link, with the added ability to control the viewpoint as well.

3.3.1 Description

In essence, possession gives the player direct control of the selected unit's head, while they control the game as per usual. To use possession, the player simply positions their selection cursor over any friendly unit and executes the possession command. The user's AR view of the world then changes to a VR view of the same environment, and their viewpoint will move and rotate to match that of the selected unit.

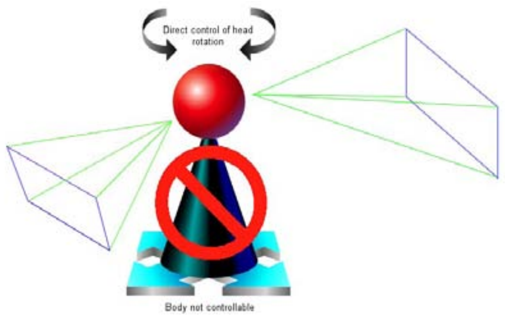
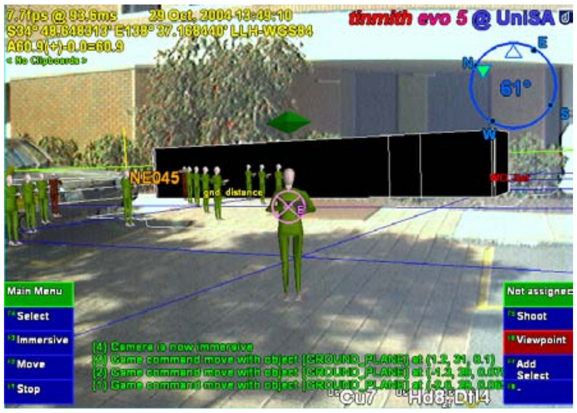


Figure 1 - User has control of the viewpoint, but not movement

Once the user is in the VR view, the user will have normal rotational control of the viewpoint and so will be able to look at the surroundings of the possessed unit, but their movement throughout the physical world will have no effect on the movement of the possessed unit, as illustrated in Figure 1. To move the possessed unit, or any other of their units, the player can perform selections and give commands while in the VR view as they do in an AR view. While possessing a unit the player can give any commands that they usually would, and this means it is possible to use a possession command on another unit and jump from one possessed unit's body to another. Because the player is only controlling the eyes of the unit, it is also possible to order the possessed unit to move to another location and the player's viewpoint will move with it.

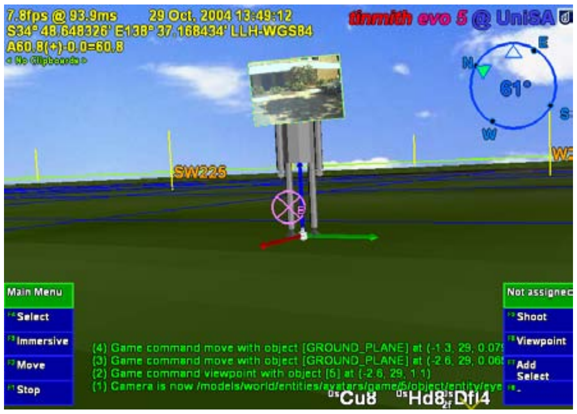
3.3.2 Benefits

Possession has several important benefits over the other VR interaction techniques that were considered. Perhaps most importantly, possession limits the freedom the player has over moving their viewpoint independently of their physical body. If a standard VR flying technique had been implemented, the player could easily move their viewpoint to any location in the virtual environment. This presents problems because the player may rely too



```
7.7fps @ 93.6ms        29 Oct, 2004 13:49:10
S34° 48.648313' E138° 37.168440' LLH-WGS84
A60.9(+)-0.0=60.9
< No Clipboards >
(4) Camera is now immersive
(3) Game command move with object [GROUND_PLANE] at (1.2, 31, 0.1)
(2) Game command move with object [GROUND_PLANE] at (-1.2, 28, 0.07
(1) Game command move with object [GROUND_PLANE] at (-2.6, 28, 0.06
```

Figure 2 - Player's view prior to a possession command



```
7.8fps @ 93.9ms        29 Oct, 2004 13:49:12
S34° 48.648326' E138° 37.168434' LLH-WGS84
A60.8(+)-0.0=60.8
< No Clipboards >
(4) Game command move with object [GROUND_PLANE] at (-1.3, 29, 0.07
(3) Game command move with object [GROUND_PLANE] at (-2.6, 29, 0.06
(2) Game command viewpoint with object [5] at (-2.6, 29, 1.1)
(1) Camera is now /models/world/entities/avatars/game/5/object/entity/eye
```

Figure 3 - Player's view immediately after a possession command

heavily on it, and also because it removes the important fog of war aspect of RTS games. As the player is limited to using the viewpoints of their units, they cannot just move their viewpoint to a convenient location and leave it there. The problems inherent in giving accurate commands over distances would make it difficult for the user to simply possess a unit and remain in that view for the entirety of the game.

