TKK Dissertations in Media Technology
Espoo 2009 TKK-ME-D-2

MOBILE THREE-DIMENSIONAL CITY MAPS

Antti Nurminen

TEKNILLINEN KORKEAKOULU
TEKNISKA HÖGSKOLAN
HELSINKI UNIVERSITY OF TECHNOLOGY
TECHNISCHE UNIVERSITÄT HELSINKI
UNIVERSITE DE TECHNOLOGIE D'HELSINKI

TKK Dissertations in Media Technology
Espoo 2009 TKK-ME-D-2

MOBILE THREE-DIMENSIONAL CITY MAPS

Antti Nurminen

Dissertation for the degree of Doctor of Science in Technology to be presented with due permission of the Department of Media Technology, for public examination and debate in Lecture Hall E at Helsinki University of Technology (Espoo, Finland) on the 10th of December, 2009, at 12 noon.

Helsinki University of Technology
Faculty of Information and Natural Sciences
Department of Media Technology

Teknillinen korkeakoulu
Informaatio- ja luonnontieteiden tiedekunta
Mediatekniikan laitos

Distribution:
Helsinki University of Technology
Faculty of Information and Natural Sciences
Department of Media Technology
P.O.Box 5400
FIN-02015 TKK
Finland
Tel. +358-9-451 2870
Fax. +358-9-451 5253
http://media.tkk.fi/

Available in PDF format at http://lib.tkk.fi/Diss/2009/9789522481931/

© Antti Nurminen

ISBN 978-952-248-192-4 (print)
ISBN 978-952-248-193-1 (online)
ISSN 1797-7096 (print)
ISSN 1797-710X (online)

Redfina
Espoo 2009

ABSTRACT

Author Antti Nurminen
Title Mobile Three-Dimensional City Maps

Maps are visual representations of environments and the objects within, depicting their spatial relations. They are mainly used in navigation, where they act as external information sources, supporting observation and decision making processes. Map design, or the art-science of cartography, has led to simplification of the environment, where the naturally three-dimensional environment has been abstracted to a two-dimensional representation, populated with simple geometrical shapes and symbols. However, abstract representation requires a map reading ability.

Modern technology has reached the level where maps can be expressed in digital form, having selectable, scalable, browsable and updatable content. Maps may no longer even be limited to two dimensions, nor to an abstract form. When a real world based virtual environment is created, a 3D map is born. Given a realistic representation, would the user no longer need to interpret the map, and be able to navigate in an inherently intuitive manner? To answer this question, one needs a mobile test platform. But can a 3D map, a resource hungry real virtual environment, exist on such resource limited devices?

This dissertation approaches the technical challenges posed by mobile 3D maps in a constructive manner, identifying the problems, developing solutions and providing answers by creating a functional system. The case focuses on urban environments. First, optimization methods for rendering large, static 3D city models are researched and a solution provided by combining visibility culling, level-of-detail management and out-of-core rendering, suited for mobile 3D maps. Then, the potential of mobile networking is addressed, developing efficient and scalable methods for progressive content downloading and dynamic entity management. Finally, a 3D navigation interface is developed for mobile devices, and the research validated with measurements and field experiments.

It is found that near realistic mobile 3D city maps can exist in current mobile phones, and the rendering rates are excellent in 3D hardware enabled devices. Such 3D maps can also be transferred and rendered on-the-fly sufficiently fast for navigation use over cellular networks. Real world entities such as pedestrians or public transportation can be tracked and presented in a scalable manner. Mobile 3D maps are useful for navigation, but their usability depends highly on interaction methods – the potentially intuitive representation does not imply, for example, faster navigation than with a professional 2D street map. In addition, the physical interface limits the usability.

UDC 004.92, 004.5, 528.9
Keywords computer graphics, virtual environments, 3D maps, mobile computing, mobile networks

TIIVISTELMÄ

Tekijä Antti Nurminen
Työn nimi Mobiilit Kolmiulotteiset Kaupunkikartat

Kartat ovat ympäristöjen ja ympäristön kohteiden visuaalisia esitysmuotoja, korostaen näiden tilallisia suhteita. Karttoja käytetään yleisimmin navigoinnissa, jossa ne toimivat ulkoisina tietolähteinä havainnoinnin ja päätöksenteon apuna. Kartografia on vienyt karttojen kehitystä yksinkertaistettuun suuntaan, jossa luonnollisesti kolmiulotteinen maailma on abstrahoitu kaksiulotteiseksi esitykseksi, jonka kohteita kuvataan geometrisillä muodoilla ja symboleilla. Abstrakti esitysmuoto vaatii kuitenkin kartanlukutaitoa.

Nykytekniikalla kartat voidaan kuvata digitaalisesti, käyttäen valittavissa olevaa, skaalautuvaa, selattavaa ja päivitettävää sisältöä. Kartat eivät enää välttämättä ole rajoittuneita vain kahteen ulottuvuuteen, tai abstraktiin esitysmuotoon. Kun luodaan todelliseen maailmaan perustuva virtuaaliympäristö, syntyy 3D-kartta. Jos esitysmuoto olisi realistinen, tarvitsisiko käyttäjän enää tulkita karttaa, ja voisiko navigointi muuttua luontevammaksi? Tähän kysymykseen vastaaminen vaatii mobiilin testausalustan. Mutta voiko 3D-kartan, resursseja vaativan virtuaaliympäristön, toteuttaa resursseiltaan rajallisille mobiililaitteille?

Tämä väitöskirja lähestyy 3D-karttoihin liittyviä teknisiä haasteita konstruktiivisesti, identifioiden ongelmat, kehittäen ratkaisut ja vastaamalla esitettyyn kysymykseen toiminnallisella järjestelmällä. Työssä keskitytään kaupunkiympäristöihin. Aluksi tutkitaan laajojen 3D-kaupunkimallien kuvanmuodostuksen optimointimenetelmiä ja haetaan ratkaisumalli yhdistämällä näkyvyystarkasteluihin, yksityiskohtien hallintaan ja grafiikkaytimille liian laajojen mallien käsittelyyn keskittyviä algoritmeja sovitettuna mobiileille 3D-kartoille. Seuraavaksi tutkitaan mobiilien tietoverkkojen mahdollistamaa potentiaalia, kehittäen tehokkaita ja skaalautuvia menetelmiä progressiiviselle kartta-aineiston siirtämiselle ja dynaamisten olioiden hallinnalle. Lopuksi mobiililaitteille kehitetään 3D-navigointikäyttöliittymä, ja tehty tutkimus validoidaan mittauksin ja kenttäkokein.

Tuloksena osoitetaan lähes realististen 3D-kaupunkikarttojen toimivan nykyisillä kännyköillä, ja 3D-laitetuella saavutetaan erinomainen piirtonopeus. Kännykkäverkot ovat myös riittäviä 3D-karttasisällön siirtämiseen käytön aikana. Todellisen maailman kohteita, kuten jalankulkijoita tai julkista liikennettä, voidaan seurata ja esittää skaalautuvasti. Mobiilit 3D-kartat soveltuvat navigointiin, mutta käytettävyys riippuu interaktiomenetelmistä – potentiaalisesti intuitiivinen esitysmuoto ei välttämättä johda 2D-tiekarttaa nopeampaan navigointiin. Lisäksi laitteiden fyysinen käyttöliittymä rajoittaa käytettävyyttä.

UDC 004.92, 004.5, 528.9
Avainsanat tietokonegrafiikka, virtuaaliympäristöt, 3D-kartat, mobiili tietotekniikka, mobiilit tietoverkot

PREFACE

The main research leading to the present thesis was performed in the *m-LOMA* EU InterregIIIA project in the Laboratory of Industrial Information Technology (INIT) of Helsinki University of Technology, Finland (TKK). Research has continued within the FP6 IST EU project Roboswarm in the Laboratory of Telecommunications and Multimedia Software (TML) of TKK, and within the SA and FP7 IST EU projects 3DWIKI and Hydrosys in Helsinki Institute for Information Technology HIIT (of TKK and University of Helsinki).

I am in gratitude to professor Juha Tuominen of INIT laboratory for providing quite a number of interesting projects and challenges during the many years I worked under him. Great thanks goes also to professor Tapio Takala, my supervisor, for long cooperation, support and mentoring in computer graphics.

The work on the m-LOMA project has been a group effort of a large number of very competent programmers and cognitive scientists. My role has been one of *primus motor*, providing impetus and resources throughout the development, initializing the work, setting up the goals, designing the system and the algorithms, and pushing the research forward over a span of several years with talented programmers and designers. Over 20 persons have contributed to the development, including the lead programmer Ville Helin who created most of the m-LOMA code base, and Nikolaj Tatti and Matti Lehtonen, continuing Ville's work. Ilpo Ruotsalainen implemented the m-LOMA networking system, Heikki Vuolteenaho ported the platform to Symbian, Antti Kantee oversaw critical parts of the core development, Mikko Rasa created lightweight widgets and a public transportation simulator, and Petteri Torvinen created our 3D city model. I am in gratitude to all my developers. Antti Oulasvirta provided valuable insight to cognitive aspects while I worked on issues related to user interaction, setting up two major field experiments. Jouni Pekkanen introduced ideas throughout the process, up to the last second and beyond.

Thanks also to my pre-examiners, Dr. Kari Pulli and Prof. Rudolph P. Darken for constructive criticism and support. Special thanks goes to Kari for having the strenght of going through every single letter in the thesis draft, contributing to both major and minor issues.

The most important support for my efforts has come from my family; my wife Niina, my daughter Unna, and our cat Piippu. I need to thank my parents Veli and Leena as well, for evoking interest in sciences when I was young, and for the drive for constantly looking up for new things, be it in nature or in technology.

Otaniemi, Espoo, 20th November 2009

Antti Nurminen

CONTENTS

LIST OF PUBLICATIONS

This thesis summarizes the following articles and publications, referred to as [P1]–[P5]:

[P1] A. Nurminen. m-LOMA - a Mobile 3D City Map. In *Proceedings of the eleventh international conference on 3D web technology (Web3D '06)*, pp. 7–18, Columbia, Maryland, USA, 2006.

[P2] A. Nurminen. Mobile, Hardware-Accelerated Urban 3D Maps in 3G Networks. In *Proceedings of the twelft international conference on 3D web technology (Web3D '07)*, pp. 7–16, Perugia, Italy, 2007.

[P3] A. Nurminen. Managing dynamic entities in mobile, urban virtual environments. *Journal of WSCG*, 16(1–3), 2008.

[P4] A. Nurminen and A. Oulasvirta. Designing Interactions for Navigation in 3D Mobile Maps. In *Map-Based Mobile Services* (L.Meng, A.Zipf and S.Winter, eds.), Springer, 2008, ISBN 978-3-540-37109-0.

[P5] A. Nurminen. Mobile 3D City Maps. *IEEE Computer Graphics & Applications*, 28(4), pp. 20–31, 2008.

LIST OF ABBREVIATIONS

3D	Three-Dimensional
AOI	Area of Interest
CAD	Computer Aided Design
COLLADA	Collaborative Digital Asset format
CPU	Central Processing Unit
EDGE	Enhanced Data rates for GSM Evolution
FOV	Field of View
fps	Frames Per Second
FPS	First Person Shooter
FTP	File Transfer Protocol
GeoVE	Geographical Virtual Environment
GIS	Geographic Information System
GML	Geographic Markup Language
GPS	Global Positioning System
GPRS	General Radio Packet System
GPU	Graphics Processing Unit
HSDPA	High-Speed Downlink Packet Access
HTTP	Hyper Text Transfer Protocol
HVS	Hardly Visible Set
J2ME	Java 2 Platform, Micro Edition
LOD	Level of Detail
M3G	Mobile 3D Graphics Format
MTU	Maximum Transfer Unit
NVE	Networked Virtual Environment
OpenGL ES	Open Graphics Library for Embedded Systems
P2P	Peer-to-Peer
PC	Personal Computer
PDA	Personal Digital Assistant
PDU	Protocol Data Unit
SOAP	Simple Object Access Protocol (deprecated)
SSH	Secure Shell
TBV	Temporary Bounding Volume
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UMPC	Ultra-Mobile Personal Computer
UMTS	Universal Mobile Telecommunication System
VE	Virtual Environment
WLAN (Wi-Fi)	Wireless Local Area Network
VRML	Virtual Reality Markup Language
VRTP	Virtual Reality Transport Protocol
X3D	Extensible 3D Format
XML	Extensible Markup Language

1 INTRODUCTION

It is necessary to follow the most recent researches because of changes in the world over time – *Claudius Ptolemy*, 150CE.

In the 1990's, both researchers and the public experienced an era of *virtual reality hype*, a phenomenon that thrived within the concept of a *cyberspace*, a networked digital space. This *virtual environment* (VE), limited only by physics and rules programmed for it, would encompass its users immersively. It would be represented as a real-time rendered, realistic 3D world, where users were depicted by *avatars*, moving within it and communicating to each other with real world metaphors - walking and talking. These schemes originated from science fiction with novels such as **William Gibson**'s *Neuromancer* and **Neil Stephenson**'s *Snow Crash*.

With the advances in 3D graphics technology and connectivity provided by the blooming Internet, researchers around the world were trying their best to make these visions happen, quickly establishing *virtual reality* (VR), or *shared virtual environments* as a branch of science. In these developments, the Internet itself was to become a cyberspace with 3D browsing capabilities. Several companies, such as **Blaxxun**, were created to manifest the visions commercially. In order to replicate the real world with a virtual one, several projects took place where entire cities were 3D modeled. However, despite the high expectations, the web did not become three-dimensional, nor did any applications emerge, except 3D games, which became a major industry.

There is another field of research related to virtual reality, which hasn't encountered so much of hype, namely cartography. When a virtual environment represents the real world, it can be understood as a three-dimensional *map*. By the end of the 1990's, technology had advanced to a level where mobile devices with graphical interfaces, such as *personal digital assistants* (PDAs), became available. Researchers saw the possibilities of running the new virtual cities in those devices, supposing that as any map, a mobile real virtual environment would be most useful during travel, in the field. However, these first experiments faced severe technical obstacles, placing the hopes of better performance on future mobile hardware.

The present thesis addresses computer graphics and interaction related issues in mobile 3D maps. The main goal is to break the technological barrier encountered in the first mobile 3D map studies, and then to proceed towards the other possibilities offered by wireless networking and positioning technologies, including on demand progressive content downloading and scalable near-real time tracking of dynamic entities. While a mobile 3D map will not offer an immersive experiment, many of the challenges are similar to shared VEs. Despite the emphasis on 3D graphics and networking optimizations, we assert the background in the more general context of cartography.

1.1 Terminology

To ensure a coherent and unambiguous use of terminology within the thesis, we define a few central terms.

3D Maps

The term "3D map" is not well defined in literature, and may invoke various ideas depending on the audience's background. Within this thesis, we use the following definition for a 3D map (extended from publication P4):

> *A 3D map is a two- or three-dimensional visualization of a three-dimensional representation of a physical environment, emphasizing the three-dimensional characteristics of this environment, intended for navigational purposes.*

This definition involves a two-stage process, separating the internal representation (map data) from its actualization (rendering). It assumes a physical 3D model (or hologram), or a single projection onto a 2D surface such as a screen. An immersive, stereo projected visualization can be special case. It does not concern the implementation of a 3D representation, but the three-dimensionality implies a capability to define geometric primitives in a non-restricted three-dimensional space. For example, if the system can only represent a height map, where no surface can exist over any other surface, we do not consider the representation to be truly three-dimensional. A 3D representation can also be obtained and held in the mind of an artist.

The method of rendering a 3D map is not defined. It could be drawn or formed by an artist. The definition differentiates a 3D map from an arbitrary virtual environment by the requirement of representing a physical space.

The definition sets a perceptual measure for conveying a three-dimensional appearance, which may be difficult to operationalize in quantitative terms, but is deemed necessary by the author to avoid extreme cases, such as projections that flatten the three-dimensional map data with brute force, for example by omitting one coordinate axis.

Finally, a 3D map is mainly intended for navigational purposes. This is an extension to the definition given in publication P4, in order to separate a 3D map from the term *GeoVirtual Environment* (GeoVE), which is an application neutral term for 3D models based on the real world, often encountered in the context of *geographical information systems* (GIS). This definition does not imply that a 3D map could not be used for other purposes than navigation, nor that a 3D model could not be used as a map; it merely indicates a primary design goal, or a use case. Figure 1.1 presents a minor tourist attraction, a miniature model of the center of Cambridge, U.K., which could be claimed to be a 3D map, when used for navigation.

A 3D map may have additional properties, which can be used as attributes to further specify a 3D map system and its characteristics (see table 1.1, extended from publication 4). Our case involves an *electronic, navigable, interactive, realistic, real-time rendered, dynamic and mobile urban 3D map*. Furthermore, in the current context of computerized navigation assistants, a "mobile 3D map" is expected to be at least electronic, navigable,

Figure 1.1: A miniature of the city center of Cambridge, U.K., with tactile navigation-assisting information for the blind (such as the you-are-here marker). This minor tourist attraction could also be described as a plain, static, urban *immobile* 3D map.

interactive and real-time rendered, running on a PDA or smart phone[1].

There are other related systems, which may claim to be 3D maps, but where the representation of the environment is restricted or has no 3D components at all. For example, car navigation systems commonly support *perspective 2D projection*, which creates an illusion of three-dimensionality through the perspective view, while the actual data is purely two-dimensional. Similarly, some early graphical computer games with 3D appearance exist, where the environment is rendered in a three-dimensional manner, but the representation is based on rules or procedures. For example, the environments for the game *Doom* were collections of "sectors", defined by the surrounding vectors ("sidedefs"), accompanied by attributes such as a floor height, ceiling height, light level, a floor texture and a ceiling texture. *Doom* and other similar systems, where objects cannot occupy space on top of each other, have been described as being based on "2.5D engines". Visually, these environments can however convey the likeness of a 3D space.

Mobile systems are expected to be physically small, to fit in the pocket, and independent of external power sources. In this sense, a device embedded permanently in a car is therefore not considered mobile.

Virtual environments and mixed reality

Milgram and **Kishino** have defined *mixed reality* (MR) as a "subclass of virtual reality technologies that involve merging of real and virtual worlds" [100]. In their terminology, real environments, for example the world directly viewed by eyes, and completely immersive virtual reality environments, experienced through computer graphics systems, constitute two extremes of a *virtuality continuum* (see figure 1.2). When computer generated entities are graphically represented within a real view, the view becomes *augmented reality* (AR). Similarly, when real world entities are represented

[1] We define smart phones broadly as mobile phones providing a platform for developing and running applications.

Attribute	Explanation
Ideal, photo-realistic	The data set and its visualization exactly match the real world; a single image is indistinguishable from a photograph.
Realistic	Map data is visualized in a realistic manner, in an attempt to approach the ideal representation. Typically, this involves sampling the real environment by digitization.
Plain	Omits surface details. Surfaces typically single-colored.
Off-line rendered	Visualization is performed prior to use (paintings, animations ...).
On-line/real-time rendered	Visualization is performed on-the-fly instead of displaying pre-rendered animation sequences or images.
Remotely rendered	Rendering is performed in a server, from where the resulting images are transmitted to the map application.
Locally rendered	Rendering performed by the device running the map application.
Fixed-view	Provides a single fixed view to the environment.
Navigable	Allows a user to control the point of view.
Passive	A navigable, real-time rendered 3D map, essentially a 3D viewer, which does not offer navigation-assisting features or content queries.
Interactive	A navigable, real-time rendered 3D map, responding to users' queries and providing navigation-assisting features.
Static	Map contents remain essentially static during use. Does not relate to the type of rendering, or navigability.
Dynamic	Contains dynamic content and time dependent elements other than the virtual camera, such as positions of near real-time tracked users, public transportation, simulations and annotation updates.
Electronic	Emphasizes the computerized means in producing the 3D view instead of a drawing or painting.
Urban/outdoor/indoor	Description of the represented environment.
Mobile	Lightweight map, which does not require external power sources.
Transportable	A bulky map, too large to fit in a pocket. If electronic, runs solely on external power supply during operation. Can be embedded in a vehicle.
Immersive	A large stereo projection system, with separate views for each eye to achieve an illusion of being immersed in the environment.

Table 1.1: Possible attributes for a 3D map.

within a virtual environment (VE), this becomes *augmented virtuality* (AV).

Figure 1.2: Milgram and Kishino's *Virtuality Continuum* ranges from the real world to completely artificial environments [100].

Milgram and Kishino also foresee that "as technology progresses, it may eventually become less straightforward to perceive whether the primary world being experienced is in fact predominantly "real" or predominantly "virtual"". Both augmented reality and augmented virtuality technologies then fall within the general framework of mixed reality. 3D maps, representing real worlds, and potentially real world entities such as people and vehicles, therefore would belong to near middle of the virtuality continuum. However, as the subject of research in this thesis involves solely computer graphics generated imagery, our 3D maps can be more accurately described

as augmented virtuality – virtual environments portraying real entities.

Streaming
In the context of networked data transmission, one often encounters the term *streaming*, usually understood as a way to process or utilize incomplete or partial data immediately as received. However, in some interpretations, streamed data is expected to be time dependent, for example in movies or music, where it is essential to present the samples in a proper (timestamp-based) order, and with an appropriate frequency. With 3D data transmissions, time dependency is not critical, unless the data includes animated sequences. To avoid possible confusion, we avoid the term *streaming* and refer simply to data transmission, or progressive data transmission, to emphasize the capability to process and utilize incomplete data.

Interactivity
When authors deal with resource sparse computerized systems, one of the most interesting qualities is *interactivity*. In graphical systems, this is often expressed in update rates, such as frames per second (fps). Sometimes, this is described only subjectively, such as "The rendering performance was just sufficient in order to navigate in models" (from [139]). Literature provides several sometimes conflicting interpretations of the observed interactivity. **Airey** provided the following observations while working on increasing the update rates for building walkthroughs, describing the rendering speed as *updates per second* (ups) [7]:

- At one update per second or less, the system is painful to use. It is necessary to use an auxiliary two-dimensional floor-plan display, or *map view*, to navigate.
- As the update rate increases from one ups to around twenty ups, interactivity appears to increase rapidly (superlinearly) before leveling off. At around six ups, the virtual building illusion begins to work. It is possible to navigate with only the three-dimensional display or *scene view*.

As for *interactivity*, we apply this observation throughout this thesis – we assume that reaching 6fps provides interactivity, and rates exceeding 20fps no longer provide a significant improvement in user experience.

1.2 Historical perspectives

Throughout the human history, there has been a need for conveying spatial information. Representations of the environment have been sung in songs, told in stories, and visualized as paintings, engravings and drawings already in the ancient world, possibly even pre-dating any written language. This would be the case if the wall painting of *Çatal Höyük*, which has been carbon dated to 6100–6300BCE, actually depicted about 80 buildings and a mountain in the background (see figure 1.3) [97, 96]. Maps on clay tablets

have survived from Mesopotamia[2], and on papyri from Egypt[3]. The highest peak of ancient knowledge in cartography and geography was **Claudius Ptolemy**'s publication *Geōgraphikē Hyphēgēsis*, "A Guide for Drawing a World Map", or *Geography* in short, in about 150CE [129, 17]. In *Geography*, Ptolemy integrated his predecessors' works and accumulated map data of the time in a coherent manner, adding his own disciplined thinking in eight separate volumes. With *Geography*, Ptolemy shaped the course of cartography for the next 1500 years.

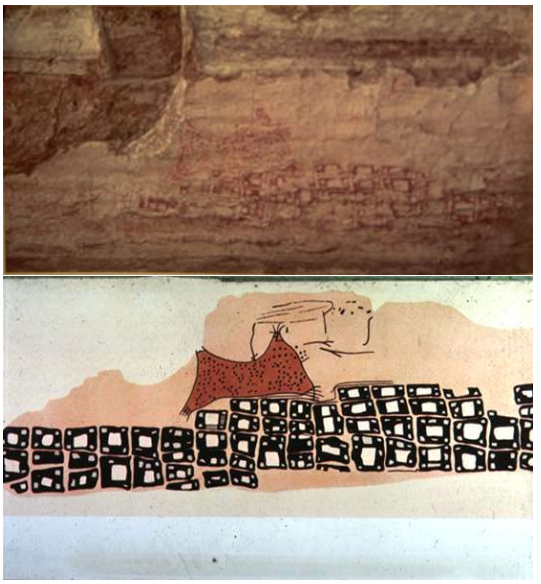

Figure 1.3: A candidate for the first known map, the wall painting of *Çatal Höyük*, from *ca.* 6200BCE (upper), and a clarifying drawing (lower).

Ptolemy defined two terms, *geography* and *chorography*. *Geography*, or *world cartography* referred to an "imitation through drawing of the entire known part of the world together with the things that are, broadly speaking, connected with it" [17], requiring skills in mathematics, such as spherical trigonometry. *Chorography*, or *regional cartography*, on the other hand, concerned regional and selective matters, with emphasis on detail and realism: "it should be the task of chorography to present together even the most minute features" ... "chorography deals above all with the qualities rather than the quantities of the things that it sets down; it attends everywhere to likeness, and not so much to proportional placements" [17]. Ptolemy stated that, in contrast to *geography*, the work on *chorography* requires mainly skills on arts.

Geography was a definite publication of principles and practices of cartography, including the use of *latitude* and *longitude* to define global positions, notes on the dynamic nature of the world causing a constant need for updating map data, and discussion on the *scaling problem* and *levels of de-*

[2]The oldest known Mesopotamian map is the *Gasur Map*, *c.* 2300BCE
[3]The oldest known Egyptian map is the *Turin Papyrus*, *c.*1160BCE.

tail, where the map designer must consider the purpose of the map and the size of the medium in regards of the type and amount of detail that should be presented. It is not certain that *Geography* contained drawn maps. However, regarding the representation of map data, the possible renderings were not of such importance, nor the accuracy of data, but the method of storing the spatial information. In *Geography*, the known world, the *oikouménē*, was described in textual form. Coastlines, long rivers and some mountain ranges were represented as curvilinear objects formed by connecting points of given coordinates, and cities, small islands, mountains and mouths of minor rivers as labels, named point-like objects. The bulk of *Geography* consists of these coordinate lists, which bear resemblance to current digital map formats. This textual encoding with global coordinates made it possible to create suitable maps on any projection or scale based on a common data set without losing detail or introducing cumulative errors, which would emerge from successive manual copying of drawn maps.

Our definition of a realistic 3D map follows Ptolemy's separation of data and its visualization: first, map data is obtained (or a representation formed in an artist's mind), including the minute details according to the chorographic principle (and including all three dimensions). Then, the representation is rendered, conveying the *likeness* of the environment.

