Antti Nurminen. 2007. Mobile, hardware-accelerated urban 3D maps in 3G networks. In: Osvaldo Gervasi and Donald Brutzman (editors). Proceedings of the 12th International Conference on 3D Web Technology (Web3D 2007). Perugia, Italy. 15-18 April 2007, pages 7-16.

http://doi.acm.org/10.1145/1229390.1229392

Mobile, hardware-accelerated urban 3D maps in 3G networks

Antti Nurminen*
Helsinki University of Technology

Figure 1: *A 3D city rendered at 60fps.*

Abstract

3D maps can visualize static and dynamic features of real environments, and act as 3D gateways to location-based information. Insufficient network speed has been a major bottleneck for dynamic download of 3D content for mobile devices. 3G network technologies promise to solve this issue, allowing faster response times and higher data rates. Similarly, mobile 3D graphics hardware should provide a dramatic increase in rendering speed. We examine wireless IP network properties, and develop an optimized network scheme suited for navigation purposes. The presented system allows free roaming in the 3D scene, while progressively downloading 3D data. For case platforms, we use two 3G Symbian smart phones, one with 3D hardware and one without. Network, 3D rendering and overall application performances are measured. For a scalable 3D engine, 3D hardware improves the rendering performance by over an order of magnitude. By using a compressed network protocol and efficiently formatted 3D data, a textured but lightweight 3D city can be progressively downloaded in 3G networks fast and without degrading application responsiveness.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques H.5.2 [Information Interfaces and Presentation]: User Interfaces—Graphical User Interfaces H.4.3 [Information Systems Applications]: Communications Applications—Information Browsers

Keywords: mobile computing, 3D maps, wireless networks, VRML

*e-mail: andy@iki.fi

1 Introduction

Mobile 3D maps are interactive guides to real environments. As navigation aids, they may provide similar features as electronic 2D maps: GPS positioning, routing, access to location-based data or participation in location-based discussion, annotation, and visualization of dynamic data, such as GPS tracked users or public transportation vehicles. However, in a 3D map, everything is portrayed within a 3D view. The key feature of such a representation is the easy recognizability of the environment. The abstract 2D representation of the traditional map requires cognitive resources and topological reasoning to relate the map with the environment, whereas a 3D view could be instantly matched based on visual cues. If this essential feature is to be maintained, simplification or abstraction of 3D maps cannot be taken to the same level as in their 2D counterparts. This pushes the requirement for visual accuracy, *veridicality*, quite high. Consequently, the spatial accuracy of any data integrated to such a system must be comparable to the accuracy of the representation, and positioned also in the third dimension. In addition, data could be associated to the underlying 3D geometry. For example, an annotation could be assigned to a specific window of a specific building, instead of using a mere street address or 2D coordinates.

The overall usability and usefulness of a 3D map constitute from several factors. Responsivity of interaction, prompt availability of content, easy-to-learn but efficient navigation features, overall capabilities of the system such as annotation of the 3D scene, advanced content query methods etc. all affect the user experience. In particular, any delays will reflect negatively in the usability of the application.

Web3D 2007, Perugia, Italy, April 15–18, 2007.
© 2007 ACM 978-1-59593-652-3/07/0004 $5.00

Our scenario is a mobile user, who travels to new cities and wishes to use the same interface, a *3D city gateway*, for navigation in all locations. At arrival, the user selects a suitable city, and the corresponding model is progressively downloaded as the user navigates within it, and within the city.

A network scheme binds together 3D modeling, 3D data structures, 3D engine properties and navigation in general. We discuss these aspects, with previous work on 3D maps and progressive model downloading mechanisms, in the context of a *thin client*, a platform with limited networking capabilities. A mobile network may contain several caveats for data transfer, which we analyze.

An overall system architecture is described, presenting our choices of *external* and *internal* interfaces. The architecture allows the use of various standard data exchange formats and integration of content databases while providing compact internal represenations and facilitating efficient data transfer and rendering.

We select two 3G smart phones for test platforms, one with 3D hardware (Nokia N93), and one without (Nokia 6630). We perform simple network benchmarks with them to find real-world network capabilities. Based on the results, we select a low level network protocol, and a general data transmission algorithm. Finally, we develop a progressive 3D data download scheme with emphasis on navigation, utilizing several optimization methods such as visibility, levels-of-detail, contribution culling and 3D data importance. A compressed, binary network protocol is developed using compiled XML descriptions. The scheme is implemented by adapting and porting an existing 3D map application.

3D hardware facilitates very high fillrates in comparison to software rendering. However, in such a situation other bottlenecks such as slow file I/O or the lack of CPU power may become dominant. Probably the most important limited resource is memory, which is easily used up by textures. We ensure scalability by an efficient caching scheme, minimize memory usage and assert that our 3D engine does not pose any CPU limitations.

The resulting rendering speeds and application performances for both platforms are compared and discussed. The results show that a significant increase in networking, and a tenfold increase in rendering speed are achievable via the new 3G networks and 3D hardware.

Main contribution. The presented work identifies several key bottlenecks and potential pitfalls in developing a networked, hardware-accelerated mobile 3D application. The network scheme is optimized for navigation, using importance measures, a binary model format, visibility information, contribution culling, allowing the use of smart pre-fetching algorithms. A lightweight XML based binary network protocol is introduced. The resulting system allows free roaming in the 3D space, simultaneously with progressive download of the model, while maintaining responsiveness. Methods such as smart memory and cache management are presented that allow scalability to texture-rich large models, fully exploiting available 3D hardware.

2 Related Work

Real-time rendered 3D virtual environments require more computational resources than any other applications. Therefore, the technical challenge in providing such environments in mobile devices is demanding. All resources in mobile devices are sparse - memory, CPU, possible GPU and storage are limited and slow. The new 3G networks bring hope to network speed, and mobile 3D hardware brings hope to rendering rates. Even with these improvements, the mobile devices remain thin devices. We discuss related work in this context.

2.1 Mobile rendering

Previous work on mobile guides applying client-rendered 3D graphics has encountered serious technical problems related to rendering speed. For example, [Kray et al. 2003; Rakkolainen et al. 2001] used laptop emulation for field studies. [Kray et al. 2003] was later ported to Nokia Communicator using a proprietary rasterizer, but the rendering scheme [Przybilski et al. 2005] and available features were very limited. [Rakkolainen et al. 2001] used a high quality model, but even after considerable simplification, frame rates were less than one per second on a PDA device. A demonstration given by [Bleschmied et al. 2005], on a Nokia 6630, suffered from a 2-3 minute initialization time, rendering speed of less than 1 fps, and limitation to only a two or three textured buildings. In [Burigat and Chittaro 2005], information retrieval directly from VRML models was studied. Restricting the viewpoint to street level, the model was embedded with visibility information for faster rendering. As in [Rakkolainen et al. 2001], an existing high-quality model was simplified to allow rendering in a mobile device, and 4-5fps was achieved for textured models.

