A Middleware Architecture for Mobile 3D Graphics*

Emmanuel Agu Kutty Banerjee Shirish Nilekar Oleg Rekutin Diane Kramer

Department of Computer Science,
Worcester Polytechnic Institute,
Worcester, MA 01609, USA

Abstract

Mobile graphics, which involves running networked computer graphics applications on mobile devices across wireless networks, is a fast growing segment of the networks and graphics industries. Running networked graphics applications in mobile environments faces a fundamental conflict; graphics applications require large amounts of memory, CPU cycles, battery power and disk space, while mobile devices and wireless channels tend to be limited in these resources. In order to mitigate mobile environment issues, some form of adaptation based on a client device's capabilities, prevailing wireless network conditions, characteristics of the graphics application and user preference, is necessary. In this paper, we describe the Mobile Adaptive Distributed Graphics Framework (MADGRAF), a graphics-aware middleware architecture that makes it feasible to run complex 3D graphics applications on low end mobile devices over wireless networks. In MADGRAF, a server can perform mobile device-optimized pre-processing of complex graphics scenes in order to speed up run time rendering, scale high-resolution meshes using polygon or image-based simplification, progressively transmit compressed graphics files, conceal transmission errors by including redundant bits or perform remote execution, all tailored to the client's capabilities. MADGRAF exposes our Mobile Adaptive Distributed Graphics Language (MADGL), an API that facilitates the programming and management of networked 3D graphics in mobile environments.

1 Architectural Firm Mobile 3D Graphics Scenario

Ulo corporation is a multi-national architectural firm with clients and workers in 50 countries across 5 continents. Ulo maintains a large database of 3D architectural drawings of various types of buildings. Some of Ulo's clients are located in remote areas in with limited internet access and low bandwidth links. Additionally, to accommodate workers at various levels, Ulo has found it useful to equip its workers with PDAs, laptops and cell phones with graphics capability. Different teams of architects work on different projects which are maintained in Ulo's database. Initially, an Ulo team visits a client and after preliminary discussions, retrieves possible design solutions and show them to the client. These serve as starting points of the design process. After the client selects a viable option and requests modifications, the architects annotate the diagrams and return to Ulo's office to make necessary amendments. Periodically, the architects return to the client to show progress and seek more feedback, towards a mutually agreeable design. In cases where the client resides in a remote location, the architects can select a disconnect operation mode, in which case the relevant drawings are pre-downloaded (hoarded) onto the client, before disconnection. On reconnection, the hoarded files are reconciled with the database.

2 Introduction

Mobile graphics, which involves running networked computer graphics applications on mobile devices across wireless networks, is a fast growing segment of the networks and graphics industries. In on-site consulting situations (such as the Ulo scenario above), an interior designer or architect can show clients preliminary designs and seek feedback towards a mutually agreeable final solution. Field service technicians repairing complex equipment such as copiers and automobiles can retrieve and playback synthetically-generated animations of repair manuals, focussing on relevant sections from new viewpoints. In electronic commerce, major corporations are already using interactive 3D catalogs to demonstrate product features, enabling rich interaction and virtual product examination.

Running networked graphics applications in mobile en-

*Funding for this work was provided in part by the National Science Foundation grant number 0303592

Proceedings of the 25th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'05)
1545-0678/05 $20.00 © 2005 IEEE

vironments faces a fundamental conflict; graphics applications require large amounts of memory, CPU cycles, battery power and disk space, while mobile devices and wireless channels tend to be limited in resources. Specific issues to be dealt with include mobile device limitations, wireless channel, mobility and network infrastructure issues [3]. *Mobile device limitations* in memory, CPU power, disk space, screen size and battery-life make it difficult to load and manipulate very high resolution geometric models or run sophisticated rendering algorithms that are necessary for visual photorealism. Highly interactive distributed applications cannot be supported due to *wireless channel issues* such as low transmission bandwidths and high, variable Bit Error Rates (BER) that lead to prohibitively high, variable model download times. For wide area networks with wireless hops, *network infrastructure issues* such as variable latency and low bandwidth caused by slow network routers, switches and backbones further exacerbate networking. *Mobility issues* such as disconnection and handoff require additional algorithms to manage the changing network topology, network address resolution, and guarantee continuous service. Finally, any progress made in algorithms that run on the Graphical Processing Unit (GPUs) of graphics cards have limited use since most mobile devices are still not equipped with 3D graphics accelerators.

