Virtual Window: A Peephole into another World

Zaheer Hamza Gary Pnematicatos Nicholas Tip Patrick Marais Gary Marsden

Technical Report CS04-19-00
Department of Computer Science
University of Cape Town

ABSTRACT

This project investigates an effective means of streaming and displaying complex 3D scenes over a low-bandwidth wireless network to a remote PDA device, which has limited graphical capabilities. The PDA is also made position-aware, allowing user interaction to occur with a new degree of freedom, mapping real world movements such as rotation to virtual environment changes.

Rendering techniques such as silhouetting optimise PDA display rate, but slow PDA processing speed means image compression to reduce network latency is unfeasible. In spite of any technical limitations, however, preliminary user testing shows that the concept and methods of interaction developed are intuitive, with good potential for further system development.

Categories and Subject Descriptors

H.5.2 [**INFORMATION INTERFACES AND PRESENTATION**]: User Interfaces - *Evaluation, GUI, Input devices and strategies.*

General Terms

Performance, Design, Experimentation, Human Factors.

Keywords

Compression, Mobile, Rendering, Interaction.

1. INTRODUCTION

The integration of mobile devices with wireless connectivity provides an exciting opportunity for developing new interaction systems. With rapidly expanding technologies, such as IEEE 802.11, connecting hand-held computers and conventional computers together will no longer be an occasional event. Instead, the devices will frequently be in close, interactive communication.

Another important area of technology research and development is that of the development of virtual environments. These allow one to create any 3D environment, even those that are not physically realisable, to model interactions and assist in research at low cost.

We attempt to merge these two concepts into a single model. This model will allow users to interact with a small viewing window, via a PDA, that is displaying some kind of complex virtual environment, which the user can manipulate in real time as they move around in physical space.
Implementing this system required three discrete modules to be developed and interoperated:

- A graphics engine to generate and render the 3D world environment on the server and PDA.
- A compression and networking sub-system to provide a wireless transport medium to transmit images economically from the renderer to the device.
- A user interface and movement-tracking technique to allow users to interact with the device in a sensible manner.

2. BACKGROUND

2.1 Compression

A common characteristic of most images is that the neighbouring pixels are correlated and therefore contain redundant information. The foremost task then is to find less correlated representation of the image. Two fundamental components of compression are redundancy and irrelevancy reduction. Redundancy reduction aims at removing duplication from the image. Irrelevancy reduction omits parts of the image that will not be noticed by the viewer.

Images can be represented by a number of different colour spaces other than RGB, such as YIQ and YCC. These colour spaces represent the image data in another format, which separates the luminosity or brightness from the hue. The Human Visual System has a low dynamic range or acuity for spatial variation in colour than it does for luminance. In other words, we are more acutely aware of changes in the brightness of details than of small changes in hue. So rather than manipulating the image in RGB colour, it can be more efficient to encode the luminance in one channel and the chrominance in two other channels. This way, we can manipulate the image to reduce its size, while still maintaining image quality according the HVS.

JPEG is the image compression standard developed by the Joint Photographic Experts Group. JPEG is a lossy compression algorithm that has been conceived to reduce the file size of

images as much as possible without affecting the quality of the image. The JPEG algorithm is publicly available; this allows it to be tailored to suit different applications, such as the requirements of this project.

2.2 Graphics Rendering

Current graphics cards are specifically designed to remove the need for the CPU to render the scene: all the CPU needs do is pass the 3D scene information to the hardware which then produces a buffer for later display. Because processors themselves are more powerful, they are able to generate more complex scene information for the graphics card. This, combined with the abstraction provided by APIs such as OpenGL, makes it easier to create complex virtual worlds.

Unfortunately, PDAs do not have the same graphic processing power as desktop computers. The lack of a dedicated GPU (Graphics Processing Unit) means that the processor has to take on the work of rendering, and of course the limited processor speed adds its own constraints.

Thus, the processing power of a server was used to calculate and generate 3D scene data, which can be captured directly from the buffer. This static image can then be displayed separately, without re-calculating the coordinate geometry. This is similar to work by Engel [2], who optimised distributed image viewing by limiting the responsibility of the PDA to simply displaying an image without any of the rendering overhead.

3. RENDERING

3.1 Overview

Maintaining 10 fps on a distributed client, particularly one with limited processing abilities, can be difficult. Thus, it was necessary to first analyze the capabilities of the network and client when developing a solution.