The problem of medium and scale is well presented by examples from the Roman Empire, which depended much on spatial knowledge, surveying the known world and creating accurate maps [143, 83, 160]. Of the city maps, the most remarkable was *Forma Urbis Romae*, a city map depicting private and public buildings, including shops, streets and staircases, emphasizing important buildings by a larger size. The gargantuan map, 18x13m in size (to the scale of 1:240), was engraved on 151 slabs of marble between 203–211CE and attached to a wall. Only 10–15% of the map has survived in over 1000 fragments [83]. Romans also possessed drawn world maps, of which one nearly complete example exists, the manuscript copy called *Tabula Peutingeriana* ("Peutinger Table"), which is constructed on a single roll of parchment, 34cm wide, and 6.82m long, portraying the entire Roman road network [87]. While *Forma Urbis Romae* is an extreme case regarding the map material and size, arbitrary sized media was not available in general, and mapmakers had to cope with commonly available mediums, such as papyri or parchment scrolls. For a world map, the shape did not afford a spatially accurate representation. In order to create a map covering a large area (85.000km of roads [143]), the map designer had to use the available medium as efficiently as possible. Consequently, in the *Peutinger Table*, most countries and land masses are greatly distorted (see Fig. 1.4). As the purpose of the map was to record the topology of the Roman road network, spatial accuracy could be compromised, while the true purpose was well served; the lengths of the roads were accurately registered. Cities, other features and services were represented by symbols, while for main cities, special iconic decoration was applied, all well suited to the term *itineraria picta* (itineraries with figures), as described by **Vegetius** [160]. This term also includes a clue towards the true Roman mobile guides used by common travelers. The *itinerari*, itineraries, were simply lists of destinations along a route [137].

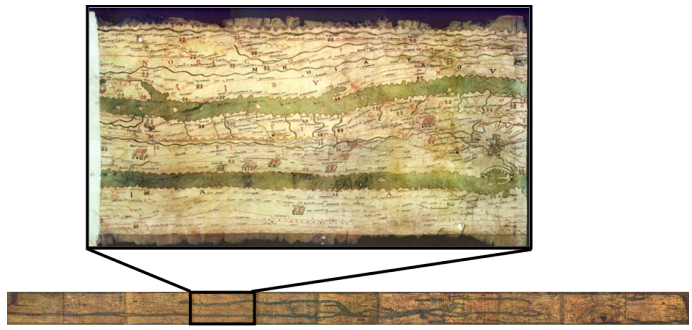

Figure 1.4: Fitting map data to given medium. The *Peutinger Table* covered the entire Roman road network on a single scroll of parchment.

During Middle Ages, cartography degraded, as did all sciences. In the so-called *tripartite*, or *T-O maps*, Ptolemy's North-up convention had been forgotten, and replaced by East-up convention, placing Jerusalem in the center of the world. Interest in maps, navigation and geography in general awakened in the 15th century, in the Age of Discovery, boosted by the introduction of movable-type printing in 1450 by **Gutenberg**. Ptolemy's *Geography* was considered *the* definite work on geography, and a multitude of *Ptolemian* maps were published during this era[4]. In the 17th century, mapmaking as an art form was at its peak, and maps were published in grandeur manner. For example, the *Toonneel der Steden van de Vereenighde Nederlanden* by **Joan Blaeu** in 1649–1651 contained over 200 exquisitely detailed town plans of the Netherlands in bird's-eye-view. Figure 1.5 presents one of these cities, the city of Arnhem. The realistic style is here in its extreme, where individual buildings have been painfully reproduced with as real – and three-dimensional – appearance as possible. *Atlases*[5], collections of maps, were heavy, bulky and expensive tomes, impractical to carry around. Subsequently, simpler, smaller and more affordable maps were produced for the commoners and travelers "to fit the pocket" [31].

In the 18th century, the emphasis of art in cartography gave way to a more scientific approach, as measurement technologies advanced and maps became increasingly accurate. However, it was not possible to establish a single, holistic representation of the environment. For example, the first accurate, triangulation based survey of an entire country, France, took place between 1718–1789 in many stages, but already in 1802 it was noticed that military needs require a different map, and another major survey was performed 1817–1880 [31]. In addition to Ptolemy's note on the dynamic nature of the world, this reflects the different needs of different users.

The modern ages have brought about technology to perform accurate spatial measurements, and digital tools to produce and present maps. This

[4] "Ptolemian" maps are often recognized by wind figures surrounding the map; however, such figures were not part of Ptolemy's map data. Ptolemy merely used wind names for depicting directions, which later evolved to the familiar compass rose shape.

[5] The term "Atlas" was first used by Mercator as the title for his world map collection published in 1585–1595 (*Atlas, Sive Cosmographicae Meditationes de Fabrica Mundi et Fabricati Figura*).

Figure 1.5: An early 3D city map, the city of Arnhem from *Theater of Cities*, 1652, by Joan Blaeu. (left) The entire map. (right) A close-up revealing the details.

new technology has already bloomed in the web, with a number of different map services, where maps are no longer bound to a static scale, but are rendered on the fly, containing various user selectable layers, presenting routes and location-based content. Furthermore, these map-based services have now become available to mobile devices as well [98]. Market analysts foresee exponential growth in the market, estimating 28 million GPS-equipped smart phone navigation software subscribers by the year 2012 in Europe, and 15 million in North America [74].

Several notions from Ptolemy still apply in the modern world, and further analogies can be registered. Ptolemy's use of a global coordinate system and textual storage for map data for producing different maps is valid, reflected in the modern digital map formats. Ptolemy even anticipated the dynamic nature of the map data. With electronic maps, this data could be updated on the fly. Brought further, maps could contain dynamic components, moving objects, which could be tracked and represented in near real time. The affordability of the medium provides an analogy between electronic maps and papyri, parchment or paper maps: the medium still poses limitations. While the *Peutinger Table* suffered from fixed height of the medium, mobile devices suffer from limited resources as well, be it screen size, memory, CPU or GPU speed, etc. The grandeur map tomes of the 17th century needed to be ported to mobile versions for travelers, with a tighter selection of content and scale. Similar porting has been done with modern electronic 2D navigation assistants. In an electronic 3D map, not just the coordinates but the appearance as well is described in digital form. Given a certain minimum amount of resources, porting a virtual environment to a mobile platform should be possible. However, the challenge involves further issues than mere scale and content selection.

1.3 Hypothesis and motivation for the research

Early maps were populated with intuitive figures, but abstraction has increased towards the modern age. Replacing figurative representation with an abstract one has been a compromise: the "easy accessibility" of figures

has given way to the compact symbols, and the skill requirements for map reading have increased. Abstract maps need to be annotated with a *legend*, explaining the various visual conventions being used [143]. Throughout the last 400 years, *pictorial maps* have persisted as a niché, usually representing the environment in three dimensions, an artistic rendering of the actual view. Despite the trend towards abstraction, traveler maps still contain pictorial components.

The advances in technology have made it possible to render maps in real time in mobile devices. While the commercially available map-based services mostly follow the traditional cartographic conventions, technology has matured to the level, where elaborate and computationally more complex systems could be devised. In this thesis, we challenge the level of abstraction in maps. Could a map become more intuitive, if it had more figurative components? What if the figurative representation were brought to its extreme, where a map represented the reality as it is, three-dimensionally, including all the visual detail? Would it no longer need interpretation at all? In Ptolemian terminology, such a system would belong to the field of *chorography*, depicting localized surroundings with "even the most minute features", and "attending to likeness". A key element separating this map from older 3D maps, such as the Dutch paintings from the 17th century, would be the interactivity (navigability) – the ability to freely move the viewpoint to any suitable position and orientation at will, providing the most suitable view for the task at hand.

In the beginning of the 21st century, first attempts at creating realistic 3D maps were made. Despite severe technical obstacles (very low rendering rates or lack of detail), positive observations were made that are well aligned with Ptolemy's chorographic principles (see Section 3.6). The initial hypothesis of intuitivity of realistic 3D maps seems to have support, and following this direction appears well motivated.

The main challenge for the present research is the fact that previous attempts at implementing realistic, mobile 3D maps faced technical obstacles, which were left unsolved. Either the level of detail or rendering speed was not sufficient to claim these systems to be navigable, realistic 3D maps. We believe that the problem of rendering a large 3D model can be mapped to one of rendering only the small subset of the model that contributes to the current view. This will involve transforming the model onto a data structure that allows fast on-the-fly selection of visible content with suitable level-of-detail. This would separate us from the traditional global removal of detail,where the entire map data set is scaled down to fit a given medium.

The use case of these early systems was one of a pedestrian, such as a tourist, navigating in an urban environment. Therefore, this will also be our scenario. Chapter 2 discusses related systems more closely.

1.4 Goals and research questions

This thesis aims at researching how far mobile 3D maps can go in terms of rendering realistic, progressively downloaded 3D models with dynamic content, and as a navigation aid for pedestrians. This can be divided to four subgoals, each addressing a separate research question.

The first subgoal is to identify the essential optimization methods for representing and rendering a static, realistic 3D model with minimal resources, suited for mobile devices. The fundamental research question is: *Can realistic urban 3D environments be rendered in current mobile devices at interactive rates?* Technically, the goal is to achieve interactive rendering rates for an urban environment consisting of at least a hundred buildings with a 10–20cm façade surface detail, increasing both the scale and detail over an order of magnitude compared to the first urban 3D map running on mobile phones, Nokia's *TellMaris Guide* (see Section 3.3). The system is considered successful if 6fps is reached, and fluent to use if 20fps is reached. The system is also required to run on existing mobile phones, without hardware and with only a few MB of memory. The research question is addressed in publications P1 and P2. P2 also provides results on the effect of 3D graphics hardware.

The second subgoal in this thesis is to research 3D model download over wireless cellular networks. The motivation is in allowing truly mobile use, where environments can be downloaded when needed. The research question is: *Can realistic 3D city maps be downloaded on the fly to mobile devices with sufficient speed for common navigation tasks, without hindering interactive use?* The technical goal is to be able to download and render a textured urban scene in less than 30 seconds in 3G networks. This question is addressed in publication P2.

The third subgoal is in designing dynamic environments, in search of optimal transmission and representation of temporary or dynamic information, including moving entities. The research question is: *Can a mobile 3D city map support dynamic content transmission and representation in near-real-time in scalable manner?* This is addressed in publications P3 and P5.

The last subgoal addresses the 3D user interface. The research question is: *Is a realistic mobile 3D map interface inherently intuitive?* 3D interfaces are developed in publication P4.

The focus of the research is in computer graphics generated imagery and 3D user interfaces depicting real environments and entities, a field that belongs to augmented virtuality. Therefore, this research does not cover nor address augmented reality or multimodal systems, nor does it attempt to make contributions to electronic 2D maps.

1.5 Research approach and methodology

The research questions defined in the previous section are deliberately set up in a simple and understandable manner, to allow interpretation, discussion and development. They will be answered constructively. First, hypotheses are issued. Then, a 3D map system is designed and implemented. The success of the platform is evaluated both quantitatively and qualitatively. Quantitative analysis is mainly applied to technical issues such as measuring rendering or networking speed with given technical constraints. However, we foresee several design issues – many of which are included in our research goals – that can only be verified by users. We consider the ability to perform focused navigation experiments in the field one of the main practical goals for the system. We intend to perform a sufficient amount of

tests to receive a reasonable, although not quantitative, amount of validation feedback. Setting up field experiments for definitive, quantitative answers to all our design issues would involve several person years of further research; this will be a future task.

Our development approach will not limit the system artificially. For example, it is not required to run the system within a web browser, or use web servers or the HTTP protocol for content delivery and dynamic entity management. Neither are possible 3D programming interfaces limited to any higher level model viewing libraries, which would prohibit the use of efficient optimization schemes. However, it is required that the presented algorithms are implementable on common mobile devices, such as PDAs running WindowsMobile or smart phones running Symbian.

1.6 Summary of research and contributions

The main contributions of the research are the following:

- Analysis and selection of computer graphics optimization methods that are suitable for mobile devices
- 3D model quality requirements for optimizable, geometrically lightweight but graphically rich 3D urban models
- A new perceptually based method for selecting the lowest level-of-detail of a texture (the dominant color)
- Demonstration of efficient and scalable out-of-core rendering of large and detailed city models in mobile devices
- Analysis and design of a networking solution for progressive downloading of 3D city models, utilizing the out-of-core rendering scheme
- Demonstration of networked progressive 3D city model download over cellular networks in mobile devices at speeds sufficient for navigation, without hindering interaction
- A novel visibility algorithm for rendering and managing dynamic entities in a scalable and efficient manner, supporting networked virtual environments and augmented reality in mobile devices
- A visibility inheritance based scheme for annotation culling
- A definition of 3D maps with potential specifying attributes
- A maneuvering classification depending on the level of available navigation freedom and user control
- User studies and field experiments as methodology for validating computer graphics algorithms and 3D model design
- Methodology for progressive development of 3D navigation user interfaces based on data gathered from field experiments

1.7 Organization of the thesis

The thesis is based on five publications, addressing the goals and research questions set for the thesis. First, related systems are discussed. Then, the thesis discusses the rendering of static 3D models, 3D model transmission with scalable management of dynamic entities, and, finally, 3D user interfaces.

2 ENABLING TECHNOLOGIES

Technologically, mobile 3D maps are very close to virtual environments. The interest in virtual environments was at its peak in the 1990's, at the era of *virtual hype*. With the emergence of consumer level graphics hardware, 3D games and Internet, many envisioned the future as a place where the physical world would be embraced by the digital realm. This enthusiasm led to the development of 3D file formats suited for virtual environments intended for use in the web, modeling projects where entire cities were re-created as 3D models, but not to any significant applications except games. The possibility of using the 3D models as mobile maps was still hindered by resource limitations of mobile hardware and lack of 3D programming interfaces for mobile platforms.

In the following, related technologies and systems are presented to provide a view to the problem field.

2.1 3D Model formats for urban environments

Virtual urban environments are created with 3D modeling software, although attempts to automatize the modeling process have been made [29, 140]. In more optimized environments, for example those intended for real time viewing, such as 3D games, the model format depends heavily on the properties of the underlying 3D engine and its space subdivision strategy [6]. *Procedural modeling* applies rules to construct models on the fly [118, 165].

To unify the heterogeneous and incompatible model formats, 3D modeling community has worked long for model standardization. Despite significant efforts, no single monolithic or extensible standard that would encompass all imaginable 3D visualization related features and that would facilitate every possible 3D modeling case has emerged, nor accepted as *de facto* format for representing virtual urban environments. Currently, there are several competing and potential standards, all applicable to the present case. Most of the available formats describe geometry as boundary representations (*B-reps*), and organize it in a *scene graph*, an acyclic directed graph. Surface detail (appearance) is supported from direct color samples (textures) to more complex procedural methods (shaders).

While the various 3D model standards share many properties, they have been created for slightly different purposes. Table 2.1 presents the main design goals of four common model formats. Both **VRML** and its modern successor, **X3D**, are intended for interactive use in networked environments as a final delivery mechanism, and include features for predefined animations, camera placement, etc. Despite the intention to use on a networked environment, VRML does not include a binary encoding nor a serialization scheme for efficient progressive transmissions. The X3D specification defines a compressed binary format [32, 75].

CityGML is based on the more abstract Geography Markup Language, **GML3**. It focuses on relations and semantics of various urban features, sup-

Format	Design goal
VRML97	*"The Virtual Reality Modeling Language (VRML) is a file format for describing interactive 3D objects and worlds. VRML is designed to be used on the Internet, intranets, and local client systems. VRML is also intended to be a universal interchange format for integrated 3D graphics and multimedia."[73]*
X3D	*"A standard that defines a royalty-free run-time system and delivery mechanism for real-time 3D content and applications running on a network."[32]*
CityGML	*"A common information model for the representation of 3D urban objects... realised as an open data model and XML-based format for the storage and exchange of virtual 3D city models." [82]*
COLLADA	*"COLLADA is a COLLAborative Design Activity that defines an XML-based schema to enable 3D authoring applications to freely exchange digital assets without loss of information, enabling multiple software packages to be combined into extremely powerful tool chains."* "We assume that most interactive applications will use COLLADA in the production pipeline, but not as a final delivery mechanism. For example, most games will use proprietary, size-optimized, streaming-friendly binary files. "[13]
M3G	*"The file format is provided as a compact and standardised way of populating a scene graph ... that complements the Mobile 3D Graphics API (M3G)."* "Several applications were identified for the Mobile 3D Graphics API, including games, map visualization, user interfaces, animated messages, product visualization, and screen savers." [1]

Table 2.1: 3D file format design goals

porting topological structures, making it useful for routing and data mining [82]. It is defined as an *information model for storage and exchange of 3D city models*.

COLLADA defines 3D assets, which can be utilized as content in application development, but is *not* a final delivery mechanism. In contrast to VRML and X3D, it is not intended for direct 3D viewing, but to be incorporated into production pipelines, where the final delivery format may be proprietary, suited and optimized for the particular case. X3D, CityGML and COLLADA are all based on XML structuring and markup conventions.

The **M3G** file format is a binary scene graph intended for use within mobile devices with the JSR-184 rendering interface, and includes support for serialization [131, 130].

Even though some of the formats are intended for direct viewing, such as VRML or X3D, it is not uncommon for developers to add more features to facilitate this in practice. For example, **Marvie** and **Bouatouch** developed VRML support for visibility relationships in urban environments [95], while **Mulloni** *et al.* split an X3D indoor model to parts and viewed them with manually constructed visibility lists [103]. **Reddy** *et al.* created support for multi-resolution data for VRML for visualization of large terrain databases with their *TerraVision II* viewer [135]. None of the discussed formats directly support these features.

2.2 Mobile 3D interfaces and mobile devices

The development of mobile 3D applications was long hindered by the lack of efficient mobile platforms, but even more by the lack of 3D rendering

interfaces. The *PocketCortona* VRML viewing library by **ParallelGraphics**, using a proprietary software rasterizer, was available in October 2000 to PDA devices running WindowsCE [117]. In addition, in early 2000's there were other proprietary solutions by **Fathammer**[1] (the *X-Forge* environment), **XenGames**, **Synergenix** (the *Mophun*), **HI Corporation** (the *Mascot Capsule*) and **In-Fusio** (*ExEn*) for a variety of platforms [168, 131, 130].

The first standardized 3D rendering interface, the *OpenGL ES 1.0* [22], saw light in August 2003. OpenGL ES, like OpenGL, is a low level rasterization API for native C/C++ programming languages. The OpenGL ES 1.0 is defined in two profiles, the *Common-Lite* and the *Common*. While the *Common* profile supports floating point arithmetics, the *Common-Lite* is a fixed-point profile "for extremely limited environments" [131], "to allow portable application behavior for applications written strictly to the minimum behavior" [22]. The OpenGL ES 1.X family is currently being superseded by OpenGL ES 2.0, which implements the *Common* profile only, and includes support for programmable vertex and pixel shading [104].

The first OpenGL ES implementations were software rasterizers, such as *Gerbera* by **Hybrid Graphics**[2] supporting WindowsCE and Symbian platforms. However, OpenGL ES was soon to be found natively in many mobile devices such as the **Nokia** *6630* (running Symbian S60), although without 3D hardware. **Dell's** *Axim X50v* PDA, running WindowsCE, was one of the first mobile devices with hardware support for OpenGL ES in Spring 2005. However, despite the VGA screen resolution (640x480) and its wealth of features, it only supported the OpenGL ES *Common-Lite* profile. Nokia's *N93* smart phone, entering the market in Summer 2006, was one of the first mobile devices to support OpenGL ES 1.1 *Common* profile. The 3D hardware in these first generation mobile 3D devices is based on the *PowerVR* technology by **Imagination Technologies**. Most mobile device manufacturers nowadays offer OpenGL ES hardware enabled models, and the market is expanding. Recently, **Apple**'s *iPhone* has become very popular, offering also a touch screen.

To provide an efficient 3D graphics API for the Java 2 Platform, Micro Edition ("J2ME"), the JSR-184 (M3G) was defined in parallel with OpenGL ES. M3G provides a direct support for the M3G scene graph format, which is part of the specification. M3G provides an interface to the underlying scene graph rendering engine, and an immediate mode, which should be compatible with OpenGL ES [131, 130]. JSR-184 is currently being superseded by JSR-297 (M3G 2.0), supporting programmable shaders [3].

The mobile J2ME 3D APIs are designed to support OpenGL ES (the M3D scene graph must be renderable with OpenGL ES functionality), but are in principle independent of the implementation of the underlying rasterization interface. Anyone implementing a JSR-184 is free to provide its own low-level OpenGL ES compatible rasterizer. However, in the presence of a 3D hardware accelerator, the manufacturer is most likely to provide a standardized rasterization API, such as OpenGL ES, and the use of a proprietary solution would not be able to benefit from the hardware. For example, **Superscape**'s implementation of JSR-184, the *Swerve ES* client,

[1] Fathammer is now part of Telcogames.

[2] Hybrid Graphics is now part of NVidia Corporation.

utilizes OpenGL ES for hardware rendering but a proprietary rasterizer for optimized software rendering [52].

Most mobile phone and PDA manufacturers, including Nokia, SonyEricsson, Motorola, Samsung, Dell, AMD, NVidia, Intel, Texas Instruments and IBM are members of the **Khronos Group**, supporting OpenGL ES. The most notable industrial body not participating in Khronos Group is **Microsoft**, which supports its proprietary *Direct3D Mobile*. M3G is being developed within the Java Community Process.

2.3 Mobile networking

Mobile networks provide data connections over cellular networks, where each radio cell is served by a base station. The implementation of the physical layer (the communication network) is invisible to an application developer utilizing the *Internet Protocol Suite* [28, 125]. Figure 2.1 presents the four Internet protocol layers, where a developer can venture with the higher level protocols, such as HTTP [19, 60], FTP [128] or SSH [166] of the *Application Layer*, or create proprietary application protocols directly over TCP [126] or UDP [127] of the *Transport Layer*.

The communication layer of cellular networks can utilize several available technologies, such as *General Packet Radio System* (GPRS), its update *Enhanced Data rates for GSM Evolution* (EDGE), or the more modern *Universal Mobile Telecommunication System* (UMTS) and its enhancements such as *High-Speed Downlink Packet Access* (HSDPA). The GPRS/EDGE belong to the "2G" family of cell technologies, and UMTS/HSDPA to "3G" technologies. EDGE and HSDPA are also called "2.5G" and "3.5G", respectively. While the maximum downlink speeds of the recent HSDPA installations reach 3.6Mbit/s, the characteristics of the connections vary greatly depending on the network load, connection quality and the capabilities of the mobile phones. Cellular network characteristics have been examined in publications P2 and P5.

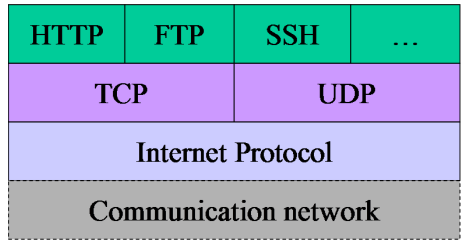

Figure 2.1: The Internet protocol stack. A developer can utilize existing higher level protocols, or create custom protocols using TCP and UDP.

3 ELECTRONIC MOBILE GUIDES AND 3D MAPS

This section provides a look into a selection of mobile guide projects and products, with emphasis on implementations applying 3D visualizations. The author of this thesis was involved with two of the projects discussed in this section, namely *TellMaris* and *Between* in the role of a project manager, responsible of technical issues and system design at **TKK**. These projects, especially *TellMaris*, had a significant influence on the work resulting in this thesis.

3.1 Early work

The evolution of pocket digital assistants with graphical user interfaces in the later half of the 1990s signaled an increasing interest in the possibilities of digital mobile guides. Original research on context-aware mobile computing applications had already taken place at **Xerox PARC** with the *PARCTAB* prototype [138]. Influenced by this and other related work of the time, such as the *InfoPad* project [90], **Abowd** *et al.* launched the *CyberGuide* project [5] (1997, Georgia Institute of Technology). *CyberGuide* was the first real attempt on creating a functional mobile guide based on location and context, supporting *maps, information access, positioning* and *communications*, both outdoors and indoors (see Figure 3.1 for *CyberGuide*'s user interface).

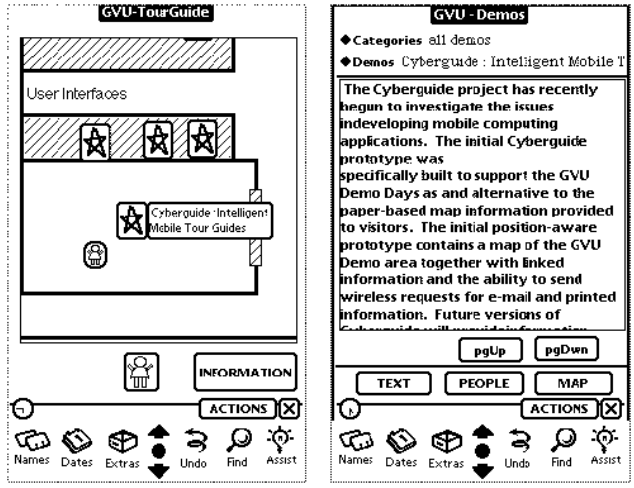

Figure 3.1: The user interface of the first mobile guide, *CyberGuide*, already provided a map interface and textual location-based information (from [5]).

CyberGuide was followed by a number of projects with varying approaches. *GUIDE* by **Cheverst** *et al.* (2000, University of Lancaster) was a web browser -based navigation system for tourists utilizing a **Fujitsu** *TeamPad 7600*, a tablet PC with a gray-scale transreflective screen with a WLAN

(Wi-Fi) connection, for the city of Lancaster [38]. *GUIDE* featured information retrieval, navigation of the city using a map, creating and following a tour, and messaging via text messages. *LoL@* by **Pospichil** *et al.* (2002, Forschungszentrum Telekommunikation Wien) was also a web browser - based tourist guide, splitting application logic between clients and a server [124]. The clients (terminals) ran simple applets, while servers were trusted with more demanding computations (coordinate transformations, routing, map preparation, media streaming). The maps of *LoL@* were generated at server side, providing two scales, an overview map and a detail map, "to overcome the limitations of the small display" [124]. *LoL@* used **HTTP** and **XML** for communications, omitting **SOAP** due to the foreseen high protocol overhead, referring to the work of another mobile guide, *CRUMPET* [155, 123]. The *LoL@* developers implemented the terminal on top of a web browser "to minimize the memory footprint of the application" [124]. The system only ran on laptops.