A major benefit is the intuitive nature of the possession technique. If the player could freely fly their viewpoint through the virtual environment, it would be difficult to understand what they were looking through, as they would become a disembodied viewpoint with no logical ties to the virtual and physical environment. Using possession, it is easy to understand that they are now looking through the eyes of another entity in the virtual world, or that each unit has a video camera attached to its head that they can look through.

In the context of an RTS game, possession also preserves the fog of war mechanism usually employed by these games. The player's view is accurately obscured by large game units, structures or terrain, and with the natural reduction in visibility over distances, the player has a quite limited field of view. The player can simply possess any of their units to see exactly what they can see, which results in an extremely realistic and life-like implementation of fog of war.

Finally, possession provides an almost identical interface for the user when they are in a VR view. They still have the same rotational control over their viewpoint and almost all commands can be given in an identical fashion to when the user is in a normal AR view, with some small exceptions described later. This is a departure from orbital views, which would radically change the user's control of their viewpoint, or scaled-world techniques, where the user would be viewing the world as a tiny object that they hold in their hand.

3.3.3 Viewpoint Control

When the player possesses a unit, their viewpoint will change to match that of the unit. This means that if the player is looking north, and the unit is looking south, when the player possesses the unit their viewpoint will be facing south in the virtual world. All movement of the player's head is then performed relative to the existing orientation of the unit's body. The best metaphor that can be drawn is that the player takes control of the turret on a tank. They can look left, right, up and down, but this movement is all relative to the body of the possessed tank. This means if the possessed unit turns to the left, the player's viewpoint will also rotate left by the same amount. Figure 2 and Figure 3 illustrate what happens to the player's view when they select a unit to possess. Note that the player did not rotate their head between these shots, but they are looking in the opposite direction after possessing the unit (the compass in the top right remains constant while the horizon shows different heading labels).

3.3.4 Display Differences

An important point to consider is that the viewpoint of the user will be moved inside of virtual objects when they are possessed. This may present a problem because when large or complex objects are possessed the model for the object may obstruct the user's view. For this reason, objects that are possessed are not drawn for the user who possesses them.

3.3.5 Interface

The player only controls the head of a unit, rather than taking control of the unit itself. This is an important distinction, as taking direct control of the unit would not accommodate management of groups at a distance.

Therefore, aside from the ability to share a viewpoint with a friendly unit, the actual control of the game remains similar to desktop varieties of previously described RTS games. Units are selected individually, or can be added or removed from a group selection, and a cursor appears above all currently selected units to indicate this. All selected units will execute any commands then given. Commands are given by simply positioning the cursor over the desired target or point, and using the menu system to select the appropriate action for the units to perform.

The only change in the interface that does occur between normal operation and the possession mode is when the user desires to select the entity currently being possessed. Because the body of the object possessed is not visible from the normal view, the user needs some other way to select it. This is achieved simply by having the player picking the ground or the sky, which selects the currently possessed object. The reason this selection criteria is made so broad is because the player is expected to often want to control the unit they are currently possessing so they can manoeuvre it around obstacles to get a clear view of the environment.

3.3.6 Observer Objects

A problem arises with this technique when the user wishes to quickly reach an area not occupied by any friendly units. Apart from wishing to spy on the actions of an enemy, a user may also wish to move their units to precise locations some distance away, which are difficult to do at long range from ground level due to the small area of the ground displayed at a distance with a perspective projection. When using applications that do not have any existing objects in the environment that are suitable for possession, the previous techniques cannot be used in their described form.

These problems using the possession technique can be overcome using objects that we term *observer objects*. Observer objects are virtual objects created specifically to make the possession technique easier and more efficient to use. Observer objects can be thought of as free-floating cameras.

There are two ways in which these objects could be implemented; either as objects under command of the user, or as intelligent autonomous agents. Having user-directed observers has the advantage that the user can instruct them exactly where to go and what to do, but managing the observers is another task that may overwhelm an already busy user. Alternatively, intelligent agents may be extremely efficient and require no user intervention at all, but this depends upon how intelligent they actually are, and in complicated applications it may be difficult to create observers that always know where they are needed.

Exactly how observer objects are implemented will always be dependant upon the application in question. In the game described in this paper, we have implemented observer objects as a combination of both the user-directed and intelligent types described above. The observers are instantiated as helicopters that take orders from the player who owns them as a first priority. When the user has not given them a command they move close to any battles that are taking place in the game and circle them slowly, facing the battle at all times. Figure 4 shows an observer platform in the sky observing a tank in the environment. This way the player can always see where something of interest is happening from the locations of their helicopter units, and can see exactly what it is by possessing the helicopter.

