Antti Nurminen. 2008. Mobile 3D city maps. IEEE Computer Graphics and Applications, volume 28, number 4, pages 20-31.

© 2008 IEEE

Reprinted with permission from IEEE.

This material is posted here with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of any of Helsinki University of Technology's products or services. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by writing to pubs-permissions@ieee.org.

By choosing to view this document, you agree to all provisions of the copyright laws protecting it.

Mobile 3D City Maps

Antti Nurminen ▪ *Helsinki Institute for Information Technology HIIT*

A research team created a living mobile 3D city by lightweight modeling and 3D optimizations, and implementing an efficient, scalable dynamic entity management solution. They designed a mobile 3D navigation interface and verified the system by field experiments. Based on the results, they discuss implications for future.

Spatial information has been conveyed in different forms throughout human history. Map-like media varied from stories with geographic depictions to figurative paintings—until knowledge and technology advanced to the point where appropriate survey and positioning methods came into use, facilitating metrically accurate maps. While early maps were heavily influenced by the surrounding culture and religion, and suffered from both intentional and unintentional errors in scale, they usually contained rather accurately drawn landmarks. The more scientific approach for mapmaking in the Renaissance yielded increasingly accurate maps, without sacrificing their beauty, and facilitating exquisite detail. As an example, in the early 17th century the cartographer family Blaeu created a complete city catalog, *Theater of Cities*, which depicted Dutch cities from a bird's-eye view in 3D, in a realistic manner (see Figure 1). Later, the artistic side of cartography diminished, as surveying became the dominant part. Simultaneously, the readability of maps was increased by simplification and abstraction.

By the end of the 20th century, technology had advanced to the point where computerized methods had revolutionized surveying and mapmaking practices. Now, the map's *purpose* drives the design, and possible errors in scale are intentional. These maps portray the environment in two dimensions, using a local rectangular coordinate system. Thematic maps lack geographic features, visualizing spatial variations of distributions of selected properties, such as population densities. Cadastral maps emphasize ownerships, road maps focus on the street topology, and tourist maps simplify the environment for navigational use, sometimes annotating the map with landmark figures. But the development has not stopped. Digital tools now let modern cartographers increase realism with unprecedented ease. For example, the US National Park Service, which produces maps for 300 million annual visitors, has continuously increased realism in pursuit of transforming map use to "become a matter of looking rather than reading."[1] In addition, maps are rendered on the fly on electronic devices, with various layers of spatial data. In car navigation systems, 2D views are being abandoned, as GPS-driven maps are now popularly rendered with the *perspective view*, in which the content is presented from the drives point of view.

In mobile devices, the computational power, memory, storage, and networking capabilities are increasing. They're being equipped with graphics hardware. For the first time, it might be possible to portray the environment with direct one-to-one mapping as 3D, real-time rendered mobile virtual environments. With wireless networking capabilities and GPS tracking, these environments could even be populated with real-world entities, such as people and vehicles. We might be at the verge of a new cartographic renaissance.

Our system

No mobile 3D map with all the envisioned features existed at the time we started our developments, despite a few attempts. In pursuit of the ultimate map, and to overcome the problems found in those systems, I instructed my team to build a 3D engine suited for navigation purposes by adapting suitable algorithms for maximal performance. For information on other systems (and the problems encountered when using them) see the "Related 3D Systems" sidebar.

Static scenes

Rendering static scenes is well established in computer graphics. The common optimization goal is to render only those primitives that contribute to the final view, using appropriate levels of detail. The trick is in reaching an output-sensitive situation without imposing severe overhead in culling processes.

John Airey's classic work defines a potentially visible set—a list of atomic objects or regions as seen from a viewpoint or a view cell.[2] Our system can pregenerate these lists for static scenes. And in other work, Daniel Cohen-Or and colleagues provide an extensive survey into various visibility algorithms for walk-through applications.[3] However, an algorithm suited for a mere walk-through might not suffice. Although the urban environment provides substantial culling potential at street level (see Figure 2), users might want to roam freely in the 3D space.

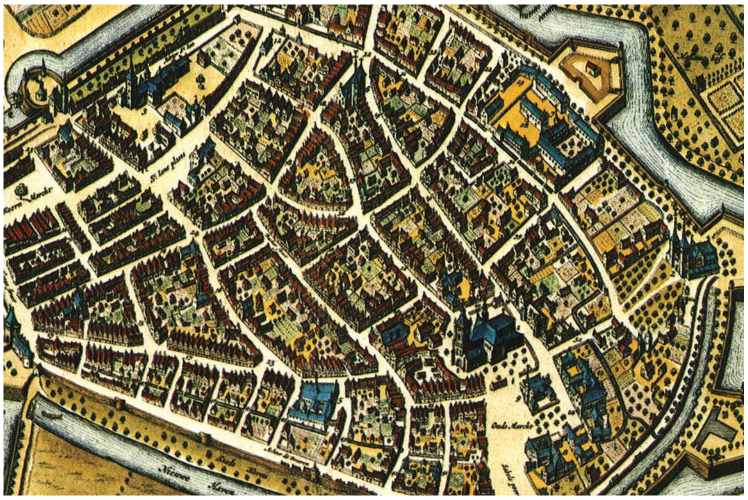

Figure 1. A close-up of an early 3D city map, the city of Arnhem from Theater of Cities, 1652, a city catalog by Joan Blaeu.

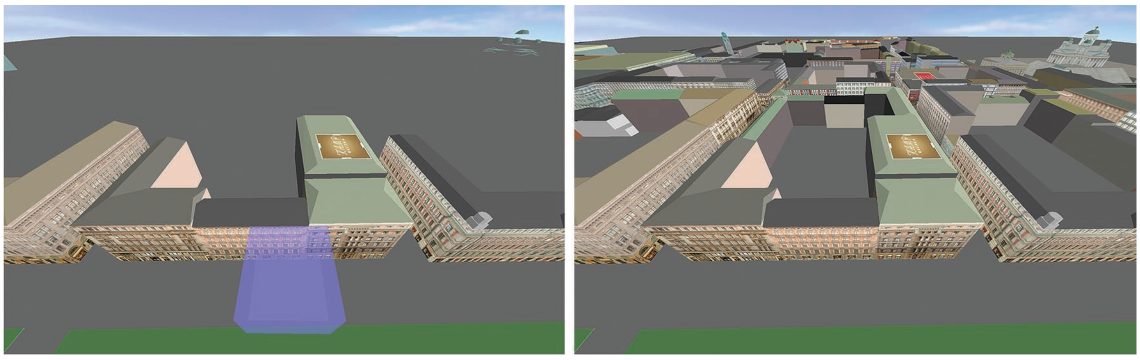
(a) (b)

Figure 2. The lure of urban walk-throughs at ground level. (a) When the viewpoint is contained within the blue view cell, only the closest buildings and some parts of major landmarks are visible (b). Above rooftops, most of the buildings are visible. At ground level, rendering could be more than an order of magnitude faster.