Most approaches for client-side rendering have applied direct model viewing, either with PocketCortona [ParallelGraphics 2005] [Rakkolainen et al. 2001; Burigat and Chittaro 2005] or JSR-184 [JCP 2005] [Bleschmied et al. 2005]. The visibility information in [Burigat and Chittaro 2005] certainly improves rendering speed, but the restriction to street level movement is prohibitive. To overcome these problems, a 3D engine suited for urban environments was developed in [Nurminen 2006] with OpenGL ES, applying preprocessing and optimized rendering techniques.

2.2 City models in networked environments

Developing a progressive model download scheme requires information about the 3D model, its representation and how it is rendered by the 3D engine. The following provides a view to various approaches, where modeling and transmission are combined.

Procedural city modeling, such as [Parish and Müller 2001; Wonka et al. 2003], can be used to create artificial cities from statistical data or grammars. The resulting models look real, but do not directly match with existing cities. [Royan et al. 2006] use similar procedural "2.5D" models instead of a full 3D presentation for compression purposes, and adapts the system for network transmission. In this scheme, only a model tree structure, building footprints, wall heights, roof types and other procedural parameters are transmitted. However, all textures are stored locally. [Quillet et al. 2006] attack the problem of texture storage by extracting and vectorizing façade features using edge detection. Their non-photorealistic rendering scheme applies potentially visible sets, but forces the viewpoint to street level. The 3D geometry onto which the edge vectors are rendered remains colorless, and the recognizability of this representation has not been verified.

To reduce detail in buildings far from the camera, *level-of-detail* (LOD) techniques can be used. [Döllner and Buchholz 2005] propose modeling buildings with *continous level-of-quality* representation (CLOQ), using a set of predefined basic components. Again, as the technique is procedural, it does not yield models of high veridicality. [Pasman and Jansen 2003] examine geometry simplification methods for augmented reality systems. In addition to continuous LOD, they consider replacing models with view-dependent impostors rendered at server side on the fly. The focus is in choosing representations where noticeable geometric error is minimal, assuming

a wireless connection of 2–10Mbit/s.

[Schmalstieg and Gervautz 1996] consider demand-driven geometry transmission for distributed virtual environments. Similiraties to out-of-core rendering are noted, where fetching from storage is slow in comparison to having models reside in memory. They present ideas for progressive transmission of hierarchical, level-of-detail geometry. Only models at the area of interest (AOI), defined by a radius, are downloaded with a pre-fetching scheme. This case, and the follow-up [Hesina and Schmalstieg 1998], do not address texturing.

[Döllner and Buchholz 2005] points out that the use of 3D city models go beyond mere visualization. 3D city models should facilitate semantic and topological searches, and generally be associable with any data. [Coors 2001] introduces the *Urban Data Model* (UDM) for 3D-GIS, where objects or features are associated with the abstractions *body*, *surface*, *line* or *point*. One of the key benefits is the hierarchical structure, allowing for example annotation information associated to *surfaces* be queried via the higher-level *body*. *CityGML* [Kolbe et al. 2005] is a major initiative on standardizing 3D city representations, covering the geometrical, topological, and semantic aspects of 3D city models.

[Brenner et al. 2001; Schilling and Zipf 2003] discuss automated 3D city model creation. While the techniques are promising, they depend heavily on the available data. The results do not tend to provide accurate detail, which is solved in [Schilling and Zipf 2003] by modeling and texturing "important buildings and sights" manually.

Despite the efforts above, we can expect most models to conform to currently supported formats, such as VRML or X3D. We separate *external* model descriptions, and develop a compact *internal* representation with basic support for geometry composed of triangles, and include common properties such as normals, texture coordinates and color. To optimize memory usage, our internal representation will use single texture LODs instead of mipmapping.

We will internally support model hierarchies, similar to the UDM [Coors 2001] (buildings consisting of meshes), to maintain associability of data and for hierarchical frustum culling. We will support billboard impostors as *point* objects. We will include support for incremental geometry LODs, and, in principle, static LODs. In the first case, a base model is equipped with smaller parts, which are culled away until contributing sufficiently to the view. In the second case, the whole object is replaced at the LOD limit. Based on our previous field studies (see section 5), in contrast to [Pasman and Jansen 2003], we suggest billboards to substitute complex geometry for small objects, such as statues, independently of distance or viewing angle.

3 Wireless networking

3.1 Target platforms

For the current discussion, we select two primary test devices, the Nokia 6630 and the N93. The 6630 is one of the first smart phones supporting 3G networks, and N93 one of the first with 3D hardware. Table 1 provides their essential specifications [Nokia 2006]. Nokia does not release accurate information about the CPUs or the real amount of RAM. For the developer, especially the amount of available memory would be essential, for cache optimization. For the N93, the amount of dynamic memory is described as "up to 50MB". External tools reveal that available RAM is 6–10MB for the 6630, and 16–22MB for the N93. Both devices support OpenGL ES Common profile for rendering, but only N93 has 3D hardware.

Platform	Nokia 6630	Nokia N93
2G Networking	GPRS/EDGE	GPRS/EDGE
3G Networking	UMTS	UMTS
Display resolution	176x208	320x240
Colors	65 536	262 144
Operating System/UI	Symbian S60 v2	Symbian S60 v3
Available RAM	6–10MB	16–22MB
Graphics acceleration	No	Yes

Table 1: *The two 3G smart phones used in trials.*

3.2 Internet in mobile environment

To assert a realistic mobile network scheme, we discuss networking in the mobile environment. The protocol that defines the internet, the *Internet Protocol* (IP), provides worldwide routing of data between computers using unique IP addresses. IP is circuitless, independent of the underlying hardware, thereby facilitating implementations from *ethernet* cables to trained pigeons carrying messages.

Two common data transfer protocols over the IP are used via the *socket* interface: the *reliable stream transport service*, TCP (*transmission control protocol*) and the *user datagram protocol*, UDP. The fundamental difference is in reliability: received TCP packets are automatically acknowledged, so the application using TCP to transmit data can always rely on the transmission. For a programmer, this is a major advantage: a TCP socket acts as a simple, reliable stream into which data can be poured, independently of the underlying reliability assurance mechanisms.