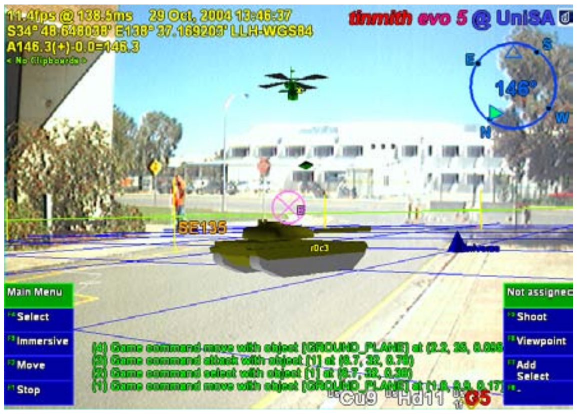
To preserve the fog of war aspect of the game, the observer is implemented as a standard game unit in this game. This prevents the player from permanently leaving it in unfriendly regions as a spy because the enemy can destroy it like any other object, costing the player a valuable and possibly irreplaceable asset.

4. AR BATTLE COMMANDER

This section provides an overview of ARBattleCommander, covering the features the game offers, the interface the players use and the platform on which it operates.

4.1 Game Overview

ARBattleCommander is structured in a similar style to a traditional desktop RTS game. The player starts with an army of military units to control, and they must destroy all of their opposition's forces. The player achieves this by giving commands to their units, such as *move* or *attack*..



```
(4) Game command move with object [GROUND_PLANE] at (2.2, 25, 0.096
(3) Game command attack with object [1] at (8.7, 32, 0.79)
(2) Game command select with object [1] at (8.7, 32, 0.38)
(1) Game command move with object [GROUND_PLANE] at (1.8, 8.9, 0.17)
```

Figure 4 - An autonomous observer object examining a point of interest to the user

4.1.1 Command Interface

To give a command, the player first selects all the units they wish to carry out the command, by aiming their cursor at them and using the *select* command from the menu. The cursor appears as a cross in the centre of the user's HMD, and is moved with the user's head or hand movement. All units that have been selected have spinning diamonds appearing above them. The player can then issue commands by aiming their cursor at the target and giving the appropriate command (move, attack or stop).

4.1.2 Unit Descriptions

There are currently five distinct types of military units implemented in the game, and these are as follows: 1) Standard Infantry is slow moving, low health, ineffective weapons, but are available in great numbers. 2) Rocket Infantry is very slow moving, very low health, but with better weapons than Standard Infantry. 3) Tank is excellent armour, high top speed, reasonable acceleration, a strong weapon, but very few available. 4) Artillery is an extremely effective weapon, reasonable top speed, average armour, but slow acceleration and relatively few available. 5) Helicopter is excellent speed, more difficult to hit than ground-based units, but has no weapons and there is only one per player, with a degree of autonomy and moves to watch conflicts as they occur.

4.1.3 Features

The most significant feature that this AR-RTS game provides over traditional RTS games is the unique game perspective. Players in traditional RTS games view the world from a top-down or isometric viewpoint, far above ground. This means that objects often cannot be drawn to scale, and it is not uncommon for large vehicles to be displayed as only slightly larger than individual people. This is particularly evident in games that incorporate large sea craft, like the popular Command and Conquer - Red Alert [33] game.

Because every object in our AR-based game is drawn from a 3D model and the player views them against an outdoor physical environment, it is natural to draw everything to real-world scale. This means that if we were to implement large sea-craft in this game, they could easily be displayed at several hundred metres in length, and at a distance would not necessarily dominate the player's view.

As previously mentioned, an important aspect of traditional RTS games is the fog of war. The AR-RTS game is played from a first-person perspective, either in the player's physical body or the body of one of their units, and so everything has a real line of sight in the virtual environment. By possessing their units, it is possible for the player to see everything that is in the line of sight of one of their units, and so this provides an extremely realistic fog of war effect. It includes not just occlusion by the environment, but also by other game objects. If something cannot be seen from the player's physical location or by any one of their units, then it cannot be viewed, just like in the physical world.

4.2 Interface

ARBattleCommander retains the omnipresent disembodied interaction metaphor provided by traditional RTS games. To this purpose, the user will still have a representation in the virtual world as a point of reference while they are in possession mode, but this avatar is removed from all direct game interactions. As the player is representing the commander of an army, the only way in which the player can interact with the world is by using the command interface to direct the actions of their game pieces.