Related 3D systems

The VR hype of the 1990s, combined with the emergence of affordable consumer-level 3D graphics, stirred the imagination of many who envisioned the future as a place where the physical world would be embraced by the digital realm. While virtual instances of the real world had been created in different applications, Web-based systems were seen as the future of reality-based VR. Despite the subsequent fall of this enthusiasm, work on networked virtual environments and mobile navigation assistants continued, and as a result, user interfaces became increasingly graphical as the technology advanced. I review these systems to learn from the concepts and solutions in pursuit of a scalable, living, mobile virtual environment.

Navigation assistants, location-based systems, and augmented reality

The first generation of mobile electronic navigation assistants visualized the environment with traditional 2D raster maps that have the same disadvantages as paper maps, namely fixed scale and static content, which are further handicapped by a small screen size. A few years later, zoomable, real-time rendered vector graphics were introduced. Recently, the 2D perspective view has gained popularity, especially in car navigation systems, providing an egocentric view to the 2D map data. Navitime is one of the most total navigation aids, providing even preliminary 3D views with low-resolution textures for local orientation, expecting faster cell networks and "more comprehensive 3D data" to lead to better usability.[1] Navigation assistants commonly support audio and textual modalities.

Mobile-map-based systems can act as gateways to location-based data. GeoNotes (a bidirectional location based information system; http://geonotes.sics.se/) experimented with annotating the environment with user messages using either textual or 2D map interfaces with

Continured next page

Related 3D systems (cont.)

PDAs, pushing content updates to clients. Google Earth is currently the most known desktop-based 3D interface to spatial data, in which the idea of browsing is turned upside down: the Earth itself is the browser, onto which the Web provides content.

Augmented reality (AR)-based systems use the world as the background, onto which they render overlay annotations. An AR system's viewpoint is limited (for example, to the mobile phone camera) but can be complemented by map views. The main technical challenge in AR is not in 3D rendering but in accurate tracking. Recently, the Mobile Augmented Reality-Based Virtual Assistant (MARA) prototype demonstrated the possibilities of sensor-based tracking in a cell phone.

Networked virtual environments

In networked virtual environments (NVEs), multiple users and other dynamic entities share a common world. NVEs range from text-based games such as multiuser dungeons (MUDs) to immersive, collaborative virtual environments. The most popular form of NVEs is networked 3D gaming, including massive multiplayer online role-playing games, and first-person shooters (FPS).

NVEs face some challenges, such as scalability and consistency. In event-rich environments, nearly simultaneous but contradictory actions can occur (for example, two clients attempting to occupy the same spot), and a neutral partner, a server, is needed for decision making. In this case, the world simulation causes a computational bottleneck in the server. In FPS games, this architecture typically limits the number of simultaneous clients to 32 to 64. The first large-scale NVE, the Simulator Network (SIMNET), and the follow-up Naval Postgraduate School Networked Virtual Environment (NPSNET), addressed computational scalability by distributing the world simulation to "players" (units participating in the simulation) which were "honest" (a term used by NPSNET designers), meaning that each workstation was trusted in the simulation. However, despite dead-reckoning optimizations, a 10-Mbit/s network started to congest as the number of simultaneous players reached 300, each transmitting their state updates to all other players. To increase networking scalability, the NPSNET team introduced interest management to filter communication based on interest expressions—for example, geographically (location and radius) or functionally (such as a tank being interested in ground vehicles).[2] Later, several NVEs used spatial localization in message filtering. The RING (a client-server system for multiuser virtual environments) system was the first to apply visibility at message servers to reduce communications.[3]

Mobile 3D maps

Mobile 3D maps portray the real environment as a virtual one, similar to their desktop counterparts, but they run—or should run—in mobile devices. The first attempts at viewing 3D cities in mobile devices suffered severely from poor performance. In the 3D City Info project,[4] models intended for use with desktop workstations were simplified, but still the rendering rate of the realistically textured model was only one frame per 8 seconds on a PDA, rendered with the Pocket Cortona Virtual Reality Modeling Language (VRML) viewing library. Related field experiments were performed with still images on Web pages. In the TellMaris project, my colleagues and I built a lightweight textured 3D city model for mobile use. With a simple rectangular spatial subdivision, the guide software, which Nokia implemented with a proprietary rasterizer on a Nokia Communicator, achieved a few frames per second (fps) with rather coarse detail (texel size 1 to 2 m). TellMaris also took the first steps toward a mobile 3D navigation user interface.[5] The Lamp3D project researched information retrieval from 3D models, and by manually incorporating visibility information into a simplified VRML model, achieved a few fps with Pocket Cortona. After mobile 3D hardware arrived, later projects still faced resource-related problems. For example, the MobiX3D viewer, written with C++ and using the OpenGL ES API, and without a caching mechanism or memory management, suffered from slow file I/O and parsing in relation to X3D**,** an XML-based 3D file format and (low resolution) textures, to the extent that textures were turned off.

Developing mobile 3D maps is no longer hindered by the lack of 3D programming interfaces: the underlying rasterizer OpenGL ES is well supported on a range of devices, as is the Java-based scene graph renderer JSR-184. In addition, the VRML viewing library Pocket Cortona is available for MobileWindows, as is Direct3D Mobile.

References

1. M. Arikawa, S. Konomi, and K. Ohnishi, "Navitime: Supporting Pedestrian Navigation in the Real World," *IEEE Pervasive Computing*, vol. 6, no. 3, 2007, pp. 21–29.
2. M.R. Macedonia, "A Network Software Architecture for Large-Scale Virtual Environments," doctoral dissertation, Computer Science Dept., Naval Postgraduate School, June 1995.
3. T.A. Funkhouser, "RING: A Client-Server System for Multi-User Virtual Environments," *Proc. 1995 Symp. Interactive 3D Graphics* (SI3D 95), 1995, ACM Press, pp. 85-ff; DOI= http://doi.acm.org/10.1145/199404.199418.
4. I. Rakkolainen, J. Timmerheid, and T. Vainio, "A 3D City Info for Mobile Users," *Computers and Graphics*, vol. 25, no. 4, 2001, pp. 619–625.
5. C. Kray et al., Presenting Route Instructions on Mobile Devices," *Proc. 8th Int'l Conf. Intelligent User Interfaces* (IUI 03); ACM Press, 2003, pp. 117–124; DOI= http://doi.acm.org/10.1145/604045.604066

We anticipated that our main benefits from applying visibility would come from saving resources, and from prioritizing network traffic. For our implementation, we divided the space into rectangular view cells (similar to the blue box in Figure 2) and precomputed approximate visibilities by sampling from selected points within each cell. We expected the visibility lists of neighboring cells to be similar, so we packed them into 3D difference clusters, applying run-length encoding. At runtime, only the current view cell needs to have its visibility list open in memory. At preprocess time, we also applied contribution culling to remove barely visible details from the scene. We chose to verify this nonconservative design with field experiments, which we'll discuss later.