To increase network utilization, TCP uses the *sliding window* paradigm: several packets can be sent consecutively before receiving any acknowledgements. To avoid congestion, TCP applies the *slow start* algorithm: data rate is progressively increased as packets are delivered. The basic TCP header is 20 bytes long, so any TCP transmission should use large payload to minimize overhead.

UDP, even though developed after TCP, is simpler. The UDP header (8 bytes) only contains a checksum for the receiving end to assure its intactness. UDP packets are not automatically acknowledged. An application listening to UDP only receives correct packets, and cannot know if any packets are lost. If a system wishes to achieve reliability, the application must send back acknowledgements. While the UDP header is smaller than the TCP header, large packets are still encouraged to reduce overhead.

In mobile networks, latencies can be over 500ms, and the *sliding window* paradigm is extremely valuable. However, a poorly designed higher level network protocol can allow the *slow start* congestion control to stall network performance. For example, HTTP v1.0 applies a separate TCP connection for each object on a web page, causing them all to suffer from the *slow start* [Parkvall et al. 2003]. In addition, *packet loss* may cause severe connection disgradation due to excessive automatic retransmissions.

Several high-level middleware solutions exist, where further abstraction is layered over UDP or TCP, to provide communication transparency to underlying programming languages, operating systems and networks. For example, CORBA maps objects defined with *interface definition language* to various languages. SOAP, *simple object access protocol*, is an XML-based protocol for exhanging messages, usually via HTTP 1.0. CORBA uses a binary protocol, while SOAP uses basically human-readable text. Due to the lengthy headers and envelopes, SOAP overhead is significant. Binary XML proposals such as the WAP Binary XML (WBXML) [W3C 1999] attempt to overcome this severe drawback.

3.3 GPRS/EDGE and 3G network performance

The underlying network technology is transparent to applications using the IP stack. There are certain settings that could be altered to increase network efficiency, such as increasing the TCP receive window size, but they are implementation dependent and not available from applications.

However, network efficiency can be improved by considering the connection characteristics. For example, in the presence of *packet loss* and high latencies, TCP tends to fall back to slow data rates, while UDP based protocols with optimized application level control could still be able to maintain reasonable transfer speeds.

The current common packet-switched wireless data transfer technologies intended for mobile phones, such as the *General Packet Radio Service* (GPRS) and the enhanced version *Enhanced Data rates for GSM Evolution* (EDGE) can theoretically provide 140.8–236.8kbit/s, depending on allocated time slots and coding methods. These speeds are seldom met. However, EDGE technology increases the transmission reliability by adding error correction data.

True 3G technologies, for example the *Universal Mobile Telecommunications System* (UMTS), should yield much higher data rates. Starting with a 384kbit/s, future UMTS systems could theoretically support up to 11Mbit/s rates. The next evolution step of UMTS, *High-Speed Downlink Packet Access* (HSDPA) is being deployed, with peak downlink rates of 1.8–3.6Mbit/s, although with heavy dependency on network load.

Our tests were performed on a UMTS network, with a maximum capacity of 384kbit/s. We measured packet loss probability by sending UDP packets 500 bytes long to the devices, which replied by retransmitting the same packet. This was iterated at least a 100 times per one measurement. The packets contained digital noise, generated at server side, to suppress any possible compression. We performed several measurements. These tests do not account for the roaming effects between neighboring cells, as we did not move during the tests. The results were encouraging. Usually, we experienced no packet loss, even for GPRS/EDGE. At worst, we detected two lost packets in a sequence of a 100 packets (tables 2 and 3).

The UDP round-trip-times were measured using the minimum packet size of one byte. For GPRS/EDGE, the delay varied from 400ms to 800ms. Occasionally, we witnessed very long round-trip times (the maximum being 2.7s), with only 2-3kB/s rates. For UMTS, the delay seemed to stay between 140ms and 150ms, to both directions, and with both devices (tables 2 and 3). However, 3G networks were sometimes unavailable indoors.

TCP maximum transfer rates were measured by pooring data to the TCP sockets. We made simple tests with 10kB, 100kB and 1MB to see if the TCP slow start has any effect on the rate of the 3G devices. The results can be seen in table 4. The first 10-100kB are transferred relatively slowly, but about 5 seconds after initiating the transmission (after 100kB), the maximum rate of about 40kB/s is achieved. As the theorerical maximum is 384kbit/s (48kB/s), the network performs quite well with TCP. The upload rate of the 6630 is less than half of the download rate, but in our case, it will not be saturated by map data requests (see section 6.2).

3.4 Networking strategy

The good and steady performance of the measured 3G network suggests using a TCP-based data transfer scheme in favour of a proprietary low-level UDP solution. The relatively high latency, 140ms, however, must be considered. A request-response protocol (remote procedure call, RPC) between a client and a server would not be efficient, unless an algorithm similar to *sliding window* in TCP is used. In addition, as the *slow start* effect is clearly visible, data should be transferred through a single TCP connection. The observed 40kB/s is not much in comparison to common 3D model sizes, and a progressive download scheme is justified.

Platform	Nokia N93	Nokia 6630
Technology	GPRS/EDGE	GPRS/EDGE
Packet loss (UDP 0.5kB)	<3%	<1%
UDP RTT	400–800ms	400–800ms
Max TCP speed downlink	13-15kB/s	10-14kB/s
Max TCP speed uplink	9-11kB/s	5-10kB/s
TCP RTT	400-800ms	400–800ms

Table 2: *Network statistics for a 2.5G (GPRS/EDGE) network in good conditions.*

Platform	Nokia N93	Nokia 6630
Technology	UMTS	UMTS
Packet loss (UDP 0.5kB)	<1%	<1%
UDP RTT	140–150ms	140–150ms
Max TCP speed downlink	40kB/s	40kB/s
Max TCP speed uplink	40kB/s	15kB/s
TCP RTT	140–150ms	140–150ms

Table 3: *Network statistics for a 3G (UMTS) network in good conditions.*

Of the currently available middleware solutions, most are too resource-consuming for mobile devices, and unnecessarily generic for a specialized application. Some solutions, such as SOAP, also produce excessive overhead. Of the available generic methods, the most potential is the binary XML [W3C 1999].

4 3D Engines and visibility

3D engines can apply several techniques to speed up rendering, which have an impact on a networking scheme. One common goal in 3D engines is to maintain *output sensitivity*, where only those objects that actually are visible to the user are rendered. In highly occluded environments, *potentially visible sets* (PVS) [Airey 1990], objects visible from each *view cell*, can be pre-calculated. In urban environments, a view below roof level is highly occluded and the PVS solution is efficient, typically culling over 90% of the scene. When the viewpoint is raised, PVS may cull only 50%.