3.2 PDA Graphical APIs

Although the rendering abilities of a PDA (such as the h4150 we used) are limited, there were two packages available that handle displaying images on screen. These two C# packages were tested for comparison before a choice was made, and were the Windows Graphical Device Interface Plus (GDI+) and Microsoft's Game API (GAPI). GDI+ is the built-in library for Windows CE, which is the operating system that runs on the Pocket PC.

In a Windows program, the output to an I/O device such as the display or printer requires going through an intermediate object known as the device context (DC). The purpose of the DC is to make the windows program device independent, so that the program will work with any type of display or printer without requiring a recompilation. In the Microsoft Foundation Classes (MFC), there is a detailed set of classes that deal with the DC. The general framework that deals with the DC is referred to as the GDI (Graphical Device Interface). In .Net, the GDI interface has been greatly refined and is referred to as GDI+ and deals with the DC in a similar manner. GDI+ needs to deal with a device context prior to performing any graphical operations and this lead to it performing poor performance. Furthermore, the GDI+ classes encapsulate some important Windows APIs to create graphical outputs. This library is un-optimized, and further contributed to the less-than-interactive frames rates GDI+ allowed.

GAPI, the alternative examined, is a small API that makes directly reading to and writing from the device's display memory possible, something which GDI+ presently lacks. GAPI consists of not much more than access to a pointer to the display memory as well as information about the display properties. It is a fast an efficient 2D graphics library and provides all the functionality needed to display complex scenes on PDA architectures, and thus was chosen for PDA image displaying.

3.3 Rendering Techniques

A number of optimizations were considered whilst developing the renderer. The first was frame coherency, where only the differences in frames are transmitted. However, according to SGI [5] there is minimal benefit in calculating and sending frame differences to the client when scenes are constantly and rapidly changing, therefore an alternative approach was needed. In order to answer the question as to how a PDAs capabilities could be exploited, entire compressed frames were sent to the device where its hardware performance was observed. To increase the frame rate on the client side, the following two techniques were investigated:

1. We attempted a frame caching technique on the client side, where recently visited frames were stored on the PDA. Whilst this technique showed improved long term results, it soon became clear that an alternative approach was necessary due to its short term performance problems, and memory-demanding nature.
2. The amount of data was minimized by transmitting wire frames as opposed to textured views. This technique proved to have the best server compression times as well as low image processing and display times when combined with a silhouette technique.

3.3.1 Frame Caching

The reason behind implementing a caching technique was to avoid relying solely on the transmission of data over the network. Theoretically, this meant that we would decrease the amount of time spent sending, receiving and processing data. As there is a mapping of real and virtual world co-ordinates i.e. a users XYZ position in the real world should correspond to position XYZ in the virtual world, we thought it would be best to store these co-ordinates in a three dimensional lookup table. So as we move around the world, the client would first check the lookup table to see if we had visited this 'region'. We use the word region as it is not always possible to be in the exact same position. If the lookup table had a recently visited entry, then the client would use these coordinates to extract the scene from memory and thus not find it necessary to send a network request to receive more data. If the table was empty, the previously described send and receive protocol would be initiated and an entry would then be added to the lookup chart.

The problems with this technique included the following:

1. Short term: The time it takes to check for and cache an image exceeds the time it takes to directly receive a frame. This adds to the delay in retrieving the next frame, reducing overall display rate.
2. Long Term: While it is possible to achieve 6fps after most scenes have been cached, cache can become extremely large, causing the PDA to become unstable. Unused frames can be deleted, but then the short term slow down above comes into effect when the region is revisited.
3. Directional combinations: Since the user has six degrees of freedom from any location, the number of frames visible from any point was potentially very large. Furthermore, if that region was *not* visited again, the lookup and cache time already spent would be wasted.

Thus, the poor performance of this approach meant it was not considered for the final system. While there were some speed gains in the long run, there was still the computational overhead of ensuring a balance is struck between the amount of data to save, the size of the cache and the short term frame delays. It was thus decided that another approach needed to be investigated.

3.3.2 Silhouetted Wire Framing

The reason behind implementing a wire frame technique was to ensure that we minimise the amount of data sent to the client and at the same time reduce the amount of processing on the PDA. Theoretically, this simple technique meant that we would decrease the amount of time spent sending, receiving and processing data. When a wire frame view is created, we can exploit the property of the objects surface areas always being blank/black, these pixels are referred to as empty. On the client side, it made sense to only display those parts that make up the outlines of the object, and ignore the empty pixels. In terms of the display, the image would look no different to the original and in essence we end up getting the black for free.