3.2 Mobile 3D map interfaces appear

Baus, Krüger and Wahlster (2001-2002, University of Saarbrücken) developed an indoor and outdoor navigation system, *REAL*, where the research focus was on providing visualizations and guidance suited for varying technical and cognitive resources [15, 16]. The indoor system utilized a handheld computer, which received content via infrared links, while the outdoor system was based on a notebook computer and a "clip-on" wearable display by *MicroOptical*. *REAL* trusted a single centralized server to produce all presentations on the fly, including selection and preparation of content, based on the available client side resources. The hybrid system was stated to "provide way descriptions that are consistent over different output media". The outdoor map visualization was sketch-like, with navigational instructions given by arrows supported by spoken directions (see Figure 3.2). The indoor 3D representations (images), again rendered at server side, are presented in Figure 3.3.

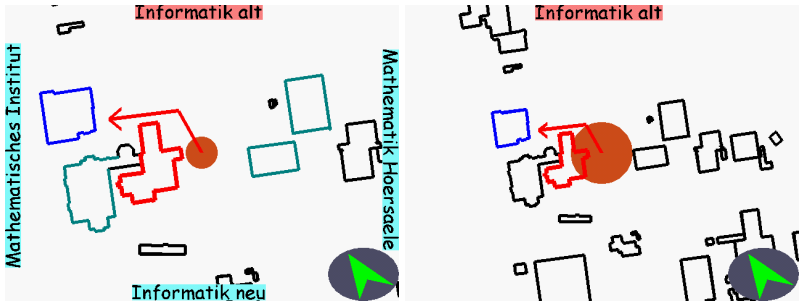

Figure 3.2: Map visualizations of the outdoor *REAL* system for a wearable "clip-on" display (from [16]). The size of the position marker depends on positioning accuracy, and the view scale on user's speed.

Krüger *et al.* developed the *REAL* system further to the *BPN* (BMW Personal Navigator) [85] (2004). The system continued to address issues of

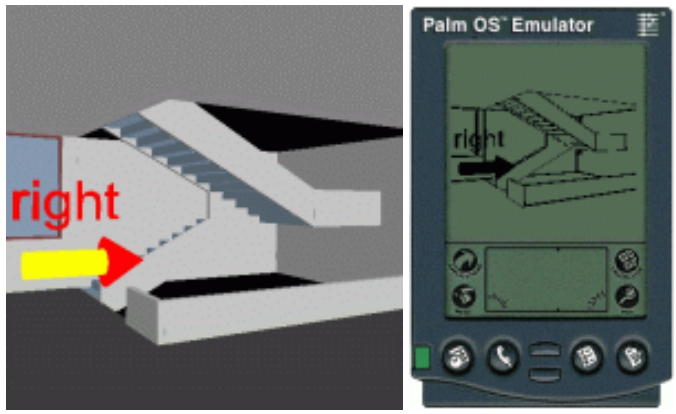

Figure 3.3: 3D visualizations of the *REAL* system for a "clip-on" display (left) and a PDA (right) (from [16]). Both visualizations were rendered at server side and sent to the client.

varying situations and capabilities, from desktop PCs to cars and pedestrians. This time, the system included client side rendered 3D graphics using the **ParallelGraphics** *Cortona* VRML browser, which was "selected from several 3D engines". The rendering was performed in three passes: 1. A bottom layer with a pre-rendered map (a texture) 2. 3D landmarks and 3. Meta-graphical elements such as arrows and markers. The device, an **HP** *iPAQ 5400*[1] was reported to load a scene of five untextured landmark buildings (appr. 50 triangles each) and 8 1024x1024 ground map textures in 10 seconds, and render it with an average rate of 5fps (see Figure 3.4). The buildings were accompanied by bitmap icons (14kB each). A larger model of 700 triangles with an uncompressed size of 85KB took 60s to load, and was rendered at 2–4fps, but "frame dropouts" increased when markers were included in the view. Increasing the number of landmarks and adding a more complex speech grammar for the system's speech recognition system soon led to problems with insufficient memory.

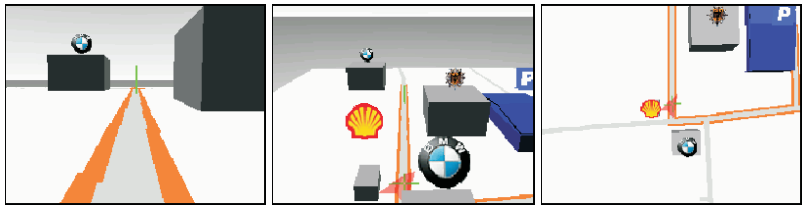

Figure 3.4: 3D visualizations of the BMW Personal Navigator for pedestrians (from [85]).

[1]The HP iPAQ 5400 contains at least 64MB of SDRAM memory and runs an Intel XScale processor at 400Mhz [50].

3D City Info
Probably the first real attempt at creating a realistic mobile 3D map was the *3D City Info* by **Rakkolainen** *et al.* (University of Tampere) in 2001 [133]. The mobile system utilized a 3D VRML model of a city center created earlier for a PC platform [8]. The model was simplified to some 5100 polygons, containing 120kB of geometry and 5MB of textures. With texturing, the models looked "highly realistic". The project used an **HP** *Jornada 548* as the rendering client[2]. The frame rate with a *Cortona CR* VRML browser was one frame every 8 seconds (0.125fps), with severe problems of running Java 2. The authors expected these problems to "be removed gradually, as hardware improves". The related field experiments were performed with static web pages on a PDA. Later, further experiments were performed with a laptop version, with subjects in a car due to the heavy equipment [157, 158]. Some of the authors, such as **T. Vainio**, have continued research on mobile navigation and 3D maps, but without experiments on functional PDA platforms [156].

The TellMaris Suite
The EU funded *TellMaris* project (2001-2003) focused on boat tourists, who would utilize laptop-based 3D maps on the sea, and mobile 3D maps within harbor cities. The project created several 3D map prototypes with different approaches.

Sintef's TellMaris OnBoard **Sintef** (Norway) built a *TellMaris OnBoard* application using *Java 3D* [72] facilitating 3D navigation and information search, applying aerial photographs as ground textures [86]. The information search component contained weather information. The interface provided a large 3D view and a smaller 2D view ("sail view" and "top view"). The software ran on laptops and desktop PCs.

Nokia's TellMaris Guide **Nokia** developed a *TellMaris Guide* (TG), a mobile 3D city guide that run on a Nokia *Communicator 9210*[3] using a proprietary *NokiaGL* rasterizer, similar to OpenGL v1.3, but with limitations [24]. TG's user interface was designed by **Katri Laakso** (of Nokia) and implemented by **Gerard Bosch** (of Nokia) based on a textured model by **Pavel Syssoev** (of TKK). The system utilized a simple regular subdivision scheme, where the 3D city model was split to a regular grid of 100x100m cells using macro scripts in the modeling software, *3D Max*, and where each cell was associated with an aggregated texture [24]. The system was able to run at a few frames per second using texture resolutions of 128x128 per cell (texel sizes of 1–2m), rendering the nearest 9 cells[4]. The system utilized the modeling software's native binary format, the *.3DS* format for (local) model storage, avoiding for example X3D, where "XML encoding would

[2] An HP Jornada 548 has 32MB of RAM and a 133Mhz 32-bit Hitachi SH 3 processor [49]
[3] Nokia 9210 has a 32bit ARM9 with 2MB user memory and 8MB execution memory [24]
[4] On an emulator, the TG achieved 10–11fps [24]

have lead to a huge increase in the size of our model" [24]. Figure 3.5 presents the Nokia TG view.

Figure 3.5: Nokia's TellMaris Guide (screen shot from an emulator).

Fraunhofer IGD's TellMaris Guide **Arne Schilling** from Fraunhofer IGD developed a version of the *TellMaris Guide* for PDAs using the PocketCortona VRML library with Java. The authors report that "The rendering performance was just sufficient in order to navigate in models with approx. 0,4 square kilometers, comprised of a terrain with 20 m resolution, a block model and textured buildings" [139]. The platform was a **HP** *iPAQ H3870* with a 266 Mhz processor. The textured buildings were the same as used in Nokia's TG, but rendering did not apply the cell-based space subdivision scheme. Further development or porting to newer platforms was hindered by discontinuity of support for the version of Java the IGD team had been using [139].

The authors observed the following limitations, which need to be considered when developing 3D maps [139]:

- "The rendering capabilities depend on the amount of memory and the chipset... it is expected that the rendering capability will grow very fast in the future as it did on PCs and laptops in the past."
- "The network bandwidth is one of the bottlenecks... UMTS will increase the transfer rates significantly" ... "Specialized compression techniques can reduce the amount of 3D data [48] which enables the transmission of 3D models combined with videos and other multimedia content."
- "The small screen size and the restricted interaction possibilities require new metaphors for exploring and navigating in 3D maps. The Graphical User Interface has to concentrate on the most relevant information allowing for these technical restrictions."

TKK's Mobile 3D Archipelago The last implementation of the TellMaris family of 3D maps was **TKK**'s *Mobile 3D Archipelago*, a 3D map suited for archipelago areas. It was implemented with OpenGL, where a topographic data set (appr. 235MB of text files) was processed and optimized for 3D rendering, attaining the original granularity, yielding appr. 14MB

of binary files, including textures. The system featured land categorization-based template ground textures and static location-based information. The case area covered 3500km^2, and was divided to a regular grid consisting of 1x1km^2 cells. Location-based information and features, including sea marks and buildings, were rendered with billboards[5] using a single aggregate texture. The first version was ported to a PDA in Spring 2003, applying **Hybrid Graphics'** *Gerbera* rasterizer, which later became the first official OpenGL ES implementation [67]. The final version was implemented by the February 2004, but reported rather late, in 2007 [113]. On a Dell *Axim X30* PDA with 624MHz XScale processor and 64MB of memory, the map ran over 10fps, using 10MB of memory (visualizing a 3$x3km^2$ area using 9 textures of 256x256 pixels each that roughly matched the original map data accuracy). On a Dell *M60* laptop with **NVidia** *Quadro FX Go700* graphics hardware, the map ran over 200fps at 1024x768 screen resolution, using 32MB of memory. Figure 3.6 presents the mobile version on a **Fujitsu-Siemens** *PocketLoox*, where it ran at 5–8fps[6]. The data was stored locally. No individually textured buildings were included in the system.

Figure 3.6: TKK's Mobile 3D Archipelago.

Fraunhofer-IGD's Mobile 3D Viewer

Blechschmied, Etz and **Haist** (Fraunhofer Institut Graphische Datenverarbeitung) presented a mobile 3D city viewer based on the JSR-184 rendering API on a **Nokia** *6630* in 2005 [21]. The system suffered from a 2-3 minute initialization time with local content, rendering speed of less than 1 fps, a limitation to only a two or three low resolution textured buildings, and frequent crashes.

[5] Billboards are images of objects that always face the viewer. See section 4.3.
[6] The PocketLoox has a 400Mhz XScale processor and 64MB of memory.

LAMP3D

In 2005, **Burigat** and **Chittaro** (University of Udine) presented *LAMP3D*, "a system for the location-aware presentation of VRML content on mobile devices" [35]. *LAMP3D* utilized the *Pocket Cortona* browser for viewing VRML content, and featured information delivery and GPS support. All information was stored locally. The authors were inspired by research by **Marvie** and **Bouatouch** [95], binding the viewpoint close to the ground and rendering only the buildings that are visible. They used an existing city model, which was simplified both in geometry and texture resolution to increase the frame rates from 1–2fps to a maximum of 4–5fps. Details of the model, the textures, or how visibility information was determined, were not provided. The device used was a Compaq *iPAQ h3970*[7]. They conclude that "computational limitations of current mobile devices do not allow for a sophisticated use of 3D graphics and a trade off must be reached between performance and quality of the representation". Figure 3.7 presents the *LAMP3D* interface.

3.4 Annotations and tracking

GeoNotes

Most mobile guide research projects have focused on navigation, where the environment is visualized with maps, providing services for tourists with the *information pull* paradigm: users ask the system *What is...?* and the system retrieves the information. Similarly, positioning is used to place the user on a spatial context, and to aid in navigation. *GeoNotes* (Swedish Institute of Computer Science), started in the summer of 2000, had the goal of using positioning information for social enhancement of the environment, where users *pushed* information to it as annotations using the global coordinates (latitude and longitude), and retrieved it by providing a position and a radius. The *Wherewhoo* database was independent of the client. The system applied social recommendation and collaborative techniques.

Between

The *Between* project (2001-2003; TKK/HIIT) developed, among other things, a radar-based egocentric interface to the vicinity of the user. The *InfoRadar* system was suited both indoors and outdoors, applying GPS positioning outdoors and WLAN positioning indoors[8] [134]. The radar interface was a bidirectional interface to the environment, where users were able to place annotations, or read annotations placed by others. The annotations included text and pictures. Users' paths were also presented on the display (see Figure 3.7).

[7] iPAQ h3900 series PDAs feature an Intel XScale processor running at at 400MHz, and 64MB of memory or more [51].

[8] InfoRadar's indoor positioning services were implemented using **Ekahau Corporation**'s WLAN positioning product.

Figure 3.7: 3D and 2D interfaces to the environment: (left) *LAMP3D* and (right) *InfoRadar*.

3.5 Commercial products

Navitime (2007) is one of the most complete navigation systems up to date, claiming a "total navigation support" [11]. It is suited for walking, driving, riding trains, buses, taxis and airplanes. The guidance includes maps, itineraries, voice prompts, progress bars etc. It even includes a 3D interface, which was designed to remedy the situations of being lost, for example after exiting a train station. The authors state that "we need faster mobile phone networks and more comprehensive 3D data to make the 3D interface more usable for massive everyday use". The system is reported to use the *V3D format*, which compresses texture data "for small screen sizes" and supports shared textures. Further technical details are not provided. Figure 3.8 presents the *Navitime* 3D interface.

Figure 3.8: The 3D interface of *Navitime* [11].

Google Earth has been recently launched for the *iPhone* and *iPod touch* platforms, providing a limited version of the desktop *Google Earth*. **Google** views *Google Earth* as a browser, where the browsing paradigm has been reversed. Instead of browsing the Web where spatial information can be found, one browses the Earth, onto which the Web provides content [76]. The mobile version does not currently support 3D buildings. The ground is textured using aerial photos.

3.6 Comparisons and discussion

Table 3.1 presents selected properties of the projects' attempts at running 3D city maps on mobile devices. Direct comparison is difficult due to the varying cases and approaches. However, most of the projects have applied the *Pocket Cortona* VRML viewing library in a straightforward manner. The high quality model of *3D City Info* clearly fails to reach interactive frame rates. *LAMP3D*'s restriction to ground level and use of (possibly manually assigned) visibility lists can be argued to provide a clear increase in frame rates. Unfortunately, no model statistics from *LAMP3D* has been published. BPN did not include textured buildings. The model geometry and texture limitations of *TellMaris Guide* (TG) together with a simple grid-based subdivision and nearest neighbor cell rendering scheme allow a device with relatively little computing power to reach near interactive rates. The TG frame rates could not be verified, as only results from an emulator were available. The subjective observation by the author of this thesis on the system was that it yielded a few fps, being nearly interactive.

Prototype	3DCityInfo	TG	BPN	BPN	LAMP3D	F-IGD 3D
Year	2001	2003	2004	2004	2005	2005
Device	Jornada 548	Nokia 9210	iPAQ 5400	iPAQ 5400	iPAQ h3970	Nokia 6630
3D API	Cortona	NokiaGL	Cortona	Cortona	Cortona	JSR-184
Format	VRML	3DS	VRML	VRML	VRML	M3G
Space sub.	Scene graph	Grid+NN	Scene graph	Scene graph	(manual?)PVS	Scene graph
Triangles	5100	>1000	250	700	(?)	>1000
Textures	5MB	19x(128x128)	5 x 14kB icon, ground tex.	21 x 14kB icon, ground tex.	(?) >1000	3x(64x64)
Start	(?)	(?)	10s	60s	(?)	>2min
fps	0.125	a few	5	2–4	4–5	<1

Table 3.1: Properties of locally rendered 3D city map prototypes. The buildings of BPN were not textured, but were marked by an icon.

Of the presented prototypes, only Fraunhofer IGD's *3D viewer* claims networked model downloading (as supported by the M3G format). The *BPN* features online annotations, but only for local replay. *TG* provided locally stored points of interest and a 3D arrow pointing at a goal, but no routing. BPN provided server side routing and speech recognition. *LAMP3D* featured local information retrieval. Most of the prototypes featured GPS support.

While none of the research projects reached a "full-featured" state (such as the recent, commercial *Navitime*), it was not their primary goal. Those with emphasis on 3D maps were interested in the possibilities offered by such a representation. For example, *3D City Info* defined its goal as a question, "Would 3D graphics add value to mobile navigation and service brows-

ing?", hypothesizing that "3D interactive environments are an intuitive and user-friendly way to view location-based information". Despite their failure of reaching interactive frame rates, their experiments with still images on PDAs were encouraging: "More likely they [subjects] also recognized their own position and the landmarks from the photo-realistic model than from the map. The visual similarity with the reality helped them to find the places in real life" [133]. In a succeeding study, which was performed in a car and a laptop with interaction, **Vainio** and **Kotala** found that "the three-dimensional model illustrates motion and changing the location more clearly than two-dimensional map alone. The visual similarities with reality support the participants to find the places or landmarks from the real city environments when a user is moving from one place to another." [157].

On the technical obstacles, **Rakkolainen** *et al.* focused on model size: "The major problem with even medium detailed city models is their big size, which affects both the rendering speed and the download time." "The major bottleneck is the data transfer... The download time could be shortened with many optimizations, e.g., by providing a model with no textures, but then the visual quality is partially lost. Level of details improves the rendering speed but does not affect the download time." [133]. On rendering issues, the authors refer to various progressive level-of-detail techniques, such as **Hoppe's** *Progressive Meshes* [70], and mention that "model simplification, geometry smoothing or other techniques could be used". They also look forward to *X3D*, the follow-up of VRML, and its players, which they consider to be "particularly suitable for mobile devices" (the research of this thesis cannot support this claim). **Burigat** and **Chittaro** had an existing model, but that ran only at 1–2fps, so they "simplified some of the most complex objects, removing unneeded geometries and deleting some elements of the representation", and "lowered the resolution of textures and, wherever possible, substituted unnecessary ones with the use of the VRML Material node" to reach their maximum 4–5fps [35]. They also limited the viewpoint to ground level and utilized ground level visibilities to some extent.

From the field experiments performed with *TellMaris Guide* [84], several positive results were recorded even though the experiments were performed with a laptop: "Users' attitudes towards the prototype were very positive and three fourths of them would like to use this kind of service rather than 2D paper maps and guidebooks." "Users were able to recognize real world objects from the 3D model and use these landmarks as navigational aids." Many users also reported that the 3D map was "more fun" to use. The authors did not take all positive feedback as granted, assuming that "there was certainly a novelty factor involved". There was also feedback on the model quality: "Some users claimed that non-textured buildings were hard to distinguish from each other, but textured buildings, especially one white house, were considered easy to recognize." Furthermore, "According to the users the 3D model should be more detailed and realistic and the target [in navigation tasks] should be highlighted in it". Finally, the authors hypothesize that "The more realistic the visualization, the more demanding it is in terms of technical resources. But the resulting realism also enables the user to recognize objects from the rendering in the real world, which is a distinct advantage over all other means of presentation."

4 A STATIC MOBILE 3D MAP

This section describes computer graphics optimization approaches to rendering large, static scenes. In such a scene, the viewpoint can be moved around, but the objects remain stationary. Furthermore, a *large* scene does not fit in the graphics subsystem. In the end of the section, the solutions for a 3D engine suited for navigating in urban environments are presented, elaborated from publication P1. The rendering speed results with 3D hardware enabled mobile devices are from publication P2.

4.1 Concepts

Output sensitivity

Consider rendering a very large and detailed scene, such as an entire city, and the screen of a mobile device. Despite the complexity of the scene and the computations that take place during rendering, the end result is placed onto that relatively small screen. With a reasonable field-of-view (FOV), most buildings lie off the screen. Of those that are in view, only a part of the façades are visible, as roughly half are behind the buildings, and many are occluded by nearby buildings, for most camera angles. Finally, if the scene is projected with the perspective transform, the buildings far away contribute very little to the scene in screen space. A small building on the other end of the city might contribute even less than a pixel to our low resolution frame buffer. The same would apply to any small detail sufficiently far from the camera. Would it then be appropriate to cull these almost undetectable details away? Would it be appropriate to represent the remaining objects with a level of detail that matches their size on the screen? Now, if our scene would consist *only* of such a description, wouldn't we be able to render it even with the tiniest of computational resources?

The situation described above, where rendering speed is dependent on the size of the output, rather than the input, is known as being *output sensitive*. In practice, the update rate would depend on the number of visible objects and their level of detail – but in the extreme case, on the number of pixels in the frame buffer. **Sudarsky** and **Gotsman** provide a definition for output sensitivity [149]:

> A visibility algorithm is called output sensitive if its runtime per frame (excluding any initializations) is $O(n+f(N))$, where N is the number of primitives in the entire model, n is the number of visible primitives and $f(N) \ll N$. $f(N)$ is the (inevitable) overhead imposed by the model.

Visibility culling

Reaching an output sensitive situation requires an efficient algorithm to cull away all parts of the scene that do not contribute to the current view. Figure 4.1 presents an intuitive view to the basic methods, namely *frustum culling*, where objects lying outside the current view frustum are culled; *back-face culling*, where surfaces pointing away from the virtual camera are culled;

occlusion culling, where objects or surfaces that are occluded by other objects closer to the virtual camera are culled; and in *contribution culling*, those objects whose contribution to the image is insignificant.

Output sensitivity is a key concept in optimizing a static scene for efficient rendering. However, achieving such a situation is quite complex, and case dependent optimizations become essential. The various culling techniques are discussed in section 4.2 in more detail.

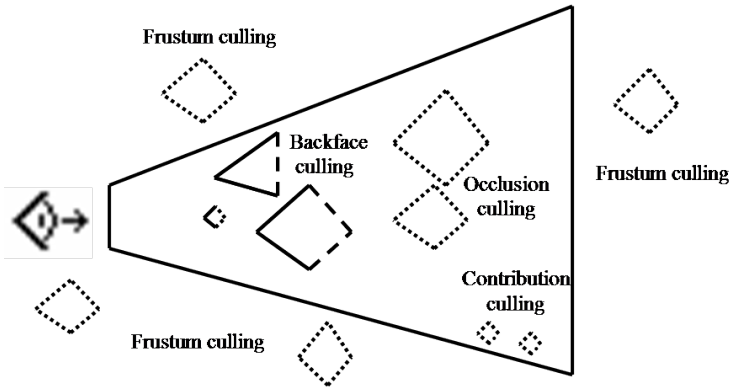

Figure 4.1: Culling techniques to reach output sensitivity.

Classification of culling algorithms

Visibility determination algorithms may yield results of varying accuracy, and can be categorized as follows [107]:

- *Conservative*. The result is overestimated; no truly visible objects are discarded. There are no artifacts caused by the visibility calculation, but the set of visible objects may be larger than necessary.
- *Exact*. The result exactly matches the truly visible objects. No hidden objects are included, nor are any visible objects discarded.
- Approximate. The result is constructed as an approximation, where some truly visible objects may be discarded, and some hidden objects included.
- Aggressive. The result is underestimated; it does not include any hidden objects, but some truly visible objects may be discarded.

Visibility can be generally determined for a single viewpoint or for a larger region of space. The algorithms are called, respectively, *from-point* and *from-region* (or *from-cell*) algorithms [107]. Similarly, depending on the atomic target of the visibility determination, they can be called *to-region* (*to-cell*) and *to-object*. An algorithm determining the visibility of objects within a given region (cell) seen from another region can be called a *cell-to-cell* algorithm.

Out-of-core rendering

When the scene to be rendered exceeds the capabilities of the graphics subsystem, the renderer needs to swap parts of the scene out and read new parts in from main memory or external media. Computational methods that depend on utilizing external memory are called *out-of-core* algorithms, applicable to a range of fields within computer sciences [4]. In the case of rendering urban 3D environments, the resources of mobile devices can clearly be expected to be insufficient (see Chapter 2), and such methods become a necessity. Out-of-core algorithms are discussed in Section 4.4.

4.2 Culling techniques for static scenes

Image space visibility

Image space visibility determination, or *hidden surface removal* (HSR) was one of the major research topics of computer graphics in the 1970s. Several algorithms were suggested [151], until the *Z-buffer* algorithm by **Ed Catmull** [37] became the *de facto* method. The Z-buffer method asserts that only the closest pixels of a scene end up in the frame buffer, resulting in a correct rendering of the scene. However, it does not solve the issue of scalability: larger scenes still require more computations, slowing down the update rate. An improvement was provided by **Greene** *et al.* with the *hierarchical Z-buffer* (HZB) [66]. The authors use dual data structures, an octree in object space, and a Z-pyramid in image space. The scene is subdivided onto an octree by starting from the root node (a very large virtual cube) encompassing the entire scene, then subdividing recursively until each leaf node contains a "sufficiently small" number of primitives. This hierarchy is then utilized in object-space visibility determination: if a node is occluded, so are its children (its sub-cubes). To speed up the node visibility test, a hierarchy of Z values is used, where the finest level is the original Z buffer.

Other conservative image space visibility determination algorithms include **Zhang's** *hierarchical occlusion map* (HOM) [167]. HOM assembles a set of potential occluders at pre-process stage, renders them onto the frame buffer at run time from the current viewpoint and scales them to a hierarchy. The bounding boxes of scene objects are then tested against this set. The *directional discretized occluders* (DDOs) algorithm by **Bernardini** *et al.* provides another image-space solution with object and image space hierarchies and pre-generated view-dependent occluders [18]. Image space algorithms depend on reading back the frame and Z buffers, causing a major bottleneck. The situation has improved to some extent with hardware support, for example by the *HP Occlusion Query* extension to OpenGL [142].