In order to mitigate the above issues, some form of adaptation based on a client device's capabilities and prevailing wireless network conditions is necessary. *Application-Aware Adaptation* [13] in mobile computing dictates any trade-offs made should exploit the specific nature of each type of mobile application. The mobile graphics industry is already embracing the adaptation theme for mobile graphics in directions that include the development of new efficient 2D vector graphics standards such as Scalable Vector Graphics (SVG), reduced command-set languages and versions of graphics libraries such as OpenGL ES, wireless gaming engines, and low-power GPUs. However, most of these directions are niche solutions and more general solutions for mobile 3D graphics have been slower to appear.

The heart of the wireless graphics problem lies in the scalability of the currently adopted polygonal mesh representations, which show greater realism as the number of faces in a mesh increases. Today's state-of-the art meshes that are automatically produced by 3D scanning can easily consist of billions of faces (or hundreds of gigabytes (GB)), each of which have to be rendered using a number of algorithmic steps, leading to an explosion in rendering complexity. Manipulating, processing and displaying such large models can be extremely unwieldy.

Research in mobile graphics tries to exploit the characteristics of graphics applications in adapting to resource constraints in mobile environments. Mobile graphics applications have varying degrees of interactivity, latency tolerance, refresh frame rate and scene complexity. Thus, different mobile applications will benefit from different forms of adaptation. For instance, a ray tracer performs thousands of floating point computations per pixel to determine the final pixel color, exhibits very low interactivity and would benefit from remote execution to offload expensive computations to a high end server. A mobile gaming application on the other hand would requires low latency and would benefit more from a reduction in the resolution of geometric meshes, images and foreground graphics content to preserve interactivity.

We present the Mobile Adaptive Distributed Graphics Framework (MADGRAF)[1], a middleware architecture for interactive 3D graphics in mobile environments. MADGRAF exposes the Mobile Adaptive Distributed Graphics Language (MADGL) which facilitates the programming and management of networked 3D graphics in mobile environments. Specific research directions being investigated as middleware components in MADGRAF include device-optimized pre-processing of complex graphics scenes in order to speed up run time rendering, automatically scaling down the resolutions of complex scenes based on target mobile device capabilities [14] the offloading of draining rendering steps to more powerful servers (remote execution and cyber-foraging) [3] efficient transmission via compression [16] progressive (batch) transmission and rendering of large models [17]concealing the effects of wireless errors by including redundant bits [18]and the use of graphics-aware network protocols We are also investigating machine learning decision algorithms in our centralized decision module at the MADGRAF server. By closely tracking resource usage at the client as well as graphics application performance, necessary trade-offs can be made to improve overall performance.

3 MADGRAF

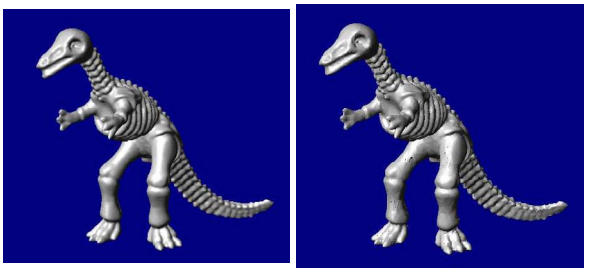
The generic functions of a graphics library such as OpenGL and DirectX include scene modeling, import of predefined scenes, scene viewing and rendering. It is instructive to note that MADGRAF and MADGL *do not* duplicate these functions, but compliment them by focussing on aspects such as the import, adaptation and rendering of remotely stored graphics files, remote execution and geometric compression, which are of direct benefit to a mobile device. Our proposed research thrusts are being developed using MADGRAF as a foundation.

