ARQuake - Modifications and Hardware for Outdoor Augmented Reality Gaming

Wayne Piekarski and Bruce H. Thomas
Wearable Computer Laboratory
School of Computer and Information Science
University of South Australia
Mawson Lakes, SA, 5095, Australia
{wayne, thomas}@cs.unisa.edu.au

Abstract

In this paper, the mobile outdoor gaming system ARQuake is discussed from an implementation point of view. The modifications to the original source code from Id Software are described, with a focus on the changes made for tracking devices, video overlays, firing weapons, and tweaks to the game to improve its visual quality. The game runs under GNU/Linux on a standard laptop mounted to a custom built backpack, containing a variety of equipment necessary to support mobile augmented reality.

1 Introduction

This paper describes some of the details behind the implementation of the ARQuake system, which was first presented in [THOM00] and then [PIEK02d]. The previously referenced papers only describe the game play, while this paper discusses the modifications which were made to transform a traditional desktop game for use on a mobile outdoor augmented reality system.

The ARQuake project is based around the Quake game engine from Id Software [IDSO01] originally released in 1996, and is a first-person perspective shoot-em up game. In Quake, the player runs around a virtual world, shooting at monsters, collecting objects, and completing objectives. Quake is desktop based, with the user interacting with it using a monitor, keyboard, and mouse. Although the game is quite old, the graphics engine is very powerful and runs on a wide range of computing hardware. With the original Quake engine being superseded by newer gaming technology, Id Software released the source code for Quake under the GNU Public License (GPL). With the availability of source code for Quake, it is possible to make extensive modifications which are not possible using the game's existing interfaces for developers. By using the Quake source code, we can leverage all the features which were available in the original game without having to rewrite our own game engine from scratch, which is very time consuming.

We took the existing Quake source code, and modified it to run on our mobile outdoor backpack computer, used for augmented reality research. This backpack contains tracking devices, a powerful laptop computer, and a head mounted display. With our modified Quake game, known as ARQuake, it is possible to walk around the real world, and play against virtual monsters drawn by the computer, using only the motion of the user's body to control the game. Figure 1 shows an example of the game being played outdoors, with virtual monsters and objects realistically appearing on the physical world's landscape.

1.1 What is augmented reality?

Augmented Reality (AR) is the process of overlaying computer-generated images over a user's view of the real world. By using a transparent head mounted display (HMD) placed on the head (such as in Figure 2), combined with a wearable computer, it is possible for the user to walk outdoors and view computer generated graphics which appear to exist in the real world. Figure 3

Figure 1 - Various monsters attacking each other and the player, with other 3D objects in background

Figure 2 - User wearing a Sony Glasstron head mounted display, with attached video camera for AR overlay

In Linux Conf Au 2003 - 4th Australian Linux Conference
January, 2003 - Perth, WA, Australia - Copyright (C) 2003 Linux Australia
Please visit http://wearables.unisa.edu.au for more information

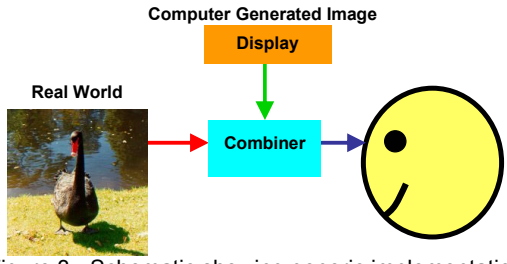

Figure 3 - Schematic showing generic implementation of an augmented reality head mounted display

shows a simplified schematic of how augmented reality is produced by combining images from the real world with computer generated graphics.

There is a wide range of research (in both technology and applications) currently being performed by the AR community, and this is discussed in two extensive survey papers by Azuma [AZUM97a], [AZUM01].

1.2 Requirements

To write applications which use mobile outdoor augmented reality technology, there are a number of hardware and software components which are required. Since the application does not run on a traditional desktop computer with mouse, keyboard, and monitor, designers must rethink the interface between the user and the application. For the ARQuake game, we require:

- A mobile computer which can be carried by the user, with enough processing power to generate the 3D graphics in real time.
- A suitable head mounted display which the user looks through to see the computer generated graphics.