For command input that is normally provided by a mouse button or keyboard in desktop RTS games, we have provided a hand-held push-button device. A pair of paddles that provide similar functionality as the previous Tinmith pinch gloves were created, shown in the screen image in Figure 5. Each paddle has five buttons, one under each finger or thumb, so the presentation and structure of the menus remains valid and meaningful. Straps are wrapped around the user's hands so the player can release their grasp of the paddles without dropping them. We felt the paddles would be a less intimidating way for novice users to control the user interface compared to the previously used Tinmith gloves [20].

To provide spatial input for selecting points and objects, there are two simple methods available through our existing Tinmith system. The first is to use a head cursor, which is simply a reticle drawn in the centre of the user's display that the player aims by turning their head directly towards their target. The second is to use a hand tracker, which is implemented using a fiducial marker attached to the paddles and tracked by the camera that is used for the live AR video display. Using the hand to point at a target seems like a more intuitive way to provide this input, although since the vision tracker is not always accurate or robust it is sometimes preferable to use the head cursor.

4.3 Game Server

The game server handles a single instance of a game in progress. The server is responsible for all game logic and transmitting the state of the game to each client, while the clients that connect to it handle player input and the display of data. Once a server is started, multiple clients can connect to it at any time. Upon their connection, the server allocates each client an army of units that are placed in the virtual environment, and the clients can then send commands for their units to the server.

The only information the client programs need to provide to the server is input events from the user, such as commands selected from the menu and the current cursor positions in 3D coordinates. The clients are aware of the current state of the game world via messages being constantly transmitted by the server.

4.4 The Tinmith System

The Tinmith system was developed as a software platform for developing mobile outdoor AR applications [21]. Our existing Tinmith software was previously demonstrated interacting with DIS protocol based simulation software [19]. The DIS protocol is an IEEE standard [1] and supports a wide range of complex message formats. Rather than implement our game server using DIS protocol, we used a much simpler format to make the implementation easier and allow messages that are specific to our game. The existing DIS interface in Tinmith was modified to also support our new game protocol.

Each object in the game transmits a message at a regular interval, and this is broadcast to all the client programs. These are known as status message, and one of these describes the current position and state of one object in the game. Player input is provided through messages sent via the TCP connection with the server, known as command messages. When a player gives a command, all the information required by the game is stored in a packet and sent to the server.

Command messages are designed to provide details from the interface to the game, describing any commands the player has given. They are also used by the server to transmit kill messages to the interface when an object no longer exists, and by a client program to signal when it is ready to join the game. It is important that these messages are reliably delivered so they are transmitted across a TCP channel to ensure none are lost. The number of command messages are proportional to the number of players. Status messages are designed to give all the essential details such as position and orientation of all game objects at any point in time. The server uses these messages to share with clients the current state of the game. These messages may be transmitted over UDP since lost packets will be quickly updated with newer messages. UDP is much more efficient than TCP in environments where there are large numbers of entities, and also support broadcasting over local networks.

Each mobile AR system runs the Tinmith software, which acts as a client program for the game server. Tinmith performs the display of all game-related information to the user through a video see-through HMD, processing input from the user through its menu system and sensor data, and tracking the movement of the user through the virtual environment. The game server manages the overall simulation of the game and is able to support multiple clients connected to it. Multiple mobile AR systems running Tinmith clients can connect to the game server to support multi-player games. Furthermore, we have written a 2D desktop based interface so that indoor users can also play the game against outdoor AR users.

The Tinmith client software runs on our latest Tinmith 2004 backpack computer [22]. The current implementation contains a Dell Inspiron 8100 laptop with Pentium-III 1.2Ghz and NVidia GeForce2Go, Pyro Firefly 1394 camera, IO-Glasses SVGA display, InterSense InertiaCube2, Trimble Ag132 GPS, custom Tinmith gloves or paddles, and NiMH batteries with 2 hours run time.

5. LESSONS LEARNED

This section provides an overview of the knowledge we gained from our iterative design process. Three major sets of tests were conducted. Each of the tests described in detail below was conducted as an informal study on our Tinmith backpack in a typical outdoor environment on our university campus.

5.1 Initial Testing

Before testing outdoors, a number of tests were performed indoors to verify the operation of the game.

5.1.1 Client-Server Communications

In the testing of the communication systems, an important problem was found with relation to the desired game speed. Without wanting to implement client-side interpolation methods as is done in the DIS protocol [1] which only updates every few seconds, a high frequency of status messages were to be generated for each object to make its movements appear smooth to the user. When a large number of entities were in the game, a large quantity of traffic was generated that could not be handled by the client programs. Many status messages sent were found to be redundant because most entities were sitting still or moving very slowly; status messages were adjusted to only be emitted when the state of the entity had changed. A timeout is used to ensure that objects are refreshed every few seconds so that newly connected clients will receive information about all objects in the game.