3.1 MADGRAF Middleware

It is envisioned that MADGRAF will service a diverse spectrum of mobile graphics applications ranging from ray

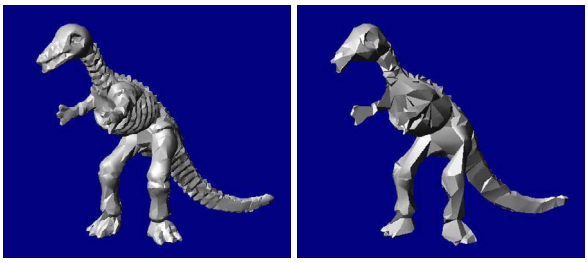
tracers to mobile games, in different ways. A set of middleware components, can be combined in various ways for different applications, exploiting any trade-offs that makes sense for a given application. Therein lies the power of the MADGRAF framework. The main middleware components in MADGRAF are:

- *Geometric simplification:* algoritms convert an input high-resolution mesh into a lower resolution version (less polygons) of the original mesh [14] Figure 1 shows several representations of the same mesh that are suitable for different mobile clients.



(a) Original Scene File (112,400 faces, 56,200 vertices, 5.09MB)

(b) Scene File 50% simplified for a Laptop, 3.2MB



(c) Scene File 90% simplified for a PDA, 560KB

(d) Scene File 98% simplified for a Cell Phone, 108KB

Figure 1. Various resolutions of a mesh suitable for different mobile clients

- *Image based simplification:* convert an input high resolution mesh into a sequence of high resolution images. The main advantage of converting to images is that even at high resolutions, images (or textures) have lower memory (storage) requirements than meshes. However, images permit less interaction than meshes [15].

- *Remote execution:* allows a portion of a computation task or rendering in the case of graphics, to be performed on a remote server. In our many envisaged usage scenarios, the original high resolution meshes will be stored on the server and remote execution could simply involve having the server start a portion of the rendering process before transmitting partially rendered results back to the client.

- *Geometry compression:* reduces the size of the geometric mesh prior to transmission [16]Unlike ASCII compression (such as gzip on Unix systems), geometry compression exploits the connectivity of input meshes and is thus more efficient for meshes.

- *Progressive Meshes:* Large high-resolution graphics mesh files can take a long time to download over most wireless networks, which generally limits interactivity. In certain situations, a quick rough preview image that could be improved iteratively, would be adequate. For instance, an experienced interior designer who is trying to match a blue leather sofa to a virtual model of a room would probably reach a quick decision just by seeing a rough rendered image of the sofa in place. Progressive transmission [17] is a hierarchical data structure which encodes a large mesh such that a small base mesh is transmitted and rendered at first, followed by batches of mesh parts which could be added iteratively to increase the resolution of the rendered image [17].

- *Unequal Error Protection (UEP):* is a Forward Error Correction (FEC) technique that adds redundant bits to a mesh prior to transmission such that a client can repair meshes that are damaged due to transmission errors [18], and sometimes eliminate retransmissions. Since the progressive mesh data structure is hierachical, each layer contains meta information about the other layers. UEP recognizes recognizes this hierarchy and adds more redundancy to more important packets.

- *Graphics-Aware Wireless Network Protocols:* Highly interactive graphics applications such as games are highly sensitive to network latency [19]. We have shown that latencies as low as 150 milliseconds affect the performance of users in the First Person Shooter (FPS) class of games. However, our earlier studies investigate only latency patterns due to infrastructure bottlenecks arising from geographically distant multi-user game servers on the Internet. Multipath fading and the hidden terminal problem [5] on wireless networks can lead to significant latencies. We would like to characterize the nature of latencies due to the wireless channel and its effect on graphics application users.