This technique has the equivalent compression ratios of transmitting flat shaded images. It also has the added benefit of decreasing the amount of data sent and the amount of time spent on computing pixel operations (as we throw away most of the data). After the RGB values are captured, it is possible for the server to determine the black pixels, i.e. zero valued RGB data, and ignore them yet still keep track of their position in the pixel buffer.

Now that the PDA could ignore much of the image and only display essential sections, processing time decreased and display rate improved. Obviously, however, the wire frame image is not ideal to view the world, so a combination of approaches was used:

1. When the PDA started moving, only wire frame images were sent through so position can be seen.
2. The moment the movement stopped, the full textured image of the last wire frame was then sent to the client for display.

This approach seemed to solve many of the latency problems we were experiencing, whilst still providing a usable view to the client.

From an HCI perspective, the wire frame view did not seem as if it would provide the best aesthetic appeal to the user, as back facing polygons were now visible. This tended to clutter scenes and make certain objects seem as if they were solids (as can be seen in the wire frame view). To address this, a hidden line removal technique was implemented that removes all lines but the silhouette edges. These are the boundaries between adjacent front facing and back facing surfaces, and are essentially just an outline to a shape. The procedure for this involves solving the partial visibility problem, so that only those parts of the silhouette edges, which are not occluded by the interior of any front facing surface, are rendered [4]. The basic method of achieving this effect is to render back facing polygons in wire frame mode and front facing polygons in a filled color, in this case black was used. A two pass rendering scheme was used, where the front faces were culled on the first iteration and the back faces are culled on the second iteration. On both passes, the width and colour of the back and front facing polygonal edges are adjusted to create silhouetted edges of constant thickness (in any colour).

Incorporating this technique further optimized image transmission and display by reducing even further the amount of data to send and process.

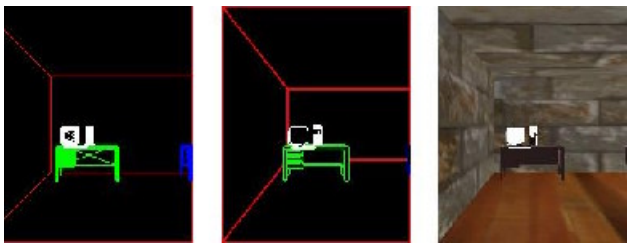

Figure 1: Wire frame vs. Silhouette vs. Texture Modes

4. NETWORK COMPRESSION

4.1 Compression Overview

As a wirelessly enabled PDA uses the 802.11b Wireless Protocol, a maximum shared bit rate of 11Mbps is allowed; though a bit rate of 1Mbps is more likely. Based on previous research, a frame rate of 10 fps is necessary for interactive use of an image/video streaming application. Since 1Mb is 125KB the network's bit rate will constrain the image sizes to 12.5 KB or less each to achieve the interactive frame rate. Based on these restrictions in bandwidth, a lossy image compression technique was required for the project to allow the frames to be transmitted at a suitable rate.

Thus, the JPEG compression algorithm was chosen based on two points.

1. It is a transform coding technique and provides greater data compression compared to predictive methods, although at the expense of greater computational requirements.

2. JPEG is a superior compression algorithm when full-screen image compression is required, as is performed in this system.

The algorithm developed for this project was slightly different to traditional JPEG, as it implements Discrete Wavelet Transforms (DWT) as opposed to Discrete Cosine Transforms (DCT). The reasoning behind this is that wavelet-based coding provides substantial improvements in picture quality at higher compression ratios than other techniques, which is a concern at high compression ratios, as well as performing fewer calculations in than DCTs, thus improving decompression time on the PDA.

4.1.2 Compression Process

Initially, the image data is separated into RGB colour space and then converted from RGB to YCC colour space. The chrominance components are then down sampled, in an attempt to take full advantage of the weaknesses of the Human Visual System, and to represent them with fewer bits [1]. Once the down sampling is complete, each YCC component is then transformed and quantised separately. This allows for each channel to be treated separately, and the properties of each to be exploited fully for improved compression ratios.

Once the quantizing is completed, the separate image components are then rearranged into a single array for the encoding process. The components are rearranged into a single array to improve the encoding and compression ratio, as encoding performs better on larger data sets. The compression library used was zlib, which uses Huffman coding and LZ77 compression.