5.1.2 Information Display

A background model for the game (consisting of a hilly landscape and horizon) was initially supplied to improve visual contrast with the game objects in AR mode, and also to provide some scenery for the immersive VR views. When the game was viewed in its normal AR mode, the background model was found to block too much of the player's view of the physical world. Several simplified landscapes were tested, but the background models were eventually abandoned altogether. In AR mode, only the important game objects are drawn, and Tinmith draws a ground plane and cloud box solely for the immersive VR views. The physical environment is captured with a video camera to provide the rest of the needed detail for the game.

5.2 First Set of Outdoor Tests

This set of testing was preformed on the system with both a fully functional user interface and game server. Although the possession technique was implemented, there were no observer objects in the game, and possession relied solely upon the player's available ground based units. Some simple game play was evaluated.

5.2.1 Selection

Accurate selection of objects less than one metre in width could be performed consistently from 30 metres away. However, because of the relatively dense groups that objects were initially created in, accurate selections among objects in a group were extremely difficult. Possession was used to jump to the viewpoints of various entities, and then used to navigate through objects in a crowd in a number of steps, which worked around the selection problem.

Use of possession for this purpose proved to be disorienting for a couple of reasons however. Firstly, because the viewpoint immediately jumps from one object to the next, and objects are not drawn when they are possessed, it is sometimes confusing as to which unit is now being possessed. Secondly, the possession requires the player's viewpoint to rotate to match that of the object. We believe that using animation techniques such as those described by Thomas and Caulder [30] to interpolate between viewpoints will help the user to better understand their possession operations.

5.2.2 Movement

Movement proved to be difficult over long distances. When attempting to command a unit to move more than approximately 50 metres away, accuracy was reduced to about 5 metres. This was due to the small fraction of the ground surface visible from such a distance away caused by depth perspective. Possession was only able to resolve this issue in the instances when there was another friendly unit nearer to the target location.

5.2.3 Situational Awareness

When there were battles taking place any distance from the player, it sometimes went unnoticed. Because the game does not currently implement any audio cues, the player must rely on their sense of sight to make them aware of what is happening. In some cases even close scrutiny of the battlefield would not have helped, because large groups of units were obstructing the player's view of the conflict.

5.2.4 Input

Throughout this test, both head- and hand-based input cursors were utilised, and use of the hand tracker can be seen in Figure 5. The head cursor was generally preferred, because the hand tracker gave slightly more jittery input than the head tracker, making distant targets more difficult to select. The vision tracking of the hand-based input was adversely affected by the varying light conditions, and we are working on developing new ways of improving this tracking outdoors.

5.2.5 Test Conclusions

In order to resolve the problems of selection in a crowd, two options have been identified. The first is to introduce a smooth AR/VR transition when the player possesses an object, similar to the transition used in the Magicbook [5]. The second solution would be to enable a higher virtual viewpoint for the player in some way, thus making it possible to see over the top of many of the units in the group. The second option is preferable, because possession was only disorienting in crowded groups, due to selecting an incorrect unit. Introducing a transition in these circumstances would make it obvious which unit was selected, but would still require the player to

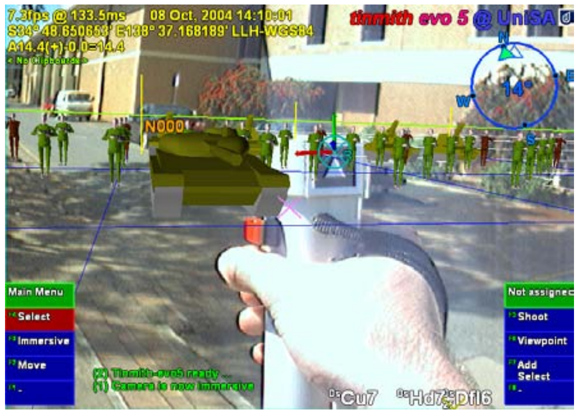

7.3fps @ 133.5ms 08 Oct, 2004 14:10:01
S34° 48.650653' E138° 37.168189' LLH-WGS84
A14.4(+)-0.0=14.4
(2) Tinmith-evo5 ready ...
(1) Camera is now immersive

Figure 5 - Using the hand-based input cursor

reorient themselves after every possession command. The transition would also add a significant delay to every viewpoint change.

To solve the problem of accuracy over distance due to ground attenuation, we provide the player with the means of virtually elevating their viewpoint significantly. The player's situational awareness is improved by allowing the player to elevate their viewpoint, and providing an overview of the state of the game. The orbital view [13] technique was investigated first. This provided a good overall view of the playing area, but made it difficult to manipulate objects since the cursors are not available. A second solution was to introduce helicopters for the player to possess. For game play reasons, it was decided that these units must be dedicated to moving and observing activities only, so that the player was not constantly occupying them by involving them in combat. By adding a level of intelligence to these, it would also be possible to further improve the player's situational awareness by having the units automatically move to look at important game events.