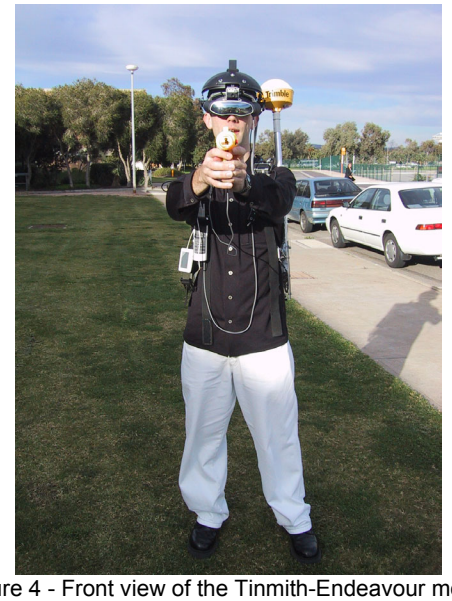
Figure 4 - Front view of the Tinmith-Endeavour mobile backpack, with gun extension for ARQuake

- Tracking devices which can measure the position and orientation of the user's head (and possibly the hands and arms as well) so the computer can accurately generate graphics to overlay the physical world.
- An input device to allow the user to control the game, shooting at monsters, changing weapons, reloading levels, etc.
- A software application which interfaces with the hardware devices, and provides augmented reality 3D overlay using optical or video methods.

2 Hardware

As part of our research, we have developed a number of backpack computers over the last 5 years which are used for mobile outdoor augmented reality. These backpacks have been designed to support research into developing new user interfaces for interacting with augmented reality computers that do not have traditional desktop input devices, with applications such as the Tinmith-Metro outdoor modelling system [PIEK01b].

While these backpacks have a number of legitimate research uses, with the addition of a small plastic gun prop, the modular backpack can be transformed into one of the world's most expensive gaming systems, and capable of taking Quake into the real world, allowing users to play against monsters that appear to really exist.

The current backpack design we use is known as Tinmith-Endeavour, and was designed in cooperation with the Defence Science Technology Organisation (see Figure 4 and Figure 5). This backpack is based on a polycarbonate plastic shell, which houses the various components that the user must carry outdoors. The polycarbonate shell has hundreds of drilled holes, which allow the rigid attachment of devices and cables to the backpack using cable ties and Velcro. In this section we will describe all the components that the user carries outside.

For a head mounted display, a Sony Glasstron PLM-700e with 800x600 SVGA resolution is used. This display is one of the highest quality ever produced, and contains a half silvered mirror to overlay images with the real world. At the time of purchase it cost A$6000 but unfortunately Sony have stopped producing this display due to lack of demand. Since the display is not replaceable and demonstrations tend to damage hardware, we demonstrate the system using other lower quality displays. For these demos, we use I-Glasses, which have low resolution LCD displays and PAL TV resolution inputs, with a cost of less than A$1000. These displays also have a half silvered mirror to overlay images with the real world. Unfortunately, there seems to be a general trend in the industry to stop making transparent HMDs, and these devices are no longer available either. Currently, the displays which are available for purchase are not transparent, and require the computer to overlay the image in software rather than using a mirror, and this will be discussed later.

For the computer to know where the user is standing, we use the Global Positioning System which is capable of providing this information almost anywhere in the world. To get better accuracy than is possible using standard GPS receivers (5 metres), we use a Trimble Ag132 GPS, which uses differential correction signals as well as advanced signal processing to improve accuracy to 50 centimetres. This unit costs around A$8000, and even further improved accuracy can be gained using Real Time Kinematic GPS units (1-2 centimetres), although the costs for these range from about A$50,000 upwards.

In order for the computer to render the correct part of the world for the user, information about the orientation of the user's head is also required. We use an Intersense IS-300, which is a hybrid sensor combining magnetometers (magnetic compass), accelerometers (tilt sensors), and solid state gyroscopes to produce fast updates which are much more accurate than a standard sensor without gyroscopes. This device costs around A$4000 to purchase, but unfortunately its accuracy suffers when exposed to magnetic distortion caused by external objects as well as the backpack itself. The best tracking is possible using a fibre optic gyroscope, which fires lasers around coils of fibre optic cable and measures the phase difference caused by the motion of the user. While incredibly accurate with drift only occurring after many hours and immune to outside interference, this device costs around A$100,000 and is currently too expensive for our budget.