Dynamic entity management

The real world's dynamic entities are moving objects, such as GPS-tracked users, public transportation, bicycles, or other vehicles. Owing to their nature, we can't directly predetermine their visibilities. Classically, a runtime culling algorithm manages moving objects by first deleting them, then inserting them in to their new positions, and finally optimizing the scene for rendering. Rearranging the scene each frame causes a significant overhead. To avoid this, Oded Sudarsky and Craig Gotsman came up with the idea of replacing the object geometry with a stationary volume, which would approximate the object's potential path. Visibility can then be determined for this temporal bounding volume (TBV) for a duration of a validity period, which unfortunately depends on the viewpoint motion.[4]

In modern cities, buildings effectively limit any movement to urban canyons—namely, streets and sidewalks. In the visibility sense, we consider streets and crossings separately, and assume that if a street segment or a crossing is visible, so are any entities on them. Figure 3 presents our management system. We imported a street database to form a routable, topological network (an adjacency graph). For visibility predetermination, we created static virtual cells (called sweep volumes) to cover the street segments and crossings. For simulation purposes, we store the actual, possibly complex street geometry in a separate data structure.

A predetermined behavior model (similar to NPSNET) describes each entity class's motion, and further parameters define individual properties, such as speed. At runtime, entities are projected to the nearest street segments, which they occupy. In contrast to Sudarsky and Gotsman's validity periods for TBVs, we use validity periods for occupancies, which are independent of viewpoint motion. In short, we estimate how long a given entity will stay on a particular street segment. Larger entities, such as a tram, could occupy more than one segment.

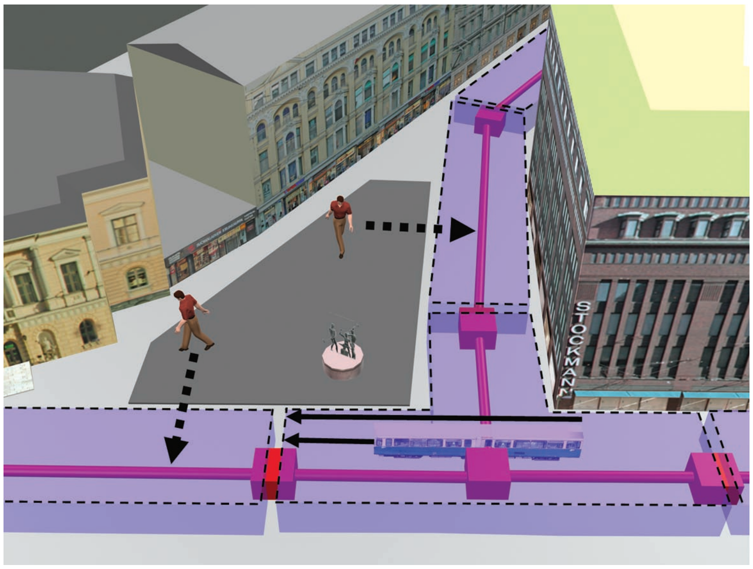

Figure 3. Dynamic entity management. Nodes (red boxes) connect street segments (red lines) to form a network. The segments are associated with static visibility cells (blue volumes). At runtime, entities are projected to the nearest segments, and their visibilities are resolved by referring to precomputed look-up tables. We use entity motion to estimate validity periods. In this example, the tram will occupy the crossing until its front reaches the next node, triggering a new occupancy and validity period determination.

The key to ensuring scalability in our system comes from two notions: first, exact positioning information comes from the clients, so we assume them to be honest. Second, parallel conflicting events don't take place in the physical world, so client states need not be validated at server side. Clients are also responsible for publishing their state updates. To view other entities, they subscribe to their locally visible segments. They can further express their interests in certain types of entities (public transportation or people), their buddies (a list of users), or single entities (the arriving bus that the user must catch). A server manages the entities by registering subscriptions and passing any incoming state updates to the subscribers.

Although our system is based on a simple street network, given accurate databases, it can be extended to a more elaborate system, including road lanes, sidewalks, and floating 2D areas such as parks. We'll look at some scalability issues in the section "System architecture and networking."

Location-based information

Our system acts as a navigation aid but also as a bidirectional portal to location-based information, where users can annotate the environment.

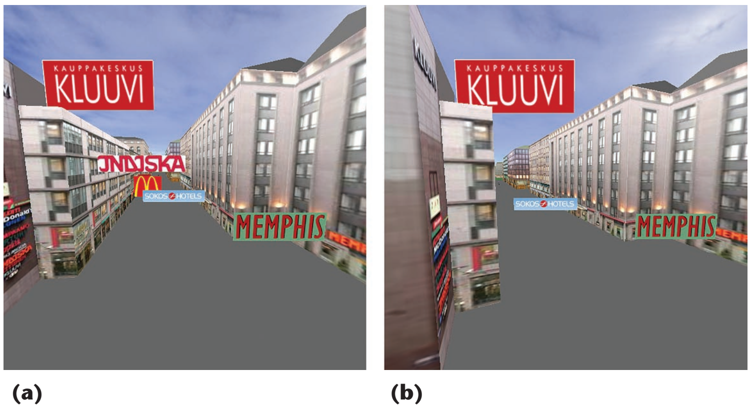

Figure 4. Approximate location-based information culling with facade visibility. Annotations "inherit" the visibilities of the surfaces to which they're attached. (a) Five billboards are visible, as all their parent surfaces are visible. (b) Two billboards have disappeared, as their parent facade is no longer visible.

Although we expect this data to be stationary, it can be temporary as well as provide dynamic content, such as near real-time updated sports results or even video clips. User messages and advertisements will have a relatively short lifetime, while company headquarters can remain in their place for decades. We don't assume such data will be integrated into the system during the modeling phase. Rather, we have to adapt to new information continuously.

We expect our data to be represented by relatively small graphical icons, or *annotation displays*. Our location-based information-culling scheme assumes that these displays are attached to something—to buildings, ground, or even entities—but not to open space. At the moment of creating a new annotation, it's accompanied by the target's identifier (a facade, a roof, an edge on the street's topology, or an entity). In all cases, visibilities have been predetermined: if the target is visible, so is the information associated with it. Figure 4 presents a situation in which two billboards at the left disappear as the facade they're attached to is no longer in view.

Modeling

The TellMaris city models (see the sidebar) reached a detail size of approximately 1–2 m. But in our project, we intended to achieve realism with more detailed textures and by applying complex geometry to major landmarks (such as to churches and railway stations). We use individual surfaces for atomic objects in visibility determination, and expect consistency from the surface normals. In addition, we can efficiently cull away the parts of the model that lie outside our current view if our model is hierarchical, discarding entire blocks and buildings with single bounding-box tests. With these requirements, we created a lightweight city model, consisting of textured facades and plain roofs, combined with buildings and blocks.

We created textures by digital photography and orthocorrected them with image-processing software. When considering ground surface details, we abandoned aerial photographs owing to the excessive presence of flattened 3D objects, such as buildings, cars, trees, and people. However, we created template textures for urban ground elements, and intended to obtain street coverage data for procedural ground-surface generation.