5.3 Second Set of Outdoor Tests

This set of tests investigated the new situational awareness techniques involving observer objects (helicopters) with a small degree of autonomy.

5.3.1 Selection and Movement

With the introduction of the helicopters, selections among groups of objects became easier. When the player's helicopter was nearby and sitting idle, it provided a very useful platform to use when making selections and a good general bird's eye view of the game area. When a helicopter was actively watching a game piece, and therefore circling around the game piece, this constant movement made accurate selections among groups difficult. Accurate selections were still possible, but it took several seconds to adjust and compensate for the constant movement of the helicopter. In the cases where selections amongst groups had to be made but the helicopter was elsewhere, it was a simple matter to order the helicopter to return so that it could be used as a viewing platform. Because of the simple intelligence that had been implemented, the helicopter would only stay in one area if the player ordered it to focus on a specific point. This presented a problem, because although it could still be utilised as a viewing platform, selections took longer because of the constant movement between different events as they happened.

Movement over distances was also far easier when a helicopter was available. Because the helicopters hover at 20 metres above ground they have a far better view of the ground below, so units could be moved accurately over great distances. In the situations where the helicopter was moving, as with selection, accuracy was still possible, but it took some time to adjust to the movement of the viewpoint. This adjustment can be seen as another challenge for players of the game to make the game more difficult yet enjoyable.

5.3.2 *Situational Awareness*

The implementation of semi-autonomous helicopters proved to be a valuable decision regarding the player's awareness. Instead of constantly looking around to see if any important events were happening, it was a simple matter to locate the position of the helicopter in the sky. If a conflict was happening, the helicopter would move over to it and observe it from close up. The constant movement of the helicopter when observing these events also helped to draw attention to it and made it easier to locate. If the helicopter is idle then the user knows there is nothing of interest occurring.

5.3.3 *AR-VR Balance*

Particular attention was paid to the length of time the user spent using AR and VR views. As noted previously, there is no reason to have an outdoor AR-RTS game that is played entirely from a VR perspective. During the testing, we found approximately 90% of the user's time was spent in the AR views.

A contributing factor was the environment the system was tested in. Extensive movement was not possible and the location was a grassy area approximately 40 metres in length near a small road, preventing the user from walking too far. Furthermore, if the user wants to walk they must switch to the AR view so they can safely navigate in the physical environment. The user walking is a major factor in ensuring the user is operating in AR mode rather than VR mode, which was an initial concern. Although the VR views when possessing a helicopter provided accurate input over long distances, it was easier to walk to units nearby when performing selections from groups.

Although possession proved to be useful for performing specific tasks that would otherwise be impossible, overall it reduced awareness of game events as they happened. This could largely be due to the fact that all game units aside from the helicopter had lower viewpoints than the user thus making other objects obstruct the user's view more. Furthermore, although the helicopter provided a good overall view of the game area, its semi-intelligent nature made it move around a lot, which made extended periods in its viewpoint slightly irritating. This could be improved by adding more intelligent logic to the helicopter.

5.3.4 *Test Conclusions*

Game play was significantly easier in this test, due to the introduction of the helicopter units. These made accurate movement over distances possible, helped when making selections from groups, and also increased situational awareness to some degree.

The intelligence present in the helicopters proved to be quite useful overall. It made it possible for the player to immediately see when a battle was taking place, and to immediately get a good view of it by possessing the helicopter. This autonomous behaviour did make the helicopter irritating to use over any extended period of time, as it would constantly move from one area to another when multiple battles were taking place. On the other hand, this provided more incentive to use the AR view for the majority of the time, and did not significantly affect the use of the helicopter for short periods of time.

A useful feature that might be implemented for helicopters is an extra command instructing them to hover above the player's head for a length of time. The player could then possess the helicopter to easily elevate their view straight up, and not have to worry about the helicopter immediately moving away to observe events elsewhere. Situational awareness still seemed to be lacking overall, as it was possible to lose sight of the helicopter and then not be aware of when battles were taking place. This could be solved easily enough through the introduction of a prompting system that gives messages when a player's units are involved in a conflict. Many desktop RTS games use sound cues to help inform the user about problems when they are otherwise concentrating on other tasks. A more effective and intuitive solution would probably require the implementation of spatial sound. That would enable the player to hear exactly what was happening at any time, and also the direction of the activity.

6. CONCLUSION

Augmented Reality provides an interesting and exciting environment for future applications. Entertainment applications are likely to play a large part in the development of this field, and consumers will undoubtedly want a diverse range of game styles. Unfortunately, due to the requirements of effective AR, games played in such an environment require that users' movement in the virtual world be tied to their movement in the physical world. This presents a problem when trying to create games that give players the freedom to escape the limitations of their own body.