A PVS solution provides a view-independent list of objects. The ones that fall outside the view frustum are culled away using *frustum culling*. In addition, objects that contribute very little to the scene may be culled away using *contribution culling* techniques. 3D engines often apply *backface culling* to avoid rasterization of surfaces that are not facing the user.

In a progressive download scheme, a PVS solution is of obvious help. Only those buildings that are visible need to be downloaded. At street level, downloading may be quite fast due to the small amount of visible objects. It may be argued that at sky, where a PVS solution only offers roughly 50% efficiency, the geometry could be easily culled using fast client-side backface culling, and a PVS solution would be unnecessary. However, in a networked scheme, these backfacing 50% of geometry and textures *need not be downloaded* in the first place. This is a substantial advantage. Frustum culling is not so potential method to minimize network traffic, as rotation may change the entire view in a fraction of a second.

Network	UMTS/6630	UMTS/N93
10kB upload	5–7kB/s	6–7kB/s
100kB upload	12–14kB/s	24–28kB/s
1000kB upload	13–15kB/s	39–41kB/s
10kB download	5–6kB/s	5–6kB/s
100kB download	22–24kB/s	22–24kB/s
1000kB download	39–41kB/s	39–41kB/s

Table 4: *The* slow start *of TCP. Each test starts by opening a new TCP connection.*

5 Navigation

With our low level networking capabilities resolved, we attack our problem from a user's perspective. Our 3D map should support navigation in all its aspects. As seen in section 4, a high performance would be expected if the viewpoint can be restricted to street level. However, this may not be appropriate for navigation tasks. As assessed by [Downs and Stea 1977], one of the primary design decisions for map making is the *purpose*. We investigate previous research on navigation to identify user requirements, and perform field experiments to verify the assumptions.

The classic model for spatial knowdledge is division to *landmark knowdlege*, *route knowledge* and *survey knowledge* [Siegel and White 1975; Thorndyke and Stasz 1980], which can be seen as a hierarchical learning model (for a child, [Holahan 1982]), where a user performs spatial sampling by moving in an environment. However, an extensive exposure to environment via observations from street level does not necessarily lead to an accurate mental model of spatial relationships [Chase 1983]. In addition, merely following a route does not necessarily develop this *cognitive map* either [Golledge et al. 1992]. Use of a secondary source, such as a map, helps navigators to directly observe spatial relations and acquire survey knowledge [Thorndyke and Hayes-Roth 1982].

[Downs and Stea 1977] divide navigation to four stages: 1) *initial orientation*, 2) *maneuvering*, 3) *maintaining orientation* and 4) *recognizing the target*. [Darken and Cevik 1999] further classify wayfinding tasks in virtual environments. When a goal is marked on a map, a navigator performs a *targeted search*. When only the target's approximate location is known, a *primed search* is performed. When the location is unknown, the navigator performs an exhaustive, *naïve* search in the entire environment. Finally, the navigator can simply roam about, conducting *exploration*.

When navigating in a large virtual world, one needs a large scale view to maintain a sense of the overall environment, but a small-scale view is required for *cue extraction*, to identify details that are used to match the two worlds [Oulasvirta et al. 2005]. This *scaling problem* is well known [Furnas 1986; Darken and Peterson 2002], and exists in both paper and electronic maps.

To research users' orientation strategies, we have performed a field study [Oulasvirta et al. 2005] using our 3D map system with locally stored data. Users were allowed freedom to move anywhere in the 3D space. We conducted tests for local orientation and wayfinding, providing an initial viewpoint at street level or at sky (looking down as a 2D map). The initial orientation angle was varied between tasks. Subjects considered local orientation tasks easy at street level, especially if the initial view was correctly oriented. On the other hand, wayfinding tests were more demanding. Targets were not marked, forcing the users to perform a primed search instead of a targeted search. Street names were not provided to focus on visual features. 8 subjects performed the tasks, of which 6 had previous experience of 3D maneuvering with 3D games.

Figure 2 presents typical cases of primed search and orientation in unrestricted 3D space. In figure 2 (left), the subject is physically located at (A) and asked to maneuver to (B). Initial view is at (1), in 90^{o}angle to the A–B route. The subject first descends and moves forward to (2) to realize his error (noticing a park), then returns to his physical location (3), where he performs orientation near street level, and finally proceeds over rooftops to destination (4). In the figure 2 (right), a subject at (A) is asked to find a landmark (B) in the 3D view. (B) lies off the screen with a 180^o initial angle difference, but is visible in the physical space. The subject ascends to sky for a better view (2), soon spots the target (3), and returns back (4). In total, we performed 26 successful wayfinding experiments, where we also observed pure *exploration* at sky level, where a subject simply browsed the 3D world before performing his task. We also conducted over 50 orientation experiments.

During navigation, users used all available cues for navigation: building shapes, façade textures, façade colors, statues etc. Façades that were not textured caused orientation problems. Statues were represented by billboards, which were recognized easily. The experiments demonstrated the importance of *scaling*. Local orientation at street level should be allowed, using sufficiently detailed models for cue matching. In addition, a view over large areas is required to allow larger scale orientation. In this case, landmarks should be viewable even from long distances. We found no evidence to support restriction of the viewpoint to street level only.

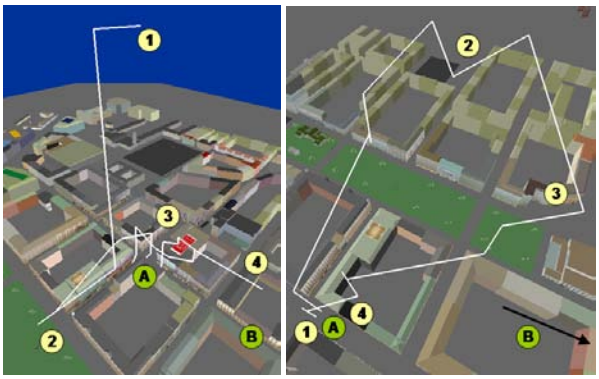

Figure 2: *3D navigation. Initial orientation and a primed search from A to B (left), and orientation at A to spot a landmark farther away at B (right).*

5.1 Progressive model download supporting navigation

The discussion and results above pose demanding requirements for a networking scheme, in addition to modeling and a 3D engine. [Schultz and Schumann 2001] discuss using *importance* in rendering, and [Coors 2002] applies that to retrieving map content from a database. We adapt and support these ideas with the purpose of assisting navigation, while maintaining high responsiveness and recognizability. We provide priority levels that can be assigned to 3D map objects. First, the status of a *major landmark* can be assigned to any classical landmarks, such as churches, railway stations or other unique and well-known buildings with high visibility. Secondly, all smaller but potentially salient cues such as statues and company logos can be marked as *minor landmarks*. Common structures receive no priority. In practice, importance is assigned to the model directly in the 3D view in *administrator* mode via menus.