Urban environments are usually populated with small but unique landmarks, such as statues. Despite their size, they might be geometrically rather complex. Wouter Pasman and Frederik W. Jansen have evaluated methods for geometrical and image-based simplification for objects in networked environments.[5] They conclude that for small objects, billboards can indeed be used to replace geometry from a distance, where perceivable error is unnoticeable. However, in navigation use, we're interested in recognizability, not perceivable error. We chose to represent all statues and similar small unique objects with billboards, and later verify their recognizability with field experiments. Figure 5 presents one of the many statues within our case area. In addition to these minor landmarks, we similarly represent major landmarks with billboards at very long ranges.

System architecture and networking

The overall design goal in our system was efficiency at all levels. We wanted to create a functional system, especially one in which the mobile client doesn't need to perform any unnecessary computations. We also wanted our system to be extendable to cover very large areas.

Content management

Real-world databases are exchanged in various formats, which don't always conform to each other. To build our prototype, we used VRML models, although any properly structured boundary representation would be suitable. For our area, road data was available in MapInfo and Shape formats as collections of separate vectors. A public-transportation-schedule database was available in a proprietary format, but we found it to be inconsistent with road data. Our location-based information (a tourist database and a restaurant database) was provided in two formats, which had different address conventions.

Our experience is that a real 3D navigation assistant and portal to location-based information can't be based on a simple model-browsing paradigm. Instead, such a system should be prepared to input multiple data formats. We separated our internal system from the external heterogeneous databases by an interface layer, which processes the data sets to an efficient, wirelessly distributable format. We developed content management tools to semiautomatically associate content with the model.

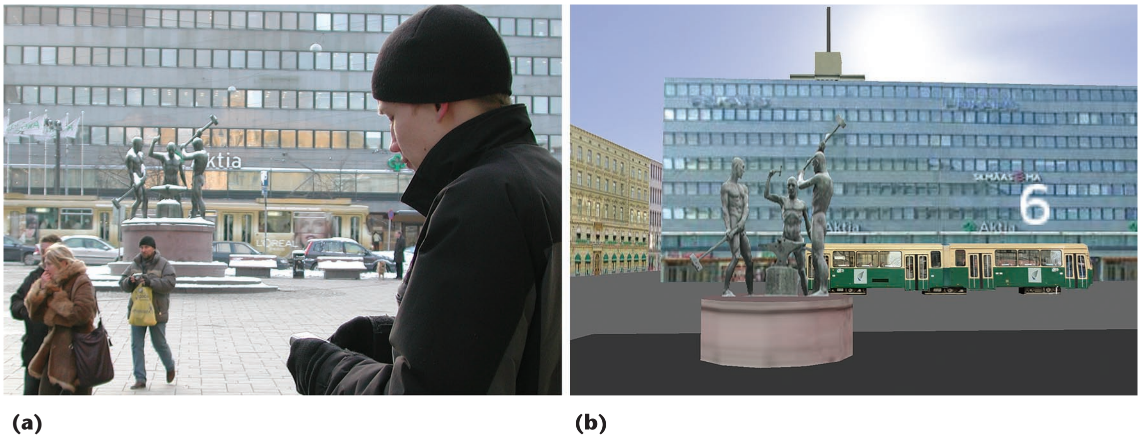

Figure 5. Replacing complex geometry with billboards. Users easily recognized real world statues (a) that were represented by billboards in the 3D model (b), independently of viewing angle.

Wireless communication

Current cellular networks still suffer from over 100 ms round-trip times, and data rates vary between 0–300 kilobytes per second, depending on location and network load. To distribute content and dynamic entity state updates wirelessly, an efficient mechanism is needed. We apply asynchronous TCP/IP communications, with a lightweight binary protocol, which we define with XML. Content queries can be prioritized—for example, by visibility and distance.

Caches and out-of-core rendering

The most limited resource of a mobile device is probably memory (RAM), while local storage capacities of 2 to 4 Gbytes can be expected from almost any device with smart card support. A mobile 3D map is a typical out-of-core rendering system, in which only a tiny subset of available data can fit into the memory. The efficiency of such a system depends on caches and data management. We use two cache levels (Figure 6): a compressed data cache, holding textures and visibility clusters, is kept in memory, while all content received from a server is stored onto local storage. A cache index is kept in memory to avoid unnecessary queries.

Scalability

Our system can be divided into static-content services and dynamic-entity services (Figure 7, next page). Static content can be distributed by any number of independent content servers. Dynamic-entity updates are distributed by fast entity servers, possibly via dead-reckoning proxies (we call them *entity proxies*). We ensured scalability by limiting the servers spatially and allowing them to subscribe to each other's data at the border. Clients choose their servers via the roaming and authentication server, which is a master server. Our 3D map service is now limited only by the mobile network capacity.

Results and performance

We implemented our client with C/C++ using the OpenGL ES 1.0 on various platforms, including Windows Mobile and Symbian S60. Our system features progressive wireless transmission of all content, including topological street data and 3D models. Clients can perform routing locally and track and simulate public transport and GPS-enabled buddies. Users can annotate the environment with messages, which act as discussion forums, and can query anything in the view. For example, buildings yield their associated services. We implemented a combined content/entity server with C++ on Linux, and a simple entity proxy with Python.

Our VRML model consists of approximately 200 textured buildings, with approximately 500 individual textures of relatively high resolution (10- to 20-cm texel size) covering a city center. Manual modeling took approximately 10 months for a single person to create, of which 8 months were spent in creating textures. We failed to find suitable coverage data for procedural template-based ground texturing, which accounts for the plainness of the streets in the figures.

The mobile client runs with software rasterization at 5 to 20 frames per second, depending on platform capabilities, viewpoint, and view range. On hardware-accelerated devices, such as Nokia N93

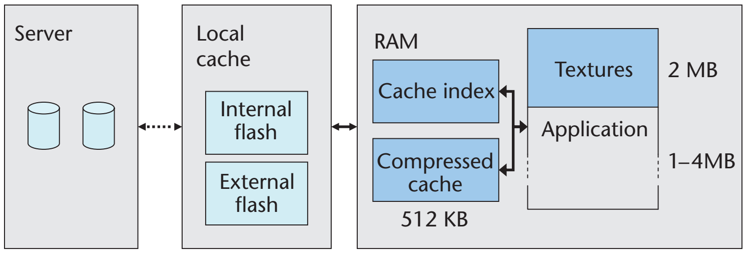

Figure 6. Out-of-core rendering and caches. Least recently used, compressed data is kept in RAM, favoring small data chunks to minimize memory card accesses. The memory ranges are configurable. Several existing systems don't use caching and memory management, and therefore have been forced to turn texturing off or can present only small models. This was a major problem with early systems in addition to rasterization speed; graphics hardware solved only rasterization, not scalability of resources.