For video input, we use a special 1394 Firewire camera known as a Firefly. This camera is capable of generating 640x480 video at 15 fps in raw RGB format to the host machine, ready for processing or display without any decompression required in software. The main advantage of this camera is that it is incredibly small, so it can be easily mounted onto a HMD without causing weight or size problems. This camera costs about A$1000, and we have also used other cheaper cameras which have slightly lower image quality and have a larger form factor.

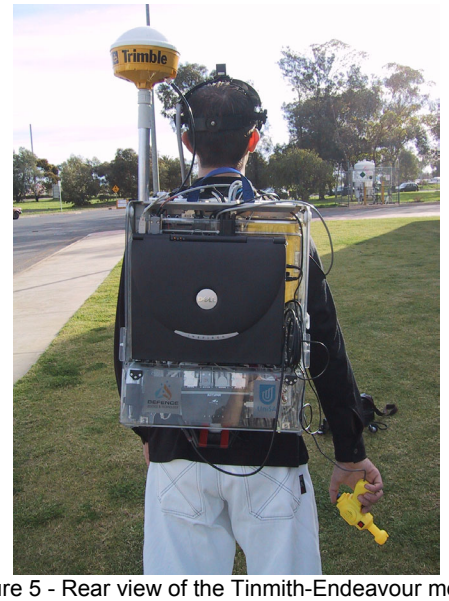

Figure 5 - Rear view of the Tinmith-Endeavour mobile backpack, with gun extension for ARQuake

For a processor, the backpack currently uses a Dell Inspiron 8100 laptop, which has a Pentium-III processor at 1.2 GHz, and an NVidia GeForce 2 graphics chipset. This laptop runs the GNU/Linux operating system (currently RedHat 7.3 with kernel 2.4.17), and uses the NVidia OpenGL 3D drivers to perform direct hardware rendering. The graphics chipset is also the first which is powerful enough to take live video streams, load them into texture memory, and display the video as a texture on arbitrary polygons in real time.

The backpack also carries a number of other devices for interfacing all the hardware, such as multiple USB hubs, a 1394 Firewire hub, and a Keyspan RS-232 to USB converter. The system operates for 2 hours with a 12V battery rated at 85 Wh, and weighs approximately 16 kg. We implemented Tinmith-Endeavour with as many off-the-shelf components as possible, and no effort has been made to miniaturise or lighten the design through custom built components

3 Software modifications

This section discusses the modifications which were made to the original Id Software version of Quake, to produce the ARQuake game discussed in this paper.

3.1 Tracking

The Quake game (as well as most other desktop based games) is controlled using the keyboard and mouse, with a monitor used for the display. The user specifies where they would like the game to move by pressing the arrow keys or steering the user with the mouse. This operation is relative because the movement is relative to the previous position and orientation of the user. Internally however, the game stores 3D position (X, Y, Z) and orientation (heading, pitch, roll) in variables, and these are stored as absolute values relative to the origin of the game universe. When the keyboard or mouse is used to control the game, relative offsets are applied to these values. By taking control of the game and modifying these values ourselves, it is possible to use the outputs from the GPS and IS-300 to control the game movements instead, making the game fully controlled by the body.

To perform this, software must be written to parse the RS-232 output of these devices, and deliver them to the Quake game engine. For this task, we use the Tinmith-evo5 software architecture [PIEK02a] which contains drivers for all the hardware on the backpack, and abstracts them to generic object types. These updates are then serialised to a string and sent by the driver process to the Quake game as UDP packets. It should be noted that there are a number of freely available software systems which provide drivers for many 3D devices,

Figure 6 - Haptic feedback gun, embedded into futuristic looking children's laser gun toy (controller not shown)

Figure 7 - USB mouse embedded into a children's bubble blowing toy, with switch and button attachments

some popular ones are VRPN [VRPN02], OpenTracker [OTRK02], and VR Juggler [VRJG02].

Inside the Quake game, as part of the display refresh loop, the incoming UDP socket is checked for update packets, which then deserialises the contents. These values are then copied into Quake's global position and orientation values, and when the 3D scene is rendered it will be using the values from the hardware devices.

With these changes to the processing loop, it is still possible to control the game using the keyboard or mouse, as this code is not modified. However, any movements made with these inputs will be overwritten when the next UDP packet arrives if the driver process is running. If hardware changes are made to the backpack, then only the driver process needs to be changed as the game modifications are implemented generically.