Exact 3D visibility

An optimal output sensitive situation would not be just conservative, but exact. **Naylor** suggested using a binary space partitioning (BSP) tree [61], constructed using particular heuristics, for describing the scene and performing a 3D-to-2D tree projection with ordered set operations on the polyhedral faces, leading to an exact, output sensitive situation [106]. The weaknesses of this algorithm have been pointed out to be the dependency

of the particular BSP tree generation heuristics and the lack of real world BSP-tree-based models [150], lack of practical performance [64], and as **Sudarsky** and **Gotsman** point out [150], it is not generally feasible to transform the more generally available boundary representation models (*B-reps*) to BSPs due to the requirements imposed on the models. Other exact visibility solutions can be found from the work by **Nirenstein** *et al.* [107] and from the *aspect graphs* arising from the fields of computer vision and photogrammetry [122, 58].

Approximate visibility and contribution culling

Approximate visibility algorithms are used to replace conservative or exact algorithms for efficiency reasons, to enable faster rendering with the penalty of allowing perceivable artifacts to emerge. For example, the *prioritized-layered projection* (PLP) algorithm applies a probabilistic estimate on the visible primitives for a given viewpoint, up to a user-specified budget [81]. If the resulting artifacts are limited to only small details, they may not affect much the viewing experience. **Andújar** *et al.* define such a set of objects that contribute only a small number of pixels to the view as the *hardly visible set* (HVS), a subset of the potentially visible set [9]. At run time, objects from the HVS are discarded up to a user-bounded threshold. This technique is also known as *contribution culling*. Contribution culling methods have been applied for view-dependent rendering of urban environments, observing that typically the rendering quality is not degraded notably [119].

Architectural walkthroughs

Aside from the theoretical frameworks, researchers also approached the visibility problem from the viewpoint of practical cases. *Architectural walkthroughs* were considered important applications in the late 1980s, characterized by interaction and a first person view. In early experiments, scenes of 5000–8000 polygons were rendered every 3–5 seconds, but increased to 9 updates per second by advancing rendering systems and separating the model database to a "Master Model" and a "Working Model", a specialized subset that was used for fast rendering [30]. Further work in this direction resulted in the concept of a *potentially visible set* (PVS) by **John Airey** [7], and **Seth Teller** and **Carlo Séquin** [153, 152]. The PVS is a list of visible objects from a certain viewpoint, or *view cell*. The visible objects can be contained within a similar object space cell. In an indoor environment, the cells can be naturally divided to rooms and corridors, connected by *portals* (doors). These solutions depend on a pre-process that generates the visibility lists. Airey provided several "sampling" approaches, which are generally considered not to guarantee conservative solutions [45], while Teller's work included a conservative, analytical but also computationally more expensive solution. In Teller's *cell-to-cell* visibility schemes, all objects that belong to a certain cell would be considered visible, if the cell itself is deemed visible, unless further (cell-to-object or eye-to-object) examinations are performed [62]. Later, **Luebke** and **Georges** [92] claimed that for dynamic cases, where the walkthrough would be a part of an interactive modeling software, pre-process is not a viable option. Their online eye-to-object solution projects the portals to 2D *cull boxes*. During scene traversal, each suc-

cessive portal is added to the aggregate cull box with simple comparisons. Cell contents (objects) are tested for visibility by projecting their bounding boxes to the aggregate cull box.

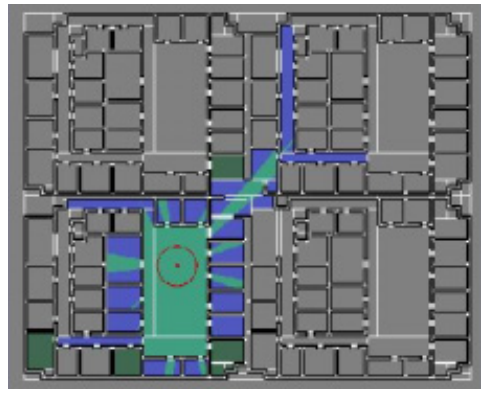

Figure 4.2: The potentially visible set in a 2D indoor scene (from [152]).

Urban walkthroughs

Urban environments pose another case of densely occluded environments. In this case, the environment is open, where occlusion is formed from several individual objects, such as buildings. At street level, the nearby buildings occlude most of the city, and give rise to another class of applications, namely *urban walkthroughs*. **Nadler** *et al.* [105] provided an analysis of street-level (2D) visibilities in urban environments, yielding a mathematical, probabilistic model for aiding urban walk-through algorithm optimizations. **Wonka** and **Schmalstieg** [163] made a less theoretical analysis of the properties of urban scenes using European cities, noting how the environment consists of mainly a ground mesh with different objects, such as buildings, trees, traffic signs, etc., placed on top of it. They also note that common views are blocked by a small set of buildings, but there are also larger areas such as railway stations where occlusion is still high, but the occlusion is made up of a larger number of occluders. They exploit the "2.5D" property of the environment to render occluder shadows with graphics hardware to 2D bitmaps, *cull maps*, where the object space is divided to a regular grid. For each frame, they select the occluders, draw the 2D cull maps, and traverse the cull map to create the PVS.

Cohen-Or *et al.* [46] partition the view space to cells at street level, and for each potentially visible object search for a single *strong occluder*, which could completely hide the object from the view cell. If such is found, the object is culled and is not included in the conservative PVS. The authors also demonstrated the system for a 3D case with spheres. In their follow-up paper [164], Wonka and Schmalstieg developed a conservative from-cell occluder fusion method applied to urban environments with precomputed visibility. They solved the unconservativeness problem inherent to sampling systems by observing that if the occluder is shrunk by ϵ, the resulting visibility is conservative within the distance of ϵ from the sample point.

Now, at pre-process time, by suitable selection of ϵ and sample points, they could guarantee the conservativeness of the resulting PVS. Their algorithm was able to fuse even disjoint umbrae, and the view space (the streets) was partitioned by a constrained Delaunay triangulation.

Marvie and **Bouatouch** followed the tradition of urban walkthroughs in [95], suited for VRML viewing, where they limited the navigation space to a joined set of convex cells, and extended the VRML format by this structure. Their system provides both cell-to-object and cell-to-cell visibilities. In the former case, objects are mainly buildings, and in the latter, the object space cells contain all objects small enough to fit in the cell. The authors call the combined visible set a *hybrid PVS*. They also discuss memory management, where parts of the model can be discarded from memory, if they are not referenced to. The system featured progressive downloading, where the content is transferred when needed (when appearing in the PVS), or prefetched. However, the system required the contents of a cell's whole hybrid PVS to be transferred before navigation in it could start. The system relied on VRML extensions, to be run on VRML viewers.

View frustum culling

Many of the occlusion culling methods described in the previous sections perform view frustum culling (VFC) as part of the occlusion culling process. However, if an implementation requires a separate VFC, hierarchical methods have been proved efficient. For example, **Möller** and **Haines** [101] presents a general, hierarchical VFC algorithm. We assume that a scene is arranged into a *directed acyclic graph* (DAG), a scene graph, where the geometry is stored in the leaves, and nodes store pointers to its children [42]. Leaves and nodes are enclosed by virtual bounding volumes. The bounding volumes can be compared to the view frustum starting from the root node, discarding entire subtrees if a node lies outside the frustum. They also provide efficient intersection tests for frustum against bounding volumes, such as bounding spheres and bounding cubes. [12] provides mechanisms for further speed-ups based on frame-to-frame coherence (*temporal coherence*) where VFC results are cached, and modified by the movement of the viewpoint, which can provide a "generous speed-up in comparison to a naïve VFC".

Back-face culling

Any flat polygon can be viewed from both sides. However, when polygons are used for constructing objects, the insides seldom need to be rendered. In this case, *back-face culling* can remove the unnecessary faces. The facing direction is defined by *polygon winding*, the order in which the polygon's vertices are specified. The back-face culling test can then be implemented by computing the normal of the polygon's projection in screen space, and testing for its sign [101]. Back-face culling reduces rasterization, but increases geometric computations. 3D programming interfaces commonly support back-face culling [22]. However, when the back-face test is performed in the graphics pipeline and not at application level, the geometry and its related detail, including textures, need to be loaded to the graphics subsystem, increasing memory consumption.

4.3 Level-of-detail methods

Due to perspective projection, objects in the distance appear smaller on the screen than those in vicinity. *Level-of-detail* (LOD) techniques exploit this fact to optimize the representation of objects and their detail to suit the estimated screen size. [162] provides an early description of building an "image pyramid" out of a set of scaled textures, successively dividing the texture by two until the smallest required size is reached (ultimately, 1x1). When rendering a scene, the most suitable scale can be selected, matching the size on the screen. The image pyramid is also called a *mipmap*. If a surface is tilted along the view direction, the matching LOD might differ in the closer and farther ends of the surface. When different LODs are blended during rendering, this is called *trilinear interpolation*. This requires the entire image pyramid to be held in the memory.

Similar pyramidal data structures can be developed for geometry. [41] and the thorough [93] provide an extensive view to the related technologies. View-independent techniques generate a set of objects independent of viewing angle, while view-dependent algorithms consider the angle as well. A common problem is the "popping" artifact, which happens when the LOD version of an object changes. **Hoppe** developed a method for avoiding this with *progressive meshes*, where the mesh is progressively simplified or refined with the *edge collapse* and *vertex split* operators [70].

Impostors are flat images of objects that can be used to represent entire objects. A *billboard* is such an image, always oriented toward the viewer. **Pasman and Jansen** compared simplification methods using a mathematical model for estimating image quality resulting from different simplification methods, including geometric and image-based techniques [121]. Their case was an augmented reality application, where object representations would be streamed from a server, computing the view dependent error on geometric distortion. As an example of the results, they demonstrate that an object of the size of one meter could be replaced by an impostor from farther than 15 meters. **Sillion** *et al.* discuss the use of impostors for replacing building geometry in urban walkthroughs [145]. They also provide examples of using the depth information of the actual geometry to create a triangulated approximation surface, which would not suffer from the parallax effect due to viewpoint movement as much as normal, planar impostors.

4.4 Out-of-core algorithms for 3D rendering

Continuing on the research on architectural walkthroughs, **Funkhouser**, **Séquin** and **Teller** implemented a system integrating levels-of-detail and visibility with real time memory management and a refresh algorithm for "choosing which objects to render at which levels of detail in each frame" [62]. Their display database system contains a *segment index* holding every objects' id, type, size and pointers to either memory or a file. The *segments* are considered just chunks of bytes with a size, with the same format in memory and file to allow fast swapping. To minimize file I/O latencies, objects belonging to a same cell (in object space) are grouped contiguously in the database file. Furthermore, they are sub-grouped according to their

LOD. Objects straddling a cell boundary are stored redundantly for each cell. At run time, their refresh algorithm fetches visible objects, and chooses a LOD version with sufficient detail for the current view. They also implemented pre-fetching, considering that "frame updates might be delayed, waiting for objects to be read from disk before they can be rendered".

Funkhouser, Séquin and Teller intuitively developed application level control for data management to optimize their system. Later, **Cox** and **Ellsworth** demonstrated how relying on operating system's abilities for virtual memory management leads to "egregious" performance, and how application level management can indeed be used to solve the problem of *out-of-core visualization*, significantly improving performance [54]. In their terminology, in out-of-core rendering, the main memory (but not the memory of the graphics subsystem) is sufficient to hold the entire data set. When main memory becomes insufficient, they call the situation *remote out-of-core visualization*[1], where data is fetched from external medium, possibly from a file server. Out-of-core algorithms have later been a central part of rendering large data sets, such as terrain [88], and are often accompanied by prefetching algorithms [53]. The *vLOD* system by **Chhugani** *et al.* was one of the first systems providing efficient walkthroughs of large models with low CPU load, combining preprocessed 3D visibilities and preprocessed LOD selection with efficient encoding mechanisms to yield interactive display of out-of-core models [39].

4.5 A 3D map engine for urban environments

Engine requirements

Of the early systems, only *LAMP3D* and Nokia's *TellMaris Guide* applied spatial subdivision, but not to the extent computer graphics optimization methods in previous sections have been discussed[2]. Of these methods, *urban walkthroughs* would best suit our case. However, would they be sufficient for a navigation application, with the common limitation of the viewpoint to the street level? Would this limitation be supported by users? We postulate that this can only be verified by navigation experiments on the field with real users. Therefore – until proved otherwise – our mobile 3D map should be able to provide free browsing in the entire 3D space, while providing interactive frame rates.

Our engine should also provide a full support for a 3D navigation user interface. Discussion on navigation related research is presented in section 6.1. From the discussion, it is obvious that landmarks pay a significant role in navigation. Therefore, the engine needed to have direct support for them. We therefore chose to render landmarks even if they lie beyond the far plane of the view frustum. Navigating too close or through virtual world objects has also been identified as a major source of disorientation [146].

[1] **Hesina** and **Schmalstieg** coined the term 'remote rendering' to describe remote out-of-core rendering [69]. However, remote rendering has later been used to describe a situation where the rendering is performed remotely, and final frames sent to the clients [132, 25].

[2] The authors of *LAMP3D* acknowledge the work by Marvie and Bouatouch [95], but similar to Marvie and Bouatouch, the system was dependent on the capabilities of VRML viewing libraries.

Therefore, collision avoidance should be applied. Navigation features that are not related to 3D rendering are discussed in section 6.

Visibility culling for a 3D city map

Publication P1 presents our static 3D map engine. Considering the lack of resources, especially memory and CPU, the engine needed to be lightweight, mainly utilizing look-up tables instead of run time computations. This is exactly what from-region precomputed potentially visible sets provide: lists of potentially visible objects from a given view cell. In our case, neither the indoor spatial subdivision to rooms and corridors (building walkthroughs), nor the 2D or "2.5" division to cells at street level (urban walkthroughs) would suffice. We needed the view space to span the entire 3D volume, so we chose a 3D regular grid subdivision.

As demonstrated by urban walkthroughs, a PVS solution provides significant savings at street level, but an overview of a city would not be so beneficial. There, a PVS would mostly cull geometry that could be dealt with in the graphics pipeline with *back-face culling*. However, let us consider our resources. For back-face culling, geometry *and* the textures need to be loaded in the graphics subsystem. Furthermore, back-face culling requires online computations for each triangle, while a PVS simply never loads the occluded geometry. Similar savings are gained by hierarchic frustum culling. Only the *bounding volumes* of buildings and, hierarchically, surfaces, are tested before the actual model data needs to be loaded in.

Our algorithm for the PVS generation pre-process is approximate to simplify the implementation, but aims for conservativeness. The scene is rendered from the corners and center of each cell to every direction using color coding for objects, extracting the colors, mapping them back to IDs, and combining onto a visibility list (a view cell is visualized in Fig. 4.6 on page 42). The pre-process can be shared over a network, where multiple *pre-process clients* render the scene, submitting the resulting visibility lists to the server.

Publication P2 presents an additional culling scheme, *contribution culling*. This is implemented by calculating the contribution of each object in pixels during the pre-process, and including only those above a fixed, minimum threshold. This adds control over the aggressiveness of the pre-process.

We trusted in field experiments to provide feedback on the possible artifacts caused by the approximate visibility sampling algorithm and the contribution culling.

Compressing visibility lists

A preprocessed PVS approach generates visibility look-up tables where potential visibility can be resolved for individual polygons, polygon aggregates such as meshes, entire objects, their bounding volumes, or object groups and their bounding volumes. The smaller this atomic unit, the more *exact* the list. However, the cost of this exactness is the larger size of the lists. **Van de Panne** and **Stewart** discuss effective visibility compression techniques and offer one solution [159]. They consider a look-up table with view cells on a row, and potentially visible *polygons* on a column. By merging similar rows

and columns, they provide a lossy compression of the table. By creating new aggregate rows and columns, they provide a lossless clustering, compressed by run-length encoding (RLE). **Bouville** *et al.* have later improved this algorithm by applying row and column pivoting operations to increase the size of the clusters, and applying more efficient compression algorithms, such as the **JBIG** [26].

Rendering individual triangles is usually more costly than for aggregates, such as *triangle strips* or *triangle fans*[3] [101]. Van de Panne and Stewart mention that polygons could be "pre-clustered" manually given some knowledge of the scene. We chose to do this at modeling phase, using the façades and rooftops as atomic objects, *meshes*. The external knowledge utilized here is the common rectangular shape of buildings.

The scheme by van de Panne and Stewart has yet another, innate problem. Their encoding algorithm breaks the natural coherence between the visibilities of adjacent view cells present in a 3D space subdivision[4]: the view cells are arranged to a one-dimensional row, and the 3D adjacency cannot be recovered fully with 1D operations, such as Bouville *et al.*'s row and column pivoting. Therefore, as described in publication P1, we chose to develop a compression scheme based on a tree structure, where a root cell includes a full list of visible objects, and the lists of the leafs (adjacent cells) only store the differences. A set of rules was established to limit the cluster size, such as a maximum amount of visibility difference, and a maximum distance from the root cell. Currently, we pack the difference lists with run-length encoding. Independently of us, **Chhugani** *et al.* developed a similar scheme for their **vLOD** 3D walk-through system, where they included LOD information to the tree and applied Huffman encoding to the lists [39]. We may still proceed toward this direction, as claimed in publication P1.

Levels of detail

Typical virtual environments tend to use mipmapping as a standard way of avoiding visual artifacts caused by texture sampling errors, even for mobile systems [99]. However, mipmapping requires all levels of detail of a texture to be held in memory. Our choice was to hold in memory only the currently needed LOD texture. For the lowest LOD, we chose a *dominant color*, the color that contributes most to the building. Quite recently, a similar but more sophisticated system was implemented by **Grabler** *et al.*, where they segment the façade texture to components, average each component and select the brightest color [65].

For geometry, the contribution culling scheme culls away detail meshes from far away, and includes them when they contribute sufficiently: the building geometry is progressively increased. Protruding features such as balconies naturally suit this system. In addition, this allows increasing the view distance in a natural manner. Distant but significant buildings are included, while partially occluded buildings (even those that might lie close to the view point) are dropped.

[3] In the presence of a *vertex cache*, and if vertices are properly localized, polygon soups can achieve or even exceed the performance of aggregate meshes

[4] With a regular 3D space subdivision, it is likely that nearly the same objects are visible from neighboring cells.

Caching

In out-of-core rendering systems, efficient access to external data along with memory and cache management is critical for system performance. In our earlier system [113], we observed a significant increase in file I/O latencies as the number of individual files in a directory grew, a situation described by **Cox** and **Ellsworth** [54]. On occasion, this effect was prohibitive, even in relatively modern devices. We measured the average speed of opening a file with the `fopen()` call as the number of files on a directory increase on **Nokia** *N93* and *N95* smart phones[5] (not published earlier; see Table 4.1). With 1000 files on a directory on an external memory card, the *N93* could open only five files in two seconds. During the call, all program execution was blocked, including responding to user interactions. Similar increases in latency were observed with multiple different devices and memory cards (all with the FAT32 file system). We expected a high number of individual files to emerge from the pre-process (see Section 4.6). Therefore, a filesystem independent local cache storage was developed.

Function	Files in dir.	Nokia N93 int.	Nokia N93 ext.	Nokia N95 int.	Nokia N95 ext.
`fopen()`	1	667	23.5	667	500
`fopen()`	10	667	24.4	666	500
`fopen()`	100	250	22.0	250	154
`fopen()`	1000	20	2.4	222	125

Table 4.1: The latency of `fopen()` call increases as the number of files in a directory increase. Units are calls in seconds.

The complete caching scheme consists of three cache levels, from a compressed memory cache and local file cache to a content server. The system is presented in Figure 4.3, and described in publication P2. Currently, all cache files are kept open at run time, and completion of writes asserted with `fflush()` calls to avoid file corruption on system hangs or crashes. The data chunks are arranged at pre-process onto a form similar to their internal binary structure. This scheme, familiar from the building walk-through by **Funkhouser** *et al.* [62], minimizes run time parsing. Networking is described further in Section 5.

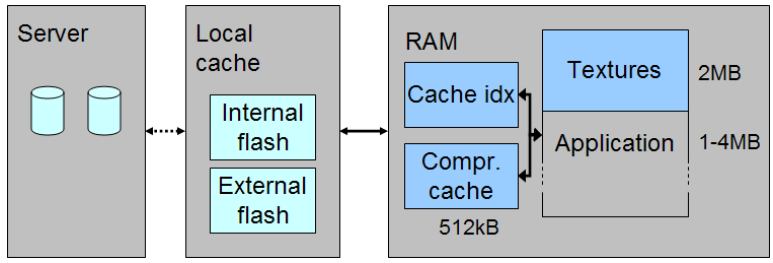

Figure 4.3: Out-of-core rendering and caching (from publication P2). During navigation, data is retrieved from a compressed memory cache, local storage or server. The memory limits are pre-configurable.

[5]Memory cards were **Transcend 1GB 80x MMC** with the *N93*, and **Nokia 1GB MicroSD** with the *N95*.

Increasing realism: a detailed 3D model

Our primary goal for developing a 3D map engine was the ability to increase both the realism and rendering speed in comparison to the early experiments, a goal supported by the feedback from these studies. This required a case model. In the sense of Ptolemy's chorographic principle, we did not intend to create a geometrically accurate model, but one that maximizes the *likeness*, capturing the *minute details*. There were two 3D models available from the case area[6], namely a 3D model created and maintained by the City Survey Division of the Real Estate Department of Helsinki, and a 3D model called *Helsinki Arena 2000* created in a project by the local phone operator **Elisa** [89]. We set our target accuracy for detail to be approximately one order of magnitude better than in the *TellMaris Guide*, namely 10–20cm, which we intended to be achievable with higher resolution textures. The model by City Survey Division was maintained in the *Microstation* CAD system. Both geometry and surface detail varied within the model in the case area. A single sample building consisted of 40.000 polygons with inconsistent surface normals. Furthermore, we could not export the textures along with the model. The *Arena 2000* model was consistently created and contained multiple levels of detail, but was completely void of textures. Our solution was to build a case model from scratch.

As discussed in previous sections, some visibility culling methods, such as Naylor's BSP tree projection [106], pose considerable requirements to 3D models. We chose to support a common 3D model boundary representation format with less strict requirements, and without being tied to the format's particular features nor to any features of available model viewers. We observed that VRML was well supported in several 3D modeling packages, and a model built with *3D Max* was capable of sustaining consistent normals, so we chose this combination for creating our reference model. Our approach for using the VRML format was in the spirit of COLLADA, by incorporating the model into our production pipeline, optimized and suited for our particular case (see Section 2).

Publication P1 provides the details of our modeling practices and designs. We reprise them here:

- Consistent normals. Every building surface points outward.
- A flat topography. Not a restriction of our engine, the decision was supported by previous observations of urban environments [163].
- Large scale landmark structures modeled with plenty of detail.
- Relatively simple ordinary buildings modeled with textured outer façades and plain colored rooftops.
- Most distinguishable medium scale features chosen by the modeling person.
- Local salient features such as statues represented by a single distance- and view-independent billboard. This decision was not supported by the observations of **Pasman** and **Jansen** [121]. In our case, we considered recognizability and lesser rendering or transmission cost to be

[6]The m-LOMA project was funded by EU InterregIIIA, requiring the focus to be on Southern Finland or Northern Estonia. With City of Helsinki as a partner, Helsinki was the only viable option for the case area.

more important than perceivable error.

- Hierarchy. Façades and rooftops were bound to buildings, and buildings to blocks (see figure 4.4). This structure poses a similarity to the *urban data model* (UDM) by **Coors** [47].
- A *mesh* as an atomic unit for the model, defined with VRML's `IndexedFaceSet`.

Our 3D map engine utilized the model hierarchy in frustum culling, and meshes in visibility determination (see Section 4.5). The presented design issues were subjected to validation by field experiments (see Section 6.6).

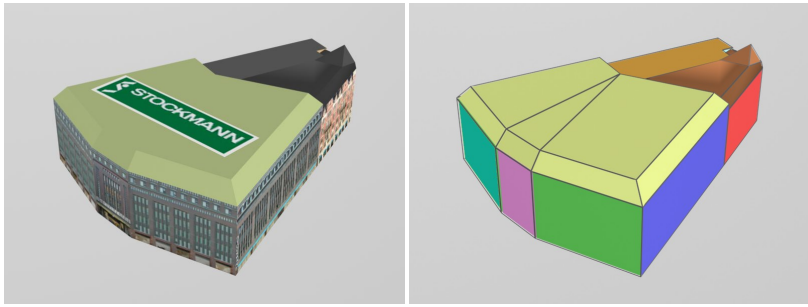

Figure 4.4: Model hierarchy and simplicity of modeling: individual meshes (façades and rooftops) constitute a building, where detail is provided by textures (from publication P1).

The pre-process pipeline

Our entire final pre-process pipeline consists of the following steps, implemented with C/C++ (elaborated from publication P1):

1. VRML parsing. Assigns ID's to meshes, determines bounding volumes, calculates mesh normals and stores the mesh hierarchy. Also stores landmark annotation (if present).
2. Texture processing. LOD texture generation into JPG or PNG files, JPG and PNG header cleaning, dominant color selection for lowest LOD. Optional: decomposition into segments with recomposition rules.
3. Visibility determination with hardware-acceleration, shared with networking.
4. Visibility list encoding.
5. Packaging data onto cache files.

Runtime out-of-core rendering

At run time, most of the client side logic related to 3D rendering lies in data management and culling processes. The following presents the main stages of rendering, modified from publication P1 (dynamic entities are discussed in section 5):

1. Fetch a visibility list

2. Fetch potentially visible meshes
3. Determine visibility of meshes with frustum culling
4. Select LOD textures based on distances of visible meshes
5. Optional: render a skybox
6. Optional: render a ground plane or ground meshes
7. Render landmarks with Z buffer writes disabled
8. Render meshes and billboards with Z tests
9. Render the user interface components (see section 6)

All data is managed through the cache system presented in figure 4.3 and discussed in section 4.5. Rendering was implemented with C/C++ and OpenGL ES 1.0 Common profile. Prior to rendering, meshes are depth-sorted in closest-first order. This reduces texturing of occluded parts of meshes, speeding up rendering (see Table 4.6). It also accelerates collision tests, which can be performed to the bounding volumes of the closest meshes against the motion vector.

The first versions of the system loaded all meshes and textures immediately when needed. However, this affected interactivity due to the I/O latencies. To improve interactivity, data fetching from local storage is currently minimized during maneuvering. When movement stops, the system starts loading data. When rendering is required due to user interaction or for updating dynamic components (see section 5), missing data is not waited, but rendering starts immediately with the data that currently resides in memory.

The system features a textured sky for increased yet artificial realism, which is not necessary for navigation. By default, it is turned off on mobile platforms, replaced by a clear blue color. The separate rendering pass for landmarks utilizes the landmark information gathered either from the VRML model or from later annotations by a system administrator. For each major landmark, a billboard must exist, replacing the landmark at long distances. The landmark rendering pass takes place before the building mesh rendering pass, and does not perform the Z test nor update the Z buffer.