A real-time strategy game is a popular genre that has not yet been implemented in outdoor AR environments, and requires an interaction metaphor fundamentally different to other AR games currently in existence. RTS games require that a single player can effectively control an entire army, and when this army is represented in full scale in the real world, it becomes obvious that the player must be able to overcome their physical limitations to achieve their goal. For these reasons, an RTS game was selected as a currently unaddressed research problem. We developed an AR-RTS game, ARBattleCommander, to be played outdoors on the mobile Tinmith AR backpack to experiment with interface techniques to overcome these limitations.

A simple way to temporarily de-couple the physical and virtual worlds was identified by using AR-VR transitions; changing the user's display of the AR environment to a purely VR view for a period of time. This enables the introduction of any of the extensively researched VR interaction techniques without making the user aware of any registration discrepancies between the physical and virtual worlds.

We also introduced a new interaction technique, *possession*. Possession provides an easily understandable means of moving the player's viewpoint, by allowing a player to view the world through the eyes of their units. While it gives them control over the viewpoint of the unit, it otherwise has the player interacting with the game in the usual way. They are still required to give commands to move their units around, and have no form of direct control over the unit they are possessing. Possession allows the player to instantly move their viewpoint to any locations where they have units, and still interact with their forces in the same way, allowing easy manipulations of large groups of units at any distance. The concept of *observer objects* was also introduced, which are objects designed to be used primarily as cameras for the user to posses and get improved external viewpoints otherwise not possible.

Informal tests of the game indicated that using the possession technique allowed for accurate input over long distances without requiring the player to move in the physical world. Furthermore, because the observer objects were implemented with a degree of autonomy and the other units were unable to get as good a view as the player in the physical world, there was no incentive to rely too heavily on the technique and thus negate the benefits of running the game on an AR platform.

7. Acknowledgements

The authors would like to thank Bruce Thomas for his help in the preparation of this manuscript. Thanks also go to all the students in the Wearable Computer Lab who have helped with this project in various ways.

8. References

[1] Institute of Electrical and Electronics Engineers. *Protocols for Distributed Interactive Simulation.* In ANSI/IEEE Standard 1278-1993, 1993.

[2] Activision. *Battlezone*. http://www.activision.com

[3] Azuma, R. *A Survey of Augmented Reality.* Presence: Teleoperators and Virtual Environments, Vol. 6, No. 4, pp 355-387, 1997.

[4] Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S., and MacIntyre, B. *Recent advances in augmented reality.* IEEE Computer Graphics and Applications, Vol. 21, No. 6, pp 34-47, 2001.

[5] Billinghurst, M., Kato, H., and Poupyrev, I. *The MagicBook - moving seamlessly between reality and virtuality.* IEEE Computer Graphics and Applications, Vol. 21, No. 3, 2001.

[6] Butterworth, J., Davidson, A., Hench, S., and Olano, T. M. 3DM: A Three Dimensional Modeler Using a Head Mounted Display. In *Symposium on Interactive 3D Graphics,* pp 135-138, Cambridge, Ma, Mar 1992.

[7] Cheok, A. D., Wan, F. S., Yang, X., Weihua, W., Huang, L. M., Billinghurst, M., and Kato, H. Game-City: a ubiquitous large area multi-interface mixed reality game space for wearable computers. In *Proceedings Sixth International Symposium on Wearable Computers, Oct 2002,* Seattle, WA, 2002.

[8] Cheok, A. D., Yang, X., Ying, Z. Z., Billinghurst, M., and Kato, H. *Touch-Space: mixed reality game space based on ubiquitous, tangible, and social computing.* Personal and Ubiquitous Computing, Vol. 6, No. 5-6, pp 430-42, 2002.

[9] Cheok, A. D., Fong, S. W., Goh, K. H., Yang, X., Liu, W., Farzbiz, F., and Li, Y. *Human Pacman: A Mobile Entertainment System with Ubiquitous Computing and Tangible Interaction over a wide outdoor area.* Personal and Ubiquitous Computing, Vol. 8, No. 2, pp 71-81, 2004.

[10] Clark, J. H. *Designing Surfaces in 3-D.* Communications of the ACM, Vol. 19, No. 8, pp 454-460, 1976.

[11] Forsberg, A., Herndon, K. P., and Zeleznik, R. Aperture Based Selection for Immersive Virtual Environments. In *9th Annual Symposium on User Interface Software and Technology,* pp 95-96, Seattle, Wa, Nov 1996.