3.2 Actions

While the rendering of the game is controlled using the position and orientation of the user's head, to interact with the game we have developed a number of plastic guns with buttons linked up to the computer. When the user presses the trigger, the gun in the game shoots, making this a very intuitive input device to use.

Rather than making moulds and trying to create our own plastic guns from scratch, it is much cheaper and easier to go to a toy store and purchase them already pre-fabricated. Many of these toys contain internals that can easily be removed, and replaced with electronics to perform whatever task is required.

The first gun design (built in 2000) is shown in Figure 6, and contains a solenoid which electrically rams a bolt against the gun to give the user haptic feedback whenever it is fired. A controller box is used to process the trigger and two other buttons, and also to apply the correct voltage to the solenoid to simulate the different weapons available in Quake. The controller is relatively simple and uses a parallel port interface, with the haptic feedback controlled by the computer. In practice, we found that the power consumption of the gun was excessive, as the solenoid uses large amounts of current. Also, the parallel port interface requires special privileges to run, and is timing critical since it uses bit banging to communicate to the controller. A more efficient interface would be to use a microcontroller in the gun and RS-232 or USB. When in use outdoors, we also found that the gun looked too realistic, and could potentially cause problems with bystanders and security guards unaware of the technology.

To simplify things as much as possible, we have recently developed a much smaller and easier to use gun, show in Figure 7. This gun is a small toy originally designed to blow bubbles, and is painted in a friendly bright yellow colour, looking more like a hair dryer than a weapon. We commonly refer to this device as the 'duck gun', and it causes fewer problems with security when shown in the public. After having its internals removed, we embedded the electronics from a broken USB mouse, wiring each of its three buttons to a switch on the trigger, and two others on the side. It plugs into the computer with a standard USB connector, and requires no software modifications to use since it appears as a standard mouse to Quake. Although it has no haptic feedback like the other gun, this design interfaces directly with most software, and has negligible current draw and no external controller box.

It should be noted that neither gun design contains a tracking device, and is simply a collection of buttons embedded into a plastic object. The firing aim of the user in the game cannot be controlled by moving the gun, and is controlled using the motion of the user's head, as is done in the desktop game. Quake always fires weapons to the crosshair in the centre of the display, and would require major changes to the game to allow an individually steerable cursor, as well as adding hardware to track the location of the gun.

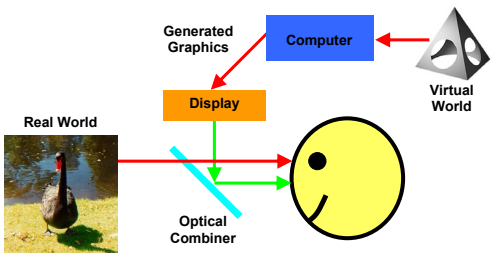

Figure 8 - Schematic of optical based augmented reality

3.3 Optical overlay

In most cases, the easiest way to perform augmented reality is to use a technique known as AR optical overlay. This technique was developed by Sutherland [SUTH68] in 1968, and was used to produce a very primitive augmented reality demonstration.

This display, along with more modern equivalents, is implemented using an optical combiner, as shown in the schematic in Figure 8. Computer generated images are sent to small displays worn by the user, which are then focused and aimed using lenses and mirrors, and combine the images with the real world using a half-silvered mirror.

The images in Figure 9 and Figure 10 show examples where the ARQuake game overlays the real world using optical combining. To make the images combine properly, the computer generates the image with black pixels where it wants the user to see the real world, and colours in pixels where it wants to show an object. For the example figures, the levels were modified so that where we want the user to see the real world, we use black textures.

This technique is simple to implement because the computer simply draws black when it wants to let in real world light, and requires no software modifications to operate, except for changes to the game levels. However, the optical combiner darkens the entire world like a pair of sunglasses, and overlaid images look ghosted because the real and virtual worlds are always both visible. Also, taking pictures of the game in action with video or still cameras is difficult through a HMD, and it is not possible for bystanders to see what the user with the HMD is experiencing.

This technique was used in the original ARQuake system in 2000, as mobile computers were only fast enough to run Quake with low resolutions, and hardware 3D acceleration was not available. The next technique, known as video overlay, overcomes many of the problems with this technique, but requires much more powerful processing to implement properly.