The ability to render incomplete scenes differentiates this system from some of the previous systems. The urban walk-through by **Marvie** and **Bouatouch** [95] required the full currently visible data set to be loaded into memory before navigation or rendering was possible. The system by **Funkhouser**, **Séquin** and **Teller** resembles our system with several similar solutions. However, it featured pre-fetching to have the data ready when needed [62]. Pre-fetching is typically based on motion vector estimates, which are available only when the viewpoint is moving (or when the navigation space is limited and topological extrapolation of the potential navigation direction is feasible). Our observation on lost interactivity due to slow I/O is in contrast to the idea of pre-fetching during movement. We keep the currently visible meshes and textures in memory, and secondarily optimize what could be visible in near future. However, this would require external information on the expected behavior of the virtual camera (if we load data only when the viewpoint is stationary, the motion vector is not available). Due to these reasons, we manage the *existing* data based on the *least recently used* algorithm, with emphasis on saving small data chunks to minimize file accesses.

A 3D city model

Our case 3D city model consisted of 183 uniquely textured buildings, covering the entire center of Helsinki, presented in figure 4.5. The buildings were built with 1029 meshes, totaling in 12610 triangles. 471 textures were created, to the minimum texel scale of 10–20cm. 143 of the textures reached the texel size of 5–10cm. The textures were created with digital photography with ortho-rectification and removal of artifacts, such as cars, wires, trees etc. in image editing software. All 14 statues from the area were created with a similar method, as billboards. The size of the original VRML file was 14.3MB[7], but the preprocessed binary set of meshes was only 714kB (including the precomputed bounding boxes, normals and façade colors without further compression). Model veridicality is discussed in section 6.

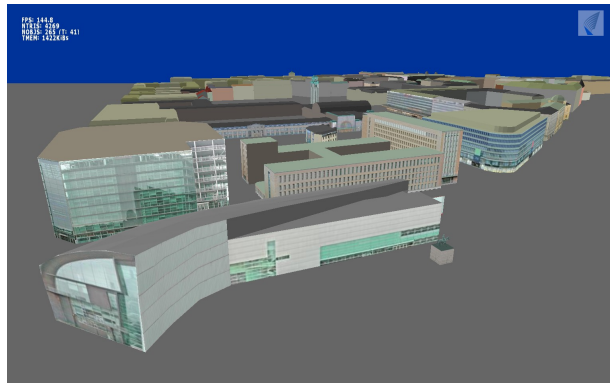

Figure 4.5: The 3D model of Helsinki, consisting of approximately 200 textured buildings and 133 "dummy" buildings. Façades visible to pedestrians were textured.

Efficiency of the 3D engine

The first version of the pre-process took a bit less than a day on an **Athlon** *3500+* machine with a **NVidia** *6800GT* graphics card. Since that, several improvements have been made, as foreseen in publication P1. For example, the original version calculated visibilities for shared vertices between view cells redundantly. Currently, the pre-process of the recent, larger model (210 textured buildings, 133 non-textured buildings, composed of 1874 meshes, 13378 triangles and 500 textures) takes 44 minutes on an **Intel** *Core i7 920* (2.667Ghz) machine with an NVidia *GeForce GTX 285* graphics card and 6GB or RAM. The output of the pre-process yields (with typical configuration) 2000 textures (4 LOD levels of each texture), over 1000 meshes and approximately 10000 PVS clusters, packed onto 140 cache files.

The PVS culls, as expected, a lot of geometry at street level (typically over 90%) and approximately half at the sky (see Figure 4.6 for the street

[7]Recent VRML exporters have yielded more compact files.

level efficiency). Frustum culling reduces roughly half of the remaining meshes. In the worst case, when having a view over the entire city, 72% in total is culled, and in the best case, at street level, as much as 98%. Example views from these positions are presented in Figure 4.7. The different views portray the different culling characteristics, where PVS dominates at street level, while frustum and contribution culling dominate at long distances, and frustum culling dominates at top-down views. In all cases, PVS culls over 50% of the meshes. 4MB was sufficient in all tested mobile devices to hold the application along with the textured façades in all situations. For **Nokia** *n-Gage*, with only 3.4MB of memory, view distance was limited to 300m and maximum resolution of textures to 128x128. This also helped in increasing the rendering speed of the n-Gage to interactive levels.

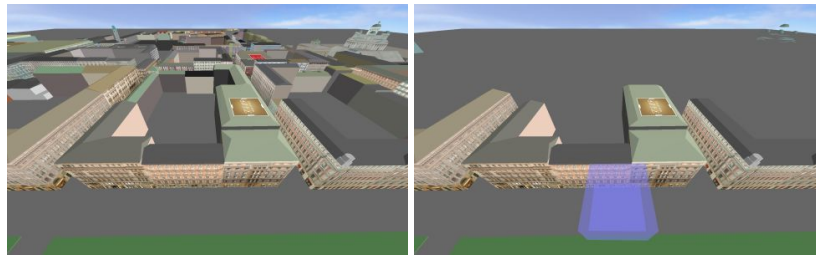

Figure 4.6: Visibility culling. The potentially visible set of the blue cell (right) contains only 23 out of the 264 meshes visible to the virtual camera (left).

Figure 4.7: Example views. (left) street-level view (1063 triangles in 71 meshes, 25 meshes textured), (middle) rooftop view over the city (2803 triangles in 246 meshes, 36 meshes textured) and (right) top-down view from the sky (3095 triangles in 235 meshes, 32 meshes textured).

Device	Laptop	PDA/6630	n-Gage
View distance	800	500	300
Texture max res.	512x512	256x256	128x128
HVS	1 pixel	1 pixel	1 pixel
PVS Resolution	768x768	320x320	208x208
VRML file size	14.3MB	14.3MB	14.3MB
Binary model size	714kB	714kB	714kB
PVS size (raw)	50MB	20MB	10MB
Clustered PVS	9MB	5MB	3MB
JPG textures	8.5MB	5MB	2MB

Table 4.2: 3D model pre-process settings and statistics for the first generation m-LOMA engine.

Meshes/position	Total	In PVS	In frustum	Culled%
Street	1029	40-100	20-60	95-98%
Crossing	1029	120-160	40-80	93-96%
Rooftops	1029	400-500	150-250	86-77%
Sky	1029	500-650	200-300	82-72%
Sky, looking down	1029	500-650	50-100	90-95%

Table 4.3: Culling efficiency statistics (a 90° field of view and 4:3 aspect ratio).

Rendering speed and details

With a view distance of 500m and maximum texture size of 256x256, the rendering speed reaches interactive rates on all our primary devices (see Table 4.4)[8]. The rasterization hardware of *N93* and *N95* boost the speed by an order of magnitude, to 60fps. Furthermore, hardware is not much affected by texture resolution, while there is a penalty for larger textures for software rasterizers. Texture detail on roughly 30% of façades is better than 10cm. Typically, 20–40 closest façades are textured. With four texture LODs, the lowest LOD (single colored LOD) is reached at approximately 300m.

Platform	Dell M60	Dell X30	Nokia 6630	Nokia N93/N95
Resolution	1024x768	240x320	176x208	240x320
Startup time	<5s	5s	16s	7s
Street	300-600fps	20-25fps	10-20fps	100-300fps
City overview	200-300fps	10-15fps	5-12fps	50-100fps
In sky, looking down	200-300fps	10-16fps	6-14fps	50-100fps

Table 4.4: Approximate rendering speeds for the "PDA/6630" preprocessed data set on typical platforms. The measured rendering speeds are based on averaged rendering speeds of selected view positions without file I/O or texture decompression and are not limited by actual display refresh rates.

Comparisons to earlier systems

For this thesis, **Rakkolainen** and **Vainio** kindly provided us both their original models that they utilized in their *3D City Info* project [133, 158]. The original "Tre-D" model was nearly comparable to our Helsinki model in size and detail. The model contained a single, large terrain, which we split to smaller pieces, including the texture. We separated building surfaces from terrain surfaces, and re-grouped them onto a hierarchy. We scaled textures to

[8] In practice, experienced update rates are slightly less, depending on allocated memory – with less memory, more data needs to be swapped in and out at run time.

closest compatible resolutions ($2^n x 2^m$), and removed our flat ground rendering pass. After our pre-process, both the full sized and mobile 3D City Info models ran in 624MHz *Axim X30* devices at interactive rates with software rasterization, and in *N93* and *N95* with hardware acceleration over 30fps, in only 4MB of memory (see Table 4.5). Our collision avoidance scheme automatically kept the viewpoint from penetrating the ground.

Prototype	3D City Info (full)	3D City Info (mobile)	m-LOMA 3D Helsinki
Triangles	8709	1572	12610
Textures (raw size)	260 (39MB)	16 (3.4MB)	500 (88.5MB)
Buildings	194	47	210 textured+133 plain
N93/N95 fps	50–300fps	50–300fps	50–300fps
Axim X30 fps	12–16fps	20–30fps	10–16fps

Table 4.5: Comparison of the 3D City Info model ("Tre-D") and the latest m-LOMA Helsinki model running on mobile devices. The rendering speeds do not include file I/O or texture decompression.

Direct comparison to the rendering rate reported by Rakkolainen *et al.*'s 3D City Info (0.125fps at street level on 133Mhz 32MB PDA) is not possible due to lack of a similar device. However, indicative performance can be deduced by 3D benchmarking. We have performed a set of measurements on our OpenGL ES benchmark software on various devices during development[9]. The following results have not been published earlier. Table 4.6 provides benchmarks of 3D matrix computations, and rendering Gouraud shaded and textured triangles with two triangle sizes and two texture resolutions on various devices[10].

The benchmark results vary greatly. CPU clock speeds do not always correlate with rasterization and 3D matrix operation speeds. For example, the 206Mhz *iPAQ H3800* renders textured triangles *faster* than the 400Mhz *PocketLoox*, which outperforms it in 3D matrix operations. The differences between 3600 and 7200 pixel triangles indicate linear fill-rate dependency. These results also demonstrate the importance of the memory subsystem, which is the key to high rasterization speeds [102]. Unfortunately, memory subsystem specifications are not generally available for mobile devices.

Combining the results, we approximate that the 206–400MHz PDAs deliver half of the fps reached with the 624MHz *X30*. For the original PDA used by Rakkolainen *et al.*, with a 133Mhz CPU, we estimate that it would probably perform at one quarter speed in comparison to the 624Mhz *X30*. However, Rakkolainen *et al.* used half a screen resolution for 3D rendering, which, based on our benchmarks, is twice as fast to render as full screen. Therefore, our best estimate is that the full Tre-D model would yield 6–8fps, and the mobile version 10–15fps on the original device, where the highest fps is achieved at street level. This is *two orders of magnitudes faster than was achieved with direct model viewing*. 3D rasterization hardware yields nearly another order of magnitude increase, but rendering in this case becomes CPU limited.

Other early models were not as detailed as the Tre-D and our Helsinki

[9] Our software rendering tests are all based on *Gerbera* rasterizer by **Hybrid Graphics**.

[10] The CPU speeds of the Nokia n-Gage and the 6630 are fetched from Internet sources [110, 10]. The specifications of the other devices are available on manufacturer sites and users' manuals [108, 109, 44, 91, 43].

models with the possible exception of the model demonstrated by **Blechschmied, Etz** and **Haist** [21]. Unfortunately, their run time client was only able to present a few low resolution textures, and model details were not reported. Given the similar PDAs and Cortona VRML libraries used in the other early prototypes, we expect a rendering speed increase of at least an order of magnitude with our engine, supported by an efficient rasterizer such as *Gerbera* by **Hybrid Graphics**.

Platform	Nokia n-Gage	Nokia 6630	Nokia N93/N95	HP iPAQ H3800	PocketLoox	Dell X30
CPU	104MHz ARM-9	220Mhz ARM-9	332MHz ARM-11	206MHz StrongARM	400Mhz PXA250	624Mhz PXA270
glTranslatef();	75294	102400	186350	171086	252525	416667
glRotatef();	23748	44444	28760	56956	103923	142959
glMultMatrixf();	21459	39938	49868	39940	78555	107181
3600p gouraud tris	2783	4476	144928	1709	4090	6468
7200p gouraud tris	1576	2452	100000	1255	1699	3565
3600p 256x256 tex tris	857	1939	120000	1281	1062	1922
3600p 128x128 tex tris	1079	2370	120000	1401	1128	3372
7200p 256x256 tex tris	497	1044	81081	747	503	1075
7200p 128x128 tex tris	573	1289	83333	757	689	1734

Table 4.6: Selected OpenGL ES benchmarks using **Hybrid Graphics** *Gerbera* rasterizer v.2.0.3. and v1.2.7 HW PowerVR MBX driver (N93/N95). Units are calls in seconds.

3D Model sizes and file formats

Rakkolainen *et al.* identified the model size as a key problem. However, they were looking forward to X3D as the future model standard, whose viewers could be "particularly suitable for mobile devices". On the other hand, the suitability of an XML-based format for mobile use was not expected to be very good in *TellMaris Guide* [24]. Recently, the issue was put to the test: **Mulloni** *et al.* created an OpenGL ES *Common-Lite* -based building walk-through system utilizing X3D, on a 624MHz **Dell** *Axim X51v* with 64MB memory. In their final version, they discarded all textures due to slow file I/O, and reported that "mean loading time for the smaller cells is about 2.5 seconds (1 second for file I/O and 1.5 seconds for parsing)", where the number of triangles per cell varied from 102 to 452 [103]. Our binary model is considerably reduced in size in comparison to the original VRML file. Due to the machine friendly binary format, the parsing time of a set of 1000–4000 visible triangles is negligible, consisting of `memcpy()` operations taking less than 1ms. Most of our time spent in swapping operations is consumed in texture decoding, but without hindering interactivity. Due to these reasons, we believe that our machine compatible binary format is viable and quite suitable for mobile use.

5 NETWORKING AND DYNAMIC ENTITIES

Mobile networking opens interesting possibilities for 3D maps. First, static models can be downloaded when needed, on the fly. Second, annotations and location-based information can be similarly transmitted. This data can be temporary, emerging and expiring at run time. Third, by near real time updates, the world can be given life by populating it with moving, dynamic entities such as vehicles and people. In this chapter, we address all these issues, based on publications P3 and publication P5.

5.1 Internet in mobile environments

During the early 3D map experiments, researchers noted that network capacity is one of the main bottlenecks for 3D model transmission [133, 139]. We therefore have a look at the properties of Internet Protocol (IP) networks [125] and the protocols available via the *socket* interface, TCP [126] and UDP [127], and measure the performance and characteristics of the cellular networks (for these concepts, refer to Section 2.3).

Both TCP and UDP provide packet transmissions between two hosts in the Internet. TCP provides reliability with the cost of increased round-trip times (RTT) and larger headers, acknowledging all packets automatically. Acknowledgments are asynchronous. Several packets can be sent consecutively before receiving acknowledgments. This mechanism is called the *sliding window* paradigm, and is intended to minimize the effect of long RTTs that would render synchronous transmissions prohibitively slow. TCP also provides network congestion control by automatic transmission speed adaptation, where network speed is gradually increased or decreased depending on the network's congestion. This gives rise to the *slow start* effect, where each TCP connection is first assumed to be slow, and speed is increased as acknowledgments arrive. Due to the slow start, non-persistent connections, such as HTTP v1.0, cannot fully utilize the available network capabilities unless large amounts of data are transferred. For example, every object on a web page requires a separate TCP connection, each suffering from this effect [120].

UDP, being simpler than TCP, only guarantees the intactness of transmitted data with a checksum, but does not ensure that each packet reaches its destination. For reliable UDP transmissions, application level logic is necessary.

In general, TCP provides a convenient interface to programmers. However, our intention is to find out which of the two methods is best suited for mobile use, and for our case. This choice depends among other things on network characteristics, such as packet loss, where some data is lost during transmission. If a connection suffers from a relatively high rate of packet loss, TCP's transmission control properties, assuming the cause to be congestion, decreases data rates, sometimes too radically. In addition, retransmissions of data tend to add to the problem. For systems where response times are critical, UDP is a better choice as the application receives the data

without waiting for packet arrival acknowledgments. For example, typical 3D first-person shooter games use UDP. First of these was **ID Software**'s *Quake*, which mainly used *unreliable*, compressed UDP packets containing full game states for each game frame on the server-to-client direction. A dropped packet did not need to be resend, as a new one followed immediately at the next simulation step [6, 23].

Publications P2 and P5 provide insight to the IP framework and assert the performance of the "2.5G" (GPRS/EDGE), "3G" (UMTS) and "3.5G" (HSDPA) networks. The results are presented in Tables 5.1, 5.2 and 5.3, and Figures 5.1 and 5.2.

The RTTs of GPRS/EDGE connections are high (400-800ms), but the connections do not suffer from excessive packet loss. Data rates vary between 10–15kB/s (downlink) and 5–11kB/s (uplink) (Table 5.1). The RTT of "3G" connections is rather stable, approximately 150ms. The uplink and downlink speeds yield a constant 40kB/s, except for the older device (*6630*) (Table 5.2). The slow start property of TCP is evident, with maximum network capacities reached after the first 100kB is transferred (Table 5.3).

The "2.5G" and "3G" results are from good conditions, measured from a static point. The "3.5G" results provide insight to the real world situation with varying signal strengths as the device is moving in the environment. During the test, the speed varies from zero to near theoretical maximum, 3.6Mbit/s, sometimes staying at older generation levels (Figures 5.1 and 5.2).

As the connections do not suffer from packet loss, and our scenario does not require near instantaneous interaction, TCP would suit our needs. Due to the slow start of TCP, a persistent connection is essential. The network speed exhibits great variation, so the application programmer cannot trust in the highest speeds advertised by operators.

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 5.1: 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 5.2: Network statistics for a 3G (UMTS) network in good conditions.

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

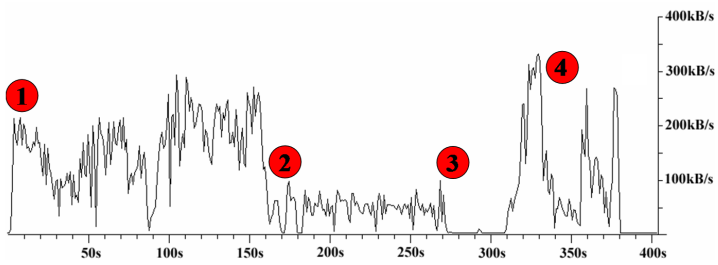

Figure 5.1: Cell network speed. 3.5G network speed ranges between GPRS/EDGE (0–16kB/s), UMTS (16–48kB/s) and HSDPA (48–384kB/s) during the first 400 seconds of use. The speed differences account for varying signal strengths and network load as the user navigates, while a content server sends as much data as possible. The path of the user is shown in figure 5.2.

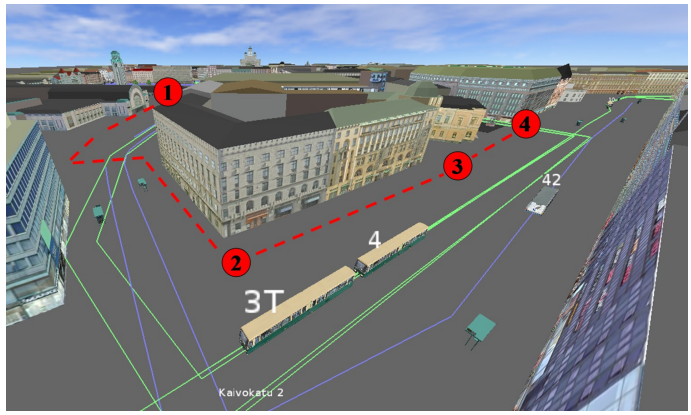

Figure 5.2: Path of the network speed test of fig. 5.1 and public transportation.

5.2 Lightweight, efficient mobile communications

When model and texture sizes exceed several megabytes and the wireless networking potential might be only a few kilobytes/sec, an efficient delivery method is necessary. Common high level middleware protocols, such as SOAP, which rely on XML enveloping, tend to lead to excessive over-

head, which is considered rather prohibitive in mobile map applications [24, 155, 123]. Attempts at standardizing network protocols for virtual environments have been made. For example, **Brutzman** *et al.* provided designs for *Virtual Reality Transfer Protocol* (VRTP), to link virtual environments in the manner similar to HTTP linking web pages, combining "multiple existing dissimilar protocols (such as HTTP, DIS, multicast streaming, Java agents, network monitoring etc.)" [33]. Such generic purpose VE network protocols have not been adapted in general. Certain model formats, such as M3G, support serialization for networking [131, 2, 130]. However, our mobile client is not a model viewer, but an application utilizing the optimization methods discussed in previous chapters. Consequently, a proprietary solution is necessary.