Our progressive download scheme will priorize landmarks over common geometry. The major landmarks should be viewable from

long distances in the 3D map, so we also support landmark impostors, billboards, that are rendered even beyond the far plane of the view frustum. In addition, marker arrows can be assigned to point to any positions outside the current view.

6 System Architecture

Figure 3 provides a view to the system architecture. We separate *external* data exhange formats by *interfaces*, where we parse the subset of available features supported by our system, into a compact *internal* format. Our internal format supports PVS, triangle meshes with incremental LODs, LOD textures and model hierarchies. Common geometry features such as normals, texture coordinates, color etc. are included in the binary mesh description. 2D data is used for address queries and routing. We allow basically any external data formats that can be used within our 3D map system, including VRML, X3D, CityGML, MapInfo, Shape and various content databases. Currently, we have parsers for VRML, MapInfo, Shape and two XML based tourist databases.

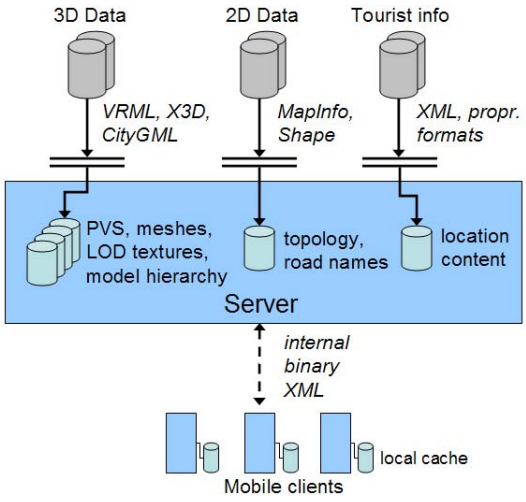

Figure 3: *System interfaces. External interfaces accept various formats, while internally the data is stored and transmitted in compact form.*

We pre-calculate PVS, applying contribution culling. This culling is performed in screen space with a simple threshold for minimal amount of pixels required for an object to be included. If the input models contain LOD levels, the PVS can directly choose appropriate meshes. The resulting PVS cells are difference-encoded to clusters. Textures are split to LODs.

Content databases are similarly parsed, and stored into a database independently of their original file format, but saving a common set of attributes. The database is accessed via normal SQL queries. We currently store PVS lists, geometry and textures into separate files instead of a content database for faster retrieval. Most of this data can remain resident in server's memory for lightspeed access, assuring the scalability of the system.

6.1 Asynchronous pipelined networking

Our low level network scheme is a request-response system (Remote Procedure Call, RPC), where the client performs requests, and server responds. However, all requests and responses are serialized into *send buffers*, which are asynchronously sent to the TCP socket. Requests and responses are also received asynchronously. More requests can be sent before all responses are received. The send buffers are organized as *tail queues* that can be rearranged prior to transmission. Several requests and responses are accumulated to the buffers before actually transmitting them (see figure 4).

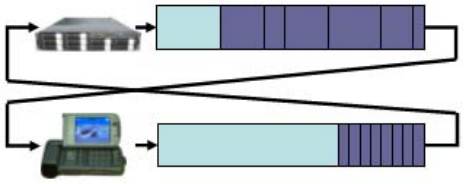

Figure 4: *Pipelined networking with send buffers. Buffer contents can be re-organized prior to actual transmission via TCP.*

Our asynchronous pipelined networking solution overcomes the overhead related to small packet sizes and removes the problems caused by round-trip latency in the common RPC scheme. The same connection is used as long as the application is running, so the *slow start* of TCP only affects the first transmissions. This scheme is able to exploit the full capacity of the network.

6.2 Network protocol

Our network protocol is described using XML. The protocol consists of *messages*, which can have attributes, specified as *fields* with the appropriate field type. The lenght of the *message* header is always 4 bytes, but the length of a *field* varies depending on the type, usually 1 or 4 bytes. Types can also be enumerated, such as the *result* field of each response. End tags are needed only for the description and are not included in the binary protocol. Binary data, for which exact field lengths cannot be defined in advance, is carried in *containers*. The container type, such as `uint32_t`, defines the container's atomic data unit. Previously defined structures, such as `texture_data`, can also be used as atomic container types. As an example of the protocol definition, table 5 presents the XML description for a *mesh* request and a response. The parse time of the binary protocol is negligible.

All requests perform a query for a single object, which is answered by a single object. For example, a mesh query would contain a 4 byte header (the enumerated message name), and the payload (mesh id) is another 4 bytes. In a case where the user raises from street level, he may instantly see perhaps 200-400 new meshes over the rooftops. When the client serializes the requests, the request buffer will be filled by 400x8 bytes = 3200 bytes. When flushed to the TCP socket, only a couple of full TCP packets are generated and sent in a fraction of a second. The response would consist of 400 response messages. With an average mesh size of 512 bytes, 200kB would need to be transferred. The overhead imposed by our protocol to the responses would be only $(4+4)/512 = 1.56\%$ (if the mesh id, 4 bytes, would be counted as part of the payload). Still, the network would be saturated for several seconds. Our buffering scheme is able to handle prioritized requests (such as landmark geometry) due to the ability of arbitrary management of the buffered *messages* until they are actually flushed to the TCP socket.