3.4 Video AR

Instead of using an optical combiner, which produces images that are ghosted and difficult to show to others, a technique known as video overlay is possible. This method uses a video camera to view the world, and the frames from the camera are combined with the rendering of the 3D graphics inside the computer, and the final display is shown to the user, as outlined in Figure 11. For this method, the HMD is completely opaque, and the user cannot see through it except with the video camera. Now that recent mobile computers are powerful enough to perform video overlay, and also that transparent HMDs are harder to find, we have switched to this technique for most of our research.

This technique produces excellent quality output limited only by the camera and display, and is easy to

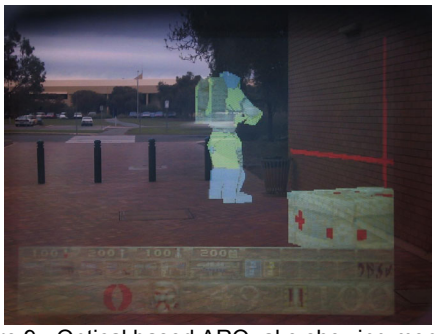

Figure 9 - Optical based ARQuake showing monster, health box, and partial grid lines on building walls

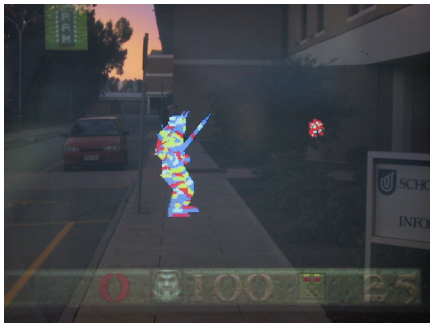

Figure 10 - Optical based ARQuake showing monster standing on walkway near building

demonstrate in large groups because the complete AR output can be seen on the laptop screen on the backpack. Game sessions can be recorded to tape with a video recorder, without having to clumsily hold a camera inside a HMD for filming.

The implementation of this technique is more involved than the optical case, since the computer now needs to perform the overlay in software. In the Tinmith-Metro application [PIEK01b] we implemented a texture mapped polygon with the incoming video stream mapped to it in real time. This polygon is attached to the user's head motion so that no matter what direction they look in, it is always visible. Figure 12 shows how this polygon is linked up to the avatar of the user. When in immersive augmented reality mode, the display appears as in Figure 13, showing the system overlaying 3D cursors over the hands in the real world. The only problem with this set up is that if the polygon is placed 2 metres away from the user's head, any objects which are

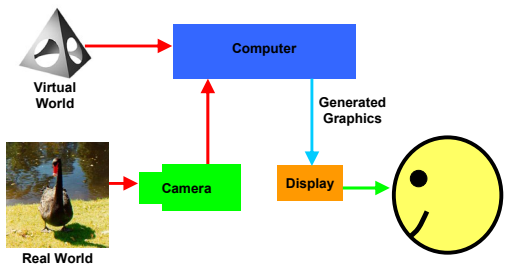

Figure 11 - Schematic of video based augmented reality

further away will be occluded by the video polygon. To prevent this, we scale all the dimensions for the video texture by an arbitrarily large number, say 10000, and so the polygon which was once 1m x 1m at a distance of 2m is now 10km x 10km and 20km away! By performing this scaling operation, any objects within 20 km of the user will not be occluded by the video display, which is adequate for the task. This projection technique can be thought of as a large drive-in theatre screen or IMAX theatre attached to your head and displaying an image at all times.

Performing this same rendering technique in Quake is not as simple however. Quake is highly optimised for speed, and so does not clear the background on the assumption that every pixel will be redrawn because all levels must fully enclose the user with no external holes showing the empty void surrounding the game. Having a projection screen at a large distance does not work because it will be outside the current room the user is in, and therefore not visible (Tinmith-Metro allows the user to see empty space, and so the video projection works).

In the original optical ARQuake, textures containing black pixels were used where we wanted the user to see the real world. With a video overlay system, black pixels will overwrite any video projection, and so some extra effort is required for the overlay. OpenGL has a number of functions for drawing images and textures using Boolean operators and special functions, which would allow the replacement of all pixels of a certain colour with the video overlay. However, these functions are emulated in software with most drivers as they are not supported in hardware and very rarely used. The trick is to use a technique which is supported in hardware to gain maximum speed and avoid stalling the 3D pipeline.