3.2 Examples of MADGRAF Middleware Usage

In the Ulo architectural firm scenario in section 1, since the Ulo architects still require a significant amount of interaction with the building models, the models must be retrieved in a geometric mesh format. Since rendering the original mesh at full resolution, may not be possible at interactive rates, *geometric simplification* based on the client's hardware configuration could be applied. The decision module at the MADGRAF server would determine what the output simplification ratio should be. The MADGRAF server also determines if it is necessary to perform part of the rendering of retrieved buildings for the client (remote execution), and also what level of compression or efficient transmission is necessary based on the bandwidth and prevailing error conditions on the wireless channel.

Since users of **interactive 3D shopping guides** expect high quality images to show off product features and can tolerate limited interactivity, image-based simplification of meshes is ideal. In a **mobile gaming** application, near and foreground characters require high interactivity and geometric simplification is preferable. Image-based simplification can be used for far and background images. In a **ubiquitous synthetic posters** application, the quality of rendered output from ray tracing engines is highly dependent on the resolution of input meshes and as little simplification as possible should be done. Additionally, this application has very low interactivity, making remote execution a natural choice. Table 1 summarizes the nature of mobile applications and related MADGRAF adaptations. In all the scenarios, although we highlight a main enabling MADGRAF adaptation, other adaptations may be used additionally. For instance, compression, unequal error protection or progressive transmission may be applied as well.

3.3 Overview of MADGRAF Operation

In this section, an overview of MADGRAF's operation is now given. Figure 2 shows the main components of the MADGRAF architecture. The key server components for adaptation are the *Adaptive Graphics Engine*, the *Adaptive Transmission Engine* and the *IntelliGraph* (a centralized intelligent decision engine). The main client-side components are the *Environment Monitor* and the *Profile Generator*.

When a MADGRAF client program requests a 3D graphics file that is stored on a MADGRAF server, the client's profile generator retrieves information on the mobile host configuration and capabilities along with the current quality of the wireless communication channel from the environment monitor, and then builds the client profile, which it sends to the server along with the originally requested file. At the MADGRAF server, the request and client profile are forwarded to the intelliGraph, which decides if scaling is required and what optimizations and conversions should be done for the mobile client. Adaptation takes place in four main ways: (1) *adaptive graphics* such as geometric mesh simplification and image-based simplification, that pre-process input scene files; (2) *efficient transmission* techniques such as geometry compression, error concealment and progressive transmission [17] that address wireless network transmission bottlenecks; (3) *architectural* and distributed systems techniques such as remote execution, distributed rendering [1], and parallel rendering which attempt to borrow rendering resources from more powerful network servers; and (4) *intelligent* caching which retains the results of expensive computations for reuse in the future. These adaptations and the decision process forms the core of our short term goals and will be expounded later as specific projects in section 4.

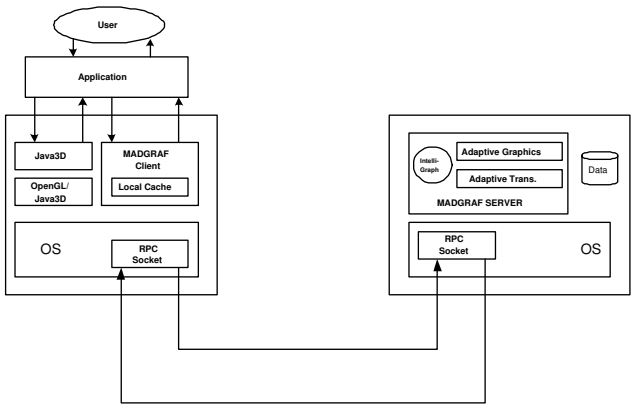


Figure 2. MADGRAF Components

The IntelliGraph on the MADGRAF server manages the entire adaptation process, and utilizes the Adaptive Graphics Engine to generate files which are best suited for use by a given mobile client, and the Adaptive Transmission Engine to transmit these files efficiently over a wireless network. The Adaptive Graphics Engine contains algorithms for geometric conversions such as scaling, polygon simplification, view-dependent optimizations and image-based techniques, while the Adaptive Transmission Engine contains efficient transmission algorithms such as progressive transmission, compression, and graphics-aware network protocols. Since a lot of the inter-conversions are computationally expensive and could potentially drain server resources, the client's profile and previously requested files are cached in the profile cache, while the resultant optimized geometric models are cached in the scene cache and may be re-used in subsequent requests.