Publication P2 defines a TCP-based network protocol suited for mobile 3D maps. The protocol data units (PDUs) are described with XML, which is tokenized onto a binary form with a Python script, outputting serializers for both C and C++. The PDUs consists of *messages*, which can have attributes, specified as *fields* with the appropriate field type. The length 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 5.4 presents the XML description for a *mesh* request and a response. The parse time of the binary protocol is negligible.

```
<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 5.4: XML specification for a mesh request and response.

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. The client serializes the requests, 400x8 bytes = 3200 bytes, and places it onto a *send buffer* (Figure 5.3). When flushed to the TCP socket, only a couple of full TCP packets are generated[1] and sent in a fraction of a second. The response is received asynchronously, and consists of 400 response messages. With an average mesh size of 512 bytes, 200kB would need to be transferred. The overhead im-

[1] The maximum transfer unit (MTU) of TCP is 1500 bytes.

posed 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 (in HSDPA networks at full speed, a second would suffice), managed and controlled by TCP. 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.

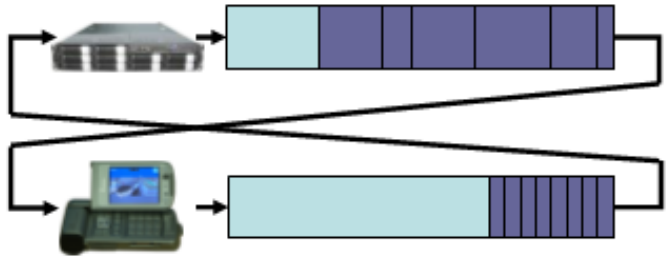

Figure 5.3: Pipelined, asynchronous request-response networking with send buffers.

The benefits of our protocol, in contrast to other lightweight protocols, say, JSON [77], include very low overhead, light-speed parsing, capability to send binary data such as textures, send buffer management, including control for maximizing TCP packet size, and asynchronous data transmission.

5.3 Networked delivery of 3D content

The 3D content download strategy depends on the application and the underlying 3D engine, or in case of a simple model viewer, on the 3D file format. For example, in VRML, the whole model with all its details need to be downloaded completely prior to rendering, while the M3G provides direct support for serialization. In general, progressive transmissions are possible only if the data is serializable.

Hesina and **Schmalstieg** considered demand-driven geometry transmission for distributed virtual environments [141]. Similarities 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. Parts of the model around an area of interest (AOI), defined by a radius, are downloaded with a prefetching scheme. This case, and the follow-up [69], do not address texturing.

Pasman and **Jansen** examine geometry simplification methods for augmented reality systems [121]. 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.

Prior to emergence of mobile 3D hardware, the possibility of *remote rendering* was considered to provide the rendering services at the server side. In this case, the mobile device acts as an interface, where the maneuvering

commands are transmitted to a server, which renders the scene and sends back the resulting images. This method induces latencies and scalability problems. The minimum latency is the round-trip-time of the network, and the transmission time for the payload (the image or the encoded stream fragment). For example, with a 20kB average frame size and 40kB/s network speed, the latency would be 150ms + 500ms = 650ms, without the contribution of rendering and encoding. **Quax** *et al.* have examined this possibility from the viewpoint of encoding, where a single encoding server with a dual core 3GHz CPU could serve 25 clients [132]. We consider the latencies prohibitive for interactive use, and aim for a scalable system where servers tend not to become computational bottlenecks.

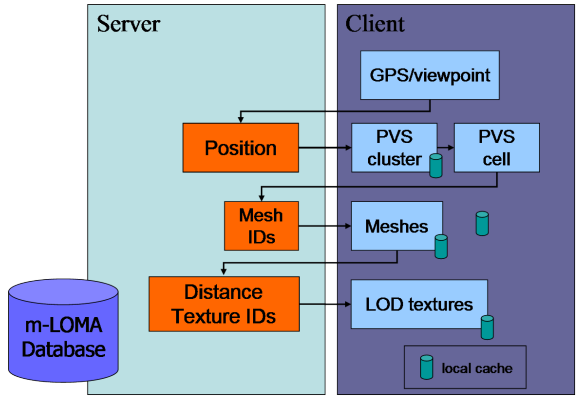

Figure 5.4: A progressive visibility-based static 3D map data download scheme. Received data is stored to local cache files for fast recovery.

Figure 5.4 presents our visibility-based progressive download scheme, merging the descriptions provided in publications P2 and P5 (with minor modifications on the figure published in [112]). Based on the current position of the viewpoint, a visibility cluster is fetched. Then, the potentially visible meshes are fetched, followed by the LOD textures required for the current view. Rendering is asynchronous, where geometry and detail is added to the scene as they arrive. Table 5.5 presents observations during the progressive download on "3G" networks. The observations depend on transmission speed. With 48kB/s, an initial view over the city is populated fully in less than half a minute. When maximum speed of 3.6Mbit/s of the "3.5G" networks is available, the view is populated fully in a few seconds.

Our system has certain resemblance to that of Hesina and Schmalstieg, as we transmit our model with levels of detail. The model geometry is incremented with our contribution culling scheme. At client side, common LOD geometry is supported in principle as mesh IDs can change between view cells. Different IDs could reference to different LODs of a single model. However, our reference model only contains static LOD buildings with some small and medium scale details that can appear as contribution culling threshold is exceeded. We explicitly support LOD textures, which have been preprocessed to LOD levels. However, instead of an AOI, our fetch is based on predetermined potential visibility. We have approximated

our statues with billboards, but use a single view independent impostor instead of constructing them on the fly at the server as in Pasman and Jansen's scenario. We do not perform remote rendering of any kind to avoid server side computational bottlenecks.

Time	Event	Data transferred
0–5s	Authentication	1–10kB
5–10s	First buildings appear	10–100kB
10–15s	Most buildings present, some textured	100kB–1MB
15-20s	All visible buildings present, nearest ones textured	200kB–2MB
25s–1min	All visible buildings textured	300kB–10MB
1min–	Possible pre-fetching of larger areas	

Table 5.5: 3D city download in cell networks (20kB/s–2Mbit/s): a user's point of view.

5.4 Location-based information

3D maps can act as a gateway to location-based information. Traditional mobile map-based information and annotation systems are widely research and implemented. For discussion on such systems, refer, for example, to [14, 98]. Our contribution focuses on visibility-based culling issues to minimize network traffic and on using the hierarchy of the 3D model as a higher level grouping mechanism.

Figure 5.5 presents the idea behind annotation and location-based content culling. All such data is associated to façades or street segments, and independently of their representation (in the figure, billboards are used), they inherit the visibility properties from the mesh they are attached to. Now, when a surface is visible to a client, the client can query or subscribe onto the associated data based on the mesh ID. By a subscription, all new annotations are received in near real time as they are created. Traditional user profile filtering can be applied at server side for further culling. Annotations and location-based content is managed similarly. This data is stored to local caches along with geometry and textures, but can possess an expiration date.

The model hierarchy can link entire buildings or building clusters such as a large shopping mall to a single *mother node*. Any reference to a mesh can then be linked to the mother node's content, possibly presented as services and annotations.

5.5 Dynamic entity management

Challenges

In networked virtual environments (NVEs), multiple users and other dynamic entities share a common world. NVEs range from text-based games such as *multi-user dungeons* (MUDs) to immersive, *collaborative virtual*

Figure 5.5: Approximate location-based information culling with façade visibility. Annotations "inherit" the visibilities of the surfaces to which they're attached.

environments (CVEs). The most popular form of NVEs is networked 3D gaming, including *massive multi-player online role-playing games* (MMORPG) such as the *World of Warcraft*, and *first person shooters* (FPS) such as the *Counter-Strike*.

The main challenge in NVEs is related to the scalability of the shared environment. There are two common bottlenecks. First, the system may face computational limits. This is typical in client-server systems, where the server is needed for decision making to ensure consistency and persistence of the virtual world. For example, as a near simultaneous action, two clients may claim to occupy the same spot or resource. The physics of the virtual world may also need to be guarded by the server. In this case, the server performs a full world simulation for each client, and eventually saturates as the number of clients increase. Typical cases are first person shooters, which provide high interaction between clients, and where server-to-client traffic increases linearly with the number of clients [59]. The system architecture typically limits the number of simultaneous FPS clients to 32–64.

The second potential bottleneck is the network. On peer-to-peer (P2P) systems, simulation is distributed to the clients, but they need to send their state updates to other clients. As the number of clients increase, so does the network traffic. The very first large scale NVE, the *SIMNET*, was a combat simulator for small unit interaction. In SIMNET, and the follow-up *NPSNET*, *Players* were expected to be *honest*. The Players were responsible for their part of the simulation, submitting their own locally calculated states and effects of interaction, such as hits on enemy vessels, to other Players. To minimize network traffic, SIMNET Players only transmitted state updates when a Player's extrapolated (dead reckoned) state (their *Ghost*) exceeded an error threshold in comparison to the real one [20]. In NPSNET-IV, state updates were distributed among all Players, saturating a 10Mbit/s Ethernet with about 300 Players. In addition, each Player had to calculate the Ghosts of all other Players to run the simulation, causing also a computational bottleneck.

To increase scalability of NVE systems, *interest management* was introduced, to filter communication based on interest expressions, for example geographically (area of interest) or functionally (a tank being interested in

ground vehicles) [136]. Later NVEs have utilized spatial localization in message filtering, and the concept of *communication visibility* [36].

Solutions

Due to their dynamic nature, visibilities of dynamic entities cannot generally be predetermined. **Chrysanthou** and **Slater** assert three requirements for an algorithm intended for managing dynamic 3D scenes [40], the abilities to

1. Change the camera view
2. Add objects to the scene
3. Delete objects from the scene

A dynamic entity is then managed by deleting it from the old position, translating it, and inserting back to the scene. This introduces significant overhead, a problem long recognized. In addition, entity visibility would need to be determined for each frame. This requirement is relieved by **Sudarsky** and **Gotsman**'s *temporary bounding volume* (TBV), which contains the object during a *validity period* [150]. During this time, run time visibility calculations rely on the TBV. A TBV is created based on a priori knowledge of the object's behavior. For objects with well-known trajectories, sweep volumes can be used. However, the method does not solve the visibility problem. In addition, the validity period depends on the motion of the viewpoint, namely user interaction, which cannot be predicted accurately. In addition to the discussions on publications P3 and P5, it should be noted that if visibility would be used as a communication culling mechanism in NVEs, the message passing server should perform visibility determination processes for each client at run time, causing a severe computational bottleneck, unless predetermination methods are applicable.

Our solution to the dynamic entity management problem is discussed in publication P3. Our algorithm exploits topologies in urban environments, where the navigable space of real world entities such as people and vehicles is limited to the street network and open spaces such as parks. **Whiting** *et al.* have performed similar topologization of CAD models for navigation purposes, including indoors [161]. However, they did not adapt the system for visibility pre-computations. In our system, street segments, crossings and open areas are assigned virtual cells covering physically navigable space (Figure 5.6). These virtual, static cells are included in the PVS pre-computation as atomic units along with the building façades and rooftops. At run time, for visibility determination, pedestrians and vehicles are mapped to the closest street segments or open areas, which they *occupy*. Now, when a virtual cell is potentially visible, we assume that the occupying entities are visible as well, in the to-region visibility sense. Message passing is managed by these visibility tables based on a publish/subscribe system. Dynamic entities publish their position, and clients subscribe to visible virtual cells. A message passing server acts as a switchboard, managing a simple subscription table.

Message updates are minimized in the manner similar to the Player/Ghost scheme. As long as a dynamic entity acts according to its behavioral function, updates are not necessary as the state can be extrapolated. We calculate the maximum time between updates with a *view independent*

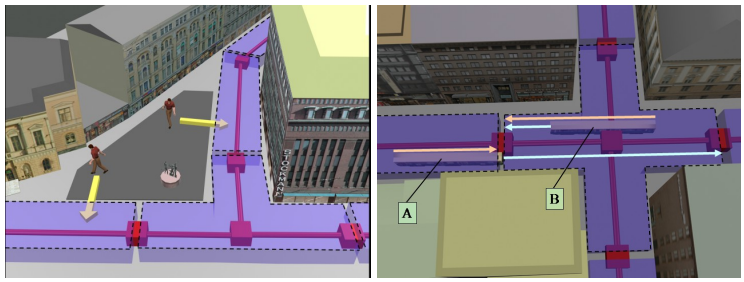

Figure 5.6: Exploiting street topology for entity management. (left) Pedestrians are projected onto a street network for approximate visibility determination, (right) cell occupations and validity periods for trams.

occupation validity period, which describes how long a given entity is likely to stay in a view cell. For large entities, an update is triggered when ever the entity leaves a cell, or part of it arrives at a new cell (the front and aft of the tram depicted in Figure 5.6). Our scheme is in contrast to the TBV, where the shape of the TBV depends on the entity motion, and its validity period depends on viewpoint's motion. The presented mechanism pushes the update logic entirely to clients, as was the case with NPSNET.

5.6 A scalable networking system for dynamic mobile 3D maps

Publication P2 combines the networking efforts to a system overview. Figure 5.7 presents the overall architecture, where static data, including 3D models, road topologies, and location-based content, typically presented in heterogeneous formats, is parsed and preprocessed at an interface layer, and placed to a database. At run time, data is contained in an efficient, binary format suited for our system. We view all external data as digital assets, independently of their format. This approach is similar to the COLLADA scheme discussed in Section 2.1.

Figure 5.8 (from P5) presents the entire run time system. For scalability purposes, all static content services can be parallelized and spatially limited to multiple servers. As rendering is performed locally, only the network speed becomes a limiting factor for content delivery. Similarly, all computations related to dynamic entities are pushed to the clients, or parallelizable proxies. Entity servers act as fast switchboards, forwarding messages initiated at clients to the subscribers. Scalability is facilitated by spatially limited entity servers, which subscribe to each other's entities at borders. The entire system provides a roaming and authentication service, that directs the moving clients to appropriate content or entity servers.

The current implementation is set up on a single combined server. A public transportation simulation based on the first version of the system has been running on a local science park over two years continuously with a simple interface proxy for mapping SOAP position messages to an internal binary format, achieving 97-99.7% compression ratios by discarding the excessive XML envelopes (described in publication P3).

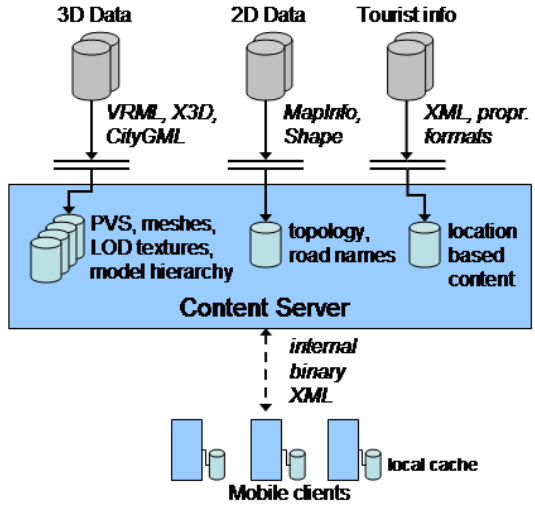

Figure 5.7: System interfaces. External interfaces accept various formats, while internally the data is stored and transmitted in compact form.

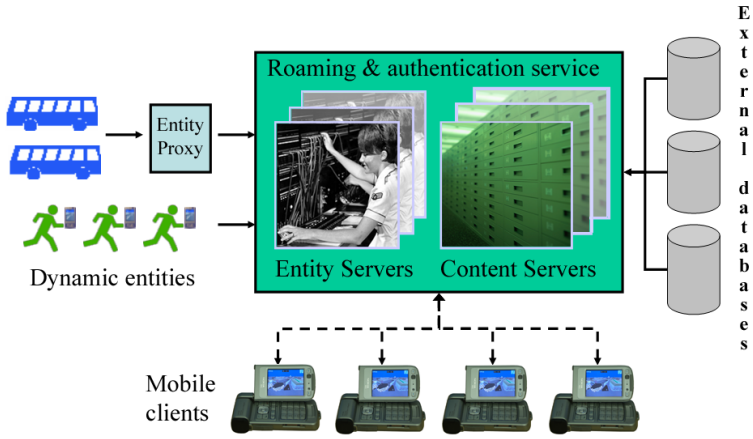

Figure 5.8: Static and dynamic 3D map services. Static content is preprocessed and distributed by independent, replicable servers. Dynamic entity servers act as switchboards. The number of servers can be increased to cover large areas, but neighboring servers must exchange information.

6 TOWARDS A MOBILE 3D MAP USER INTERFACE

Mobile 3D maps have been hypothesized as being *intuitive* in contrast to the abstract 2D maps due to the supposedly shorter cognitive distance between the real world and the 3D representation, which attempts to replicate it. With a 3D city model built for realism and an efficient 3D engine suited for mobile devices, this can be put to the test on the field. However, in order to use the 3D representation during navigation, users must be able to navigate in the 3D model. We expect that the potential intuitivity of the representation may not be transferred to users, unless the user interface - the navigation interface - is intuitive as well. A 3D representation, realistic or not, poses significant challenges for mobile use. The number of degrees of freedom would require a high amount of inputs for a full control, while in mobile phones, inputs are rather limited, and on the other hand, full control may not be the best alternative for easy maneuvering.

This chapter discusses approaches for developing a 3D navigation interface, focusing on users. This development is incremental. Instead of simply creating a 3D navigation interface and measuring its performance on the field, we attempt to understand the phenomena related to navigation itself, including users' needs, navigation tasks, and the differences between the 3D and traditional map representations. We first gather empirical data from 3D navigation with a deliberately straightforward 3D maneuvering interface. We analyze the results for possibly emerging common navigation patterns, apply this knowledge for an improved interface, and perform another experiment, in hopes of conclusive results.

The details of the formal experiment set-ups and their direct quantitative results are out of the scope of this thesis, and can be found from [116, 114]. In our context, we focus on the findings relevant for developing a 3D navigation interface. In addition, the experiments have provided a wealth of indirect results and observations. They have been published in publications P2 and P5. Publication P4 presents, partly based on the first experiment, definitions, categorizations, and a larger framework for mobile 3D navigation interface development.

6.1 Navigation

Navigation is not a single, monolithic task. It is a complex process, consisting of several factors from way-finding and other mental processes to physical movement. In order to create a good navigation interface, to serve the *purpose* of the map [57], the interface should support and reflect the related processes.

Darken and **Peterson** define *navigation* as the aggregate task of *way-finding* and *motion* [56]. Way-finding is the cognitive element of navigation, involving planning and observations, building up a *cognitive map*, an internal spatial model of the environment. Motion is the motoric element, the act of getting somewhere, namely travel. They describe *maneuvering* as a subset of motion, consisting of "smaller movements that may not nec-

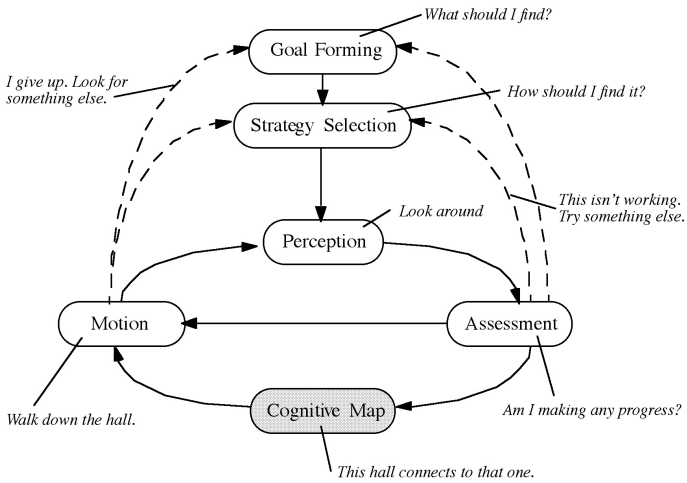

Figure 6.1: A general framework for the navigation process (reproduced from [56], an adaptation from [78]).

essarily be a part of getting from "here" to "there" but rather adjusting the orientation of perspective, as in rotating the body or sidestepping". We also define *micro-maneuvering*, which takes place in the virtual environment if a single supporting maneuver, such as a view orientation, consists of a series of actions.

Downs and **Stea** divide a navigation task to four main stages, 1) *initial orientation*, 2) *maneuvering*, 3) *maintaining orientation* and 4) *recognizing the target* [57]. Navigation tasks that take place in virtual environments are further categorized by **Darken** and **Cevik** [55]. 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*.

Figure 6.1 presents a model of the navigation process by **Jul** and **Furnas** [78], as modified by Darken and Peterson [56]. The process starts by *goal forming*, continues with *strategy selection*, and continues in the loop of *perception* (cue extraction), *assessment* of progress, and *motion*. This loop involves way-finding and develops the cognitive map as the navigator moves within the environment, observing it. Publication P4 presents a variation of the presented navigation model involving two distinguished forms of actions, *pragmatic* and *epistemic* [79]. While this model was utilized in the navigation interface development, it is a sole contribution by **Antti Oulasvirta**.

Navigation processes involve the use of spatial knowledge in problem solving. The classic hierarchical model builds on the concepts of *landmark knowledge*, *route knowledge* and *survey knowledge* [144]. The importance of landmarks and other structural features of urban environments was established in the seminal work by **Lynch** [94]. Use of a secondary source, such as a map, helps navigators to directly observe spatial relations and acquire

survey knowledge [154]. However, map usage is a situated action, interactive and dynamic in nature, where the navigator, the current task and the environment constitute an integral whole [148]. The task dependency also poses the *scaling problem*, well known in both paper and electronic maps [63, 56]. For example, 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 [115].

6.2 Controlling navigation: maneuvering

Publication P4 proceeds to discuss the innate problems related to controls of mobile devices in the context of 3D maneuvering. As a general framework for reducing the degrees of freedom, **Hanson** and **Wernert** have modeled this as a mapping from control space G to navigation space N, which provides movement on a *guide manifold*, a constrained subspace of the navigation space [68]. Commonly, the higher level interaction schemes are described with metaphors. Research on virtual environments has provided several metaphors that support maneuvering for specific tasks, general movement, or assistance [147]. Publication P4 discusses the various approaches to aid navigation, including *teleportation*, *click-and-fly*, various steering techniques, automatic adjustment of camera angle to maximize *orientation value* [34, 80], *interest fields* [68], and *attentive camera* [71]. Some solutions are not considered viable. For example, teleportation, which provides instantaneous travel to a target, is known to cause disorientation [27].

The numerous navigation interaction schemes involve varying amounts of user control. Publication P4 categorizes them to a set of *maneuvering classes* depending on the level of freedom of control (see Table 6.1).

Maneuvering class	Freedom of control
Explicit	The user controls motion with a mapping depending on the current navigation metaphor.
Assisted	The navigation system provides automatic supporting movement and orientation triggered by features of the environment, current navigation mode, and context.
Constrained	The navigation space N is restricted and cannot span the entire 3D space of the virtual environment.
Scripted	Animated view transition is triggered by user interaction, depending on environment, current navigation mode, and context.
Automatic	Movement is driven by external inputs, such as a GPS device or electronic compass.

Table 6.1: Maneuvering classes in decreasing order of navigation freedom.

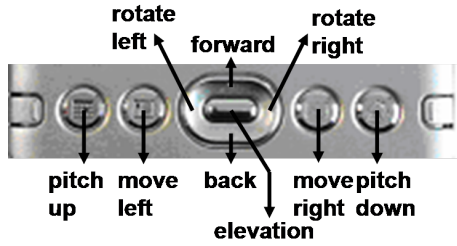

Figure 6.2: A direct mapping from a PDA's buttons to movement. While possibly cumbersome, this UI did not restrict navigation in 3D space, allowing us to gather valid maneuvering data from field experiments.

6.3 A 3D navigation field experiment with explicit controls

Our first formal field experiment consisted of navigation and pointing tasks using our 3D model and 3D engine. Navigation tasks involved movement in the environment, with the initial view at the current physical location. Pointing tasks involved only orientation in the physical environment, but maneuvering in the 3D world. Pointing tasks were initiated both from the 3D view and the real world. Our 3D map was installed on a **Dell** *Axim X30* PDA with 624Mhz XScale processor and 64MB RAM. The mapping from the PDAs joypad and buttons to movement was explicit. Figure 6.2 presents the simple controls that allowed subjects to move forward and back, turn left and right, and elevate/descend. They could also look up or down, or move sideways, resulting in four available degrees of freedom.

Figure 6.3 presents a navigation task (from publication P4). A test subject physically located at `(A)` maneuvers a 3D map view from `(A)` to `(B)` (Figure 6.3A), to spot a landmark known to her in advance (a major church). Figure 6.3B presents the initial 3D view. In the beginning of this task, the subject orients to the environment, spotting a recognizable façade (Figure 6.3C), orienting the 3D view towards the same façade (6.3D). The subject then determines a direction of movement and starts to maneuver along a road. After two blocks, the subject finds the target landmark and stops. Figure 6.3E presents the path and points of orientation resulting from performing the task. Figure 6.3F presents the related joypad state as a function of time. To perform this simple task, the subject performed a total of 20 rotations and 10 forward movements, using only two degrees of freedom from the available four. The subject did not move backward. The task took 135 seconds to complete, and the controls were used in two sequences: 5 seconds during initial orientation and 29 seconds during maneuvering towards the target. The remainder of the time, the subject observed the environment and waited for a car and a tram to pass (the tram remained in view for almost a minute). All actions on controls were performed sequentially.

In Figure 6.4 (upper, navigation path published in P4, Fig. 10.4A) a subject is asked to navigate from `(A)` to `(B)`. This is the subject's first exposure to the 3D model. Instead of performing the task, the subject chooses to *explore* the virtual environment.

In Figure 6.4 (lower left, published in P2 and P5), the subject is phys-

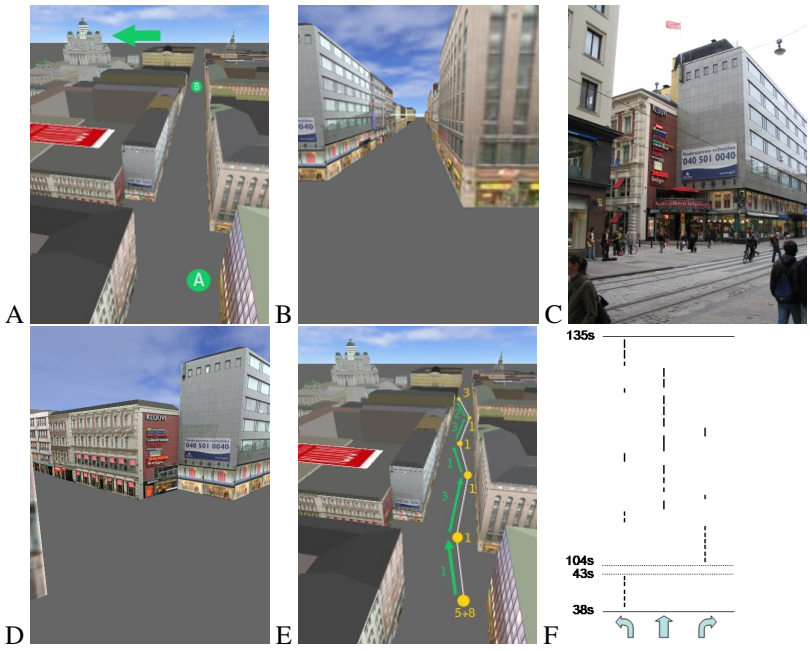

Figure 6.3: A navigation experiment with maneuvering at street level.

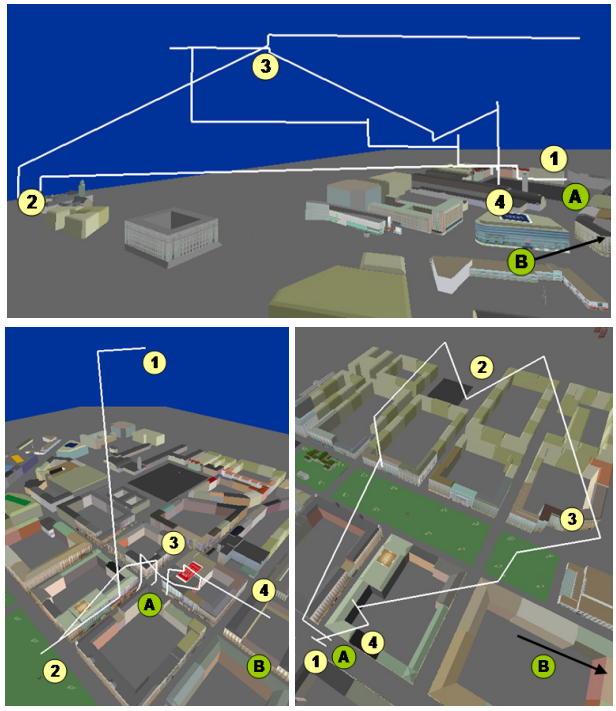

Figure 6.4: Navigation experiments. Exploration (up). Initial disorientation, local orientation and navigation towards target (lower left). Elevating the view point to spot a distant target (lower right).

ically located at (A) and asked to maneuver to (B). Initial view is at (1), in 90°angle to the (A)-(B) route. The subject first descends and moves forward to (2), just 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 figure 6.4 (lower right, published in P2), a subject at (A) is asked to find a landmark (B) in the 3D view. (B) lies off the screen with a 180° 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).

6.4 Improving navigation interaction

In our first experiment, we performed a total of 26 successful way-finding experiments, and 50 orientation experiments. Navigation with direct controls was considered cumbersome by most subjects, which was expected. However, the experiments provided valuable data. We observed the navigation tasks defined by Downs and Stea, navigation types by Darken and Cevig (except targeted search as we lacked target markers), and the need to scale the view depending on the current task – to lower the view for local orientation, or elevate it to spot a target further away. Subjects frequently put effort in finding a suitable view to match their task. This often involved *micro-maneuvering* – unnecessary and repeated adjustments to counter the poor affordability of the direct controls, observable in Figure 6.3. Nor the start position or the target were marked during navigation experiments. Users found it particularly difficult to remember the target's exact position, increasing cognitive load.

Publication P4 states goals for mobile navigation controls in pursue of *transparency*, where a user would be embedded in performing a task to such extent that he would be unaware of the actual interface [111]:

- minimize cognitive load (as defined by working memory load, amount or duration of cognitive task processing, or complexity of mental computations)
- minimize motor effort and procedural complexity
- minimize use of time

Furthermore, publication P4 asserts goals for the 3D view for navigation support:

- maximize information that helps orientation
- maximize information that helps performing the current task
- minimize information that leads to disorientation
- maximize information that helps forming an accurate cognitive map

Fulfilling these objectives ensures that the user

- does not get lost
- is able to find and visit all places of interest
- is able to re-visit places
- feels familiar with the space

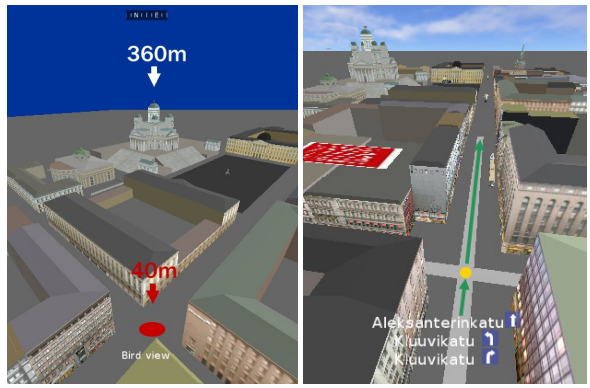

Figure 6.5: Navigation features. (left) *Landmark View* turns toward the nearest landmark, presenting it and the user's current position with *Markers*. (right) *Tracks* restrict navigation to the street network, reducing micromaneuvering.

The challenge in UI development has now become a question of finding a suitable combination of metaphors, assisting functionalities and navigation constraints that match and support the navigation tasks, suited for the limited controls. Based on our data from the field experiments, and further ideas from literature, we created an improved version of the interface. Table 6.2 provides a short list of navigation aids that have been implemented. Figure 6.5 presents three of the improvements, namely *Landmark View*, *Tracks* and *Markers*. The *Landmark View* animates the viewpoint smoothly to an overview position, showing both the current GPS position and a landmark, pointed by markers and provided by distances from the view position. If GPS is not available, the last view position is shown. *Tracks* limits navigation to the street network. However, as presented in Figure 6.6, if the view is rotated when on the tracks, it is simultaneously translated farther to provide a better view on the opposing façade. Similarly, the *Tracks* view on Figure 6.5 demonstrates the view angle resulting from automatic tilting as the viewpoint is moved up from the street level. At street level, the view is tilted slightly upward, to reveal features of the façades, as shown on Figure

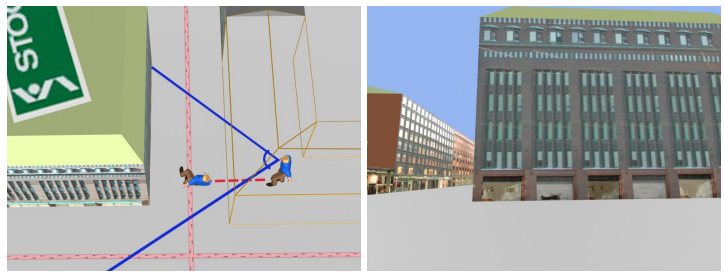

Figure 6.6: A navigation assisting scheme. When on *Tracks*, rotating the view automatically adjusts the view's distance from the opposing façade to provide a better view.

Interaction aid	Purpose and functionality
Tracks	Minimize micro-maneuvering. Restrict navigation space to a street network. Provide street names.
Orientation value	Maximize view's orientation value. Orient downward when elevating, and upward when descending; when in *tracks mode* at street level, translate away from the opposing façade for a better view.
Speed adjustment	Match motion with user's needs. Adjust speed automatically and smoothly based on elevation, being slower at street level and faster at sky.
View landmark	Orientation aid. Trigger an animated view transition to present a landmark and the user's current position
Change viewpoint	Minimize micro-maneuvering. Scripted view level transition to three predetermined view levels (street level, rooftop level, sky) with automatic orientation.
Markers	Decrease cognitive load, enable targeted search instead of primed search. Marker arrows point at, for example, the start point and the target, releasing users from remembering the exact 3D positions.
Fly-to-target	A scripted action to fly to a target. If fast transition is needed, a smooth but fast fly is less disorienting than teleportation, and demands less from the user than manual maneuvering.
Orbit mode	Aid for recognizing a target. View is locked towards a point, and controls are mapped to a cylindrical coordinate system, orbiting around the point.

Table 6.2: Aids for 3D navigation.

6.6 (right). Users are allowed to override the automatic tilting with explicit controls.

Publication P4 describes other potential features as well, such as GPS driven maneuvering, assisted camera schemes, and routing. Most of these have been implemented, such as the GPS functionalities and routing with visualization aids and automatic route walkthroughs.

6.5 A 3D navigation field experiment with improved controls

A second focused field experiment was conducted with the improved interface, using the functionalities listed in Table 6.2. The *Fly-to-target* always flew to the predetermined target of the current experiment task. Other functionalities, such as GPS positioning and routing, were left out due to two reasons. First, GPS positioning was observed to be inaccurate and error prone. Second, our goal was to observe maneuvering and navigation, not guided route following. The device in the second experiment was a 3D hardware-accelerated smart phone, the **Nokia** *N93*. The results and details of the experiment are available at [114], and provides insight to the different bodily conduct, use of cues and self-localization in the presence of a professionally drawn 2D map and our 3D map. In our context, we draw attention to the secondary, qualitative results, also available in publication P5.

The new functionalities were used often, especially *Change Viewpoint*

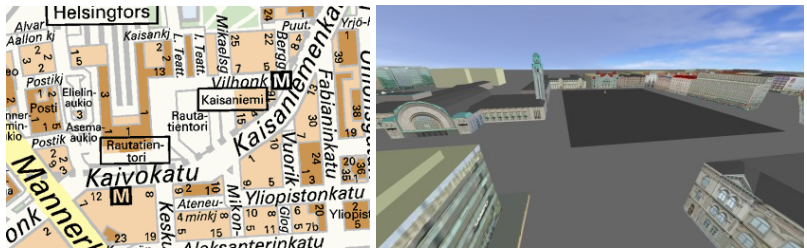

Figure 6.7: A 2D street map[1] and a 3D map of the same area.

(elevate/descend with automatically adjusted tilt). In addition, *Tracks* and *View Landmark* were used quite often. *Orbiting* was tried out, but not found useful. Once subjects learned a maneuvering routine, they didn't explore alternatives. Although the subjects were different than in the first test, in general we noticed increased performance and reduced cognitive load.

In the two experiments, comparisons were made against 2D maps. In the first experiment, the 2D map was simulated by a top-down view without street names. That did not yield as good a performance as the 3D map. In the second experiment, the 2D map was a professionally drawn raster map, with clearly marked street names (the map was not in scale due to the emphasized street names). Figure 6.7 presents the two maps. The street names on the 2D map proved to be a superior cue in comparison to the cues provided by the 3D map – statues, parks, recognizable buildings or building features and rooftop logos of companies. However, as a secondary strategy, some subjects attempted to use street numbers as additional cues. This proved to be a an inferior strategy due to poor correspondence between the map and the physical world – they could not be matched well. When subjects used the *Tracks* mode, their performance improved, although not to the level of the street map. The results and the feedback from subjects indicated that positioning functionality would be more important in the 3D maps than in the 2D maps.

6.6 Validation and veridicality

Chapters 4 and 5 have provided technical benchmarks on the presented system. However, answers to the other issues raised in earlier chapters can be found from the two field experiments. Publication P5 collects the most important ones together. We reprise them here, along with general observations arising from this work.

What is sufficient rendering speed?

In Section 1.1, we defined a rendering system to be interactive if it reached 6fps, and assumed that over 20fps would not provide a significant improvement. Our goal was therefore to reach 6fps at minimum, and attempt to reach or exceed 20fps. In our first experiment with the **Dell** *X30*, the update

[1] 2D map courtesy of City Survey Division of the Real Estate Department of Helsinki.

rate varied between 8–16fps[2]. Users reported this sluggish, but sufficient for navigation. In the second experiment with **Nokia** *N93*, over 20fps was reached in all situations. Users reported this smooth. Without an explicit and controlled rendering speed experiment, we cannot provide a quantitatively supported estimate for the minimum rendering speed; we can only provide an estimate. We believe that less than 10fps would yield too unpleasant an user experience for a 3D map to be widely acceptable, while 20fps or better will be pleasant.

What is sufficient rendering distance?
One of our initial requirements was to be able to render at least 100 buildings. In navigation use, the ability to observe the target on the map is critical for route planning. The farther one can see, the farther one can plan. Similarly, large landmarks, which aid orientation, may be visible and identifiable from several kilometers by the human eye. However, in the small display of the mobile device, the perspective transformation diminishes these distant features. We do not have quantitative data on how distant objects are perceived on mobile displays with a given field-of-view. Our field experiments have been performed with a 500m view radius, which was sufficient for the experiments, including over 100 potentially visible buildings, although the distant buildings were very small. At that distance, individual buildings become difficult to recognize, but the structure of the environment can still be perceived. In an overall view, a roughly $1km^2$ area can be viewed. To support orientation with landmarks, in our implementation we render emphasized landmarks even from several kilometer's distance. In general, with our current knowledge, we would not suggest limiting view distance to less than 500m.

Is it necessary to build special lightweight 3D models for mobile devices?
Traditional adaptation of large maps to a smaller medium has involved manual culling and selection processes, where the entire data set is scaled down. If 3D models are created with good design (including coherent normals, shared vertices, appropriate hierarchies and levels of detail), this can be automatized to a level where the selection and culling can be performed at run time, for any given view, at interactive update rates, even in resource limited mobile devices. Our case model was simple in geometry, but more complex models would be applicable with appropriate level-of-detail management. The critical issue is not mobility, but quality of the models, which becomes evident when attempting optimizations for rendering.

Given the small screen size of mobile devices, do we need detail?
A mobile, navigable 3D map is an interactive application. Even though the small screen size definitely accentuates the *keyhole property* of the view, users can maneuver closer to a cue in order to recognize it, if it is available at sufficient resolution. Figure 6.8 (from publication P5) presents a situation, where a subject attempts to verify a navigation target (the edge between

[2]During the first experiment, our cache system was not finalized and file I/O affected the rendering rates.

Figure 6.8: When "realism" fails. Local salient cues, the two company logos, are hard to recognize from a 3D model due to lack of detail.

two buildings) by two company logos, which were salient to him in the physical world, but hardly recognizable in the 3D view on the smart phone's display. In this case, our supposedly realistic model fails to meet the quality requirements – our 10–20cm texel size is insufficient. This example also implies the importance of cues.

Can the view in 3D be limited to street level?

3D rendering can be greatly accelerated, if viewpoint is forced to the street level, and only the visible façades are rendered. However, navigation frequently involves changes in view elevation, from local cue extraction near street level to sky views for longer distance way-finding. A limitation to street level would severely undermine a 3D map's potential in navigation.

Veridicality and realism

During our development, we have made several assumptions related to model veridicality (recognizability) and engine properties. We combine the observations from publication P5 here.

- *Recognizability*. Most buildings, and all landmarks, including statues represented by view-independent billboards, were recognized easily.
- *Street level contents.* Our façade textures could not replicate the street level due 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.
- *Façade inaccuracies*. Our initial model lacked a few façade textures, and some had been copied from adjacent, similar façades. 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 areas and added trees. However, our subjects were not generally seeking cues from the ground. A couple of subjects requested crosswalks.
- *Texel accuracy*. Mostly sufficient, but in local orientation, where no landmarks were visible, the 10–20cm texel accuracy was sometimes insufficient (see figure 6.8).
- *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 only colored, not textured. The roofs were not very visible from the physical viewpoint, but quite visible in the 3D view when subjects raised their viewpoint. This was found irritating to a few subjects. Roofs were not considered important for navigation, but the labels we had placed on top of a few selected landmark buildings were used often.
- *Topography*. While we built the initial 3D model, we had no topographic data available, so the resulting model was flat, despite our engine's support for topography. During our first trial, subjects did not even notice that topography is missing, so we left it be. However, on the second experiment, we placed one navigation task on elevated terrain, featuring stairs, where subjects reported difficulties in matching the 3D view with reality. In future, we would recommend including topography to 3D city models.
- *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 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.
- *Inherited visibility*. The location-based information culling mechanism presented in Figure 5.5 was not experimented on the field, but initial feedback from potential users was encouraging.
- *Free comments*. A few subjects reported that inanimate window reflections were distracting. Long view range was considered essential for orientation with major landmarks.

6.7 Open issues

We have incrementally improved our 3D navigation interface, but it is not yet finalized, nor perfect. However, our cyclic approach of developing and experimenting has kept us on the right track. Improving the interface will go hand in hand with a better understanding of small scale cognitive phenomena related to navigation tasks. Essential to this would be improved registration of user behavior and state. To this end, an accurate mobile eye tracking system could provide another leap in navigation research. As an example, an experiment for resolving the situation of being lost would benefit

greatly from such technology.

In addition to the observations resulting from our field experiments, there are several further factors that affect the usefulness of mobile 3D maps. For example, lighting conditions are obviously important. In direct sunlight, the screens of mobile devices are nearly impossible to view due to lack of contrast, unless the devices are equipped with transreflective displays. Although a developer cannot affect environmental conditions, this could be taken into account during the modeling phase, creating more contrast rich models, with emphasis on cues. The screen size itself is a problem, as very small details cannot be recognized without closer observation, cutting down the informativeness of any given view. This cannot be much improved by manipulating the field of view (FOV), as the FOV is always a compromise. A wide FOV provides a better view on the vicinity, while a narrower FOV yields more details from far away features. Allowing a user to manipulate the FOV would involve registering further controls. From our current data, we cannot quantify these issues, nor produce conclusive guidelines. However, these questions can be answered by further field experiments, which our platform has made possible.

The subjects in our field experiments were all able to read traditional maps, and were relatively familiar with the test environment. A short introductory teaching session for using a 3D map might not compensate for years of 2D map usage. We currently have no data to compare the learning curves of different map representations. Indeed, the *intuitivity* of a representation might be better described by the time required to understand it, for unfamiliar persons. Such experiment is yet to be performed. With our current experience, we hypothesize that 2D maps would require more conceptual learning, while 3D maps would require more practice on interaction.

Recent 3D hardware enabled mobile devices such as **Apple**'s *iPhone* and **Nokia**'s *N900* are equipped with touch screens. These devices already host accelometers, and affordable magnetometers are on the market. The key issue with 3D maps, interaction, might benefit greatly from these technologies, and will be the focus in our next generation 3D map interaction design.

Our system has been designed for and only tested with pedestrians. Other cases, such as car navigation, would require different design choices, and are open for future research.

7 CONCLUSIONS AND FUTURE

This thesis has addressed the challenges of a potential next generation interactive navigation system, a mobile 3D map that *presents together even the most minute features, attending to likeness* of the environment. This work has focused on four fundamental research goals. The methodology has been constructive. We can now assess the status of the questions.

1. *Can realistic urban 3D environments be rendered in current mobile devices at interactive rates?* Yes. It is possible to render large, detailed 3D cities in mobile devices at interactive rates (over 5fps) even without hardware acceleration. With 3D hardware, rendering rates exceed 30fps.

2. *Can realistic 3D city maps be downloaded on the fly to mobile devices with sufficient speed for common navigation tasks, without hindering interactive use?* Yes. Current cellular networks (40kB/s) are sufficient in supporting progressive transmission of detailed 3D models during navigation, given an efficient networking scheme. Interactivity is not hindered, given an efficient caching scheme.

3. *Can a mobile 3D city map system support dynamic content transmission and representation in near real time in scalable manner?* Yes. Urban environments and entity behaviors can be exploited to limit network traffic to be linear with the number of visible dynamic entities and location-based information, serving mobile clients in near real time without network congestion, given reasonable entity densities. Scalability is achieved by spatially limited entity servers.

4. *Is a realistic mobile 3D map interface inherently intuitive?* No. A realistic 3D representation can facilitate a recognition of the vicinity with a wealth of cues. However, the real question for intuitive use lies in the appropriate support of interaction methods and representation for the navigation task at hand. A 3D map can fully support all navigation tasks, but easy maneuvering in the virtual environment is challenging. Dependency on other technologies, such as positioning, appears to be more critical for 3D maps than traditional maps. Even with an appropriate interface, the 3D representation may not be the best one for all navigation tasks: self-locationing with a map that directs the focus on diagnostic cues such as street names appears to be more efficient, as long as these cues are available. As a corollary, this may hint towards the highest potential of detailed 3D city maps: when diagnostic cues such as street names are *not* available, matching becomes difficult. In this case, detailed 3D maps may prove their value. Also, in certain types of tasks, such as pinpointing a target, the ability for accurate representation may become invaluable.

We have developed a system for shared virtual environments, suited for mobile devices, where the representation is based on real-world structures.

The system, implemented with C/C++ and OpenGL ES 1.0 Common Profile, runs on mobile phones in a small amount of memory. Without 3D hardware acceleration, interactive rates are achieved for large, detailed models, and in the presence of 3D hardware, rendering rates become very satisfactory. In essence, we have successfully transformed the traditional and lossy map data "pre-process", where the entire data set is manually adapted to available medium, to a non-lossy process, which arranges the data to such structures that on-the-fly selection is fast and feasible. This selection process is even incorporated to cover dynamic entities in a scalable manner. We call this *adequacy mapping*, where a given data set is transformed to a representation that is most adequate for a given purpose. With our platform, we have been able to perform field experiments that have provided valuable data for understanding and improving a 3D navigation interface.

But will such optimizations be necessary in future? Isn't the next generation of mobile platforms and networks going to solve all technical issues?In 3D gaming industry, demands have always exceeded resources. Subsequently, the industry is not relying on hardware solutions only. For example, although modern 3D cards can hold over 1GB of texture memory, **Infinity Ward**'s 2009 title, *Modern Warfare 2*, applies 'texture streaming', similar to our LOD management, for detail.Simultaneously, good solutions, such as **id Software**'s optimization algorithms for *Quake* (1996), imposing strict requirements for 3D models, live long and prosper, and are used for example in **Valve Software**'s 2009 title *Left for Dead 2*. With 3D city models, we anticipate increase in size and detail over our small case model, be the source automatic laser scanning or community efforts. Consequently, we believe that optimized and task-suited 3D engines such as ours will continue to provide a better solution for 3D maps than generic viewers. In mobile use, these optimizations also manifest themselves as longer battery life.

As usual, answers rise new questions. We assumed to have reached *realism* by accurate textures. Now, witnessing the occasional failure of our accuracy, the follow-up question is, "what is sufficiently accurate"? Currently, there are no technical or perceptual tools for determining the "realism", or sufficiency, of a 3D model. In future, we will address this issue, experimenting with the recognizability of the environment.

Considering the *purpose* of a 3D map, or any map, there is still room for speculation of the ultimate representation. *Veridicality* concerns all maps; unless cues in a 2D map match the real world, users become disoriented. A "realistic" 3D map can provide redundant, multiple cues, which are not necessarily as diagnostic as traditional cues, but the redundancy provides potential for spotting at least something that can be matched. For navigation use, can we then apply traditional cartographic practices of simplification by emphasizing potentially salient cues? Can we emphasize 3D landmarks by a larger size, as the Romans did already in their *Forma Urbis Romae*, and which was done with our 2D raster street map (Fig. 6.7)? If we purposefully break the scale of the map, what happens to integration of legacy 2D-location-based information data sets, which, even with accurate global coordinates, would no longer match with the model? Furthermore, city engineers require exactness of positioning and geometry. These and other issues call for future research and collaboration between 3D GIS and computer graphics scientists, application developers and end users.

8 SUMMARY OF PUBLICATIONS AND CONTRIBUTIONS OF THE AUTHOR

Publication [P1]

This publication attacks the problem of rendering realistic 3D maps in mobile devices using computer graphics methods. The main obstacles are solved with efficient visibility culling methods, applying preprocessed potentially visible sets, a novel visibility list management system, explicit memory and level-of-detail management and caching. Realism is pursued with relatively detailed textures. In contrast to previous urban 3D rendering systems, the viewpoint is not limited to street level but can span the entire 3D space. For the lowest LOD texture, the publication presents a simple algorithm for selecting a dominant façade color instead of the average color. The publication discusses 3D model requirements that enable efficient culling processes, and presents an example system including a model and a preprocess stage. As a result, a large and detailed 3D city model is rendered at interactive and near interactive rates in PDAs and mobile phones for the first time. The design decisions that were made during development are validated based on initial results from a field experiment.

Publication [P2]

This publication develops a progressive 3D model download scheme and measures the effect of 3D hardware on mobile devices. Cellular network properties and results from a navigation field experiment are discussed. Based on the visibility and LOD based rendering scheme of P1, and observed real world navigation patterns, an efficient and lightweight progressive model transfer scheme is developed using a binary XML based protocol over TCP/IP. The networking scheme emphasizes progressive download and caching while maintaining interactivity. In contrast to earlier systems, pre-fetching is not considered essential or feasible, although possible for situations where all visible content has been downloaded. It is asserted that 3G networks are sufficiently fast for on-the-fly model download during navigation without hindering interactivity. The impact of 3D hardware on mobile devices is measured and discussed. It is noted that rasterization speed increases over an order of magnitude, although this alone does not solve the original problems of rendering 3D city maps; large models require an efficient out-of-core algorithm. This publication provides further details to the out-of-core algorithm that was shortly presented in P1.

Publication [P3]

In this publication, mobile networking capabilities are utilized to leverage a static 3D map to a dynamic, shared virtual environment where moving entities can be managed in a lightweight manner. The publication exploits the urban environment and presents a scalable solution for transmitting and exchanging states of dynamic entities, including position. The solution is based on precomputed visibility, providing a substantial improvement over existing dynamic entity visibility culling algorithms. Most computations are

transferred to clients based on their honestness, leaving a server to act as a fast message passing switchboard. A public transportation simulator with near real time vehicle tracking, based on the first version of the system, is described as an example of the potential of the system.

Publication [P4]

This publication provides a definition for 3D maps, and develops a 3D navigation interface for mobile devices. The discussion is based on literature reviews and a field experiment, stressing the necessity of user studies. The challenges are analyzed, and a maneuvering control categorization presented, based on the level of user interaction. An advanced 3D maneuvering user interface is designed, matched to the various navigation tasks. The publication has been co-published with Antti Oulasvirta. Sections 1.3.3 (Support for embodied interaction) and 1.4 (A model of interactive search on mobile maps) and references to pragmatic and epistemic action within other sections are contributions of A. Oulasvirta. The rest has been contributed by the author of this thesis.

Publication [P5]

This publication draws together the efforts of publications P1, P2, P3 and P4, presenting the entire scalable dynamic mobile 3D map system and user interface features. An inheritance-based visibility culling scheme for location-based information is presented, applied to the networking scheme. Results from two field experiments are presented, validating the various design decisions.

BIBLIOGRAPHY

[1] Tomi Aarnio. JSR-184: Mobile 3D graphics API for J2ME. http://www.jcp.org/en/jsr/detail?id=184, 2005.

[2] Tomi Aarnio. M3G API overview. ACM SIGGRAPH 2005 Course #35: Developing Mobile 3D Applications With OpenGL ES and M3G, August 2005.

[3] Tomi Aarnio. JSR-297: Mobile 3D graphics API 2.0 public review draft. http://www.jcp.org/en/jsr/detail?id=297, June 2008.

[4] James M. Abello and Jeffrey Scott Vitter, editors. *External memory algorithms*. American Mathematical Society, Boston, MA, USA, 1999.

[5] Gregory D. Abowd, Christopher G. Atkeson, Jason Hong, Sue Long, Rob Kooper, and Mike Pinkerton. Cyberguide: a mobile context-aware tour guide. *Wirel. Netw.*, 3(5):421–433, 1997.