The benefits of our protocol, in contrast to other lightweight protocols, say, JSON [Json.org 2006], include very low overhead, lightspeed parsing, capability to send binary data such as textures, send buffer management, including control for maximizing TCP packet size, and asynchronous data transmission.

```
<message name="mesh_data_request">
   <field name="id" type="uint32_t"/>
</message>
<message name="mesh_data_response">
  <field name="result" type="result"/>
  <field name="id" type="uint32_t"/>
  <container name="data" type="uint8_t"/>
</message>
```

Table 5: *XML specification for a mesh request and response.*

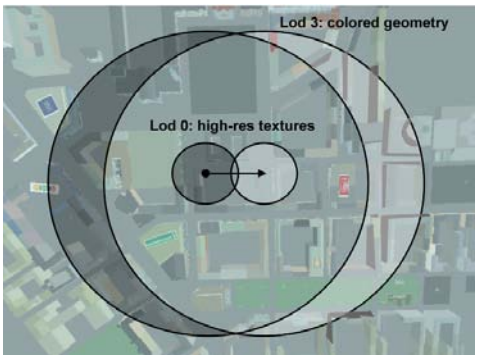

Figure 5: *Progressive model downloading. As the user moves, new visible geometry is requested. When the user stops, texture LODs are updated.*

6.3 Network scheme

To allow map data prioritization, we separate all map contents while maintaining their associations and hierarchies: 3D geometry, textures, location-based information and dynamic data can be transmitted in any order. After establishing a TCP connection, the client first requests a model hierarchy, which also provides landmark information. Based on the current location (for example, a GPS position), global landmark impostors are requested in distance order. Then, the current location is sent to receive a PVS cluster, and the current viewcell decoded to identify the currently visible meshes. If no geometry resides in memory or local caches, all visible meshes are requested, prioritizing landmark geometry. If the user does not move and all geometry is queried, textures are requested, in distance order.

As the user moves, our difference encoded PVS cell structure directly provides the newly visible mesh id's (the difference between the previous and the current cell), which are used to query the actual meshes. As geometry is received, it is inserted to the internal scene structure and rendered. Requests, responses and rendering are asynchronous. Again, textures are requested after all geometry requests are sent (independently of the received geometry), with priorities on global landmark textures and local landmark impostor textures. Figure 5 presents the movement of a user, with the highest and lowest LOD ranges with visibility differences.

The system allows application of any higher level pre-fetching scheme for situations where all visible objects are downloaded and the network is idle.

6.4 3D hardware considerations

Mobile 3D programming interfaces such as OpenGL ES and JSR-184 provide standardized 3D programming. From the viewpoint of such an API, a hardware-accelerated platform is no different from a software implementation. However, with hardware, rendering is pipelined. [Möller and Haines 1999] divide this pipeline to the three conceptual stages of *application, geometry* and *rasterizer* (given that a hardware geometry stage exists).

Rendering speed in a pipelined system is dependent on the slowest stage. Applications should manage the scene efficiently to feed the graphics hardware. Geometry should be arranged in hardware compatible forms, such as *vertex arrays*, using connected primitives, such as *triangle strips*. Visibility algorithms providing *output sensitivity* should be applied to minimize rasterization. In general, state changes in the graphics pipeline should be avoided to keep the pipeline fully occupied. [Möller and Haines 1999] provide several other optimization techniques that 3D programmers should be aware of. Our 3D engine takes full use of such methods, and is described in more detail in [Nurminen 2006].

6.5 Caching

In a mobile environment, probably the most limited resource is the available memory. When rendering a large, textured scene, the entire textured model will not be likely to fit in the memory. *Out-of-core* algorithms attack this problem. [Funkhouser et al. 1992] applies a memory management algorithm that computes the necessary LOD models to store in memory, estimating which will be rendered next. [Chim et al. 1998] assumes that the local cache storage can be exhausted, and sends the server the local cache contents and client's location to receive model updates. The local cache mechanism uses access scores to predict the access probability of models.

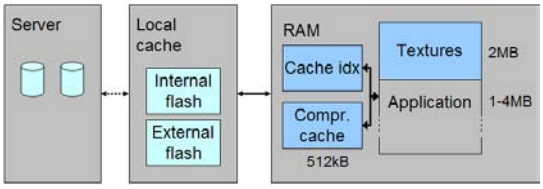

Figure 6: *3D data caching. The application first inspects a RAM cache for compressed objects, or the local cache indices, before sending a request to a server.*

Our implementation has been programmed to be memory-efficient. We use explicit memory management and are able to free memory in any situation. We pre-configure upper memory limits for application data, such as geometry and tourist data, and textures. In addition, any data originally in a compressed format can be held in a RAM cache. Currently, this cache holds JPG textures and PVS clusters.

Our memory management system applies a method similar to the *least recently used* algorithm. When the application memory limit is exceeded, objects outside the current PVS cell and, if necessary, outside the view frustum, are released, in reverse distance order. When texture memory is exceeded, textures outside the current PVS cell (and possibly view frustum) are released in LOD order. Anticipating file I/O to be inefficient for several small files in comparison to a few large files due to access latencies, we favour smaller objects to be stored in the RAM cache. As PVS clusters tend to be of similar size and relatively small due to pre-process restrictions, the RAM cache is first clensed of large textures.

The cache mechanism and memory usage are presented in figure 6. When new data is needed, the application first checks the RAM caches for compressed data, or the local storage. Data on local storage is organized into cache files, on internal flash memory, storage

card, internal hard drive or any such device (selectable by user). A 3D environment may consist of hundreds or thousands of individual objects. Mobile devices still use the FAT filesystem, which suffers from severe increase in access latencies when such amounts of small files are stored. We have frequently observed file access latencies of the order of seconds when distributing our data over thousands of separate files. To overcome this, we use cache files that hold 100 objects each, accompanied by corresponding index files. These indices also remain in memory. Therefore, the application knows immediately if an object is present in the local cache or if a network request must be performed. When new data arrives from a server, it is stored and the index file updated. The number of objects held in cache files could be optimized for each system (and storage device) by benchmarking the access latencies and seek times at user's discretion. However, as such tests tend to be relatively lengthy, we have chosen a value that limits the number of individual files of our current data set to a hundred or less, yielding fast access in all cases.

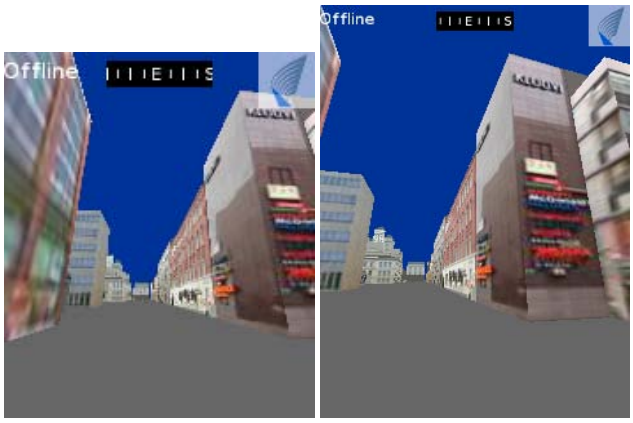

Figure 7: *A street-level scene rendered by Nokia 6630 in 78ms (left), and Nokia N93 in 5ms (right). The rendering quality is similar.*

7 Results

We have created an optimized network scheme for progressive 3D model download in 3G networks. We have also optimized a 3D map application for 3D hardware. The system has been implemented and installed on two 3G smart phones. The table 6 provides results regarding rendering speeds for typical navigation situations at street level, above rooftops and looking down from the sky. Nokia 6630 reaches interactive rates even without 3D hardware, but the 3D accelerated Nokia N93 performs extremely well, rendering scenes an order of magnitude faster, sometimes reaching 3-5ms (200-300fps) at street level (figure 7). In practice, we have been able to use over 500m visibility range, reaching over 60fps independently of the viewpoint. Figure 1 presents such a case, where the N93 renders over 200 façades and rooftops using over 50 unique textures.