After much experimenting, OpenGL stencil buffers were the only way we found to implement video overlay using hardware acceleration. An interesting thing to note is that the TNT2 and GeForce2 implement stencil buffers differently using hardware at some colour bit depths, and

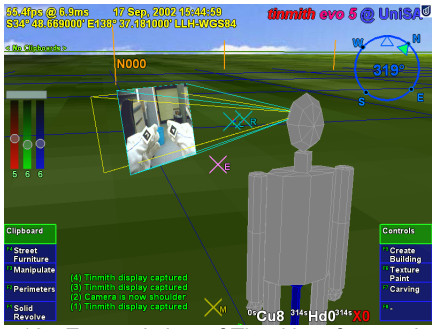

Figure 12 - External view of Tinmith software showing relationship between user's head, projection video, and 3D objects (video display is normally further away)

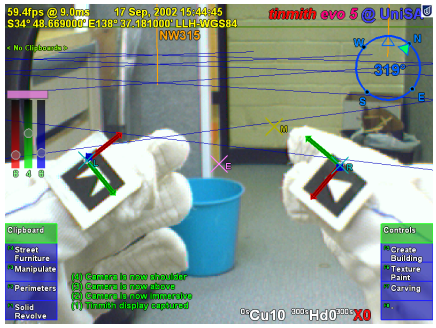

Figure 13 - Immersive Tinmith screen shot showing 3D cursor objects appearing to be floating over video image


Figure 14 - Monster falls over when shot at by the player


Figure 15 - Various monsters attacking after being released by the opening of a door


Figure 16 - Monsters shoot at each other while player attacks, yellow gun is seen at the bottom of the display


Figure 17 - Close up view showing the outcome of a virtual battle, with monsters overlaid onto the ground

using emulation at other depths, so it is important to test this carefully. We initialise the stencil buffer to all zeros at the start of each frame, and during normal draw operations we write zeros. In code that draws texture mapped polygons, we put in a small test which checks for a texture named 'black'. When this texture is used, we set the stencil draw value to one and so any pixels drawn to the display for this triangle (that pass the normal Z-buffer tests) will set the stencil buffer to one at these points. After the completion of the Quake render operation, a polygon using the video texture and filling the entire display is drawn, but with an extra test function that only allows drawing where the stencil buffer is set to one. The result is that all pixels drawn with the 'black' texture are replaced with video, but we don't replace actual black coloured pixels contained in other textures, so it is still possible to have objects which contain black pixels. This technique is used to render the outdoor action images in Figure 14, Figure 15, Figure 16 and Figure 17, and the quality of these images is excellent with vivid colours and sharp edges.

3.5 Miscellaneous

While the previous sections covered the major changes which were required to make ARQuake work in an outdoor environment, there are a number of other tweaks which were made to improve game play and the quality of the output for the user. Some of these tweaks were made by modifying configuration files, and others were made by directly editing the source code.

With the use of head mounted displays, it is important that Quake is configured to use the same field of view (FOV) as the display being used, otherwise the objects will not overlay correctly. Normally, Quake runs with a FOV of around 70-90 degrees, whereas a typical HMD is 20-30 degrees. At this FOV however, Quake does not render the gun properly and the user cannot tell what gun they are using. To correct this, the code which places the gun relative to the user was modified to use variables, and it can be dynamically repositioned as the FOV varies.

Since Quake is meant to be played on a desktop, the user needs feedback so they know when they are being hit, and large weapons cause recoil. When playing ARQuake, it does not make sense for the game to recoil or move because it is under the control of the tracking devices. This movement is distracting to the user because the entire game jolts before being corrected by the 3D trackers, but the video remains motionless, and is confusing since the overlay illusion is broken. The code which implements this feature has been disabled to keep the game stable and under the full control of the trackers.

Quake is normally a very dark and gloomy game, and many levels use this to increase the sense of presence, but this makes the level very hard to see on a HMD, especially when working outdoors under bright sunlight. A number of settings in Quake which control the lighting have been modified so they are at full brightness, producing the most saturated displays possible and reducing the dark and gloomy effect somewhat. Effects such as lighting generated by the motion of the user, rockets, and explosions, as well as colouring changes to indicate invincibility and bio-suit mode have been disabled because they are incompatible with the rendering technique used for the video overlay.

With the use of the plastic gun as the input device, many users initially incorrectly thought that by aiming this gun it would control where the bullets would fly. To help convey this better, a large crosshair was placed in the centre of the display making it more obvious that this is where the user is aiming at.