[6] Michael Abrash. *Ramblings in Realtime*. Dr. Dobb's Sourcebook, 2000.

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

[8] Arttu Heinonen anb Simo Pulkkinen and Ismo Rakkolainen. An information database for vrml cities. In *Proceedings of the IEEE International Conference on Information Visualization*, pages 469–473. IEEE, 2000.

[9] Carlos Andújar, Carlos Saona-Vázquez, Isabel Navazo, and Pere Brunet. Integrating occlusion culling and levels of detail through hardly-visible sets. *Computer Graphics Forum*, 19(3), 2000.

[10] Fone Arena. Nokia 6630 - the first nokia 3G smartphone. http://www.fonearena.com/reviews/Nokia-6630-review.php, 2004.

[11] Masatoshi Arikawa, Shin'ichi Konomi, and Keisuke Ohnishi. Navitime: Supporting pedestrian navigation in the real world. *IEEE Pervasive Computing*, 6(3):21–29, 2007.

[12] Ulf Assarsson and Tomas Möller. Optimized view frustum culling algorithms. Technical report, Chalmers University of Technology, March 1999. http://www.cs.chalmers.se/staff/uffe.

[13] Mark Barnes and Ellen Levy Finch. Collada - digital asset schema release 1.5.0, April 2008.

[14] Jörg Baus, Keith Cheverst, and Christian Kray. A survey of map-based mobile guides. In Liqiu Meng and Alexander Zipf, editors, *Map-based mobile services - Theories, Methods and Implementations*, pages 197–213. Springer, 2005.

[15] Jörg Baus, Christian Kray, and Antonio Krüger. Visualization of route descriptions in a resource-adaptive navigation aid. *Cognitive Processing*, 2(2-3):323–345, 2001.

[16] Jörg Baus, Antonio Krüger, and Wolfgang Wahlster. A resource-adaptive mobile navigation system. In *IUI '02: Proceedings of the 7th international conference on Intelligent user interfaces*, pages 15–22, New York, NY, USA, 2002. ACM.

[17] J. Lennart Berggren and Alexander Jones. *Ptolemy's Geography, An Annotated Translation of the Theoretical Chapters*. Princeton University Press, 41 William Street, Princeton, New Jersey, USA, 2000.

[18] Fausto Bernardini, James T. Klosowski, and Jihad El-Sana. Directional discretized occluders for accelerated occlusion culling. *Computer Graphics Forum*, 19(3):507–516, 2000.

[19] Tim Berners-Lee, Roy Thomas Fielding, and Henrik Frystyk Nielsen. Hypertext transfer protocol – http/1.0. http://tools.ietf.org/html/rfc1945, May 1996.

[20] Brian Blau, Charles E. Hughes, Michael J. Moshell, and Curtis Lisle. Networked virtual environments. In *SI3D '92: Proceedings of the 1992 symposium on Interactive 3D graphics*, pages 157–160, New York, NY, USA, 1992. ACM Press.

[21] Heiko Bleschmied, Markus Etz, and Jörg Haist. 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, 2005.

[22] David Blythe. OpenGL ES common/common-lite profile specification, version 1.0.02. http://www.khronos.org/opengles, 2005.

[23] Shawn Bonham, Daniel Grossman, William Portnoy, and Kenneth Tam. Quake: An example multiuser network application - problems and solutions in distributed interactive simulations. Technical report, University of Washington, May 2000. CSE 561 Term Project Report.

[24] Gerard Bosch i Creus. 3d environments in wireless information devices. Master's thesis, Espoo-Vantaa Institute of Technology, Degree Programme in Computer Engineering, 2002.

[25] Azzedine Boukerche and Richard Werner Nelem Pazzi. Remote rendering and streaming of progressive panoramas for mobile devices. In *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia*, pages 691–694, New York, NY, USA, 2006. ACM Press.

[26] Christian Bouville, Isabelle Marchal, and Loïc Bouget. Efficient compression of visibility sets. In *International Symposium on Visual Computing (ISVC 2005)*. ISVC, Springer-Verlag, 2005.

[27] Doug A. Bowman, David Koller, and Larry F. Hodges. Travel in immersive virtual environments: An evaluation of viewpoint motion control techniques. In *Proceedings of VRAIS'97*, pages 45–52, 1997.

[28] Robert Braden. Requirements for internet hosts – communication layers. http://tools.ietf.org/html/rfc1122, 1989.

[29] Claus Brenner, Norbert Haala, and Dieter Fritsch. 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, 2001.

[30] Frederick P. Brooks, Jr. Walkthrough—a dynamic graphics system for simulating virtual buildings. In *SI3D '86: Proceedings of the 1986 workshop on Interactive 3D graphics*, pages 9–21, New York, NY, USA, 1987. ACM.

[31] Lloyd A. Brown. *The story of maps*. Little, Brown and Company, 1949.

[32] Don Brutzman and Leonard Daly. *X3D Extensible 3D Graphics for Web Authors*. Morgan Kaufmann Publishers, Elsevier Inc, 2007.

[33] Donald P. Brutzman, Michael Zyda, Kent Watsen, and Michael R. Macedonia. Virtual reality transfer protocol (vrtp) design rationale. In *WET-ICE '97: Proceedings of the 6th Workshop on Enabling Technologies on Infrastructure for Collaborative Enterprises*, pages 179–186, Washington, DC, USA, 1997. IEEE Computer Society.

[34] Henrik Buchholz, Johannes Bohnet, and Jurgen Dollner. Smart and physically-based navigation in 3D geovirtual environments. In *IV '05: Proceedings of the Ninth International Conference on Information Visualisation*, pages 629–635, Washington, DC, USA, 2005. IEEE Computer Society.

[35] Stefano Burigat and Luca Chittaro. Location-aware visualization of vrml models in gps-based mobile guides. In *Web3D '05: Proceedings of the tenth international conference on 3D Web technology*, pages 57–64, New York, NY, USA, 2005. ACM Press.

[36] Michael V. Capps and Seth J. Teller. Communication visibility in shared virtual worlds. In *WET-ICE '97: Proceedings of the 6th Workshop on Enabling Technologies on Infrastructure for Collaborative Enterprises*, pages 187–192, Washington, DC, USA, 1997. IEEE Computer Society.

[37] Edwin Earl Catmull. *A subdivision algorithm for computer display of curved surfaces.* PhD thesis, University of Utah, 1974.

[38] Keith Cheverst, Nigel Davies, Keith Mitchell, Adrian Friday, and Christos Efstratiou. Developing a context-aware electronic tourist guide: some issues and experiences. In *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 17–24, New York, NY, USA, 2000. ACM, ACM Press.

[39] Jatin Chhugani, Budirijanto Purnomo, Shankar Krishnan, Jonathan Cohen, Suresh Venkatasubramanian, David S. Johnson, and Subodh Kumar. vlod: High-fidelity walkthrough of large virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, 11(1):35–47, 2005.

[40] Yiorgos Chrysanthou and Mel Slater. Computing dynamic changes to bsp trees. *Computer Graphics Forum (Eurographics 1992)*, 11:321–332, 1992.

[41] Paolo Cignoni, C. Montani, and R.Scopigno. A comparison of mesh simplification algorithms. *Computers & Graphics*, 22(1):37–54, February 1998.

[42] James H. Clark. Hierarchical geometric models for visible surface algorithms. *Commun. ACM*, 19(10):547–554, 1976.

[43] Dell Co. Dell axim x30 owner's manual. http://support.euro.dell.com/support/edocs/systems/aximx30/en/index.htm, 2004.

[44] Hewlett-Packard Co. Quickspecs compaq ipaq pocket pc h3800 series. http://h18000.www1.hp.com/products/quickspecs/10977_na/10977_na.HTML, 2002.

[45] Daniel Cohen-Or, Yiorgos L. Chrysanthou, Cláudio T. Silva, and Durand Durand. A survey of visibility for walkthrough applications. *IEEE Transactions on Visualization and Computer Graphics*, 09(3):412–431, 2003.

[46] Daniel Cohen-Or, Gadi Fibich, Dan Halperin, and Eyal Zadicario. Conservative visibility and strong occlusion for viewspace partitioning of densely occluded scenes. *Computer Graphics Forum*, 17(3), 1998.

[47] Volker Coors. 3D-GIS in networking environments. In *International Workshop on 3D Cadastres*. FIG Commission 2, November 2001.

[48] Volker Coors. Dreidimensionale karten für location based services. In Alexander Zipf and J. Strobl, editors, *Geoinformation mobil*. Wichmann Verlag, October 2002.

[49] Hewlett-Packard Corporation. Hp jornada 548 pocket pc. HP Product Specifications (http://www.hp.com), 2000.

[50] Hewlett-Packard Corporation. Hp ipaq pocket pc h5400 series. HP Reference Guide (http://www.hp.com), April 2003.

[51] Hewlett-Packard Corporation. ipaq pocket pc h3900 series. HP Getting Started Guide (http://www.hp.com), February 2003.

[52] Superscape Corporation. Superscape technology – swerve client. http://www.superscape.com/gamedelivery/3dtechnology/swerveclient.php (last accessed Fall 2008), 2008.

[53] Wagner T. Corrêa, James T. Klosowski, and Cláudio T. Silva. Visibility-based prefetching for interactive out-of-core rendering. In *PVG '03: Proceedings of the 2003 IEEE Symposium on Parallel and Large-Data Visualization and Graphics*, pages 1–8, Washington, DC, USA, 2003. IEEE Computer Society.

[54] M. Cox and D. Ellsworth. Application-controlled demand paging for out-of-core visualization. *Visualization Conference, IEEE*, 0:235, 1997.

[55] Rudolph P. Darken and Helsin Cevik. Map usage in virtual environments: Orientation issues. In *Proceedings of IEEE Virtual Reality 99*, pages 133–240. IEEE, 1999.

[56] Rudolph P. Darken and Barry Peterson. Spatial orientation, wayfinding and representation. In K. M. Stanney, editor, *Handbook of Virtual Enviroment Technology*, chapter 28. Lawrence Erlbaum Associates, Inc., 2002.

[57] Roger M. Downs and David Stea. *Maps in Minds: Reflections on Cognitive Mapping*. Harper & Row, New York, 1977.

[58] David W. Eggert, Kevin W. Bowyer, and Charles R. Dyer. Aspect graphs: State-of-the-art and applications in digital photogrammetry. In *Proceedings of the 17th Congress of International Society for Photogrammetry and Remote Sensing (ISPRS)*, pages 633–645. ISPRS, 1992.

[59] Johannes Färber. Traffic modelling for fast action network games. *Multimedia Tools Appl.*, 23(1):31–46, 2004.

[60] Roy Thomas Fielding, Jim Gettys, Jeffrey C. Mogul, Henrik Frystyk Nielsen, Larry Masinter, Paul J. Leach, and Tim Berners-Lee. Hypertext transfer protocol – http/1.1. http://tools.ietf.org/html/rfc2616, June 1999.

[61] Henry Fuchs, Zvi Kedem, and Bruce Naylor. On visible surface generation by a priori tree structures. *Computer Graphics (Proc.SIGGRAPH'80)*, 14(3):124–133, 1980.

[62] Thomas A. Funkhouser, Carlo H. Séquin, and Seth J. Teller. Management of large amounts of data in interactive building walkthroughs. In *SI3D '92: Proceedings of the 1992 symposium on Interactive 3D graphics*, pages 11–20, New York, NY, USA, 1992. ACM Press.

[63] George W. Furnas. Generalized fisheye views. In *SIGCHI 1986*, pages 16–23. ACM SIGCHI, 1986.

[64] Craig Gotsman, Oded Sudarsky, and Jeffrey A. Fayman. Optimized occlusion culling using five-dimensional subdivision. *Computers and Graphics*, 23, 1999.

[65] Floraine Grabler, Maneesh Agrawala, Robert W. Sumner, and Mark Pauly. Automatic generation of tourist maps. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, pages 1–11, New York, NY, USA, 2008. ACM.

[66] Ned Greene, Michael Kass, and Gavin Miller. Hierarchical z-buffer visibility. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 231–238, New York, NY, USA, 1993. ACM Press.

[67] Khronos Group. Khronos member hybrid graphics delivers world's first OpenGL ES API software implementation. http://www.khronos.org/news/press/Releases/hybrid-feb-17-04.html, February 2004.

[68] Andrew J. Hanson and Eric A. Wernert. Constrained 3D navigation with 2D controllers. In *IEEE Visualization*, pages 175–182, 1997.

[69] Gerd Hesina and Dieter Schmalstieg. A network architecture for remote rendering. Technical Report TR-186-2-98-02, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, April 1998. human contact: technical-report@cg.tuwien.ac.at.

[70] Hugues Hoppe. Progressive meshes. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 99–108, New York, NY, USA, 1996. ACM.

[71] Stephen Hughes and Michael Lewis. Attentive camera navigation in virtual environments. In *IEEE International Conference on Systems, Man & Cybernetics*, pages 96–103, 2000.

[72] Sun Microsystems Inc. Java3D API specification. http://java.sun.com/javase/technologies/desktop/java3d/releases.html, 2002.

[73] The VRML Consortium Incorporated. The virtual reality modeling language, international standard iso/iec 14772-1:1997, 1997.

[74] Berg Insight. Mobile maps and navigation. Market Report, 2007.

[75] ISO/IEC. Iso/iec cd 19776-3 – X3D encodings – part 3: Binary encoding. http://www.web3d.org/x3d/specifications/x3d_specification.html, December 2004.

[76] Michael T. Jones. Google's geospatial organizing principle. *IEEE Computer Graphics and Applications*, 27(4):8–13, 2007.

[77] Json.org. Introducing json. http://json.org, 2006.

[78] Susanne Jul and George W. Furnas. Navigation in electronic worlds: A chi 97 workshop. *SIGCHI Bulletin*, 29(4), 2007.

[79] David Kirsh and Paul Maglio. On distinguishing epistemic from pragmatic action. *Cognitive Science*, 18:513–549, 1994.

[80] Szilárd Kiss and Anton Nijholt. Viewpoint adaptation during navigation based on stimuli from the virtual environment. In *Web3D '03: Proceedings of the eighth international conference on 3D Web technology*, pages 19–26, New York, NY, USA, 2003. ACM.

[81] James T. Klosowski and Cláudio T. Silva. The prioritized-layered projection algorithm for visible set estimation. *IEEE Transactions on Visualization and Computer Graphics*, 6(2):108–123, 2000.

[82] Thomas H. Kolbe, Gerhard Gröger, and Lutz Plümer. Citygml - interoperable access to 3D city models. In Oosterom, Zlatanova, and Fendel, editors, *First International Symposium on Geo-Information for Disaster Management GI4DM*. Springer Verlag, March 2005.

[83] David Koller, Jennifer Trimble, Tina Najbjerg, Natasha Gelfand, and Marc Levoy. Fragments of the City: Stanford's Digital Forma Urbis Romae Project. *Proceedings of the Third Williams Symposium on Classical Architecture, Journal of Roman Archaeology, Supplementary Series*, 61:237–252, 2006.

[84] Christian Kray, Christian Elting, Katri Laakso, and Volker Coors. Presenting route instructions on mobile devices. In *IUI03*, pages 209–224. ACM, 2003.

[85] Antonio Krüger, Andreas Butz, Christian Möller, Christoph Stahl, Rainer Wasinger, Karl-Ernst Steinberg, and Andreas Dirschl. The connected user interface: realizing a personal situated navigation service. In *IUI '04: Proceedings of the 9th international conference on Intelligent user interface*, pages 161–168, New York, NY, USA, 2004. ACM, ACM Press.

[86] Katri Laakso, Ove Gjesdal, and Jan Sulebak. Tourist information and navigation support by using 3D maps displayed on mobile devices. In *Mobile HCI 2003 Workshop on HCI in Mobile Guides*, 2003.

[87] Annalina Levi and Mario Levi. Itineraria picta: Contributo allo studio della tabula peutingeriana. *The Journal of Roman studies*, 60:242, 1967.

[88] Peter Lindstrom and Valerio Pascucci. Terrain simplification simplified: A general framework for view-dependent out-of-core visualization. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):239–254, 2002.

[89] Risto Linturi, Marja-Riitta Koivunen, and Jari Sulkanen. Helsinki arena 2000 - augmenting a real city to a virtual one. In *Digital Cities, Technologies, Experiences, and Future Perspectives*, pages 83–96, London, UK, 2000. Springer-Verlag.

[90] Allan Christian Long, Jr., Shankar Narayanaswamy, Andrew Burstein, Richard Han, Ken Lutz, Brian Richards, Samuel Sheng, Robert W. Brodersen, and Jan Rabaey. A prototype user interface for a mobile multimedia terminal. In *CHI '95: Conference companion on Human factors in computing systems*, pages 81–82, New York, NY, USA, 1995. ACM.

[91] Fujitsu Ltd. Pocket loox user's manual. http://www.pc-ap.fujitsu.com/uguide/pocketloox/looxeng.pdf, 2002.

[92] David Luebke and Chris Georges. Portals and mirrors: simple, fast evaluation of potentially visible sets. In *SI3D '95: Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 105–106. ACM, 1995.

[93] David Luebke, Benjamin Watson, Jonathan D. Cohen, Martin Reddy, and Amitabh Varshney. *Level of Detail for 3D Graphics*. Elsevier Science Inc., New York, NY, USA, 2002.

[94] Kevin Lynch. *The Image of the City*. Cambridge: M.I.T.Press, 1960.

[95] Jean-Eudes Marvie and Kadi Bouatouch. A VRML97-X3D extension for massive scenery management in virtual worlds. In *Web3D '04: Proceedings of the ninth international conference on 3D Web technology*, pages 145–153, New York, NY, USA, 2004. ACM Press.

[96] Stephanie Meece. A bird's eye view - of a leopard's spots. the Çatalhöyük 'map' and the development of cartographic representation in prehistory. *Anatolian Studies: Journal of the British Institute at Ankara*, 56:1–16, 2006.

[97] James Mellaart. Excavations of catal hyuk, 1963. *Anatolian Studies: Journal of the British Institute at Ankara*, 19:17–177, 1964.

[98] Liqiu Meng, Alexander Zipf, and Stephan Winter, editors. *Map-based Mobile Services*. Springer, 2008.

[99] Ville Miettinen. Building scalable 3D applications. ACM SIGGRAPH 2005 Course #35: Developing Mobile 3D Applications With OpenGL ES and M3G.

[100] Paul Milgram and Fumio Kishino. A taxonomy of mixed reality visual displays. *IEICE Transactions on Information Systems*, E77-D(12):29–33, December 1994.

[101] Tomas Möller and Eric Haines. *Real-time rendering*. A. K. Peters, Ltd., Natick, MA, USA, 1999.

[102] Steven Molnar. The pixelflow texture and image subsystem. *Computers & Graphics*, 20(4):491 – 502, 1996. Hardware Supported Texturing.

[103] Alessandro Mulloni, Daniele Nadalutti, and Luca Chittaro. Interactive walkthrough of large 3D models of buildings on mobile devices. In *Web3D '07: Proceedings of the twelfth international conference on 3D web technology*, pages 17–25, New York, NY, USA, 2007. ACM.

[104] Aaftab Munshi and Jon Leech. OpenGL ES common profile specification version 2.0.22 (full specification). http://www.khronos.org/opengles, April 2008.

[105] Boaz Nadler, Gadi Fibich, Shuly Lev-Yehudi, and Daniel Cohen-Or. A qualitative and quantitative visibility analysis in urban scenes. *Computers & Graphics*, 23(5):655 – 666, 1999.

[106] Bruce F. Naylor. Partitioning tree image representation and generation from 3D geometric models. In *Proceedings of Graphics Interface '92*, pages 201–212, 1992.

[107] Shaun Nirenstein, Edwin Blake, and James Gain. Exact from-region visibility culling. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*, pages 191–202, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

[108] Nokia. Nokia N93 Technical Specifications. http://www.forum.nokia.com/devices/N93, 2006.

[109] Nokia. Nokia N95 Technical Specifications. http://www.forum.nokia.com/devices/N95, 2008.

[110] Nokia-Tuning.net. Nokia mobile phones processors. http://www.nokia-tuning.net/index.php?s=processor, 2009.

[111] Donald Norman. *The Psychology of Everyday Things*. Basic Books, New York, 1988.

[112] Antti Nurminen. m-loma – a mobile portal to location-based information. In *Eurographics 2006 (short papers)*. Eurographics, 2006.

[113] Antti Nurminen. A mobile 3D archipelago. In *Proceedings of IASTED Computer Graphics and Imaging 2007*. IASTED, 2007.

[114] Antti Oulasvirta, Sara Estlander, and Antti Nurminen. Embodied interaction with a 3D versus 2D mobile map. *Personal and Ubiquitous Computing*, 2008. Accepted for publication in Special Issue on Mobile Spatial Interaction.

[115] Antti Oulasvirta, Annu-Maaria Nivala, Ville Tikka, Lassi Liikkanen, and Antti Nurminen. Understanding users' strategies with mobile maps. In *Mobile Maps 2005 - Interactivity and Usability of Map-based Mobile Services, a workshop*. Mobile HCI, 2005.

[116] Antti Oulasvirta, Antti Nurminen, and Annu-Maaria Nivala. Interacting with 3D and 2D mobile maps: An exploratory study. Technical Report 1, HIIT, 2007.

[117] ParallelGraphics. Pocket cortona. http://www.parallelgraphics.com/products/cortonace/notes, 2000.

[118] Yoav I. H. Parish and Pascal Müller. Procedural modeling of cities. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 301–308. ACM Press, 2001.

[119] Jong-Seung Park and Bum-Jong Lee. Hierarchical contribution culling for fast rendering of complex scenes. In L-W. Chang, W-N Lie, and R. Chiang, editors, *Advances in Image and Video Technology*, Lecture Notes in Computer Science, pages 1324–1333. Springer Berlin, 2006.

[120] Stefan Parkvall, Eva Englund, Peter Malm, Tomas Hedberg, Magnus Persson, and Janne Peisa. Wcdma evolved – high-speed packet-data services. Ericsson Review 2, Ericsson, 2003.

[121] Wouter Pasman and Frederick W. Jansen. Comparing simplification and image-based techniques for 3D client-server rendering systems. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):226–240, 2003.

[122] Harry Plantinga and Charles R. Dyer. Visibility, occlusion, and the aspect graph. *International Journal of Computer Vision*, 5(2):137–160, November 1990.

[123] Stefan Poslad, Heimo Laamanen, Rainer Malaka, Achim Nick, Phil Buckle, and Alexander Zipf. Crumpet: creation of user-friendly mobile services personalised for tourism. In *3D Mobile Communication Technologies*. IEEE, 2001.

[124] Günther Pospichil, Martina Umlauft, and Elke Michlmayr. Designing lol@, a mobile tourist guide for umts. In *Proceedings of Mobile HCI 2002*, pages 140–154. Mobile HCI, Springer-Verlag, 2002.

[125] Jon Postel. Dod standard internet protocol. http://tools.ietf.org/html/rfc760, January 1978.

[126] Jon Postel. Rfc 761 – transmission control protocol. http://tools.ietf.org/html/rfc761, January 1980.

[127] Jon Postel. Rfc 768 – user datagram protocol. http://tools.ietf.org/html/rfc768, August 1980.

[128] Jon Postel and Joyce Reynolds. Rfc 959 – file transfer protocol. http://tools.ietf.org/html/rfc959, October 1985.

[129] Claudius Ptolemy. *Geōgraphikē Hyphēgēsis*. The Great Library of Alexandria, ca. 150.

[130] Kari Pulli, Tomi Aarnio, Ville Miettinen, Kimmo Roimela, and Jani Vaarala. *Mobile 3D graphics: with OpenGL ES and M3G*. Morgan Kaufmann, 2008.

[131] Kari Pulli, Tomi Aarnio, Kimmo Roimela, and Jani Vaarala. Designing graphics programming interfaces for mobile devices. *Computer Graphics and Applications*, 25(6):66–75, 2005.

[132] Peter Quax, Bjorn Geuns, Tom Jehaes, Wim Lamotte, and Gert Vansichem. On the applicability of remote rendering of networked virtual environments on mobile devices. *ICSNC*, 0:16, 2006.

[133] I. Rakkolainen, J. Timmerheid, and T. Vainio. A 3D city info for mobile users. *Computers and Graphics*, 25(4):619–625, 2001.

[134] Matti Rantanen, Antti Oulasvirta, Jan Blom, Sauli Tiitta, and Martti Mäntylä. Inforadar: group and public messaging in the mobile context. In *NordiCHI '04: Proceedings of the third Nordic conference on Human-computer interaction*, pages 131–140, New York, NY, USA, 2004. ACM Press.

[135] Martin Reddy, Yvan Leclerc, Lee Iverson, and Nat Bletter. Terravision ii: Visualizing massive terrain databases in vrml. *IEEE Comput. Graph. Appl.*, 19(2):30–38, 1999.

[136] Michael R.Macedonia. *A Network Software Architecture for Large Scale Virtual Environments*. PhD thesis, Naval Postgraduate School, Monterey, California, June 1995.

[137] Benet Salway. Travel, *itineraria* and *tabellaria*. In Colin Adams and Ray Laurence, editors, *Travel & Geography in the Roman Empire*. Routledge (Taylor & Francis), 2001.

[138] Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *WMCSA '94: Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*, pages 85–90, Washington, DC, USA, 1994. IEEE Computer Society.

[139] Arne Schilling, Volker Coors, and Katri Laakso. Dynamic 3D maps for mobile tourism applications. In L. Meng, T. Reichenbacher, and A. Zipf, editors, *Map-based Mobile Services - Theories, Methods and Implementations*, Earth and Environmental Science, pages 227–239. Springer, 2005.

[140] Arne Schilling and Alexander Zipf. 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*, pages 39–ff, New York, NY, USA, 2003. ACM Press.

[141] Dieter Schmalstieg and Michael Gervautz. Demand-driven geometry transmission for distributed virtual environments. *Computer Graphics Forum*, 15(3):421–431, 1996.

[142] Noel D. Scott, Daniel M. Olsen, and Ethan W. Gannett. An overview of the visualize fx graphics accelerator hardware. HP Journal, May 1998.

[143] John Rennie Short. *The World Through Maps: A History of Cartography*. Firefly Books, P.O.Box 1338, Buffalo, New York 14205, USA, 2003.

[144] A.W. Siegel and S.H. White. The development of spatial representations of large-scale environments. In H.W. Reese, editor, *Advances in Child Development and Behaviour*, volume 10, pages 9–55. Academic Press, 1975.

[145] François Sillion, George Drettakis, and Benoit Bodelet. Efficient impostor manipulation for real-time visualization of urban scenery. *Computer Graphics Forum*, 16(3), 1997.

[146] Shamus P. Smith and Tim Marsh. Evaluating design guidelines for reducing user disorientation in a desktop virtual environment. *Virtual Reality*, 8(1):55–62, 2004.

[147] Rory Stuart. *The Design of Virtual Environments*. McGraw-Hill, 1996.

[148] Lucy Suchman. *Plans and Situated Actions: The Problem of Human-machine Communication*. Cambridge University Press, 1987.

[149] Oded Sudarsky and Craig Gotsman. Output-sensitive visibility algorithms for dynamic scenes with applications to virtual reality. *Computer Graphics Forum*, 15(3):249–258, 1996.

[150] Oded Sudarsky and Craig Gotsman. Output-sensitive rendering and communication in dynamic virtual environments. In *VRST '97: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 217–223, New York, NY, USA, 1997. ACM, ACM Press.

[151] Ivan E. Sutherland, Robert F. Sproull, and Robert A. Schumacker. A characterization of ten hidden-surface algorithms. *ACM Comput. Surv.*, 6(1):1–55, 1974.

[152] Seth J. Teller and Carlo H. Séquin. Visibility preprocessing for interactive walkthroughs. *SIGGRAPH Comput. Graph.*, 25(4):61–70, 1991.

[153] Seth Jared Teller. *Visibility Computations in Densely Occluded Polyhedral Environments*. PhD thesis, University of California at Berkeley, 1992.

[154] Perry W. Thorndyke and Barbara Hayes-Roth. Differences in spatial knowledge acquired from maps and navigation. *Cognitive Psychology*, 14:560–589, 1982.

[155] Martina Umlauft, Günther Pospischil, Georg Niklfeld, and Elke Michlmayr. Lol@, a mobile tourist guide for umts. *Information Technology and Tourism*, 5:151–164, 2003.

[156] Teija Vainio. Towards more personalized navigation in mobile three-dimensional virtual environments. In *Intelligent User Interfaces, a position statement on a workshop Beyond Personalization*. ACM, 2005.

[157] Teija Vainio and Outi Kotala. Developing 3D information systems for mobile users: some usability issues. In *NordiCHI '02: Proceedings of the second Nordic conference on Human-computer interaction*, pages 231–234, New York, NY, USA, 2002. ACM Press.

[158] Teija Vainio, Outi Kotala, Ismo Rakkolainen, and Hannu Kupila. Towards scalable user interfaces in 3D city information systems. In *Mobile HCI '02: Proceedings of the 4th International Symposium on Mobile Human-Computer Interaction*, pages 354–358. Mobile HCI, 2002.

[159] Michiel van de Panne and James Stewart. Effective compression techniques for precomputed visibility. In *Rendering Techniques '99 (Proceedings of the 10th EG Workshop on Rendering*, Springer Computer Science, pages 305–316. Eurographics, Eurographics Association, 1999.

[160] Flavius Renatus Vegetius. Epitoma rei militaris. In Michael D. Reeve, editor, *Epitoma Rei Militaris*. Clarendon Press, 2004.

[161] Emily Whiting, Jonathan Battat, and Seth Teller. Topology of urban environments. In *Proceedings of CAAD Futures'07*, 2007.

[162] Lance Williams. Pyramidal parametrics. In *SIGGRAPH '83: Proceedings of the 10th annual conference on Computer graphics and interactive techniques*, pages 1–11, New York, NY, USA, 1983. ACM Press.

[163] Peter Wonka and Dieter Schmalstieg. Occluder shadows for fast walkthroughs of urban environments. *Computer Graphics Forum*, 18(3), 1999.

[164] Peter Wonka, Michael Wimmer, and Dieter Schmalstieg. Visibility preprocessing with occluder fusion for urban walkthroughs. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*, pages 71–82, London, UK, 2000. Springer-Verlag.

[165] Peter Wonka, Michael Wimmer, François Sillion, and William Ribarsky. Instant architecture. *ACM Trans. Graph.*, 22(3):669–677, 2003.

[166] Tatu Ylönen and Chris Lonvick. The secure shell (ssh) protocol architecture. http://tools.ietf.org/html/rfc4251, January 2006.

[167] Hansong Zhang. *Effective occlusion culling for the interactive display of arbitrary models*. PhD thesis, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 1998.

[168] Siyka Zlatanova and Edward Verbree. Technological developments within 3D location-based services. In *Proceedings of the International Symposium and exhibition on Geoinformation*, pages 153–160, October 2003.

TKK DISSERTATIONS IN MEDIA TECHNOLOGY

TKK-ME-D-1 Sampo Vesa
Studies on Binaural and Monaural Signal Analysis – Methods and Applications

ISBN 978-952-248-192-4 (print)
ISBN 978-952-248-193-1 (online)
ISSN 1797-7096 (print)
ISSN 1797-710X (online)