Figure 2. Original image (left), uncompressed image (right).

4.1.3 Decompression Process

To improve the quality of the decompressed wire frame images, a simple and effective technique was implemented. As the wire frame image hold only extreme values for the pixels, e.g. for a red pixel the value will be 255,0,0 (RGB) and for black 0,0,0, etc., it allows one to manipulate the values to become close to the original pixel values. When the image is converted from YCC to RGB, any pixel values below 127 can be zeroed (made black) and any above 126 can be set to the extreme value for that pixel, i.e. set to 255. This technique results in any blurring of the pixels, due to the compression process, being greatly reduced and the lines in the decompressed image appearing sharper.

4.2 Networking

The TCP/IP was chosen for the wireless connection, since the guaranteed delivery of packets was essential. Discarding lost packets may result in a lost image if the user has only moved a small distance, such as a one frame change, and nothing would be displayed on the client.

Both the client and server were developed asynchronously. The server, because it had to listen for PDA interface events whilst being able to send frames intermittently, and the PDA for the same reason: frames arrived in an unsolicited manner, whilst still allowing the user to interact with the device.

5. INTERFACE & INTERACTION

5.1 Overview

Virtual Window provides the user with an unprecedented method of interaction with a virtual environment. The aim was to give real world movements a corresponding change in the virtual world. This was achieved by attaching a 3D wireless mouse to the PDA. Now, as one's body was rotated in any direction, a gyroscope in the mouse picked up motion and generated a perspective change in the virtual environment. However, there also needed to be way to navigate around the world in lieu of real-time positional information. For this, comprehensive design and validation methods were undertaken to ensure the interface was intuitive and satisfactory to user requirements.

5.2 Design

As part of a holistic approach to system design, a good deal of emphasis was placed on the user requirements and interaction styles applicable to navigating around the world, that also allows user movement in a physical space when rotating. The Star Model [3] of user-centered design was followed, which combines design phases such as prototyping or evaluation in a multi-iterative approach that ensures a high degree of attention to system usability and efficiency.

As a result of user interviews and testing, two methods of navigating around the world were decided upon. The first was a tilt method, where the user could offset the PDA from a starting point, and hold it in that position to begin walking forward in the world. Returning the PDA to its start position cancelled the walk movement. The second method was standard use of the PDA to generate a walk movement, in a similar method to interactive computer games.

5.3 Validation

Once the system, including its interface, was created, a full usability experiment was conducted on 10 participants. The participants were randomly assigned to two groups, who each tested the system and only one of the walking methods, to avoid learning effects and possible biasing.

The format of the test for each user was as follows:

1. The user spent at least five minutes using the system in whichever way s/he saw fit, to complete a task given at the start of the session.
2. A short questionnaire was given to the participant, to record their opinion on various topics.
3. A structured interview was conducted to elicit user opinion in more detail.

The survey and interview provided qualitative data with which to judge the system. Quantitative data was provided by observing the users whilst they interacted with the system, and measuring their time to comfort (i.e. successfully navigating to a specified point), and the number of mistakes they made whilst getting to their destination.

6. RESULTS

6.1 PDA Display

As mentioned in Chapter 4, the difference we found between GAPI and GDI+ is considerable. The following table represents the time for each package to perform the pixel operations required for display. Since GAPI has no Device Context to work through, it is roughly 25 times faster than GDI+ in textured mode, and 8 to 10 times faster in silhouetted mode.

Table 1: GAPI vs. GDI: Time to perform pixel operations

Mode	GAPI (ms)	GDI+ (ms)
Textured	145.7333333	3601.833
Wire frame	32.96666667	332.8065
Silhouette	29.73333333	244.5

Extensive testing was also performed using GAPI to determine the best frame size for transmission and display. 160x200 was decided on, since it provided a good balance between ease of viewing, and small image size.

Table 2. PDA Display Rates

Method	Time for Single Frame (s)	Max Frames Per Second
Textured	0.1856663	5
Wire framed	0.0713667	14
Silhouetted	0.0680333	15

The table above presents the average time to render a single frame, including pixel operations and screen update, and shows the maximum number of frames per second using the package.

6.2 Compression Results

The previous results represent images in their raw, i.e. uncompressed form. However, image display time is far exceeded by the time it takes to transmit an uncompressed image over the WiFi connection. Thus, compression was used in an attempt to reduce the waiting time on the PDA.