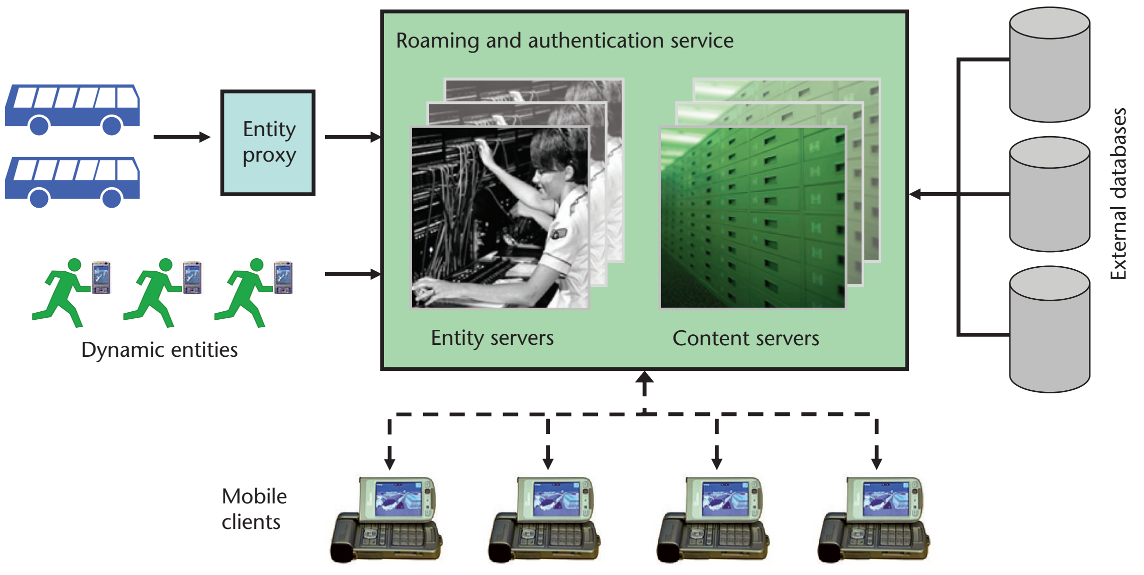

Figure 7. Static and dynamic 3D map services. Static content is preprocessed and distributed by independent, replicable servers. Dynamic entity servers act as switchboards. We can increase the number of servers to cover large areas, but neighboring servers must exchange information.

and N95 smart phones, rendering speed exceeds 30 fps with a 500 m view distance, displaying dozens of textured buildings. The size of the Symbian executable is less than 300 Kbytes. Although we have launched the system with only 1 Mbyte of allocated heap memory, we need 8 Mbytes to smoothly run the full-textured model with traffic simulation.

The system uses available wireless networks to the fullest. Figure 8 provides a plot of the network speed at server side while a user walks the path of Figure 9. In Figure 9, outside the railway station (1), the High-Speed Downlink Packet Access (HSDPA) connection is available between (2) and (3). The connection reduces to an older 3G cell phone technology and is lost at (3) due to temporary constructs atop the sidewalk. During download, the client remains interactive. Table 1 (next page) presents the experienced speed of content download in 3G networks. Current 3G cell networks are sufficient to support normal navigation, but prefetching, in which map content is downloaded before it's needed, might be advisable to account for signal losses.

We integrated our system with a local public-transportation tracking system via a SOAP network interface, which provides estimated times of arrival (ETAs) to bus and tram stops with a granularity of 1 minute. A dynamic entity proxy parses the SOAP stream, reduces it by 97 to 99.7 percent, and transmits ETA updates in a compact form to the dynamic entity simulation running in the mobile clients. We programmed a set of vehicle behaviors for collision avoidance and to accommodate for inaccuracies in route and positioning data. Our system has no problems managing the vehicles, although owing to the poor granularity, positioning isn't accurate enough to be useful. The local traffic transportation department is currently implementing a fast GPS-based positioning system. Figure 9 shows a common daytime situation.

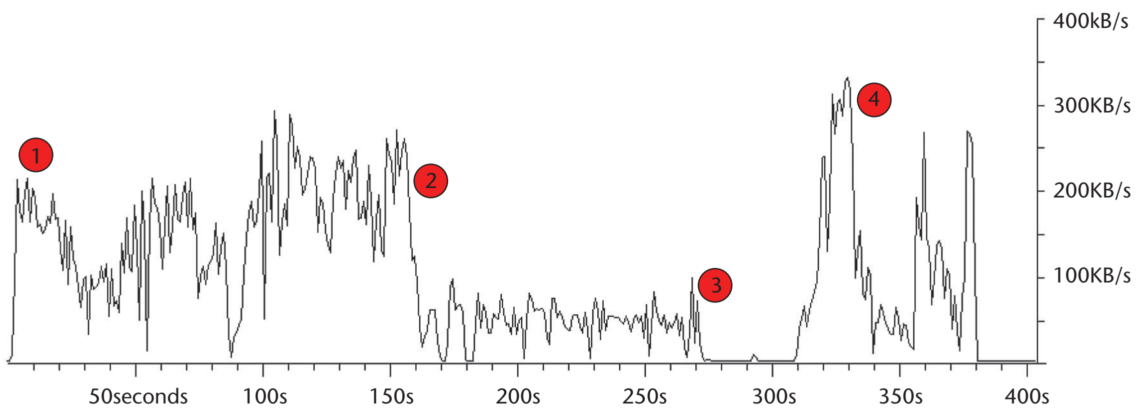

Figure 8. Cell network speed. The network connection varies between the High-Speed Downhill Pocket Access HSDPA (48–384KB/s) and older-generation cell networks, or is sometimes completely lost during the first 400 seconds of use. The speed differences account for varying signal strengths and network load as the user navigates, while a content server sends as much data as possible. Figure 9 shows the user's path.

Field experiments: A 3D navigation interface

We conducted various informal tests in the field, yielding technical results such as model download speed. However, we realized that no appropriate navigation experiments exist for mobile 3D maps. To better understand users' orientation and navigation strategies in the presence of a 3D interface, we set up two focused, exploratory field experiments, using a special build of the client, in which only maneuvering related functionalities were present. For the first test, we used a PDA with software rendering; for the second, a 3D hardware-accelerated mobile phone. Although the tests targeted cognitive aspects, we also received general qualitative feedback, validating our design decisions and providing implications for future 3D map developments.

Navigation tasks

We anticipated that we would observe a natural emergence of navigational usage patterns from the field experiments. Our goal was, and still is, to develop a mobile 3D interface based on observations from real situations. From existing literature, we acknowledged the different aspects of navigation, and prepared to provide a set of solutions to cover all of them.

Roger Downs and David Stea separate navigation into four stages: initial orientation, maneuvering, maintaining orientation, and recognizing the target.[6] Furthermore, Rudy Darken and Helsin Cevik classify way-finding tasks in virtual environments to a targeted search where a target is marked; a primed search in which only the approximate target location is known; a naive search, in which the navigator must exhaustively search the environment; and exploration, in which the environment is simply browsed without a clear goal.[7]

To maintain orientation, a navigator needs to constantly observe the environment to extract cues and match them with the map. The navigator and the map are challenged by the scaling problem: suitable cues might vary from major landmarks to small details, depending on the situation. We expected landmarks to play a critical role as anchors against which our subjects triangulate their position.