To help with the debugging of the game, and to monitor its operation when in use outdoors (considering that the hardware can often cause problems) we also add extra position and orientation information to the status bar at the bottom, which is available at all times instead of having to use the Quake console.

When demonstrating the game to people not familiar with the system it takes a short amount of time to get used to and so for these cases, we modify the start up of the levels to give the user lots of weapons and invincibility. This makes it easier for the user to start playing straight away and not worry too much about being hit by the monsters and trying to find items. When the accuracy of the GPS tracker is poor (due to bad satellite coverage) it can sometimes be difficult to walk around accurately and pick up objects, and so this makes the game easier to play straight away.

One interesting side effect of the modifications to the Quake source code is that multiplayer support is included automatically. Using wireless network adaptors, it is possible to have people outdoors playing against other mobile computers, as well as users with the standard desktop Quake indoors. In most scenarios though, the user on the desktop machine will usually win because a mouse and keyboard allows much faster running and turning than is possible in the physical world, especially when carrying heavy and fragile computing equipment.

4 Level and character design

Since ARQuake is based on the same source code as Quake, we use the standard Quake mapping tools such as WorldCraft to design and edit levels. Any existing maps for Quake can be used straight away, but in most cases should be modified to take advantage of the AR overlay and to make it as playable as possible.

For objects that are required to be transparent, so that the video overlay will work, the textures must be set to the internal name 'black' so that the code modifications discussed earlier can do the stencil buffer overlay. To walk around in what appears to be an open environment, a large room should be created using the 'black' texture for all walls, floor, and roof. While it is possible to leave the level fully textured without video overlay, this is not really augmented reality because the real world is not

visible - it is more of a mobile virtual reality system instead.

Objects should be as brightly lit as possible, because when played outside everything else is bright, and so it does not make sense to have dark objects and shadows during the middle of the day. When using a HMD, colours tend to wash out when sunlight gets in on the sides of the display, and so bright colours which are saturated produce the easiest to see output. The characters used in Quake are also quite dark, and so the skins for these were recoloured to make them as visible as possible, an example conversion is shown in Figure 18 and Figure 19.

When designing levels, it is important to select monsters that are not too powerful for the user to fight against. Since the user must physically move their body to play the game, it is not possible to dodge bullets and perform other rapid operations that would normally be expected. For most of our levels, we select monsters that are reasonably slow and not too powerful so the user has a chance to play the game without being overwhelmed.

For cases when there is no GPS tracking (such as when demonstrating indoors) the game still works, but the user cannot move from the starting position. For these cases, we have developed special levels which have all the monsters accessible from the start position, and weapons are given to the user so they do not have to walk around and pick them up. Having these special levels is important also for debugging as all development is done indoors, and only when the software is fully tested is it taken outside to be trialled.

5 Limitations

Quake is a game which was optimised to run on desktop platforms, and to use certain tricks to improve the quality of the display and the frame rate by sacrificing certain features. These optimisations limit the changes that can be made when modifying the source code (without a major rewrite) and make some changes more difficult than others (for example, the video overlay). While Quake was never designed to be used for outdoor augmented reality, the fact that we have been able to change a game written so many years ago quite easily is a tribute to its excellent design.

Currently we have reached the limits of what the Quake engine is capable of doing for us, as other features we desire are not possible to implement without major changes:

- As mentioned previously, the gun cannot be aimed independently since Quake assumes that the user steers the gun and the head with the same input device.
- The video overlay has some glitches due to the use of the stencil buffer, and there are cases where objects are not occluded correctly although these are rare and hard to spot.

Figure 18 - Original Quake character with dark and gloomy textures

Figure 19 - Quake character using modified colour maps to improve visibility

- The camera model of Quake is confusing, with it being tricky to measure the scaling factors used when rendering maps to ensure that the display accurately matches the real world - we have managed to get everything to register properly but it was a trial and error approach. Quake is designed to render approximate models that look good for the user, but not necessarily accurate.
- Actions such as laying down or looking around corners (without exposing the body) are not possible, because Quake assumes a rigid body which is always standing upright.
- Quake uses an optimised model format which is different than the traditional VRML, 3DS, and DXF files used for modelling. Converting to and from Quake's format is difficult, and requires a human to manually intervene at certain stages.