The progressive download utilizes the 3G networks to the fullest, with minimal overhead. We reach the maximum measured data rates, 40kB/s, if we exclude the time spent for storing data onto local caches. Due to our compact protocol and buffered use of TCP, this is also the rate for actual model data transfer. The *slow start* of TCP allows the use of full network capacity in about 5-10 seconds after the beginning of data transfer.

Our case city model contains almost 200 individually textured buildings. The landmark-priotized scheme with visibility optimizations provides a navigable map in 15-30 seconds. The user may start exploring the map, or orientate with the local environment, maintaining responsiveness as the model is progressively transmitted. In practice, after a minute or two of exploration, most of the lightweight model geometry (3MB original VRML, 714kB binary) is downloaded, and several important buildings are textured (the complete model contains 3-9MB of JPG textures, depending on the maximum allowed resolution). In the case of street level navigation, geometry is downloaded without saturating the network, and buildings become textured quickly.

Platform	Nokia 6630	Nokia N93
Resolution	176x208	240x320
View distance	360m	360m
Contribution culling	16 pixels	16 pixels
Memory usage	4–8MB	4–8MB
Street	50–100ms	3–10ms
City overview	100–250ms	10–20ms
In sky, looking down	80–150ms	10–20ms

Table 6: *Approximate rendering speeds for a two 3G phones, with and without 3D hardware.*

Latencies related to local file I/O affect the overall performance. We have minimized this by caching compressed data in memory and using indexed file aggregates, but still each read and write blocks the application, reducing the effective download rate. In addition, the experienced framerate reduces from 60-100fps to 30-60fps. Allowing texture storage on the fly would further degrade the performance, causing noticeable split-second freezes. We currently load or store textures during the short breaks when the user is not maneuvering, given no geometry is pending. Using internal flash memory instead of external storage cards provides a considerably more pleasant experience. With texture loading turned off while moving, and geometry being loaded from network or internal flash, the abovementioned 60fps is usually maintained.

The size of our Symbian 3D map executable is approximately 260kB. The memory usage is 4–8MB, depending on the allowed cache sizes. We have launched the application with only 1MB, even though no textures were loaded, and less than 10 buildings could be fit into the memory.

8 Conclusions and future

We have developed an optimized, progressive urban 3D model download scheme for mobile devices in 3G networks, suited for any navigation situation. Smart buffering and a compact network protocol provide maximal throughput. The network protocol is easy to extend, and suits any RPC scheme, where minimal overhead, fast parsing and a small memory thumbprint are essential. The potential applications vary from any mobile client-server systems to smart home robot communications.

A 3D hardware compatible engine performs excellently on our test platform, maintaining good responsiviness during data transfer. Users have provided very positive initial feedback regarding the overall experience with the 3D map, with network speed sufficient for navigation. Based on further user experiments, we may still optimize the networking scheme, for example by devicing smart pre-fetching algorithms.

The system proves that the current level of technology has matured to the point where real-time rendered mobile 3D applications with progressive model download schemes can become commonplace. However, the economical viability of such a system depends on cost

efficiency related to 3D modeling, integrating external databases, and manual work needed for data association and metadata annotation. In addition, the system should scale for thousands of concurrent users. Our experiences are encouraging. Our case area, an entire city center, took 8 person months to model manually, of which 6 months were used in creating textures. Automatic or semiautomatic texture acquisition methods could significantly speed up this process. Modeling practices allowing optimization methods such as visibility pre-calculations should be further investigated in association with standardization work on formats (X3D and CityGML, for example). Our system provides metadata and location-based data annotation directly via the 3D interface in the *administrator* mode. This is fast, for example annotating a landmark requires only a few seconds. As 3D data may reside in the server's memory and the network protocol is suited for lightspeed parsing, the system scales well and is mainly limited by the local network speed. For the potential user, minimizing data transfer costs may be critical. Compact binary model representation together with the lightweight network protocol ensure that data transfers remain within reasonable limits.

9 Acknowledgements

This work has been supported by the EU Interreg IIIA project m-LOMA, and partially by the EU IST project ROBOSWARM. The author wishes to thank the programmers Ville Helin, Nikolaj Tatti, Heikki Vuolteenaho, Antti Kantee and Ilpo Ruotsalainen for their invaluable talents.

References

AIREY, J. M. 1990. *Increasing Update Rates in the Building Walkthrough System with Automatic Model-Space Subdivision and Potentially Visible Set Calculations*. PhD thesis, UNC Chapel Hill.

BLESCHMIED, H., ETZ, M., AND HAIST, J. 2005. Providing of dynamic three-dimensional city models in location-based services. In *MOBILE MAPS 2005 - Interactivity and Usability of Map-based Mobile Services. A workshop.*, Mobile HCI.

BRENNER, C., HAALA, N., AND FRITSCH, D. 2001. Towards fully automated 3d city model generation. In *Proc. Workshop Automatic Extraction of Man-Made Objects from Aerial and Space Images III*, International Society of Photogrammetry and Remote Sensing.

BURIGAT, S., AND CHITTARO, L. 2005. Location-aware visualization of vrml models in gps-based mobile guides. In *Web3D '05: Proceedings of the tenth international conference on 3D Web technology*, ACM Press, New York, NY, USA, 57–64.

CHASE, W. 1983. Spatial representations of taxi drivers. In *Acuisition of Symbolic Skills*, D. R. Rogers and J. A. Sloboda, Eds. Plenum, New York.

CHIM, J. H. P., GREEN, M., LAU, R. W. H., LEONG, H. V., AND SI, A. 1998. On caching and prefetching of virtual objects in distributed virtual environments. In *MULTIMEDIA '98: Proceedings of the sixth ACM international conference on Multimedia*, ACM Press, New York, NY, USA, 171–180.

COORS, V. 2001. 3d-gis in networking environments. In *International Workshop on 3D Cadastres*, FIG Commission 2.