The Environment Monitor gathers information and statistics about the MADGRAF client's operating environment, which are then used to build the client's profile, and is a key component for mobile host and communication chan-

Application	Interactivity	Latency	Scene Complexity	Frame Rate	Main MADGRAF Adaptation
Architectural Firm	Medium	High	Very high	Low	Geometric simplification
Online Shopping	Medium	Medium	Low	Medium	Image-based simplification
Realtime 3D Game	High	Very low	Low	High	Geom. and Image-based simplif.
Remote Ray Tracer	Low	High	High	Low	Remote execution

Table 1. Mobile Graphics Applications

nel adaptation. The information collected include *static attributes* such as the hardware specification (CPU, memory, hard disk and screen size) of the mobile host, *dynamic attributes* of the operating environment (channel error rate, download times, latency and system battery power) and *depreciating resources* such as battery power. The client profile may also include application requirements on speed and interactivity, Quality of Service (QoS) parameters and location information that may be useful for some optimizations.

4 MADGRAF Research Projects

Using MADGRAF as a foundation, we now define the research thrusts which we are currently pursuing in mobile 3D graphics. While the results of each thrust shall generally lead to the development of a module of the MADGRAF system, the results should be general enough to apply to a wide range of mobile graphics systems. Due to space constraints, we simply outline these research thrusts which are expounded further in [1]:

- *Budget-based Graphics Transcoders for Heterogeneous Mobile Clients:* which develop metrics for the pre-processing of input meshes, subject to a *budget* on available memory, CPU or other mobile device resource.
- *Remote Execution for Mobile Graphics:* in which based on prevailing conditions, each of the various stages of the 3D graphics pipeline can be flexibly mapped either a client or surrogate server using our novel *pipeline-splitting* mechanism.
- *Efficient Transmission of Graphics Content and Wireless Networking:* where we combine progressive transmission, geometric compression, Unequal Error Protection (UEP) and graphics aware network protocol in order to satisfy application-dependent restrictions on latency, interactivity, image resolution and other suitable performance criteria.
- *Real-Time Performance Monitoring and Tracking:* In which we instrument the mobile device in order to constantly receive feedback the demand and supply of its system resources and wireless channel condition.
- *Machine Learning-Based Intelligence using History-Based Linear Prediction:* in which by observing the resource usage of certain graphics models and optimizations, we develop heuristics for the automatic selection and employment of our middleware modules for the benefit of the mobile device.
- *MADGL API Definition:* in which we are defining an intuitive powerful programmable interface to our system to increase flexible usage and rapid prototyping.
- *MADGRAF Server Optimization:* in which we are investigating scalability issues on the server based on workload analysis. We are also implementing *out-of-core* algorithms which are paging techniques used in the pre-processing of extremely large models that cannot fit into the server's memory.

5 Current MADGRAF Prototype

Our initial investigations showed that very little research on adapting 3D graphics applications in mobile environments has been done. To get a concrete sense of the issues involved, as well as to fuel our research efforts, we chose to develop MADGRAF while prototyping. A minimal prototype has been developed. We shall now describe briefly our accomplishments thus far.

5.1 Accomplishments Thus Far

- *Minimal Working Prototype:* MADGL and MADGRAF were developed initially using an object-oriented design methodology with final implementation in Java, which due to its platform independence, allows MADGRAF to run on various mobile devices with little modification. A few third party tools were also used in developing MADGRAF, either as offline pre-processing tools (3D model converters) or as third-party components (polygon simplification, VRML viewers) which have been integrated into the prototype to reduce development time. The high resolution polygonal meshes used models were retrieved from the Stanford 3D Scanning mesh repository [4]. Figure 3 is a screen shot of our MADGRAF prototype.