First field experiment: direct maneuvering

Our first experiment consisted of eight subjects, seven males and one female, who had lived in Helsinki on average 13.8 years. We gave them proximate pointing tasks, originating either in the 3D map or in the real world, and navigation tasks. We varied the map view between 3D and emulated 2D views (top-down view in 3D, without street names) and randomized initial orientation. We balanced the test by varying the task order and route direction. We collected performance measures, subjective workload ratings, interaction logs, video recordings, and verbal protocols. Finally, we synchronized the logs with video data.

We conducted the experiment with a direct mapping between controls and movement: subjects were able to rotate, move, tilt, and elevate the view. Figure 10 (next page) presents typical navigation behavior. Our subjects addressed the scaling issue by frequently elevating and descending the viewpoint, requiring adjustments to view orientation. They found free movement cumbersome, especially at ground level. They considered such micromaneuvering, where the focus moves from the task to steering the viewpoint, unnecessary and irritating. Other work details the first experiment.[8]

Second field experiment: More features available

Considering feedback from the first experiment, along with maneuvering constraints that Andrew Hanson and Eric Wernert suggested[9] for maximizing a view's orientation value, our second-generation interface introduced the following features and assisting functionalities:

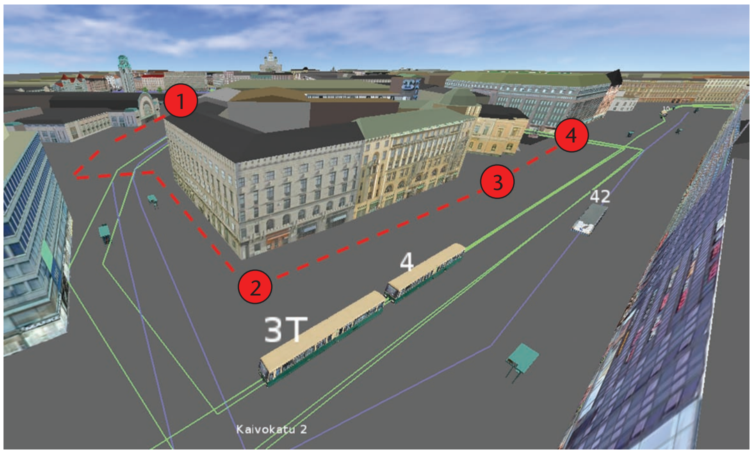

Figure 9. Path of the network speed test of Figure 8. We walked the presented path from (1) to (4) several times while continuously downloading data, experiencing significant variation in data rates (see Figure 8). The figure also presents our dynamic entity simulation with public transportation, suffering from inaccurate route data.

Table 1. 3D city download in 3G networks.

Time	Event	Data transferred
0–5 sec	Authentication	10 Kbyte
5–10 sec.	First buildings appear	10–100Kbyte
10–15 sec.	Most buildings present, some textured	100kByte–1Mbyte
15-20 sec.	All visible buildings present, nearest ones textured	200 Kbyte –2Mbyte
20 sec.–1 min.	All visible buildings textured	300kByte–10Mbyte
1min– and above	Possible pre-fetching of larger areas	

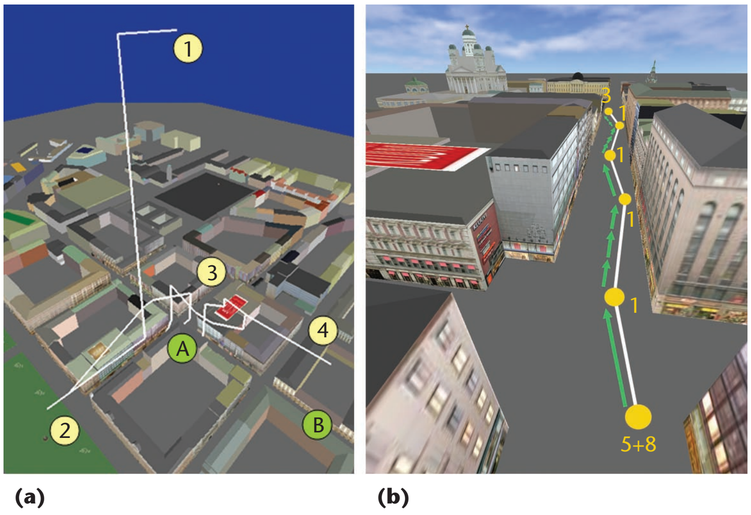

Figure 10. Navigation field experiments. (a) A subject at A performs a navigation task, with the initial viewpoint at sky level (1). The subject descends, makes an error in the initial direction estimate (2), returns toward his physical location for local orientation (3), and then starts navigating toward a target B at rooftop level (4). (b) With free maneuvering at street level, a subject needed 30 operations to orientate, maneuver along a street, and spot a target (the church at upper left).

- The tracks functionality limit maneuvering to the street network at ground level—thereby simplifying it—and shows street names.
- The View Landmark scripted action so that users can view landmarks easily.
- The Change Viewpoint scripted action so that users can change the viewpoint between street level and rooftop level, with automatic tilting of the view.
- When a user rotates the view at street level with Tracks on, the system automatically shifts the viewpoint backward for better view of the opposing façade.
- The system varies movement speed according to elevation level.

For the second experiment, we provided Orbiting as an alternate navigation mode. In this mode, view direction is fixed to a target, and viewpoint orbits around it. We assigned a 2D arrow pointing at the target to allow targeted search instead of primed search, but didn't yet provide routing, route markers, or GPS.

This experiment was built on similar navigation tasks as the first one. However, 3D navigation tasks started at street level, facing the target. Usually, the subject wasn't at that location, and self-location in 3D was necessary. We had 16 male subjects, all of whom were familiar with 3D games. We measured their visuospatial working memory span with a Corsi test, and spatial-rotation skills with an adaptation of the Manikin task. We trained the subjects in a fake city model by completing 15 practice tests while thinking aloud. The 2D map was a raster map professionally crafted for tourist use with street names and landmarks.

General qualitative feedback on the improved interface was quite positive. The simple constraints and improvements minimized unnecessary micromaneuvering and let subjects better focus on their tasks. Again we received comments for improvement, for example, View Landmark should always turn toward the user's favorite landmark. Full results from the second experiment are available in other work.[10]

3D Navigation: Further results and discussion

During the experiments, we witnessed most aspects of navigation. Each navigation task was initiated by local orientation, where subjects commonly searched for reference points, using cue scan, primed search, or egocentric alignment. Self-location was central, happening in 75 percent of the cases in the second experiment (both 2D and 3D). Primed search was troublesome in 3D, given that the subjects needed to remember the target position, while targeted search posed fewer problems. We also observed pure exploration. One subject decided to browse the world during his first navigation tasks. His performance increased until he claimed zero cognitive load for 3D tasks. In his verbal report, he explained that after he learned the controls and the map contents, he could concentrate solely on the task at hand.