[12] id Software. *Quake.* http://www.idsoftware.com

[13] Koller, D. R., Mine, M. R., and Hudson, S. E. *Head-Tracked Orbital Viewing: An Interaction Technique for Immersive Virtual Environments.* 1996.

[14] Liang, J. and Green, M. Geometric Modelling Using Six Degrees of Freedom Input Devices. In *3rd Int'l Conference on CAD and Computer Graphics,* Beijing, China, Aug 1993.

[15] Mine, M., Brooks, F. P., and Sequin, C. H. Moving Objects In Space: Exploiting Proprioception In Virtual-Environment Interaction. In *Int'l Conference on Computer Graphics and Interactive Techniques,* pp 19-26, Los Angeles, Ca, Aug 1997.

[16] Mixed Reality Systems Laboratory Inc. *AquaGauntlet.* http://www.mr-system.com/project/aquagauntlet/

[17] Multigen. *SmartScene.* http://www.multigen.com

[18] Ohshima, T., Satoh, K., Yamamoto, H., and Tamura, H. AR2Hockey: a case study of collaborative augmented reality. In *Proceedings. IEEE 1998 Virtual Reality Annual International Symposium, Mar 1998,* pp 268-75, Atlanta, GA, USA, 1998.

[19] Piekarski, W., Gunther, B., and Thomas, B. Integrating Virtual and Augmented Realities in an Outdoor Application. In *2nd Int'l Workshop on Augmented Reality,* San Francisco, Ca, Oct 1999.

[20] Piekarski, W. and Thomas, B. H. Tinmith-Metro: New Outdoor Techniques for Creating City Models with an Augmented Reality Wearable Computer. In *5th Int'l Symposium on Wearable Computers,* pp 31-38, Zurich, Switzerland, Oct 2001.

[21] Piekarski, W. and Thomas, B. H. An Object-Oriented Software Architecture for 3D Mixed Reality Applications. In *2nd Int'l Symposium on Mixed and Augmented Reality,* Tokyo, Japan, Oct 2003.

[22] Piekarski, W., Smith, R., and Thomas, B. H. Designing Backpacks for High Fidelity Mobile Outdoor Augmented Reality. In *3rd Int'l Symposium on Mixed and Augmented Reality,* Arlington, Va, Oct 2004.

[23] Piekarski, W. and Thomas, B. H. Augmented Reality Working Planes: A Foundation for Action and Construction at a Distance. In *3rd Int'l Symposium on Mixed and Augmented Reality,* Arlington, Va, Oct 2004.

[24] Pierce, J. S., Forsberg, A., Conway, M. J., Hong, S., Zeleznik, R., and Mine, M. R. Image Plane Interaction Techniques in 3D Immersive Environments. In *Symposium on Interactive 3D Graphics,* pp 39-43, Providence, RI, Apr 1997.

[25] Robinett, W. and Holloway, R. Implementation of flying, scaling and grabbing in virtual worlds. In *Proceedings of the 1992 symposium on Interactive 3D graphics,* pp 189-192, Cambridge, Massachusetts, United States,

[26] Sachs, E., Roberts, A., and Stoops, D. *3-Draw: A Tool For Designing 3D Shapes.* IEEE Computer Graphics and Applications, Vol. 11, No. 6, pp 18-24, 1991.

[27] Starner, T., Leibe, B., Singletary, B., and Pair, J. MIND-WARPING: towards creating a compelling collaborative augmented reality game. In *Proceedings of IUI 2000: International Conference on Intelligent User Interfaces, 9-12 Jan. 2000,* pp 256-9, New Orleans, LA, USA, 2000.

[28] Stoakley, R., Conway, M. J., and Pausch, R. Virtual Reality on a WIM: Interactive Worlds in Miniature. In *Conference on Human Factors in Computing Systems,* Denver, Co, May 1995.

[29] Thomas, B., Close, B., Donoghue, J., Squires, J., De Bondi, P., Morris, M., and Piekarski, W. ARQuake: an outdoor/indoor augmented reality first person application. In *Proceedings of Fourth International Symposium on Wearable Computers - ISWC, 16-17 Oct. 2000,* pp 139-46, Atlanta, GA, USA, 2000//.

[30] Thomas, B. H. and Caulder, P. R. *Applying Cartoon Animation Techniques To Graphics User Interfaces.* ACM Transactions on Computer-Human Interaction, Vol. 8, No. 3, 2001.

[31] Thomas, B. H. and Piekarski, W. Outdoor Virtual Reality. In *Int'l Symposium on Information and Communication Technologies,* pp 226-231, Dublin, Ireland, Sep 2003.

[32] Westwood Studios. *Dune 2.* http://www.westwoodstudios.com

[33] Westwood Studios. *Command & Conquer: Red Alert.* http://www.westwoodstudios.com