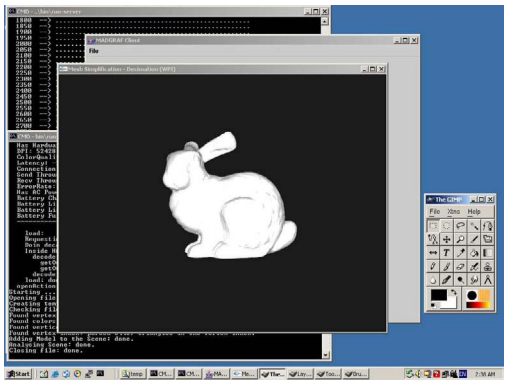


Figure 3. MADGRAF client screen shot

- *Automatic Constraints Detection:* Since we wanted to make MADGL and MADGRAF easy to use, we chose not to burden the client with specification of its resource constraints, configuration and network conditions, which are used by the server's IntelliGraph in decision-making. Constraints detection is done automatically once a MADGRAF client requests a graphics file. As such, it was necessary to develop platform-dependent modules for detecting client configurations and periodically sample prevailing device and network conditions. The MADGRAF Constraints/Environment detection module was developed using native code since it involves accessing specific operating system variables, which was not possible in pure Java due to Java's platform independent design. On a Win32 client, system constraints were accessed by querying the windows registry.

- *Energy Profiler:* One key success is the development of *PowerSpy* [8], a software-only power meter for determining power usage on a Windows mobile client, to enable the server to make energy-efficient decisions.

- *Pipeline splitting for remote execution validated:* We instrumented a version of Mesa, a popular software implementation of the OpenGL graphics library, with networking code to create Remote Mesa (RMesa). Our preliminary tests show that our idea of remote execution for mobile graphics using fine-grained pipeline-splitting, has merits and are expounded in [7].

- *Preliminary system performance measurements:* Our working prototype allowed us to get a rough sense of how long the retrieval, transmission, simplification and rendering of meshes a given resolution would take. These early measurements reinforced some of our initial assertions, and are informing our current directions. Table 2 is a sample of measurments on our current prototype. Since graphics applications are so different in nature, we are developing a wide range of test applications and perform detailed measurements as well as perceptual experiments to evaluate how much distortion is experience by the human visual system.

6 Related Work

Some of the adaptation concepts used in MADGRAF such as polygonal mesh simplification, image-based simplification [15], geometry compression [16] and progressive transmission[17] have been proposed separately in the literature. However, combining them in an integrated system for mobile environments is novel. Specifically, the application of mobile environment constraints in determining which sets of adaptations to use and to what extent, is a key contribution of the MADGRAF project. Augmented reality systems [11] integrate a number of these techniques but are immersive, while MADGRAF is non-immersive. Web-based graphics systems, [12],also deal with latency and bandwidth constraints on network performance.

A few related complete systems are also worth mentioning. The ARTE system [2] implements primarily adaptive graphics techniques such as polygon simplification and LOD techniques, but does not use remoted execution, parallelism, or any distributed rendering. ARTE also uses a simple decision model which does not address specific device or wireless channel constraints. Repo3D [10] is a distributed graphics library which primarily presents an object-oriented framework for the distribution of input graphics models, but does not address issues of resource-constraints.

7 Conclusion and Future Work

We have presented the Mobile Adaptive Distributed Graphics Framework (MADGRAF), a middleware architecture for mobile 3D graphics. The MADGRAF prototype is serving as a proof of concept to validate our concepts. Our research thrusts and key accomplishments thus far have all been presented.

In order to keep our goals realistic, we have limited the middleware components described in this paper. However, several ideas for long term research are already being considered. Other research themes that we would like to explore longer term include prefetching and hoarding techniques that allow offline operation in a disconnected mode, cyber-foraging [3] which allow a mobile graphics client to "lease" the services of new servers in new locations as it moves and parallelization of server-side algorithms to enable rendering on clusters. Other research thrusts include using MADGRAF to scale graphics output to heterogenous displays in ubiquitous environments and point-based graphics for mobile devices.