We attempted to match navigation functionalities with navigation tasks. However, we were only partially successful. Change Viewpoint was the most used function (1.8 times per task on average), and Tracks and View Landmark were also used often (1.3 and 0.72 times per task). During the experiment, subjects increased their use of the Tracks function but reduced their use of other features, such as Orbiting, which was only tried but not really used for recognizing the target. Generally, once subjects learned a maneuvering routine, they didn't explore alternatives. We noticed a general tendency toward increased performance and reduced cognitive load for both map types. The results of the Corsi test indicate that spatial visual working memory span is associated with performance in 3D but not 2D.

In the first experiment, the emulated 2D map didn't provide street names or other easily recognizable cues. Overall, task completion was faster in 3D than in the emulated 2D. In the second experiment, the professional 2D map yielded better performance. This demonstrates the significance of representation—not all maps are equal. But why was navigating with 2D faster than with 3D? Despite the initial self-location in 3D due to our setup, there would still have been a difference. First, cue scan in 2D was simplified to finding street names, a good strategy as long as the names were visible in both the map and the environment, whereas in 3D, users had to choose from multiple, less efficient cues. With the 2D map, using street numbers as cues was an inferior strategy due to poor correspondence between the map and the physical world. Second, 3D still suffered from the frequent need for maneuvering the viewpoint. 3D efficiency improved with the use of Tracks. Third, our supposedly realistic view might not have offered small-scale cues for local orientation.

We avoided using GPS, routing, electronic compasses, and any other auxiliary guiding mechanisms to focus on the improved maneuvering features and to avoid technical issues related to poor GPS signal strengths. However, the results from the second test suggest that these functionalities are essential for 3D.

Field experiments: 3D engine and content validation

In addition to navigation-related results, we used the feedback from the two field experiments to validate technical issues, including 3D engine and model design decisions, providing implications for future 3D map developments.

Performance and battery life

We achieved 8–20 fps for PDA devices with software rendering for the first experiment. Subjects reported the performance sluggish but sufficient. For the second experiment, we achieved 30–60 fps for a smart phone with 3D hardware. Subjects reported the performance quite adequate.

System stability and battery life were put to the test during the mobile field experiments, which lasted from 1–2.5 hours per subject. We didn't experience any system crashes. With 3D hardware-enabled smart phones, battery life wasn't an issue even for consecutive experiments. However, with PDAs and software rendering, two hours of outdoor experimentation in near-zero-centigrade temperature took batteries (and subjects) to their limit. The reasonable power consumption is partially due to the episodic nature of use. For example, full-speed traffic simulation consumed batteries in a couple of hours, but when reduced to a few simulation steps per second, batteries lasted almost a day. Rendering updates for a GPS-driven, automatic viewpoint following can be similarly limited.

Realism, veridicality, and procedurality

To validate our building models, we didn't populate the model with external logos that could have acted as matchable cues. The results of the first experiment validated our initial model as follows:

- *Recognizability.* Most buildings, and all landmarks, including statues represented by billboards, were recognized easily.
- *Street-level contents*. Our facade textures couldn't replicate the street level owing to excessive occluders, such as cars and people, and weekly varying window contents. However, our subjects were able to neglect the street level without trouble.
- *Facade inaccuracies*. Our initial model lacked a few facade textures, and some had been copied from adjacent, similar facades. When subjects had learned to trust the model, these artifacts caused disorientation.
- *Color differences*. Perceived color differences between adjacent buildings caused disorientation in places where the colors were considered a dominant feature.
- *Ground.* Plain gray ground disoriented subjects at parks and other open spaces. Consequently, we created flat, colored models and added trees. However, our subjects weren't generally seeking cues from the ground. A couple of subjects requested crosswalks.

Once we attended to the minor problems with the model, we were then able to receive further feedback from our second experiment:

- *Texel accuracy.* Mostly sufficient, but in local orientation, where no landmarks were visible, the 10 to 20 cm texel accuracy was sometimes insufficient (see Figure 11 on next page).
- *Geometry*. Our model was geometrically lightweight. However, only features that had a particular meaning to our subjects, such as university stairs, or a specific object marking a meeting place were considered missing.
- *Roofs*. Our roofs were merely colored, not textured. The roofs weren't very visible from the physical viewpoint, but quite visible in the 3D view when subjects raised their viewpoint; a few subjects found this irritating. Roofs weren't considered important for navigation, but the labels we placed on top of a few selected landmark buildings were often used.
- *Topography*. When we built the initial 3D model, we had no topographic data available, so the resulting model was flat. During our first trial, subjects didn't notice that the topography was

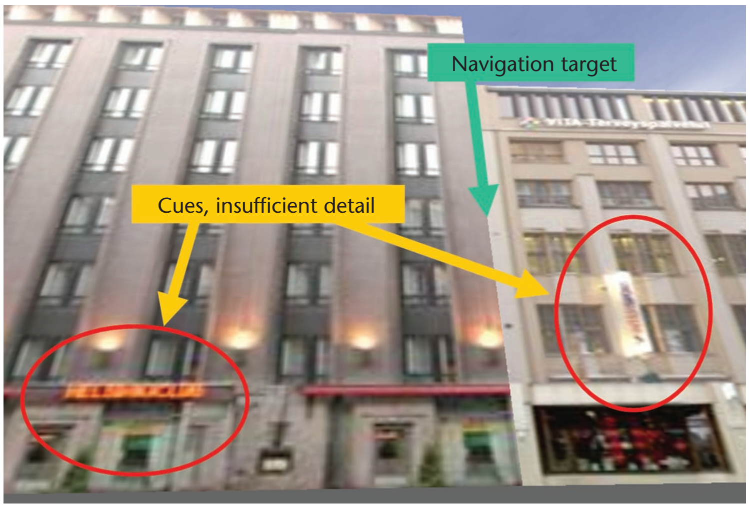

Figure 11. Model veridicality put to the test. When locating a target marked by an arrow, a subject notes that the two nearby logos, salient in the physical world, are hardly recognizable.

missing, so we left it alone. However, in the second experiment, we placed one navigation task on elevated terrain, featuring stairs, where subjects reported they found it difficult to match the 3D view with reality.

- *Free comments.* A few subjects reported that inanimate window reflections were distracting. Having a long view range was considered essential for orientation with major landmarks.

We had a goal of realism and tried to achieve it with detailed facade textures and statue billboards. While our model isn't photorealistic, we reached good recognizability for most navigation tasks. Our evidence hints that for local orientation, 3D models should be able to represent smaller than 10 to 20 cm details, or at least provide accurately positioned, highly recognizable local cues. The disorientation caused by slight differences between the model and the real world indicates that city models should be quite accurate and detailed. We wouldn't expect procedural buildings to be good candidates for navigation use, unless they were based on real samples. Regarding realistic visualization, we were encouraged by occasional comments, such as the following, provided by a subject who was initially against 3D navigation: "I recognized it [a statue] immediately, so ... the map was useless; I did not have to use it at all."

Small mobile display and details

The small screen size was no excuse for a less accurate model. When subjects needed more details, they moved the viewpoint to have a better look. However, in conditions with bright light, the display's dynamic range was reduced: only items with high contrast could be recognized easily. We expect the small screen size to afford fewer highlighted cues, markers, and other overlaid information than a larger screen, although our experiments weren't conclusive.