It should be realised that most first person perspective shooter games have the same limitations as Quake, and implement features which are similar since they are designed to be played on desktop machines. Although they may have rendering engines which are many years newer and more up to date, most of these newer features are useless for an AR conversion because one of the main changes we make is to disable lighting and shading effects to improve the visibility of the display. Another disadvantage to newer games is that the AI for the characters tends to be a lot smarter than previously, which means that the user (who is already at a disadvantage by being limited to physical movement constraints) will have an even harder time beating computer generated monsters that are already difficult. Most importantly of all, the source code must be available in order to make the changes necessary, as they require access to the internals of the renderer which is not possible using standard extension mechanisms.

Rather than trying to extend existing games to support AR, we are now exploring modifications to our current AR software Tinmith-Metro [PIEK01b] to support playing games. This application currently supports a wide range of modelling tasks, and is capable of handling the rendering of entities in a number of file

formats very easily. By integrating in the plastic gun it will be possible to support Quake-like functionality but with extra features not previously possible.

Finally, it should be realised that some concepts from Quake will never be possible when playing ARQuake, because these actions are impossible in the real world. Actions such as jumping large distances (which would break the backpack), flying, and swimming, are not possible without extra extensions to the backpack such as a jet pack and breathing apparatus.

6 Conclusion

The ARQuake project has shown how it is possible to take an existing open source game Quake, and modify it from the desktop paradigm to mobile outdoor augmented reality. The modifications are relatively simple, and the same ideas could be used to modify many games of similar design if source code is available. We hope that by demonstrating the ease with which changes can be made, that game developers will include hooks into their software to support 3D trackers, HMDs, and video overlays from the start, making it possible to play more games on exotic equipment and open up exciting new possibilities for computer entertainment.

7 Acknowledgements

The ARQuake project was originally conceived by Dr. Bruce Thomas of the Wearable Computer Lab at the University of South Australia, and with Wayne Piekarski (see Figure 20) led a student project group (Ben Close, John Donoghue, John Squires, Phil DeBondi, and Michael Morris) to integrate the 3D tracker controls into Quake, develop the haptic feedback gun, and create new levels and recoloured monsters. Wayne Piekarski now maintains the ARQuake project in sync with the existing Tinmith hardware and software systems for demonstrations, and added the texture overlay feature and other playability tweaks. The latest ARQuake demonstration levels were designed by Arron Piekarski. Finally, this project would not have been possible without Id Software donating the Quake source code to the public under the GNU Public License.

For more information about the project, as well as copies of our papers, pictures, and videos, please visit our web site at http://wearables.unisa.edu.au

Figure 20 - Wayne Piekarski and Dr Bruce Thomas

8 References

[AZUM01] 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, Nov 2001.

[AZUM97a] Azuma, R. *A Survey of Augmented Reality.* Presence: Teleoperators and Virtual Environments, Vol. 6, No. 4, pp 355-385, 1997.

[IDSO01] Id Software. *Quake.* http://www.idsoftware.com

[OTRK02] Studierstube Augmented Reality Project. *OpenTracker.* http://www.studierstube.org/opentracker

[PIEK01b] Piekarski, W. and Thomas, B. 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.

[PIEK02a] Piekarski, W. and Thomas, B. H. *Tinmith-evo5 - A Software Architecture for Supporting Research Into Outdoor Augmented Reality Environments.* Technical Report, University of South Australia, Adelaide, SA, Report No. CIS-02-001,Jan 2002, http://www.tinmith.net/papers/piekarski-tr-arch-2002.pdf.

[PIEK02d] Piekarski, W. and Thomas, B. H. *ARQuake: The Outdoor Augmented Reality Gaming System.* ACM Communications, Vol. 45, No. 1, pp 36-38, 2002.

[SUTH68] Sutherland, I. A Head-Mounted Three-Dimensional Display. In *Proceedings Fall Joint Computer Conference,* pp 757-764, Washington, DC, 1968.

[THOM00] 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 *4th Int'l Symposium on Wearable Computers,* pp 139-146, Atlanta, Ga, Oct 2000.

[VRJG02] Virtual Reality Applications Centre. *VR Juggler - Open Source Virtual Reality Tools.* http://www.vrjuggler.org

[VRPN02] University of North Carolina. *Virtual Reality Peripheral Network.* http://www.cs.unc.edu/Research/vrpn