Percent Reduction	*No Caching*			*With Caching*		
	Reduction Time(ms)	GZip Time(ms)	Total Time(ms)	Reduction Time(ms)	GZip Time(ms)	Total Time(ms)
05	136729	5943	142672	3825	4560	8385
20	340383	4950	345334	3819	4031	7850
30	455495	4376	459871	3809	3473	7282
50	653118	3152	656270	3857	2436	6293
80	802992	1225	804217	3836	1136	4972
99	925709	26	925735	3845	50	3895

Table 2. MADGRAF prototype performance measurements

References

[1] Agu E., MADGRAF: A Distributed Architecture for Rendering Interactive 3D Graphics Applications in Mobile Environments, Technical Report, Worcester Polytechnic Institute, 2003.

[2] Boier-Martin I. M., Adaptive Graphics, IEEE Computer Graphics and Applications, 2,1 (Jan-Feb 2003), 6-10.

[3] Satyanarayanan M, "Pervasive Computing: Vision and Challenges." IEEE Personal Communications , August 2001.

[4] Stanford 3D Scanning Repository, http://graphics .stanford.edu/data/3Dscanrep/, November 2003

[5] Pahlavan K and Krishnamurthy P, "Priniciples of Wireless Networks: A Unified Approach", Prentice Hall, 2002

[6] Narayanan D, FLinn J, and Satyanarayanan M, "Using History to Improve Mobile Application Adaptation, in Proc. 3rd Workshop on Mobile Computing Systems and Applications (WMCSA) 2000.

[7] K Banerjee and E Agu, "Remote Execution for 3D Graphics on Mobile Devices", submitted for publication

[8] K Banerjee and E Agu, "PowerSpy: Fine-Grained Software Energy Profiling for Mobile Devices", submitted for publication

[9] Flinn J and Satyanarayanan M, "PowerScope: A tool for Profiling the Energy Usage of Mobile Applications", in Proc. 2nd IEEE Workshop on Mobile Computing Systems and Applications, New Orleans LA, February 1999

[10] Macintyre B and Feiner S, A Distributed 3D Library, in Proc. 25th Annual Conference on Computer Graphics and Interactive Techniques (1998), ACM Press, pp. 361-370.

[11] Gleue T and Dahne P, Design and Implementation of a Mobile Device for Outdoor Augmented Reality in ARCHEOGUIDE Project, in Proc. ACM Web3D 2002, pp. 161-168

[12] Lau R.W.H, Li F, Kunii T.L, Guo B, Zhang B, Magnenat-Thalmann N, Kshirsagar S, Thalmann D, Gutierrez M, Emerging Web Graphics Standards and Technologies, IEEE Computer Graphics and Applications,

[13] Noble B, Satyanarayanan M, Narayanan D, Tilton J E, Flinn J and Walker K, "Agile Application-Aware Adaptation for Mobility", in proc. ACM Symposium on Operating Systems Principles, Saint Malo, France, pp. 276-287.

[14] Luebke, D.P, "A Developer's Survey of Polygonal Simplification Algorithms", IEEE Computer Graphics and Applications, May/June 2000, pp 24-34

[15] Decoret X, Durand F, Sillion F and Dorsey J, "Billboard Clouds for Extreme Model Simplification", in Proc. ACM SIGGRAPH 2003

[16] Deering M, Geometry Compression, in Proc. ACM SIGGRAPH 1995, pp. 13-20

[17] Hoppe H, Progressive Meshes, in Proc. 24th annual conference on Computer Graphics and Interactive Techniques (1997), ACM Press/Addison-Wesley Publishing Co. pp. 189-198

[18] G. Al-Regib and Y. Altunbasak, "An unequal error protection method for packet loss resilient 3-D mesh transmission IEEE INFOCOM, vol. 2, pp. 743-752, New York City, NY, June 2002

[19] Sheldon N, Girard E, Borg S, Claypool M and Agu E, "The Effect of Latency on User Performance in Warcraft III, in Proc. ACM NetGames, 2003