Walk-throughs

In most of the navigation tasks, subjects changed their level to suit their needs. Subjects performed local orientation at street or rooftop level, long-distance pointing from higher viewpoints, and navigation at rooftop levels. These reflect the common problem of scaling, present in all map applications, as mentioned earlier. There was no evidence to support limiting the viewpoint to the ground level.

Approximate-visibility determination

Our approximate-visibility scheme was generally successful, and the scene was rendered properly in almost all situations. During the first experiment, only one noticeable artifact was present: a small statue disappeared in a certain view cell. We tracked the problem to a missing cell sample point. The second experiment didn't suffer from noticeable visibility artifacts.

We didn't use company logo billboards during our experiments, but initial feedback from potential users who were shown the pictures in Figure 4 was encouraging: facade culling was considered a good feature.

Our goal was to re-create the real world on mobile devices, supporting progressive 3D model download, and populating the world with dynamic entities in a scalable manner. The presented work combines various optimization techniques, which we have validated. We demonstrated that such living mobile 3D maps are a reality: mobile networked real virtual environments can fit in a pocket (see Figure 12).

Our field experiments yielded valuable information regarding several assumptions and hypotheses related to 3D maps and common optimization scenarios. Furthermore, it's now clear that a graphically rich 3D visualization doesn't automatically imply intuitive navigation or good navigation performance. A navigation interface should provide useful cues promptly, without interfering with the actual navigation task. Efficient interaction methods play a key role. We have taken a few steps toward better usability, without involving "more comprehensive" 3D data or faster cell networks.

A realistic 3D representation has strengths not available to 2D: a 3D world can be more precise than a 2D representation. Indeed, we feel that the key benefit of a 3D representation is the higher potential accuracy for presenting spatial data, offering a better platform for multiple cues and small-scale features, which are best exploited in local orientation. This is in contrast to the recent trends toward procedural city modeling, where artificial buildings are created in pursuit of mainly imitating architectural styles. In addition, real 3D environments such as Google Earth are now being populated by 3D cities, but without clear guidelines for model quality. Our next step will be a model veridicality experiment in pursuit of definitive 3D model accuracy guidelines for use in mobile navigation, including illustrative and nonphotorealistic visualization alternatives.

Our user scenario has been one of a pedestrian, who can afford stopping at will to observe the environment. Even in this case, the use is sporadic, and our user can attend to the device only for short periods of time without being interrupted. For car navigation, the issue is further emphasized. We'll continue our interface development with a tight iterative approach: each new feature will be field tested without the burden of a full usability study, performing rigorous field experiments only to evaluate major stages of development. We'll then move on with the pursuit of the ultimate navigation interface, combining the best aspects of 2D, 3D, and audio representations.

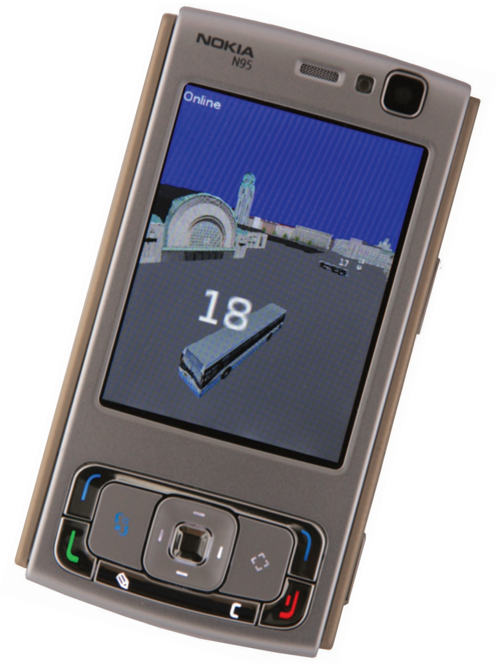

Figure 12. The modern, living mobile 3D at city running 60 fps on a smart phone.

Acknowledgments

I thank lead programmers Ville Helin and Nikolaj Tatti for their lightspeed skills and heroic efforts, Petteri Torvinen for professional 3D modeling, and the whole programming team at the Helsinki University of Technology. Special thanks goes to Antti Oulasvirta and Sara Estlander of the Helsinki Institute for Information Technology for designing and performing the field experiments. The EU Interreg IIIA project m-LOMA, City of Helsinki Public Transportation Department, and EU FP 6 IST project Roboswarm helped to support this work.

References

1. T. Patterson, "Getting Real: Reflecting on the New Look of National Park Service Maps," *Int'l Cartographic Assoc., 2002 Mountain Cartography Workshop*, Int'l Cartographic Assoc., 2002; www.mountaincartography.org/mt_hood/pdfs/patterson1.pdf.
2. J.M. Airey et al., "Towards Image Realism with Interactive Update Rates in Complex Virtual Building Environments," *Proc. 1990 Symp. Interactive 3D Graphics* (SI3D 90), ACM Press, 1990, pp. 41–50.
3. D. Cohen-Or et al., "A Survey of Visibility for Walkthrough Applications," *IEEE Trans. Visualization and Computer Graphics*, vol. 9, no. 3, 2003, pp. 412–431.
4. O. Sudarsky and C. Gotsman, "Output-Sensitive Rendering and Communication in Dynamic Virtual Environments," *Proc. ACM Symp. Virtual Reality Software and Technology* (VRST 97), ACM Press, 1997, pp. 217–223.
5. W. Pasman and F.W. Jansen, "Comparing Simplification and Image-Based Techniques for 3D Client-Server Rendering Systems," *IEEE Trans. Visualization and Computer Graphics*, vol. 9, no. 2, 2003, pp. 226–240.
6. R.M. Downs and D. Stea, *Maps in Minds: Reflections on Cognitive Mapping*, Harper & Row, 1977.
7. R.P. Darken and H. Cevik, "Map Usage in Virtual Environments: Orientation Issues," *Proc. IEEE Virtual Reality 99*, IEEE Press, 1999, pp. 133–240.
8. A. Oulasvirta, A. Nurminen, and A.-M. Nivala, *Interacting with 3D and 2D Mobile Maps: An Exploratory Study*, , tech. report 1, Helsinki Inst. for Information Technology, 2007.
9. A.J. Hanson and E.A. Wernert, "Constrained 3D Navigation with 2D Controllers," *Proc. IEEE Visualization*, IEEE CS Press, 1997, pp. 175–182.
10. A. Oulasvirta, S. Estlander, and A. Nurminen, "Embodied Interaction with a 3D versus 2D Mobile Map," *Personal and Ubiquitous Computing*, 2008, accepted for publication in the special issue on mobile spatial interaction.

Antti Nurminen *is a project manager in the Ubiquitous Interaction group of Helsinki Institute for Information Technology, Helsinki University of Technology, and University of Helsinki. His research interests include networked virtual environments and computer graphics in general. Nurminen received his MSc in engineering physics from the Helsinki University of Technology, and is currently working on his PhD on mobile 3D maps. Contact him at andy@cs.hut.fi.*