A server was developed to compress frames as they were produced by the renderer, and achieved an on-the-fly compression speed of 24 frames per second, with a compression ratio of 19:1. This reduced image size from 93.5Kb, to around only 5Kb, meaning network transmission occurred in under 5ms.

However, the PDA proved to be fatally ineffective at decompressing the incoming images. A textured image took at least 1500ms to decompress, since the YCC has a number of passes which need to be executed. Attempting to use the standard RGB color space reduced decompression time to approximately 500ms, but this is still to slow to allow interactive frame rates.

Because of this, our demonstrable implementation transmitted uncompressed frames, which provided only 4 frames per second due to network latency.

6.3 User Testing

The survey used to question test participants asked them to rate various aspects of the system, with a score of 1 representing strong approval, and 5 representing strong disapproval. The questions focused on the ability to walk or navigate in the world, look around the environment, and understand or easily view the images being displayed on screen.

The mean value for ease of walking of the group that tested the tilt method, was 3.6, contrasted with a mean of 1.8 for the group that used the D-pad. There was also a correspondingly high number of mistakes in navigation by the tilt group, with a score of 12 versus 3.5 for D-padding. These differences can largely be attributed to network lag: a slow refresh speed meant tilt users were unsure of when their actions were initiated or completed, leading to overcompensation. However, there were other complaints unrelated to refresh rate, including a dislike of having PDA movements work in two different ways (looking and walking), requiring some sort of mode change.

However, during interviews afterwards, all of the participants were excited by the Virtual Window concept, and found the move to dynamic interaction methods from traditionally static ones to be natural and intuitive.

7. FUTURE WORK

Because of the novelty of the Virtual Window concept, combined with the short time span over which the project ran, there is great potential for the project to be developed further.

With the increase in mobile processing and rendering capabilities, it is becoming more and more likely that the graphics work can be shifted to the PDA, completely removing the middle networking and compression tier. Even if the graphical capabilities are not sufficient for remote rendering, the PDA may be able to handle decompression as processing power increases.

The system can also be enhanced by adding position-relative information to the display. For example, at certain locations, or when looking at certain objects, an information text or audio message can load, giving relevant information about whatever is in focus. This has potential for use in the tourism or educational sectors, amongst others.

Lastly, some sort of triangulation or other position-aware method can be implemented that gives the system constant positional information, removing the need for users to navigate through an environment manually, and opening up options on another plane of interaction.

8. CONCLUSIONS

This project found that developing for platforms that are limited in some respects requires the development and testing of a number of different solutions to find the optimal solution. For example, wire framing increases display rate more than cache coherency, even though the latter method has frames pre-loaded. It also found that techniques to reduce processor load on handheld devices is extremely important, since the limited display abilities and packages available do not provide a friendly environment for image display.

The research also found that compressing images to try and obviate the network bottleneck fails to increase system speed, since even though the server is able handle transformation with ease, the same cannot be said for the PDA.

Ultimately, any work done to optimize or further develop the system implemented can be done with confidence, since extensive user testing showed that the interaction methods attempted will be well received by users, and will have excellent real-world potential for expansion.

9. ACKNOWLEDGMENTS

We would like to thank our supervisors, Patrick Marais and Gary Marsden, for giving us the guidance and support needed to complete this project.

10. REFERENCES

[1] Deknuydt, B., Desmet, S., Van Eycken, L., and Oosterlinck, A. A Human Visual System based Block Classification Algorithm for Image Sequences Coders. *Proceedings SPIE Conference on Visual Communications and Image Processing*, 2308.2, (September 1994), 1401-1410.

[2] Engel, K., Sommer, O., and Ertl, T. A Framework for Interactive Hardware Accelerated Remote 3D-Visualisation. Available at: http://www.vis.uni-stuttgart.de/~engel/VisSym2000.pdf.

[3] Hix, D., and Hartson, H. *Developing user interfaces: ensuring usability through product & process*. John Wiley & Sons, New York, 1993.

[4] Raskar., R., and Cohen.,M. Image Precision Silhouette Edges. *Proceedings of the Symposium on Interactive 3D Graphics (I3DG)*, (Atlanta GA, April 1999), ACM Press, 135-140.

[5] SGI. OpenGL Vizserver™ 3.1: Application-Transparent Remote Interactive Visualization and Collaboration. Available at: http://www.sgi.com/pdfs/3263.pdf