COORS, V. 2002. Resource-adaptive interactive 3d maps. In *SMARTGRAPH '02: Proceedings of the 2nd international symposium on Smart graphics*, ACM Press, New York, NY, USA, 140–144.

DARKEN, R. P., AND CEVIK, H. 1999. Map usage in virtual environments: Orientation issues. In *Proceedings of IEEE Virtual Reality 99*, IEEE, 133–240.

DARKEN, R. P., AND PETERSON, B. 2002. Spatial orientation, wayfinding and representation. In *Handbook of Virtual Enviroment Technology*, K. M. Stanney, Ed. Lawrence Erlbaum Associates, Inc., ch. 28.

DÖLLNER, J., AND BUCHHOLZ, H. 2005. Continuous level-of-detail modeling of buildings in 3d city models. In *GIS '05: Proceedings of the 13th annual ACM international workshop on Geographic information systems*, ACM Press, New York, NY, USA, 173–181.

DOWNS, R. M., AND STEA, D. 1977. *Maps in Minds: Reflections on Cognitive Mapping*. Harper & Row, New York.

FUNKHOUSER, T. A., SÉQUIN, C. H., AND TELLER, S. J. 1992. Management of large amounts of data in interactive building walkthroughs. In *SI3D '92: Proceedings of the 1992 symposium on Interactive 3D graphics*, ACM Press, New York, NY, USA, 11–20.

FURNAS, G. 1986. Generalized fisheye views. In *SIGCHI 1986*, ACM SIGCHI, 16–23.

GOLLEDGE, R., GALE, N., PELLEGRINO, J., AND DOHERTY, S. 1992. Spatial knowledge acquisition by children: Route learning and relational distances. *Annals of the Association of the American Geographers 82*, 2, 223–244.

HESINA, G., AND SCHMALSTIEG, D. 1998. A network architecture for remote rendering. Tech. Rep. TR-186-2-98-02, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, Apr. human contact: technical-report@cg.tuwien.ac.at.

HOLAHAN, C. 1982. *Environmental Psychology*. Random House, Inc., New York.

JCP, 2005. Jsr 184: Mobile 3d graphics api for j2me. http://www.jcp.org/en/jsr/detail?id=184.

JSON.ORG, 2006. Introducing json. http://json.org.

KOLBE, T. H., GRÖGER, G., AND PLÜMER, L. 2005. Citygml - interoperable access to 3d city models. In *First International Symposium on Geo-Information for Disaster Management GI4DM*, Oosterom, Zlatanova, and Fendel, Eds. Springer Verlag, March.

KRAY, C., ELTING, C., LAAKSO, K., AND COORS, V. 2003. Presenting route instructions on mobile devices. In *IUI03*, ACM, 209–224.

MÖLLER, T., AND HAINES, E. 1999. *Real-time rendering*. A. K. Peters, Ltd., Natick, MA, USA.

NOKIA, 2006. Nokia n93 technical specifications. http://www.nokia.com.

NURMINEN, A. 2006. m-loma - a mobile 3d city map. In *Web3D '06: Proceedings of the eleventh international conference on 3D web technology*, ACM Press, New York, NY, USA, 7–18.

OULASVIRTA, A., NIVALA, A.-M., TIKKA, V., LIIKKANEN, L., AND NURMINEN, A. 2005. Understanding users' strategies with mobile maps. In *Mobile Maps 2005 - Interactivity and Usability of Map-based Mobile Services, a workshop*, Mobile HCI.

PARALLELGRAPHICS, 2005. Pocket cortona. http://www.parallelgraphics.com/products/cortonace.

PARISH, Y. I. H., AND MÜLLER, P. 2001. Procedural modeling of cities. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 301–308.

PARKVALL, S., ENGLUND, E., MALM, P., HEDBERG, T., PERSSON, M., AND PEISA, J. 2003. Wcdma evolved – high-speed packet-data services. Ericsson Review 2, Ericsson.

PASMAN, W., AND JANSEN, F. W. 2003. Comparing simplification and image-based techniques for 3d client-server rendering systems. *IEEE Transactions on Visualization and Computer Graphics 9*, 2, 226–240.

PRZYBILSKI, M., CAMPADELLO, S., AND SARIDAKIS, T. 2005. Mobile, on demand access of service-annotated 3d maps. In *Proceedings of the 23rd IASTED International Conference on SOFTWARE ENGINEERING*, IASTED, 448–452.

QUILLET, J.-C., THOMAS, G., GRANIER, X., GUITTON, P., AND MARVIE, J.-E. 2006. Using expressive rendering for remote visualization of large city models. In *Web3D '06: Proceedings of the eleventh international conference on 3D web technology*, ACM Press, New York, NY, USA.

RAKKOLAINEN, I., TIMMERHEID, J., AND VAINIO, T. 2001. A 3d city info for mobile users. *Computers and Graphics 25*, 4, 619–625.

ROYAN, J., BALTER, R., AND BOUVILLE, C. 2006. Hierarchical representation of virtual cities for progressive transmission over networks. In *3DPTV*.

SCHILLING, A., AND ZIPF, A. 2003. Generation of vrml city models for focus based tour animations: integration, modeling and presentation of heterogeneous geo-data sources. In *Web3D '03: Proceeding of the eighth international conference on 3D Web technology*, ACM Press, New York, NY, USA, 39–ff.

SCHMALSTIEG, D., AND GERVAUTZ, M. 1996. Demand-driven geometry transmission for distributed virtual environments. *Computer Graphics Forum 15*, 3, 421–431.

SCHULTZ, R., AND SCHUMANN, H. 2001. Importance Driven Rendering - Using Importance Information in the Rendering Process. In *Hamza M., Sarfraz M. (ed.): Computer Graphics and Imaging (CGIM 2001) Conference Proceedings*, 66–71.

SIEGEL, A., AND WHITE, S. 1975. The development of spatial representations of large-scale environments. In *Advances in Child Development and Behaviour*, H. Reese, Ed., vol. 10. Academic Press, New York, 9–55.

THORNDYKE, P., AND HAYES-ROTH, B. 1982. Differences in spatial knowledge acquired from maps and navigation. *Cognitive Psychology 14*, 560–589.

THORNDYKE, P., AND STASZ, C. 1980. Individual differences in procedures for knowledge acquisition from maps. *Cognitive Psychology 12*, 137–175.

W3C, 1999. Wap binary xml content format. http://www.w3.org/TR/wbxml.

WONKA, P., WIMMER, M., SILLION, F., AND RIBARSKY, W. 2003. Instant architecture. *ACM Trans. Graph. 22*, 3, 669–677.